Information Acquisition and Fusion in the Mobile Perception Laboratory

Bruce A. Draper Shashi Buluswar
Allen R. Hanson Edward M. Riseman
Dept. of Computer Science
University of Massachusetts
Amherst, MA., USA. 01003*

1 Introduction

The development of vision systems capable of robustly interpreting complex imagery has been an elusive goal. Systems which rely on a single sensor, or even on a single representation of derived image data, are fundamentally limited by their restricted view of the data and by the unreliability of abstracted information. In order to attain the generality required in real-world situations, systems will have to incorporate redundant views of the image data, both from multiple sensors and from multiple processes operating on the image data. In order to increase the reliability of system processes, transformations of the data must take place in small steps, introducing the requirement for multiple levels of representation. Finally, the true test of any vision system is its performance over time and over diverse sets of input; research efforts must begin to take into account the kind of information that is required from a vision system and the overall system architecture in which it will be embedded.

The University of Massachusetts Mobile Perception Laboratory (MPL) is an autonomous outdoor vehicle, similar to CMU's NAVLAB II [24], that was built by UMass as an experimental testbed for high-level computer vision. Our goal in developing MPL is the integration of many of the vision algorithms developed over the past decade, at UMass and elsewhere, into a system which is capable of exhibiting useful goal-oriented autonomous navigation in real-world scenarios.

To accomplish such high-level tasks, MPL has to acquire many types of information about its environment, not all of it image related. Rather than performing sensor level fusion of the image data, we have focused on the *types* of information required and the *representations* needed to express them. The problem, as we see it, is to integrate the information needed for a specific task (using task-specific and general constraints) by combining the appropriate representations at the appropriate time, whether they are derived from different sensors or from different interpretation techniques applied to a single sensor. We refer to this task as information fusion, rather than sensor fusion.

2 The Mobile Perception Laboratory (MPL)

The Mobile Perception Laboratory is a significantly modified U.S. Army HMMWV ambulance (see Figure 1), with actuators to control steering, braking and acceleration. MPL is a completely self-contained system with generators, air conditioners and computing facilities for autonomous operation. All physical

*This work was supported by Rome Labs under contract F30602-91-C-0037 and by DARPA/TACOM under contract DAAE07-91-C-R035

modifications to the HMMWV, with the exception of sensor and computing systems installation, was performed by RedZone Robotics, Inc., a Pittsburgh-based firm specializing in custom robotics. The computers and sensors were added later at the University of Massachusetts.

Figure 1: The UMass MPL

2.1 Physical Lay-out

The internal layout of the MPL is shown in Figure 2. The vehicle contains two on-board programming stations with color display capabilities. The driver's side of the vehicle contains one of the programmers stations, the 10KW generators, and the power conditioners. On the passenger's side there are four enclosed and air-conditioned computing frames with 19" racks for the on-board computers. The first frame holds a 6u VME cage for the steering, brake and throttle controllers, as well as an M68020 processor for driving and monitoring the individual controllers. The second cage contains a four-processor SGI IRIS 4/300, which is the vehicle's primary computational engine. The third cage holds a second 6u VME cage for the Datacube image processing equipment (a MaxVideo20, a Digicolor digitizer and an ROI-Store) which is connected to the SGI by a bus-to-bus repeater. The fourth cage is for future growth, and currently houses video recording equipment; it is reserved for the Image Understanding Architecture (IUA) currently being jointly developed by UMass, Amerinex Artificial Intelligence Corp., and Hughes Research Laboratories [28].

2.2 Sensor Configuration

The MPL's sensor configuration is designed to be highly flexible. In addition to odometry, MPL has two sensor platforms. The first is a long, enclosed box with a glass front mounted on the leading edge of the cab for fixed, forward-looking sensors. At present, this contains a pair of stereo black-and-white video cameras and a color CCD camera for road following. The second platform is the Staget, a mechanically-stabilized

Figure 2: Internal Layout of the MPL

platform supplied by TACOM with controllable pan and tilt. The Staget is mounted above and behind the forward-looking sensor box, to give it an unobstructed 360° field of view. It currently has a color video camera and a FLIR sensor, both with controllable focus and zoom. Plans are currently underway to add a LADAR range-finder as the sixth sensor.

3 Information Fusion

The MPL hardware platform was designed and constructed to support research and development in the area of robust autonomous navigation. The long term goal of this effort is to build a system capable of performing high level reasoning tasks (for example, achieve the goal "stop at the third house on the left."), generating plans for achieving the goal, and executing the plan(s) by incorporating perceptual reasoning capabilities within the actions performed by the vehicle. The vehicle must be capable of interacting with its environment in an intelligent way by utilizing diverse forms of information that may be available, such as maps, partial models of objects and places within the environment, and knowledge of its own capabilities and limitations. It must know when its systems fail, such as getting lost, and what to do to recover from various failure modes. In unknown environments (where little or no a-priori information is available), the vehicle must be capable of building internal models of sufficient quality to achieve subsequent navigation goals within that environment.

To achieve these goals, new and existing techniques for planning, reasoning, perception and action must be integrated into a complete system and information in many different forms must be fused into coherent structures. The emphasis here, however, is on navigation using landmarks to determine position and some of the generic tools constructed to support information fusion during this process. Landmarks are defined to be visually recognizable stationary objects for which a partial model exists. The model may be geometric, as in the case of a particular building or sign, or they may be non-geometric, as in the case

of a particular group of trees or rocks.

Underlying the notion of a landmark is that of an object; landmarks are objects in the traditional sense, and landmark recognition shares many of the problems of general object recognition in computer vision. MPL must be capable of recognizing both modeled objects, such as landmarks, and unmodeled objects such as obstacles. It must also be able to recognize classes of objects, like buildings, and specific objects, like the UMass football stadium. It also must have the ability to recognize both man-made artifacts, like stoplights, and natural objects, such as trees and hills.

To achieve this degree of generality, it is necessary to integrate many types and forms of information. In the case of objects, relevant information includes shape, texture, color, depth, and context. Extraction of these properties from images has proven to be difficult and it is commonly believed that the information derived from many different processes operating at very different *levels of representation* will be required. To recover shape, for example, it is useful to reason not only about images, but about points, lines and surfaces as well. Other types of information require other representational hierarchies, but they all require multiple levels of representation. This implies that information fusion is a generic problem in vision, and is not restricted to data fusion, algorithm combinations, or representations.

In the evolving MPL system, information at various levels of representation is transformed into information at other levels of representation by means of *visual procedures*, some of which control sensors, some of which control virtual sensors (such as stereo), and some of which operate on data representations extracted by previous operations. Visual procedures create, refine, or combine data representations called *tokens*, which are symbolic descriptions of visual properties at the various abstraction levels. Under this model, the information fusion problem becomes one of coherently and consistently integrating the information contained in the multiple representations into tokens that describe the current environment in a way that is relevant to the current task.

To illustrate this concept, we first describe one strategy for information fusion in some detail, showing how color, shape and position information can be combined to accomplish the task of landmark-based navigation. This is followed by a description of the generic tools being constructed to deal with information fusion requirements in vision in general.

3.1 Landmark-based Navigation

As an autonomous vehicle moves, its estimated position in the world becomes less and less accurate as odometry errors accumulate. One solution to this problem is *landmark-based navigation*, in which the vehicle periodically compares what is sees to landmarks contained in a map or model, updating its position by locating itself relative to them. For the MPL, the basic approach is to utilize the initial position estimate and information about the color and shape of one or more known landmarks in order to determine the position and orientation of the vehicle relative to them.

It should be noted that there are other mechanisms for locating the position of a vehicle within a global frame of reference. Global Positioning Satellite (GPS) systems can determine positions expressed as longitude and latitude to within a few meters. Radio beacons or other forms of electronic signposts can be set up in controlled environments to determine a vehicle's position. In both cases, however, the vehicle is dependent upon external systems.

The principle advantage of landmark-based navigation over these other technologies, however, is that it allows an autonomous vehicle to directly interact with objects (landmarks), rather than indirectly through global map coordinates. As a result, it is not as susceptible to errors in the global map. For example, if the task is to go to building X using a GPS system, then the MPL has to look up the coordinates of building X on its map and use the GPS to drive to those coordinates. If the position of the building on the map is incorrect, however, there is no way for the vehicle to recover from, or even notice, this error. Using landmark-based navigation, on the other hand, the vehicle will go to the proper position relative

to building X, even if the map contains small errors. In fact, if enough other landmarks are visible, a landmark-based navigation system should even be able to notice and correct the error in the global map.

3.1.1 2D vs. 3D Landmarks

Landmark-based navigation can be accomplished by recognizing landmarks in two dimensions or three. 2D landmark recognition systems identify and determine the image position (and hence line of sight) of landmarks, but make no attempt to determine the exact 3D position and orientation of a landmark relative to the camera. Such systems determine the position of the camera by triangulating among several landmarks, and work well in landmark-rich environments with long, unobstructed views.

In MPL, on the other hand, we are working on three-dimensional landmark recognition, where a landmark is any object that remains fixed relative to the global frame of reference and can be modeled in terms of straight line segments. At the core of the approach is a pose determination algorithm by Kumar and Hanson [19] that determines the position and orientation of the camera (vehicle) relative to the landmark, given a set of 3D line segments (from the model), a corresponding set of 2D (image) line segments, and an initial estimate of the vehicle's (camera's) position. The algorithm finds the position and orientation of the camera (vehicle) that minimizes the sum of the residual squared error between model line segments and the planes formed by the corresponding image line segments and the camera focal point.

The pose algorithm is therefore able to fix the position and orientation of the vehicle/camera system in a global frame of reference by determining its position and orientation relative to a single landmark. It assumes, however, that the correspondence between image lines and model lines is given, even though finding the image-to-model correspondence can be a difficult and computationally expensive process. Grimson, for example, has shown that constraint-based, depth-first tree search matching algorithms are exponentially expensive in the number of model lines [16]. We minimize the cost of finding the data-to-model correspondence by using a local search algorithm for geometric model matching developed by Beveridge [3], but nonetheless experiments suggest that the cost of matching image lines to data lines is still $0(m^2d^2)$, where m is the number of model lines and d is the number of data lines [3].

3.2 Using Color to Focus Attention

The computational cost of matching model lines to data lines implies, at a very practical level, that it is infeasible to match a model to the entire set of lines extracted from a typical outdoor scene (e.g. Figure 3). Such scenes simply produce too many line segments. It is therefore critical to focus the matcher's attention on a smaller, selected set of line segments that contain the landmark. In MPL's landmark-based navigation behavior, we use color information to focus the search for a landmark.

In particular, MPL focuses the search for landmarks by classifying pixels according to whether or not they match the expected color of the landmark, and then forming *regions of interest* (henceforth *regions*) out of connected components of landmark pixels. Line segments are then extracted from the regions and matched to the landmark model.

As a general approach to object recognition in complex imagery, pixel by pixel classification has not met with much success. However, pixel classification is being used in a very different way here, and there are two main reasons why this approach works:

- **Non-parametric Classification Techniques.** The last decade has seen a great deal of progress in non-parametric classification techniques such as decision trees and neural networks. These techniques provide a higher level of performance than earlier methods because they do not make assumptions about independence of features or gaussian distributions that are typically violated by the data.

- **Classification as a Focusing Mechanism.** MPL does not use the pixel labels as a solution, but rather as a focusing mechanism for the later model matching process. As a result, false positives

(in which non-landmark pixels are erroneously classified as landmarks) can be easily tolerated; the line-based model matcher will discover that there is no landmark in the corresponding region and reject it. False positives increase the cost of landmark recognition, but do not significantly degrade its performance. As a result, the classifier can be tuned to produce one-sided errors, making extra false-positive mistakes to avoid making false negatives [9], and the system will perform well even when the classifier is not highly accurate.

3.3 Multivariate Decision Trees

The non-parametric classification technique used by MPL is the multivariate decision tree, in which feature space is recursively subdivided by linear threshold units (LTUs). Each division attempts to separate, to the extent possible, landmark pixels from non-landmark pixels. If the two sets of pixels (landmark and non-landmark) are linearly separable, the LTU will separate them and the multivariate decision tree is completed. If the two sets are not linearly separable, the LTU divides feature space so as to separate them as much as possible, and the multivariate decision tree recursively trains new LTUs on the two halves of feature space. The result, therefore, is a tree of linear threshold units, dividing feature space into arbitrary (multi-dimensional) polygons (see [25, 9] for more detailed descriptions).

3.4 Learning Weights in the Linear Threshold Units.

Several methods exist for learning tests in a linear threshold unit. Brodley and Utgoff [5] discuss four such methods: the Recursive Least Squares (RLS) [29], the Pocket Algorithm [15], Thermal Training [14], and CART's coefficient learning method ([4]). The RLS method is recommended for dual-class classification.

The RLS algorithm is a recursive version of Gauss' Least Squares algorithm, which minimizes the mean squared error between the estimated and true values $\Sigma(y_i - \hat{y}_i)^2$ over a number of training instances, where y_i is true value of instance i and y_i is its estimated value.

RLS incrementally updates the weight vector W according to

$$W_k = W_{k-1} - K_k(X_k^T W_{k-1} - y_k),$$

where W_k is the weight vector instance k, of size n, W_{k-1} is the weight vector for instance $k-1$, X_k is the instance vector (X_k^T is X_k transposed), and y_k is the class of the instance $K_k = P_k X_k$, P_k is the $n \times n$ covariance matrix for instance k, and

$$P_k = P_{k-1} - P_{k-1} X_k [1 + X_k^T P_{k-1} X_k]^{-1} X_k^T P_{k-1}.$$

RLS requires that the initial weights be set (to 0 if little is known about the weights).

The covariance matrix should reflect the uncertainty in the weights. If the weights are initialized to 0, the matrix should consist of 0 values everywhere except along the diagonal, which should be set to a very large value. Young [29] recommends that the diagonal be initialized to 10^6.

3.5 Building the Tree

The optimal discriminant function can be determined by training over a set of known instances. If a set of instances is linearly separable, a single test is sufficient for separation. Typically, however, a large set of instances is seldom linearly separable. In order to classify such instances a decision tree must be built, which recursively creates hyperplanes of different orientations that separate the non-homogeneous partitions resulting from a previous attempt at separation. A homogeneous partition resulting from a separation is, by definition, non-separable, and is labeled as a class. A decision tree therefore consists of nodes that are either decisions or classes. The LTU is therefore a discriminant function in the form of a

multivariate decision tree; the resulting tree is much more compact and efficient than a tree comprised of univariate decisions [25].

Figure 3: An outdoor scene comprising of brick buildings and a natural background

3.6 Performance

Initial tests indicate that the multivariate decision trees are surprisingly accurate, even when the feature space is restricted to the RGB values of the pixels (after smoothing). Figure 4shows the result of looking for a brick building in the image shown in Figure 3, with very small regions suppressed. Notice that the only two large regions in the image correspond to the two brick buildings in the scene. (The multivariate decision tree had been trained on similar images taken from a nearby location.) Draper et. al. [9] shows similar results on another set of outdoor images.

One of the great advantages of classifying pixels from RGB data is that the classification can be done at frame rate on processors such as the Datacube MaxVideo20. This permits attention to be focused on interesting regions in real time, greatly reducing the total computational cost of landmark recognition.

4 General Tools for Information Fusion.

The landmark-based navigation strategy discussed above is one example of information fusion at the level discussed earlier. Here, odometry is used to generate an initial estimate of the vehicle's position. This estimate is used to index into the global map and to select which landmark should be looked for. MPL then points the camera in the expected direction, and classifies each pixel according to whether its color matches that of the selected landmark, producing focus of attention regions. Line segments are then extracted from

Figure 4: Binary image focussing attention on the brick buildings

each region and used as input to the geometric model matcher (along with the initial position estimate). The set of model line/image line correspondences are then used by the pose determination algorithm to determine the position and location of the vehicle. In this way, odometry, color and shape information are brought together to accomplish a specific task.

This landmark recognition strategy is, however, only one example of fusing information for a specific task. Other tasks include using natural objects with irregular boundaries as landmarks, and acquiring models of new landmarks. For these tasks, other approaches will be necessary. Therefore, we are developing a more flexible approach to information fusion based ultimately on learning. The first step is to organize the various types of data derived from images as sets of *tokens* in a hierarchically structured visual database. Visual procedures, or prescribed sequences of visual procedures, with built-in control information, can then be viewed as operators over the tokens and token sets. To support these capabilities, a new version of the Intermediate Symbolic Representation (ISR) [7] has been developed specifically for the MPL [10]. Due to the real time constraints of navigation, the new ISR (called ISR3) has been implemented as an in-memory database for visual information. Strictly speaking, it is a misnomer to call ISR3 a database; although it does have facilities for writing data sets to files and reading them back again, file access is too slow a data transfer mechanism for real-time vision, as required for MPL. ISR3 keeps data in memory, and should therefore be called a data store.

4.1 The Intermediate Symbolic Representation (ISR): Data Repository for Information Fusion

The ISR is a symbolic database for visual data that was originally developed at the University of Massachusetts [7], a version of which has since been incorporated in the KBVision system from Amerinex

Artificial Intelligence. ISR was motivated by the belief that computer vision requires more than image-like arrays of data; it depends on symbolic representations of abstract image events such as lines, regions and surfaces and on mechanisms for efficiently accessing these elements under various types of constraints (such as spatial proximity). Visual procedures operate on these symbolic records, called *tokens*, or on sets of tokens, and generally produce new tokens, or fill in fields of existing tokens, or both.

The ISR serves as the data repository at the center of an information fusion system. Visual modules access data, in the form of tokens, in the ISR and create new tokens which they add to the database. What distinguishes the ISR from other systems for the management of visual information is that its data representations and access routines have been optimized for computer vision [10]. In particular, ISR3 provides access functions for retrieving tokens by spatial location and feature value, as well as by name. Such access to tokens is important because tokens are rarely isolated entities; typically, they have important symbolic, numeric and logical relations to each other. Hierarchies of tokens are important, as when surfaces are defined by ordered sequences of lines, and lines are defined by pairs of endpoints. It is therefore important to be able to access all the points in a given model, for example (a form of indirect named access).

Just as frequently, tokens are organized less rigidly into sets, such as the set of lines extracted from a single image. In such cases, it is important for a visual module to be able to access subsets of these tokens both spatially (for example all the lines in the upper corner of an image) or associatively (all the long lines in an image). The types of access can be easily combined in the ISR, for example to access the long lines in the upper corner of an image. In other cases, a visual procedure may need to iterate over all the tokens in a set, for example to find the average length of the long lines in the upper corner of the image. For these reasons, ISR includes operations for iterating over the tokens in a set, as well as taking the union, intersection and differences of sets[1].

4.2 Learning

Once visual interpretation has been cast as a process of controlled application of visual operators to hierarchically abstracted tokens, it is natural to begin to think of the process in terms of sequences of operators that start from simple tokens (possibly raw sensor signals) and, working through a set of intermediate representations, produce highly abstract tokens, such as landmark positions. More importantly, this structure allows the conceptualization of the space of operator sequences, allowing information fusion and visual control to be cast as a learning problem. In particular, given an abstract goal, we can search through the space of operator sequences for strategies (sequences of visual procedures and internal control information) that reliably achieve the goal on training images.

The Schema Learning System, developed by Draper [11], is a system that learns to satisfy abstract recognition goals by building strategies that apply visual procedures to intermediate representations. A user (or"teacher") provides SLS with a library of visual procedures and a set of training images. By applying the visual procedures to the training images, SLS learns what combinations of operators will produce the desired representation, and how reliable each operator is. The net result is a system which selects the most appropriate intermediate representations to reason about (relative to the goal)and builds an interpretation (control) strategy that determines how the information at the various levels is fused to produce new representations that are closer to achieving the goal.

4.2.1 Recognition Graphs

Interpretation strategies are represented in SLS as generalized multi-level decision trees called *recognition graphs* that direct both hypothesis formation and hypothesis verification, as shown in Figure 5. The premise

[1] The complement of a set of tokens is not well defined, since there is an infinite universe of possible tokens.

behind the formalism is that object recognition is a series of small hypothesis verification tasks interleaved with representational transformations. Recognition begins with trying to verify hypotheses at a low level of representation, separating to the extent possible hypotheses that are reliable from those that are not. Verified hypotheses (or at least, hypotheses that have not been rejected) are then transformed to a higher level of representation, where a new verification process takes place. The cycle of verification followed by transformation continues until goal hypotheses (in the figure example, a 3D pose hypothesis) are verified, or until all hypotheses have been rejected.

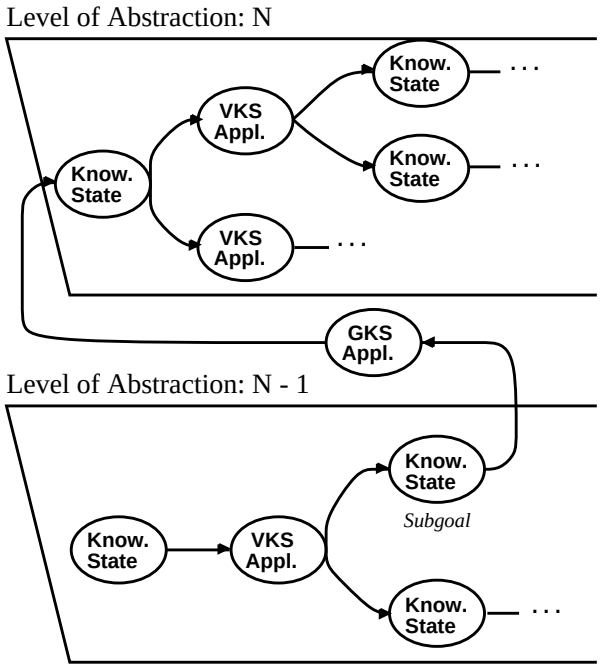

Figure 5: A recognition graph. Levels of the graph are decision trees that verify hypotheses using feature measurement procedures (FMPs). Hypotheses that reach a subgoal are transformed to the next level of representation by a transformation procedure (TP).

The structure of the recognition graph reflects the verification/transformation cycle. Each level of the recognition graph is a decision tree that controls verification at one level of representation by measuring features of intermediate-level ISR tokens. When a token is determined to be reliable by the decision tree, a transformation procedure (TP) transforms it to another level of representation, where the process repeats itself.

As defined in the field of operations research, decision trees are a form of state-space representation composed of alternating *choice states* and *chance states*. When searching for a path from the start state to a goal state, an agent is only allowed to choose where to go next from a choice state. If the current state is a chance state the next state is selected probabilistically[2]. The search process is therefore similar to using a game tree against a probabilistic opponent.

In SLS, the choice states are hypothesis knowledge states as represented by sets of hypothesis feature values. The choice to be made at each knowledge state is which feature (if any) to measure next. Chance states in the tree represent feature measurement states, where the chance concerns which value will be

[2]Operations research terminology is based on trees rather than spaces, so it refers to choice nodes and chance nodes rather than choice states and chance states, and to leaf nodes and root nodes rather than goal states and start states.

returned. Token (i.e. intermediate representation) verification is an alternating cycle in which the control strategy selects which feature to measure next and the measurement routine probabilistically returns a feature value. Thus tokens advance from knowledge states to feature measurement states and then on to new knowledge states. The cycle continues for each token until it reaches a subgoal state, indicating that it has been verified and should be transformed to a higher level of representation, or a failure state, indicating that the token is an unreliable hypothesis and should be rejected.

In general, SLS learns in advance which features to measure at each knowledge state in order to avoid making run-time control decisions. As a result, when SLS builds a recognition graph it leaves just one option at each choice node. Often, however, the readiness of a feature to be measured cannot be determined until run-time, in which case SLS will leave several options at a choice node, sorted in order of desirability[3]. At run-time the system will choose the highest-ranking feature that is ready to be executed.

4.3 Inference Algorithms

The inference algorithm by which SLS infers recognition graphs from training images will not be discussed in detail; interested readers are referred to [11, 12]. However, we will briefly mention some of its most important characteristics. First, SLS's inference algorithm is a syntactic, logic-based algorithm that makes no assumptions about the visual procedures or representations it is manipulating other than the declarative knowledge about the type of hypotheses that each takes as input and produces as output. As a result, although Table 4.3 shows some of the visual procedures and representations that have been used in SLS experiments to date, this list is by no means exclusive: SLS could just as easily manipulate generalized cylinders as wire-frame models, and most likely any algorithm found in the image understanding literature could be included in an SLS strategy.

Second, SLS's learning algorithm tries to minimize the expected cost of recognition, subject to the constraint that a strategy must recognize every object instance in the training set. As a prerequisite for the minimization process, SLS estimates the expected cost of every visual procedure and the likelihood of each feature, based on information gained from the training images.

Finally, SLS's inference algorithm is strictly a generalization algorithm. SLS starts by learning a strategy for finding the target object in the first training image; it then generalizes this strategy to account for the second training image, and the third, and so on, until eventually the strategy is general enough to find the object in every training image. As it generalizes, each new strategy is guaranteed to be less likely to fail on a new image than the strategy it was generalized from, and this implies that SLS's strategies can be analyzed using techniques introduced by Valiant [27].

5 Conclusion

Using a special case as an example, we have argued for a general framework for information fusion in computer vision. The framework revolves around a database of intermediate representations, including, but not limited to points, lines, regions and surfaces, and a library of visual procedures that operate on those representations. As a special case, we examined one strategy for landmark recognition in which the estimated pose of the vehicle is used to select a landmark on the map, the color of that landmark is used to focus the systems attention on a specific regions of the image and a geometric model matching algorithm is used to determine the refine the estimates of vehicle pose relative to the landmark. As such, this strategy integrates three representations: 1) coordinate transforms for vehicle pose, 2) regions of interest for the

[3] This is just one of many complications that arise from multiple-argument visual procedures. In general, we will describe SLS as if all VPs took just one argument in order to keep the description brief; see Draper [11] for a more complete description.

Transformation Procedure	Input	Output	Ref
Moravec Operator	Image or ROI	2D Points	[Moravec 1981]
Line Extraction	Image or ROI	2D Lines	[Boldt & Weiss 1989]
Region Segmentation	Image or ROI	Region Set	[Beveridge, et. al. 1989]
Color Classification	Region Set	Region (sub)Set	[Duda & Hart 1973]
Polygonal Approx.	Region	2D Lines	
Parabola Fitting	Region	Parabola	
Pencil Grouping	2D Lines	Pencil	[Collins & Weiss 1990]
Vanishing Point Anal.	Pencil	3D Orientation	[Collins & Weiss 1990]
Trihedral Grouping.	2D Lines	Trihedral Jcts	
Trihedral Angle Anal.	Trihedral Jcts	3D Orientation	[Kanatani 1988]
Reprojection(1)	3D Orient. & Region	3D Plane	
Reprojection(2)	3D Orient. & 2D Points	3D Points	
Absolute Orientation	3D Points	Pose	[Horn 1987]
3D Pose Determination	Line Corresp.	Pose	[Kumar 1992]
Subgraph Isomorphism	2D Lines	2D Lines	[Ullman 1976]
Point Resection	2D Points	Pose	[USGS]
Geometric Model Match	2D Lines	Pose	[Beveridge & Riseman 1992]

Table 1: The current library of transformation procedures (TPs) in SLS. SLS integrates procedures at a syntactic level, based solely on knowledge of the types of hypotheses a procedure takes as input and produces as output. As a result, new procedures can be easily added to its library.

focus of attention and 3) straight lines, for use by the model matcher. It also uses three visual procedures: 1) pose-based landmark selection, 2) color-based classification and 3) geometric model matching.

This landmark recognition strategy, however, is just a special case, used to find the positions of modeled man-made objects when a rough estimate of that position is already available. Other integration strategies will be needed for other tasks. The Schema Learning System, together with ISR, is a system for acquiring these integration strategies by the application of machine learning techniques.

References

[1] Beveridge, J. R., R. Weiss and E. Riseman "Optimization of 2-Dimensional Model Matching," *Proc. of DARPA Image Understanding Workshop*, Palo Alto, CA, 1989, pp. 815-830.

[2] Beveridge, J. R. and E. M. Riseman "Can Too Much Perspective Spoil the View? A Case Study in 2D Affine and 3D Perspective Model Matching," *Proc. of DARPA IUW*, San Diego, CA, 1992.

[3] Beveridge, J.R. and Riseman, E.M. "Hybrid Weak-Perspective and Full-Perspective Matching," *IEEE Conf. on Computer Vision and Pattern Recognition,* June 1992, pp. 432-438.

[4] Breiman, L., Freidman, J.H., Olshen, R.A. and Stone, C.J. *Classification and Regression Trees,* Wadsworth Inc., Belmont, CA., 1984.

[5] Brodley, C.E. and Utgoff, P.E. "Multivariate Decision Trees," *Machine Learning,* to appear.

[6] Boldt, M., R. Weiss and E. Riseman RToken-Based Extraction of Straight Lines,S *IEEE Trans. SMC*, Vol. 19, 1989, pp. 1581-1594.

[7] Brolio, J., Draper, B.A., Beveridge, J.R. and Hanson, A.R. "The ISR: an intermediate-level database for computer vision", *Computer*, 22(12):22–30 (1989).

[8] Collins, R. and R. Weiss "Vanishing Point Calculation as a Statistical Inference on the Unit Sphere," *Proc. of IEEE ICCV*, Osaka, Japan, 1990, pp. 400-403.

[9] B. Draper, C. Brodley and P. Utgoff. "Goal-Directed Classification using Linear Machine Decision Trees," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* to appear.

[10] B. Draper, A. Hanson and E. Riseman. "ISR3: A Token Database for Integration of Visual Modules," *Proc. of the ARPA Image Understanding Workshop,* Washington, D.C., April 1993, pp. 1155-1161.

[11] B. Draper. "Learning Object Recognition Strategies", Ph.D. thesis, University of Massachusetts, May 1993. Available from the Dept. of Computer Science as Tech. Report 93-50.

[12] B. Draper, A. Hanson, and E. Riseman. "Learning Blackboard-based Scheduling Algorithms for Computer Vision," *International Journal of Pattern Recognition and Artificial Intelligence,* 7(2), to appear.

[13] Duda, R.O. and Hart, P.E.. *Pattern Classification and Scene Analysis.* John Wiley & Sons: New York, 1973.

[14] Frean, M. *Small nets and short paths: Optimising Neural Computation.* Ph.D. thesis, Center for Cognitive Science, University of Edinburgh. 1990.

[15] Gallant, S.I., "Connectionist Expert Systems", *Communications of the ACM, 31,* 152-169, 1986.

[16] Grimson, W.E.L. *Object Recognition by Computer.* Cambridge, MA.: MIT Press, 1990.

[17] Horn, B. K. P. "Closed-form solution of absolute orientation using unit quarternions," *J. Opt. Soc. A., Vol. 4*, 1987, pp. 629-642.

[18] Kanatani, K., "Constraints on Length and Angle", *Computer Vision, Graphics, and Image Processing*, 41:28-42, 1988.

[19] Kumar, R. and Hanson, A.R. "Determination of Camera Position and Orientation," *Proc. of the DARPA Image Understanding Workshop,* Palo Alto, CA., May 1989, pp. 870–881.

[20] Kumar, R. Model Dependent Inference of 3D Information from a Sequence of 2D Images, Ph. D. Thesis and Technical Report TR92-04, Department of Computer Science, University of Massachusetts (Amherst), 1992.

[21] Moravec, H., "Robot Rover Visual Navigation", *UMI Press*, 1981.

[22] Nilsson, N.J. *Learning Machines.* McGraw-Hill, New York, 1965.

[23] Pomerleau, D.A., Gowdy, J. and Thorpe, C.E. "Combining Artificial Neural Networks and Symbolic Processing for Autonomous Robot Guidance," *Proc. of the DARPA Image Understanding Workshop,* San Diego, CA, Jan. 1992. pp. 961–967.

[24] Thorpe, C., Hebert, M., Kanade, T., and Shafer, S., "Toward Autonomous Driving: The CMU Navlab", *IEEE Expert*, V.6, No. 4, August 1991.

[25] Utgoff, P.E., and Brodley, C.E., "An Incremental Method for Finding Multivariate Splits for Decision Trees", *Proc. of the Seventh International Conference on Machine Learning*, Austin, TX, 1990, Morgan-Kaufman.

[26] Ullman, J.R., "An Algorithm for Subgraph Isomorphism", *Journal of the ACM*, 23(1):31-42, 1976.

[27] Valiant, L.G. "A Theory of the Learnable", *Communications of the ACM*, 27(11):1134-1142, 1984.

[28] Weems, C., S. Levitan, A. Hanson, E. Riseman, J. Nash and D. Shu RThe Image Understanding Architecture,S *IJCV*, Vol. 2(3), 1989, pp. 251-282.

[29] Young, P., *Recursive Estimation and Time-Series Analysis*, New York: Springer-Verlag.