# Automating Knowledge Acquisition For Aerial Image Interpretation

David M. McKeown,Jr.,Wilson A. Harvey
January 28, 1987
CMU-CS-87-102

Invited Paper Presented At
The DARPA Image Understanding Workshop
Los Angeles, February 23-25, 1987.



This research was primarily sponsored by the Defense Mapping Agency, under Contract DMA 800-85-C-0009 and partially supported by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-81-K-1539. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Mapping Agency, of the Defense Advanced Research Projects Agency, or the US Government.

# Table of Contents

## List of Figures

## Abstract

The interpretation of aerial photographs requires a lot of knowledge about the scene under consideration. Knowledge about the type of scene: airport, suburban housing development, urban city, aids in low-level and intermediate level image analysis, and will drive high-level interpretation by constraining search for plausible consistent scene models. Collecting and representing large knowledge bases requires specialized tools. In this paper we describe the organization of a set of tools for interactive knowledge acquisition of scene primitives and spatial constraints for interpretation of aerial imagery. These tools include a user interface for interactive knowledge acquisition, the automated compilation of that knowledge from a schema-based representation into productions that are directly executable by our interpretation system, and a performance analysis tool that generates a critique of the final interpretation. Finally, the generality of these tools is demonstrated by the generation of rules for a new task, suburban house scenes, and the analysis of a set of imagery by our interpretation system.

## 1. Introduction

In this paper we describe a collection of software tools, ISCAN/RULEGEN/SPATS, for interactive acquisition of spatial knowledge, automated compilation of this knowledge into a rule-based scene interpretation system, and the production of performance analysis statistics to aid in incremental refinement of spatial knowledge. This work is focused on knowledge acquisition and performance analysis tools for SPAM, a knowledge-based system designed to interpret aerial photographs for mapping and photo interpretation. We have reported on SPAM research results in the context of airport scenes[1, 2].

We address a broad set of topics within the overall framework of knowledge acquisition. First and foremost we are interested in automating the process by which an interpretation system, such as SPAM, can collect and represent new knowledge to improve performance on existing interpretation tasks, or in attempting to begin to become proficient in new ones. For the airport task we primarily relied on spatial constraints found in books on airport design[3, 4, 5] and, to a lesser extent, by observations of relationships found in aerial imagery. Other task domains, such as suburban house scenes, do not appear to have codified spatial organizations, although they exhibit similar patterns across many examples. In lieu of such information the ability to indicate and measure spatial relationships in representative imagery becomes more important. ISCAN is our first attempt to provide a graphical user interface, appropriate in an image-based domain, which has a model of the types of knowledge required by SPAM during the interpretation process. Such an interface may also provide individuals such as cartographers, remote sensing and photo interpreters, and other non-programmers with a mechanism for adding knowledge to SPAM without a detailed understanding of the underlying system.

A second research goal is to explore the generality of the SPAM architecture for a variety

of tasks within the general domain of aerial image interpretation. RULEGEN is a tool that compiles spatial and structural knowledge, stored as collections of rule schemata, and generates productions that are executed by SPAM. RULEGEN was partly motivated by difficulties encountered in extending and generalizing SPAM, which was developed to interpret airport scenes, to interpret simple suburban house scenes. Many of these difficulties impacted our ability to easily add and delete rules to measure the effect of knowledge in various phases of the interpretation process. Changes in the knowledge base often generated unforeseen interactions between the control of rule execution within the interpretation phases. In a system with over 500 productions the management of such changes became a significant burden. As we began to address these issues it became clear that the solution was to view SPAM as an interpretation architecture within which we could embed specific task knowledge. RULEGEN was developed to automatically generate the core task-independent evaluation and control functions that represent the SPAM interpretation architecture and to take task-dependent knowledge in the form of rule schemata and compile productions whose execution was embedded within this core. Therefore, the performance system, SPAM, can be completely generated by RULEGEN when it is supplied with appropriate task-dependent knowledge.

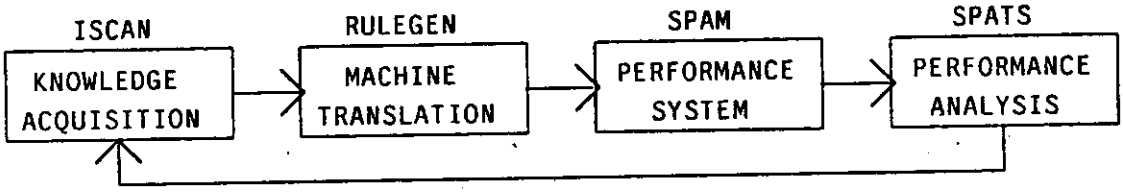



**Figure 1-1:** Overview of Knowledge Acquisition For SPAM

Finally, SPATS was motivated by a need to automate the evaluation of the interpretations produced by SPAM within the context of idealized human photo interpretation. The goal was to measure the size of the interpretation space explored by SPAM, the number of competing hypotheses, and the correctness of those hypotheses during each interpretation phase. By varying the image segmentations presented to SPAM or by generating SPAM systems with different types of spatial knowledge we can now more rigorously evaluate and explore knowledge effects using SPATS. Figure 1-1 is an abstract overview of the relationship between these tools. While this particular focus on acquisition, compilation, and performance evaluation might appear to be somewhat parochial, we believe that these issues will be seen to be central to other researchers in computer vision working along similar lines.

Section 1.1 briefly outlines some related research and describes our views on knowledge acquisition for computer vision. Section 1.2 gives the layout of the remainder of the paper.

### 1.1. Knowledge Acquisition For Vision

Previous efforts to investigate knowledge acquisition within the context of systems for image interpretation have primarily focused on spectral properties of objects in the image or viewpoint specific spatial relationships. Early work by Barrow and Popplestone[6] addressed the problem of describing relations between picture elements with predicates like ADJACENT(x,y) or ABOVE(x,y). Using this methodology "rules" could be formulated from these predicates and attached to individual elements of a picture. For example, in the context of face recognition, a nose would be defined by the rule: "ABOVE(x,mouth) and LEFT-OF(x,right-eye) and RIGHT-OF(x,left-eye)". These rules were to be embedded into a resolution theorem proving paradigm. This work was a basis for the ISIS[7] system which added the use of an interactive segmentation system. It allows a user to interactively specify representative regions with a particular interpretation, and then invoked an intensity classification segmentation process to attempt to extract the remaining parts of the scene.

Recently, the VISIONS system[8, 9] has reported similar attempts to make interpretations by propagating low-level process output, such as lines or regions, up to an intermediate level, which combines the low-level output with computed attributes such as color, texture, or orientation. Interpreted objects are defined in terms of these intermediate elements. Loosely speaking these classification systems use "knowledge" such as *the sky has a pixel intensity greater than 30 but less that 125 in the blue band.* In fact, one must resort to density weighting functions much as in statistical pattern recognition for remote sensing. This "knowledge" is highly sensor and scene dependent. Other measures such as height, size (in pixels), and relative spatial position (e.g. *sky is above the house* and *grass is below the house*) are also employed. Again, these viewpoint dependent quantities will vary, not only from domain to domain, but from image to image. Ultimately *sky is blue* and *grass is green* allows for a direct mapping between regions and the associated high-level interpretation. However, this mapping represents a rather shallow use of knowledge whose robustness is questionable. For example, consider the effect of averaging the RGB components of a color image into a monochromatic image. While the scene geometry remains unchanged, without the direct mapping of region spectral properties into a semantic interpretation (*sky is blue*) it is difficult to see how to operationalize much of the spatial knowledge. Thus, although there appears to be a spatial component, it is predicated on strong mapping between color and interpretation.

In our work with SPAM we have attempted to identify sources of knowledge that did not suffer from these drawbacks, and utilize spatial relationships in such a way that a chain of reasoning exists, generated from the application of many constraints across multiple levels of interpretation. While spectral knowledge can play a role in certain domains we believe that there are many types of spatial knowledge that can be expected to be more effective in driving the knowledge-based interpretation of aerial imagery. In terms of acquisition

and utilization, we believe that Figure 1-2 lists 5 types of knowledge that are available and appear to us to be effective in aerial image interpretation tasks.

Type 1: Knowledge for the determination and definition of appropriate scene domain primitives. This includes knowledge of the image segmentation process, the image analysis tools that can reliably extract these primitives, and the appearance of the primitives in the image.

Type 2: Knowledge of spatial relationships and constraints between the scene domain primitives.

Type 3: Knowledge of model decompositions that determine collections of primitives which form "natural" components of the scene. These components can be characterized as sub-models that accumulate support for local interpretations and provide a context within which global analysis can be performed.

Type 4: Knowledge of methods for combining these components into complete scene interpretations.

Type 5: Knowledge of how to recognize and evaluate conflicts between competing interpretations.

**Figure 1-2:** Types Of Knowledge Utilized In SPAM

### 1.2. Layout of the Remainder of Paper

In the following section we briefly describe the architecture of SPAM. We discuss the kinds of knowledge that SPAM utilizes and therefore needs to be acquired for an interpretation task. In Section 3 we describe the ISCAN/RULEGEN/SPATS tools and in Section 4 give an example of the schemata produced by ISCAN and used by RULEGEN to generate a SPAM interpretation system. Finally, in Section 5 we give an example of suburban house scene interpretation by a SPAM system generated using the ISCAN/RULEGEN/SPATS tools. We also compare the structure of the original hand generated SPAM system with those generated using these knowledge acquisition tools.

## 2. The SPAM Architecture

SPAM represents four types of interpretation primitives, *regions*, *fragments*, *functional areas*, and *models*. SPAM performs scene interpretation by transforming image *regions* into scene *fragment* interpretations, aggregating these fragments into consistent and compatible collections called *functional areas*, and selecting sets of functional areas that form *models* of the scene. Loosely speaking there are four *phases* of interpretation. Each of these four phases operationalizes one or more of the five types of domain knowledge. In order to build a SPAM system we must be able to acquire knowledge for each interpretation phase as described in Figure 2-1.

As shown in Figure 2-2 each phase is executed in the order given above. SPAM drives

Phase 1: Region-to-fragment

Assigns the image region data a set of fragment interpretations based solely on local properties (2-D shape characteristics, texture, 3-D depth/height, etc.) and knowledge about the classes of objects found in the scene.

Phase 2: Local-consistency-check

Pair-wise tests are performed on the fragment interpretations that utilize spatial knowledge about the scene under consideration. The confidence of those interpretations supporting one another are incremented based on the quality of the test.

Phase 3: Functional-area

Sets of mutually consistent interpretations that share similar functions or are spatial decompositions of the scene are grouped into cliques called functional areas.

Phase 4: Model-generation

Sets of functional areas are grouped together into scene segments. The segments with the largest number of functional areas become distinct scene models. Any conflicts encountered when combining functional areas are resolved by a default strategy, using the accumulated support for each interpretation, or by specific knowledge added by the user.

**Figure 2-1:** Interpretation Phases In SPAM

from a local, low-level set of interpretations to a high-level, more global, scene interpretation. There is a set of hard-wired productions for each phase that control the order of rule executions, the forking of processes, and other domain-independent tasks. However this "bottom-up" organization does not preclude interactions between phases. For example, prediction of a fragment interpretation in *functional-area* phase will automatically cause SPAM to reenter *local-consistency* phase for that fragment. Other forms of top-down activity include stereo verification to disambiguate conflicting hypotheses in *model-generation* phase and linear alignment in *region-to-fragment* phase. Figure 2-3 shows the refinement/consistency/prediction paradigm used in SPAM within each interpretation phase. Knowledge is used to check for consistency among hypotheses, to predict missing components using context, and to create contexts based on collections of consistent hypotheses. Prediction is restrained in SPAM in that hypotheses cannot predict missing components at their own representation level. A collection of hypotheses must combine to create a context from which a prediction can be made. These contexts are refinements or spatial aggregations in the scene. For example, a collection of mutually consistent runways and taxiways might combine to generate a runway functional area. Rules that encode knowledge that runway functional areas often contain grassy areas or tarmac may predict that certain sub-areas within that functional area are good candidates for finding such regions. However, an isolated runway or taxiway hypothesis cannot directly make these predictions. In SPAM the context determines the prediction. This serves to decrease the combinatorics of hypothesis generation and to allow the system to

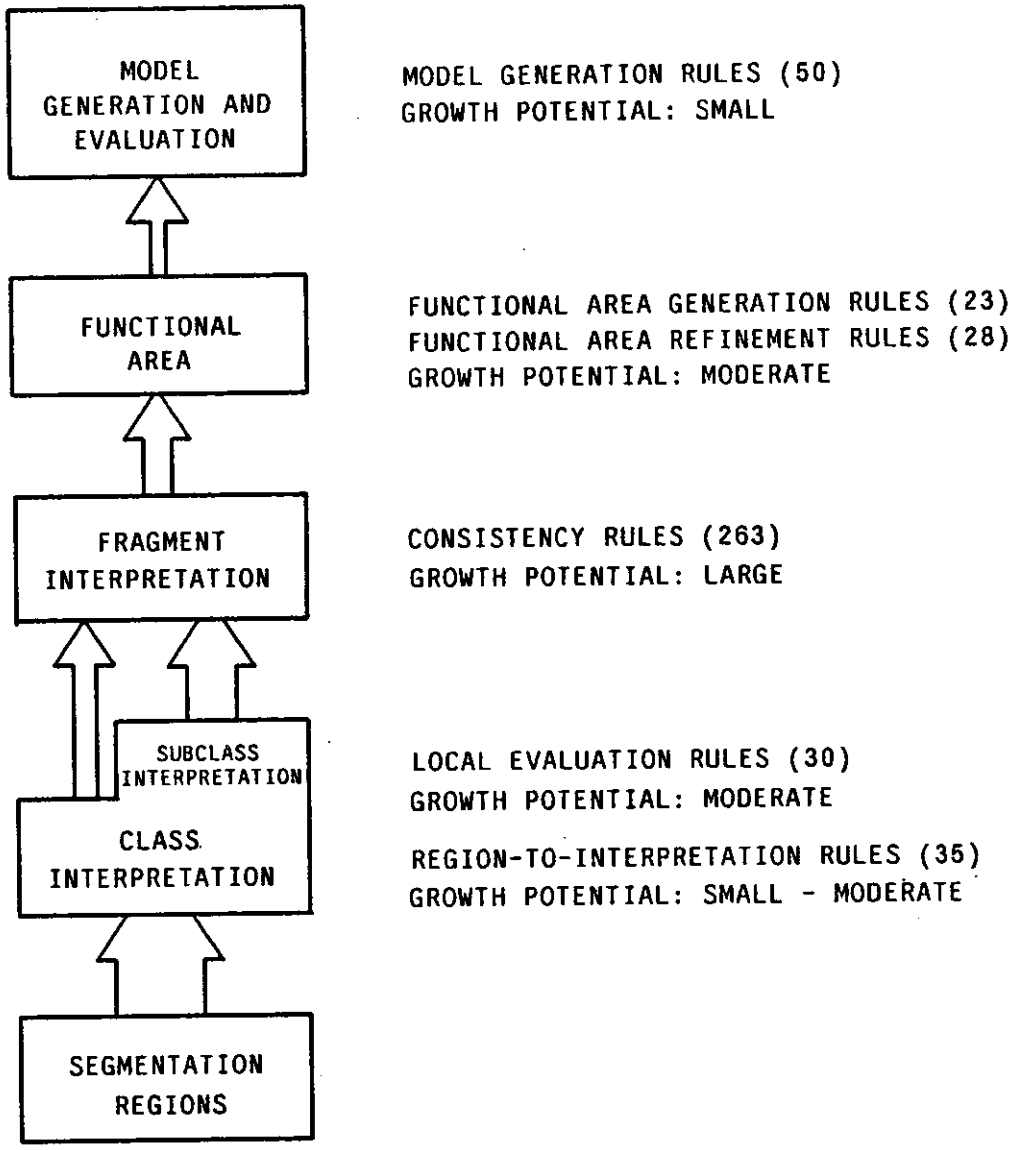



**Figure 2-2:** Interpretation Phases In SPAM

focus on those areas with strong support at each level of the interpretation.

## 2.1. Knowledge Acquisition In SPAM

In order to automate knowledge acquisition for SPAM we must be able to identify the kind of knowledge required for each of the interpretation phases described in the previous section. In this section we describe this with respect to the 5 types of knowledge defined in Figure 1-2.

The type of knowledge required in *region-to-fragment* phase is the definition of the shape and appearance properties of objects in the task domain, organized as coarse classes of similar objects with specializations based on finer intra-class distinctions. For example,

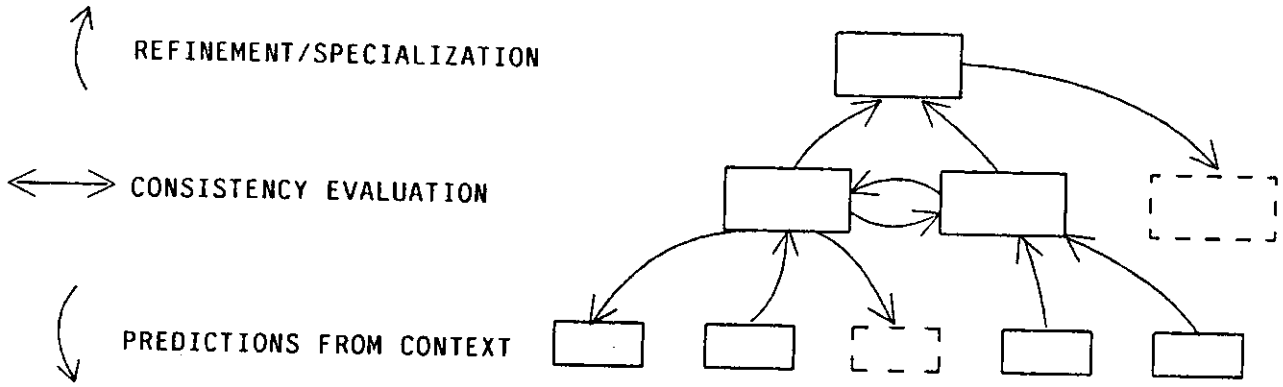



**Figure 2-3:** Refinement, Consistency, and Prediction in SPAM

in a coarse sense, linear features such as roads, runways, taxiways can be grouped in one class, while hangars, maintenance buildings, control towers, and terminal buildings would be another coarse class in an airport interpretation task. Each of the members of the class can be specialized with constraints such as runways are never curved, while roads may be curved. Heights, sizes, and specific shape criteria might be used to specialize the building class. This type of knowledge is best represented as Type 1 in Figure 1-2.

During *local-consistency-check* phase knowledge of the structure or layout of the task domain, i.e. airports, suburban housing developments, is used to provide spatial constraints for evaluating consistency among fragment hypotheses. Type 2 knowledge is required from the user or other sources. For example, *'runways intersect taxiways'*, and *terminal buildings are adjacent to parking apron'* are the kinds of knowledge in terms of spatial relationships that we would like to capture for an airport interpretation task. It is important to assemble a large collection of such consistency knowledge since these tests are used to assemble fragment hypotheses found to be mutually consistent as contexts for further interpretation.

There are two types of knowledge necessary to perform *functional area* phase. The first is primarily Type 3 knowledge which defines collections of objects that form spatial decompositions within the task domain. For example, knowledge that *runways*, *taxiways*, and the *grassy areas* that separate them from the area where planes takeoff and land can be used as one partition of the overall airport scene. Within this context Type 5 knowledge aids in prediction of missing components, selection of competing hypotheses, or in defining methods for disambiguating conflicting interpretations. For example, 'if a runway functional area has been formed and it contains a terminal building fragment then use stereo verification to confirm or refute that fragment hypothesis.' In other cases knowledge that simply selecting the competing fragment with the highest confidence based upon cumulative application of *region-to-fragment* and *local-consistency-check* rules may be appropriate.

Finally, during *model generation* phase, Type 4 knowledge consisting of how to combine spatial decompositions and Type 5 knowledge consisting of how to recognize and evaluate conflicts that arise during this aggregation must be acquired. However, much of this is simply selecting a strategy, i.e., 'use the functional areas with the highest confidence that have no conflicts', or 'find the maximal set of compatibles regardless of confidence'. The process for performing these alternative combinations is, in some sense, hardwired in the SPAM architecture as a set mutually exclusive methods and only the method is directly specified during knowledge acquisition. In the following section we describe the restructuring of the SPAM organization necessary in order to represent these kinds of knowledge.

## 2.2. Schematization of SPAM

In order to to make SPAM amenable to knowledge acquisition our approach has been to reduce the SPAM architecture to a set of generic control productions supported by scene-specific knowledge that can easily be generated by a program. Experimentation with the system architecture is now straightforward since the actual production generation is centralized in one program. Each piece of knowledge is encoded as a schema, with different schemata used to represent different types of knowledge. Schemas can easily be collected (or partitioned) to form new knowledge bases. Since the schemas are simply text files, it is trivial to combine different schemata to produce more complete knowledge bases. A discussion of this representation can be found in Section 4, a detailed description of the internals of schemata is found in Appendix I, and examples of the generation of productions from a schema is illustrated in Appendix II.

This implementation has restructured SPAM such that within any interpretation phase, no rule has to know of the existence of any other rule. These intra-phase interactions were difficult to identify in the hand generated system, and made it very difficult to perform large wholesale changes to the knowledge base. Since there is a uniform interface to all rules within a particular phase, it is easier to allow users to specify interphase events such as calling consistency-checking within model-generation phase. The functional-area phase is an example of one part of the system that required some generalization for use on other domains. Originally developed with airports in mind, functional-areas had no shape constraints. However we have found cases in our suburban house scene experiment where shape constraints in addition to compatibility constraints are required. RULEGEN gives us the opportunity to easily propagate these changes to the different systems we have built. In the following section we briefly describe the ISCAN/RULEGEN/SPATS system organization.

## 3. Tools For Knowledge Acquisition In SPAM

There are several reasons why knowledge acquisition for SPAM appears to be relatively straight-forward. First, SPAM uses simple, pairwise tests to represent spatial consistency. Second, it is ordinarily easy for humans to characterize situations where special knowledge, either derived from further image analysis, or from additional consistency testing, can be used to disambiguate conflicting hypotheses. For example, if the two hypotheses differ in that one suggests there is an object above the ground plane, e.g.. a hangar or building, and the other is at the ground plane, a runway or road, then invoke an image analysis tool, stereo verification, to determine the preferred hypothesis.

The implication of the first observation is that the user is not forced by the architecture to conceptualize complex spatial consistency rules encompassing many primitives. For example, SPAM represents, *runways intersect taxiways that are oriented towards the tarmac* as two independent tests. One is an geometric intersection test between runway and taxiway fragment hypotheses, the other is an orientation test between taxiways and tarmac hypotheses. This is done to accommodate errorful image segmentation data which may not produce all of the primitives required for a more complex match. Therefore, partial matching on pairwise primitives is preferred since at least pairwise consistency can be recognized and propagated as necessary. A second, more pragmatic reason, is the desire to not require complex matching of productions in our implementation language, OPS5[10, 11]. The second observation is a function of the task domain, the available image analysis tools, and our design of the SPAM architecture. A small set of several dozen geometric tests appear to suffice to represent the spatial relations that human users characterize as important for describing relationships between scene domain objects. Finally, we believe that it is possible to find images for a class of scenes, say, 20 commercial airports, which would allow us to acquire a cross-section of spatial relationships representative of commercial airports. This approach also lends itself toward exploring systems that would automatically synthesize interesting properties and learn the importance of various spatial relationships.

Knowledge acquisition systems range from interactive user dialogue via structure editors to acquisition systems that are tightly coupled with a task performance system. The degree to which the knowledge acquisition system itself utilizes knowledge may range from enforcing a particular knowledge representation, to a system which decides what to ask a user and asks for as little information as necessary to remedy specific problems[12, 13, 14]. ISCAN falls somewhere in this continuum, toward the former method. It primarily enforces a particular schema representation for various types of knowledge utilized by SPAM. However, it also uses knowledge of the SPAM architecture to recognize conflicts and missing or incomplete information. But it performs as an observer and does not elicit or suggest remedies. It is also decoupled from the performance system, partly do to the long execution times of SPAM[1], and partly due to what we believe is a complex task domain

which makes credit assignment for particular actions of the system difficult to analyze.

In the remainder of this section we describe our first attempt at knowledge acquisition for aerial image interpretation using the ISCAN user interface, the RULEGEN compiler, and the SPATS performance analysis tool. Figure 3-1 shows a detailed organization of the knowledge acquisition system overview presented in Figure1-1.

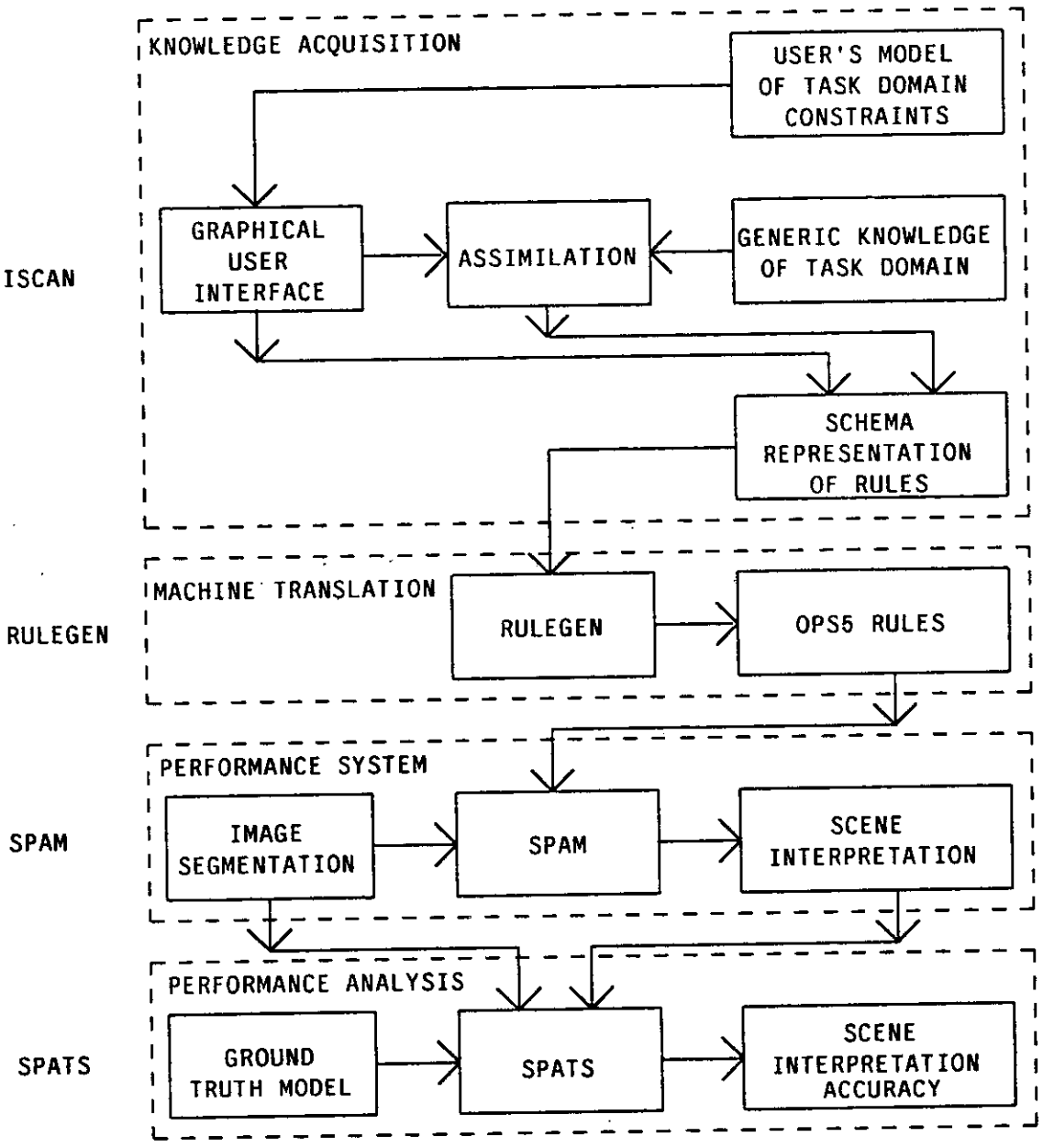



**Figure 3-1:** Knowledge Acquisition For SPAM

### 3.1. The ISCAN User Interface

ISCAN currently supports two methods of knowledge acquisition. The first method is that of a structured editor which allows users to add, delete, or modify knowledge represented as schemata. The second method is the use of interactive image segmentation to generate metric information such as area, perimeter, distance, and shape descriptions. This information is then integrated into the schemata as values or ranges of values for various constraints. In either case the output of ISCAN session is a file containing the task specific schemata necessary to compile a SPAM system.

As a structured editor ISCAN allows the addition, deletion, and modification of schemata for each phase in SPAM. It assists the novice user, asking questions to define the set of attributes for each schema and allowing example schemata to be displayed. ISCAN accommodates the more experienced user by foregoing the question/answer sessions and permitting the attributes to be entered directly. It maintains certain specific meta-knowledge such as knowing which attributes of a region must be computed and which can simply be matched. Much of the bookkeeping specific to the SPAM architecture is automated, thereby allowing the novice to concentrate on the task domain and not on whether attributes are filled in correctly. For example, some region attributes are precomputed, but some are too expensive to precompute for every region. It is really an implementation detail to know which attributes must be computed and which are precomputed.

Because the nature of the interpretation task is visual, ISCAN also provides a graphical interface for defining spatial relationship and performing measurements directly on an image. The user displays a representative image containing classes of objects or a particular site such as an airport. Associated with each image is a camera model[15, 16] that allows the graphics interface to generate constraints in terms of metric values rather than in image specific coordinates. For example a representative road width constraint can be specified as *between 10 and 15 meters* rather than *between 10 and 15 pixels*. The actual measurement is performed by ISCAN and is reported to the user. ISCAN can gather statistics over many examples to allow for a more robust range of constraints.

Since the measurements are always in terms of ground distances this allows for complete independence between the scale of the image under interpretation and the acquired knowledge base. This independence is a basic requirement for robust scene interpretation systems. With the scene constraints in mind, the user displays an image with characteristics of the general type of scene that is to be interpreted. After making measurements on the image directly, the user is questioned about the classes of objects in the scene, the shape characteristics of those objects, and their spatial relationships to one another. If this is a scene type that has previously been analyzed, it is possible that generic knowledge applicable to that scene can be applied. This can be added to the knowledge

base being built. An example of this from the airport domain might be that every airport has at least one runway. This generic knowledge, if any, is coalesced with the knowledge about the scene given by the user.

ISCAN recognizes potential inconsistencies that may be generated during an interactive session or those that exist at the end of the session. It provides limited help in correcting such problems, a topic for future work. Some of kinds of inconsistencies recognized include:

- The inconsistent definition of class and subclass fragment interpretation and the violation of class hierarchies.
- The lack of a local-consistency schema for a class or subclass fragment interpretation.
- Multiple local-consistency schemata with identical fragment interpretations which potentially can generate inconsistent constraints.
- The omission of a subclass fragment interpretation from all functional areas.
- The definition of functional area descriptions and recognition of inconsistent combinations of component fragment interpretations.
- The specification of conflict resolution tests must be unique.
- The definition of model generation components that do not specify a method for selection.

We feel that the use of representative imagery to acquire general spatial relationships greatly increases a users ability to add such constraints to SPAM. This is primarily due to the ability to query ISCAN to display existing constraints involving fragment hypotheses and to detect conflicts or duplication than in a textual environment. Because the nature of our task is inherently visual, and visual tasks are done almost effortlessly by people, it appears that an it is easier for a person to give examples than to explain what is being extracted.

However, an area for future research for automating knowledge acquisition beyond user interaction is via learning by example. As we have discussed, our current work is focused on tools that aid in the translation of a users model of the task constraints into schemata. There are other sources of spatial knowledge that are amenable to automated extraction of constraints without user involvement. Figure 3-2 shows hand segmentations generated for use in performance analysis as *ground truth* data for Dulles International and Andrews AFB. Figure 3-3 illustrates a similar type of ground truth data, but perhaps not as detailed as the hand segmentations. It is generally available to aircraft pilots as Flight Information Publications, or FLIPcharts[5], published by the FAA and the Defense Mapping Agency. The goal of such research is to uncover spatial constraints by examining a large number of examples of airports whose spatial relationships are made explicit in either of

these formats. Such a system must not only develop reasonable ranges of values such as *'airports have at least one runway, but no more than 7'* but must develop the subset of useful spatial relationships from the set of all possible relationships. We plan to explore this approach as a method to expand the scope of knowledge acquisition in ISCAN.

### 3.2. The RULEGEN Compiler

With the scene knowledge encoded as schemata, we need to put it into a form that can be utilized by our interpretation system. RULEGEN is a compiler which performs schema-to-production translation. This compiler has procedural knowledge of the SPAM control structure. Some of the functions the compiler must perform include:

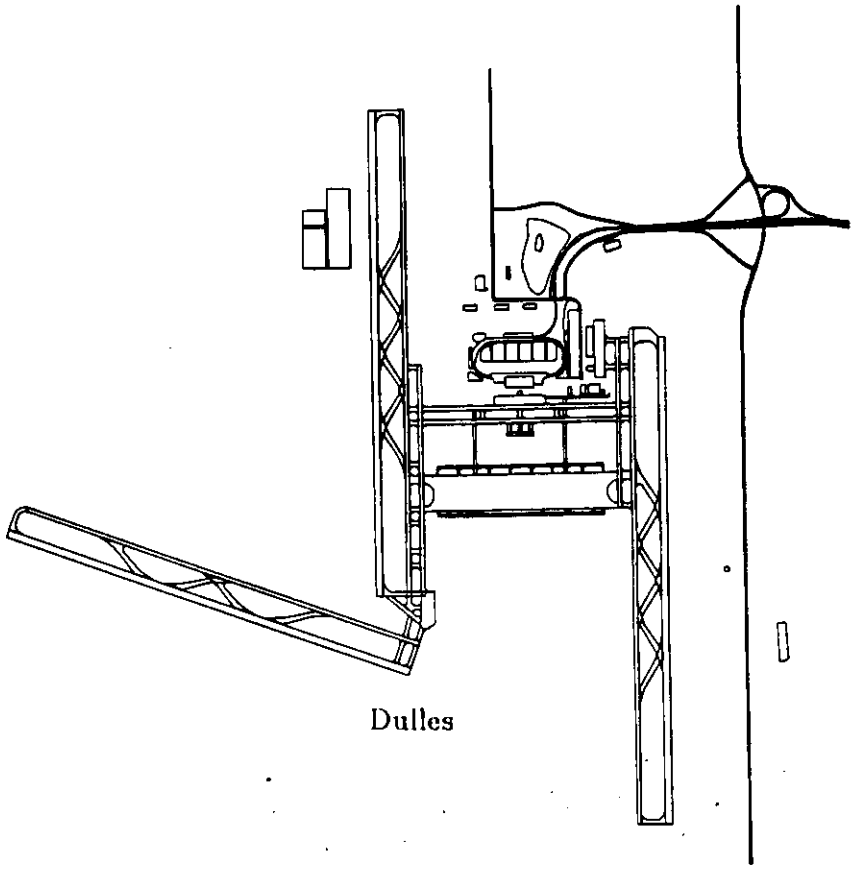
- Efficiently initializing each rule so that large conflict-sets do not slow down the OPS5 conflict resolution process.
- The automatic generation of error-checking productions to make the system more robust, and trace productions for performance analysis.
- Managing control productions to efficiently match all the desired data in working-memory.
- The generation of interface functions for computations such as image analysis and geometric computation performed outside of OPS5 environment.
- The generation of data-structures representing the boundary values of the scene constraints.

In the SPAM architecture, region interpretations come mostly bottom-up, with top-down prediction and verification. With processing going in two directions, the management of control in a production system is non-trivial. RULEGEN handles the rule interactions and the order of rule firings by generating appropriate control rules to achieve the desired results. The data-structures and control productions vary from phase-to-phase because the type of processing that occurs in each phase is very different. Appendix II gives some detailed examples of the actual expansion of a schema into a collection of productions executable by SPAM.

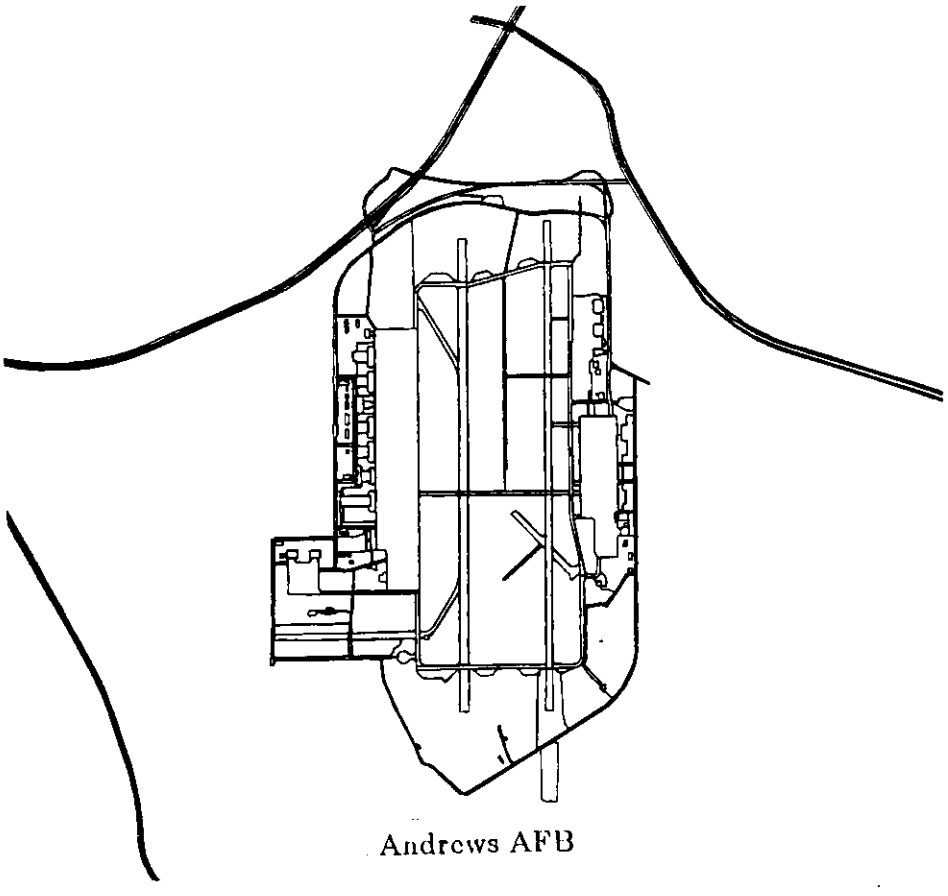
### 3.3. SPATS: Automating Performance Analysis

An often overlooked component of any interpretation system are tools that aid in incremental refinement of knowledge and in the measurement of the effects of various types of knowledge within the performance system. As a part of our work in knowledge acquisition we have developed a performance analysis program, SPATS, that gives us some insight into the overall accuracy of the scene interpretation. SPATS uses a region-based hand segmentation with correct interpretation attributes associated with each region as a baseline with which to compare the SPAM interpretation. In the case of machine-segmented data, we compute region overlap with hand-segmented data in order to generate a correct interpretation. In some cases, ambiguous results must be resolved

**Figure 3-2:** Ground Truth Segmentations For Dulles and Andrews AFB

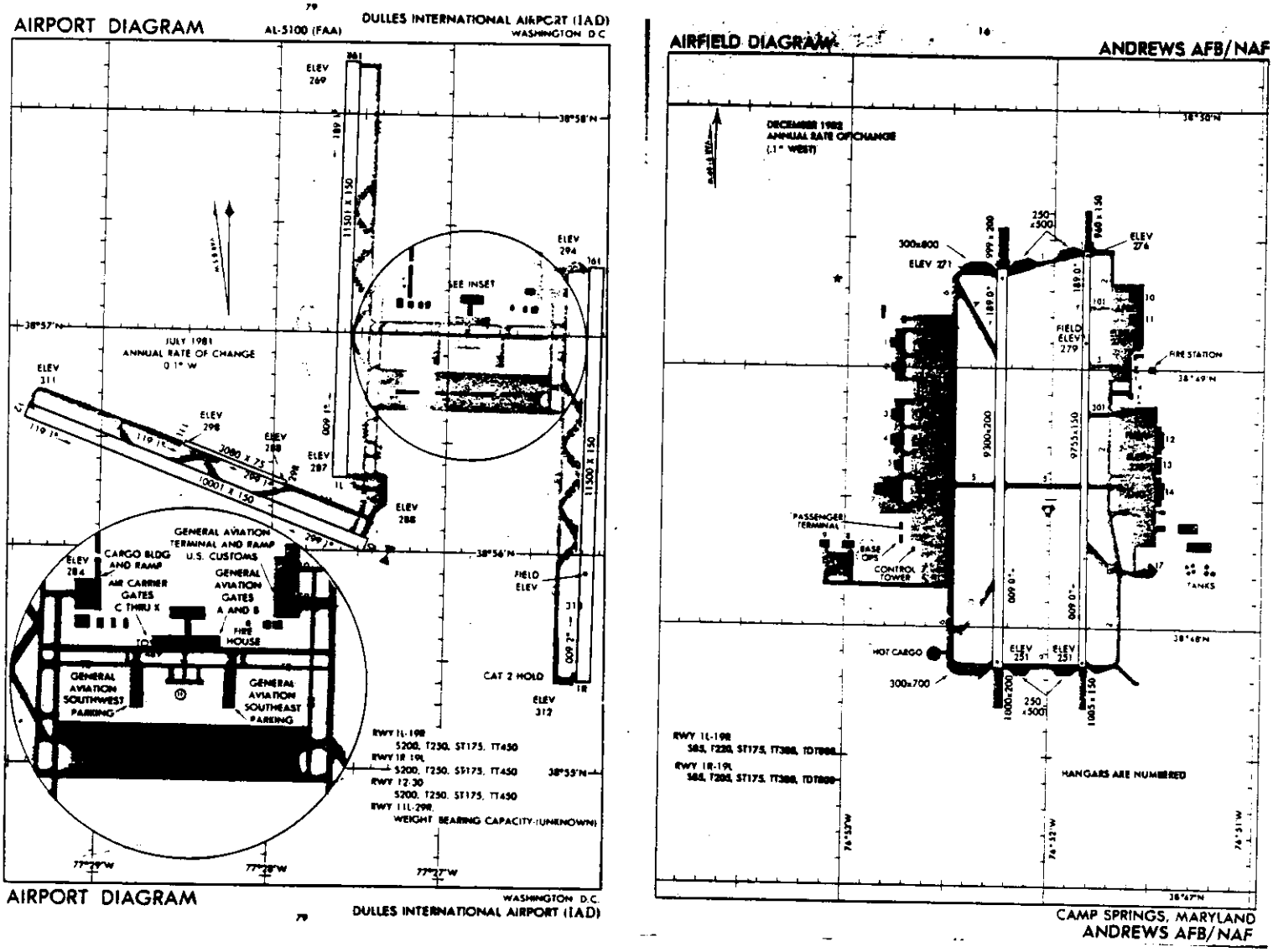


Dulles



Andrews AFB

**Figure 3-3:** Flight Information Charts With Airport Layouts For Dulles and Andrews AFB



manually before statistics can be generated. A log file generated by SPAM at each phase of interpretation is used to acquire the internal state of the SPAM interpretation. At a gross level we need a consistent method to measure the accuracy of scene interpretations generated with alternative or refined knowledge. In terms of the 'debugging' of knowledge, we require an indication of where one might spend time improving the knowledge base to improve scene interpretation. SPATS attempts to summarize the important performance statistics in a succinct manner for each phase of processing by SPAM.

For the *region-to-fragment* and *local-consistency* phases, we require statistical measures that accurately reflect the performance of geometric knowledge in classifying the initial image segmentation. Factors such as the number of competing hypotheses, as well as the number of correct and incorrect hypotheses, are most useful. SPATS provides this information in tabular form (see Appendix III). This information can be compared within, as well as across class and subclass boundaries, to give some indication of the effectiveness of the geometric constraints. One measure, the correct branching factor, defines how many interpretations there were, on the average, for each correct interpretation. This branching factor increases from zero (with zero signifying no correct interpretations) as the number of competing hypotheses increases. As this number increases, the effectiveness of the associated geometric knowledge decreases. To rectify this problem, we would then try to isolate which class or subclass constraints were too weak and attempt to tighten them, or look for new sources of knowledge that would increase our ability to discriminate.

For the *functional-area* phase, SPATS checks the integrity of the functional-areas generated by SPAM. This involves using the functional area declarative knowledge to check that all the fragment interpretations fit the definition of that functional-area type. Statistics giving the correctness of each functional-area, the number of compatible and incompatible fragment interpretations contained within the boundary of the functional area, but not found to be components of the functional area, are generated. This gives us a measure of the cohesiveness of the functional area. One would expect small numbers of incompatibles to be present, with some compatibles. If a large number of compatible fragment interpretations are present questions about why they did not participate as members of the functional area can lead to the modification of geometric consistency rules or the recognition that knowledge of new relationships should be sought. Finally, these mismatches between functional area definitions and geometric consistency can indicate whether the user's definition of a functional-area is appropriate.

For *model-generation* phase, the constituent functional-areas of the various scene models are compared. This is done as a first attempt at quantifying the differences between each of the scene models, if more that one consistent model is generated by SPAM. Currently a complete analysis involving the accuracies of the included interpretations has

not yet been completed in SPATS.

As a general methodology for the evaluation of a knowledge base we have found that running SPAM on hand-segmented ground truth region data is a valuable test of the interpretation process. It presents to SPAM a "perfect" low-level segmentation, effectively decoupling the low-level image analysis. The results from this type of experiment can be used to argue the issue of whether the interpretation problem is fundamentally one of dealing with errorful segmentations and should be remedied by working to improve the segmentation. Even a 'good' low-level segmentation requires significant high-level knowledge in order to generate a scene interpretation. The use of ground truth data also makes it easier to avoid a common problem exhibited by computer vision systems of unknowingly developing intermediate and high-level vision components which rely on machine segmentations that can be characterized as over-segmented or under-segmented. In our view, one should at least make explicit these assumptions if they are a factor in the interpretation process.

SPATS is a useful tool for indicating gaps, weaknesses, or inconsistencies in various types of knowledge in SPAM. A statistical approach must be used due to the large number of segmentations involved in the interpretation process, as well as the inaccuracies in assigning interpretations to those segmentations. Future research is being focused in the refinement and addition of new measures, particularly in *model-generation* phase, and looking at techniques for making performance analysis a more active component of SPAM.

In the following Section we give some detailed examples of the schema-based knowledge representation generated by ISCAN and used by RULEGEN to compile SPAM systems. Section 5 describes the use of ISCAN/RULEGEN to generate a SPAM system for a new suburban house scene task.

## 4. A Schema-Based Knowledge Representation

Our schema-based knowledge representation is one method for linking knowledge acquisition as performed in ISCAN with knowledge utilization in SPAM. The focus of this work was to develop an intermediate representation for the domain specific knowledge used by SPAM as a target description for knowledge acquisition and as a source description for automatic generation of the interpretation system. Therefore, some important properties for the representation are as follows:

- Sufficiently general to represent the kinds of knowledge and spatial relationships utilized in each of the SPAM interpretation phases.
- Could be compiled into our target production system language, OPS5.
- Easily organized or partitioned into independent knowledge sets.
- The format is understandable by non-programmers. The knowledge and its

purpose should not be obscured by the implementation language.

The schema-based representation has been conducive to experimentation. It is far easier to add and delete knowledge and to measure the effect on system performance, as the rule generation is performed automatically; only the control productions that embody the SPAM architecture are hand crafted. Improvements in structuring rules are easy to propagate. If an improved method is developed, the generating functions in RULEGEN are modified appropriately and all generated productions are updated. Since our goal is to produce a working system that handles a variety of aerial interpretation tasks it is easier to test the generality of the SPAM architecture if we can generate task specific systems. Hand generation of SPAM is not an attractive alternative, especially in light of our requirement to perform experimentation by adding and removing specific types of domain knowledge. In the following examples of schemata for various interpretation phases it should be emphasized that users do not directly edit this textual representation. ISCAN generates and enforces the schemata syntax, designed to simplify parsing by RULEGEN, during user interactions.

### 4.1. A House Fragment Rule

The following is an excerpt from the RULEGEN schema file for the *region-to-fragment* phase of SPAM, used to interpret suburban-housing scenes. This set of attribute-value pairs will generate data-structures and productions allowing fragment interpretations for regions of type 'house' to occur.

```
'CLASS'               = 'house'
'REGION-DEPENDENCES'  = ''
'FRAG-DEPENDENCES'    = 'object-type compact && hypothesis unknown'
'SHAPE-CONSTRAINT'    = 'area && 50.00 <= value <= 150.00'
'SHAPE-CONSTRAINT'    = 'ellipse-length && 12.00 <= value <= 18.00'
'SHAPE-CONSTRAINT'    = 'ellipse-width && 10.00 <= value <= 20.00'
'SHAPE-CONSTRAINT'    = 'ellipse-linearity && 0.00 <= value <= 3.50'
```

Each of the schema attributes are described below.

CLASS

Determines the class of object to which this set of constraints is applicable. In this case, the rule defined will apply to houses.

REGION-DEPENDENCES

Makes sure that a given set of attributes have been computed *before* any house interpretation rules can be fired. In this case, there are no computed attributes that must exist before this rule can fire.

FRAG-DEPENDENCES

Allows a constraint rule to depend on the success of a previous constraint rule. In this case, the constraint rule for the object type "compact" must have previously executed successfully (e.g. a compact interpretation created) in order for this rule to

fire.

SHAPE-CONSTRAINT

Defines the actual constraints that comprise the class definition. Any number of these constraints can occur here. Currently, 2-D shape characteristics, intensity characteristics, depth measures, and texture measures fall into this category. In this case, four constraints completely define the class-type house. For example, the first constraint limits houses to have areas between approximately 50 and 150 square meters.

The first three attributes generate a small set of control productions that determine if this rule applies to a given region. Each SHAPE-CONSTRAINT generates a single production for the particular constraint given.

When this rule becomes applicable, an initialization production fires and creates a subtask which is matched by each of the constraint productions. All of these productions are allowed to fire, each determining the "goodness" of the match for a single constraint. Finally, a domain-independent production looks at the accumulated scores and decides whether a house interpretation should be made. Appendix I gives a detailed description of the each schema-type and spatial constraint currently available in ISCAN. Appendix II contains the actual productions generated by RULEGEN for each of the schemata in Section 4.1 and Section 4.2.

### 4.2. A House-Road Consistency Rule

For the second phase of SPAM, local-consistency-check, RULEGEN uses a new set of attributes to define a rule. The basic idea, as discussed in Section 2, is to generate pairwise tests that exploit the fact that although there may be a large number of errorful fragment hypotheses, only small numbers will be mutually consistent. The following is one such local consistency test, *houses-are-parallel-to-roads*, others might include, proximity of houses to each other, distance from roads, orientation of a house to a driveway, etc.

```
'RULENAME'    = 'houses-are-parallel-to-roads'
'CONFIDENCE'  = '0.8'
'HYPOTHESES'  = 'house && road'
'GEOMETRICS'  = 'orientation'
'SUBTYPES'    = 'parallel'
'BOUNDS'      = '0.00 <= value <= 0.50'
```

RULENAME

Attaches a unique, human-readable name to the rule, so we know what it does. In this case, this rule will determine whether a road is parallel to a house.

CONFIDENCE

Assigns a confidence value to this rule, which describes its discrimination ability. It is a number between 0 and 1, with a value of 1 implying that the rule can perfectly

(uniquely) determine that the participating interpretations are correct. In this case, the rule is believed to be a good characterization of one spatial relationship between houses and roads in suburban house developments and is given a confidence value of 0.8.

HYPOTHESES

Defines the classes of interpretations to which this rule applies. In this case, the rule applies to the relationship between houses and roads.

GEOMETRICS

Translates the high-level spatial relation into a low-level geometric test. In this case, "parallel" is translated as "orientation".

SUBTYPES

Further defines the translation of the high-level spatial relation. In this case, "parallel" implies that the orientation of the interpretations is being tested (see the previous description), and that the particular type of orientation test should be "parallel".

BOUNDS

Completes the definition of the high-level spatial relation by defining the error tolerance. In this case, the geometric test has a tolerance of 0.5 radians.

The first three attributes define the high-level significance of this rule. The last three attributes describe, in low-level terms, the high-level intentions. For this example, the rule may be read as *houses being parallel to roads* means for a particular house, a road must be oriented parallel to that house, within a tolerance of 0.5 radians. Thus, a house fragment hypothesis and a road fragment hypothesis will support each other if this test is successful.

### 4.3. A House Functional-Area Definition Rule

The functional-area phase of SPAM groups individual hypotheses into mutually supporting collections of hypotheses that represent meaningful sub-parts of the overall scene model. The knowledge in this phase defines these scene sub-parts.

```
'FA-NAME'       = 'house-area'
'SEED-REGION'   = 'house'
'DEFINITION'    = 'driveway && grassy-area'
```

FA-NAME

Assigns a name to this functional-area definition.

SEED-REGION

Defines the principle hypothesis of a functional-area. If this type of hypothesis does not exist, no functional-area of this type can exist. Here, we designate the interpretation type 'house' as our seed region.

DEFINITION

Enumerates the possible constituents of this type of functional-area. The functional-area 'house-area' can contain only houses, driveways, and grassy-areas.

### 4.4. A Suburban-Scene Model Rule

For the final phase of processing, model-generation, SPAM combines functional-areas together to form models of the entire scene. During this process, conflicting interpretations will be identified and must be resolved in order to obtain a final, consistent model. Knowledge about the types of conflicts, and ways to disambiguate them, is encoded in this phase.

```
'CONFLICT'      = 'house && driveway'
'RESOLUTION'    = 'function && stereo'
```

CONFLICT

This attribute specifies the name of the conflict type that needs special attention in order to be disambiguated. In the example at hand, house-driveway conflicts will be specifically addressed.

RESOLUTION

Defines the type of process to use to do the disambiguation. In this case, we know that houses have height and driveways do not, therefore invoke a stereo process to determine whether or not the region has height.

Those conflicts not enumerated are handled by a default resolution strategy that takes into account the confidences of the individual interpretations, as well as the amount of support for each interpretation in the context of the current scene model.

## 5. A New Task Domain For SPAM

In this section we will give a brief example of one of the interpretation tasks used in our experiments to date. We show some results of the interpretation of a suburban house scene by SPAM built completely using the ISCAN/RULEGEN system. Figure 5-1 is a photograph of three of the suburban house scene images used by Hwang[17] at the University of Maryland. Our intent was to replicate this work using the ISCAN/RULEGEN system to generate a SPAM with spatial knowledge of suburban house scenes. Figures 5-2 and 5-3 show a human segmentation and machine segmentation of one of the six suburban house scenes used in this experiment. This replicates work performed by Hwang[17] on this image set. The goal is to segment and identify the houses, roads, grassy areas, and driveways in the aerial image. This is a somewhat simpler task that the original airport scene interpretation task performed by SPAM, but it turned out to be of reasonable scale to evaluate and refine ISCAN/RULEGEN.

The SPAM knowledge base for these images were developed with measurements using ISCAN on two training images from the set. The knowledge base was refined iteratively, first using the hand segmented set of regions, then modified using machine segmentation data. Currently, SPAM has been run on using both hand and machine segmentations for three of the six images, the two training and one test image. The image in this example is

the test image.

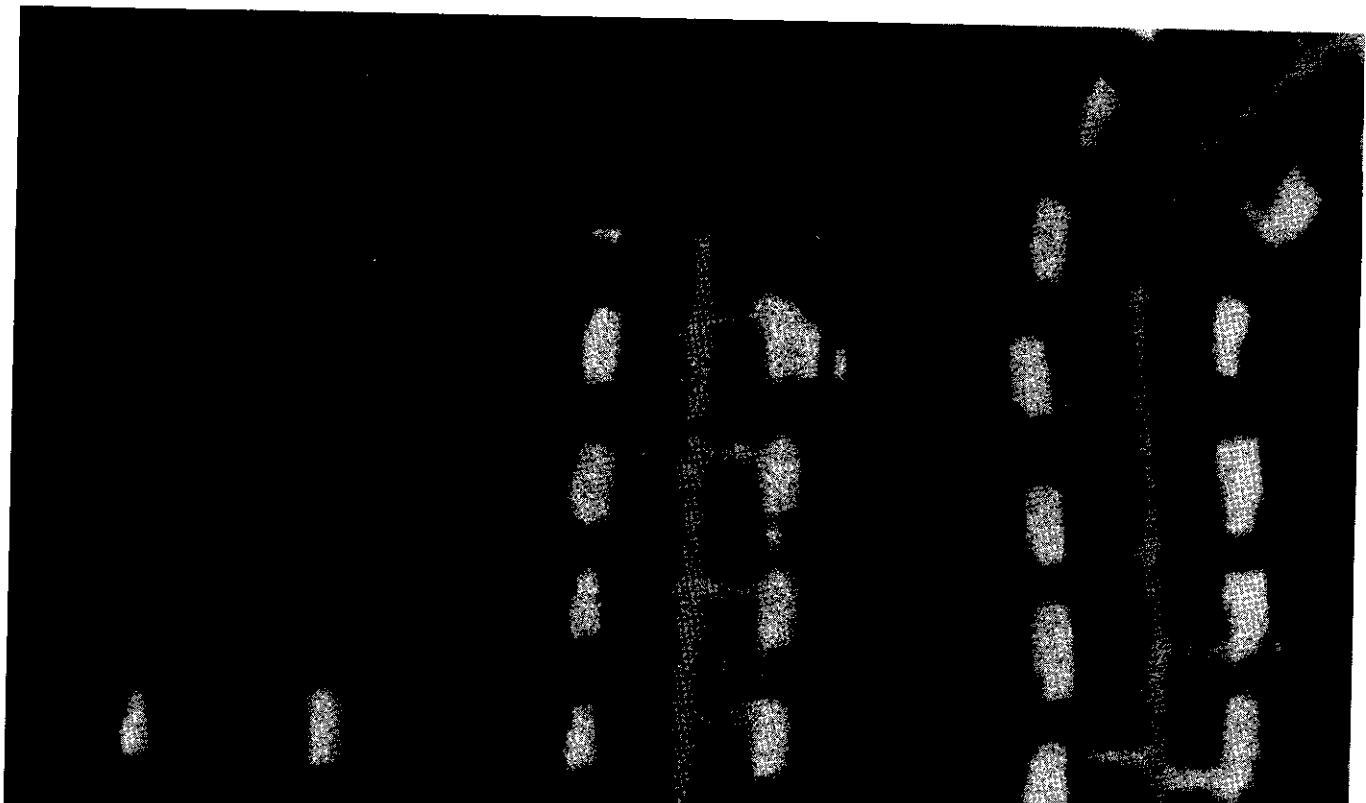
The purpose of the hand segmentation is to provide "ground truth" for our automated analysis programs that are used to generate region-by-region interpretation statistics used to measure SPAM's performance in region labeling and overall scene interpretation. We have also found it useful to run the hand segmentations through SPAM in early phases of rule development in order to completely decouple the low-level image analysis from the interpretation system. This noise-free approach allows us to uncover gross omissions or unexpected interactions between the local consistency rules. Figure 5-3 is the result of running our image segmentation system, MACHINESEG[18], which uses region-growing and shape extraction simultaneously to look for characteristic linear, compact, and blob regions. Although the image is relatively uncomplicated several houses are missed, some are only partially segmented, and the roads and driveways are oversegmented into multiple pieces. However, this is reasonable in the context of current computer vision segmentation capability. Figures 5-4 and 5-5 show functional areas generated by SPAM for houses and roads, respectively. Figure 5-4 shows the functional area generated from the hand segmentation in Figure 5-2 including regions whose fragment interpretation were 'house' or 'grassy area' Figure 5-5 shows the functional area including 'roads' and 'driveway' hypotheses for the machine segmentation in Figure 5-3. We feel that the functional areas are quite good good in both cases and are similar to results generated by Hwang[17]. While direct comparisons of two knowledge-based systems using different methodologies are not the subject of this paper, it is important to point out that these results were generated by automatic compilation of user-defined knowledge tailored to the suburb house scene task within the framework of the SPAM interpretation architecture.

### 5.1. Structural Differences In Hand versus Machine Generation

One goal for RULEGEN was to be able to reproduce the hand crafted version of SPAM reported on in[1, 2] for airport scenes. This system, which we will call SPAM-1, contained over 500 hand-crafted OPS5 productions, and was used to interpret airport scenes of National Airport, Los Angeles International and NASA AMES Moffett Field. In contrast SPAM-2 was built with the RULEGEN compiler by manually extracting the primitives and constraints from SPAM-1 and encoding them as schemata. SPAM-2 was verified on the same airports as SPAM-1 giving quite similar results. It has since been used on other airports, not tested with SPAM-1, such as Dulles International, Andrews Air Force Base, and San Francisco, with mixed results. SPAM-2 is now the basis for future work in airport scene analysis. SPAM-3 is the suburban house scene system and was built entirely using ISCAN and RULEGEN.

Figure 5-6 gives a breakdown of productions comprising each of the SPAM interpretation phases for each of the three systems. With this data, we will try to characterize some of the differences between SPAM-1 and SPAM-2 and characterize the emergence of domain

**Figure 5-1:** Suburban House Scene Imagery



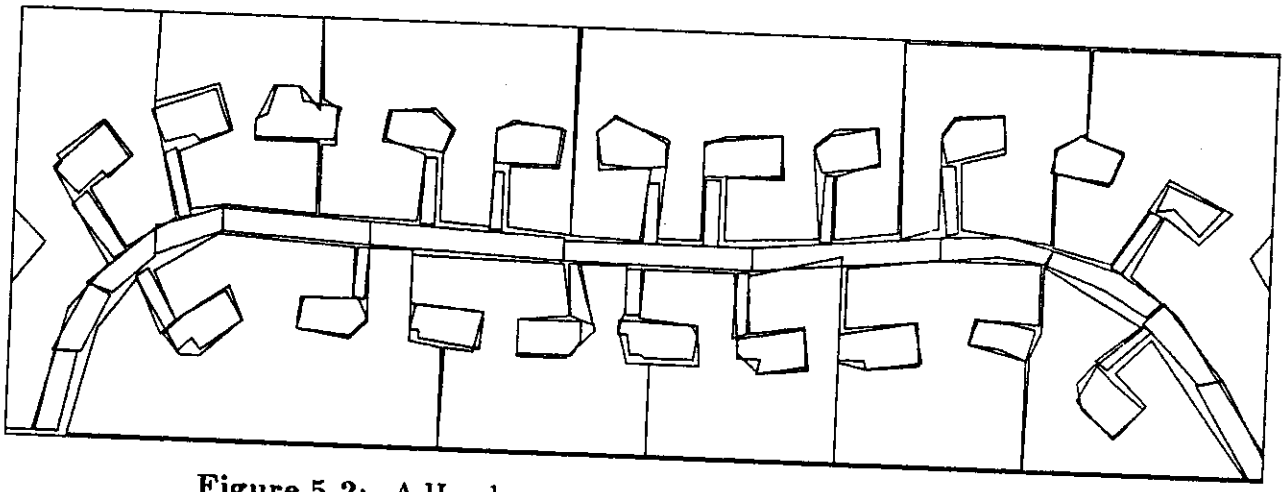


**Figure 5-2:** A Hand segmentation of a suburban scene

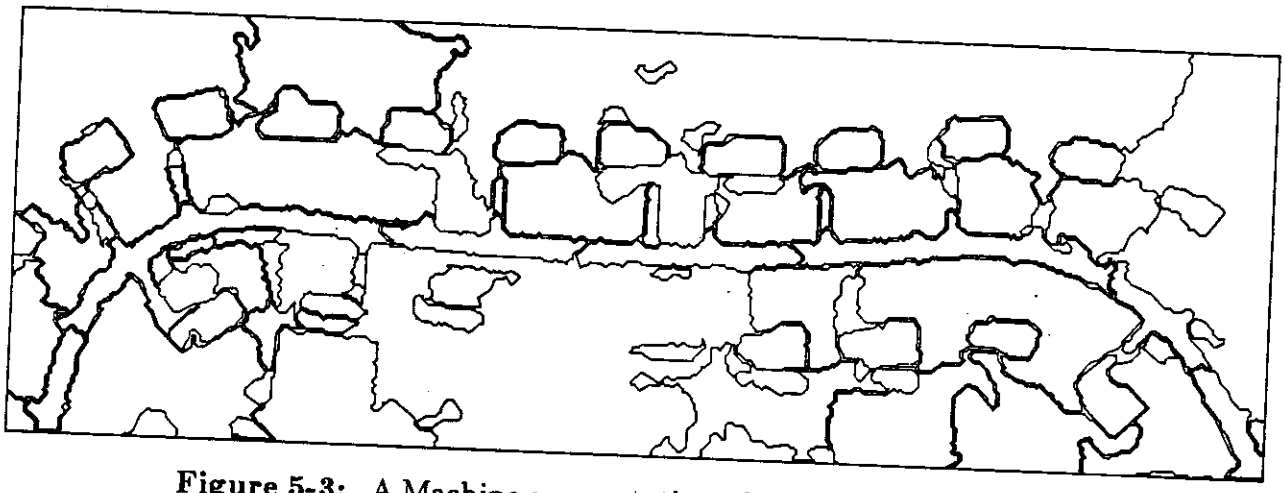


**Figure 5-3:** A Machine segmentation of a suburban scene

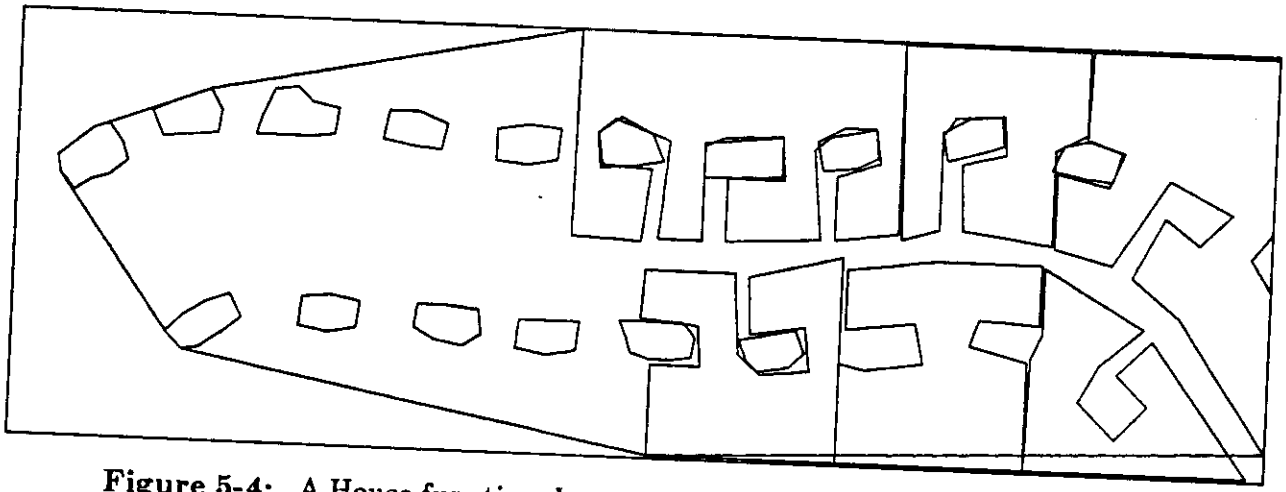


**Figure 5-4:** A House functional-area result from hand segmentation

independent knowledge as a result of the restructuring of the SPAM architecture as

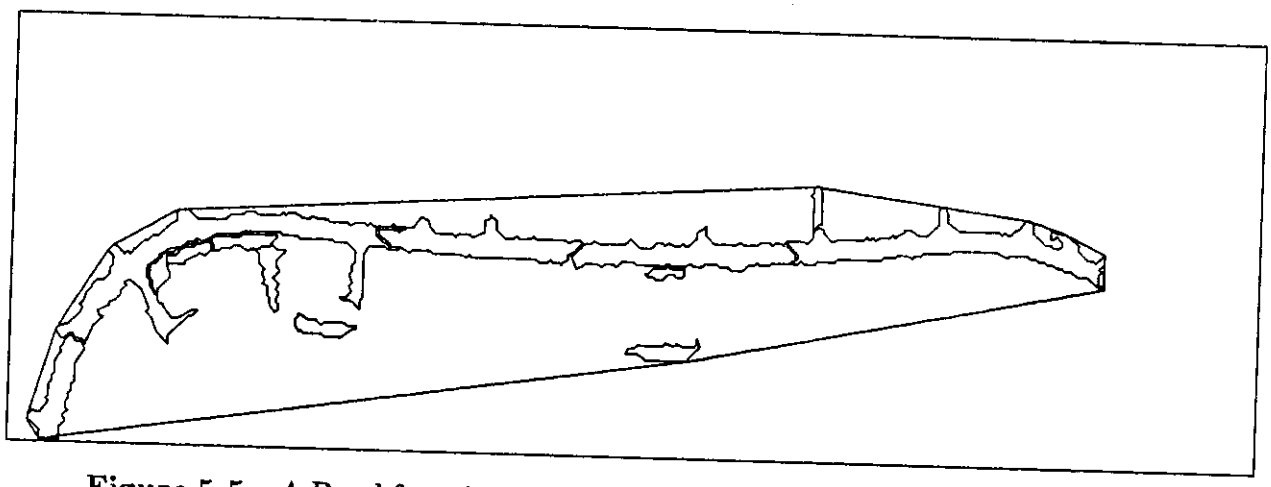


**Figure 5-5:** A Road functional-area result from machine segmentation

| Task | Interpretation Phases | | | | |
|---|---|---|---|---|---|
| | RTF | LCC | FA | MG | MISC |
| (SPAM-1) Airport (Hand) | 120 | 304 | 23 | 50 | 16 |
| (SPAM-2) Airport (Rulegen) | 54 | 297 | 1 | 6 | 0 |
| (SPAM-3) Suburb (Rulegen) | 32 | 99 | 1 | 1 | 0 |
| Rulegen Domain Independent | 37 | 7 | 11 | 35 | 7 |

RTF: region-to-fragment phase
LCC: local-consistency check phase
FA: functional area phase
MG: model generation phase
MISC: miscellaneous interphase control

**Figure 5-6:** Rules generated by Interpretation Phase

described in Sections 2.2 and 4. To do a proper comparison of SPAM-1 to SPAM-2, one must add the number of domain-independent productions to the number of generated productions for SPAM-2. For example, if we do the comparison for the *region-to-fragment* phase (RTF), we find that there are 120 productions in the hand-crafted system and 91 (54 domain + 37 domain independent) productions in the machine generated system.

The decrease in the number of productions is somewhat due to the experience gained in during the hand-coding of SPAM-1[11] applied to RULEGEN. In addition, the desire to generalize the SPAM architecture forced us to consider how to gain efficiency as well as generality. The decoupling of domain-dependent knowledge from the SPAM control rules actually lead to a decrease in the number of OPS5 productions being generated. In the case of the last two phases, *functional-area* and *model-generation*, it is clear that most of the knowledge is now encoded by the domain-independent rules or migrated to procedural knowledge. This is due to the more abstract functions provided by these phases, such as grouping, merging, and splitting which appear to be task independent and are not

AIRPORT SCENE TASK

| Classes | Subclasses |
|---|---|
| linear | runway, taxiway, road |
| compact | hangar-building, terminal-building |
| small-blob | parking-apron, parking-lot |
| large-blob | grassy-area, tarmac |

| Functional-Area Types | Definition |
|---|---|
| runway | taxiway, grassy-area, tarmac |
| road | grassy-area |
| hangar-building | parking-apron, tarmac, road |
| terminal-building | parking-apron, parking-lot, road |

SUBURBAN-HOUSING SCENE TASK

| Classes | Subclasses |
|---|---|
| linear | driveway, road |
| compact | house |
| blob | grassy-area |

| Functional-Area Types | Definition |
|---|---|
| road | driveway |
| house | driveway, grassy-area |

**Figure 5-7:** Class, Subclass, and Functional Area Definitions For Both Tasks

knowledge intensive. Thus the bulk of the domain knowledge appears to be in the *region-to-fragment* and *local consistency* phases. Once fragment interpretations are generated along with associated chains of consistent relationships the aggregation of these fragments into functional areas is now mostly procedural. What knowledge remains is the definition of the functional area groups, model definitions, and methods to resolve conflicts. The net result is a more general system with fewer productions and, though not explicit from this data, faster execution times.

For the suburban-house scene task, the amount of knowledge required appears to be significantly less than for the airport task. This is not surprising since the number of productions in the *region-to-fragment* and *local-consistency* phase is directly related to the number of geometric and spatial constraints used to interpret the scene. Figure 5-7 gives a comparison of the two tasks in terms of the number of interpretation classes, subclasses and functional areas. A simpler scene type intuitively implies that fewer scene primitives are present and that a smaller number of spatial constraints are available. One would expect, therefore, that the amount of knowledge required to interpret the less

complex scenes would decrease from that required for the more complex ones. This is exactly the case for comparisons of SPAM-2 and SPAM-3. It should be noted that we currently have preliminary SPAM systems configured with larger numbers of subclasses, primarily to increase the number of types of buildings, taxiways, and roads, in order to remedy problems faced in the interpretation of additional airports by SPAM-2. Thus, the amount of knowledge required for more general airport interpretation, and the difference between tasks, may be significantly larger than is indicated by these comparisons. It is unlikely that we will have to make similar increases for other suburban house scenes. Even with this disparity in the absolute amount of knowledge, the preponderance of knowledge in both tasks, as reflected in the number of productions, appears to be in the first two interpretation phases regardless of inherent task complexity. A more precise characterization of the amount of knowledge needed to interpret a particular scene type, related to a measure of apparent task complexity, would be an interesting result from this work. This may become possible as more task domains are implemented within the SPAM architecture.

## 6. Conclusion

The SPAM project is an experiment in the use of large amounts of knowledge in aerial image interpretation. To do significant exploration requires tools for knowledge acquisition. In this paper, we have described a collection of tools for knowledge acquisition, automated compilation of knowledge, and performance analysis. Several types of knowledge that are required for aerial image interpretation systems are described. The use of knowledge in SPAM and its representation as schemata for knowledge acquisition and compilation is discussed. The results of a completely automated generation of a SPAM system for a new task domain are described and show the generality of the knowledge acquisition tools. Some preliminary analysis of the effects of decoupling domain-independent knowledge from the interpretation system are presented,

In summary, by focusing on automated knowledge acquisition and compilation we have generated a more manageable interpretation system for experimentation and measurement. This flexibility gives us the capability to investigate the automated construction of knowledge-based image interpretation systems for a variety of tasks. It is difficult to envision how SPAM could have progressed from its initial hand crafted version to a more general system capable of performing multiple tasks without the development of these tools.

Future research includes expanding the range of aerial image interpretation tasks performed using the new SPAM architecture. We are also interested in the development of techniques for further automation of the knowledge acquisition process by using collections of hand-segmented imagery and existing large scale databases such as the FLIPcharts described in Section 3.1. One goal is to investigate the use of more knowledge

intensive techniques for knowledge acquisition toward systems capable of automatic selection of scene primitives and important spatial relationships.

## 7. Acknowledgments

Larry Eshelman and John McDermott provided some not terribly unfair comments on an early version of this paper. Tom Mitchell made us finally realize that a reorganization of the paper was in order. They did the best they could.

Lambert Wixson and Brian Yamauchi implemented major portions of SPATS and ISCAN. Robert Lai aided with the preparation of this paper. Thanks to Bob Simpson for inviting us to present it at the DARPA Image Understanding Workshop.

## 8. Bibliography

1. McKeown, D.M., Harvey, W.A. and McDermott, J., "Rule Based Interpretation of Aerial Imagery," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-7, No. 5, September 1985, pp. 570-585.
2. McKeown, D.M., McVay, C.A., and Lucas, B. D., "Stereo Verification In Aerial Image Analysis," *Optical Engineering,* Vol. 25, No. 3, March 1986, pp. 333-346, Also available as Technical Report CMU-CS-85-139
3. Froesch, C., and Prokosch, W., *Airport Planning,* John Wiley and Sons, Inc., New York, N. Y., 1946.
4. Horonjeff, R., and McKelvey, F. X., *Planning and Design of Airports,* McGraw-Hill Book Company, New York, N. Y., 1983, Third Edition
5. Defense Mapping Agency, "Flight Information Publication: Instrument Approach Procedures and Airport Diagrams," St. Louis Air Force Station, MI 63118, 7 July 1983.
6. H.G. Barrow and H. Popplestone, "Relational Descriptions in Picture Processing," *Machine Intelligence,* Vol. 6, 1971, pp. 377-396.
7. J.M. Tenenbaum and S. Weyl, "A Region-Analysis Subsystem for Interactive Scene Analysis," *International Joint Conference on Artificial Intelligence,* Vol. 2, 1975, pp. 682-687.
8. A. R. Hanson and E. M. Riseman, *VISIONS: A computer system for interpreting scenes,* Academic Press, New York, 1978, pp. 303-333.
9. Robert Belknap, Edward Riseman, and Allen Hanson, "The Information Fusion Problem and Rule-Based Hypotheses Applied to Complex Aggregations of Image Events," *DARPA Image Understanding Proceedings,* 1985, pp. 279-292.
10. Forgy, C. L., "The OPS5 User's Manual," Tech. report, Carnegie-Mellon University, Department of Computer Science, 1981.
11. Brownston, L., Farrell, R., Kant, E., and Martin, N., *Programming Expert Systems in OPS5,* Addison-Wesley, Reading, MA, 1985.

12. Buchanan, B.G., "Some Approaches to Knowledge Acquisition," Tech. report STAN-CS-85-1076, Stanford University, July 1985.
13. Larry Eshelman and John McDermott, "MOLE: A Knowledge Acquisition Tool That Uses Its Head," *Proceedings of the National Conference on Artificial Intelligence,* 1986, .
14. John McDermott, "Making Expert Systems Explicit," *Information Processing 86,* H.J. Kugler, ed., Elsevier Science Publishers B.V., 1986, pp. 539-544.
15. McKeown, D.M., "MAPS: The Organization of a Spatial Database System Using Imagery, Terrain, and Map Data," *Proceedings: DARPA Image Understanding Workshop,* June 1983, pp. 105-127, Also available as Technical Report CMU-CS-83-136
16. McKeown, D.M.,, "Digital Cartography and Photo Interpretation from a Database Viewpoint," in *New Applications of Databases,* Gargarin, G. and Golembe, E., ed., Academic Press, New York, N. Y., 1984, pp. 19-42.
17. Shang-Shouq Vincent Hwang, *Evidence Accumulation for Spatial Reasoning in Aerial Image Understanding,* PhD dissertation, University of Maryland, 1984.
18. McKeown, D.M., Denlinger, J.L., "Map-Guided Feature Extraction from Aerial Imagery," *Proceedings of Second IEEE Computer Society Workshop on Computer Vision: Representation and Control,* May 1984, Also available as Technical Report CMU-CS-84-117

# Appendix I

A short description of the attributes available for each phase, and their legal values, follows. The available geometric and spatial relationships are also given.

### Region-to-Fragment

For the region-to-fragment phase, knowledge about the expected shape of the classes of objects appearing in the scene is encoded. The <region-attribute> is a characteristic computed for each of the segmentation regions coming from the segmentation process, whether hand or machine.

```
'CLASS'                  = '<hypothesis>'
'REGION-DEPENDENCES'     = '<any string>'
'FRAG-DEPENDENCES'       = '<any string>'
'SHAPE-CONSTRAINT'       = '<region-attribute> && <range>'
        <any number of shape-constraints>
```

The following is a sample region-to-fragment schema used by the suburban-house scene version of SPAM. RULEGEN uses this schema to produce productions that define, via shape-characteristics, the subclass "house" within the SPAM system.

```
'CLASS'                = 'house'
'REGION-DEPENDENCES'   = ''
'FRAG-DEPENDENCES'     = 'object-type compact && hypothesis unknown'
'SHAPE-CONSTRAINT'     = 'area && 50.00 <= value <= 150.00'
'SHAPE-CONSTRAINT'     = 'ellipse-length && 12.00 <= value <= 18.00'
'SHAPE-CONSTRAINT'     = 'ellipse-width && 10.00 <= value <= 20.00'
'SHAPE-CONSTRAINT'     = 'ellipse-linearity && 0.00 <= value <= 3.50'
```

The attributes available to characterize the geometric constraints for a single scene primitive are summarized below. Most of these attributes are precomputed prior to being loaded into the interpretation system. The others are computed as they are needed. For example, if texture measures are used only to discriminate between the different subclasses of the class called blob, then texture need only be computed for that much smaller subset of regions that are interpreted as blob regions.

| <region-attribute> | <range> | <status> |
|---|---|---|
| texture-low | [0 - 100] | dynamically-computed |
| texture-moderate | [0 - 100] | dynamically-computed |
| texture-high | [0 - 100] | dynamically-computed |
| location-lat | [0 - 10000000] | precomputed |
| location-lon | [0 - 10000000] | precomputed |
| orientation | [0 - 2pi] | precomputed |
| ellipse-width | [0 - 5000] | precomputed |
| ellipse-length | [0 - 10000] | precomputed |
| mbr-width | [0 - 5000] | precomputed |
| mbr-length | [0 - 10000] | precomputed |
| depth-low | [0 - 100] | dynamically-computed |
| depth-moderate | [0 - 100] | dynamically-computed |
| depth-high | [0 - 100] | dynamically-computed |
| curvature | [0 - 1] | dynamically-computed |

| | | |
|---|---|---|
| ellipse-linearity | [0 - 1000] | precomputed |
| mbr-linearity | [0 - 1000] | precomputed |
| compactness | [0 - 1] | precomputed |
| fractional-fill | [0 - 1] | precomputed |
| area | [0 - 10000000000] | precomputed |
| perimeter | [0 - 10000000] | precomputed |

### Local-Consistency

We now describe the attributes and geometric relations used in defining a local-consistency rule. The knowledge represented makes explicit ambiguous spatial/relational concepts such as "close-to", "oriented-toward", or "far-from". This is done by imposing bounds on each spatial relation, and using a confidence function to smooth out the discontinuities associated with simply using thresholds.

```
'RULENAME'      = '<any string>'
'CONFIDENCE'    = '[0 - 1]'
'HYPOTHESES'    = '<hypothesis1> && <hypothesis2> && ...'
'GEOMETRICS'    = '<spatial-relation>'
'SUBTYPES'      = '<sub-relation>'
'BOUNDS'        = '<range>'
```

An example local-consistency schema follows, which defines the rule that houses should be parallel to roads.

```
'RULENAME'    = 'houses-are-parallel-to-roads'
'CONFIDENCE'  = '0.8'
'HYPOTHESES'  = 'house && road'
'GEOMETRICS'  = 'orientation'
'SUBTYPES'    = 'parallel'
'BOUNDS'      = '0.00 <= value <= 0.50'
```

The set of possible primitive spatial relations are listed below. This small set has been found to be expressive enough to describe local-consistency relations for the scenes SPAM has interpreted thus far i.e. the airport and suburban-housing scenes.

| <spatial-relations> | <sub-relations> | <range> |
|---|---|---|
| distance | centroid | [0 - 10000] |
| | average | " |
| | least | " |
| | greatest | " |
| orientation | toward | [0 - pi] |
| | parallel | " |
| | perpendicular | " |
| intersection | nil | [t, nil] |
| overlap | nil | [0 - 1] |

### Functional-Area

The knowledge encoded in the functional-area phase is somewhat implicit. It is represented by the associations made between hypotheses when one defines a functional-area type. The associated objects are located in close, physical proximity to one another

and have similar functions. Each of the FA-NAME attributes defines a functional-area type which will be used as a part of the overall scene model.

```
'FA-NAME'       = '<any string>'
'SEED-REGION'   = '<hypothesis>'
'DEFINITION'    = '<hypothesis1> && <hypothesis2> && ...'
```

The following functional-area schema defines the functional-area type *terminal* as being composed of terminal-building, road, parking-lot, and parking-apron hypotheses.

```
'FA-NAME'       = 'terminal'
'SEED-REGION'   = 'terminal-building'
'DEFINITION'    = 'parking-lot && parking-apron && road'
```

The SEED-REGION attribute forces the interpretation system to create terminal functional-areas only if a terminal-building hypothesis exists that is consistent with one or more hypotheses of the types occurring in the DEFINITION attribute.

### Model-Generation

The knowledge embedded in the model-generation phase has to do with using the context in which a particular region is found to determine which of several conflicting interpretations are correct. Commonly occurring conflicts can be enumerated, and more expensive knowledge-intensive operators can be applied to resolve these conflicts in the context of a particular scene model. The general syntax of a model-generation schema looks as follows:

```
'CONFLICT'      = '<hypothesis1> && <hypothesis2>'
'RESOLUTION'    = '<keyword> [&& <keyword-data>]'
        <any number of resolutions>
```

For example, consider the following schema:

```
'CONFLICT'      = 'hangar-building && parking-lot'
'RESOLUTION'    = 'function && stereo'
```

This schema will invoke a stereo operator to decide whether or not a region has height, so that the interpretation system can decide between the hangar-building or the parking-lot hypothesis.

| <keywords> | <keyword-data> |
|---|---|
| default | none |
| function | name of a function used to do resolution |
| conclusion | name of a function used to combine results |

If there is more than one resolution specified, then there must be a conclusion resolution specified. The conclusion will take the results of all of the resolution strategies and determine what the final result will be.

## Appendix II

Some examples of the productions generated by RULEGEN are now given. Because the high-level rule descriptions were given along with the schemata in the previous appendix, here we will attempt to describe how the productions actually implement semantics of each rule.

### Region-to-Fragment

Using the example schema for the region-to-fragment phase given in Appendix I, the system generated OPS5 productions defining the 'house' subclass. The first production finds an uninterpreted region in working-memory, and sets up a subtask which constrains OPS5 conflict-resolution to the productions in the given group only. These other productions apply the geometric constraints and leave the results of each test in a special LISP data-structure. Finally, domain-independent productions finalize this process by doing the final test evaluations, deciding whether or not an interpretation should be created, and removing the now obsolete subtask.

```
(p RTF::HS::initialize-HS-attributes
    (rtf-task ^region <name> ^data <token>)
    (region ^symbolic-name <name> ^house nil)
    (fragment ^symbolic-name <name>
        ^object-type compact
        ^hypothesis unknown)
  -->
    (make rtf-subtask ^ruleset HS::match-HS-attributes
        ^region <name> ^data <token> house)
)

(p RTF::HS::match-HS-area
    (rtf-subtask ^ruleset
        { <ruleset> = HS::match-HS-attributes }
        ^region <name> ^data {} <hyp>)
  { (rtf-rule-constants ^ruleset <ruleset>
        ^attribute area) <constants> }
    (region ^symbolic-name <name> ^area <value>)
  -->
    (bind <index> (litval constants))
    (call OPS::match-score <name> <hyp> <value>
        (substr <constants> <index> inf))
)

(p RTF::HS::match-HS-ellipse-length
    (rtf-subtask ^ruleset
        { <ruleset> = HS::match-HS-attributes }
        ^region <name> ^data {} <hyp>)
  { (rtf-rule-constants ^ruleset <ruleset>
        ^attribute ellipse-length) <constants> }
    (region ^symbolic-name <name>
        ^ellipse-length <value>)
  -->
    (bind <index> (litval constants))
    (call OPS::match-score <name> <hyp> <value>
        (substr <constants> <index> inf))
)
```

```
(p RTF::HS::match-HS-ellipse-width
    (rtf-subtask ^ruleset
        { <ruleset> = HS::match-HS-attributes }
        ^region <name> ^data {} <hyp>)
  { (rtf-rule-constants ^ruleset <ruleset>
        ^attribute ellipse-width) <constants> }
    (region ^symbolic-name <name>
        ^ellipse-width <value>)
  -->
    (bind <index> (litval constants))
    (call OPS::match-score <name> <hyp> <value>
        (substr <constants> <index> inf))
)

(p RTF::HS::match-HS-ellipse-linearity
    (rtf-subtask ^ruleset
        { <ruleset> = HS::match-HS-attributes }
        ^region <name> ^data {} <hyp>)
  { (rtf-rule-constants ^ruleset <ruleset>
        ^attribute ellipse-linearity) <constants> }
    (region ^symbolic-name <name>
        ^ellipse-linearity <value>)
  -->
    (bind <index> (litval constants))
    (call OPS::match-score <name> <hyp> <value>
        (substr <constants> <index> inf))
)
```

**Local-consistency**

Another example schema from Appendix I, for the local-consistency phase of SPAM, produces a set of productions defining the spatial relationship constraining houses to be parallel to roads. The first two productions, `call` and `init`, establish a subtask which, again, constrains the conflict-resolution process to the current production group. The next two productions, `invalid-type` and `no-rule-constraints`, do error checking. The next production, `exit`, removes the current subtask so that the remaining local-consistency rules can fire. The next two productions, `choose-RD` and `stop-choosing`, implement a loop in OPS5, so that all the computations can be performed at one time. At this point, domain-independent control productions take over and coordinate the spawning of subprocesses to do the low-level spatial calculations. When these processes have completed, the results are placed into working memory and control is allowed to pass back to this production group. Finally, the last two productions, `satisfied` and `unsatisfied`, will match this result data and create subtasks that will be used by domain-independent productions to update confidences appropriately.

```
(p LCC::houses-are-parallel-to-roads::*call*
    (consistency-task
        ^hypothesis house ^fragment <id>
        ^region <name> ^misc <con>)
  -->
    (make lcc-subtask
        ^rulename HS::houses-are-parallel-to-roads
        ^hypothesis house ^fragment <id>
        ^region <name> ^misc <con>)
```

```
)

(p LCC::houses-are-parallel-to-roads::*init*
  { (lcc-subtask
       ^rulename
     { <rulename> = HS::houses-are-parallel-to-roads }
       ^hypothesis house ^fragment <id> ^region <name>
       ^misc <con>) <subtask> }
   (lcc-rule-constants ^rulename <rulename>)
 -->
   (call OPS::dumpstate)
   (remove <subtask>)
   (make lcc-rule-set ^rulename <rulename>
       ^hypothesis house ^fragment <id>
       ^region <name> ^misc <con>)
   (make lcc-chain ^rulename <rulename>
       ^taskname start-choose-mode)
)

(p LCC::houses-are-parallel-to-roads::*invalid-type*
  { (lcc-subtask
       ^rulename
     { <rulename> = HS::houses-are-parallel-to-roads }
       ^hypothesis { <hyptype> <> house }) <subtask> }
 -->
   (remove <subtask>)
   (write (crlf) (tabto 9)
       <rulename> -- Invalid hypothesis
       <hyptype> for this ruleset.
       (crlf))
)

(p LCC::houses-are-parallel-to-roads::*no-rule-constants*
  { (lcc-subtask
       ^rulename
     { <rulename> = HS::houses-are-parallel-to-roads }
       ^hypothesis house) <subtask> }
 - (lcc-rule-constants ^rulename <rulename>)
 -->
   (remove <subtask>)
   (write (crlf) (tabto 9)
       <rulename> -- No rule constants
       for this ruleset.
       (crlf))
)

(p LCC::houses-are-parallel-to-roads::*exit*
  { (lcc-rule-set
       ^rulename HS::houses-are-parallel-to-roads)
    <ruleset> }
 - (geometry)
 - (queue)
 -->
   (remove <ruleset>)
)

(p LCC::houses-are-parallel-to-roads::*choose-RD*
   (lcc-chain ^rulename
     { <rulename> = HS::houses-are-parallel-to-roads }
       ^taskname start-choose-mode)
```

```
   (lcc-rule-set ^rulename <rulename> ^region <name0>
       ^fragment <id0> ^misc <conf0>)
   (fragment ^lcc-participant yes ^hypothesis road
       ^symbolic-name { <name1> <> <name0> }
       ^fragment-token <id1> ^confidence <conf1>)
   (lcc-rule-constants ^rulename <rulename>
       ^constants <thresh0-1> {})
  -->
   (call OPS::queue-task orientation parallel <name0>
      <name1> <thresh0-1> <id0> <conf0> <id1> <conf1>)
)

(p LCC::houses-are-parallel-to-roads::*stop-choosing*
  { (lcc-chain ^rulename
     { <rulename> = HS::houses-are-parallel-to-roads }
       ^taskname start-choose-mode) <chain> }
    (lcc-rule-set ^rulename <rulename>)
  -->
    (remove <chain>)
  )
(p LCC::houses-are-parallel-to-roads::*satisfied*
    (lcc-rule-set
       ^rulename
     { <rulename> = HS::houses-are-parallel-to-roads }
       ^fragment <id0>)
    (lcc-rule-constants ^rulename <rulename>
       ^constants <min> <max>)
  { (geometry ^type orientation ^subtype parallel
       ^frag1 <idT> ^con1 <cT> ^frag2 <id> ^con2 <c>
       ^values { <value> >= <min> <= <max> })
    <geometry> }
    (fragment ^fragment-token <id0>)
  -->
    (remove <geometry>)
    (bind <score>
       (OPS::geometric-score 0.0 <threshold> <value>))
    (bind <eltlen> 3)
    (make lcc-updates ^rulename <rulename>
         ^elt-len <eltlen> ^fragment <idT>
         ^data <id> <c> <score>)
    (make lcc-updates ^rulename <rulename>
         ^elt-len <eltlen> ^fragment <id>
         ^data <idT> <cT> <score>)
  )
 (p LCC::houses-are-parallel-to-roads::*unsatisfied*
   (lcc-rule-set
       ^rulename
   { <rulename> = HS::houses-are-parallel-to-roads })
   (lcc-rule-constants ^rulename <rulename>
       ^constants <threshold>)
 { (fragment ^fragment-token <idT> ^test-count <count>)
     <fragment> }
 { (geometry ^type orientation ^subtype parallel
       ^frag1 <idT>) <geometry> }
   -->
   (modify <fragment>
       ^test-count (compute <count> + 1))
   (remove <geometry>)
 )
```

# Appendix III

Figure 1 is an example of the output generated by SPATS for the region-to-fragment phase of SPAM. The explanations are generated as part of the final output for easy reference.

```
    Id: Moffett1

Column                          Explanation
------                          -----------
Class/Subclass: Class/Subclass to be analyzed
GndTth:         # of occurrences of the class/subclass in gnd truth table.
                Each class entry is the sum of its subclass entries.
WMEs:           # of region WMEs whose symbolic names match the subclass's
                gnd truth IDs and that have subclass or class interps.
                NOTE: The difference between the class entry and the sum
                of its subclass entries is the # of region WMEs with only
                class interps.
CorrWMEs:       # of the aforementioned WMEs which contained the
                correct subclass or class interpretation.
IncorrWMEs:     # of the aforementioned WMEs which did not contain
                the correct subclass or class interpretation.
                NOTE:  The sum of the CorrWMEs and IncorrWMEs entries for
                each class or subclass should add up to its WMEs entry.
CorrBF:         The branching factor for the correct interpretations
                of the subclass or class.  The BF shows how many
                interpretations, there were, on the average, for each correct
                interpretation.  If a class/subclass has a CorrBF of 0, then
                it had no correct interpretations.
IncorrBF:       The branching factor for the incorrect interpretations
                of the subclass or class.
```

| Class/subclass | GndTth | WMEs | CorrWMEs | IncorrWMEs | CorrBF | IncorrBF |
|---|---|---|---|---|---|---|
| linear | 40 | 40 | 40 | 0 | 6.65 | 0.00 |
| runway | 2 | 2 | 2 | 0 | 3.00 | 0.00 |
| taxiway | 36 | 36 | 36 | 0 | 6.89 | 0.00 |
| road | 2 | 2 | 2 | 0 | 6.00 | 0.00 |
| compact | 9 | 9 | 5 | 4 | 7.00 | 6.00 |
| hangar-building | 9 | 9 | 5 | 4 | 7.00 | 6.00 |
| terminal-building | 0 | 0 | 0 | 0 | 0.00 | 0.00 |
| small-blob | 6 | 6 | 4 | 2 | 8.50 | 4.00 |
| parking-apron | 3 | 3 | 1 | 2 | 7.00 | 4.00 |
| parking-lot | 3 | 3 | 3 | 0 | 9.00 | 0.00 |
| large-blob | 12 | 12 | 12 | 0 | 5.92 | 0.00 |
| grassy-area | 11 | 11 | 11 | 0 | 6.09 | 0.00 |
| tarmac | 1 | 1 | 1 | 0 | 4.00 | 0.00 |
| Final Stats: | 67 | 67 | 61 | 6 | 6.66 | 5.33 |

**Figure 1:** Example SPATS output for region-to-fragment phase.

From these statistics, one can see that the system was able to correctly interpret the linear and large-blob classes without any any misinterpretations at all. For the compact class, notice that the number of hangar-building interpretations is identical to the number of compact interpretations. This shows us that the geometric constraints for the subclass hangar-building are not discriminatory enough. This shows up in the correct branching factor as well.

An example of the output for the functional-area phase is given in figure 2. This summarizes all the correct and incorrect hypotheses participating in the created functional-areas. We can use this information to determine the status of the high-level groupings generated by SPAM. If it is recognized that many incorrect interpretations are being used to support correct interpretations (or visa-versa) when creating a functional-area, then the local-consistency knowledge is at fault, as it is not properly characterizing the spatial layout of the scene.

```
    Id: Moffett1

Functional Area Type: All functional areas
Functional Area ID: All functional areas
Total # of functional-areas: 83
Total # of fragments (from FA, consistent-fragments,
                      and inconsistent-fragments lists): 60
Total # of the above fragments found in ground truth file: 60
Fragments composing FA(s):
        # of correct fragment hypotheses:    60
        # of incorrect fragment hypotheses:  17
Consistent-Fragments:
        # of correct fragment hypotheses:      1
        # of incorrect fragment hypotheses:    4
Inconsistent-Fragments:
        # of correct fragment hypotheses:      1
        # of incorrect fragment hypotheses:   15
Correct fragment hypotheses table:
                                Ground Truth Types
                    RW    TW    RD    HG    TB    PA    PL    GA    TM
        RW           2     0     0     0     0     0     0     0     0
        TW           0    38     0     0     0     0     0     0     0
        RD           0     0    23     0     0     0     0     0     0
Frag.   HG           0     0     0     3     0     0     0     0     0
Types   TB           0     0     0     0     0     0     0     0     0
        PA           0     0     0     0     0     6     0     0     0
        PL           0     0     0     0     0     0    13     0     0
        GA           0     0     0     0     0     0     0    27     0
        TM           0     0     0     0     0     0     0     0     0

Incorrect fragment hypotheses table:
                                Ground Truth Types
                    RW    TW    RD    HG    TB    PA    PL    GA    TM
        RW           0     4     1     0     0     0     2     0     1
        TW           0     0     0     0     0     0     1     0     0
        RD           0    86     0     5     0     0     5     0     2
Frag.   HG           0    19     0     0     0     0     1     0     0
Types   TB           0    73     5    12     0     0     1     0     6
        PA           0    13     0    11     0     0     3    38     9
        PL           0    86     0    39     0     9     0    63     2
        GA           0    14     0    11     0     3     6     0     1
        TM           0     0     0     4     0     1     1     6     0
```

**Figure 2:** Example SPATS output for functional-area phase.