Experiments in Knowledge Refinement for a Large Rule-Based System

Wilson A. Harvey, Jr. Milind Tambe

August 1993
CMU-CS-93-195

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This work was sponsored by the Defense Advanced Research Projects Agency under Contract DACA76-92-C-0036, and by the Air Force Office of Scientific Research under Contract F49620-92-J-0318.

Keywords: diagnostic systems, knowledge refinement, rule-based systems, computer vision

Abstract

Knowledge-refinement is a central problem in the field of expert systems. For rule-based systems, refinement implies the addition, deletion and modification of rules in the system so as to improve the system's overall performance. The goal of this research effort is to understand the methodology for refining large rule-based systems, as well as to develop tools that will be useful in refining such systems.

The vehicle for our investigation is SPAM, a production system (rule-based system) for the interpretation of aerial imagery. Complex and compute-intensive systems like SPAM impose some unique constraints on knowledge refinement. More specifically, the credit/blame assignment problem for locating pieces of knowledge to refine becomes difficult. Given that constraint, we approach the problem in a bottom-up fashion, i.e., begin by refining portions of SPAM's knowledge base, and then attempt to understand the interactions between them. We begin by identifying gaps and/or faults in the knowledge base by comparing SPAM's intermediate output to that of an expert, then modifying the knowledge base so that the system's output more accurately matches the expert's output. While this approach leads to some improvements, it also raises some interesting issues concerning the evaluation of refined knowledge at intermediate levels and of interaction between the refinements. This paper presents our initial efforts toward addressing these issues.

1 Introduction

Knowledge-refinement is a central problem in the field of expert systems [Buchanan and Shortliffe, 1984]. It refers to the progressive refinement of the initial knowledge-base of an expert system into a high-performance knowledge-base. For rule-based systems, refinement implies the addition, deletion and modification of rules in the system so as to improve the system's *empirical adequacy*, i.e., its ability to reach correct conclusions in the problems it is intended to solve [Ginsberg *et al.*, 1988].

The goal of our research effort is to understand the methodology for refining large rule-based systems, as well as to develop tools that will be useful in refining such systems. The vehicle for our investigation is SPAM, a production system (rule-based system) for the interpretation of aerial imagery [McKeown *et al.*, 1985, McKeown *et al.*, 1989]. It is a mature research system having over 600 productions, many of which interact with a variety of complex (non-rule-based) geometric algorithms. A typical scene analysis task requires between 50,000 to 400,000 production firings and an execution time of the order of 2 to 4 cpu hours[1].

Large, compute-intensive systems like SPAM impose some unique constraints on knowledge refinement. First, the problem of credit/blame-assignment is complicated; it is extremely difficult to isolate a single culprit production (or a set of culprit productions) to blame for an error observed in the output. As a result, the methodology adopted in well-known systems such as SEEK and SEEK2 [Politakis and Weiss, 1984, Ginsberg *et al.*, 1988], or KRUST [Craw and Sleeman, 1991], cannot be directly employed to refine knowledge in SPAM. These systems identify errorful output and backward-chain through the executed rules to localize the source of the error. A second problem is that SPAM's long run-times make it difficult to rely on extensive experimentation for knowledge refinement. Iterative refinement and generate and test methods typically require many experiments to improve the knowledge base. In particular, SPAM's long run-times prohibit a thorough search of the space of possible refinements.

Given SPAM's constraints, we have employed a bottom-up approach for knowledge refinement. Specifically, we begin by refining small portions of SPAM, and then attempt to understand the interactions of these refinements and their impact on intermediate results. Fortunately, SPAM is already divided into four phases, facilitating the identification of the "modular" pieces on which we focus our refinement efforts, as well as our evaluation efforts (discussed below). We begin by identifying gaps and/or faults within small portions of SPAM's individual phases by comparing their output to that of an expert. The knowledge is modified to more accurately match the expert's output. We then evaluate the new output to see how well the refined knowledge is performing. This method chains forward from the refined knowledge to an evaluation of it's performance. This is in contrast to the backward-chaining systems cited above, which identify errorful output and reason backward to find the faulty knowledge (i.e., assign credit/blame). Backward chaining is difficult when the interactions between rules are complex. In particular, in SPAM, this implies backward-chaining hundreds of thousands of rule firings as well as applications of complex algorithmic constraints, which would be extremely difficult.

However, forward-chaining does not obviate the credit/blame assignment problem. It introduces the problem in a new form — that of evaluating the impact of the knowledge refinements. In particular, in forward-chaining systems, given complex rule interactions, refinement in one part of the system need not improve the end result. This may occur because the refined portion may contribute to overall improvement in a relatively small number of cases; or a second culprit component involved in the chaining may actually minimize the impact of the refinements. Hence, there

[1] Using a highly optimized C-based OPS5 [Kalp *et al.*, 1988] running on a DEC 5000/200.

is a need for evaluation and analysis of *intermediate* results. Of course, these intermediate results should more than exactly evaluate the parts of the system refined as a result of interaction with the expert — they should progressively yield a better understanding of how to change the rules to better the system's overall performance. SPAM's decomposition into phases helps us to a certain extent: the endpoints of the phases serve as potential intermediate evaluation points.

In our work so far, we have focused on the second phase in SPAM local-consistency (LCC). This phase was chosen because most of SPAM's time is spent in this phase, and because it shows the most potential for future growth. LCC performs a modified constraint satisfaction between hypotheses generated in SPAM's first phase. It applies constraints to a set of plausible hypotheses and prunes the hypotheses that are inconsistent with those constraints. For example, objects hypothesized as hangar-buildings and objects hypothesized as parking-aprons would participate in the constraint *hangar-buildings and parking-aprons are close together*. This rule is realized as a distance constraint on pairs of objects hypothesized to be hangar-buildings and parking-aprons. A successful application of an LCC constraint provides support for each pair of hypotheses, and an unsuccessful application decreases support for that pair of hypotheses. In essence, each constraint in the LCC phase classifies pairs of hypotheses — either the constraint supports that pair, or it does not. (LCC and SPAM are described in further detail in Section 2.)

In working toward refining these constraints, we posed several questions:

1. What role does this constraint play in the interpretation process?
2. If the constraint does play a role, is it positive (helpful) or negative (unhelpful)?
3. If the role is positive, and the constraint is numeric, can we optimize the constraint values?
4. How effective (powerful) is this constaint?
5. What is this constraint's impact on run time?

To address these questions, we began by asking a user to manually compile a database of correct intermediate outputs — the *ground-truth* database. The database and the system outputs were compared. An automatic procedure was created to adjust the SPAM knowledge base so that the correspondence between the ground-truth database and the knowledge base improved. This adjusted knowledge was then re-run through the system and the entire set of intermediate outputs (refined or otherwise) were evaluated. The somewhat surprising result of this analysis was that the expert's input did not help as much as expected. This result raised questions about why the expert's input was not as helpful as anticipated, and how the results should be evaluated.

In order to better understand both the interactions between constraints and the effects of modifying the knowledge base, we did a brute-force evaluation of the results, providing some answers to the questions we posed above. Namely, the distance constraints do play positive roles in SPAM's interpretation process, though they are not very powerful. More importantly, we now see that refined knowledge may apply selectively. That is, the set of objects affected by individual constraints largely overlaps, implying that SPAM could reduce computation by selectively applying constraints and still achieve the similar performance. Tools eliciting the expert's input must carefully take these interactions into consideration. Finally, we can also conclude that intermediate result evaluation is not straight-forward. For example, the complex structures that SPAM generates using the results of constraint application, called *functional-areas*, must be matched and these matches evaluated.

Having provided some of the motivation for this work, we begin by describing the SPAM image interpretation system system. Following this background material, we present our refinement methodology and an analysis of our results. We then re-evaluate the methodology and discuss further experimental results. Finally, we will outline some areas of future work.

2 Background

Before we describe the refinement methodology, it is useful to understand SPAM, LCC, and how LCC fits within the SPAM architecture. It is interesting and important to consider the nature of the computations performed in each phase, as well as the interactions between each phase.

SPAM is a production system architecture for the interpretation of aerial imagery with applications to automated cartography and digital mapping [McKeown *et al.*, 1985, McKeown *et al.*, 1989]. It tests the hypothesis that the interpretation of aerial imagery requires substantial knowledge about the scene under consideration. Knowledge about the type of scene, whether airport, suburban housing development, or 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. SPAM has been applied in two task areas: airport and suburban house scene analysis.

As with many computer vision systems, SPAM attempts to interpret the 2-dimensional image of a 3-dimensional scene. The particular goal of the SPAM system is to interpret an image segmentation, composed of image regions, as a collection of real-world objects. For example, the output given an input image containing an airport would be a model of the airport scene, describing where the runway, taxiways, terminal-building(s), etc., are individually located. SPAM uses four basic types of scene interpretation primitives: *regions*, *fragments*, *functional-areas*, and *models*. SPAM performs scene interpretation by transforming image *regions* into scene *fragment* interpretations. It then aggregates these fragments into consistent and compatible collections called *functional-areas*. Finally, it selects sets of functional-areas to form *models* of the scene.

The first phase of SPAM is called region-to-fragment (RTF). This is a traditional heuristic classification process, using knowledge about the classes of features that occur in the scene to map a segmentation to a set of interpretations. Local properties of the segmentation, such as shape, texture, or height, are used to select which interpretations to generate. Examples of the type of knowledge used in region-to-fragment would be *runways are typically 50 to 80 meters wide*, or *houses are 8 to 10 meters high.*

The second phase of SPAM is called local-consistency check (LCC). This phase performs a modified constraint satisfaction between the interpretations generated in region-to-fragment. The knowledge in this phase consists of the constraints between the different pairs of objects. An example constraint would be *runways have perpendicular taxiways.* There are many such constraints in the system — some numeric (e.g., distance) and some non-numeric (e.g., intersection). Each constraint classifies the participating hypotheses to be consistent, inconclusive or inconsistent. The collective action of several constraints is used to determine whether two hypotheses strongly support one another.

The functional-area (FA) phase groups together those hypotheses that support one another, where support is computed from the results of the previous phase (LCC). A functional-area is defined as a group of hypotheses that are similar in function and are in close physical proximity to one another. For instance, a terminal functional-area is defined to contain only terminal-building, parking-lot, road, and parking-apron hypotheses. It is physically represented by the convex hull of the features in the functional-area.

Finally, the model-generation (MODEL) phase uses the functional-areas and combines them based on a number of heuristics, including number of conflicts, number of supported hypotheses, and area of coverage. Conflicts are identified and resolved, either using support within the context of the model, or by invoking some process that would provide additional knowledge. For example, if in the context of a model a region of the image was interpreted as both a taxiway and a hangar-building, a stereo process could be invoked to help resolve the conflict based on the region's height estimate.

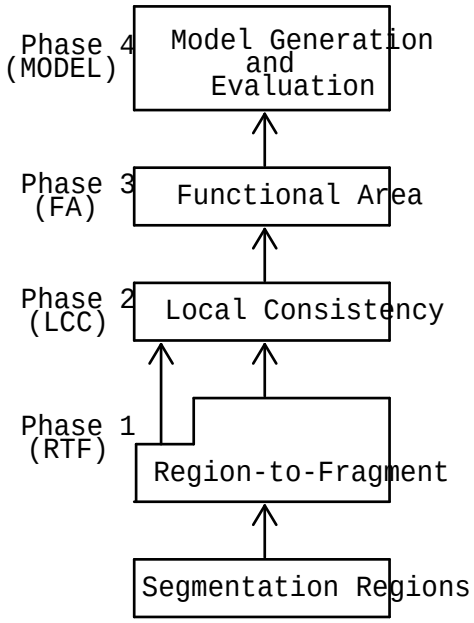

Figure 1: Interpretation phases in SPAM.

Multiple models are commonly generated and these models can be used as contexts for further processing.

As shown in Figure 1, each interpretation phase is executed in the order given. SPAM drives from a local, low-level set of interpretations to a more global, high-level, scene interpretation. There is a set of hard-wired productions for each phase that control the order of rule executions, the processing of geometric computations, and other domain-independent tasks. However, this organization does not preclude interactions between phases. For example, prediction of a fragment interpretation in the *functional-area* (FA) phase will automatically cause SPAM to reenter the *local-consistency check* (LCC) phase for that fragment. Other forms of such activity include stereo verification to disambiguate conflicting hypotheses in the *model-generation* (MODEL) phase and to perform linear alignment in the *region-to-fragment* (RTF) phase.

Another way to view the flow of processing in SPAM is that knowledge is used to check for consistency among hypotheses; contexts are created based on collections of consistent hypotheses, and are then used to predict missing components. A collection of hypotheses must combine to create a context from which a prediction can be made. These contexts are spatial aggregations in the scene. For example, a collection of mutually consistent runways and taxiways might combine to generate a runway functional area. The context of a runway functional area then predicts that certain sub-areas within that functional area are good candidates in which grassy areas or tarmac regions may be found. 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 focus on those areas with strong support at each level of the interpretation.

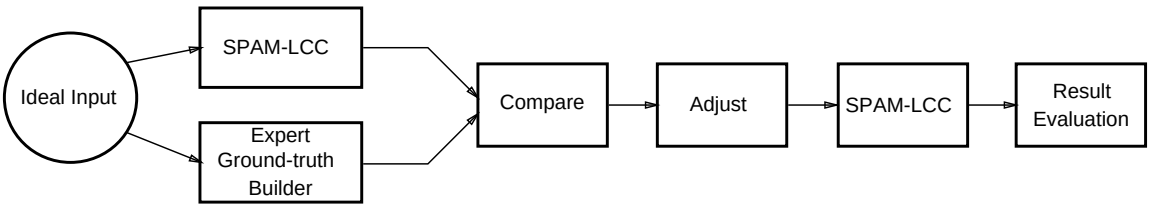

Figure 2: A schematic of the knowledge refinement process.

3 Using Expert Input for Knowledge Refinement

As stated in Section 1, we have focused our work on the second phase of SPAM's processing, LCC, because of the large amount of time spent in this phase, and because of it's potential for growth. There are several types of constraints used in LCC: distance, orientation, overlap, along-side, and others. We chose to focus on the distance constraint because it is used extensively and, therefore, appears to play an important role in interpretation.

The distance constraint supports a pair of hypotheses by classifying them as either being consistent, inconclusive, or inconsistent. This classification is based upon how close the objects are to one another and what numeric bounds are assigned to be consistent, inconclusive, or inconsistent. Refining this constraint implies modifying the range of distances that classify pairs of hypotheses as being consistent, inconclusive, or inconsistent, so as to improve the number of correct classifications. For example, if the original bounds settings are 0–100 meters, 100–150 meters, and 150–∞ meters, for each of the consistent, inconclusive, and inconsistent ranges, respectively, then a refinement of this constraint may change the bounds to be 0–124 meters, 124–187 meters, and 187–∞ meters.

In order to refine the range settings in this manner, we need to compare the output of SPAM to the "ideal" output for LCC. Tools were built to allow a knowledgable user (the *expert*) to produce a ground-truth and to compare that ground-truth to SPAM's output. This type of comparison permits a quantitative evaluation of system performance that can be used to drive the refinement. An automatic procedure is then used to adjust the constraint range settings so that SPAM's output optimally matches the expert's.

We have focussed our investigation on hangar-building functional-areas, which are made up of hangar-building, access-road, parking-apron, and tarmac objects[2]. This focus was motivated by our previous investigations (see [Harvey *et al.*, 1992]) which showed that the constraints used to generate hangar-building functional-areas were not performing well.

An individual constraint's performance should improve with this method, though it is also important that the system's overall output improve using the adjusted contraint. We must evaluate embedded performance at a place within the system where the intermediate results have been used, and are easy to evaluate. We chose to evaluate performance at the end of SPAM's third phase, FA.

In summary, once we have chosen the distance constraint from LCC for refinement, we use feedback from an expert to *compare* with an individual constraint. Then, we *adjust* the range settings for this constraint based on the previous comparison, and, finally, *evaluate* the intermediate performance using the adjusted constraint. A schematic of this process is shown in Figure 2. Of these three components — *compare, adjust,* and *evaluate* — the first two permit the isolation and improvement

[2]The objects represented by a particular functional-area type are defined by the SPAM user.

of individual constraints, while the third allows us to evaluate the performance of the refined knowledge in the context of the system's intermediate output. We will describe these components in more detail and then discuss some of the results from applying this process to SPAM running on three different datasets.

3.1 Compare: Comparison with Expert Ground-Truth

A natural method for generating an ideal *ground-truth* for LCC is to allow a knowledgable user (the *expert*) to enumerate those constraints that should exist between each pair of objects in the ideal input. A graphical tool was built that allows the user to display the ground-truth database in an interactive window. Using a mouse, the expert can then select pairs of hypotheses that he/she feels satisfy a given constraint. In this way, the expert can easily build up a database of constraints and object pairs satisfying those constraints.

The output of LCC can now be compared with the expert generated ground-truth[3]. Such a comparison is informative as it allows a quantitative measure of error to be computed. The result of applying each LCC constraint is compared with the expert and classified in the form of a confusion matrix. The confusion matrices contain the usual cells (true-positives, false-positives, true-negatives, false-negatives). A *true-positive* entry in the confusion matrix indicates situations where the expert and LCC both conclude that the constraint supports a pair of hypotheses. A *true-negative* entry indicates situations where the expert and LCC both conclude that the constraint does not support a pair of hypotheses. A *false-positive* entry is one where LCC concludes support, while the expert does not. A *false-negative* entry is one where the expert concludes support, while LCC does not. A single matrix represents the results for a single constraint (e.g., distance) and a single pair of classes (e.g., hangar-buildings versus roads). For the numeric distance constraint, each cell of the confusion matrix is a histogram: the horizontal axis denotes the distance, and the vertical axis denotes the number of entries for that value of distance. An example of the confusion matrix output for the hangar-building versus road distance constraint is shown in Figure 3. A similar confusion-matrix is created for each distance constraint that exists between each pair of object classes.

The confusion matrix can easily be modified to include inconclusive entries (where the expert has not provided any answer). This results in a three-by-three matrix representation. However, our refinement procedure and the ensuing analysis remains the same whether we use the two-by-two or the three-by-three matrix. Therefore, we will use the two-by-two confusion matrix, as this simplifies the language of the discussions that follow.

3.2 Adjust: Optimization of Constraint Performance

The performance of the distance constraint in LCC depends on the values that delimit the consistent, inconclusive, and inconsistent classification ranges. The range for each classification type is reflected in the cells of the confusion matrices. By examining the overlap of each of the cells in a confusion matrix we can determine how well SPAM's distance constraint is performing. For example, the overlap of true positives with false positives can tell us how the constraint can be modified to achieve the greatest number of true positives without introducing too many false positives.

[3]In order to ensure that any errors in LCC's output are directly attributable to the constraints, we must specify "ideal" inputs. For each of the sets of data that we run through SPAM we maintain a ground-truth database containing all the objects in the scene with their correct hypothesis labels. An ideal input to the LCC phase, a set of hypotheses that are 100% correct, can easily be generated from this ground-truth and run through the system. This is done to remove any ambiguity as to where errors in SPAM's output might come from.

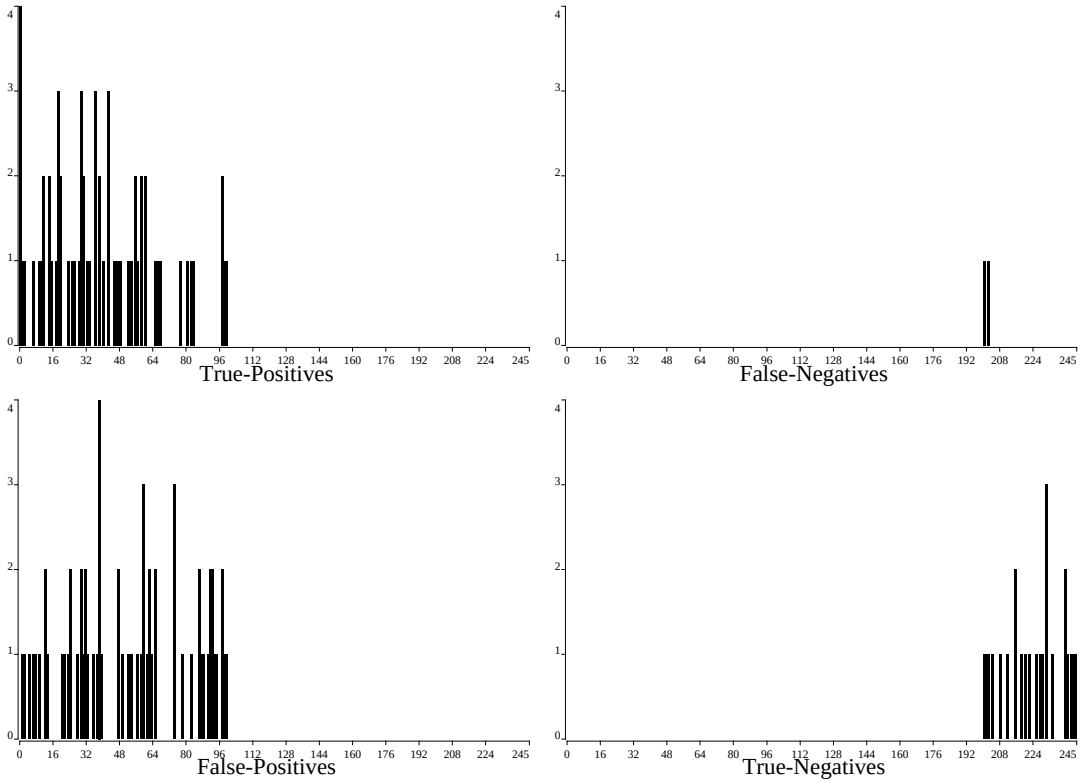

Figure 3: Confusion matrix showing correctness of SPAM's hangar-building/road distance constraint.

By adjusting these ranges, we can improve the classification performance of the constraint. For the distance constraint, we have developed an automatic process for adjusting the constraint bounds to generate an "optimal" set of ranges. To define the measure that we are optimizing, let us consider the ideal case when the system's output exactly matches the expert's. In our confusion matrix representation, only the diagonal cells will be populated.

Our current objective function seeks to maximize the number of elements in the diagonal cells of the matrix — true-positives and true-negatives (see Figure 3[4]). It weighs all cells in the confusion matrix equally and increases as more entries move from the off-diagonal cells to the diagonal cells. Automated bounds selection is achieved by doing an exhaustive search through the space of possible bounds settings, evaluating each setting with this objective function, and outputting the set of bounds that maximize the objective function. These bounds are then used as the new range settings for the distance constraint, and are compiled into a new SPAM system[5].

A pictorial view of this space of bounds settings is shown in Figure @reffig-adjustspace. This is a slice of the space of the objective function evaluations for hangar-building and road hypotheses. The horizontal axis represents the value of the distance constraint setting for each classification range (in this case, 0 to 500 meters). The vertical axis (height) represents the value of the objective function at each point in this space. Two things become clear when viewing the space of possible bounds settings in this manner. First, there is more than one optimal range setting. Currently, we choose the optimal setting that maximizes the consistent (positive) range. Additionally, one can

[4]For a 3x3 matrix, this would include true-inconclusives.

[5]Extending this method for other numeric and non-numeric constraints remains an interesting issue for future work.

Figure 4: A slice from the space of possible bounds settings for SPAM's hangar-building/road distance constraint.

see that there are a large number of local minima. A simple hill-climbing algorithm would not find globally optimal point(s), showing that the exhaustive search may be necessary. Our experience is that the exhaustive search can be done in a reasonable amount of time (less than five minutes on a DEC 5000/200).

3.3 Evaluate: Embedded Evaluation

The objective function described above, coupled with *adjust*, the range adjustment procedure, will improve the performance of isolated individual constraints. But how should we evaluate the impact of this adjustment on the rest of the system? As discussed in Section 1, evaluation by running SPAM and examining the end result is not very useful. An intermediate evaluation point that is somewhat closer to the adjusted constraint is necessary. This intermediate point should also facilitate evaluation. For this reason, we measured SPAM's performance after the third phase (FA), the first point at which the results of constraint application is utilized.

The embedded evaluation function we use is straight forward. We execute SPAM with the new, adjusted bounds settings and compare the results at the end of the FA phase to those obtained using the original bounds settings. The adjusted bounds will produce different sets of consistent, unknown, and inconsistent hypotheses, as well as (possibly) different sets of functional-areas. We compare the same functional-area in both runs and compute both the difference in the number of correct hypotheses, and the difference in the number of incorrect hypotheses. The number of correct hypotheses should increase and the number of incorrect hypotheses should decrease. The score of each functional-area is the increase in the corrects plus the decrease in the incorrects. A total score for each run is computed by summing these scores for each functional-area. So, a positive overall score means that we have improved SPAM's performance by including more correct interpretations and/or excluding more incorrect interpretations.

4 Results and Analysis

We began by collecting baseline results for four SPAM datasets: Moffett (Moffett Air Force Base), DC National (Washington National Airport), SF International (San Francisco International Airport), and Dulles (Dulles International Airport). We then applied our refinement methodology to each dataset (built expert ground-truth, applied the adjustment procedure, then ran and evaluated the results). The original and adjusted bounds settings are shown in Figure 5.

Distance Constraint Name (hangar-building vs. ...)	Original Bounds Setting (meters)			Adjusted Bounds Setting (meters)		
	consistent	inconclusive	inconsistent	consistent	inconclusive	inconsistent
parking-aprons	0– 50	50–150	150–∞	0–47	47– 54	54–∞
access-roads	5–100	100–200	200–∞	0–59	59– 60	60–∞
tarmacs	15–100	100–200	200–∞	0–78	78–102	102–∞
hangar-buildings	0–200	200–300	300–∞	0–70	70– 73	73–∞

Figure 5: Original and adjusted bounds settings for SPAM's hangar-building distance constraints.

The table in Figure 6 shows our baseline and refinement results for the four datasets mentioned above. All experiments were performed on a DEC 5000/200 with 64MB of memory running MACH 2.6.

Dataset	Bounds Setting	Time (cpu hrs)	# Rule Firings	# of FAs	Total # Correct	Total # Incorrect	Change in # Correct	Change in # Incorrect	Score (Score/Total)
Moffett	original	0.35	106367	14	19	111	—	—	—
"	optimized	0.34	104903	13	11	100	−6	−10	+4 (3.1%)
DC Ntnl	original	0.58	158182	25	42	209	—	—	—
"	optimized	0.57	156453	25	38	167	−4	−42	+38 (15.1%)
SF Intl	original	1.39	320993	49	46	235	—	—	—
"	optimized	1.41	320329	37	21	135	−12	−49	+37 (13.2%)
Dulles	original	3.45	585115	80	136	576	—	—	—
"	optimized	3.45	575145	77	74	428	−62	−133	+71 (9.9%)

Figure 6: SPAM performance, measured along several dimensions, for both the original and refined hangar-building distance constraint settings.

The first column, labeled Dataset, is the name of the airport from which the data comes. The second column, Bounds Setting, indicates the version of the constraint bounds that were used; *original* is the baseline version and *optimized* is the refined version of the constraint bounds. The column labeled Time is the amount of time in CPU hours it takes for the SPAM system to execute through it's third phase. The Rule Firings column is the number of production rules fired during the three phases. The column labeled FA shows the number of functional-area objects generated by the system. The next two columns, Total Number of Corrects/Incorrects, display the total number of correct and incorrect hypotheses (determined by comparing them to our ground truth) that participate in the generated functional-areas. The final three columns are the results of applying our embedded evaluation function to the set of resulting (hangar-building) functional-areas. The percentage is computed by comparing the score with the total number of corrects and incorrects from the baseline (original) version.

It can be seen that the results improve somewhat for all four datasets. In the case of the Moffett dataset we observe that, according to the evaluation score, the performance of the system improves only slightly (3.1%). For National, the performance has improved, mostly due to the loss of many (42) incorrect hypotheses. San Francisco and Dulles also show improvements, again, mostly due to the elimination of incorrect hypotheses (false-positives). Additionally, in all cases, the number of rule firings slightly decreased while the computation time was largely unaffected.

The results obtained from using our refinement methodology seem positive. However, in spite of focussing exclusively on constraints dealing with hangar-buildings, and significant changes in the distance constraint bounds, the surprising result was that only moderate improvements in performance were achieved.

4.1 Discussion

Gaining only small improvements in performance given the large changes in the constraint bounds prompted us to ask the following questions:

1. Are we evaluating our results correctly?
2. Why didn't the expert input help as much as expected?
3. Do the hangar-building distance constraints play as large a role in LCC as hypothesized?

In answering these questions, we have identified several issues in our methodology which we discuss below.

We begin by examining *evaluate*, the embedded evaluation function. We observe that the number of incorrects is significantly higher than the number of corrects. This creates an imbalance in the evaluation function. For instance, consider the scores generated when we allow the distance constraint's "consistent" and "inconsistent" ranges to decrease to zero. This allows very few objects to be considered consistent, though the score will tend to increase because the number of incorrect interpretations eliminated is comparitively large. This is in spite of decreasing the size and, therefore, the utility, of the generated functional-areas. In fact, the evaluation function will still yield a positive result even when no functional-areas are generated.

Ideally, we want an evaluation function that will peak when the best outputs are generated (an upside-down U-shaped curve). It would be possible to achieve this if we could modify our existing evaluation function in one of two ways. We could weight the composite pieces of the function (number of corrects, number of incorrects) such that we get the desired behavior. This is somewhat arbitrary, as these weights would have to be heuristically determined. We could also add more terms to the evaluation function. From our experience with SPAM, we know that measures like the area of the functional-area (coverage), number of elements, density, overlap with other functional-areas, etc., are good indicators of correct results. However, we again encounter the problem of having to assign arbitrary weights to these terms; weights that will likely not be generally applicable to all datasets. This is a common problem with evaluation functions that use weighted linear sums (e.g., see [Cooke, 1991]). Therefore, neither method seemed reasonable.

Although we sought a numeric evaluation function that would capture the salient features of a good functional-area, we resorted to doing a subjective visual inspection of the functional-areas generated by the system. We note that the embedded evaluation function does behave correctly when the constraint's "consistent" range increases toward infinity, and it seems to yield reasonable results at intermediate values. Therefore, we felt justified in using this evaluation function in conjunction with our observations as a reasonable measure of our results.

Next, we consider the question of why the expert's input didn't significantly help to improve the performance of the distance constraints. As described in Section 3.1, we produce an expert ground truth *separately* for each type of constraint. This makes two assumptions: (1) the constraints do not interact strongly with one another; (2) object pairs (selected by a single pair-wise constraint) can be considered in isolation from other objects within the scene (context does not play a major role). Both of these assumptions imply that the constraints are assumed to be independent of one

another. We anticipated some interaction between constraints, but not enough to affect the results of our refinement.

We reasoned that, if we were not violating either of these assumptions, we would be able to turn off a single constraint type and see dramatic changes in the size and number of functional-areas produced. The table in Figure 7 shows the results of turning off the hangar-building distance constraints, leaving only the action of the hangar-building orientation constraints. We compare this data to the baseline data in Figure 6. We observe that, except for San Francisco International, we do not see major changes in the number of of corrects and incorrects generated. If we consider the coverage of the generated functional-areas, we see little change in the results of any of the datasets. A visual inspection of the generated functional-areas confirms that major changes did not occur. The generated functional-area groups get smaller, but they do not radically change in size.

Dataset	Bounds Setting	Time (cpu hrs)	# Rule Firings	# of FAs	Total # Correct	Total # Incorrect	Change in # Correct	Change in # Incorrect	Coverage (Change)
Moffett	off	0.32	92033	13	9	78	−8	−32	−0.8%
DC Natl	off	0.52	140286	24	32	122	−10	−82	−3.1%
SF Intl	off	1.26	281917	20	4	77	−10	−19	−9.9%
Dulles	off	3.00	497261	71	47	322	−81	−213	−12.2%

Figure 7: Results of turning off the hangar-building distance constraints.

As stated above, some interaction was expected, but the inter-constraint interaction was much stronger than anticipated. These results imply that we have violated our independence assumptions. The distance constraint seems to be selectively applicable, i.e., it largely overlaps with the other constraints (orientation), but it is necessary for the inclusion of some subset of hypotheses. This result is a confirmation of SPAM's hypothesis that geometric constraints are weak predictors, especially in the presence of noisy sets of hypotheses. It is the weak support of multiple constraints that determines whether or not two objects are consistent with one another.

This result helps to answer our final question concerning the role that the hangar-building distance constraints play in SPAM's interpretation process. The distance constraints are important in that they provide yet another increment of support to consistent objects. Additionally, they uniquely support some objects in the datasets. These objects would otherwise be left out of the final functional-area results were it not for the action of the distance constraints. However, as noted, their effect is weak when evaluated in the context of the final results.

This result also has ramifications for our optimization process, as it assumes that each constraint is independent of every other constraint. We believe we can fix this assumption violation in at least one way. We observe that a constraint is critical to those objects that are affected *only* by that constraint (and no other). Therefore, we can isolate these objects from our results and run the optimization process using only these objects. The independence assumption is then guaranteed to be valid, and the constraint bounds are tailored to those objects that are uniquely affected by that constraint.

We still required a deeper understanding of the affects of changing the bounds on our constraints in SPAM's knowledge base. As stated previously, we cannot exhaustively search the space of input/output pairs due to the long execution times. However, we can sample the space of outputs and draw conclusions from the trends observed in the results. This is the topic of the next section.

5 A Brute-Force Evaluation Method

The results from Section 3 indicated that our embedded evaluation method (in particular, the evaluation function) needed to be improved. However, because the features used to evaluate the functional-area results are related in complex ways, an analytical understanding of the evaluation function is difficult. Therefore, we pursued a more data-driven approach.

As previously cited, SPAM's long run times prohibit the use of exhaustive search methods for refinement. This limitation can be circumvented to a degree by appropriately choosing experiments to run and observing the system's behavior. In this way, we sample the space of possible bounds settings and hence, sample the system's output behavior.

We used our refinement method, without the embedded evaluation portion, as presented in Section 3. We ran five experiments for each data set, allowing SPAM to execute through it's third phase (FA). Each experiment corresponded to a modification of the distance constraint bounds, as follows:

off	constraint disabled;
low	bounds set to minimum value;
optimized	optimal bounds setting arrived at by using adjustment procedure;
original	original bounds setting arrived at by hand tuning;
high	bounds set to maximum value;

We, again, collected statistics on run-time, number of production firings, number of functional-areas generated, and number of correct and incorrect hypotheses included in those functional-areas. In this table, we added a column labeled Coverage. This column indicates the total area covered by the set of generated hangar-building functional-areas. This yields an approximate evaluation, allowing a comparison of the amount of change in the area of coverage from one run to another. Evaluation was done by comparing each run to the *original* bounds settings. These results, with the results of our previous experiments, are collected and presented in Table 8. Entries in the table prefixed with *orient-* are the results for analogous bounds modifications applied to the hangar-building orientation constraints[6]. From this data, we can make several observations.

From the increase in run time (from *off* to *original*), we can see that the distance constraint is having some impact on the results. The increase in the number of correct hypotheses and the drop in the number of incorrects reveals that this constraint is playing a positive role in all cases. The orientation constraint is also seen to be playing a positive role.

Finding the best setting for the bounds of the constraints is a more difficult problem. Again, an analytical evaluation function for this task would be very complex, taking into account relationships between numbers of corrects/incorrects, sizes of functional-areas, run time, as well as other factors. For the Moffett data set, the number of corrects increases, while the number of incorrects increases, but at a slower pace. From this we would conclude that the bounds should be set to the maximum value because the number of corrects increases significantly with only a modest increase in run time. However, doing the same evaluation for DC National implies that the optimized value would be best. The other data sets, San Francisco and Dulles International Airports, show a trend similar to DC.

Overall, such an analysis suggests that the bounds for the distance constraint should be chosen on a case by case basis. This is an important observation for SPAM, as this suggests that a set of specific "case" airport models may more effectively aid interpretation than a single generic airport

[6]We have not yet generated optimized bounds settings for the hangar-building orientation constraint.

Dataset	Bounds Setting	Time (cpu hrs)	# Rule Firings	# of FAs	Total # Correct	Total # Incorrect	Change in # Correct	Change in # Incorrect	Score (Score/Total)	Coverage (10^6 meters^2)
Moffett	original	0.35	106367	14	19	111	—	—	—	1.066
"	off	0.32	92033	13	9	78	−8	−32	+24 (18.5%)	1.058
"	low	0.33	96921	13	9	78	−8	−32	+24 (18.5%)	1.066
"	optimized	0.34	104903	13	11	100	−6	−10	+4 (3.1%)	1.066
"	high	0.39	125868	15	27	159	7	44	−37 (−28.5%)	2.081
"	orient-off	0.33	99533	14	18	96	−1	−15	+14 (10.8%)	1.062
"	orient-low	0.35	107775	15	18	117	−2	3	−5 (−3.8%)	1.456
"	orient-high	0.37	116064	15	20	118	0	5	−5 (−3.8%)	1.458
DC Ntnl	original	0.58	158182	25	42	209	—	—	—	2.153
"	off	0.52	140286	24	32	122	−10	−82	+72 (28.7%)	2.088
"	low	0.56	147793	24	31	121	−11	−83	+72 (28.7%)	2.045
"	optimized	0.57	156453	25	38	167	−4	−42	+38 (15.1%)	2.045
"	high	0.69	207483	25	84	410	42	201	−159 (−63.3%)	2.261
"	orient-off	0.53	145014	25	33	192	−9	−17	+8 (3.2%)	2.076
"	orient-low	0.58	160564	25	48	216	6	7	−1 (0.4%)	2.139
"	orient-high	0.61	175162	25	62	225	20	16	+4 (1.6%)	2.184
SF Intl	original	1.39	320993	49	46	235	—	—	—	3.898
"	off	1.26	281917	20	4	77	−10	−19	+9 (3.2%)	3.548
"	low	1.71	307989	21	4	83	−10	−16	+6 (2.1%)	2.496
"	optimized	1.41	320329	37	21	135	−12	−49	+37 (13.2%)	2.598
"	high	1.73	342905	54	74	383	19	126	−107 (−38.1%)	5.401
"	orient-off	1.60	300127	49	44	227	−2	−7	+5 (1.8%)	3.427
"	orient-low	1.70	323317	45	43	221	−1	−2	+1 (0.4%)	3.497
"	orient-high	1.72	328498	49	43	233	−2	−3	+1 (0.4%)	3.728
Dulles	original	3.45	585115	80	136	576	—	—	—	4.668
"	off	3.00	497261	71	47	322	−81	−213	+132 (18.5%)	4.161
"	low	3.40	556772	71	47	332	−81	−203	+122 (17.1%)	3.583
"	optimized	3.45	575145	77	74	428	−62	−133	+71 (9.9%)	3.583
"	high	3.92	723468	80	266	1138	130	562	−432 (−60.7%)	6.570
"	orient-off	3.16	530973	78	125	511	−10	−60	+50 (7.0%)	4.313
"	orient-low	3.49	595258	78	144	572	9	1	+8 (1.1%)	5.069
"	orient-high	3.57	623175	79	151	596	15	23	−8 (−1.1%)	5.135

Figure 8: Sampled SPAM performance, measured along several dimensions while changing constraint settings.

model. Therefore, selective application of distance constraints, depending upon the dataset or on the particular airport model being used, could lead to improvements in overall runtimes.

The modest improvement in results given the significant changes in the distance constraint bounds also suggests that more constraints may be required for better functional-area results. When analyzing SPAM's runway functional-areas (see [Harvey *et al.*, 1992]), two new constraints (*pointed-toward* and *alongside*) were added to improve the generated functional-areas.

6 Future Work

Though our goal is to improve the interpretations generated by SPAM, we have begun by improving our understanding of how the individual components of SPAM operate and how they interact. We began our investigation by posing several questions about SPAM's constraints. We've been able to show that SPAM's distance constraint plays a positive role in the interpretation process. Choosing an "optimal" value for a single constraint is difficult and sometimes seems to be scene dependent. The effectiveness of both the distance and orientation constraints is measured to be weak, though this verifies one of SPAM's original hypotheses. Finally, we've determined that, of the two constraints looked at so far (distance and orientation), the applicability of both overlaps a great deal. Overall, this work has provided a foundation for understanding the affects of modifying or adding knowledge SPAM, and has helped to determine how SPAM's knowledge might change to improve performance.

There is still much to be done. In the short term, there are several obvious problems that we've not addressed. First, we need to look more closely at the applicability of the constraints and characterize (as suggested in 4.1) the cases where each constraint can be applied. Second, it is unclear if the bounds optimization procedure we developed will extend to non-numeric constraints. Additionally, we need to look more closely at the issue of an evaluation function. Ultimately, we will need to extend the analysis to simultaneously perform refinement across multiple constraints.

Overall, we believe that it will be possible to build a heuristic system that would automate the knowledge-refinement process, similar to the automated system in [Ginsberg *et al.*, 1988, Politakis and Weiss, 1984]. Within such a system, we would like to discover ways not only to improve the current constraints, but to automate methods for determining what new knowledge may be needed.

References

[Buchanan and Shortliffe, 1984] Bruce G. Buchanan and Edward H. Shortliffe. *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project*. The Addison-Wesley series in artificial intelligence. Addison-Wesley, Reading, Massachusetts, 1984.

[Cooke, 1991] Roger M. Cooke. *Experts in Uncertainty: Opinion and Subjective Probability in Science*. Environmental Ethics and Science Policy Series. Oxford University Press, New York, New York, 1991.

[Craw and Sleeman, 1991] S. Craw and D. Sleeman. The flexibility of speculative refinement. In *Proceedings of the Eighth International Workshop on Machine Learning (ML91)*, pages 28–32, 1991.

[Ginsberg *et al.*, 1988] A. Ginsberg, S. Weiss, and P. Poliatkis. Automatic knowledge-base refinement for classification systems. *Artificial Intelligence*, 35:197–226, 1988.

[Harvey *et al.*, 1992] W. Harvey, M. Diamond, and D. McKeown. Tools for acquiring spatial and functional knowledge in aerial image analysis. In *Proceedings of the DARPA Image Understanding Workshop*, pages 857–873, San Mateo, CA, January 1992. Morgan Kaufmann Publishers, Inc.

[Kalp *et al.*, 1988] D. Kalp, M. Tambe, A. Gupta, C. Forgy, A. Newell, A. Acharya, B. Milnes, and K. Swedlow. Parallel ops5 user's manual. Technical Report CMU-CS-88-187, Computer Science Department, Carnegie Mellon University, November 1988.

[McKeown *et al.*, 1985] D. M. McKeown, W. A. Harvey, and J. McDermott. Rule based interpretation of aerial imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(5):570–585, September 1985.

[McKeown *et al.*, 1989] D. M. McKeown, W. A. Harvey, and L. Wixson. Automating knowledge acquisition for aerial image interpretation. *Computer Vision, Graphics and Image Processing*, 46(1):37–81, April 1989.

[Politakis and Weiss, 1984] P. Politakis and S. Weiss. Using empirical analysis to refine expert system knowledge bases. *Aritificial Intelligence*, 22:23–48, 1984.