Model-based vision for car following

Henry Schneiderman
Marilyn Nashman
Ron Lumia

National Institute of Standards and Technology
Robot Systems Division
Building 220, Room B127
Gaithersburg, MD 20899
e-mail: hws@cme.nist.gov, nashman@cme.nist.gov, lumia@cme.nist.gov

ABSTRACT

This paper describes a vision processing algorithm that supports autonomous car following. The algorithm visually tracks the position of a 'lead vehicle' from the vantage of a pursuing 'chase vehicle.' The algorithm requires a 2-D model of the back of the lead vehicle. This model is composed of line segments corresponding to features that give rise to strong edges. There are seven sequential stages of computation: 1. Extracting edge points; 2. Associating extracted edge points with the model features; 3. Determining the position of each model feature; 4. Determining the model position; 5. Updating the motion model of the object; 6. Predicting the position of the object in next image; 7. Predicting the location of all object features from prediction of object position. All processing is confined to the 2-D image plane. The 2-D model location computed in this processing is used to determine the position of the lead vehicle with respect to a 3-D coordinate frame affixed to the chase vehicle. This algorithm has been used as part of a complete system to drive an autonomous vehicle, a High Mobility Multipurpose Wheeled Vehicle (HMMWV) such that it follows a lead vehicle at speeds up to 35 km/h. The algorithm runs at an update rate of 15 hertz and has a worst case computational delay of 128 ms. The algorithm is implemented under the NASA/NBS Standard Reference Model for Telerobotic Control System Architecture (NASREM) and runs on a dedicated vision processing engine and a VME*-based multiprocessor system.

1. Introduction

In recent years there has been increasing interest in the development of autonomous driving methods. Much effort has gone into the development of visual guidance using road cues. More recently, there has been interest in sensor guided methods whereby an autonomous vehicle is guided by following a lead vehicle. This paper describes a vision-based algorithm for tracking the position of a "lead vehicle" from the vantage of a pursuing "chase vehicle."

2. Model-based visual tracking algorithm

This algorithm is designed to track a target of known geometry. The target -- the back of the lead vehicle -- is modelled by line segments where each segment corresponds to a feature that gives rise to a strong edge. Motion is tracked along 3 degrees of freedom: the 2-D position of the target and a scale factor proportional to the range from the camera to the target. The overall processing involves a loop of seven sequential stages of computation (see Figure 1):

1. Edge extraction - Extracting edge point position and orientation.
2. Data association - Determining which edges most likely correspond to each model feature.
3. Feature location- Least squares estimation of each feature location using the edge points associated to that feature.
4. Position and relative size determination - Least squares estimation of overall model position using the estimated model feature locations.
5. Motion model update - Recursive least squares update of motion models for each position parameter using estimate of position.

*Certain commercial equipment, instruments, or materials are identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by NIST, nor does it imply that the materials or equipment identified are necessarily best for the purpose.

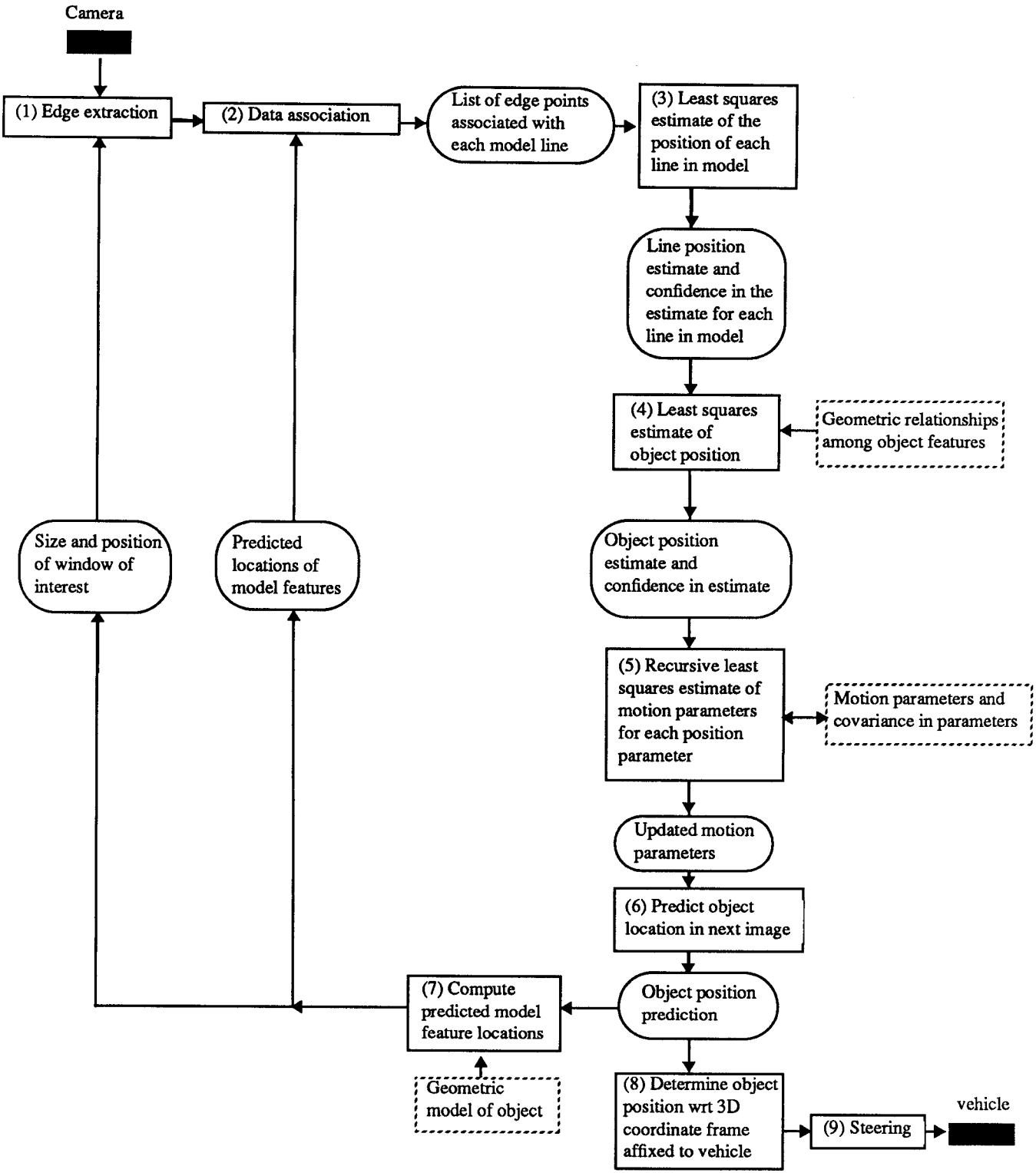

Figure 1. Processing overview

6. Model position and size prediction - Use motion models to predict object location and size in next image frame.
7. Feature position prediction - Use predicted object location and size to determine the predicted location of all model features.

This sequence of operations is repeated for each new image. Video imagery from a CCD video camera mounted above the cab of the vehicle provides the input to stage 1.

In Figure 1, computational modules are shown by rectangular boxes. The primary data exchanged between the computational modules are shown by ovals. Interaction between the geometric model and various computations are indicated by the dashed boxes. All processing within the loop is represented with respect to the 2-D image plane. The 2-D target location computed in this processing is used to compute the position of the lead vehicle with respect to a 3-D coordinate frame affixed to the chase vehicle (stage 8). Using the computed position of the lead vehicle, the HMMWV chase vehicle is steered to follow the path of the lead vehicle (stage 9). The steering algorithm and the vehicle are described in [1 2]. The geometric model of the target and each of the other computations are fully described below.

2.1. Target model

Two types of information are modeled in this system: static information which is specified *a priori* and dynamic information that is continually updated to describe the current state of the system.

2.1.1. Static information

The 2-D appearance of the back of the lead vehicle is modeled. This is a static model representing the fixed geometric relationships among the visual features on this target. The model is composed of a collection of line segments. Each line segment corresponds to a physical feature that gives rise to a strong edge. Figure 2 shows the back of the lead vehicle. Figure 3 shows the ten line segments used to model this target.

This model is organized as two lists: a list of the line segments and a list of the corner points giving the intersections or end points of the line segments. Each line segment cross-references the two corner points that form its end points. The relative location of each corner point in the image is specified with respect to an origin at the center of the object when the object is viewed a nominal range from the camera.

2.1.2. Dynamic information

The location and the size of the target are measured from each new image. The location is represented by the x and y coordinates (c_x, c_y) of the object center with respect to the image coordinate frame. The size of the object is represented in terms of a scale factor. This scale factor is the ratio of the current size of the target in the image plane to the size of the target as viewed at the nominal range (see 2.1.1). It is assumed that the target's orientation is fixed with respect to the pursuit vehicle. Hence the location of the target in the image plane is represented by the 3-vector:

$$\text{location} = \begin{bmatrix} c_x \\ c_y \\ \text{scale} \end{bmatrix} \tag{1}$$

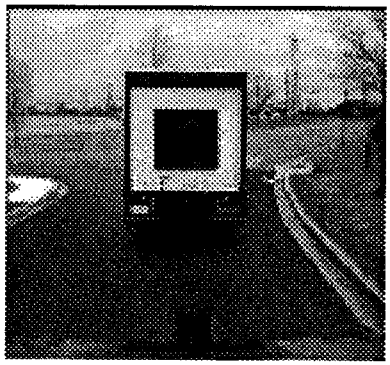

Figure 2. Target used for tracking

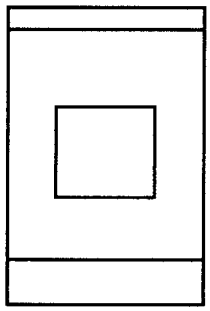

Figure 3. Line segments used to model target

The dynamic behavior of each of these parameters is represented by a second order motion model:

$$c_x[t] = a_{cx0} + a_{cx1}t + a_{cx2}t^2$$

$$c_y[t] = a_{cy0} + a_{cy1}t + a_{cy2}t^2 \quad (2)$$

$$scale[t] = a_{scale0} + a_{scale1}t + a_{scale2}t^2$$

t= time

The parameters of each motion model comprise the dynamic model. After each measurement of location (c_x, c_y, *scale*), the parameters of each motion model (e.g. a_{cx0}, a_{cx1}, a_{cx2} for c_x) are updated (see 2.7). The purpose of the dynamic model is to represent the motion of the target.

2.2. Initialization of models

The static model is derived off-line and entered by an operator into a data file that is read by the computer system at start-up.

It is assumed that initially the target is not moving. Therefore, the higher order parameters of each motion model are set to zero. The low order parameter of each motion model (a_{cx0}, a_{cy0}, and a_{scale0}), specifying the location of the target, are established by a teleoperator. The teleoperator does this by manually positioning and scaling the static model such that it aligns with the appearance of the target in the image. The static model is visually represented as a graphic overlay on the live video.

2.3. Edge extraction

In the first processing step, edge extraction is performed on the input scene (stage (1) in Figure 1). For every point in the image, edge magnitude and edge orientation are computed using a two-dimensional *3 X 3* spatial gradient operator. The direction, θ, of each point in the image is defined to be perpendicular to the direction of the gradient of the intensity function *f(x,y)* at that point:

$$\theta = \text{atan}\frac{\nabla f_x(x,y)}{\nabla f_y(x,y)} + \frac{\pi}{2} \quad (3)$$

The magnitude of each edge pixel, *mag*, is given by:

$$mag = \sqrt{(\nabla f_x(x,y))^2 + (\nabla f_y(x,y))^2} \quad (4)$$

Using a non-maximum suppression algorithm, those edge pixels whose magnitude is greatest in the direction across the edge are selected as edge points. A description of the non-maximum suppression edge extraction algorithm can be found in [3]. A binary edge image is produced by thresholding the edge points. The threshold level is set to a value which removes weaker edges. A threshold value of 8 in a grey level range of 0 to 255 proved to be effective. It should be noted, at this stage of operation, no effort is made to distinguish edges corresponding to model features from other edges present in the input image.

To reduce the amount of data processed by the data association algorithm (stage (2) in Figure 1), all edges that fall outside a window of interest are excluded. This window of interest is chosen to include the entire target but to exclude, as much as possible, the rest of the image (e.g. the hood of the vehicle, trees, grassy shoulders, houses, etc.). During execution, the lateral position of the window of interest shifts in order to keep it centered on the target. In addition to centering, the size of the window of interest changes as a function of the relative target size. Currently, seven sizes are used.

2.4. Data association

The raw edges in each image are produced by various visual entities including target features, shadows, telephone poles, pot holes, cracks, and other vehicles. A data association algorithm is used to determine which of these raw edges are likely to be associated with each target feature and to discard those edges that do not seem to be associated to any target feature (e.g. those caused by shadows, telephone poles, etc.).

The data association algorithm compares each edge pixel to the location of each predicted model feature. The predictions of model feature locations are generated using predictive filtering on the previous sequence of images (stages 6 and 7 in Figure 1, see 2.8). An edge pixel must satisfy two criteria to be associated to a model feature. The first criterion is two-dimensional spatial proximity of the edge point to the predicted model feature location. The second criterion is similarity of direction of the edge point with the angular orientation of the model feature.

To reduce the amount of computation time, a target was chosen that only contained vertical and horizontal lines. Horizontal lines are represented by their y intercept:

$$y = b_{yi} \tag{5}$$

Similarly, vertical lines are represented by their x intercept:

$$x = b_{xi} \tag{6}$$

The first step in the data association procedure determines if the edge direction of the candidate edge point, θ_i, is either nearly horizontal or vertical:

$$\left|\frac{\pi}{2} - \theta_i\right| < \delta \qquad \text{or} \qquad |0 - \theta_i| < \delta \tag{7}$$

If the edge point satisfies the horizontal criteria its distance is measured to all predicted horizontal line features for which:

$$x_{j1} < x_i < x_{j2} \tag{8}$$

where x_{j1} and x_{j2} are the x coordinates for the two endpoints of the j^{th} predicted horizontal line feature.
x_i is the x coordinate of the i^{th} edge point.

The distance metric is then simply:

$$d_{ij} = |b_{yj} - y_i| \tag{9}$$

where b_{yj} is the predicted y intercept for the j^{th} predicted horizontal line feature.
d_{ij} is the distance between the i^{th} edge point and the j^{th} predicted horizontal line feature.

Similarly, if the edge point satisfies the vertical criteria, its distance is measured to all predicted vertical line features for which:

$$y_{j1} < y_i < y_{j2} \tag{10}$$

where y_{j1} and y_{j2} are the y coordinates for the two endpoints of the j^{th} predicted vertical line feature.
y_i is the y coordinate of the i^{th} edge point.

$$d_{ij} = |b_{xj} - x_i| \tag{11}$$

where b_{xj} is the predicted x intercept for the j^{th} predicted vertical line feature.
d_{ij} is the distance between the i^{th} edge point and the j^{th} predicted vertical line feature.

If the minimum of these distances for this edge point is less than a distance threshold, ζ, the edge point is associated to that model feature for which it is minimum.

2.5. Feature location determination

The location of each model feature is determined (stage 3 in Figure 1) using the edge points matched to that feature.

If the feature is a horizontal line segment its y intercept is estimated. Similarly, if the feature is a vertical line segment, its x intercept is estimated.

Since the lines are represented by just a scalar intercept value, the least squares computation of this value is simply an average. If the feature is a vertical line, its x intercept, $\bar{b}_x$, is determined as the average of all x coordinates of edge points associated to that feature. Similarly, if the feature is a horizontal line, its y intercept, $\bar{b}_y$, is determined as the average of all y coordinates associated to that feature:

$$\bar{b}_{xj} = \frac{1}{N}\sum_{i=1}^{N_{xj}} x_{ij} \qquad \bar{b}_{yj} = \frac{1}{N}\sum_{i=1}^{N_{yj}} y_{ij} \tag{12}$$

Where x_{ij} is the x coordinate of the i^{th} edge point associated to the j^{th} vertical line feature
y_{ij} is the y coordinate of the i^{th} edge point associated to the j^{th} horizontal line feature.
N_{xj} is the number of points associated to the j^{th} vertical feature.
N_{yj} is the number of points associated to the j^{th} horizontal feature.

This computation is performed for all model features.

In addition, a confidence measure is computed for each of these estimates. A measure of confidence is needed because some estimates of feature location will be more accurate than others. Inaccurate estimates of location are due largely to incorrect data associations. For example, if a shadow of similar orientation appears near a predicted feature location it is quite possible for shadow edge points as well as true feature points to be associated to the model feature.

The confidence measure chosen is the square root of the inverse of the variance in the estimate:

$$w_{xj} = \frac{1}{\sigma_{xj}} \qquad \sigma^2_{xj} = \frac{1}{N}\sum_{i=1}^{N_{xj}} (x_{ij} - \bar{b}_{xj})^2 \qquad w_{yj} = \frac{1}{\sigma_{yj}} \qquad \sigma^2_{yj} = \frac{1}{N}\sum_{i=1}^{N_{yj}} (y_{ij} - \bar{b}_{yj})^2 \tag{13}$$

Where w_{xj} is the confidence associated with the j^{th} horizontal feature and w_{yj} is the confidence associated with the j^{th} vertical feature.

Confidence is defined to be *0* for features which do not have a minimal number of data associations. This minimum threshold is equal to *0.3 * predicted length of feature*. Therefore if a feature is predicted to be *50* pixels in length, it must match at least *15* edge points. Confidence also has an upper limit:

$$\text{if} \quad \sigma^2 < 0.1 \quad \text{then} \quad w = \frac{1}{\sqrt{0.1}} \tag{14}$$

This upper limit reflects the discrete sampling of the image. Differences in variance below this threshold are more likely attributable to feature location with respect to the discrete image grid rather than the physical distribution of data in the observed world. These thresholds and limits were determined on the basis of empirical evidence.

A higher variance (lower confidence) will typically indicate the presence of more than one visual entity (e.g. both the feature and a shadow). Conversely, a smaller variance (higher confidence) indicates a more reliable measurement.

2.6. Object location determination

The location of the object (c_x, c_y, *scale*) is determined (stage 4 in Figure 1) by a weighted least squares fit to the feature locations. The weight assigned to each feature is given by the feature's confidence measure (see 2.5). More specifically, the location of the object is determined by minimizing the least squares residual, J_l:

$$J_l = \sum_{i=1}^{M_x} w_{xi}^2 (c_x + \text{scale}^* u_i - \bar{b}_{xi})^2 + \sum_{i=1}^{M_y} w_{yi}^2 (c_y + \text{scale}^* v_i - \bar{b}_{yi})^2 \tag{15}$$

where M_x is the number of horizontal features
M_y is the number of vertical features
u and v are the distances, in x and y respectively, of each feature from the center of the object when it is viewed from a nominal range, specified *a priori* (see 2.1.1).
w_x and w_y are feature confidence measures (see 2.5).

In matrix form this becomes:

$$J_l = (\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x})^T \boldsymbol{W} (\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}) \quad \boldsymbol{b} = \begin{bmatrix} \bar{b}_{x1} \\ \bar{b}_{x2} \\ \cdots \\ \bar{b}_{y1} \\ \bar{b}_{y2} \\ \cdots \end{bmatrix} \quad \boldsymbol{A} = \begin{bmatrix} 1 & 0 & u_1 \\ 1 & 0 & u_2 \\ \cdots & \cdots & \cdots \\ 0 & 1 & v_1 \\ 0 & 1 & v_2 \\ \cdots & \cdots & \cdots \end{bmatrix} \quad \boldsymbol{W} = \begin{bmatrix} \frac{1}{\sigma_{x1}^2} & 0 & \cdots & 0 & 0 & \cdots \\ 0 & \frac{1}{\sigma_{x2}^2} & \cdots & 0 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{1}{\sigma_{y1}^2} & 0 & \cdots \\ 0 & 0 & \cdots & 0 & \frac{1}{\sigma_{y2}^2} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} c_x \\ c_y \\ scale \end{bmatrix} \tag{16}$$

This least squares residual is quadratic in the unknown parameter vector, **x**, and can therefore be solved uniquely and directly using any classical least squares method. Also, by this choice in the weighting matrix, *W*, the solution gives the best linear unbiased estimate, $\bar{\boldsymbol{x}}$, when the errors, the components of $(\boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}})$, are uncorrelated. For this implementation, the square root method was used to solve for the least squares estimate of the parameters. The square root method is described in 2.7.

Also as a measure of confidence in this estimate, the following measures are computed:

$$w_x = \sqrt{\frac{1}{J_l} \sum_{i=1}^{M_x} w_{xi}^2} \qquad w_y = \sqrt{\frac{1}{J_l} \sum_{i=1}^{M_y} w_{yi}^2} \qquad w_{scale} = \sqrt{\frac{1}{J_l} \left(\sum_{i=1}^{M_x} w_{xi}^2 + \sum_{i=1}^{M_y} w_{yi}^2 \right)} \tag{17}$$

J_l measures the consistency of the feature locations in terms of their known geometric relations. If the residual, J_l is small, the feature locations are geometrically consistent. A larger J_l indicates inconsistent feature locations. These confidence measures scale the inverse of J_l by the total confidence of all the features that affect the estimate of that parameter.

2.7. Estimation of motion models for each position parameter

Using the new estimate of target location, the parameters of each second order motion model (see 2.1) are recomputed (stage 5 in Figure 1) by a recursive weighted least squares computation with exponential decay. This is illustrated for the motion model associated with c_x below. The computation is identical for the motion model parameters corresponding to c_y and *scale*. For the motion model associated with c_x, the parameters $(a_{cx0}, a_{cx1}, a_{cx2})$ are determined by minimizing J_m after each measurement of c_x:

$$J_m = w_x^2[t]\,(c_x[t] - (a_{cx0} + a_{cx1}t + a_{cx2}t^2))^2 + \lambda w_x^2[t-1]\,(c_x[t-1] - (a_{cx0} + a_{cs1}(t-1) + a_{cs2}(t-1)^2))^2 +$$

$$\lambda^2 w_x^2[t-2]\,(c_x[t-2] - (a_{cx0} + a_{cx1}(t-2) + a_{cx2}(t-2)^2))^2 + \ldots. \tag{18}$$

Where λ: $0 < l < 1$

It should be emphasized that this least squares residual accounts for the entire trajectory history of c_x but gives increasingly less weight to older data in accordance with an exponential decay profile.

To solve for $(a_{cx0}, a_{cx1}, a_{cx2})$, the square root information filter algorithm[4] is used. This is a closed form solution to the least squares problem. It is also a recursive method; that is, the solution is updated as new data becomes available without having to explicitly store older data. The square root information filter is based on solving the least squares prob-

lem using a square root method. To illustrate the square root computational method, the problem is first expressed in batch form, i.e. as if all the data existed at once, rather than sequentially. Re-writing equation (18) in vector-matrix form gives:

$$J_m = (b_1 - A_1 x)^T W^T W (b_1 - A_1 x) \tag{19}$$

where:

$$b_1 = \begin{bmatrix} c_x[0] \\ c_x[-1] \\ \cdots \\ c_x[-t] \end{bmatrix} \quad A_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & (-1)^2 \\ \cdots & \cdots & \cdots \\ 1 & -t & (-t)^2 \end{bmatrix} \quad W = \begin{bmatrix} w_x[0] & 0 & \cdots & 0 \\ 0 & \lambda^{1/2} w_x[-1] & \cdots & 0 \\ \cdots & \cdots & \cdots & 0 \\ 0 & 0 & 0 & \lambda^{t/2} w[-t] \end{bmatrix} \quad x = \begin{bmatrix} a_{cx0} \\ a_{cx1} \\ a_{cx2} \end{bmatrix} \tag{20}$$

To simplify the time indexing, the current time is always set to $t = 0$. A and b are defined to absorb W:

$$J_m = (b - Ax)^T (b - Ax) \quad \text{where} \quad A = WA_1 \quad \text{and} \quad b = Wb_1 \tag{21}$$

As mentioned above, a square root method is used to solve for x, such that J_m is minimized. The basis of this method is that the Euclidean norm, J_m, of the vector $(b - Ax)$ will not change if the vector is multiplied by an orthogonal matrix T. Therefore, x can be equivalently determined by minimizing:

$$J_m = (Tb - TAx)^T (Tb - TAx) \tag{22}$$

An orthogonal matrix T is chosen such that the matrix product TA is equal to R, where R is partitioned into a $3x3$ upper triangular matrix, U, and a $(N-3)x3$ null matrix, 0, consisting of the remaining rows:

$$TA = R = \begin{bmatrix} U \\ 0 \end{bmatrix} \tag{23}$$

To find such a T, the QR factorization of A can be computed:

$$A = QR = Q \begin{bmatrix} U \\ 0 \end{bmatrix} \tag{24}$$

Where Q is orthogonal and R is the same as above.
Multiplying both sides of the equation by the transpose of Q gives:

$$Q^T A = R = \begin{bmatrix} U \\ 0 \end{bmatrix} \tag{25}$$

Comparing equations (23) and (25), T is given by the transpose of Q, computed from the QR factorization of A.

The QR factorization exists for any full rank matrix, A. A number of methods exist for computing this factorization including Gram-Schmitt, Modified Gram-Schmitt, Fast Givens Rotations, Householder Reflections, and the QR algorithm [4 5 6 7]. The Householder method was chosen for this application. It offers a combination of simplicity, numerical stability and computational efficiency.

Once T is found, the residual can be rewritten:

$$J_B = (Tb - TAx)^T (Tb - TAx) = \left(\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} - \begin{bmatrix} U \\ 0 \end{bmatrix} x \right)^T \left(\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} - \begin{bmatrix} U \\ 0 \end{bmatrix} x \right) = [z_1 - Ux]^T [z_1 - Ux] + z_2^T z_2 \tag{26}$$

Where the product Tb is partitioned into a vector z_1 consisting of the first 3 rows and the vector z_2 consisting of the remaining $N-3$ rows:

$$Tb = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \tag{27}$$

Since only the first term on the far right side in equation (26) is a function of x, only this term has to be minimized. This term is non-negative and x can be determined such that it is equal to zero, or equivalently:

$$z_1 = Ux \tag{28}$$

This is an exact set of linear equations (3 equations and 3 unknowns) that can be solved uniquely. Moreover, backsubstitution (see [7 8]) can be used to solve for x since U is an upper triangular matrix. This x then gives the solution to the batch least squares problem of minimizing equation (21).

When data is available in a sequential manner (as it is for this problem), rather than resolving a new batch problem after each observation, the problem can solved much more efficiently using the following recursive method. When a new measurement, $c_x[0]$, becomes available, it is combined with the least squares pair (U, z_1) in the following form:

$$y = \begin{bmatrix} c_x[0] \\ z_1\sqrt{\lambda} \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 0 & 0 \\ & \sqrt{\lambda}UV & \end{bmatrix} \tag{29}$$

This form is equivalent to a the batch form of equation (22) that incorporates the new data, $c_x[0]$. The top row of M corresponds to the current data i.e. $t=0$. The older data (U, z_1) has aged one cycle, and U is post-multiplied by V to achieve this time shift of -1:

$$\begin{bmatrix} 1 & t-1 & (t-1)^2 \\ 1 & t-2 & (t-2)^2 \\ \cdots & \cdots & \cdots \\ 1 & t-n-1 & (t-n-1)^2 \end{bmatrix} = \begin{bmatrix} 1 & t & t^2 \\ 1 & t-1 & (t-1)^2 \\ \cdots & \cdots & \cdots \\ 1 & t-n & (t-n)^2 \end{bmatrix} V \qquad V = \begin{bmatrix} 1 & -1 & 1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \tag{30}$$

Also, to achieve exponential weighting with time, the older data is multiplied by the square root of λ. The equivalence to equation (22), becomes more obvious if we remember that (U, z_1) was obtained by multiplying raw data by an orthogonal matrix, T:

$$y = \begin{bmatrix} c_x[0] \\ z_1\sqrt{\lambda} \end{bmatrix} = \begin{bmatrix} c_x[0] \\ \sqrt{\lambda}Tb \end{bmatrix} = T_1 \begin{bmatrix} c_x[0] \\ \sqrt{\lambda}b \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 0 & 0 \\ & \sqrt{\lambda}UV & \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ & \sqrt{\lambda}TAV & \end{bmatrix} = T_1 \begin{bmatrix} 1 & 0 & 0 \\ & \sqrt{\lambda}AV & \end{bmatrix} \tag{31}$$

Where $$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & & \\ \cdots & T & \end{bmatrix}$$

Since T_1 is orthogonal, from equation (22), we know that the least squares problem is unaltered if the data is multiplied by an orthogonal matrix.

The form of equation (29) allows recursive formulation of the problem where a new measurement can be efficiently combined with the older data given by the previous solution, (U, z_1). The attractiveness of this method is that the entire sequence of past data is expressed in a form, (U, z_1), of small fixed size. This is much more efficient than explicitly solving a batch problem using all the data in raw form, i.e. equation (20), after each new observation is acquired.

Using the square root method, x is then solved such the least squares residual, J_r, is minimized:.

$$J_r = (y - Mx)^T (y - Mx) = \| y - Mx \| \tag{32}$$

This will give the updated solution for x. The new values of z_1 and U that are generated by this process are then used to form another new pair y and M, equation (29), when a new data point $c_x[0]$ is acquired. This whole process is called the square root information filter. This name reflects the fact that the algorithm involves updating U which is proportional to a square-root of x's information matrix when the samples of x are statistically independent and identically distributed. The information matrix is defined as the inverse of the covariance matrix, Σ, of the state, x; that is, if x is thought of as a random vector, then:

$$\mathrm{E}\left((x - \bar{x})\, (x - \bar{x})^T \right) = \Sigma \propto (U^T U)^{-1} \tag{33}$$

2.8. Prediction of object location in next frame

In order to perform data association on the next image, the object's location in this next image must be predicted (stage 6 in Figure 1). Time indexing is maintained such that the current image is always time $t = 0$ (see equation (30) and discussion proceeding equation). Therefore to predict the x coordinate of the object's location, c_x, in the next image, $t = 1$, the motion model is used:

$$c_x[1] = a_{cx0} + a_{cx1}(1) + a_{cx2}(1)^2 \tag{34}$$

The y coordinate, c_y and scale factor, *scale*, are predicted similarly.

After the prediction of the object's location is established for the next image at $t = 1$, the predicted feature locations can be determined (stage 7 in Figure 1) in terms of their offset from the object center. For example the j^{th} horizontal feature is given by:

$$b_{xj} = c_x[1] + u_j \mathrm{scale}\,[1] \tag{35}$$

The predicted location of every feature in the model is computed this way.

2.9. Determination of target position with respect 3D coordinate frame affixed to chase vehicle

The position of the target with respect to a 3-D coordinate frame affixed to the vehicle is determined (stage 8 in Figure 1):

$$z(\mathrm{m}) = \frac{TargetSize\,\{\mathrm{m}\}}{TargetSizeImage\,\{\mathrm{pixels}\}\,\frac{\mathrm{mm}}{\mathrm{pixel}}} FocalLength\,\{\mathrm{mm}\} \tag{36}$$

$$x(\mathrm{m}) = \frac{(C_x\,\{\mathrm{pixels}\} - ImageCenterX\,\{\mathrm{pixels}\})\,\frac{\mathrm{mm}}{\mathrm{pixels}}\, z\,\{\mathrm{m}\}}{FocalLength\,\{\mathrm{mm}\}}$$

$$y(\mathrm{m}) = \frac{(C_y\,\{\mathrm{pixels}\} - ImageCenterY\,\{\mathrm{pixels}\})\,\frac{\mathrm{mm}}{\mathrm{pixels}}\, z\,\{\mathrm{m}\}}{FocalLength\,\{\mathrm{mm}\}}$$

These computations require knowledge of several intrinsic camera parameters: the focal length, the image center, and the pixel size on the CCD. Also, the actual physical size of the target must be known.

The 3-D position determined in this way is then used to steer the vehicle. The steering algorithm used is described in [2].

3. Hardware and developmental testbed

To run actual autonomous car following experiments requires a complete autonomous navigation system which includes a perception system, a steering/control system, and a robotic vehicle. Such a system has been developed at the National Institute of Standards and Technology (NIST) using a U.S. Army High Mobility Multipurpose Wheeled Vehicle (HMMWV). This paper has described the perception for autonomous car following, the HMMWV and steering/control system (stage 9 in Figure 1) are described in [1] and [2].

The development environment for the vision system consists of a Sun SPARCstation 2, a Pipelined Image Processing Engine (PIPE), a VME-based multiprocessor system. The system described in this paper is contained in a larger, multipurpose implementation of NASREM, [10 11 12 13]. Also, a complete control system architecture proposed for intelligent vehicles is found in [14].

A CCD video camera is mounted above the cab of the HMMWV. This gives a view of the road similar to that of a driver but from a higher perspective. The video input is read into PIPE. The incoming images are digitized to provide 8-bit grayscale images that are 242x256 pixels in size. Edge extraction is performed on the images. Only edges that fall within a *15* degree threshold about either the horizontal and vertical are retained. The Iconic-to-Symbolic Mapper (ISMAP) stage of PIPE [16] then performs a symbolic mapping operation converting information from an image format to two lists of pixel position, one for vertical edges and one for horizontal edges. These lists are mapped onto the memory of one of the microprocessors via a specialized PIPE-VME interface board. The edge extraction and symbolic mapping operations are pipelined.

The remaining processing is divided among 5 microprocessors in the VME backplane. These processes are pipelined. The throughput or update rate of the system is 15 hertz. The limiting factor is the computational time required for the PIPE processing. The worst case computational delay time from image acquisition to computing the steering is 128 ms. All inter-processor communication is done through semaphored global memory. For a detailed description of our software engineering practices refer to [15].

All program development for the VME-based multiprocessor system is done on a Sun SPARCstation 2. All code on this system is written in the Ada programming language. Program development for PIPE is done on a personal computer using the PIPE graphical programming language, ASPIPE [16].

4. System evaluation

This algorithm has been used to run the HMMWV under closed loop steering control (throttle was controlled manually) while it pursued a vehicle with the target shown in Figure 2. Testing was performed on roads on the grounds of the National Institute of Standards and Technology (NIST). The roads traversed contained moderate turns, several intersections, and moderate shadows. Top speeds of *35* km/h were achieved. Following distances varied from *5* to *15* meters.

The algorithm has also been tested using video taped scenes recorded from the camera mounted above the cab of the vehicle. This provided a method for performing tests without the vehicle, thereby allowing safe testing of the vision algorithm. Using a video-taped road scene, the algorithm was able to maintain tracking over a 17 minute video taped sequence that included a variety of speeds, turns, weaving, and both short and long following distances. The algorithm also performed well when there were deviations from fixed target orientation as assumed by the algorithm (see 2.1.2).

Although the algorithm performed quite well in the tests described, its limitations were also evident. In particular, it was not always reliable when there were strong shadows falling close to several model features simultaneously or when the background exhibited strong straight edge features (e.g. buildings, telephone poles) parallel and close to model features. Also a trade-off between tracking speed and robustness was evident. By increasing the exponential decay, see equation (18), abrupt motions could be tracked but the algorithm would be significantly less robust in the presence of shadows and various backgrounds.

5. Conclusion

A visual tracking algorithm that supports autonomous car following has been described. This algorithm has been used as part of a complete navigation system to autonomously drive a HMMWV such that it followed the path of a lead vehicle.

In future work, we will strive to further improve the robustness of the tracker in the presence of ambiguous back-

grounds and shadows while maintaining the ability to track abrupt target motion.

References

1 S. Szabo, H. Scott, K. Murphy, S. Legowik, R. Bostelman. "High-Level Mobility Controller for a Remotely Operated Land Vehicle." Journal of Intelligent and Robotic Systems. Vol. 5, pp 63-77. 1992.

2 K. Murphy. "Navigation and Retro-Traverse on a Remotely Operated Vehicle." Proceedings of the IEEE Conference on Intelligent Control and Instrumentation. Singapore, February, 1992.

3 A. Rosenfeld, A. Kak, *Digital Picture Processing*, Volume 2, Second Edition. Academic Press, 1982.

4 G. Bierman. *Factorization Methods for Discrete Sequential Estimation.* Academic Press. New York, 1977.

5 C. Lawson and R. Hanson. *Solving Least Squares Problems.* Prentice-Hall. Englewood Cliffs, New Jersey, 1974.

6 G. Golub and C. Van Loan. *Matrix Computations.* 2nd Ed. John Hopkins University Press. Baltimore, 1989.

7 G. Strang. *Linear Algebra and Its Applications.* 3rd Ed. Harcourt Brace Jovanovich. Orlando, Florida. 1988.

8 B. Noble and J. Daniel. *Applied Linear Algebra.* 3rd Ed. Prentice-Hall. Englewood Cliffs, New Jersey. 1988.

9 J. Albus, H. G. McCain, R. Lumia. "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)." NIST Technical Note 1235. Gaithersburg, MD, July, 1987.

10 K. Chaconas, M. Nashman. "Visual Perception Processing in a Hierarchical Control System." NIST Technical Note 1260. Gaithersburg, MD, March, 1989.

11 J. Fiala. "Manipulator Servo Level Task Decomposition." NIST Technical Note 1255. NIST, Gaithersburg, MD, October, 1988.

12 L. Kelmar. "Manipulator Servo Level World Modeling." NIST Technical Note 1258. NIST, Gaithersburg, MD, March, 1989.

13 A. Wavering. "Manipulator Primitive Level Task Decomposition." NIST Technical Note 1256. NIST, Gaithersburg, MD, October, 1988.

14 J. Albus, M. Juberts, S. Szabo. "A Reference Model Architecture for Intelligent Vehicle and Highway Systems." Isata 25th Silver Jubilee International Symposium on Automotive Technology and Automation. Florence, Italy. June, 1992.

15 J. Fiala. "Note on NASREM Implementation." NIST Internal Report 89-4215. Gaithersburg, MD, December, 1989.

16 Aspex Inc. "PIPE--An Introduction to the PIPE System." New York, 1987.