Interactive Model-Based Image Understanding

Daryl T. Lawton Warren F. Gardner

Abstract

This paper describes a general architecture for an interactive model-based vision system. A human specifies a limited amount of information which establishes a context for autonomous interpretation of images. Object models are described by constraints specifying necessary geometrical properties and relationships between objects. The use of constraints allows for flexible object instantiation. A user can indicate an object in a scene and this directs perceptual processing routines as well as constraining future object instantiations. This interactive model-based concept has been applied to the domain of vehicle tracking, and this paper concludes with several processing examples from this domain.

1 Introduction

Efforts to develop intelligent and autonomous systems for operation in complex, natural domains have been largely unsuccessful to date, in spite of continued advances in the underlying technologies. There remain unresolved and fundamental difficulties in terms of the necessary computational power, the required complexity of perceptual systems which can operate in outdoor environments, and the corresponding complexity of planning and reasoning systems. Previous work has addressed many of these problems by stressing the importance of telerobotic and interactive systems [9, 11]. This is a realistic approach to fielding advanced technology in the short term, and also provides a long term framework for developing autonomous systems. An interactive, semi-autonomous system can significantly amplify the capabilities of a human, and also yields an evolutionary approach as autonomous system capabilities are developed and begin to replace human controlled functions.

The approach described here is to develop a model-based vision system that a human can interactively control. The inspiration for this system comes from the interactive vehicle tracking system described in [5]. The human is responsible for building a 3D model of the world by instantiating object models. These models are maintained by the vision system and used to constrain future processing. Communication between the vision system and the human can then take place in the context of this shared model of the world which makes possible infrequent, semantically meaningful, and low bandwidth communication.

The particular system we present is for tracking vehicles in outdoor scenes. A human can manipulate

Figure 1: Model-based system architecture

models of objects such as gravity, roads, and vehicles to interpret imagery from a camera. Once an interpretation is in place, the vision system can autonomously refine and extend the interpretations, detect and track vehicles, and report back to a human about unusual occurrences or behavior that cannot be accounted for. For example, a human will indicate that a particular area is a road, and another area is a vehicle. The system can then track the vehicle and determine if it goes off the road, or if it is behaving inconsistently with respect to the model of a vehicle. We begin by reviewing the basic architecture of the interactive model-based vision system, and then discuss this system within the context of vehicle tracking.

2 System Architecture

The model-based system architecture is shown in Figure 1. The system is made up of two databases that a human can access and manipulate through a user interface. The human accesses models of the various types of objects stored in the Object Model Database to build an interpretation of the current scene which is stored in the World Model Database. As the user manipulates object models within the world model, the models are projected back against images for interactive control and to initiate processing.

The Object Model Database contains generic models of objects, relationships, and events for terrestrial scenes. This involves objects such as terrain patches, roads, vehicles, and gravity. We distinguish

between two different types of objects: terrestrials which correspond to conventional objects found in the world such as roads and cars, and primitives which correspond to basic entities and relationships which make up terrestrial objects. Primitive objects describe characteristics such as shape constraints, material composition, and relationships between parts. Thus, in addition to describing its shape, the model of a car needs to include that a car is acted on by gravity and will have a preferred type of orientation and attachment with respect to the ground surface. Object models are described by sets of constraints [1, 3, 6, 8] which must be satisfied. A simple constraint is that the value of some parameter associated with an object model is bounded. More complicated constraints deal with relations between objects. The human will in general specify a limited amount of information for an object, and the system will use the constraints and associated processing actions to then refine the instantiation of an object.

The World Model Database describes the three dimensional world of objects and situations surrounding the vision system. It is initially formed by the human accessing models in the object data base and instantiating them. There are two types of controllers associated with the World Model Database. The Constraint Controller checks for consistency in the world model. The constraint controller uses the constraints which define an object or relationship to refine an instantiation or to find a violation or inconsistency and ask the human for help. The Perceptual Processing Controller deals with the extraction of information from images. The constraints in an object model specify the types of processing that are necessary to obtain this information. When the human indicates that a road is located somewhere, this constrains the type of tracking and feature extraction processes that are used.

The user interface consists of images, with object models displayed as graphical overlays. When the user instantiates an object model, the model is projected back against the image allowing the user to interactively manipulate the object within the world model.

3 Object Models

The interactive model-based system described in the previous section was applied to the domain of vehicle tracking. This domain contains complex lighting conditions and independently moving objects, yet there are also constraints which can be exploited through temporal and geometric models. Three models were developed to enable the tracking of vehicles in outdoor environments. The three types of objects modeled were roads, gravity, and vehicles.

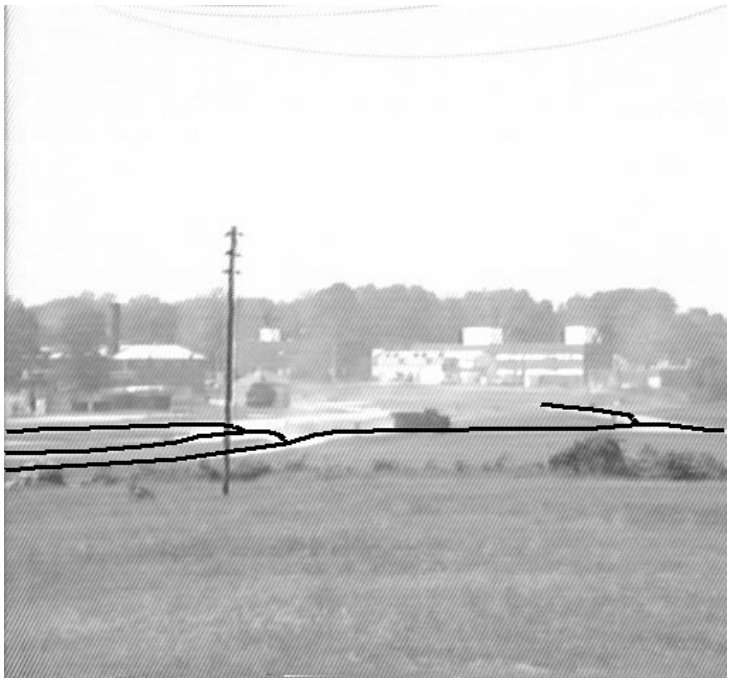

Figure 2: 2D road model

3.1 Road Model

Knowledge of the position of a road is obviously beneficial to a vehicle tracking system. Information about road locations can restrict the amount of processing necessary to detect and track vehicles. The road model consists of a sequence of 2D or 3D line segments. A road is instantiated by drawing this sequence of lines on top of an image. This can be done by picking points with a mouse, or by tracking a vehicle moving along the road. Figure 2 shows a 2D road model that was instantiated by a user with a mouse.

3.2 Gravity Model

Knowledge of the direction of gravity is useful in constraining the orientations of other objects in the world model. The position and orientation of falling objects are obviously affected by gravity. Man-made objects such as buildings lie parallel to the direction of gravity, and vehicles are constrained to travel at angles close to the plane perpendicular to the gravity direction.

The gravity direction is represented by a 3D vector. This direction is presented graphically in two ways. The vector is given a planar base and drawn within a sphere to aid the user in determining its current 3D position as shown in Figure 3. A vector in the direction opposite gravity is also drawn in a

Figure 3: User interface for the gravity model

lighter color to aid the user in determining the current gravity direction when the planar base interferes with the view of the gravity vector. This representation also acts as the user interface for the gravity model. A user positions the gravity vector within the sphere by clicking on the vector and dragging it with a mouse. The 3D gravity vector is also shown by projecting the direction onto an image at different positions, creating the 2D gravity field shown in Figure 4. As a user moves the gravity vector within the sphere, the gravity field overlaying the image reflects the change in direction.

3.3 Vehicle Model

The most important model for a vehicle tracking system is certainly the vehicle model. The vehicle model geometry consists of six parameters: length, width, height, hood length, hood height, and wheel height. A perspective view of the vehicle model is shown in Figure 5. A user instantiates a vehicle within the world by placing this graphical model in the image. The six degrees of freedom of the graphical model can be manipulated using a mouse. Grabbing the vehicle with the left mouse button pressed, the user can move the vehicle parallel to the image plane. Pressing both left and middle buttons allows the user to move the vehicle perpendicular to the image plane, and pressing only the middle button allows the user to rotate the vehicle model.

4 Difference Tracker

The type of image processing or vehicle tracking that is employed is dependent upon the models that are instantiated. The current processing routines consist of two types of trackers and restricted segmentation

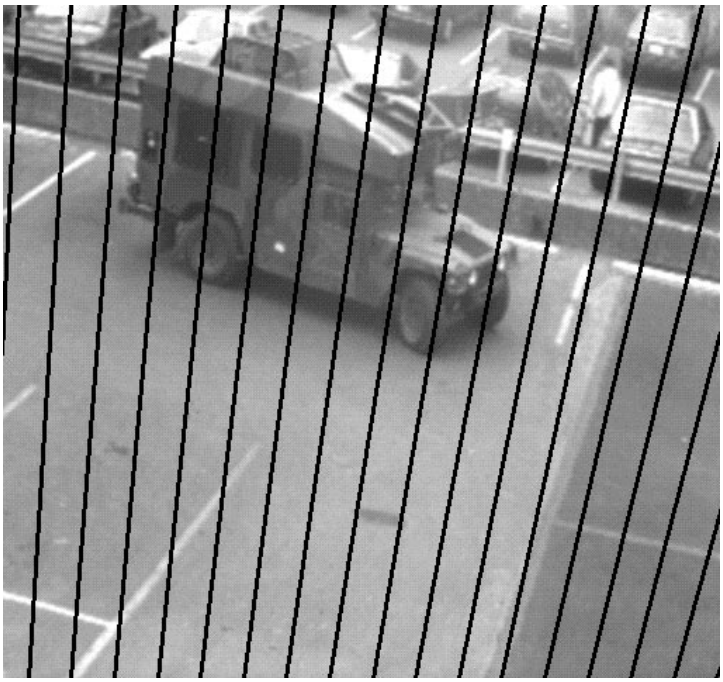

Figure 4: Gravity model projected onto an image

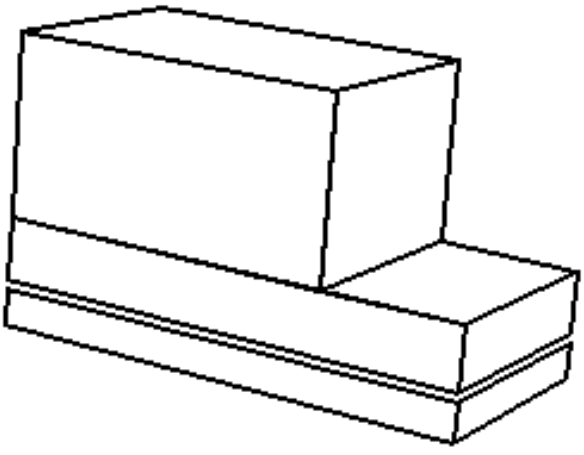

Figure 5: Perspective view of the vehicle model

Figure 6: Image areas lying near the road model are smoothed

and interest operators which are applied when models are instantiated. One type of tracker is dependent upon a 2D or 3D road model, and can yield information allowing the automatic instantiation of a vehicle model. This tracker does not require the presence of these models, but given a road model this tracker is able to exploit the information provided by the model to construct a more efficient and intelligent solution to the tracking problem. This tracker is called a difference tracker, since it relies on information obtained from image differencing or subtraction.

The difference tracker operates with respect to an instantiated 2D or 3D road model. It determines regions above the indicated road areas which are changing over time and are also moving in a consistent direction. The tracker locates the front and rear of a vehicle and can use this information to instantiate a vehicle model. If a 3D road model has been instantiated, the location of the vehicle can be further constrained. The information recovered from this tracker can also be used to restrict the extraction of features for the local translation tracker discussed in Section 5.

The first step in tracking an object is to reduce the image noise by convolving consecutive images in a motion sequence with a low-pass filter. The images shown in this paper were smoothed using a Gaussian filter. If no models are present, the entire image must be convolved with this filter. However, given a 2D road model, the filter is only convolved with pixels that are close to the road. The road model shown in Figure 2 is used to constrain the smoothing process, resulting in the image shown in Figure 6.

Once the images have been smoothed, the algorithm begins to search for areas of motion that lie near the road. This is accomplished through image subtraction. Pixels from temporally consecutive images that are situated near the road model are subtracted. If the result of this subtraction is greater than a threshold, the environmental object corresponding to this pixel position is assumed to have undergone motion. This pixel is marked as a motion pixel, and a region growing process begins.

A 3D road model with accompanying depth information would constrain the size of the projection of an object traveling along the road. However, without any depth information the size of an object within the image is unknown. Since the size of the object within the image is unknown, the search for all areas of motion associated with an object is accomplished through region growing. Once a pixel near

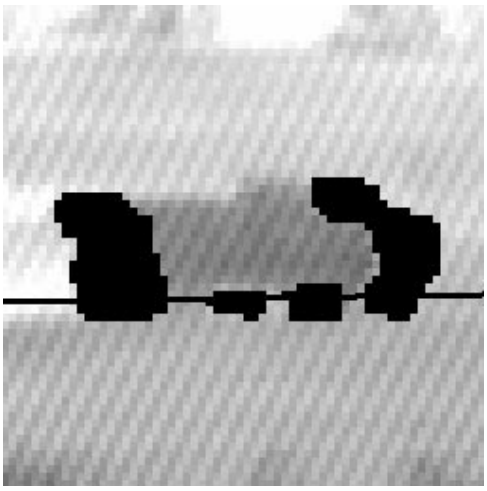

Figure 7: Areas of motion extracted by region growing

the road has been identified as a motion pixel, its neighbors are also examined using the subtraction technique discussed above. If any of the neighboring pixels contain motion, their neighbors are also examined. This recursive procedure continues until no more motion pixels can be found. The areas of motion found using the first two images in a motion sequence are shown in Figure 7. In this figure the areas of motion are shown overlaid on a sub-image extracted from the first image in the motion sequence. The entire image is shown in Figure 2.

Once the areas containing motion have been identified, the centroid of these areas is located. The motion pixels are fitted with a bounding box and the center of this box is taken as the centroid as shown in Figure 8. Over time a 2D trajectory can be constructed. Figure 9 shows the results of processing a five image sequence. The trajectory is overlaid on the last image in the motion sequence. In the presence of a 3D road model the 3D motion of the object can be inferred. The introduction of a simple easily manipulated road model results in an efficient and simple tracker. This tracker can be used to scan road areas for any changes, notifying a user or instantiating a vehicle model when they occur. The instantiation of a vehicle model spawns the more sophisticated local translation tracker discussed in the next section.

5 Local Translation Tracker

Vehicles have a limited turning radius, resulting in an axis of rotation that is often far away from the vehicle. A vehicle tracker was constructed based upon this constraint and using the concepts presented

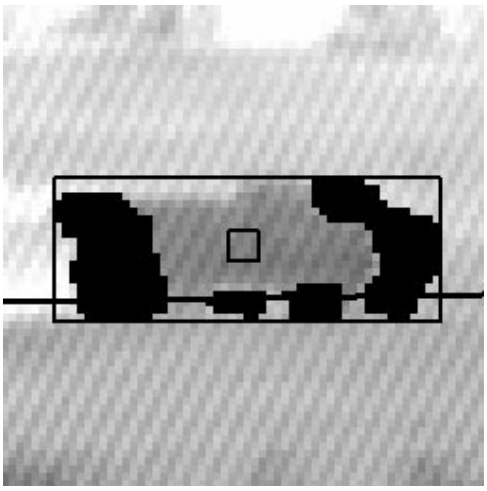

Figure 8: Center of the areas of motion

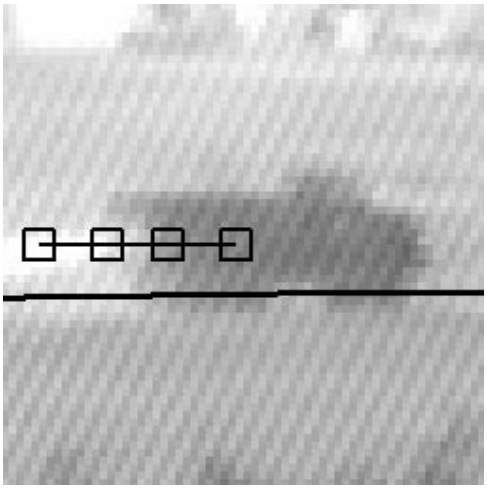

Figure 9: Two dimensional motion trajectory

in [4]. A vehicle is subdivided into local regions and each region is treated as if it had undergone purely translational motion. The extraction and grouping of features can be done automatically, but is more efficient and reliable when directed by a vehicle model. Instantiation of the vehicle model shown in Figure 5 spawns the local translation tracker. This tracker consists of feature extraction and grouping, and 3D trajectory computation via local translational decomposition and edge matching. The information derived from this tracker can be used to determine a 3D road model, as well as refine the attributes of the instantiated vehicle model through temporal modeling with the Kalman Filter. This tracker is discussed in more detail in [2].

5.1 Feature Extraction from a Model

The local translation tracker requires features which can be matched in successive images. The type of features used are conventional masks of image pixels, extracted from distinct areas of the image. In the examples shown in this paper, the masks are 9x9 pixel arrays. These features are extracted through the use of the Moravec interest operator [7]. The information provided by models is used to direct feature extraction. The local translation tracker estimates the local translation of different portions of the vehicle, thus each portion requires a certain density of features for this estimation. Sequential images are registered by calculating a 2D displacement parallel to the image plane. This requires only a small set of features over the entire image. The vehicle model is used to determine the appropriate density of features at each image position. Figure 10 shows a set of features extracted from an image with a vehicle model. Notice that the density of features is much higher within the vehicle model where the features are used to calculate several 3D displacements. Outside the vehicle the features are used to calculate a single 2D displacement, and so fewer features are necessary.

The vehicle model is also used to group the features lying on the vehicle. The local translation tracker computes a direction of translation at different points along the surface of the vehicle. Vehicle motion is locally planar, so the majority of rotation will be about the normal to this plane. Therefore, features that lie close together with respect to the length of the vehicle have similar trajectories. The local translation tracker groups features using this distance criterion. The results presented in this paper were all produced by grouping the vehicle features into three groups: the front, center, and rear features. These groups are shown in Figure 10.

The vehicle model is also used to determine areas in which feature extraction should be performed due to occlusion. Areas in the background are occluded by the vehicle, and as the vehicle moves these

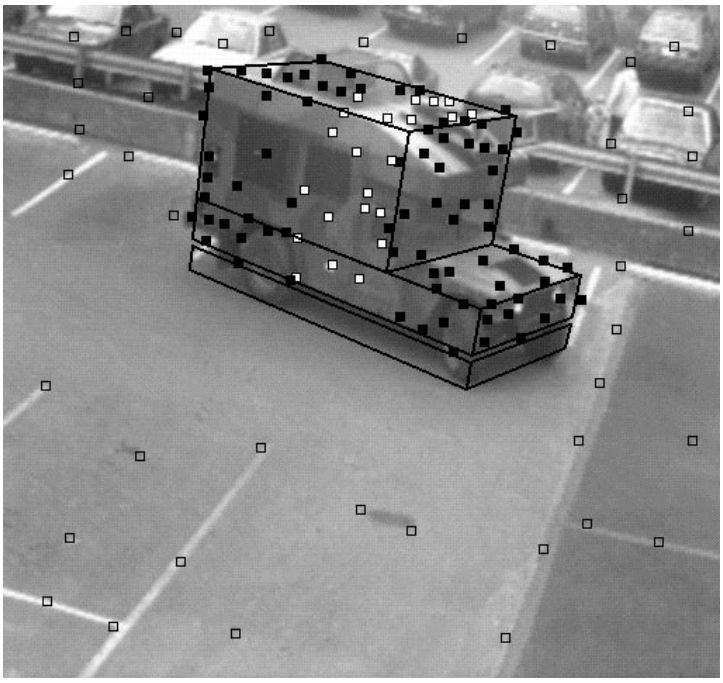

Figure 10: Features grouped using a vehicle model

areas become visible and feature extraction is performed. Self-occlusion by the vehicle is also detected using the vehicle model. In this case, vehicle rotation exposes previously occluded areas on the vehicle.

The feature positions used to derive the local translation vectors are not direct measurements of the position of the vehicle. It is necessary to measure quantities in the image that are directly related to the position of the vehicle for the purpose of initializing the vehicle position, and measuring the quality of subsequent estimated positions. Searching for vehicle features such as headlights or wheels is one way to measure the position of the vehicle. Currently, the edges of the vehicle model provide the necessary measurements.

A sub-image is constructed about each edge that is to contribute to the measurement. The size of this sub-image is determined using the covariance matrices associated with the vehicle motion model. The sub-image is then convolved with a one dimensional edge mask oriented in the direction of the edge. Each row of the resulting sub-image is then summed, and the maximum values are considered possible positions for the edge. The position of the vehicle can be determined given three edge positions. Currently, three edges are used to solve for the vehicle position, and then the error sums of additional edges are used to verify the solution. The position is calculated using the three maximum positions for each of the three edges, so twenty-seven possible positions are determined. The additional edge error

sums are added to the error sum for each possible position, and the maximum value is chosen as the position of the vehicle. A more detailed discussion of the vehicle motion model is given in [2].

5.2 Locally Planar Motion

Often the motion of a vehicle is restricted to a plane determined by the local road or surface orientation. In this case, it is possible to associate 3D information with extracted image features. The directions of motion are constrained to be parallel to the given plane, so the possible directions of motion for the local translation tracker are restricted to a circle on the unit sphere whose orientation is parallel to the plane of motion. Locally planar motion also results in a smaller temporal filter.

The gravity model defines a plane which can be used to restrict the motion of the vehicle model. Instantiation of the gravity model places constraints on the orientation of the vehicle model. If a vehicle model is instantiated, gravity will initially be aligned with the vehicle axis. As the gravity direction is changed, the vehicle orientation also changes. When a vehicle model is manipulated in the presence of a gravity model, the user will be restricted to rotations about the gravitational axis.

5.3 Processing Example

The local translation tracker described in this section was tested on several image sequences. The sequence presented here was shot from a hand-held camera and contains a turning vehicle. This type of sequence contains rotational motion which is the most difficult case of motion for the local translation tracker. The camera was uncalibrated, so the camera center and focal length was unknown. The optical axis was assumed to pierce the camera at the center of each image. The focal length was set by the user prior to the tracking process. This sequence contained forty-five 512x480 images. For the results shown in this section a focal length of 950 pixels was used (field of view of 30.2 degrees). Initially a vehicle model and a gravity model were instantiated by a user. The position of the gravity model is shown in Figures 3 and 4. The initial vehicle position is shown in Figure 11. Figures 12-15 show the position of the vehicle model at image frames 10, 20, 30, and 40 respectively.

The actual vehicle length was unknown, so a value of 4 meters was chosen in order to calculate motion statistics. The remaining system parameters were then scaled relative to the truck length. Figure 16 shows the forty-five vehicle positions projected onto the plane perpendicular to the direction of gravity. The vehicle starts in the upper left hand corner of the plot and moves to the lower right corner. The height of the vehicle above the gravity plane is shown in Figure 17. This figure shows that

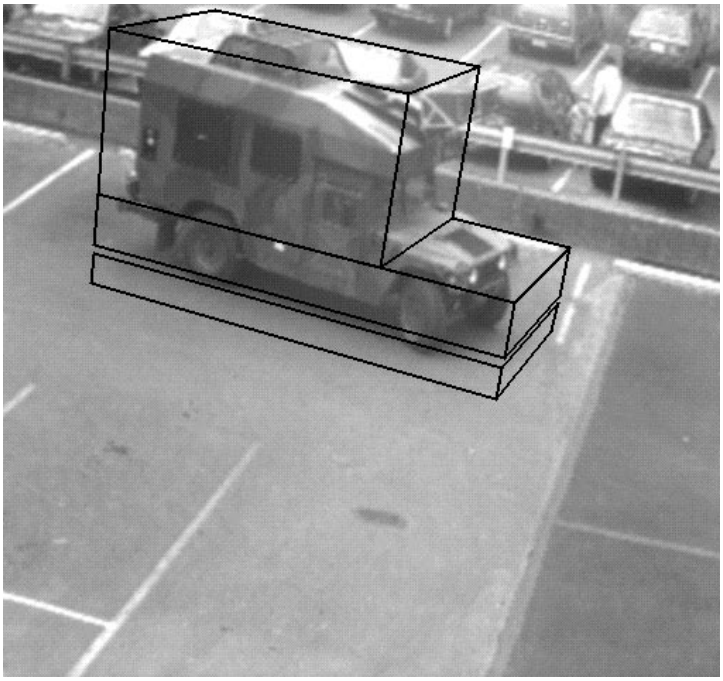

Figure 11: User-instantiated vehicle model

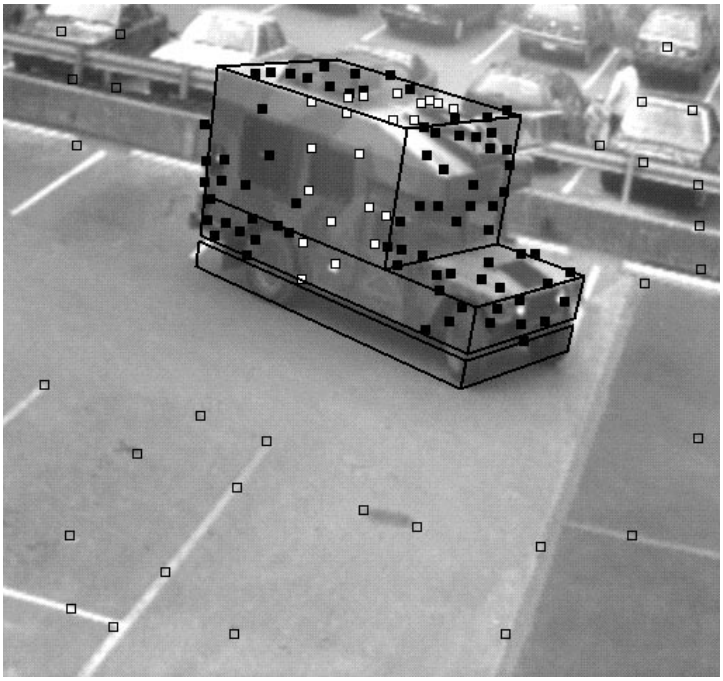

Figure 12: Vehicle model and features for image frame 10

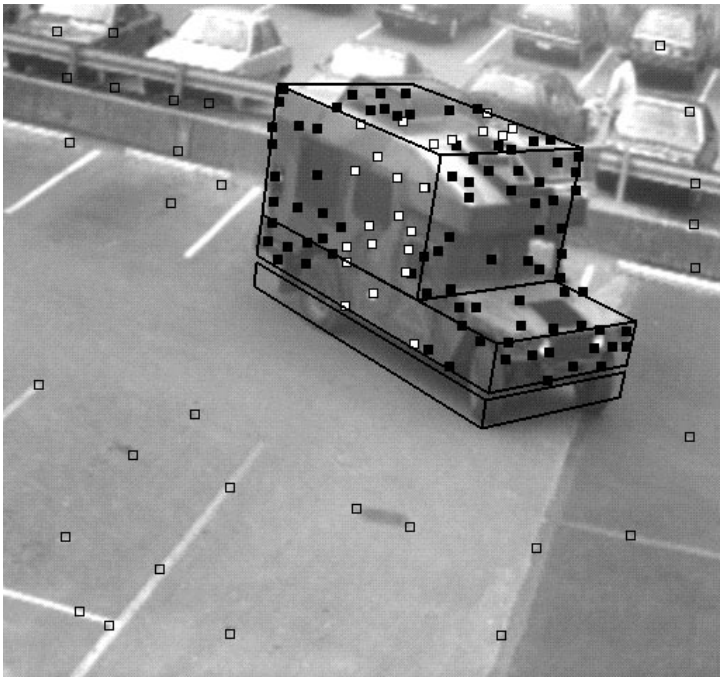

Figure 13: Vehicle model and features for image frame 20

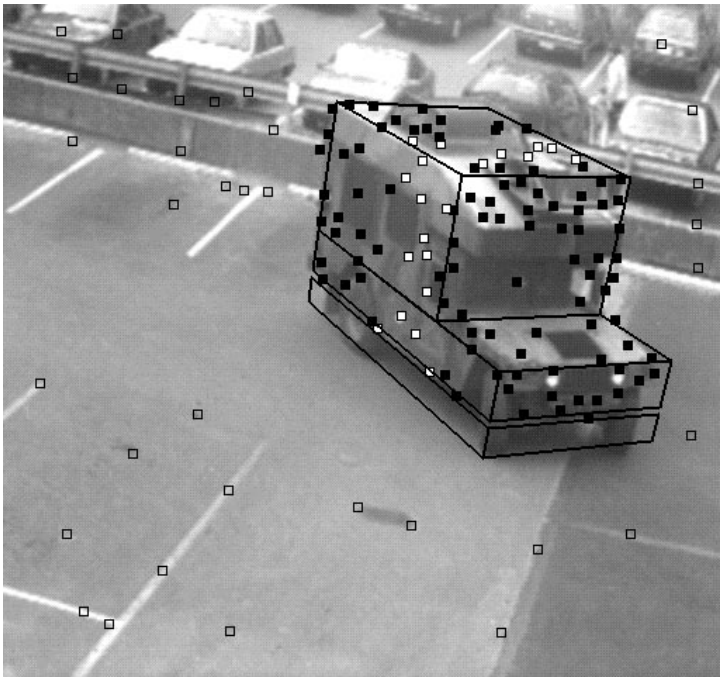

Figure 14: Vehicle model and features for image frame 30

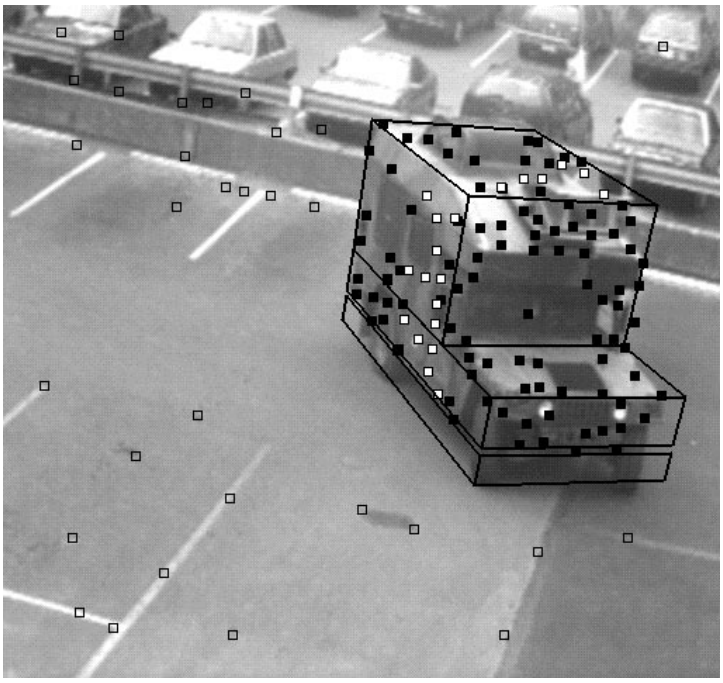

Figure 15: Vehicle model and features for image frame 40

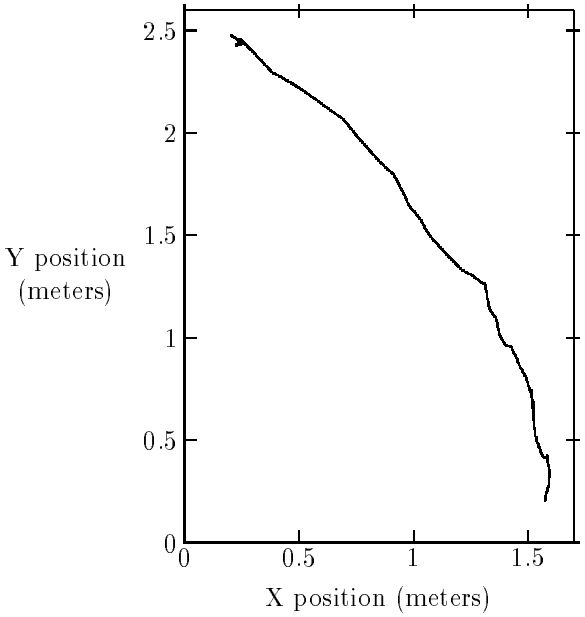

Figure 16: Vehicle trajectory for the 45 frame sequence

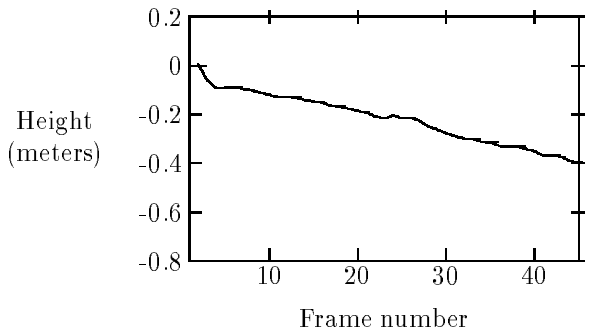

Figure 17: Height of the vehicle above the gravity plane

the vehicle is moving slowly away from the plane of gravity. This is due to errors in the focal length, vehicle model, and the user-instantiated gravity plane. The height information shown in Figure 17 is presented to the user, so that the appropriate action can be taken. The user can then adjust any number of these parameters in an attempt to correct for the height error.

6 Future work

This paper described a general architecture for an interactive model-based vision system, and explored this system within the context of vehicle tracking. The model-based tracking system was able to successfully track vehicles under complex, noisy conditions, thus illustrating the power in constraining the environment through the use of a few carefully chosen models. There are several areas in which this work can be extended.

- Extending the user interface to use a wide range of interactive devices such as a data glove and other 3D positioning devices.
- Extending the number and complexity of the models that are used. This includes developing an explicit inheritance hierarchy of models.
- Using more involved perceptual processing, especially for segmentation which is sensitive to the material properties of objects and for the extraction of features to match directly to models.
- Extending the techniques to make use of information from multiple cameras.

The current user interface consists of a pull-down menu and a mouse. The models are instantiated by choosing items from the menu. The user can then manipulate the objects in the world database by using the mouse. The introduction of more sophisticated interactive devices such as a data glove

would allow the development of a more intuitive interface. Users could grab objects from the object database and place them into the world database quickly and accurately. Another limitation of the current system is that the models are manipulated and projected onto a static image. A real-time system would necessitate users instantiating the models in a dynamic scene. This type of instantiation will present new complexities, but also result in additional opportunities for exploiting the user input.

Another area that warrants future research is in the development of models. Perceptual processing such as segmentation which is sensitive to the material properties of objects would result in more robust objects. Knowledge about additional object components would also be valuable. The method of combining constraints imposed by instantiated objects is another area for future research. Currently inter-object constraints are represented implicitly within each of the object models. This is reasonable when dealing with a small number of models, but as the number and complexity of models grow there is a need to represent and maintain these constraints explicitly. One way this could be accomplished is through the use of an existing constraint satisfaction algorithm such as DeltaBlue [10].

The use of multiple cameras would provide valuable 3D information that is unavailable from a single camera. However, the model instantiation and manipulation processes become more involved when dealing with multiple input sources. Multiple input sources also raises problems of camera calibration. The use of multiple cameras in an interactive model-based system is an interesting research problem, and it raises interesting sub-problems such as the interactive model-based calibration of a set of cameras.

References

[1] A. Borning, "The programming language aspects of thinglab, a constraint oriented simulation laboratory," *ACM Transactions on Programming Languages and Systems*, vol. 3, no. 4, p. 353ff, 1981.

[2] W. F. Gardner and D. T. Lawton, "Interactive model-based vehicle tracking," Tech. Rep. GIT-GVU-95-33, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA, September 1995.

[3] D. T. Lawton, "Constraint-based inference from image motion," in *Proceedings of AAAI-80*, 1980. Stanford, CA.

[4] D. T. Lawton and W. F. Gardner, "Translational decomposition of flow fields," in *Proceedings of the DARPA Image Understanding Workshop*, pp. 697–705, April 1993. Washington, D.C.

[5] D. T. Lawton, W. F. Gardner, and J. Kim, "An interactive model based vision system for vehicle tracking," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 403–409, May 1993. Atlanta, GA.

[6] W. Leler, *Constraint Programming Languages: Their Specification and Generation*. Reading, MA: Addison-Wesley, 1988.

[7] H. P. Moravec, "Towards automatic visual obstacle avoidance," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, p. 584, August 1977.

[8] J. Mundy, P. Vrobel, and R. Joynson, "Constraint-based modeling," in *Proceedings of the DARPA Image Understanding Workshop*, May 1989. Stanford, CA.

[9] C. Ruoff, J. Bowyer, T. Brooks, J. Hanson, K. Holmes, and B. Wilcox, "Autonomous ground vehicles: Control system technology development," tech. rep., Jet Propulsion Laboratory, Pasadena, CA, October 1984.

[10] M. Sannella, J. Maloney, B. Freeman-Benson, and A. Borning, "Multi-way versus one-way constraints in user interfaces: Experience with the deltablue algorithm," Tech. Rep. 92-07-05a, Department of Computer Science and Engineering, University of Washington, May 1993.

[11] R. Steeb, S. Cammarata, S. Narain, J. Rothenberg, and W. Giuarla, "Cooperative intelligence for remotely piloted vehicle fleet control, analysis, and simulation," tech. rep., Rand Corporation, Santa Monica, CA, 1986.