Communications

A Flexible Tool for Prototyping ALV Road Following Algorithms

SVEN J. DICKINSON, STUDENT MEMBER, IEEE, AND
LARRY S. DAVIS, MEMBER, IEEE

Abstract—**A production system model of problem solving is applied to the design of a vision system by which an autonomous land vehicle (ALV) navigates roads. The ALV vision task consists of hypothesizing objects in a scene model and verifying these hypotheses using the vehicle's sensors. Object hypothesis generation is based on the local navigation task, an *a priori* road map, and the contents of the scene model. Verification of an object hypothesis involves directing the sensors toward the expected location of the object, collecting evidence in support of the object, and reasoning about the evidence. Constructing the scene model consists of building a semantic network of object frames exhibiting component, spatial, and inheritance relationships. The control structure is provided by a set of communicating production systems implementing a structured blackboard; each production system contains rules for defining the attributes of a particular class of object frame. The combination of production system and object-oriented programming techniques results in a flexible control structure able to accommodate new object classes, reasoning strategies, vehicle sensors, and image analysis techniques.**

I. Introduction

The development of an autonomous land vehicle (ALV) involves the development of computer vision techniques by which a vehicle can autonomously navigate itself through the environment. Although the goals for the ALV are broad, including both on- and off-road navigation, the work presented here is primarily concerned with the road-following task. Fig. 1 presents a view as seen from a camera mounted on top of the ALV. From images such as these, the ALV vision system constructs a model of the environment; this scene model contains the objects visually identified by the ALV. Based on this collection of objects, the vehicle plans a course and moves through the environment.

For the road-following task, the scene model contains either objects that represent the road or objects from which the location of the road can be deduced. Obviously, the direct detection of a patch of road would be most useful; however, in the event that the ALV vision system cannot directly identify the road, the detection of other objects may suggest the location of the road. For example, telephone poles and ditches often run parallel to the road; their presence may thus provide clues as to its location and direction. In certain cases, major landmarks contained in a road map such as buildings may be used to infer the road location; however, such information is more useful in registering the vehicle to some absolute location.

Faced with the task of building a scene model, the ALV vision system must decide which of the above objects should be sought in order to locate the road. Hence, the first step in constructing the scene model is the joint decision of what object to search for in the world and where to search for it. The second step performed by the ALV vision system is to gather evidence confirming the existence of the specified object at the hypothesized location. The third and final step involves reasoning about the evidence. If the confirming evidence is sufficient, the object is inserted into the scene model; otherwise, the object may be hypothesized elsewhere or a different object may be hypothesized.

Manuscript received January 25, 1988; revised October 15, 1988.
The authors are with the Center for Automation Research, University of Maryland, College Park, MD 20742.
IEEE Log Number 9035503.

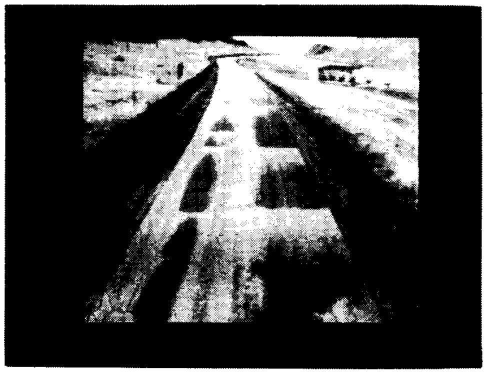
Fig. 1. Typical ALV road image.

Construction of the scene model is complex. The selection of which object to track depends on the navigation goals of the ALV, the history of object tracking, the contents of the scene model, and information from the road map. Verifying the existence of an object requires directing vehicle sensors towards the object, fusing data from different sensors, and selecting algorithms for image analysis. Methods for performing all these tasks are continually evolving as the road-following task becomes better understood. New objects must be tracked by the ALV, new sensors are made available to track objects, and new image processing techniques are identified for sensor image feature extraction. The successful evolution of an ALV vision system hinges on the ability of its control structure and knowledge representation schemes to accommodate these changes.

ALV road following under arbitrary conditions has not yet been accomplished; however, under certain environmental restrictions, success can be achieved. Previous work [14] describes the design of an ALV system that met very specific objectives. Under constrained conditions, the system proved to be both successful and efficient. As the goals of the ALV program evolved, we required a flexible tool to test new road-following strategies, including new sensors and object classes. However, the specific control structure inherent in the original system failed to accommodate such prototyping. As a result, we propose the design of a system for constructing an ALV scene model offering a unique implementation based on a set of communicating production systems. A production system model of computation is particularly effective when the system is expected to undergo many changes by many software engineers. Although intended to support the tracking of many objects through a variety of sensors, the immediate goal has been to emulate the feedforward capability of the system discussed in [14]. Hence, we restrict our domain of objects to patches of road, and we restrict our sensors to a single video camera.

In the following section, we provide an overview of the system before exploring in detail the representation and control schemes.

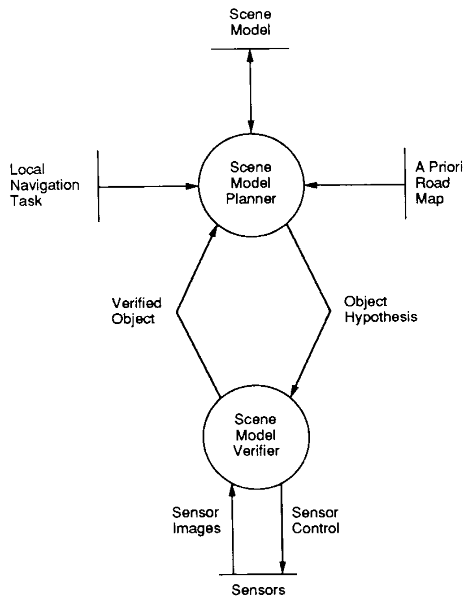

Fig. 2. Scene model builder dataflow.

In Section III, we discuss object modeling and describe the object classes currently available in the system, whereas in Section IV, we discuss the scene model network. Sections V and VI discuss the control strategies involved in the selection and verification of objects, respectively. We conclude with a series of results demonstrating the system's capabilities and discuss the evolution of the system.

II. System Overview

The task of building a scene model for the ALV consists of two major subtasks: 1) deciding what object to look for and where to look for it and 2) verifying that the object exists in the world.

These two functions are performed by the scene model planner (planner) and scene model verifier (verifier), respectively; together, they form the scene model builder (builder). The data flow diagram for the builder is presented in Fig. 2. The planner, in addition to interpreting and updating the scene model, is aware of the local navigation task and initiates queries to the *a priori* road map. The verifier controls the movement of the sensors and acquires the sensor image data. In a hypothesize-and-test paradigm, the planner sends object hypotheses to the verifier, and the verifier returns verified objects to the planner.

The local navigation task is a function of the global navigation task and the location of the vehicle; for example, a local navigation task might be to follow the road for 100m or turn right at the first intersection after a certain landmark is identified. The road map contains *a priori* information about the ALV environment, including the approximate locations, sizes, and compositions of roads, intersections, and landmarks. Neither the navigator (module defining the local navigation task) nor the road map have been implemented; they are not the focus of this research. However, for demonstration purposes, a simplistic version of both exists to answer queries from both the planner and verifier.

The dataflow of the builder proceeds as follows. The planner first determines the scene model requirements of the local navigation task; for example, following a straight road requires that the left and right road boundaries be contained in the scene model. Next, the planner looks at the road map and the partial scene model and decides what objects may be useful in locating the road; for example, it might decide that a road patch, a ditch, or even a row of telephone poles is sufficient to define a road boundary. The planner then decides the

planar-ribbon
attributes:
search-location
search-strategy
back-connected-planar-ribbon
front-connected-planar-ribbon
has-part-left-world-segment
has-part-right-world-segment
expected-width
actual-width
actual-width-confidence
parallelism-confidence
total-confidence

road-patch
attributes:
prior-road-straightness
left-world-segment-type
right-world-segment-type
inherited-frames:
planar-ribbon

Fig. 3. Road patch/planar ribbon frames.

type and expected location of the object to be tracked, hypothesizes the object, and passes the hypothesis to the verifier.

The verifier attempts to verify the hypothesis by directing the vehicle's image sensors towards the expected location of the object. The object is then located in the sensor images, and its image location is mapped to a three-dimensional (3D) location based on a fixed point of reference. The confidence with which the object is found becomes a measure of its verifiability. Once the confidence is defined, the hypothesis is returned to the planner for inspection. If the object is deemed sufficiently verified, it is added to the scene model. Otherwise, the planner determines the next course of action; for example, the object may be hypothesized in a different location, or a new object may be hypothesized.

III. Modeling World Objects

Objects in the ALV scene model exhibit the following relationships:

- **Component Relationships:** For example, an intersection is made up of four connecting roads, stop lights, etc. These component objects, in turn, can be decomposed into their component objects, e.g., a road may be defined as a pair of left and right segments, where each segment represents the border between road and shoulder or shoulder and background. Primitive objects such as lines and surfaces extracted from an image cannot be decomposed.
- **Spatial Relationships:** For example, telephone poles are often located near the road and run parallel to the road.
- **Property Inheritance:** For example, a 3D line segment may be defined by a pair of endpoints. The edge separating the road surface from the shoulder is a specialization of a 3D line segment; thus, in addition to its properties specific to a road edge, it inherits the endpoint properties of the 3D line segment.

To accommodate these relationships, frames have been chosen to model objects [9]. A frame is a data structure containing a set of slots (or attributes) that encapsulate the relevant knowledge pertaining to an object. Slots may contain values, e.g., the width of a road patch, or pointers to related frames, e.g., component and spatially related frames. Property inheritance among frames is accomplished by supplementing a frame's slots with the inherited frame's slots. The following sections describe the object frames defined in the system.

A. The Road Patch

A planar ribbon is defined as a pair of facing and parallel 3D line segments. A road patch is a specialization of a planar ribbon whose 3D line segments represent the left and right features of the road. Thus, a road patch frame inherits the attributes of a planar ribbon. The road patch and planar ribbon frames are depicted in Fig. 3. Road patches are oriented and may be connected together to form a piece of road; the front of one road patch may be connected to

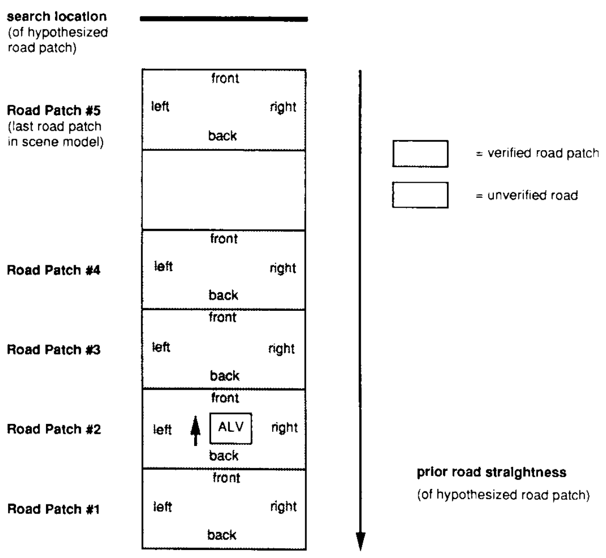

Fig. 4. Series of road patches.

the back of another. The orientation of a road patch is based on the assigned orientation of the initial road patch; typically, the back end of the initial road patch is closest to the ALV, whereas the front end is furthest from the ALV. The left and right features, i.e., 3D segments, are oriented looking from the back to the front of the road patch. Fig. 4 depicts the vehicle with respect to a series of road patches. The following attributes comprise the road patch frame:

- **search location:** The search location specifies the expected location of the road patch in terms of a 3D line segment. The orientation of this segment, which is called a rib, is such that the left and right boundaries of the road are expected to pass through the left and right endpoints, respectively; hence, the direction of the line segment is perpendicular to the expected direction of the road in that vicinity. Fig. 4 shows the search location for the next road patch.
- **search strategy:** The search strategy specifies the manner in which the road patch is to be verified. It is assumed that the vehicle is initially located on an *a priori* verified road patch. Road patch 1 in Fig. 4 is the initial road patch. From that point on, each road patch is verified according to one of the following strategies:
  1) *connected:* The connected search strategy is used to verify a road patch that is connected in the front and/or the back to another road patch. Road patches 2, 3, and 4 in Fig. 4 are examples of connected road patches.
  2) *disconnected:* The disconnected search strategy is used to verify a road patch that is connected in neither the front nor the back to another road patch. Road patch 5 in Fig. 4 is a disconnected road patch.
- **front (back) connected planar ribbons:** If the road patch is front (back) connected to the back (front) of another road patch, this attribute points to the connected road patch.
- **has part left (right) world segment:** This attribute points to the frame representing the left (right) road patch segment, which is a boundary of the road.
- **expected width:** The expected width is based on the actual width of the road patch in closest proximity.
- **actual width:** The actual width is defined as the average perpendicular distance from the endpoints of the left road patch segment to the ray defined by the right road patch segment. This is illustrated in Fig. 5.
- **actual width confidence:** The actual width confidence is defined as follows:

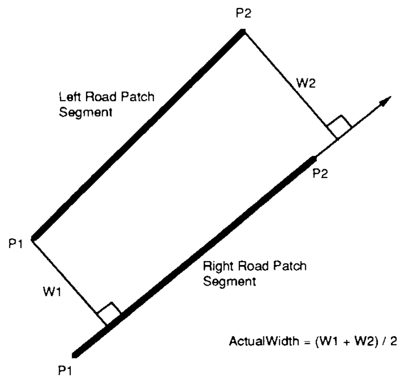

Fig. 5. Calculation of actual width confidence.

$$\max\left(0.0,\ 1.0 - \left|\frac{\text{ActualWidth} - \text{ExpectedWidth}}{\text{ExpectedWidth}}\right|\right).$$

Intuitively, the fractional term represents the deviation of the actual width from the expected width.

- **parallelism confidence:** The parallelism confidence is defined as follows:

$$\frac{90 - \theta}{90}$$

where θ is the acute angle between the left and right road patch segments. The maximum confidence is achieved when the two segments are parallel, whereas minimum confidence is achieved when the segments are orthogonal.

- **total confidence:** The total confidence is a function of the width confidence, the parallelism confidence, and the total confidence of the left and right component features. The function used in our experiments was

$$\begin{gathered}0.40\ \text{ParallelismConfidence} + 0.20\ \text{WidthConfidence}\\ +\,0.20\ \text{LeftRoadPatchSegmentTotalConfidence}\\ +\,0.20\ \text{RightRoadPatchSegmentTotalConfidence}.\end{gathered}$$

The total confidence function has been empirically determined and gives added weight to road patches whose left and right road patch segments are parallel.

- **prior road straightness:** The prior road straightness specifies the length (in meters) of straight road from the front of the last verified road patch extending back into the scene model. The prior road straightness of the hypothesized patch in Fig. 4 is given by the arrow.
- **left (right) world segment type:** The left (right) world segment type specifies whether the component left (right) road patch segment defines the edge between the road surface and the road shoulder surface (road edge) or the edge between the road shoulder and the vegetation or background (shoulder edge).

B. The Road Patch Segment

A world segment is defined as a 3D line segment. A road patch segment is a specialization of a world segment representing a road feature, i.e., the boundary between the road surface and the shoulder surface, or the boundary between the shoulder surface and the vegetation or background. Thus, a road patch segment frame inherits the attributes of a world segment. The road patch segment and world segment frames are depicted in Fig. 6. Road patch segments are oriented and may be connected together to form a continuous linear feature; the front of one road patch segment may be connected to the back of another. The orientation of a road patch segment is based

world-segment
attributes:
search-location
search-strategy
endpoint-A
endpoint-B
endpoint-A-connected-world-segment
endpoint-B-connected-world-segment
part-of-planar-ribbon
has-part-camera-segment
total-confidence

road-patch-segment
attributes:
world-segment-feature
endpoint-A-orientation
endpoint-B-orientation
continuity-confidence
inherited-frames:
world-segment

Fig. 6. Road patch segment/world segment frames.

camera-segment
attributes:
search-window
search-strategy
endpoint-A
endpoint-B
endpoint-A-connected-camera-segment
endpoint-B-connected-camera-segment
part-of-world-segment
camera-image
method-of-extraction
total-confidence

road-patch-camera-segment
attributes:
camera-segment-feature
endpoint-A-orientation
endpoint-B-orientation
inherited-frames:
camera-segment

Fig. 7. Road patch camera segment/camera segment frames.

on the orientation of its parent road patch. The following attributes comprise the road patch segment frame:

- **search location:** The search location specifies the expected location of the road patch segment in terms of a 3D point through which the road patch segment is expected to pass. This point is one of the endpoints belonging to the search location of its parent road patch.
- **search strategy:** The search strategy specifies the manner in which the road patch segment is to be verified. Each road patch segment is verified according to one of the following strategies:

  1) *connected:* The connected search strategy is used to verify a road patch segment that is connected in the front and/or the back to another road patch segment.
  2) *disconnected:* The disconnected search strategy is used to verify a road patch segment that is connected in neither the front nor the back to another road patch segment.

- **endpoint A (B):** Endpoint A specifies the 3D coordinates of endpoint A (B). It is calculated by applying a flat earth inverse perspective transformation [4] to endpoint A (B) of its component road patch camera segment frame.
- **endpoint A (B) connected world segment:** If the road patch segment is connected at endpoint A (B) to another road patch segment, this attribute points to the connected road patch segment.
- **part of planar ribbon:** This attribute points to the parent road patch frame.
- **has part camera segment:** This attribute points to the frame representing the component road patch camera segment.
- **total confidence:** The total confidence is a function of the continuity confidence (defined below) and the total confidence of its component road patch camera segment. The total confidence is calculated as follows:

$$0.30\ \text{ContinuityConfidence} + 0.70\ \text{RoadPatchSegmentTotalConfidence}$$

- **world segment feature:** The world segment feature specifies whether the road patch segment defines the edge between the road surface and the road shoulder surface (road edge) or the edge between the road shoulder and the vegetation or background (shoulder edge).
- **endpoint A (B) orientation:** This attribute defines the orientation of endpoint A (B) as front or back.
- **continuity confidence:** The continuity confidence, which is defined only when the road patch segment is connected, measures the degree to which the road patch segment and its connected neighbor lie on the same vector. The continuity confidence is calculated as follows:

$$1 - \frac{180 - \theta}{180}$$

where θ is the angle between the two road patch segments.

C. The Road Patch Camera Segment

A camera segment is defined as a two-dimensional (2D) line segment extracted from a camera image. A road patch camera segment is a specialization of a camera segment representing the 2D projection of a 3D road feature. Thus, a road patch camera segment frame inherits the attributes of a camera segment frame. The road patch camera segment and camera segment frames are depicted in Fig. 7. Road patch camera segments are oriented and may be connected together to form a continuous 2D linear feature; the front of one road patch camera segment may be connected to the back of another. The orientation of a road patch camera segment is based on the orientation of its parent road patch segment. The following attributes comprise the road patch camera segment frame:

- **search window:** The search window specifies the expected location of the road patch camera segment in terms of a two-dimensional rectangular window defined over a camera image. If the road patch camera segment is to be verified using the connected search strategy, the window is located adjacent to that from which the connected road patch camera segment was extracted. However, if the road patch camera segment is to be verified using the disconnected search strategy, the window is centered around the point defined by applying a flat earth direct perspective transformation [4] to the search location of its parent road patch segment.
- **search strategy:** The search strategy specifies the manner in which the road patch camera segment is to be verified. Each road patch camera segment is verified according to one of the following strategies:

  1) *connected:* The connected search strategy is used to verify a road patch camera segment that is connected in the front and/or back to another road patch segment.
  2) *disconnected:* The disconnected search strategy is used to verify a road patch camera segment that is connected in neither the front nor the back to another road patch camera segment.

- **endpoint A (B):** The two endpoints are the result of applying a linear feature extractor to the search window. The endpoints are constrained to lie on the border of the search window and define a 2D line segment.
- **endpoint A (B) connected camera segment:** If the road patch camera segment is connected at endpoint A (B) to another road patch camera segment, this attribute then points to the connected road patch camera segment.
- **part of world segment:** This attribute points to the parent road patch segment frame.
- **camera image:** This attribute points to the camera image on which the search window is defined.
- **method of extraction:** Currently, the linear features (and their confidence) are entered by hand; however, the authors plan to integrate the image processing techniques outlined in [8]. In that ap-

proach, the method of linear feature extraction depends on the search strategy. If the search strategy is connected, a method based on a one-dimensional Hough transform is applied to the search window using the connected endpoint as a pivot. If the search strategy is disconnected, a method based on a 2D Hough transform is applied to the search window. In both cases, the confidence would be a function of the size of the Hough peak, which in turn is weighted by the gradient magnitude.

- **total confidence:** The total confidence ranges from 0 to 1 and is based on the strength of the extracted linear feature.
- **camera segment feature:** The camera segment feature specifies whether the road patch camera segment defines the edge between the road surface and the road shoulder surface (road edge) or the edge between the road shoulder and the vegetation or background (shoulder edge).
- **endpoint A (B) orientation:** This attribute defines the orientation of endpoint A (B) as front or back.

The addition of attributes describing photometric properties of the image might improve the extraction of linear features from the image. For example, an *expected contrast* attribute, which measures the change in brightness across the road boundary, could assist in selecting from among the possible methods of extraction or determine the parameters of a particular method.

IV. The Scene Model

The scene model is central to the ALV vision system. It is accessed by the planner in determining what object to search for and by the ALV navigator when plotting a course through the scene model objects. As the domain of scene model objects grows larger, so does the variety of requests to the scene model. It is important that the semantics of the access commands be stable while the structure of the scene model evolves to accommodate new object classes. Hence, the structure of the scene model is transparent to those modules that access it. As in the case of the scene model objects, the scene model is a frame defining a set of query functions and hiding all implementation details. For example, queries can be made to determine the length of straight road at the end of the scene model or what road boundary, i.e., road or shoulder edge, has been verified most successfully.

Given the present limited domain of scene model objects, i.e. road patches and their components, a connected graph of road patches serves as an adequate scene model. However, as more object classes are defined, such an implementation may become insufficient; a structure accommodating both spatial and symbolic data would be preferred. Lawton *et al.* [7] propose a spatial grid in which each grid element (or pixel) contains a vector of pointers to objects that occupy that position; such a scene model supports both structural and geometric queries. Because the domain of objects does not yet include intersecting roads, the current implementation of the scene model is quite simple. The road patches are kept in a list ordered by their occurrence along the road; successive road patches in the scene model are not necessarily connected. When inserted into the scene model, each road patch is "trimmed" so that the front and back endpoints of its component road patch segments are square.

The frames comprising the road patches can be divided into two layers. The upper layer, including the road patch frame and its two road patch segment component frames, contains objects and their components as they are defined in the world. The lower layer, including the road patch camera segment frames, contains objects as they are defined in the sensor images. This division facilitates the incorporation of new sensors to the vehicle. For example, if we add a range scanner to the ALV, a road patch segment frame would then be given an additional slot defining a component road patch range segment. Aside from the additional attribute, the only impact on the road patch segment frame would be an alteration of the *endpoint A (B)* calculation to include the data from the road patch range segment and an alteration of the total confidence function to include the road patch range segment total confidence.

V. The Scene Model Planner

The function of the ALV navigator is to examine the scene model and plot a course for the vehicle with respect to the objects. If the local navigation task is to navigate the vehicle down the center of the road, in order for the navigator to be successful, the scene model must contain a sufficient number of relevant objects. For example, a scene model containing a series of connected road patches would be sufficient; the navigator need only plot a smooth curve between the left and right components of each road patch. However, a scene model containing houses would not provide sufficient information for the navigator to plot a path down the road; the navigator would be unsure of the proximity of the houses to the road. It is up to the planner to decide, based on the local navigation task, what objects should appear in the scene model. However, the planner's decision to track a particular object depends on more than the relevance of the object to the local navigation task. Choosing an object for verification should also depend on the history of tracking that object. For example, if the planner repeatedly fails to verify a particular class of object, it should hesitate to attempt further verification; however, if verification fails due to the object being far from the vehicle, this should not prevent the planner from attempting to verify the object as the vehicle moves closer to the object.

The planner is implemented as a frame whose slots point to the modules with which the planner communicates, i.e., the scene model, the *a priori* road map, the local navigation task, and the verifier. The unique aspect of this frame is that it inherits the capabilities of a production system and provides a rule database, a factual database, and a conflict resolution strategy. Based on the local navigation task, the *a priori* map, and the scene model, the planner decides what type of object to track and verify. To simplify the initial implementation of the planner, we have assumed a constant local navigation task of following the road and an *a priori* road map, which contains the approximate locations of intersecting roads along with an approximate road width, ad infinitum. Hence, the production system consistently selects a road patch for verification by instantiating a road patch frame whose attributes are undefined.

The next task of the production system is to choose the search location of the road patch hypothesis. The production system first queries the scene model for the directional history of the road. If the direction has varied erratically, the planner's confidence as to the location of the road patch is low. Imposing the constraint that the hypothesized road patch must be connected to the last road patch in the scene model improves the likelihood of verifying the road patch hypothesis. Hence, the production system defines the search strategy as "connected"; the search location is defined to be the leading edge of the connected road patch.

The architecture of the planner has been designed to facilitate the prototyping of new road-following strategies. The connected search strategy is just one possible strategy and is based on the feedforward algorithm described in [14]. However, in that system, the architecture was based on the feedforward algorithm; no alternate strategies could be easily accommodated. To demonstrate the flexibility of our system, we provide an alternate road-following strategy called the disconnected search strategy. If a section of road has been analyzed in a previous image, rather than systematically reverifying the road with connected search locations, the planner can hypothesize the search locations at larger intervals of, say, 10 m. A second application of the disconnected strategy is to hypothesize the search locations at larger intervals if the planner is confident that the road is straight based on prior tracking. To demonstrate the integration of the two road-following strategies, we combine the connected search strategy with the second application of the disconnected search strategy. The planner applies the connected strategy until it has accumulated at least 10 m of straight road in the scene model. Hypothesizing that the upcoming road is straight, it extrapolates the next search location a distance of 10 m from the end of the scene model.

Before the road patch hypothesis is transmitted to the verifier, the planner must decide which left and right features should define the road patch. This decision is based on the features defining the nearest

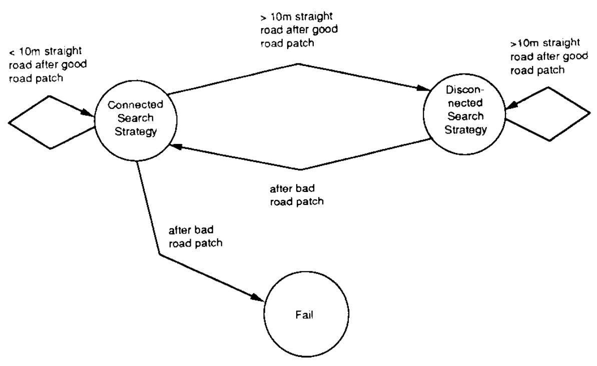
> 10m straight road after good road patch
< 10m straight road after good road patch
>10m straight road after good road patch
Connected Search Strategy
Disconnected Search Strategy
after bad road patch
after bad road patch
Fail

Fig. 8. Scene model planner states.

verified road patch and the confidence with which those features were verified. The expected road width is defined as the actual road width of the nearest verified road patch in the scene model.

The planner is now ready to send the road patch hypothesis to the verifier, where evidence is gathered in support of it. Once complete, the verifier returns the hypothesis to the planner; all the attributes in the road patch frame are now defined. If the evidence is deemed acceptable by the planner, it will add the verified object to the scene model. However, if the evidence is considered unacceptable, several options are available to the planner:

1) hypothesize the object at a different location;
2) hypothesize a different object;
3) retain the verified components of the unverified object.

Currently, only option 1) is implemented and proceeds as follows: If the unverified road patch hypothesis is disconnected, i.e., the planner ventured out beyond the end of the scene model to hypothesize the road patch, the hypothesis is abandoned, and a connected road patch is hypothesized back at the end of the scene model. If the unverified road patch hypothesis is connected, the planner aborts the road following task. Fig. 8 summarizes the actions taken by the planner.

When additional object classes become defined, the planner may take advantage of option 2); for example, the planner may choose to hypothesize a ditch beside the road if it was unsuccessful in verifying the road. When different search strategies become defined, the planner may take advantage of option 3). For example, although a road patch may not have been successfully verified, one of its component road patch segments may yield a high confidence; the planner may choose to insert this component into the scene model.

We chose to implement the planner as a production system to provide the flexibility required to accommodate the many changes it is expected to undergo. Granted, the control inherent in our current planner is quite simple, and a production system model of computation appears to offer little advantage. However, as more road-following strategies are introduced and more object classes defined, we feel that a production system will better accommodate the increased complexity and evolution.

The current implementation of the planner is strictly top-down, i.e., the planner decides what to look for and directs the verifier to find it; no accommodation has been made for unexpected objects in the field of view such as obstacles or holes in the road. Although this capability is essential for a successful road-following system, we have chosen to separate the top-down and bottom-up tasks. It is felt that an additional component is required to identify unexpected objects and deposit them into the scene model; this component is beyond the scope of this paper.

VI. The Scene Model Verifier

The role of the verifier is to receive an object hypothesis from the planner, collect evidence in support of the object, and return the verified object to the planner. More specifically, when the verifier receives an object hypothesis in the form of a sparsely defined frame, it proceeds to fill in the empty attributes; if the object has component parts, e.g., a road patch segment, the verifier must create and define these frames. To accomplish this task, a separate blackboard has been assigned to each class of object. In Section VI-A, we describe the structure of an object blackboard; the mechanism by which blackboards communicate to verify an object is presented in Section VI-B.

A. The Object Blackboard

When an object hypothesis is posted on the blackboard corresponding to its class, knowledge sources are activated to fill the empty attributes of the hypothesis. As in the case of the planner, each blackboard is implemented as a frame, providing a set of attributes and inheriting the capabilities of a production system. The attributes provide links to other blackboard frames and system modules, e.g., vehicle pilot and image processor. The production system rules control the activation of knowledge sources, i.e., when the left-hand side of a rule matches the contents of the factual database, the right-hand side activates a knowledge source. Ties are resolved by the conflict resolution strategy.

Blackboard frames, like object frames, possess both component and inheritance relationships; spatial relationships are undefined for blackboard frames. For example, the road patch blackboard has an attribute pointing to the road patch segment blackboard; although there are two component road patch segments for each road patch, there is only one road patch segment blackboard on which every instance of a road patch segment object is posted. When an object blackboard is instantiated, it may inherit the attributes and rules of other object blackboards. Consider, for example, the instantiation of a road patch blackboard; the blackboard now contains the attributes and rules of both the road patch and planar ribbon blackboard. Thus, for every attribute of a road patch object, there exists one or more rules invoking knowledge sources defining that attribute.

If we look carefully at the rule(s) corresponding to the *total confidence* attribute, we find that they originate in the planar ribbon blackboard since this attribute was inherited from the planar ribbon object. These rules control how the confidence is defined for a generic planar ribbon. For some objects that may inherit the planar ribbon attributes, a simple, generic confidence measure may be sufficient; however, in the context of a road patch, an alternate confidence measure may be more appropriate. Since there is no way for the planar ribbon to anticipate which objects may inherit its attributes, it is impossible to provide a set of rules in the planar ribbon blackboard that define the *total confidence* of a road patch object. Instead, the planar ribbon blackboard retains the generic rules, whereas the road patch provides its own rules. When the road patch blackboard inherits the planar ribbon blackboard, the redundant *total confidence* rules originating from the planar ribbon blackboard are automatically deactivated.

B. Top-Down Hypothesis Verification

When the planner hypothesizes an object, the verifier invokes a top-down approach to verify the object. In Fig. 9, this approach has been applied to the verification of a road patch hypothesis. Once the planner creates the road patch hypothesis, it posts it on the road patch blackboard (a specialization of a planar ribbon blackboard). The rules belonging to the road patch blackboard, acting as demons, invoke knowledge sources to define the attributes of the now sparsely defined road patch hypothesis. The rule antecedents ensure that the attributes are defined in a specific order. When rules fire to define the *has part left world segment* component, the activated knowledge source creates a road patch segment object hypothesis, defines a subset of its attributes, and posts it on the road patch segment blackboard. At this point, control is transferred to the road patch segment blackboard, whereas the road patch blackboard is put to sleep.

Responding to a new object hypothesis on their blackboard, the rules belonging to the road patch segment blackboard proceed to define the attributes of the road patch segment object. When rules fire

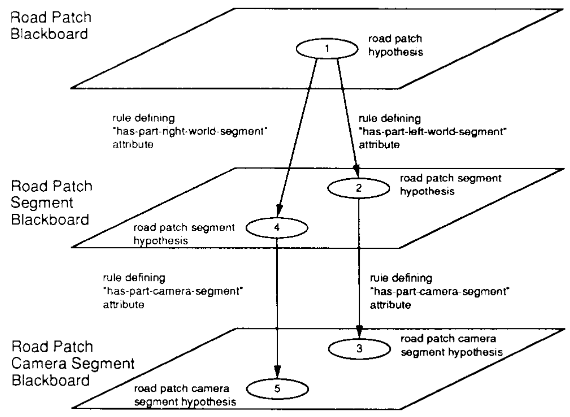
Road Patch Blackboard
road patch hypothesis
1
rule defining "has-part-right-world-segment" attribute
rule defining "has-part-left-world-segment" attribute
Road Patch Segment Blackboard
2
road patch segment hypothesis
road patch segment hypothesis
4
rule defining "has-part-camera-segment" attribute
rule defining "has-part-camera-segment" attribute
Road Patch Camera Segment Blackboard
3
road patch camera segment hypothesis
road patch camera segment hypothesis
5

Fig. 9. Top-down hypothesis verification.

to define the *has part camera segment* attribute, the activated knowledge source creates a road patch camera segment hypothesis, defines a subset of its attributes, and posts it on the road patch segment blackboard. Control is transferred to the road patch camera segment blackboard, and the road patch segment blackboard is put to sleep.

The rules at the road patch camera segment blackboard proceed to define where and how the road patch camera segment is to be searched for in the camera image. When the rules fire to define *endpoint A (B)*, a knowledge source applies the *method of extraction* to the *search window* contained in the *camera image*. The *method of extraction* returns the predominant linear feature (*endpoint A (B)*) in the search window along with a confidence measure (*total confidence*). A more powerful approach would return a set of segments that would lead to multiple competing road patch hypotheses. Although such a bottom-up strategy has been designed, it has not been implemented; as a result, only one road patch hypothesis is active at any time.

At this point, all the attributes are defined, and control is passed back up to tne road patch segment blackboard, where the remaining road patch segment hypothesis attributes are defined. Similarly, when its attributes are defined, control is passed up to the road patch blackboard. The next attribute, *has part right world segment*, repeats the entire process until eventually control is once again at the road patch blackboard. Once the last attribute, the road patch *total confidence*, is defined, the completed road patch hypothesis is returned to the planner for evaluation. Note that since the size of the image search windows is constant, the size of the road patch will vary according to the effects of perspective projection.

This system of communicating blackboards offers many advantages to the system builder. Each modular blackboard controls the definition of a single object class; as new classes are created, new blackboards are defined. If the definition of a class is changed, i.e., attributes are added or deleted, new sets of rules are added or deleted. Since rules map to single attributes, the alteration of one set of rules will have little or no impact on rules corresponding to other attributes. Within each blackboard, the inherent advantages of a rule-based system are clear. Rule-based activation of knowledge sources provides a data-driven, flexible control structure, whereas English-like rules provide readability and support maintainability.

VII. Experimental Results

In this section, we demonstrate the system on a sequence of images taken from the Martin Marietta ALV test track in Denver, CO. The system was not tested on the vehicle; rather, the simulation moves the vehicle through the discrete positions from which the images were acquired. In addition, the location and confidence of all road patch camera segments was input manually; no image processing was performed. The lack of image sequences with ground truth information prevented extensive testing of the system, particularly in cases where the road is obscured or ambiguous in the image.

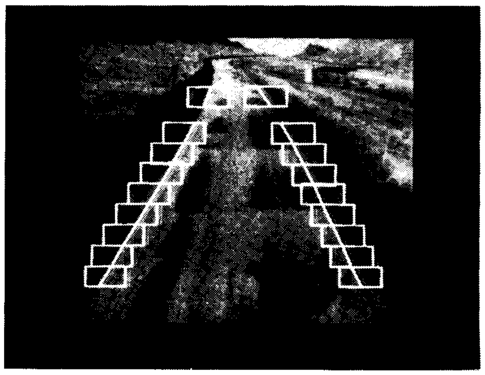

Fig. 10. Tracking a straight road—Frame 1.

The current implementation runs in the Maryland Franz Lisp environment [1], under UNIX[1] 4.3BSD on a VAX[2] 11/785. As described earlier, all system modules are frames implemented using the Maryland Franz Flavors package [15]; the production system frames inherited by the planner and verifier blackboards are implemented using YAPS [2]. YAPS is an antecedent-driven production system similar to OPS5 [5], but it offers more flexibility. All communications between modules (e.g., verifier, planner, scene model, road map, navigation task) and between blackboard layers is accomplished through the message passing facility included in the Maryland Franz Flavors package. Functions bound to the frames are implemented in Lisp; C routines are called from the Lisp environment for numerically intensive processing. All image display functions are provided by a Vicom image processor.

The first sequence is shown in Fig. 10 and demonstrates the construction of a scene model containing a straight road. It is assumed that the vehicle is initially seated on the road with known position and orientation. If the initial position and orientation were unknown, a bottom-up road patch hypothesis generation strategy could be used to find the road in the field of view; again, this strategy has been designed but not implemented. At the bottom of the image, the initial search windows are placed according to the expected location of the road in the image; the search windows are indicated by the rectangular boxes, which contain the extracted line segments. From then on, the road patch connected search strategy is repeatedly invoked to verify successive connected road patches. Following the insertion of the eighth road patch into the scene model, over 10 m of straight road have been accumulated. In this case, the disconnected search strategy is invoked, resulting in a search location 10 m beyond the end of the scene model. Because the road was correctly predicted to be straight, the road patch is successfully verified. With approximately 20 m of straight road in the scene model, the disconnected search strategy is again invoked; however, when the 3D search location of the third road patch is mapped to the current image, the search windows are out of bounds (off the top of the image). Subsequently, as depicted in Fig. 11, a new image is acquired. Since the vehicle is aware of the distance it has traveled since the last image was acquired, it can predict the expected projection of the road patch search location in the new image; subsequent processing continues as before.

In the second sequence, which is shown in Fig. 12, the ALV attempts to track a curved road. As in the previous sequence, the initial portion of the road is straight; the same steps are used to build the initial eight road patches, and the search strategy is changed from connected to disconnected. However, because the vehicle could not predict the upcoming curve in the road, the predicted search location

[1] UNIX is a trademark of Bell Laboratories.
[2] VAX is a trademark of Digital Equipment Corporation.

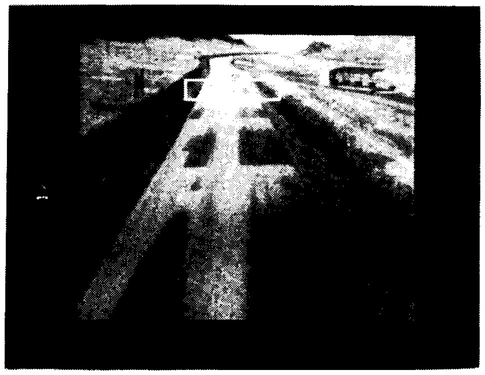
Fig. 11. Tracking a straight road—Frame 2.

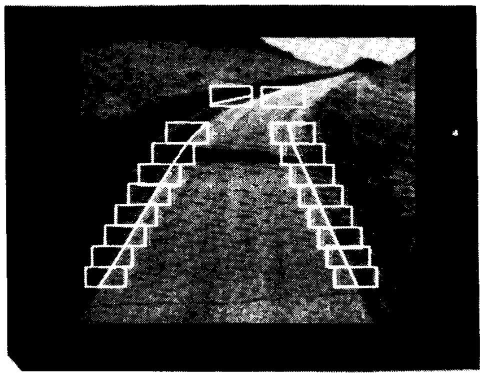
Fig. 12. Tracking a curved road—Frame 1.

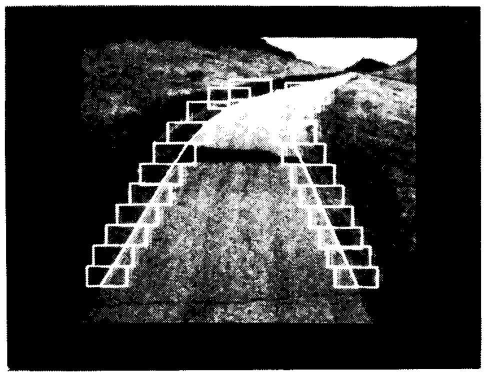
Fig. 13. Tracking a curved road—Frame 2.

is off the road, ultimately yielding a road patch with very poor total confidence (due to lack of parallelism and poor width). The planner aborts the disconnected search strategy and invokes the connected search strategy from the previously verified road patch in the scene model. As a result, the curve is successfully navigated, as is shown in Fig. 13.

VIII. System Evolution

In Section VII, we demonstrated two search strategies invoked by the planner to verify a road patch. Because these strategies are rather simplistic, we expect that they will evolve with time and have designed the control structure to accommodate such evolution. In this section, we demonstrate the flexibility of the control structure by exploring the effects of altering the search strategies used by the planner. For each proposed change, we discuss the necessary modifications to the code and present the results of running the modified system on sample images. The first two examples discuss trivial modifications and are intended to familiarize the reader with the notation; the third and final example presents a more interesting modification. Although, in this section, we discuss modifications to the planner, the control structure also facilitates modifications to the verifier, as is briefly discussed in Section VI-B.

A. Reducing the Static Road Projection

When the scene model has accumulated at least 10 m of straight road, the planner switches from a connected search strategy to a disconnected search strategy; the next road patch is hypothesized at a distance of 10 m from the last road patch in the scene model. In this section, we demonstrate the changes required to alter this distance from 10 to 5 m. Among the YAPS rules that collectively define the road patch *search strategy*, the following rule defines the *search strategy* as disconnected:

```
(defp define-disconnected-search-strategy
      (hypothesized object -object)
      (goal (define search strategy for road patch -object))
test (null (<- -object 'search-strategy))
     (cond ((not (null (<- -object 'prior-road-straightness)))
     (>= (<- -object 'prior-road-straightness)
     MIN-ROAD-STRAIGHTNESS)))
->
     (<- -object 'set-search-strategy 'disconnected)).
```

The antecedent of the rule (the four expressions preceding the "—>") is a conjunction of conditions that must be satisfied in order for the consequent (the expression following the "—>") to be executed. The first expression in the antecedent matches a road patch hypothesis created by the planner. The second expression represents the current goal of the planner; in this case, the planner is attempting to define the *search strategy* of the road patch hypothesis. The next two expressions, called test clauses, specify further conditions that must be met: The *search strategy* must be previously undefined, and the *prior road straightness* must exceed 1000 cm (the value of MIN-ROAD-STRAIGHTNESS). The symbol "<-" indicates message passing between objects; for example, in the first test condition, a message is passed to the road patch hypothesis, which is bound to the variable object, requesting the value of the *search strategy* attribute. If both these conditions are met, the *search strategy* is defined as disconnected. With our new strategy, we wish to invoke the disconnected search strategy after accumulating 5 m of straight road. Thus, the required change to the system is simply to redefine the global variable (MIN-ROAD-STRAIGHTNESS) to equal 500 cm.

The next step in modifying our strategy involves the following rule, which defines the *search location* of the road patch hypothesis:

```
(defp define-disconnected-search-location
      (hypothesized object -object)
      (goal (define search location for road patch -object))
test (null (<- -object 'search-location))
     (eq (<- -object 'search-strategy) 'disconnected)
->
     (<- -object 'set-search-location
     (list (<- (<- *yaps-db* 'scene-model)
     ':predict-extended-left-feature-seed
     MAX-EXTENDED-SEARCH-DISTANCE)
     (<- (<- *yaps-db* 'scene-model)
     ':predict-extended-right-feature-seed
     MAX-EXTENDED-SEARCH-DISTANCE)))).
```

In this rule, the consequent defines the *search location* as a value resulting from sending two queries to the scene model and requesting points extrapolated from the left and right road patch segments, respectively, of the last road patch in the scene model. To support

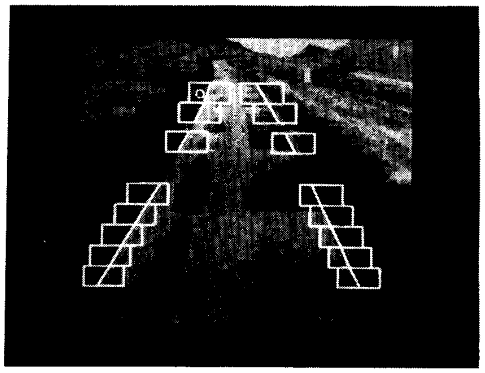
Fig. 14. Reducing the static road projection.

our new strategy, the global variable MAX-EXTENDED-SEARCH-DISTANCE must be redefined to equal 500 cm. Having altered the MIN-ROAD-STRAIGHTNESS and MAX-EXTENDED-SEARCH-DISTANCE global variables, we obtain the results depicted in Fig. 14.

B. Dynamic Road Projection

With our last modification, the planner repeatedly hypothesizes the next road patch 5 m out, provided that the last road patch was successfully verified. We now demonstrate how to make the planner a little braver by having it move out 10 m after the first 5-m extension. The reasoning, of course, is that as we accumulate more straight road, we expect more to lie ahead. At some point, however, the verifier has trouble verifying road patches that are too far ahead; we choose 10 m as our limit. To accommodate this dynamic projection search strategy, we add the following YAPS rule to the planner:

```
(defp define-dynamically-disconnected-search-strategy
     (hypothesized object -object)
     (goal (define search strategy for road patch -object))
test (null (<- -object 'search-strategy))
     (cond ((not (null (<- -object 'prior-road-straightness)))
     (>= (<- -object 'prior-road-straightness)
     MIN-ROAD-STRAIGHTNESS)))
     (eq (<- (<- (<- *yaps-db* 'scene-model)
     ':retrieve-most-recent-road-patch) 'search-strategy)
     'disconnected)
-->
     (<- -object 'set-search-strategy 'dynamically-disconnected)).
```

In the test clauses, we check that the *search strategy* of the road patch hypothesis is undefined and make sure that we have accumulated sufficient straight road in the scene model. In addition, we check that the previously verified road patch was verified using the disconnected search strategy. If all these conditions hold, the *search strategy* is defined to be dynamically disconnected.

To define the *search location* for this new strategy, we add the following YAPS rule:

```
(defp define-dynamically-disconnected-search-location
     (hypothesized object -object)
     (goal (define search location for road patch -object))
test (null (<- -object 'search-location))
     (eq (<- -object 'search-strategy) 'dynamically-disconnected)
-->
    (<- -object 'set-search-location
    (list (<- (<- *yaps-db* 'scene-model)
    ':predict-extended-left-feature-seed
    MAX-EXTENDED-SEARCH-DISTANCE)
    (<- (<- *yaps-db* 'scene-model)
    ':predict-extended-right-feature-seed
    MAX-EXTENDED-SEARCH-DISTANCE)))).
```

In this rule, the rule consequent defines the *search location* to be 10 m (the value of MAX-EXTENDED-SEARCH-DISTANCE) from the last road patch in the scene model. The results of this new strategy are depicted in Fig. 15.

C. Dynamically Decreasing Road Projection

Returning to our original search strategy, the planner aborts the disconnected search strategy in the event of an unverifiable road patch hypothesis. Rather than returning to the connected search strategy, we might try to relax our projected search location and rehypothesize the road patch halfway between the unverified road patch and the last verified road patch in the scene model. We have as our original rule

```
(defp road-patch-disconnected-to-connected-retry
     (hypothesized object -object)
     (goal (process road patch hypothesis -object))
test (eq (<- -object 'search-strategy) 'disconnected)
     (cond ((not (null (<- -object 'total-confidence)))
     (< (<- -object 'total confidence)
     MIN-ROAD-PATCH-CONFIDENCE)))
-->
     (<- -object 'set-search-strategy 'connected)
     (<- -object 'set-back-connected-planar-ribbon
     (<- (<- *yaps-db* 'scene-model)
     ':retrieve-most-recent-road-patch))
     (<- -object 'set-search-location
     (list (<- (<- (<- -object 'back-connected-planar-ribbon)
     'has-part-left-world-segment) 'endpoint-B)
     (<- (<- (<- -object 'back-connected-planar-ribbon)
     'has-part-right-world-segment) 'endpoint-B)))
     (<- -object 'set-has-part-left-world-segment nil)
     (<- -object 'set-has-part-right-world-segment nil)
     (<- -object 'set-actual-width nil)
     (<- -object 'set-actual-width-confidence nil)
     (<- -object 'set-parallelism-confidence nil)
     (<- -object 'set-total-confidence nil)).
```

The antecedent of this rule checks that the confidence of the road patch was too low, i.e., that the road patch was unverified, and that the disconnected search strategy was invoked to verify the road patch. The consequent of the rule clears the contents of the road patch hypothesis, resets the *search strategy* to be connected, sets the *back connected planar ribbon* to the last road patch in the scene model, and sets the *search location* to the end of the scene model. Our new search strategy results in the following rule:

```
(defp disconnected-to-halfway-retry
     (hypothesized object -object)
     (goal (process road patch hypothesis -object))
test (eq (<- -object 'search-strategy) 'disconnected)
     (cond ((not (null (<- -object 'total-confidence)))
     (< (<- -object 'total-confidence)
     MIN-ROAD-PATCH-CONFIDENCE)))
-->
     (<- -object 'set-search-location
     (<- -object 'set-search-location (halfway
     (<- (<- (<- *yaps-db* 'scene-model)
     ':retrieve-most-recent-road-patch)
     'search-location)
     <- -object 'search-location))))
     (<- -object 'set-has-part-left-world-segment nil)
     (<- -object 'set-has-part-right-world-segment nil)
     (<- -object 'set-actual-width nil)
     (<- -object 'set-actual-width-confidence nil)
     (<- -object 'set-parallelism-confidence nil)
     (<- -object 'set-total-confidence nil)).
```

In this case, the *search location* is calculated by a function finding the midpoint between the last road patch in the scene model and the unverified road patch. The new rule bears close resemblance to the original rule; we have simply removed the first two components of

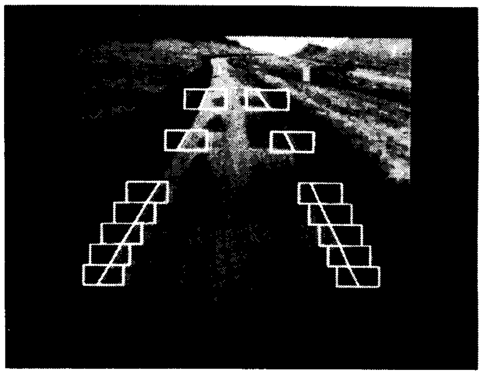
Fig. 15. Dynamically increasing road projection.

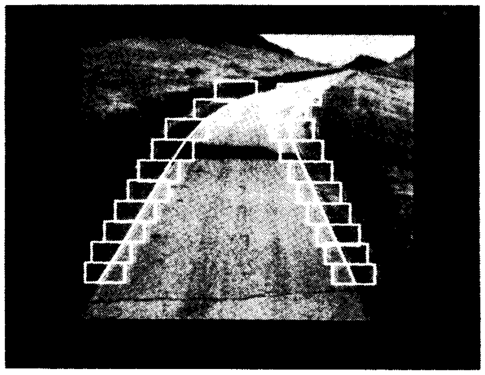
Fig. 16. Dynamically decreasing road projection.

the rule consequent and have provided a new definition of the search location. Applying this strategy to the unverifiable road patch in Fig. 12, we obtain the results depicted in Fig. 16.

IX. Related Work

As mentioned earlier, the system described in this paper functionally resembles previous work at the University of Maryland [14]. In particular, the road is represented by a series of road patches defined by their left and right boundaries. Although Waxman *et al.* address the initial location of the road (bootstrap procedure) as well as successive road following (feedforward procedure) [14], we focus only on the road-following task. The improved architecture accommodates the prototyping of new road-following strategies and vision techniques much better than its predecessor.

The decomposition of an object both by component and by level of abstraction, and the construction of hierarchical frame networks, bear close resemblance to techniques used in the VISIONS system [6]. In the VISIONS system, the long term memory (LTM) contains *a priori* visual knowledge of the world, whereas the short term memory (STM) represents the interpretation of the scene. Both the LTM and STM are structured as a hierarchy of levels of representation defining the levels of object abstraction. The control strategy first decides which partial model (frame network) to focus on, expands (hypothesizes) a node, and finally verifies the node. Although originally defined for outdoor house scenes, this work has been extended to the road-following task [3].

Lawton *et al.* [7] describe a system also resembling the VISIONS system. The STM acts as a dynamic scratchpad for the vision system, which contains object hypotheses, incoming imagery, and the results of feature extraction. When hypotheses accumulate sufficient evidence, they are moved to the LTM, which includes *a priori* terrain representations. The control structure provides both top-down and bottom-up hypothesis instantiation over the network hierarchies.

The VITS vision system [13] running on the ALV at Martin Marietta classifies the pixels of a color image as on-road or off-road and builds a road representation consisting of points in three space that form a polygonal approximation of the road boundary. Although the image processing techniques to extract the road boundary have yielded impressive results, the system lacks any reasoning ability concerning the shape of the road. In addition, the vision system is strictly bottom up. Sets of boundary points (scene models) are continually generated and passed to the reasoning subsystem, where a trajectory is computed; the higher level reasoning processes appear to have little effect on the lower level image processing.

As in the case of VITS, the Navlab mobile robot at Carnegie-Mellon [11] classifies the pixels of a color image as either on road or off road. A Hough voting procedure is then applied to determine the location of the road's vanishing point, whereas a calibrated flat-earth model yields the 3D location of the road's centerline. Obstacle detection and terrain analysis are performed by region growing on overlapping maps of surface normals; each resulting region (limited by surface discontinuities) is fitted by a plane or quadric surface. The framework within which the vision system is executed is provided by a blackboard system referred to as CODGER (for details, see [12]). The modules (obstacle avoidance, pilot, helm, color vision, display, and a central database) communicate by posting and retrieving tokens from the database; CODGER provides extensive synchronization facilities and is suited to execution on multiple processors.

Smith and Strat [10] describe an information manager that is the core of a sensor-based autonomous system. A centralized knowledge database, which is accessible to a community of independent asynchronous processes, is proposed. The representation scheme organizes data tokens in both an octree and a semantic network, thus supporting both spatial and semantic queries. The independent asynchronous processes can be activated by demons embedded in the database or by procedure call.

X. Conclusions

The system described in this report provides a flexible architecture for constructing an ALV scene model. The representation of objects as networks of frames offers a natural grouping of knowledge; the multiple layers of abstraction facilitate the addition of new sensor features in support of existing world objects. Construction of the frame network is provided by a set of modular blackboards providing top-down instantiation of the frames in the network. Each blackboard, implemented by a production system, is an "expert" in defining a particular class of frame; the English-like rules governing the invocation of knowledge sources are easy to understand and narrow the gap between control specification and implementation. From a system maintenance standpoint, all object frames, blackboards, production system tools, and object-oriented programming tools are off the shelf; these facilities are documented, tested, and readily accessible. The implementation languages supporting the system cover the needs of the programmer; YAPS offers high-level encoding of control strategy, Lisp provides symbolic manipulation, C speeds up numerical processing, and Flavors facilitates interobject communication.

The system is currently being expanded to support new planning and verification strategies. The planner is being supplemented with strategies for road following in the event that a connected road patch cannot be verified. This includes proceeding past an unverified road patch, provided that it contains a verified component, and invoking an exhaustive search for road patches in a given area; the latter strategy will be accomplished using bottom-up verification in which road patch camera segments posted at lower levels generate instances of road patches at upper levels. This integration of top-down and bottom-up verification will remove some of the burden placed on the planner of accurately predicting the location of an object.

Acknowledgment

Many people deserve thanks for their contribution to this work. In particular, the authors would like to thank J. Le Moigne and R. Waltzman for insightful discussions at the early stages of the project. D. Harwood and S. Antoy offered helpful suggestions during the design phase, and T. Kushner and T. K. Siddalingaiah provided hardware support in the implementation. The authors would also like to thank S. Stevenson and A. Rosenfeld for their excellent comments on earlier drafts of this paper.

References

[1] E. M. Allen, R. H. Trigg, and R. J. Wood, "The Maryland artificial intelligence group Franz Lisp environment, Variation 2," Tech. Rep. TR-1226, Dept. Comput. Sci., Univ. Maryland, Nov. 1983.

[2] E. M. Allen, "YAPS: Yet another production system," Tech. Rep. TR-1146, Dept. Comput. Sci., Univ. Maryland, Dec. 1983.

[3] R. C. Arkin, E. M. Riseman, and A. R. Hanson, "AuRA: An architecture for vision-based robot navigation," in *Proc. 1987 DARPA Image Understanding Workshop* (Los Angeles, CA), Feb. 1987, pp. 417-431.

[4] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973, pp. 393-396.

[5] C. L. Forgy, "OPS5 user's manual," Tech. Rep. CMU-CS-81-135, Dept. Comput. Sci., Carnegie-Mellon Univ., July 1981.

[6] A. R. Hanson and E. M. Riseman, "VISIONS: A computer system for interpreting scenes," in *Computer Vision Systems*. New York: Academic, 1978.

[7] D. T. Lawton, T. S. Levitt, C. C. McConnell, P. C. Nelson, and J. Glicksman, "Environmental modeling and recognition for an autonomous land vehicle," in *Proc. 1987 DARPA Image Understanding Workshop* (Los Angeles, CA), Feb. 1987, pp. 107-121.

[8] J. Le Moigne, A. M. Waxman, B. Srinivasan, and M. Pietikainen, "Image processing for visual navigation of roadways," Tech. Rep. CAR-TR-138, Cent. Automat. Res., Univ. Maryland, July 1985.

[9] M. Minsky, "A framework for representing knowledge," in *The Psychology of Computer Vision*, P. H. Winston, Ed. New York: McGraw-Hill, 1975.

[10] G. B. Smith and T. M. Strat, "Information management in a sensor-based autonomous system," in *Proc. 1987 DARPA Image Understanding Workshop* (Los Angeles, CA), Feb. 1987, pp. 170-177.

[11] C. Thorpe, M. H. Hebert, T. Kanade, and S. Shafer, "Vision and navigation for the Carnegie-Mellon Navlab," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 3, May 1988.

[12] A. Stentz and S. Shafer, "Module programmer's guide to local map builder for ALVan," Dept. Comput. Sci., Carnegie-Mellon Univ., July 1985.

[13] M. A. Turk, D. Morganthaler, K. D. Gremban, and M. Marra, "VITS—A vision system for autonomous land vehicle navigation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 3, May 1988.

[14] A. M. Waxman, J. Le Moigne, L. S. Davis, E. Liang, and T. Siddalingaiah, "A visual navigation system for autonomous land vehicles," Tech. Rep. CAR-TR-139, Cent. Automat. Res., Univ. Maryland, July 1985.

[15] R. J. Wood, "Franz flavors: An implementation of abstract data types in an applicative language," Tech. Rep. TR-1174, Dept. Comput. Sci., Univ. Maryland, June 1982.

Petri Net-Based Emulation for a Highly Concurrent Pick-and-Place Machine

ANNA SCIOMACHEN, STEVE GROTZINGER, AND FRANCESCO ARCHETTI

Abstract—**The emulation of pick-and-place machines, in order to compute their throughput rate for different products, is a key ingredient for capacity planning and scheduling of high-volume assembly lines where these machines are used.**

In this communication it is shown how an emulator can be built in the framework of Petri nets and how it can be instantiated for a particular setup and sequence of placements.

Three advantages are argued for the proposed emulator. First, the emulation code can be generated automatically. Second, the model graphically represents the concurrency and synchronization aspects of the machine's operations. Third, it allows a formal representation of the machine's operations.

Manuscript received January 28, 1988; revised February 23, 1989.

A. Sciomachen and F. Archetti are with the Department of Information Sciences, University of Milan, Milan, Italy.

S. Grotzinger is with Manufacturing Research, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

IEEE Log Number 8930423.

I. Introduction

The activity reported in this communication is part of a research project aimed at establishing a framework in which to develop modeling and simulation tools for an important class of CNC machines, namely, pick-and-place machines, whose use is becoming increasingly important in high-volume/low-cost production of printed circuit boards.

Typically, these machines have a very high throughput potential or burst rate resulting from a highly concurrent design. This very feature makes them vulnerable to severe performance degradation when the sequence of placements and the arrangement of the components on the delivery carrier are not interrelated so as to exploit the concurrency of the machine.

In order to plan the capacity of the assembly line and to schedule the production, an accurate estimation of the machine cycle time for a given product is required before the start of production.

Unfortunately, creating a setup and sequencing for a new board requires extensive and time-consuming machine reprogramming and components loading. The elimination of this trial-and-error method on the machine is the main reason behind the need to develop emulators for these machines. DPERT, an emulator of a specific machine of this class, has been written in PL/1 and extensively tested and validated for different setups and pick-and-place sequences [2].

In this communication, a different approach, first suggested in [4], based on Petri net models and the use of a software tool called MEGAS [10] for their analysis and simulation, is shown to be a general framework for the development, validation, and numerical utilization of an emulator for this machine.

Petri nets that represent asynchronous concurrent systems [9], have been increasingly used recently to model manufacturing systems [1], [3], [6] and their control [7], [11]-[13].

One advantage of Petri net based models over other design languages is that they can be analyzed for structural properties of the system-like boundedness and liveness, i.e., absence of deadlocks.

Petri net-based models are executable. This feature is particularly important for the problem considered in this communication because it allows the emulation to be generated automatically, using a package such as MEGAS, from the Petri net and the production data streams without the need of writing more software for the specific emulation instance.

The challenge in emulation by Petri nets for this machine is to model, in a dynamic environment, the concurrency, synchronization, and control of the machine's operations. One wants to keep the model complete yet easy to understand and modify, executable and yet with a good run-time performance.

In Section II, the structure of the machine is briefly outlined stressing the role of concurrency and the situations which may adversely affect the throughput rate. In Section III, after a brief reminder of the basic definitions and properties of Petri nets, the model of the machine is presented. To the authors' knowledge, this model is considerably more complex than many presented in the literature. This section is written not only to describe a specific model but also to outline a general methodology for building Petri nets models. In Section IV, it is explained how the Petri net model can be