Abstract

The VEIL (Vision Environment Integrating Loom) project focused on integrating advanced knowledge representation (KR) technology with image understanding technology. VEIL developed a more declarative approach to the construction of vision systems and produced a tool that incorporates that methodology. Systems were constructed in a more principled fashion that made it possible to share and reuse software across systems. Experiments in two main areas were carried out. We first demonstrated the utility of using Loom as a software engineering tool for a specific vision application (runway detection). We also demonstrated the benefits Loom provides for image understanding itself (event detection).

The major innovations in this work are as follows:

1) applied a methodology that maximizes use of declarative knowledge (as opposed to procedural knowledge) in vision systems, thereby enabling us to apply modern software development techniques. The criteria for recognizing objects was stated explicitly in a formal language (instead of being buried in code) making it easier to understand and maintain an application and keep it consistent. Extending the recognition capabilities of the software was made easier.

2) use of this declarative system construction methodology to facilitate the process of integrating high-level vision routines (such as for recognizing sequences of scenes) with low-level routines that recognize picture elements.

3) enabling interaction with the system at a level of abstraction appropriate to the domain task. This includes associating collateral information with the objects recognized by low-level image understanding programs.

4) development of a foundation for a vision ontology.

This work leveraged off the Loom Knowledge Representation system. Loom captures the best features of object-oriented programming, data-driven programming, problem solving, and constraint programming, through the use of an underlying logic-based representation scheme. This system is a powerful tool that incorporates very strong, frame-based representation capabilities, explicit term subsumption, and a number of powerful reasoning paradigms (including logical deduction, object-oriented methods, and production rules). Loom also provides knowledge representation integrity through consistency checking, and provides truth maintenance. Infusing these facilities into the vision problem area, where strong KR capabilities have not yet been developed will significantly alter and improve the methodology for the construction of vision systems. We also developed spatial and temporal reasoning capacities (critical for vision), along with mechanisms to exercise flexible control strategies and incremental scene processing. Finally, Loom was interfaced to a variety of vision processing elements to provide a new tool of extended capabilities. The net result is a powerful software environment for the development of vision systems.
# Table of Contents

Abstract

Part I. Introduction and Background
  1. Introduction
  2. Background
    2.1 Other Related Work.
    2.2 Overview of Loom Capabilities.
    2.3 Research Plan
      2.3.1 Using Loom in an Existing Image Program.
      2.3.2 Using Loom to Solve Domain Problems.
    2.4 Application Domain.
  3. Report Organization

  4. Software Engineering Experiment Overview
  5. Declarative Programming
  6. Airport Example
    6.1 Runway Hypothesis Generation
    6.2 Runway Hypothesis Verification
    6.3 Refinement of Hypotheses
  7. Using Knowledge Representation
    7.1 Representation Aspects
    7.2 Reasoning Aspects
  8. Status and Results
  9. Future Directions

Part III. Loom Applied to Domain Reasoning
  10. Domain Reasoning Experiment Overview
  11. Underlying Technology
    11.1 RADIUS
    11.2 Loom
      11.2.1 Definitions


List of Figures

Figure 1. Standard Runway Markings for Instrument Runway. .................................................. 13
Figure 2. Runway Detection Algorithm. .................................................................................. 14
Figure 3. Boston and Los Angeles Airports. ........................................................................... 15
Figure 4. Eliminate Contained APars. ................................................................................... 16
Figure 5. Join Common Segments. ......................................................................................... 16
Figure 6. Merge Colinear Apars Across Gaps ............................................................ 17
Figure 7. Big Distance Mark Description ............................................................................ 19
Figure 8. Basic Runway Concept Definitions ....................................................................... 20
Figure 9. Good Runway Concept Definitions ....................................................................... 20
Figure 10. Boston: Excellent Runways .................................................................................. 23
Figure 11. Los Angeles: Selected Runways .......................................................................... 23
Figure 12. Vehicle Hierarchy of the Domain Model ........................................................... 27
Figure 13. Domain Model for Visible Objects in VEIL ....................................................... 30
Figure 14. Image M16 — Plain and with Model Overlay ...................................................... 32
Figure 15. Extended Bounding Boxes for Computing Nearness ............................................. 34
Figure 16. Event Definition and Corresponding Loom Query ............................................... 35
Figure 17. Field Training Exercise Event Found in an Image Sequence ............................... 36
Figure 18. Close-Up View of Field Training Exercise ........................................................... 37
Figure 19. Map of Model Board 2 ....................................................................................... 43
Figure 20. Image M4 ............................................................................................................ 44
Figure 21. Image M6 ............................................................................................................ 45
Figure 22. Image M8 ............................................................................................................ 46
Figure 23. Image M12 ......................................................................................................... 47
Figure 24. Image M16 ......................................................................................................... 48
Figure 25. Image M19 ......................................................................................................... 49
Figure 26. Image M24 ......................................................................................................... 50
Figure 27. Image M27 ......................................................................................................... 51
Figure 28. Image M32 ......................................................................................................... 52
Figure 29. Image M33 ......................................................................................................... 53

List of Tables

Table 1. Geometric Functions and BCDE Objects .................................................................. 33
Part I.

Introduction and Background

1. Introduction

VEIL was implemented and tested in the area of aerial photointerpretation. The application makes use of high-level reasoning and knowledge representation facilities provided by the Loom system to produce more capable implementations than is possible without such facilities. This application is interesting and useful in its own right, and complements other related research programs funded by DARPA. Experience with the application has validated the VEIL methodology and provides guidelines on how other applications may be implemented in VEIL.

VEIL addressed the need to incorporate strong knowledge representation capability within computer vision systems. Many previous approaches to image understanding addressed the need for knowledge representation, but none took full advantage of the strong technology that had been built up in this area over the past decade for other areas of AI. At the same time, traditional knowledge representation research had not paid particular attention to the unique demands that would be required for successful application of knowledge representation technology to the demanding computer vision problem. We adapted the Loom Knowledge Representation System to computer vision needs. The research focused on integrating advanced knowledge representation (KR) technology with appropriate image understanding technology to develop a substantive and unique tool, called VEIL (for Vision Environment Integrating Loom), for generation of vision systems.

The Loom system supports the construction and maintenance of “model-based” applications—Loom’s model specification language facilitated the specification of explicit, detailed domain models. Its underlying knowledge representation system provided powerful built-in tools for reasoning with domain models, and it provided a variety of services for editing, validating, and querying the structure of Loom models. Vision applications benefited from adopting a more declarative approach to software specification in three different ways, all of which were gained by using a system like Loom:

1) The use of explicit, declarative knowledge structures, which made an application easier to debug and maintain. For example, the object recognition procedures encoded in a vision application were easier to identify, comprehend, and explain when they were phrased in a declarative rather than a procedural language.

2) The implementation of symbolic computation, which played an increasingly important role at the higher levels of vision processing. At these
levels, significant leverage was obtained by using Loom’s deductive reasoning facilities.

3) Finally, by declaratively specifying modules within a vision application that were easier to share and reuse. The use of declarative models served as a complement to evolution towards standards such as the Image Understanding Environment (IUE).

2. Background

Image understanding programs often incorporate a number of representations that were amenable to representation in a general KR formalism. However, due to the need to represent many other objects outside a formal KR system, image understanding systems generally used ad hoc methods at all levels of representation, thereby failing to take advantage of the strong capabilities that had been developed for this kind of technology in systems such as Loom. In the vision context, Loom provides control mechanisms for reasoning and a means of representing scene knowledge.

Loom incorporated previous and current research in knowledge representation and inference technology into a system designed to be used directly by applications programs. Loom was a continuously evolving system—each release of Loom supported additional reasoning capabilities, and for the past several years a new version of Loom was released approximately once every year. Loom has been distributed to over 80 universities and corporations.

The functionality, in terms of representational and inferential power, that Loom made available to users exceeded that delivered by current-generation expert system shells, while the inference technology represented by Loom’s description classifier had no analog in that technology. A key feature of Loom was the high-level of integration between its various embedded inference technologies, as demonstrated by the fact that typical Loom applications made use of three distinct programming paradigms (data-driven, object-oriented, and logic programming). Because Loom supports a more declarative style of programming, applications constructed using Loom have been easier to debug, maintain, explain, and extend than those based on that current-generation technology. These benefits were derived partly from the inherent nature of the language, and partly from the powerful support tools that the language made possible.

Within image understanding applications, such as aerial image analysis and vehicle navigation, many different kinds of basic objects are extracted from the image. These included image pixels (intensity, color, or range), individual edge elements from intensity or range images, connected edge elements, line segment approximations to the connected edges, edge-based contours, connected regions, surfaces, collections of regions or surfaces, and collections of other basic objects. All of these descriptions may have also had information regarding the viewing position or time for a sequence of data. Some of these were derived from other
representations in the list and were possibly linked to each other by spatial or semantic relationships.

Many of these object descriptions represented well-defined image structures with well-defined extraction techniques, and were not usually represented in the same terms that were used by knowledge representation systems. Nor was it desired that such a system as Loom attempt to embody every kind of object. Rather, the interest in a formal KR technique was for the power it brings to bear on the middle and high levels of image analysis, not the very low levels. For example, in the earlier aerial analysis system developed at USC/IRIS [Huertas 90], extraction of runway structures proceeded in a direct fashion from edge detection and line segment formation through extraction and verification of runway rectangles. These early stages of analysis were time consuming and entailed processing many (50,000+) basic elements to find the candidate structures. The techniques used were highly specialized processes which Loom could not improve upon. However, when further analysis of the aerial image was undertaken to extract higher levels of potential structure (e.g., taxiways, buildings, airplanes), using models of these same structures (via Loom's representation system), along with Loom's powerful reasoning capabilities, offered a new disciplined approach to handling this phase of the recognition problem, which constituted a marked improvement upon the then existing techniques. Thus, the VEIL system was able to combine the data-driven geometric reasoning modules specified above with model-driven symbolic reasoning tasks under a single programming environment to produce a more powerful tool than either system could have provided alone.

An additional problem with previous systems was that even when high-level knowledge was used, much of the information about the structure of the scene was embedded in the programs implementing the extraction (procedural knowledge representation) rather than in a declarative form. This made any changes and addition of knowledge for domains difficult. The Loom KR system increased the amount of declarative knowledge representation that could be done, thereby alleviating the problem.

2.1 Other Related Work.

There have been several attempts to develop high-level reasoning systems for image understanding. Three such systems, described below, are: SPAM developed by McKeown and his colleagues at CMU [McKeown 89]; VISIONS developed by Hanson, Riseman and colleagues at University of Massachusetts [Draper 89]; and ICC [Silberberg 87] developed at Hughes Research Labs. We believe that all of these systems suffered from the major drawback that they forced one to commit oneself to developing large vision systems with a single inference engine, whereas there were and are many tasks in vision better handled by different control structures. The Loom knowledge representation system
allowed the system designer to combine various control/knowledge representation paradigms in an explicit fashion.

SPAM was developed for and had been applied to aerial image analysis applications. It was basically a rule-based system implemented in OPS5. It processed a given image in the following phases: segmentation, class interpretation (of segmented regions), fragment interpretation, functional area analysis and finally, model generation and evaluation. Some interaction between different phases was possible. SPAM handled the demands of low level and high level analysis and handled the large data accompanying aerial images. We evaluated the drawbacks of SPAM to be in specific commitments to the manner in which images must be analyzed, and in largely procedural representations of knowledge, which obscured the control knowledge any designer wishes to express in this system. Under Loom, the problem solving strategy was explicit and easily modified to allow the designer to experiment with different structures.

VISIONS was essentially a “schema” or “frame” based system. Knowledge about each object was encoded in a schema, which was specialized to find an instance of this object. Schema instances communicated with each other via a global blackboard. This architecture had some advantages for parallel implementation. VISIONS was a very general purpose system that could, at least in principle, be applied to a variety of domains. In VISIONS, the knowledge representation was primarily procedural and many of the schemas needed to be about intermediate objects for which we had no intuition.

The ICC system had been used to represent and apply high level domain knowledge to target detection. An object was modeled, using frames, according to its 2D appearance in the image using collections of regions and lines, taking into account spatial, temporal, and contextual knowledge. A semantic network was used to represent a limited amount of relationships between objects. The image features were represented in the Symbolic Pixel Array [Payton 84] which allowed efficient retrieval of pixel and object properties and object spatial relationships. The analysis, which was both bottom-up and top-down, followed the hypothesize and verify paradigm: when an object was hypothesized, it gathered information for its slots which provided evidence for or against the hypothesis. A confidence measure for a hypothesis was computed based on the degree of belief and disbelief provided by the evidence. The declarative nature of the modeling in ICC was analogous to representation in Loom. There were at least two problems in ICC: first, the search strategy in ICC was computationally intensive for large numbers of scene objects, and second, the semantic network representation in ICC was limited.

Our objective in VEIL was to build a system that allowed for more declarative knowledge representation, where the generic vision processing (such as some of the geometric reasoning) was separated from the domain knowledge. In addition, it provided multiple reasoning techniques, something which was particularly
appropriate for machine vision. We believed that these attributes were missing in the previous systems such as SPAM, VISIONS, and ICC, and were of great advantage for programming, in general, and image understanding, in particular.

By encouraging users to represent significant portions of their vision applications in VEIL, we made available to them a large variety of features (including term subsumption reasoning [MacGregor 90b], role hierarchies, multiple knowledge bases, etc.) that were absent in the frame systems found in existing vision processing systems.

### 2.2 Overview of Loom Capabilities.

Loom [MacGregor and Bates 1987, Brill 1993] 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.

Loom has an immediate means for representing non-spatial properties of object models and model instances. For the vision domain, Loom was to be extended by adding spatial representations for two and three dimensions. The extension included specialized procedures to compute spatial properties such as what objects were located at a pixel or at a location in the world, and how objects were located with respect to one another. Frequently, the answer to a spatial question posed at the symbolic level was computed by descending a level down and performing computations with the more detailed spatial structures that inhabited that lower level.

The Loom system provides a powerful set of tools for formally defining the vocabulary and operations that applied to an application domain. Terms defined in Loom were automatically checked for consistency. Unlike the more primitive database notion of a “view relation,” which could only be used within database queries, a Loom term defined a concept that could be used within assertions as well as within queries. Once defined, a term could be used throughout a Loom-based application to promote the uniformity and conciseness of expressions that accessed the knowledge base. Loom’s term definition facility was key to deriving an ontology that could be shared across multiple applications, as described below.

Concurrent Loom development resulted in new extensions to the system, and a support environment for Loom enhanced its usability. Contexts were added and expanded in Loom in order to support VEIL reasoning. In particular, contexts were made into first class objects with the ability to make assertions. This made the use of contexts as a representation for individual images more convenient and allowed the rapid development of an event detection capability. A Web-based support environment for Loom called Ontosaurus was recently released that contains tools to facilitate browsing, querying, and editing Loom knowledge bases. Tools exist that enable Loom to retrieve data stored in a relational DBMS, and
future plans call for interfacing Loom’s successor to a persistent object storage system.

2.3 Research Plan

Our plan of work was straightforward. We developed a new vision programming environment tool, VEIL, that integrated powerful knowledge representation capabilities with useful vision processing techniques. To accomplish this end, we used the Loom knowledge representation system as a basis, extended it for application to vision system problems, incorporated vision processing algorithms, and applied the resulting tool to two image understanding problems to demonstrate its utility. Since efficiency was such a major issue for vision systems, we also evaluated the resulting system and improved performance where indicated. It was our view that the resulting system would offer major new capabilities to declaratively model vision problems, apply new kinds of reasoning to the vision problem, and to make the construction and maintenance of vision systems easier and more cost effective.

The research was conducted in two phases. The first phase demonstrated the utility of Loom as a high-level mechanism which could provide a much needed facility for knowledge representation within the image understanding (IU) domain. Following successful demonstration of the utility of Loom for this purpose, we undertook development of additional capabilities for Loom which addressed the particular requirements of the IU domain. These new capabilities were demonstrated on a slightly different problem than that used for the basic effort. We describe the basic effort immediately below. These two phases are described separately in Parts II and III of this report.

2.3.1 Using Loom in an Existing Image Program.

Our objective was to explore how Loom applies to programs used in the computer vision problem domain. We chose the runway detection and analysis task since we already had a “hard coded” version of the program, we had experience in helping with the transfer of this application to a Prolog based system, and this program operated at several different levels of analysis (low-level image analysis and higher-level geometric reasoning).

This runway detection system has grown and developed over a number of years, but it has not been easy to modify it to work on different problem domains, or to extend it in the current domain. This failure is due partly to the lack of general knowledge representation and reasoning capabilities which would allow a better separation of the knowledge about airports used in the analysis and the programs that implement the analysis.

Vision techniques, such as line finding, segmentation, perceptual grouping, and 3-D shape descriptions, were integrated with the Loom system. To accomplish
this goal, we had to discover methods of tying the reasoning provided by Loom to mid- and low-level processing techniques that were common in the vision community. Finally, we developed incremental control strategies designed to reason about specific objects or regions within an overall scene so that objects of high interest could be rapidly recognized.

The low-level processing (e.g. edge and segment extraction, initial grouping) did not benefit from the general representation and reasoning schemes of Loom and remained in Lisp. The higher-level analysis was better suited to using reasoning and representations available in Loom. These included the reasoning about where to look for other airport structures given the initial runway locations, analysis of connections between these structures, and a more general description and analysis of the markings on the runways. Given this reasoning, we approached the use of Loom in an incremental fashion by first developing the higher-level analysis and knowledge in Loom and only later moving the use of Loom to the middle and lower level processing. Having these capabilities at all three levels then allowed us to incorporate feedback mechanisms that explored the images for further evidence and process portions of images at higher resolutions.

### 2.3.2 Using Loom to Solve Domain Problems.

Loom had already provided powerful representation and multiple reasoners such as logical deduction, object-oriented methods, and rule production. Loom was developed for more traditional AI problems and does not have some of the capabilities required for computer vision, such as a geometric reasoning capability, thus a part of the experiment identified the current limitations and developed techniques to address them.

For VEIL the following capabilities were added to Loom: spatial representation and reasoning; flexible control of instance recognition and classification; and incremental scene processing. In the area of spatial reasoning we added new constructs for representing such notions as coordinate location (2-D and 3-D), regions, distances, and nearness and adjacency relationships. Furthermore, we integrated the high-level representation of Loom with visual recognition processes exploiting geometrical and/or functional descriptions of physical objects. The method we used is similar to the technique used by Haarslev [Haarslev et al., 1994].

**Spatial Reasoning.** As discussed above, VEIL implemented multiple spatial reasoning algorithms designed to process both high and low levels of spatial representations. The spatial reasoner for top level (symbolic) knowledge was handles relatively general, declarative representations of spatial knowledge. The lower-level processing was performed by one or more special-purpose algorithms already developed for other vision processing systems (e.g., [Payton 84]). One of our tasks was to write a query translator to transform symbolic spatial queries into equivalent queries on the lower level knowledge structures.
For our vision applications, we expected that most symbolic spatial knowledge would be created through the process of abstracting lower level spatial representations. One of our tasks was to implement a VEIL component to perform such abstractions. Thus, the system would have the option of answering queries either by translating symbolic level queries into lower level queries, or by abstracting relevant spatial knowledge into the symbolic level, and then processing the query at that level. The latter method left the system with the option of saving (caching) the abstracted knowledge. One example of this involves the use of a declarative description of a convoy as “a group of vehicles located on a road.” Once a particular group of vehicles is recognized as being a convoy, further queries and processing no longer needs to reference the low-level objects that make up that convoy.

Our architecture overlayed a declarative spatial representation on top of a highly optimized spatial reasoner that used specialized representation structures tuned to the needs of high performance algorithms. If the vision processing community achieves standardization of data structures and algorithms for representing and reasoning with spatial knowledge during this contract, we will investigate the possibility of converting our architecture to match that standard. Placing a high level layer of reasoning above such a standard would yield a means for delivering high performance spatial reasoning to a relatively wide community of users. It also allowed us to develop a capability to detect events, which are sequences of images with domain importance. Armored movements and field training exercises are examples of such domain-level events.

**Temporal Reasoning.** Detecting events meant that capability for representing and reasoning with temporal knowledge was needed. We used the context mechanism added in Loom version 3.0 in order to implement a snapshot temporal model. This provided a natural representation for a series of images taken at different times. For monitoring a given site for changes, we were also able to exploit the hierarchical nature of Loom contexts to allow a shared background model. This “site model” provides a single, shared repository for information that does not vary with time. Most of the buildings and terrain can thus be shared among all of the individual “image models.” The individual image models allowed the tracking of the positions of vehicles as they moved from one image to another. The ability to create queries that spanned several images made the construction of an event detector straightforward.

**Enhanced Understandability.** In addition to guiding the system, the use of a declarative domain model (particularly one expressed in Loom) has other benefits as well. Declarative representations of knowledge are in general easier for a human to understand than procedural representations, since all knowledge is explicit. Also, inferences derived from a base of declarative knowledge are explainable—a relatively straightforward derivation of support had to exist for each derivable fact. Practical benefits were that declaratively represented portions of a program were in general easier to debug and maintain (because they
were modular and explainable), easier to extend (because of their modularity), and easier to share and reuse (because declarative representations reduce the use of clever or obscure encodings of knowledge). For these reasons, it was desirable that significant portions of an application be represented declaratively.

2.4 Application Domain.

In order to provide operational feedback, and to evaluate the success of our endeavors, we applied the evolving tool to a common vision domain, namely the photointerpretation of aerial imagery. This domain is of vital importance to the military and contained a rich variety of objects, both man-made (buildings, transportation networks, power transmission lines, and pipelines) and natural. Most of the objects are stationary but mobile objects are also present. The test domain is explained in more detail later.

The domain of aerial images contained a rich variety of man-made and natural objects. Major man-made objects included buildings, transportation networks (roads, railroads, runways etc.), power transmission lines, and pipelines. Most of these objects were stationary and changed slowly, but important mobile objects were also present (trucks, cars and airplanes). The images also contained natural terrain and vegetation. Some of the objects were very large with complex structures, while others were very small.

Typical aerial images were of “natural” scenes, where neither the illumination nor the nature of the observed surfaces could be easily controlled. This implied that, not only was the domain complex, but also the signal that we had to start with was far from ideal; usually, low level algorithms produce segmentations that differ significantly from the desired result. This richness and complexity made the task of aerial image analysis extremely challenging.

3. Report Organization

A detailed report of the application of Loom to the development and extension of an existing program for runway detection is described in Part II of this report. The application of Loom to the problem of integrating higher-level knowledge and detecting semantically meaningful events is described in Part III. Part IV provides a summary.

Appendix A reports on related support work for the Image Understanding Environment (IUE). This work was also performed as part of the VEIL contract. The Image Understanding Environment 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.
Part II.

Loom Applied to the Implementation of Vision Systems.

The first part of the VEIL project focused on integrating advanced knowledge representation technology (provided by Loom) with current image understanding technology to develop advanced tools for the generation of vision systems. This effort was aimed at eliminating a weakness in computer vision technology in the realm of higher level representations. The resulting hybrid system exhibits improved shareability, maintainability and reusability of code for computer vision systems.

4. Software Engineering Experiment Overview

VEIL integrates advanced knowledge representation technology (as developed in Loom) with image understanding technology to develop advanced tools for the generation of vision systems. The goal of this part of the project is to improve capabilities in high level computer vision systems through the use of mature, highly developed knowledge representation and reasoning techniques. We used advanced knowledge representation for computer vision to improve shareability, reuse and to simplify the development of high level vision programs. We have applied the Loom knowledge representation language to existing computer vision programs with an improvement in readability and extensibility without a substantial loss in execution time.

This experiment investigated 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 transformed an existing mature 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.

5. Declarative Programming

Domain knowledge may be represented procedurally, as program code, or declaratively. Declarative representations take many forms, but the distinction is that the representation itself is not executable program code but is data used by the program. A declarative specification provides a formal, semantically well-founded description that offers numerous benefits. Such a specification is more readable and easier to maintain and is subject to automatic validation and verification techniques. The description uses 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. Finally, the descriptions can be shared, modified, and reused by other applications more easily than procedural specifications.

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. We use declarative descriptions for the generic objects such as a runway and the markings on a runway to control the processing of the data. We use Loom’s classification, query and production rule capabilities to select the final objects from the scene. 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;
- 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.

6. 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]. Each of these primitives is a long thin ribbon (represented as an image feature called an *apar*), though the size and relations among them vary. Figure 1 shows the common markings for an instrument runway.

![Figure 1. Standard Runway Markings for Instrument Runway.](image-url)

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 [Huertas *et al* 1990], we will give only a brief description of these techniques in this report.

6.1 Runway Hypothesis Generation

The basic steps in finding runway hypotheses (which are also used for the taxiway hypothesis generation) are described in the flow chart of Figure 2. Runway generation begins by generating two sets of edges, stronger edges for the
runways and weaker edges for the markings. These edges are grouped into straight line segments and then grouped into anti-parallel pairs (ribbons, or apars). The runway hypothesis generation then proceeds through a series of filtering and grouping steps: Filter out contained apars, group apars sharing a common segment, and group colinear apars across gaps. Twice, the results are filtered to remove very short runway fragments (aspect ratio filtering). These steps produce a reasonable number of hypotheses (e.g. 14) from the original set of many ribbons (e.g. 18,000). The numbers of objects (i.e. 26,410 segments) are taken from the Boston Logan International Airport example. They are typical of the numbers for other airports.
The details of the algorithm are described below.

- 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 and 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, Figure 3). The reduction in numbers is similar for other examples, but much less pronounced for airports with all runways in the same direction (Los Angeles, Figure 3).

![Figure 3. Boston and Los Angeles Airports.](image)
• 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 4 illustrates this operation. Typically, about one-third of the apars survive this filtering.

![Figure 4. Eliminate Contained APars.](image)

• 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 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 5 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).

![Figure 5. Join Common Segments.](image)

• 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 6
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 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.

### 6.2 Runway Hypothesis Verification

The verification step requires analyzing the hypotheses to find the specific markings. Figure 1 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. Knowledge representation systems do not currently support spatial reasoning for searches so the details of the search fall on the image analysis system.

First the true ends of the runways must be located. The positions of the markings on the runway are well defined once the true end of the runway is known. But, the hypothesized end of the runway is not always the true end due to errors in the input or extensions of the paved runway surface. The true ends of the runway 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

![Figure 6. Merge Colinear Apars Across Gaps](image-url)
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.

At this stage in processing, the image scale is approximately known but the program does not assume it has the exact scale or the exact position of the threshold marks. Furthermore, in the original edge and feature extraction, the larger marks are often broken into shorter apars. Therefore the extraction allows for a considerable tolerance in the location (especially along the runway) and size of the marks (especially the length). The initial set of markings is used to refine the scale and then to filter out other markings in incorrect positions.

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). Both of these are very narrow (roughly 1 pixel) so they tend to break up into many small pieces.

6.3 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 well-defined markings, but some runway markings are missed due to errors in the anticipated position, errors in the edge detection, the thresholds used for the edge detection, or because they are very low contrast in the image. Furthermore, the markings are near the image resolution limit with widths of one or two pixels for the smaller markings. All of these problems are countered by the refinement steps.

More markings are located by reapplying the edge, line and apar finding procedures on small windows (50 by 50 pixels) of the image with very low thresholds and using a replicated version of the image so that small marks can be readily found. Also, by using the locations of previously found markings, the image scale can be determined more precisely and the expected location of the new marks can be specified more exactly (i.e. relative to other established markings). The same processing used to evaluate candidates for the original set is used for the refinement, except that the input is taken from the data extracted in the window in the image rather than taken from the globally extracted features. Descriptions of the size and location of marks are used to rule out candidates and to determine if the extracted apars are appropriate. Loom provides support for the creation and use of such descriptions. This was exploited in the new implementation.

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.

7. Using Knowledge Representation

Through the initial hypothesis generation and initial verification, there is little use of any high level knowledge representation techniques. For this basic analysis, Loom concepts are used to:

- describe the elementary objects, such as the runway length and width, the types and shapes of the markings;

- describe the constraints on objects, such as the required distance between various types of markings, or the number and kinds of markings that must be located. An example of big distance marks is given in Figure 7:

  (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)
     (distance-between threshold 1000) )
     (spacing-between 102))
  )

**Figure 7. Big Distance Mark Description**

- describe different classes of runways based on quantitative (and qualitative) differences in the set of markings. These are illustrated by Figure 8, which shows several basic runway types;
• and describe different quality classes based on the presence of recognized image features (markings) on the runway. Figure 9 shows the description of a good runway, one that is clearly identified.

The use of Loom knowledge representation capabilities for the runway models, the marking models, the descriptions of the extracted runways, and for the evaluation of the extracted runways contributes to the simplification of the resulting program. Since we were starting with an existing program for the task we are able to compare the differences in procedural embedding of domain knowledge and declarative representation of that same knowledge.

### 7.1 Representation Aspects

In the initial implementation, all the size and position information was specified explicitly 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 to one from one for each marking) and moved all the descriptions into a more understandable form (i.e. the Loom descriptions). Some of these advantages are available using standard data structures, but these are not well suited for global data structures and general queries to extract values.

In the description in Figure 7 for big distance marks (so called by the program because of the physical size of the marking) we see that a big-distance mark is a type of generic-mark, which in turn has several roles (or slots). We also see the distance between this marking and others and the spacing (i.e. across the runway) between pairs of big distance marks. The marking concepts contain the basic properties and the relations between markings. To describe the position properties and relations relative to the underlying runway hypothesis would require extensions to Loom to handle geometric relationships and uncertainty.

For our application, moving basic descriptions of this type out of the procedural representations (in this case Lisp procedures) into the declarative specification (i.e. Loom) simplified the implementation. The descriptions are explicitly represented by the Loom concepts and thus can be used by all procedures in the analysis. Although some of the same advantages can be obtained by using appropriate data structures directly in Lisp, Loom provides both the programming style and the retrieval mechanisms that simplify the implementation. In this application, the three original procedures for each separate distance marking (plus three more used for the refinement step) were replaced by a single procedure for all markings (and this procedure is roughly the same size as each of the previous individual ones).

Two advantages of declarative descriptions are shareability and reuse. The descriptions used here are still specific to the problem domain so they are not easily shared with other applications. The declarative descriptions were easier to modify and extend than the procedural specifications so that extensions of marking refinement to cover all markings was trivial, once it was implemented for one of them.

7.2 Reasoning Aspects

In the earlier implementation, the refinement operations and final runway selection were controlled directly by the user. By using Loom reasoning and retrieval mechanisms it is possible to automatically choose which runway hypothesis and which markings need more analysis. Thus runways with extra markings (along with the markings themselves) can be selected or missing markings can be indicated by the retrieval mechanisms rather than by procedures that examine all options. This query mechanism is used to select which runways analyze further to clean up extra marks or find more.
The Loom production rule facility offers a modular means for defining such things as the heuristics that implement object detectors. 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, but it would be easy to use rules that trigger on the detection of potential runways that are not yet recognized as good. Such rules would then direct the low-level image analysis routines to expend more effort looking for the missing items. This would provide a global expectation-driven flow of control.

The declarative specification makes dependencies in descriptions and interpretations explicit rather than keeping them hidden. These dependencies are more than the inheritance of object descriptions (as in CLOS or C++) since a potential runway becomes a good runway by virtue of changes and additions to its associated descriptive markings rather than changes in the object class. The Loom constraint checker computes whether a hypothesis generated by an object detector satisfies a set of domain constraints. These changes are also recognized by Loom production rules. They can fire when a runway of a given interpretation is recognized — i.e., when enough markings are identified.

8. Status and Results

The implemented system generates runway hypotheses and verifies them by location markings found from the initial set of potential markings (i.e. the thin apars). It also applies initial filtering to the hypotheses (based on whether any appropriate markings are found). Further automatic refinements include finding more distance marks, verifying the threshold mark (which delineates the end of the runway), finding more center and side stripes, and updating the image scale (i.e. feet per pixel). The execution times are roughly a minute (Sun Sparc 10) to compute the initial hypothesis and the initial set of markings. The time required for the computation of initial apars or even reading them in to the program is greater than the time used for hypothesis generation and verification. 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), but each subwindow (window selection, edge detection, apar extraction, evaluation, display) requires 2-3 seconds. Because Loom accesses are used through the program, it is impossible to separate out execution costs, but execution time is dominated by the basic image feature extraction and grouping processes.

As an example of our results, we show the selected runways from the Boston image in Figure 10. All 4 runways are classified as excellent (i.e. better than the good runway of Figure 9). 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 10. Boston: Excellent Runways

Figure 11. Los Angeles: Selected Runways
Figure 11 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.

The results for the initial (non-Loom) version of the program were not as complete. Since the refinement steps were difficult to run, they were never completed. The declarative representations made this possible. Additionally, in the current version we introduced the use of a general spatial index for many of the spatial operations that changed the computation from one best described as taking days to one taking minutes. In terms of computation time, this change was more important than any improvements in speeds of machines used for the project.

9. Future Directions

The work on runway analysis is completed but we will be applying general knowledge representation techniques to other application domains in our general research work. The major areas of future work are:

- extending the use of Loom to “lower” levels of the vision processing to see where the computation is overwhelmed by the volume of image features;

- applying techniques similar to those used in the runway program to building extraction and analysis (this adds three-dimensional reasoning issues to the problem domain);

- applying Loom to higher level problems in vision such as reasoning about changes in the image using the objects (i.e. buildings) extracted by other processing. This is addressed in the next part of this report, describing the application of Loom to domain reasoning.

- extending Loom to directly handle spatial concepts used by computer vision algorithms.

Loom is a general purpose symbolic reasoner. Loom’s strong point is reasoning about domain facts and recognizing instances based on those facts. The visual recognition tasks that VEIL undertakes involve searching for evidence of the existence of features known to exist (i.e. runways and their markings). 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.