Knowledge Representation for Computer Vision: The VEIL Project

Keith Price*
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273
price@usc.edu

Thomas Russ, Robert MacGregor
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
tar@isi.edu

Abstract

The VEIL project focuses on integrating advanced knowledge representation technology (provided by Loom) with image understanding technology to develop advanced tools for the generation of vision systems. This effort is aimed at exploring the weaknesses of current knowledge representation technology for computer vision tasks in the realm of spatial reasoning and addressing a weakness in computer vision technology in the realm of higher level representations. The result will improve shareability and reusability of code for computer vision systems.

1 Introduction

The VEIL (Vision Environment Integrated with Loom) project is a collaborative effort between ISI's Loom project (Knowledge Representation) [MacGregor & Burstein 1991] and the Computer Vision group. It is an experiment both to see how traditional knowledge representation technology can be applied to computer vision and to explore extensions to knowledge representation systems to directly aid computer vision research (especially in the area of spatial reasoning).

This effort investigates the benefits available to vision applications obtainable via the introduction of declarative programming techniques, specifically, techniques available using advanced symbolic processing technology found in a modern knowledge representation system. In typical vision applications today, a programmer invents specialized data structures and carefully crafts a suite of vision processing algorithms that exploit those data structures. The result is most often a highly specialized piece of code that cannot be reused for a different domain, or applied to applications other than the one originally intended. The Image Understanding Environment [Mundy *et al.* 1993] addresses some of these issues, including sharing and reuse of basic data structures and processing algorithms, but does not deal with higher level representation issues that are the focus of this work.

The VEIL project aims to develop a technology whereby much of the work that goes into the development of specialized vision processing modules results in software that can be shared or reused by multiple applications. Knowledge representation techniques have been a part of computer vision research from the beginning (for example see [Winston 1975, McKeown *et al.* 1985, Draper *et al* 1989]). One difference is that this project combines an existing powerful knowledge representation system with relatively mature computer vision programs and techniques. This project will form the basis for incorporating knowledge representation technology in future computer vision research.

In order to study the knowledge representation issues directly, we decided to transform an existing program for runway detection and analysis into one built using the Loom system and declarative programming techniques. This strategy has several advantages. First, we know that the algorithm works, and second, we can directly explore the benefits of using knowledge representation technology. This paper will discuss some of the issues of declarative programming, briefly describe the airport analysis system, and present results of the effort in incorporating knowledge representation in computer vision.

2 Declarative Programming

The key approach in VEIL is the application of declarative programming techniques to vision processing, leveraged by the reasoning capabilities of the Loom knowledge representation system. A declarative specification of an application (or even a portion of an application) provides a formal, semantically well-founded description that offers numerous benefits. Such a specification

- is more readable and easier to maintain than a procedurally-specified program;

* This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by Rome Laboratories under Contract No. F30602-93-C-0064. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

- is subject to automatic validation and verification techniques;
- represents a high-level language specification. Thus, it does not rely on a specific choice of data structures. Algorithms are specified by the heuristic rules they employ and/or the changes they effect rather than by how they operate;
- can be shared and reused by other applications.

3 Relation of VEIL to IUE

The Image Understanding Environment (IUE) represents a major step towards the introduction of sharable object oriented specifications into the vision domain. The intended domains of the IUE and VEIL have some overlap, but the IUE explicitly does not consider higher-level knowledge representation issues or reasoning techniques.

Loom provides a very expressive domain modelling language, an integrated suite of deductive reasoners (for performing general symbolic processing), and an environment for creating, editing, viewing, and saving knowledge base objects.

4 Airport Example

We developed a project to explore the use of standard knowledge representation techniques in computer vision. The goals of the project include improvements in both computer vision and knowledge representation techniques. To this end, we started from a relatively mature application and incrementally changed the program to replace procedural specifications of knowledge with declarative representations of knowledge.

Detection and analysis of aerial views of airports provide the first application for Loom. This application defines primitive concepts for such objects as runways, center stripes, blast pad markings, distance markings, and taxiways. [Huertas, *et al.* 1990]

Airports are described by a generic model: a collection of generic runways, which are long thin ribbons with markings (smaller ribbons) in specific locations. Our system locates potential runways through a sequence of filtering and grouping operations followed by a hypothesis verification step. Since these are described in detail in earlier work, we will give only a brief description of these techniques in this paper.

4.1 Runway Hypothesis Generation

The basic steps in finding runway hypotheses (which are also used for the taxiway hypothesis generation) are:

- Generate edges using e.g., the Canny edge detector [Canny 1986]. Find connected sequences of edge elements and form straight line segments from these curves [Nevatia & Babu 1980]. Two sets of edges and line segments are generated, one with a relatively large mask (size of 9) and high threshold (strength of 10) for runway hypotheses and the other with a smaller mask (size of 7) and lower thresholds (8) for markings. These result in 25,000-90,000 line segments for typical images.
- Group straight line segments into anti-parallel pairs (that indicate ribbons), called apars. These pairs are limited by width (one set for markings is narrow, about 1 to 12 pixels, and the other set for potential runways is much wider, around 20 to 60 pixels). These widths are based on very rough approximations of the image scale and the generic description of the possible runways (which have defined limits on widths) and markings (which have very specific widths). The program generates 18,000 to 35,000 apars for the images.
- Find dominant directions using a histogram of apar directions. The apar is weighted by its length in the histogram accumulation. The histogram should have a few very dominant peaks, which correspond to runway directions. The later processing is applied to selected apars for one direction at a time (except for taxiways) which greatly reduces the computation time. Similar histogram analysis on widths could be used to further restrict the valid runway widths, but is not needed. This reduces the set of apars from 18,000 to 1,000 to 3,000 for airports with runways in multiple directions (Boston). The reduction in numbers is similar for other examples, but much less pronounced for airports with all runways in the same direction (Los Angeles).
- Eliminate apars contained within larger ones. This noise-cleaning step reduces the number of elements to analyze. The extra apars, which are eliminated, have many causes, but most are caused by the markings (i.e. an apar formed by the two sides of the runway, and two more formed by the side and the center stripe). Figure 1 illustrates this operation. Typically, about one-third of the apars survive this filtering.
- Join apars that share a common line segment. These breaks in large apars are caused by a gap on one side. This operation maintains colinearity (since the line

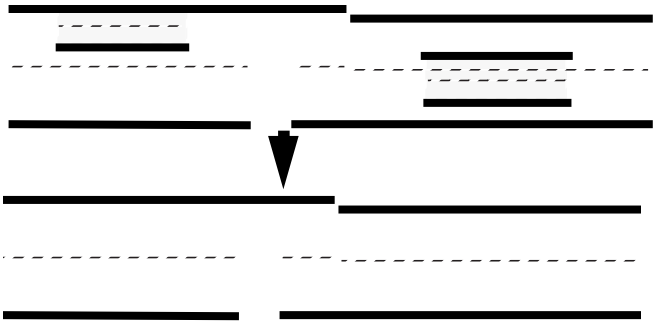

Figure 1. Eliminate the small Apar contained within the larger one.

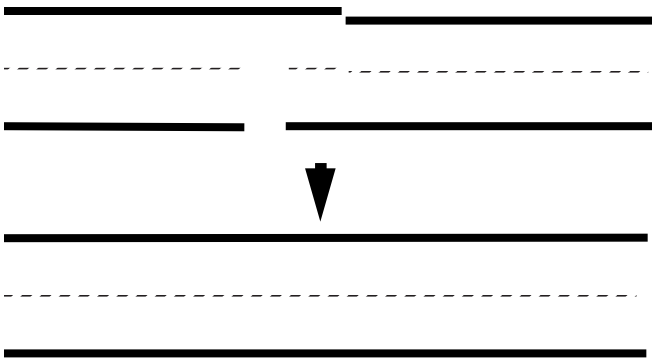

Figure 2. Merge Apars that contain a common segment.

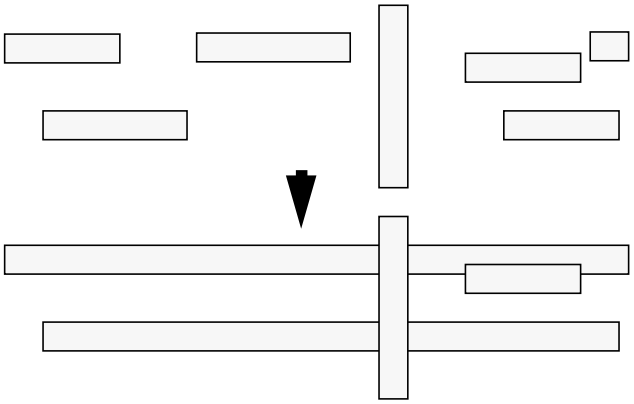

Figure 3. Merge colinear Apars across gaps.

segment is straight) and creates new merged fragments that were not in the original image data. After this step, reapply the previous step to eliminate contained apars. Figure 2 shows how this operation works, reducing the total number of apars by about 10% to 15%. At this point a filtering on aspect ratio is applied to remove very short hypotheses from further consideration (ratio of length to width less than 1). This removes about half of the remaining hypotheses (with about 150 to 250 remaining).

• Merge colinear apars across gaps. The gaps are formed by missing edge and apar data, by actual crossing runways or other occlusions. This step also creates new merged fragments. The gap must be analyzed to determine if the merger is valid (e.g. taxiways do not cross runways). This step has the potential for serious errors if the allowed gaps are too large (or too small) and if the definition of colinear allows the hypothesis direction to drift. Figure 3 shows this operation. This reduces the total number of fragments to roughly two-thirds of the previous number. A second aspect ratio filtering (greater than 10) is applied here to get the final hypotheses (for the Boston image 4 or 5 remain for each of the three directions).

These filtering operations depend only on a generic description of the runway and are all relatively efficient operations given the right data structures (especially

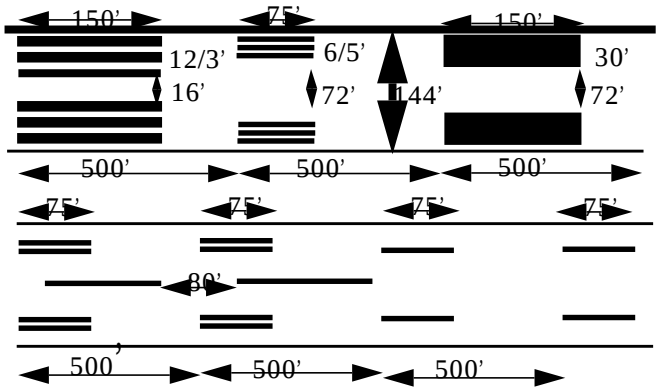

Figure 4. Markings on an instrument runway. From the top left these are referred to as, threshold mark, touchdown marks, big distance marks, and small distance marks (4 pairs).

spatial index). In the original reports on this effort, the run times were very large. Most of the reduction came from using data structures such as the spatial index to greatly reduce searches through the data.

4.2 Runway Hypothesis Verification

The verification step requires analyzing the hypotheses to find the specific markings. Figure 4 illustrates the markings for an instrument runway. The dimensions and spacings are given in feet. Each marking would appear in the image as an apar of a specific size (e.g. 30 feet wide and 150 feet long). Using an initial scale gives the size range for each marking apar an indication of its position relative to other markings and relative to the runway hypothesis.

First the true ends of the runways must be located. These are indicated by the threshold marks (top left of Figure 4). Rather than find the marks themselves, it is easier to find the apar in the center of the runway formed by the gap between the two marks. The threshold mark is located by searching along the center line of the runway hypothesis to find the relatively dark apar of this gap. Once the threshold mark is found the other distance marks are located relative to it. Each mark is located by looking for apars in the appropriate locations and selecting according to the description of the marking.

Center lines and side stripes are found by looking for marks in specific locations relative to the runway hypothesis (in the center, along either side). Side stripes are very narrow (roughly 1 pixel) so they tend to break up into many small pieces.

4.3 Refinement of Hypotheses

The initial markings are located using the large set of apars generated by the global edge detection process at the beginning. This is sufficient for finding the well-defined markings, but misses many of the marks. More markings are located by reapplying the edge, line and apar finding procedures on small windows of the image (and using a replicated version of the image so that

```
(tell (create big-distance generic-mark)
 (about big-distance
  (width-in-feet 30)
  (length-in-feet 150)
  (distance-between touchdown 500)
  (distance-between small-distance 500)
  (distance-between threshold 1000)
  (spacing-between 102)
  (distance-between touchdown 500)))
```

Figure 5 Loom description for Big Distance Marks.

small marks can be more readily found). Since some marks have been located, the image scale can be determined more precisely and the expected location of the new marks can be specified more exactly. The same descriptions of the marks are used to determine if the extracted apars are appropriate.

Additional refinements include merging the many side stripe fragments using the same procedure used for runway hypotheses, which reduces the number of individual side stripe fragments to one-tenth of the original numbers. The updated scale information is also used to eliminate distance marks that were within the original ranges, but are not close enough when the scale is known more accurately.

In the initial implementation, all the size and position information was specified directly in the extraction and analysis procedures. The first part of this project rewrote these procedures to use Loom to describe the markings (sizes, relative positions and position on the runway). This simplified the implementation (by reducing the number of procedures) and moved all the descriptions into a more understandable form (i.e. the Loom descriptions). Figure 5 gives the Loom description for big distance marks (called this by the program because of the physical size of the marking). From this, we know that a *big-distance* mark is a type of *generic-mark* (which in turn has several roles (or slots)). We also know the distance between this marking and other marks and the spacing (across the runway) between pairs of big distance marks. This shows the basic properties of the marking and the relations between it and other markings. Some properties and relations could be described as relations to the underlying runway hypothesis, but these geometric relationships would require extensions to Loom.

5 Loom Implementation

In the basic implementation of the runway analysis programs, the Loom language supports:

- the specification of definitional classes, such as the minimum runway length and types of markings that define a “precision instrument runway”;

```
(loom:defconcept a-runway
 :roles (runway-object))
(loom:defconcept a-runway-taxi
 :is (:and a-runway
   (:at-least 4 has-center-line-mark)
   (:at-least 2 has-side-stripe-mark)))
(loom:defconcept potential-runway
 :is (:and a-runway-taxi
  (:at-least 1 has-threshold-mark)
  (:at-least 1 has-touchdown-mark)
  (:at-least 1 has-small-distance-mark)
  (:at-least 1 has-big-distance-mark)))
```

Figure 6 Loom concepts for different runways

```
(loom:defconcept good-begin-runway
 :is (:and potential-runway
  (:at-least 1 has-threshold-mark
       begin-mark)
  (:at-least 1 has-touchdown-mark
       begin-mark)
  (:at-least 1 has-big-distance-mark
       begin-mark)
  (:at-least 2 has-small-distance-mark
       begin-mark)))
(loom:defconcept good-end-runway
 :is (:and potential-runway
  (:at-least 1 has-threshold-mark
       end-mark)
  (:at-least 1 has-touchdown-mark
       end-mark)
  (:at-least 1 has-big-distance-mark
       end-mark)
  (:at-least 2 has-small-distance-mark
       end-mark)))
(loom:defconcept good-runway
  :is (:and good-end-runway
            good-begin-runway))
```

Figure 7 Loom concepts for a good runway

- the specification of constraints on objects, such as the required distance between various types of markings, or the minimum width of a runway.

Loom concepts can be used to describe different classes of runways based on quantitative (and qualitative) differences in the set of markings. These are illustrated in Figure 6, which shows the basic runways, and Figure 7, which shows the description of a good runway.

These Loom concepts provide the basic classification of runways located in the image. The advantages of using a knowledge representation like Loom can be illustrated by the results of this experiment.

When these kinds of knowledge are moved out of the procedural representations (in this case in Lisp) into the declarative specification of Loom, the upgraded application becomes easier to comprehend. The descriptions are explicitly represented by the Loom concepts rather than coded in various programs and these descriptions are used by all the programs. Although some advantages could be obtained by using appropriate data structures directly in Lisp, Loom provides both the programming style and the retrieval mechanisms that lead to easier comprehension.

The knowledge is now in a more accessible form, which helps make it easier to extend and maintain. The declarative specification makes dependencies explicit rather than keeping them hidden. These dependencies are more than the inheritance of object descriptions (as in CLOS) since a runway becomes an excellent runway by virtue of changes in its descriptive markings rather than changes in the object class.

The descriptions are easier to share and reuse (because domain attributes are now formally specified in a language with a well-defined semantics), and the implementations become smaller when several functions are collapsed into a single one. The programs for all distance markings were collapsed into a single one after the conversion to Loom.

In the earlier implementation, the refinement operations were under direct user control. By using Loom retrieval mechanisms it is easier to automatically choose which runway hypothesis and which markings need more analysis, either by eliminating extra or invalid markings or by searching specifically for more markings of that type.

6 Reasoning Aspects

Converting to a Loom-based application makes new kinds of reasoning available to a developer. Loom provides a query language that operates over the objects, classes, and definitions in an application. In addition to allowing queries about the objects in the domain, Loom also allows meta-queries about the definitions of classes. The meta-level opens the possibility of building a more introspective reasoner.

The Loom production rule facility offers a modular means for defining such things as the heuristics that implement *object detectors*. The Loom constraint checker computes whether a hypothesis generated by an object detector satisfies a set of domain constraints. When conditions specified by a production rule are met, the rule is executed, thus allowing options for alternative control of the processing. At this time, we have not implemented significant production rules in the program.

7 Status and Results

The current system contains the automatic generation of runway hypotheses, and finds markings using the initial set of potential markings (i.e. the thin apars). An initial filtering is applied to the hypotheses (based on whether any valid markings are found). Further automatic refinements include finding more distance markings, verifying the threshold mark (which delineates the end of the runway) and updating the image scale (i.e. feet per pixel). The execution times are roughly a minute (Sun Sparc 10) for the initial hypothesis and initial set of markings. The refinement times depend on how many new markings must be found (and especially on side stripe and center lines since these require a search along the length of the hypothesis). As an example of our results, we show the selected runways from the Boston image in Figure 8. All 4 runways are classified as *excellent* (i.e. better than the *good* runway of Figure 7). Both ends of all runways are in the image, but the borders are cut off in this display. This image is the easiest in our set of images and all the runways are found clearly. The additional very short runway is not indicated since it does not have the distance marks.

Figure 9 shows the selected runways for one of the images for Los Angeles. The left side of the bottom two runways is not in the image so these have possible valid markings on only one end. The markings themselves are not as clear as for Boston and, overall, fewer are found. Figure 10 shows the only runway that was determined to be *excellent* by the same criteria as in the Boston image. In this case the second of the upper runways is missing a distance mark at 2500 feet.

8 Future Directions

Our future work (on the computer vision side) will include:

- exploring how far we can push Loom to the lower levels of the processing (i.e. turning apars into runway fragments and hypotheses);
- building Loom implementation for analysis of airports at higher levels (i.e. location and analysis of taxiways, aircraft, and functions of buildings); and
- implementing building detection programs in Loom.

Loom is a general purpose symbolic reasoner. Loom's strong point is reasoning about domain facts and recognizing instances based on those facts. The recognition task that VEIL undertakes involves searching for evidence of the existence of structures known to exist in runways. This involves reasoning by reference to a prototype of a runway. Loom could be enhanced by the addition of support for reasoning with concept prototypes. This enhancement would not only benefit VEIL, but would be useful in many other domains as well.

To more fully exploit Loom's existing symbolic reasoning capabilities within a vision application, the VEIL project will lead to extensions of Loom to per-

Figure 8 Boston Logan image with potential runways and their markings. The image has been brightened to make the markings display more visible.

Figure 9 Los Angeles image with potential runways and their markings. The image has been brightened to make the markings display more visible.

Figure 10 Los Angeles International -- Excellent runway only.

form processing specifically for use by vision applications. The most fundamental of the planned extensions will be in the area of spatial representation and reasoning. We will begin by adding to the (declarative) Loom language constructs that express the kinds of spatial relationships present in vision applications.

We will incorporate into the system implementations of existing spatial reasoning algorithms. A primary engineering challenge is to preserve the original performance of the spatial reasoners within the more declarative setting that Loom provides.

We will be testing how well we meet this challenge by recoding IRIS vision applications to use the new Loom capabilities, and then comparing performance between the original and Loom-based versions of the same application.

REFERENCES

[Canny 1986] J.F. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 8, No. 6, pp. 679-698, November 1986.

[Draper *et al* 1989] B. Draper, R. Collins, J. Brolio, A. Hanson, and E. Riseman, "The Schema System," *International Journal of Computer Vision, Vol. 2*, No. 3, pp. 209-250, 1989.

[Huertas *et al.* 1990] A. Huertas, W. Cole, and R. Nevatia, "Detecting Runways in Complex Airport Scenes," *Computer Vision, Graphics, and Image Processing, Vol. 51*, No. 2, pages 107–145, August 1990.

[MacGregor & Burstein 1991] R. MacGregor and M. Burstein, "Using a Description Classifier to Enhance Knowledge Representation," *IEEE Expert, Vol 6*, No. 3, pages 41-46, June 1991

[McKeown *et al.* 1985] D. M. McKeown, Jr., W. A. Harvey, and J. McDermott, "Rule Based Interpretation of Aerial Imagery," *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 7*, No. 5, September. 1985, pp. 570-585.

[Mundy *et al.* 1993] J. Mundy and The Gang of Ten, "The Image Understanding Environment: Overview," in *Proc. ARPA Image Understanding Workshop*, Washington, DC, April 1993, pp. 283-288.

[Nevatia & Babu 1980] R. Nevatia and K. R. Babu, "Linear Feature Extraction and Description," *Comp. Graphics and Image Processing, Vol. 13*, 1980, pp. 257-269.

[Winston 1975] P. H. Winston, "Learning Structural Descriptions from Examples," in *The Psychology of Computer Vision*, P. H. Winston, Ed., New York:McGraw Hill, 1975, Chapter 5.