Efficient Car Recognition Policies

Ramana Isukapalli
101 Crawfords Corner Road
Lucent Technologies
Holmdel, NJ 07733 USA
ramana@research.bell-labs.com

Russell Greiner
Department of Computing Science
University of Alberta
Edmonton, AB T6G 2E8 Canada
greiner@cs.ualberta.ca

Abstract

Many tasks require an imaging system to identify an object, such as the type of a car; in many cases, it is critical to make this identification *quickly*, as well as *accurately*. This paper addresses the challenges of producing recognition systems that consider both of these objectives. In general, an "(recognition) policy" specifies when to apply which "imaging operators", which can range from low-level edge-detectors and region-growers through high-level token-combination–rules and expectation-driven object-detectors. Given the costs of these operators and the distribution of possible images, we can determine both the expected cost and expected accuracy of any such policy. Our task is to find a maximally effective policy — typically one with sufficient accuracy, whose cost is minimal. We compare various ways to produce such policies in general, and show that policies that select the operators that maximize *information gain per unit cost* work effectively.

Keywords: vision, recogition, decision theory, real time systems

1 Introduction

Recognition systems typically try to determine what objects are within a given scene. Of course, it is critical that the recognition systems be *accurate*. It is typically important that the process also be *fast*: For example, to work in real-time, a recognizer examining the frames of a motion picture will have only 1/30 seconds to identify an object. Or consider a web-searcher that is asked to find images of, say, aircraft. Here again speed is critical — and most searchers do in fact sacrifice some accuracy to gain efficiency (*i.e.*, they quickly return a large number of "hits", only some of which are relevant). This paper addresses the challenge of producing a recognizer that is both sufficiently accurate and sufficiently efficient.

Section 2 provides the framework, showing how our framework generalizes the standard (classical) approaches to image interpretation, then presents a formal description of our task: given a distribution of possible images and an inventory of "operators", produce a "policy" that specifies when to apply which operator, towards optimizing some user-specified objective function. It also uses a simple blocks-world example to illustrate these terms. It concludes by describing three different policies that could be used. The rest of this paper demonstrates that one of these policies, "INFOGAIN" (which uses information gain to myopically decide which operator is most useful at each step), is more effective than the other obvious contenders. Section 3 provides an empirical comparison of these approaches in the context of the simple blocks-world situation. Section 4 presents a real-world example: identifying cars from their rear tail light assemblies. The extended [IG01] provides a survey of the related work, placing our work in the context of related frameworks for building [PL94], and analysing [RH92], vision system; and of related approaches, such as information theory [GJ96] and influence diagrams [LAB90]. It also discusses other applications of our framework — in particular, showing that it can be used when recognizing faces, within the modern eigenvector approach.

2 Framework

2.1 Standard Approaches

There are many approaches to recognition. A strictly "bottom-up" approach performs a series of passes over *all* of the information in the scene, perhaps going first from the pixels to edgels, then from edgels to lines, then to regions boundaries and then to descriptions, until finally producing a consistent interpretation for the entire scene. Most "top down", or "model driven", systems likewise begin by performing several bottom-up "sweeps" of the image — applying various low-level processes to the scene to produce an assortment of higher-level tokens, which are then combined to form some plausible hypothesis (*e.g.*, that the scene contains a person, etc.). These systems differ from strictly bottom-up schemes by then switching to a "top-down" mode: given sufficient evidence to support one interpretation, they seek scene elements that correspond to the parts of the proposed real-world object that have yet to be found [LAB90].

Notice the model-based systems have more prior knowledge of the scene contents than the strictly bottom-up schemes — in particular, they have some notion of "models" — which they exploit to be more efficient. We propose going one step further, by using additional prior knowledge to further increase the efficiency of an interpretation system. Consider a trivial situation in which we only have to determine whether or not a "pick cadillac" appears in an image; and imagine, moreover, we knew that the *only* pink object that might appear in our images is a cadillac. Here it is clearly foolish to worry about line-detection or region-growing; it is sufficient, instead, to simply sweep the image with an inexpensive "pink" detector.

$c_{1,1}, t_{1,1}, s_{1,1}$	$c_{2,1}, t_{2,1}, s_{2,1}$			
		$c_{3,3}, t_{3,3}, s_{3,3}$		
				$c_{5,5}, t_{5,5}, s_{5,5}$

Figure 1: A simple image of 25 sub-objects

This illustrates the general idea of exploiting prior knowledge (*e.g.*, which objects are we seeking, as well as the distribution over the objects and views that might appear) to produce an effective recognition process. In general, we will assume our system also has access to the inventory of possible imaging operators. Given this collection of operators, an "recognition policy" specifies when and how to apply which operator, to produce an appropriate interpretation of an image.

Our objective is to produce an *effective* recognition policy — *e.g.*, one that *efficiently* returns a *sufficiently accurate* interpretation, where accuracy and efficiency are each measured with respect to the underlying task and the distribution of images that will be encountered. Such policies must, of course, specify the details: perhaps by specifying exactly which bottom-up operators to use, and over what portion of the image, if and when to switch from bottom-up to top-down, which aspects of the model to seek, etc. These policies can include "conditionals"; *e.g.*, terminate on finding a pink object in the scene. They may also specify applying a *particular* operator only to *select* regions of the image (*e.g.*, seek only near-vertical edges, only in the lower left quadrant of the image). Based on the information available, this recognition policy could then use other operators, perhaps on other portions of the image to further combine the tokens found.

2.2 Input

We assume that our recognition system "*RS*" is given the following information:

⋆ The **distribution** of images that the *RS* will encounter, encoded in terms of the distribution of objects and views that will be seen, etc. (Here we assume this information is explicitly given to the algorithm; we later consider acquiring this by sampling over a given set of training images.)

As a trivial example, we may know that each scene will contain exactly 25 sub-objects, each occupying a cell in a 5×5 grid; see Figure 1. Each of these cells has a specified "color", "texture" and "shape", and each of these properties ranges over 4 values. (Hence, we can identify each image with a $5 \times 5 \times 3 = 75$ tuple of values, where each value is from $\{1, 2, 3, 4\}$.) Moreover, our *RS* knows the distribution over these 4^{75} possible images; see below.

⋆ The **task** includes two parts: First, what objects the *RS* should seek, and what it should return — *e.g.*, "is there an airplane in image" or "is there a DC10 centered at $[\,43, 92\,]$ in the image", etc. In our trivial blocks-world case, we simply want to know which of the images is being examined.

Second, the task specification should also specify the "evaluation criteria" for any policy, which is based on both the expected "accuracy" and its expected "cost". In general, this will be a constrained optimization task, combining both hard constraints and an optimization criteria (*e.g.*, minimize some linear combination of accuracy and cost, or perhaps maximize the likelihood of a correct interpretation, for a given bound on the expected cost).

For this blocks-world task, we want to minimize the expected cost and also have 100% correctness.

⋆ The **set of possible "operators"** includes (say) various edge detectors, region growers, graph matchers, etc. For each operator, we must specify

- its input and output, of the form: "given a set of pixel intensities, returns tokens representing the regions of the same color";
- its "effectiveness", which specifies the accuracy of the output, as a function of the input. This may be of the form: "assuming noise-type W, can expect a certain ROC curve" [RH92];
- its "cost", as a function of (the size of) its input and parameter setting.

When used, each operator may be given some arguments, perhaps identifying the subregion of the image to consider.

Here, we consider three operators: O_c (resp., O_t, O_s) for detecting the "value" of color (resp., "texture", "shape"); each mapping a $[\,x, y\,]$ location of the current image to a value in $\{1, 2, 3, 4\}$. (Note that location is an argument to the operator.) We assume that each operator, when pointed at a particular "cell", will reliably determine the actual value of that property at the specified location, and it does so with unit cost.

For each situation, we assume our recognizer will be given a series of tasks, but will always have the same objective and criteria; *e.g.*, it is expected to look for the same objects in each image, and has a single objective function. ([IG01] considers the obvious generalization, in which the environment could ask different questions for each image, and impose different penalties.)

At each stage, our *RS* will use its current knowledge (both prior information — *e.g.*, associated with the distribution and the operators — and information obtained by earlier probes) to decide whether to terminate, returning some interpretation; or to perform some operation, which involves specifying both the appropriate operator and the relevant arguments to this operator, and then recur.

2.3 Policies

Each *RS* corresponds to a large decision tree, whose leaf nodes each represent a complete interpretation (which is returned), and whose internal nodes each correspond to a sequence of zero or more operator applications, followed by a test on the original data and/or some set of inferred tokens. Each arc descending from this node is labeled with a possible result of this test, and its descends to new node (containing other operators and tests) appropriate for this outcome.

As such *explicit* strategy-trees can be enormous, we instead represent strategies *implicitly*, in terms of a "*policy*" that specifies how to decide, at run-time, which operator to use. Figure 2 shows a general recognition strategy using any

```
$RS_\chi(T = \langle C_{max}, P_{min}\rangle$: TaskSpecification,
      $P(\ )$: Distribution,    $O = \{o_i\}$: Operators,    $I$: Image)
  Initialize cost $C := 0$    Evidence $\vec{O} = \langle\rangle$
  While [$C < C_{max}$ & $P_{min} > \max_x P(S = x \mid \vec{O})$] do
    Select [$o$ : operator; $a$ : arguments]    (based on policy $\chi$, $P(\cdot)$)
        (Note $a$ may include the region to consider)
    Apply $o(a)$ to $I$, yielding $v$
    Extend $\vec{O} := \vec{O} + \langle o(a), v\rangle$
    Update $P(\cdot \mid \vec{O})$, based on result
    Update $C := C +$ Cost[ $o(a)$ ]
  Return Best Interpretation: $argmax_x P(S = x \mid \vec{O})$
```

Figure 2: Recognition algorithm,
for policy $\chi \in \{RandPol,\ BestHyp,\ InfoGain\}$

of the policies. We will consider the following three policies:

Policy RandPol**:** selects an operator randomly.[1]

Policy BestHyp**:** first identifies the object that is most likely to be in the scene (given the evidence seen so far, weighted by the priors, etc.) and then selects the operator that can best verify this object [LHD$^+$93, p370]: That is, after gathering information from k previous operators $\vec{O} = \langle o_1 = v_1, \ldots, o_k = v_k\rangle$, it computes the posterior probabilities of each possible interpretation S_i, $P(S_i \mid \vec{O})$. To select the next operator, BestHyp will first determine which of the scenes is most likely *i.e.*, $S^* = argmax_S\{P(S \mid \vec{O})\}$) and then determines which operator has the potential of increasing the probability of this interpretation the most: Assume the operator o returns a value in $\{v_1, v_2 \ldots, v_j\}$; then o might increase the probability of S^* to $\text{best}(S^*, o) = \max_j\{P(S^* \mid \vec{O}, o = v_j)\}$. Here, BestHyp will use the operator

$$o^*_{BH} \quad = \quad argmax_o\{\ \text{best}(S^*, o)\ \}$$

Policy InfoGain**:** selects the operator that provides the largest information gain (per unit cost) at each time. This policy computes, for each possible operator and argument combination o, the expected information gained by performing this operation:

$$IG(S,\ \vec{O},\ o) = H(S|\vec{O}) - \sum_j P(o = v_j \mid S, \vec{O})\, H(S|\vec{O}, o = v_j)$$

where $H(S|O) = \sum_i P(S = s_i \mid O)\ \log P(S = s_i \mid O)$ is the current entropy of the distribution over the interpretations S given the evidence E, for $E = \vec{O}$ or $E = \{\vec{O}, o = v_j\}$.

InfoGain then uses the operator

$$o^*_{IG} \quad = \quad argmax_o\{\ EIG(\vec{O},\ o)\ /C(o)\ \}$$

which maximizes $[EIG(\vec{O},\ o)/C(o)]$, where $C(o)$ is the cost of applying the operator.

3 Simple Experiments: Blocks World

This section presents some experiments using the simple blocks world situation presented above, designed simply to illustrate the basic ideas, and to help us compare the three policies described earlier.

Basic Experiment: For each "set-up": We first generated a set of 1000 images, each with 25 sub-objects, by uniformly assigning values for color, texture and shape for each of the sub-objects randomly, for each of 1000 images; we also assigned each a "prior distribution" p_i to these images (this corresponds to taking an empirical sample, with replacement). For each run, we randomly select one of the 1000 images to serve as a target for identification, then used each of the three policies to identify the image.

After observing $O_k = \{o_1 = v_1, o_2 = v_2, \ldots, o_k = v_k\}$ based on the operators in the first k iterations, RandPol randomly selects a cell $C[i,j]_{RP}$ and an operator o_{RP} to probe the value for a property (color, texture, etc.), insisting only that o_{RP} was not tried earlier on $C[i,j]_{RP}$ in any previous iterations of this run. BestHyp chooses a cell $C[i,j]_{BH}$ and an operator o_{BH} to maximize the posterior probability of the most likely image, as explained earlier. Finally, InfoGain chooses $C[i,j]_{IG}$ and o_{IG} such that $EIG(\vec{O}, o_{IG})/C(o_{IG})$ is the maximum over all possible cell and operator combinations. For each of these policies, the posterior probability is updated after applying the chosen operator on the cell. The process is repeated until the image is identified — *i.e.*, all other contenders are eliminated.

We considered 10 set-ups (each with its distribution over objects), and performed 5 runs for each set-up. Over these 50 runs, RandPol required on average 5.82 ±0.27 probes, BestHyp 5.44 ±0.32 probes and InfoGain 5.32 ±0.13. InfoGain is statistically better than the other two policies, at the $p < 0.1$ level.

Accuracy Curve: We can consider other evaluation criteria. If the sensors are noise-free, any policy should eventually identify the target, if there is no limit on cost. However, it is impractical to expect unlimited cost. The following study investigates how the accuracy of identifying an image varies for a given cost, for each policy.

In this set-up we generated 500 images randomly, and randomly picked one, call it I^*, as the target image to be identified. We varied the number of probes allowed to identify this image from 1 through 8 and noted the posterior probability of I^* (given these probes) as a function of the number of probes. We repeated this for 10 different targets for this set-up and then repeated the entire procedure for 5 randomly generated set-ups. Figure 3(a) shows the results. Once again, InfoGain clearly identifies the target with a higher probability for a given number of probes. Equivalently, InfoGain can retrieve more information about the target for a given cost. BestHyp performs nearly as well, but RandPol trails the other two policies, *i.e.*, for a given cost it cannot identify the object with the same confidence as the other two.

Identify an image with HIGH (but $<$ 1.0) probability: Identifying an image with very high probability can often be very expensive. However, we may be interested in identi-

[1] We will use the term "operator" to refer to the operator *instantiated with the relevant arguments*. Also, we will further abuse notation by writing $o_i = v_i$ to mean that the value v_i was obtained by applying the (instantiated) operator o_i to the image.

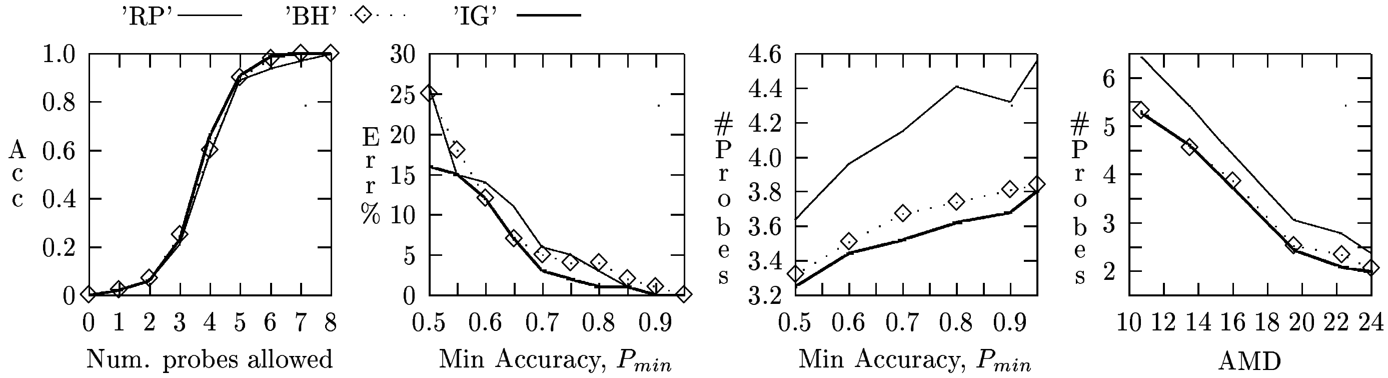

Figure 3: (a) Probes vs. Accuracy (b) Min. Accuracy vs. Error (c) Min. Accuracy vs. # Probes (d) AMD vs. # Probes

fying the image with a "reasonably high probability", thus reducing the cost of identification. To study this effect, we generated 100 images randomly, and randomly picked one as the target. We fixed the required "degree of accuracy" for identifying the target at some value $P_{min} \in [0, 1.0]$. While running the policy, whenever any image's probability exceeds P_{min}, the *RS* system returned the (possibly incorrect) image. We repeated this for 10 different targets and ran this for a total of 10 different set-ups. We varied the values of P_{min} between 0.5 and 0.95.

There are two ways to measure the results: First, the *RS* did not always find the correct image; Figure 3(b) shows the number of errors for the various policies (as a function of the minimum accuracy required, P_{min}). Second, we can ask how many probes were required to make a decision; see Figure 3(c).

Once again, INFOGAIN clearly requires the least number of probes, on average, to identify the target with any accuracy; it is also the most accurate. BESTHYP requires slightly more probes for a given accuracy than INFOGAIN, while RANDPOL requires many more probes. Here the difference between INFOGAIN and RANDPOL is significant at the $p < 0.2$ level.

Image Difference: Clearly, the number of probes required depends critically on the difference between the "correct" image and the nearest contender. This quantity may well depend on the "average Hamming distance"; *i.e.*, for image I_1, we compute the Hamming distance between I_1 and I_j for $j = 2..n$ (here viewing each image as a 75-tuple of values), and let $Sep(I_1)$ be the smallest of these values. We similarly computed $Sep(I_k)$ for the other $(n-1)$ images; then let $AMD(\{I_k\}) = \sum_i p_i Sep(I_i)$ be the average minimum Hamming distance.

We ran the experiment mentioned above for various set-ups, measuring this "average minimum Hamming distance" (AMD) vs the number of probes needed for each identification policy; see Figure 3(d). Here, we varied the number of images n, for each set-up from 10 to 1000 in a total of 15 set-ups to obtain different Hamming Distance values. (Clearly, as the number of images increases, $Sep(I_k)$, and hence AMD, decreases.) Throughout this range, we see that INFOGAIN needed the least number of probes for a given AMD value to accurately identify the object, followed by BESTHYP and RANDPOL.

4 Scaling-Up: Identifying Cars from *RTLA*

The example used in the previous section was intentionally very simple, for pedagogic reasons. In general, we would like a system capable of finding cars, airplanes and any other general objects that we encounter everyday. Here, we clearly need to go beyond the simple operators shown above. The framework needs to accommodate a set of operators which can do more complex things to be useful for this task (like detecting edges from pixels, grouping low level features to perceptually significant tokens, finding regions of the same color in an image, etc.). This section shows that this current framework is sufficient to handle a useful (if still fairly simple) real-world application: identifying a type of car from a particular type of image.

4.1 Framework

Our goal here is identifying the make and model of a car (*e.g.*, Toyota Corolla, Nissan Sentra, Honda Civic, etc.) given an image of the car showing its "rear tail lights arrangement" (*RTLA*); see Figure 4. These results have many obvious applications, both in providing a way to help traffic surveillance by object identification [HR96; KWM93], as well as providing an extension to the indexing methods used by various researchers [FJ92] to identify objects.

This problem is also well-suited to our techniques. First, to specify the **task**, we provide the *RTLA*s for various types of cars, which each precisely specify the number of regions, the color of each region, their relative sizes (in terms of the number of pixels as seen in an image) and their geometrical arrangement with respect to each other. It also specifies the expected accuracy of identifying a car and the maximum cost that can be incurred.

The **distribution** consists of the vehicles (cars, pick-up trucks and vans) that are typically found in a parking lot or on the road — here, we gathered these images from various New Jersey parking lots. Here, our system is given the relevant **distribution** information: (1) the prior probability of seeing each type of vehicle $\{p_i\}$; (2) the distribution, over the 2-D image space, for where the car's *RTLA* can be seen; and (3) how these tail-lights will actually be seen (wrt the shape, color and size) for each type of car.

We use three **operators** for this task, each taking a region R as input:

- $o_{shape}(R)$ finds the shape of R, in two steps. It first finds the corners of the region (using "SUSAN" [Smi96]), then

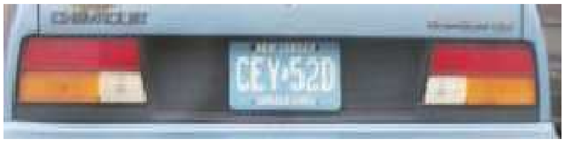

Figure 4: Tail lights of Chevrelot

analyzes the extracted corners to return a geometrically significant shape. As corners are identified by scanning the region twice, the computational cost for this step is $2n$ (where n is the number of pixels in the region). The second step (analyzing the corners to determine a shape) involves sorting the corners based on the x-coordinate (or the y-coordinate) and doing a simple geometric analysis. Hence, this total computation can be done in $O(k^2 + n)$ computations where k is the number of corners; as the number of corners is bounded, we found this cost is well approximated by $2n + 80000$ microsecs.

- $o_{color}(R)$ for detecting the color of R, as a $\langle rgb \rangle$ triple. As the image is segmented to give homogenous regions, this operator can simply return the color at any pixel. We empirically found the cost of applying this operator is just 1 microsec.
- $o_{size}(R)$ for detecting the size (number of pixels) of R. As the area of every region is noted during region extraction, the cost of using this operator is again 1 microsec.

4.2 Recognition Procedure

Figure 2 shows the algorithm that is used to identify a type of car from a given *RTLA*. The input to each such algorithm (for each χ policy), is

- task specification T, specifying required accuracy of identification, upper limit of the cost of identification, etc.
- distribution D of cars, giving the prior probability of seeing any car's *RTLA*, and its appearance in an image
- the set of imaging operators O
- the color-segmented image I

Given these inputs, the first step of recognition is region extraction. The current system automatically extracts all regions from the entire segmented image. (Later versions will let the policy decide where to extract regions, etc.) Once this is done, the *RS*, following its policy (which is one of {RANDPOL, BESTHYP, INFOGAIN}) begins an iterative process: At each step, it choses an imaging operator $o \in O$ and a region R in I, (possibly) based on its current understanding of the distribution of possible cars – *i.e.*, the posterior distribution $P(\ Car\ =\ x\,|\,\vec{O}\ =\ \vec{v})$ after finding the values $\vec{v}$ for the executed operators $\vec{O}$. As explained in Section 2.3, RANDPOL selects an operator o_{Rand} and a region R_{Rand} randomly with the only condition that o_{Rand} was not tried on R_{Rand} earlier. BESTHYP chooses an operator o_{BH} and a region R_{BH} such that this combination can best confirm the most likely candidate (car). INFOGAIN chooses an operator o_{IG} and a region R_{IG} such that o_{IG} has the maximum information gain per unit cost (*i.e.*, $EIG(O_k, o)/C(o)$) of all the possible combinations. The o operator is applied over R, and the *RS* then updates its distribution based on the outcome observed. The process is repeated until a car is identified with sufficient accuracy (as specified in T) or the total cost of the operators applied so far, exceeds the maximum limit of the cost specified in T.

4.3 *RTLA* Experiments

With this framework, we performed experiments similar to the ones done in the blocks-world domain. We created a database of thirty different cars from NJ parking lots. We emprically found that our shape detecting operator is 90% accurate (*i.e.*, it detects the shape correctly 9 out of 10 times), and the color and size detecting operators are perfect (they always return the correct answer). Here, we first considered $P_{min} = 0.9$ and $C_{max} = 500$ msec. For each set-up, we assigned random probability values to each of the 30 different cars. For each run we selected one of these cars as the target and successively attempted to identify it using each of the three policies, noting both the cost and correctness of each policy.

We then picked a new car as target, repeating the whole process for a total of 10 targets. We repeated the process for a total of 5 different set-ups (*i.e.*, a total of 50 runs) and found the accuracy and mean cost of identification for each of the three policies.

We found that INFOGAIN had an accuracy of 96%, followed by BESTHYP 72% and RANDPOL 56%. INFOGAIN also has the least cost computed in terms of CPU time, averaging 93.0 ± 22.67 msec. per run, followed by BESTHYP 139.0 ± 52.83 msec. and RANDPOL 720.80 ± 159.77 msec.[2]; which is significant at the $p < 0.1$ level. (RANDPOL is a lot more expensive than the other two policies because it randomly chose the most expensive shape detecting operator (o_{shape}) more often than the other two policies, and applied it indiscriminately to uninteresting regions.)

Accuracy for a given cost: Here, we set a bound on the total cost C_{max} (and left $P_{min} = 0.9$). allowing the recognizer to make a probe only if the sum of the cost of the current probe and the prior probes was under C_{max}, We varied C_{max} from $[1 - 1000]$ msec. per run and noted the accuracy of each policy for each value. As in the previous case, we ran the experiments with each policy for various different C_{max} values, for a total of 5 different set-ups with 10 runs in each set-up. The percentage of correct identifications for each cost is plotted against the cost; see Figure 5(a). BESTHYP has a good accuracy for low costs, but INFOGAIN has much better costs than the other two policies for reasonably high costs.

Recognition with high probability: Here we consider identifying the object with a "reasonably high probability". For each run, one of the cars was randomly picked as the target to be identified. The degree of accuracy with which the car needs to be identified, P_{min} was varied from 0.20 to 0.90. While running each policy whenever the probability of any car, say c^*, exceeds P_{min}, it is returned (possibly incorrectly) as the target. We repeated this experiment with 10 different runs in each set up, for a total of 5 set-ups. We evaluated the experimental results in the same two ways dis-

[2] All these programs were run on a PC running Liunx 2.0.35 OS on an Intel Pentium 200 MHz. processor with 32 MB. RAM.

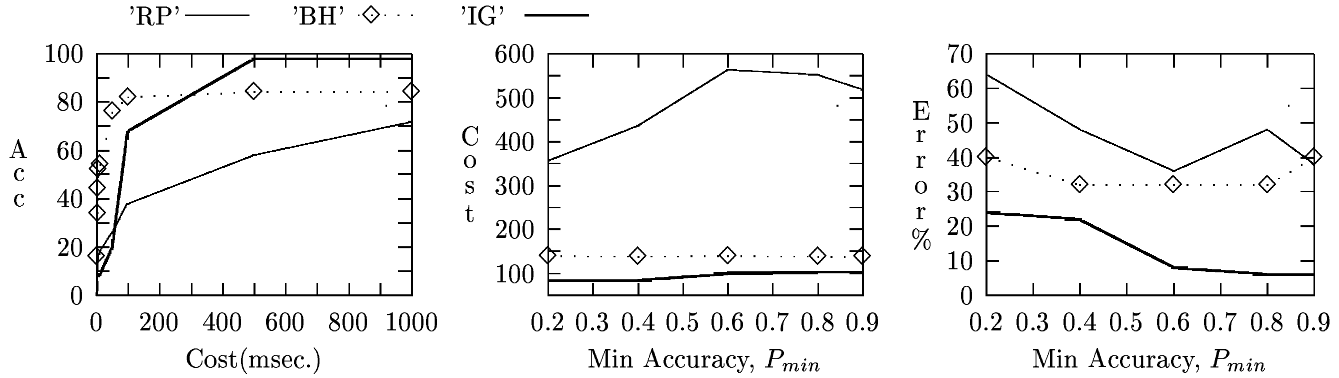

Figure 5: (a) Cost vs. Accuracy (b) Min. Accuracy vs. Cost (c) Min. Accuracy vs. Error

cussed above: (1) Plotting P_{min} against the cost required to identify a car with probability $\geq P_{min}$ (Figure 5(b)) — note INFOGAIN has the least cost followed by BESTHYP and then RANDPOL. RANDPOL has such a high cost as it chooses the most expensive operator (o_{shape}) quite often and applies it on several regions without gaining any useful information. (2) Plotting P_{min} against the percentage of wrong identifications, for each of policy (Figure 5(c)). Again, INFOGAIN has the least number of errors for a given accuracy, followed by BESTHYP and RANDPOL.

5 Conclusions

Future Work: While our results in the cars domain show that our ideas can be applied to a complex, real-world domain, there are a number of extensions to further scale up our approach. Some are relatively straightforward: *e.g.*, extending the set of operators to cover more features; this will help deal with more complex objects, like complete cars (not just *RTLA*s). We were also able to use this framework, *mutatis mutandis*, to deal with alternative approaches to recognition – *e.g.*, eigenfaces [TP91], and even eigenfeatures [PMS94], for face-recognition; see [IG01].

In other contexts, however, we will need to deal with thornier issues, such as operators that rely on one another. This may be because one operator requires, as input, the output of another operator (*e.g.*, a line-segment produces a set of tokens — notice this precondition-situation leads to various planning issues [CFML98]), or because the actual data obtained from one operator may be critical in deciding which next operator (or parameter setting) to consider next: *e.g.*, finding the fuselage at some position helps determine where to look for the airplane's wings.

Clearly we will need to re-think our current myopic approach to cope with these multi-step issues; especially as we expect heuristics will be essential, as this task is clearly NP-hard [Sri95]. Finally, all of this assumes we have the required distribution information. We are now beginning to explore techniques for acquiring such information from a set of training images.

Contributions: This paper has three main contributions: First, it provides a formal foundation for investigating *efficient* image interpretation, by outlining the criteria to consider, and suggesting some approaches. Secondly, our implementation is a step towards *automating* the construction of effective recognition systems, as it will automatically decide on the appropriate policies for operator applications, as a function of the user's (explicitly provided) task and the available inventory of operators. Finally, it presents some results related to these approaches — in particular, our results confirm the obvious point that information gain (as embodied in the INFOGAIN policy) is clearly the appropriate measure to use here — and in particular, it is better than the BESTHYP approach. This observation is useful, as there are deployed imaging systems that use this BESTHYP approach [LHD+93].

References

[CFML98] S Chien, F Fisher, H Mortensen, and E Lo. Using ai planning techniques to automatically reconfigure software modules. In *Lecture Notes in CS*, 1998.

[FJ92] P.J. Flynn and A.K. Jain. 3-d object recognition using invariant feature indexing of interpretation tables. *CVGIP*, 1992.

[GJ96] G Geman and B Jedynak. An active testing model for tracking roads in satellite images. *IEEE PAMI*, 1996.

[HR96] R Huang and S Russell. Object identification: A bayesian analysis with application to traffic surveillance. *Artificial Intelligence*, 1996.

[IG01] R Isukapalli and R Greiner. Efficient interpretation policies. Technical report, University of Alberta, 2001.

[KWM93] D Koller, J Weber, and J Malik. Robust multiple car tracking with occlusion reasoning. Technical report, UCB, 1993.

[LAB90] T S Levitt, J M Agosta, and T O Binford. Model-based influence diagrams for machine vision. In *UAI*, 1990.

[LHD+93] T S Levitt, M W Hedgecock, J W Dye, S E Johnston, V M Shadle, and D Vosky. Bayesian inference for model-based segmentation of computed radiographs of the hand. *Artificial Intelligence in Medicine*, 1993.

[PL94] A R Pope and D Lowe. Vista: A software environment for computer vision research. In *IEEE CVPR*, 1994.

[PMS94] A Pentland, B Moghaddam, and T Starner. View-based and modular eigenspaces for face recognition. In *IEEE CVPR*, 1994.

[RH92] V Ramesh and R M Haralick. Performance characterization of edge operators. In *Machine Vision and Robotics Conference*, 1992.

[Smi96] S Smith. Flexible filter neighbourhood designation. In *Proc. 13th Int. Conf. on Pattern Recognition*, 1996.

[Sri95] S Srinivas. A polynomial algorithm for computing the optimal repair strategy in a system with independent component failures. In *UAI*, 1995.

[TP91] M Turk and A Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 1991.