Title
Pseudoreal-time activity detection for railroad grade-crossing safety

Journal
IEEE Transactions on Intelligent Transportation Systems, 5(4)

ISSN
1524-9050

Authors
Kim, ZuWhan
Cohn, T E

Publication Date
2004-12-01

Peer reviewed

Technical Correspondences

Pseudoreal-Time Activity Detection for Railroad Grade-Crossing Safety

ZuWhan Kim and Theodore E. Cohn

***Abstract*—It is important to understand the factors underlying grade-crossing crashes and to examine potential solutions. We have installed a camera in front of a locomotive to examine grade-crossing accidents (or near accidents). We present a computer vision system that automatically extracts possible near-accident scenes by detecting the activity of vehicles crossing in front of the train after signals are ignited. We present a fast algorithm to detect moving objects recorded by a moving camera with minimal computation. The moving object is detected by: 1) estimating the ego motion of the camera and 2) detecting and tracking feature points whose motion is inconsistent with the camera motion. We introduce a pseudo-real-time ego-motion (camera-motion) estimation method with a robust optimization algorithm. We present experiments on ego-motion estimation and moving-object detection. Our algorithm works in pseudoreal-time and we expect that our algorithm can be applied to real-time applications such as collision warning in the near future, with the development of hardware technology.**

***Index Terms*—Computer vision, ego-motion estimation, motion detection, railroad grade-crossing safety.**

I. Introduction

According to the U.S. Department of Transportation, in 2000, a total of 2895 accidents occurred at railroad crossings, causing 306 deaths [1]. The property damage in such events is large, as is personal injury. Therefore, it is important to understand the factors underlying grade-crossing crashes and to examine potential solutions. While there are databases that record collisions and the prevailing conditions, little systematic attention has been paid to the activity that is antecedent to a collision, persons or vehicles crossing the tracks in front of an oncoming train.

Video data is an important source of examining the grade-crossing accidents (or near accidents). We have worked with the California Department of Transportation to install three video cameras (one facing forward and the others facing the sides) in front of a locomotive that is operated along the *San Joaquin Rail Corridor* (about 280 mi from Bakersfield, CA, to Emeryville, CA). There are about 700 crossings on this route. The video data is recorded in real time in an MPEG format (320×240 resolution, 30 frames/s) in the *data-collection device* installed on the locomotive. The-data collection machine also is equipped with a global positioning system (GPS). GPS information is collected every second. When the train arrives at the Oakland, CA, AMTRAK facility, the recorded video data (about 4–8 h) and GPS information are transferred to the *server computer* via wireless communication.

Manuscript received December 1, 2003; revised July 15, 2004 and July 21, 2004. This work was supported by CalTrans/PATH under RTA 65A0089. The Associate Editor for this paper was F.-Y. Wang.

Z. Kim is with the California PATH/Computer Science Division, University of California, Berkeley, Richmond, CA 94804 USA (e-mail: zuwhan@berkeley.edu).

T. E. Cohn is with the School of Optometry and the Department of Bioengineering, University of California, Berkeley, Richmond, CA 94804 USA (e-mail: tecohn@spectacle.berkeley.edu).

Digital Object Identifier 10.1109/TITS.2004.838507

Fig. 1. Example video frame. The vehicle inside of the rectangle is moving to the left.

While more than 6 h of video data are obtained daily, the number of useful video scenes is small. First, the average number of crossing accidents/incidents is only about five per million train miles. The number of activities that we are interested in may be larger, but the ratio of the useful information is still too small for manual examination. Next, much of the gathered data contain images obtained between images. One easy way of removing useless scenes is to use the GPS coordinates. We have a database that contains the GPS coordinates of most of the crossings; thus, we can retain only those scenes near the crossings. About 75% of the data can be removed in this way, but the volume of the data remains too large to manually examine. In this paper, we present research on the automated detection of the activity near the crossings.

An example of a video frame is shown in Fig. 1. Our purpose is to develop a moving-object detection algorithm in which the camera is also in motion (mostly moving forward). The algorithm needs to be fast enough to process a large volume of data daily (does not need to be in real time) and robust enough to detect most of the (near) accident events with a small number of false alarms. There has been no fast enough (pseudoreal-time) algorithm for detecting moving objects from a moving camera because of the following difficulties.

- It is hard to estimate the camera motion (ego motion) robustly, except to use an accurate optical flow that requires a large amount of computation.
- The three-dimensional (3-D) structure of the background is also unknown. To robustly estimate both camera motion and the background structure, a large amount of computation involving a large number of frames is required. (Otherwise, it will result in noisy camera parameter estimation for some of the frames.)

We present a pseudoreal-time algorithm that addresses these difficulties. Our algorithm uses a relatively small number (100–200) of corner features and their matches. The ego motion is estimated from the corner matches from a small number of frames (two, in our example) and the corner points that move inconsistently to the camera motion are detected. Both the corner matching and the camera-motion estimation (from a small number of frames) may introduce noise. To handle such noise, we use a *robust technique* to estimate the camera motion and use temporal information to find only corners that move in a *consistently inconsistent* manner to the camera motion. This removes the corners detected by false matches or wrong camera-motion estimation.

In Section II, an introduction on the ego-motion estimation is presented. We present our algorithm in Section III and show experiments in Section IV. Finally, the conclusion and the future work are given in Section V.

II. Background

The goal of the ego-motion estimation is to recover the motion of the camera by using image measurements of fixed points in the scene. When the 3-D coordinates (or constraints among them, such as planarity) of the feature points are known (for example, [2] or camera calibration problems), another coordinate system (world coordinate) is often introduced and the rotations and the translations of each and every frames are estimated according to the world coordinate. However, in this application, we do not have any information on the 3-D coordinates of the feature points. Therefore, we only estimate the relative configuration (rotation and translation) of the camera with respect to the previous one. For each frame, we recover the rotation and the translation of the camera with respect to the previous frame.

When the number of frames is small enough, it is tractable to estimate camera configurations of all the frames as well as the 3-D coordinates of the feature points at the same time, [3], [4]. However, such a calculation deals with a matrix of substantial dimensions, including all the camera parameters and the image and world coordinates of the feature points. Therefore, it is not efficient to process hours of video data with this approach. Hence, we limit the scope of this paper to the two-frame case described earlier.

In this section, we introduce a brief survey on the ego-motion estimation from two image frames. Most of the contents in this section are also found in the photogrammetry literature (e.g., [5]).

A. Projective Geometry

In the basic pinhole camera model, a 3-D point $\mathbf{X} = (X\ Y\ Z)^\top$ on *camera coordinates* is mapped to the point on the image plane (*image coordinates*) $\mathbf{u} = (u\ v)^\top$, where $u = fX/Z, v = fY/Z$ and f is the focal length. In a matrix form

$$\begin{pmatrix} wu \\ wv \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}.$$

For simplicity, we redefine $\mathbf{u} \equiv (wu\ wv\ w)^\top$. For a finite projective camera, the internal camera parameters

$$\mathbf{K} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

where α_u and α_v represent the focal length of the camera in terms of pixel dimensions in the u and v direction, respectively, and $(u_0\ v_0)^\top$ is the coordinate of the image center.

Usually, a separate coordinate system, *world coordinates*, which is different from the camera coordinates, is introduced. In this case, the Euclidean transformation (rotation $\mathbf{R}$ and translation $\mathbf{T}$) between the world and camera coordinates is introduced as $\mathbf{u} = \mathbf{K}(\mathbf{R}\mathbf{X} + \mathbf{T})$.

Using this equation, $\mathbf{K}, \mathbf{R}$, and $\mathbf{T}$ can be estimated when we know the correspondences between points in the world and image coordinates (see [5] for details).

B. Two-View Camera Geometry

Consider two cameras (or two camera configurations) $\mathbf{C}$ and $\mathbf{C}'$. The *baseline* is defined as the line segment joining two camera centers. For each camera, the *epipole* is defined as the point of intersection of the baseline and the image plane. Consider a 3-D point $\mathbf{X}$ and its projection $\mathbf{u}$ on $\mathbf{C}$. $\mathbf{X}, \mathbf{u}$, and the baseline lie on the same plane, which is the *epipolar plane*. This is defined as the intersection of the epipolar plane with the image plane. For any image point $\mathbf{u}$ of $\mathbf{C}$, its correspondence of $\mathbf{C}', \mathbf{u}'$ also lies on the same epipolar plane (and the corresponding epipolar line) regardless of the 3-D position of $\mathbf{X}$. Furthermore, for any corresponding points $\mathbf{u}$ and $\mathbf{u}'$

$$\mathbf{u}'^\top \mathbf{F} \mathbf{u} = 0 \tag{1}$$

where $\mathbf{F}$ is the *fundamental matrix*. With the fundamental matrix, we can get epipolar lines. For any point $\mathbf{u}$, the corresponding epipolar line is

$$\mathbf{l}' = \mathbf{F}\mathbf{u}. \tag{2}$$

In our application, $\mathbf{C}$ and $\mathbf{C}'$ are the same camera with different position and orientation. Regard $\mathbf{C}$ is the camera configuration of the previous frame and $\mathbf{C}'$ of the current frame. For simplicity, we let the camera coordinate of $\mathbf{C}$ be the world coordinate. Then, the configuration of $\mathbf{C}'$ is represented by a rotation $\mathbf{R}$ and a translation $\mathbf{T} = (T_X\ T_Y\ T_Z)^\top$.

We introduce a new coordinate system called *normalized coordinates*. A normalized coordinate is obtained from the corresponding image coordinate given the internal camera parameters $\mathbf{K}$: $\mathbf{x} = (x\ y\ 1)^\top = \mathbf{K}^{-1}(u\ v\ 1)^\top$.

For $\mathbf{C}$, a point $\mathbf{X} = (X\ Y\ Z)^\top$ on the world coordinate can be easily transformed to a point $\mathbf{x} = (x\ y\ 1)^\top$ on the normalized coordinate

$$x = X/Z, \text{ and } y = Y/Z. \tag{3}$$

For $\mathbf{C}'$, the transformed point

$$\mathbf{x}' = (x'\ y'\ 1)^\top = (X'/Z'\ Y'/Z'1)^\top \tag{4}$$

where $\mathbf{X}' = (X'\ Y'\ Z')^\top = \mathbf{R}\mathbf{X} + \mathbf{T}$.

As the counterpart of (1), for any two corresponding points $\mathbf{x}$ and $\mathbf{x}'$

$$\mathbf{x}'^\top \mathbf{E} \mathbf{x} = 0 \tag{5}$$

where $\mathbf{E}$ is the *essential matrix*. In fact,

$$\mathbf{E} = [\mathbf{T}]_\times \mathbf{R} \tag{6}$$

where $[\mathbf{T}]_\times$ is the *skew-symmetric matrix* of $\mathbf{T}$ (see [5] for details).

C. Ego-Motion Estimation

Assume that the internal parameters $\mathbf{K}$ are known (they can be obtained from a separate calibration procedure). The ego-motion estimation is to recover $\mathbf{R}$ and $\mathbf{T}$ from a set of point correspondences, where their relationship is nonlinear.

One approach is to first estimate the essential matrix $\mathbf{E}$ and to calculate $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$ (the estimates of $\mathbf{R}$ and $\mathbf{T}$) from it. From a point correspondence, we get two linear equations (with respect to the parameters of $\mathbf{E}$) by applying it to (5). Given a number of point correspondences, we can easily get $\hat{\mathbf{E}}$ by, for example, using the least-square estimation. Once $\hat{\mathbf{E}}$ is obtained, we get $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$ by applying the singular value decomposition (see [5] for details).

However, this method does not provide accurate estimation [6], because the parameters of $\mathbf{E}$ are not directly related to the physical properties and minimizing their errors assuming, for example, Gaussian error

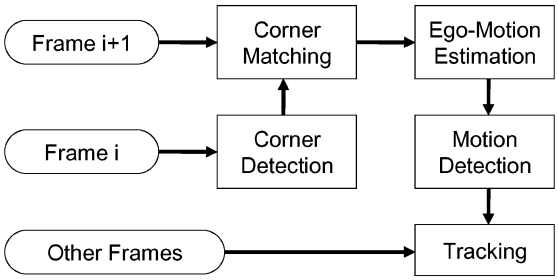

Fig. 2. Flowchart of the moving-object detection algorithm.

distributions (the least-square estimation) is not meaningful. We want to minimize meaningful errors; for example, the reprojection error

$$\mathrm{Err}_{\mathrm{rprj}}(\mathbf{x}, \mathbf{x}', \hat{\mathbf{X}}) = \sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \tag{7}$$

where $(\mathbf{x}_i, \mathbf{x}'_i)$ is the measured position of the ith correspondence ($\mathbf{x} = \mathbf{x}_{1,\ldots,n}$ and $\mathbf{x}' = \mathbf{x}_{1,\ldots,n}$, where n is the number of points), $\hat{\mathbf{x}}$ is the estimate of the "true" 3-D positions of the points, $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}'_i$ are the projection of $\hat{\mathbf{x}}_i$ on $\mathbf{C}$ and $\mathbf{C}'$ and $d()$ is the Euclidean distance. We can estimate $\mathbf{R}$ and $\mathbf{T}$ by solving nonlinear equations of (3), (4), and (7) on the parameters of $\hat{\mathbf{R}}$, $\hat{\mathbf{T}}$, and $\hat{\mathbf{x}}_i$, respectively. We can use any iterative method and the result of the first approach (which uses the least-square estimate of the essential matrix) is usually used as a initial guess. The initial values of $\hat{\mathbf{x}}_i$ can be obtained from the initial values of $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$. From (4), we get

$$\begin{pmatrix} x'_i \hat{Z}'_i \\ y'_i \hat{Z}'_i \\ \hat{Z}'_i \end{pmatrix} = \hat{\mathbf{R}} \begin{pmatrix} x_i \hat{Z}_i \\ y_i \hat{Z}_i \\ \hat{Z}_i \end{pmatrix} + \hat{\mathbf{T}}.$$

We get two linear equations on $\hat{Z}'_i$ from this equation and can estimate $\mathbf{x}_i$ from it.

Note that $\mathbf{R}$ is a 3×3 matrix, but has only three degrees of freedom (DoF). Therefore, we have six more constraints to make $\mathbf{R}$ a valid rotation matrix. Reference [6] uses a unit quaternion to represent the rotation to effectively enforce these constraints.

III. Moving-Object Detection

The flowchart of the moving-object detection algorithm is shown in Fig. 2. For each frame, we apply an eigen-value-based corner detector [7]. When a corner is detected, a small 9×9 template of the gray-level image is extracted. Then, we search for the matches of the extracted templates in the next frame. The match score is based on the correlation and the search is performed on a 9×9 search window.

Fig. 3 shows detected corner features and their matches (shown as optical flows). Note that there are many line features. The matching process can suffer from the *aperture problem* and we may have some false matches. Now, our goal is to find the corners of which the movements are not consistent with others. This is done by estimating the ego motion of the camera and checking the consistency of the movement. Finally, we track detected moving corner features for several more frames to remove the noise (falsely detected motion).

An effort on a similar problem is found in [8]. It introduces the *multibody fundamental matrix* (a set of fundamental matrices) to segment motions into several groups. In fact, many of the moving objects (corner features) can be detected without fully recovering the ego motion, but only by recovering the fundamental matrix (or the essential matrix). When we know the fundamental matrix, we can calculate the corresponding epipolar line of a point (2) and can detect a moving object as long as its motion does not accidentally lie on the corresponding epipolar line. However, this "accidental alignment" of the motion often occurs in our application. An example is shown in Fig. 4. Most of the object motions in which we are interested occur in a horizontal direction in the middle of the image, while the epipolar lines in the middle of the images also are mostly horizontal.

Fig. 3. Detected corner features (square boxes) and their optical flows. Note that the line features can cause false matches (aperture problem). The optical flows were exaggerated 3 times for visibility.

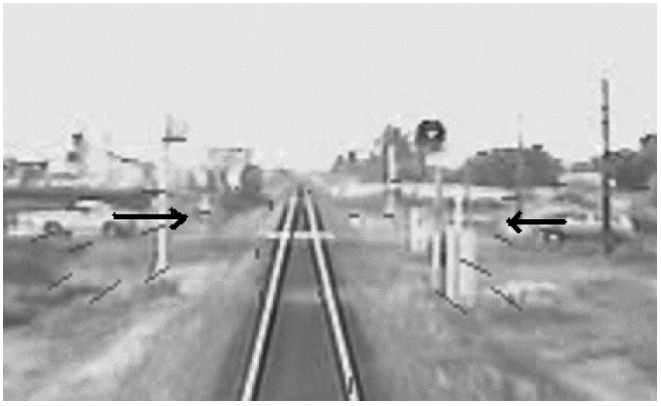

Fig. 4. We need to recover the ego motion fully because the "accidental alignment" of the object motion and the corresponding epipolar line occurs frequently.

In fact, this is an intrinsic ambiguity that cannot be removed unless we know the 3-D coordinate (or depth) of the point. However, this ambiguity can be addressed if we constrain the 3-D coordinates of the objects. For example, no visible object can have a negative Z-coordinate, nor (usually) stays in front of a moving train closer than a certain distance (say, 10 m). To apply these constraints, we need to recover the 3-D coordinates of the points. Therefore, we recover the full camera parameters $\mathbf{R}$ and $\mathbf{T}$.

A. Ego-Motion Estimation for Railroad-Crossing Imagery

In our application, the ego motion is mostly a forward translation (T_Z) with a very small amount of other translation (T_X and T_Y) and rotation. However, the ego motion along the Z axis is difficult to recover robustly [9]. For example, the pitch (rotation about the Y axis) and T_X causes similar optical flows (*bas-relief ambiguity*). Therefore, standard ego-motion estimation methods (Section II-C) do not give good estimation; they result in many false alarms in motion detection. In this section, we introduce an augmentation for the present application.

1) Error Model: The reprojection error (7) was introduced based on the assumption of a Gaussian error distribution of the reprojected

points. This assumption is valid when the scene is static and all the matches (correspondences) are correct. However, in our application, we also need to consider the errors caused by false matches and by object motion. If we consider these outliers, the error distribution is no longer Gaussian, but close to a binomial or multinomial distribution. This error can be handled in the optimization procedure by applying *robust estimation* techniques, such as least-trimmed square. To simulate robust estimation techniques efficiently, we change the objective function such that it minimizes the number of matches that do not fit to the current camera model

$$\mathrm{Err}_{\mathrm{sigmoid}} = \sum_i e(\mathbf{x}_i, \mathbf{x}_i', \hat{\mathbf{x}}_i) \tag{8}$$

where

$$e(\mathbf{x}_i, \mathbf{x}_i', \hat{\mathbf{x}}_i) = \begin{cases} 1, \text{ if } \hat{Z} < 0 \text{ or} \hat{\mathbf{x}} \text{ is right in front of the camera or} \\ \mathrm{sigmoid}\left(d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}_i', \hat{\mathbf{x}}_i')^2; \mu\right), \text{ otherwise} \end{cases}$$

and μ is the mean of the sigmoid function. We allow maximum 1.0 reprojection error. Therefore, $\mu = 1.0$.

In many cases, optimizing only with $\mathrm{Err}_{\mathrm{sigmoid}}$ does not give good result because numerically $\mathrm{sigmoid}(x) = \mathrm{sigmoid}(y) = 1.0$ for any large numbers of x and y. For many point correspondences, the reprojection errors of the initial estimate are large enough to cause this numerical error. Therefore, we propose a two-step optimization. For the first, say, 15, iterations, we apply the reprojection error, then apply $\mathrm{Err}_{\mathrm{sigmoid}}$ for the rest of the iterations.

2) Optimization: We use an iterative optimization method (e.g., the gradient-descent method) to minimize the objective function. Usually, the initial estimates of the camera parameters are obtained from the least-square estimate of the essential matrix $\hat{\mathbf{E}}$ (see Section II-C). However, in our case, involving forward ego motion and outliers, the estimates of the rotation parameters $\hat{\mathbf{R}}$ are highly unstable when obtained in that way. Therefore, we assume pure translation at the initial estimation when there is no rotation $\mathbf{E} = [\mathbf{T}]_\times$ from (6). In this instance, we can use the initial estimates of $\hat{T}_X = (\hat{E}_{3,2} - \hat{E}_{2,3})/2, \hat{T}_Y = (\hat{E}_{1,3} - \hat{E}_{3,1})/2$, and $\hat{T}_Z = (\hat{E}_{2,1} - \hat{E}_{1,2})/2$.

The next step is to apply the iterative optimization on $\hat{\mathbf{R}}, \hat{\mathbf{T}}$, and $\hat{\mathbf{x}}_i$. We apply a gradient-based optimization algorithm. However, the optimization process can be extremely time consuming, because the number of parameters to optimize is $3n + 6$, where n is the number of the point matches. Also, the initial estimation error of $\hat{\mathbf{x}}_i$ can be very large because its estimation is very sensitive to $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$. For example, if we do not have a correct estimate on the *point of expansion*, we will get many $\hat{Z}_i$s with negative values.

Therefore, we optimize $\hat{\mathbf{x}}_i$ and the camera parameters ($\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$) separately. For each iteration step, we can first optimize (update) $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$ given the point matches ($\mathbf{x}_i$ and $\mathbf{x}_i'$) and $\hat{\mathbf{x}}_i$s, then optimize $\hat{\mathbf{x}}_i$s given $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$ (triangulation). In fact, the parameters of $\hat{\mathbf{x}}_i$ ($\hat{X}_i, \hat{Y}_i$, and $\hat{Z}_i$) are highly correlated among each other since $\hat{X}_i = \hat{x}_i \hat{Z}_i$, and $\hat{Y}_i = \hat{y}_i \hat{Z}_i$ (note that we need to estimate $\hat{X}_i, \hat{Y}_i$, and $\hat{Z}_i$ separately, mainly because x_i and y_i are not the real values, but just observations that contain sampling errors). However, when we assume that $\hat{Z}_i$ is the only independent variable and $\hat{X}_i = x_i \hat{Z}_i$ and $\hat{Y}_i = y_i \hat{Z}_i$ (i.e., when we assume $\mathbf{x}_i$ is correct), we get a closed-form solution of $\hat{Z}_i$ optimizing the square sum of the distance error, given $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$.

$$\begin{aligned} &\arg_{\hat{Z}_i} \min\{(\hat{x}_i - x_i')^2 + (\hat{y}_i - y_i')^2\} \\ &= \frac{(r_{x_i}\hat{T}_Z - \hat{T}_X)(x_i'\hat{T}_Z - \hat{T}_X) + (r_{y_i}\hat{T}_Z - \hat{T}_Y)(y_i'\hat{T}_Z - \hat{T}_Y)}{(r_{x_i}\hat{T}_Z - \hat{T}_X)(r_{x_i} - x_i') + (r_{y_i}\hat{T}_Z - \hat{T}_Y)(r_{y_i} - y_i')} \end{aligned} \tag{9}$$

where $r_{x_i} = \hat{R}_{1,1}x_i + \hat{R}_{1,2}y_i + \hat{R}_{1,3}$, and $r_{y_i} = \hat{R}_{2,1}x_i + \hat{R}_{2,2}y_i + \hat{R}_{2,3}$ (see the Appendix for the proof). In fact, the objective function that we minimize for $\hat{\mathbf{x}}$ (9) is different from the objective function (8). However, this approximation is good enough since our goal is not to accurately estimate $\mathbf{x}$, but just to find inconsistent motion. In [10], Hartley and Sturm introduced an optimal triangulation method minimizing the reprojection error. However, we did not apply this method because the optimization process is complex and time consuming.

In this application, we separate the iterative update procedure of $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$ because the convergence characteristics for $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$ are very different from each other. Assume that $\mathrm{Err}()$ is the objective function to be minimized. For each iteration, we apply the following optimization steps.

Step 1) Estimate $(\partial \mathrm{Err})/(\partial \hat{\mathbf{R}}), \hat{\mathbf{R}} \leftarrow \hat{\mathbf{R}} - \alpha(\partial \mathrm{Err})/(\partial \hat{\mathbf{R}})$.
Step 2) Optimize $\hat{\mathbf{x}}_i$'s given $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$.
Step 3) Eestimate $(\partial \mathrm{Err})/(\partial \hat{\mathbf{T}}), \hat{\mathbf{T}} \leftarrow \hat{\mathbf{T}} - \beta(\partial \mathrm{Err})/(\partial \hat{\mathbf{T}})$.
Step 4) Optimize $\hat{\mathbf{x}}_i$'s given $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$.

α and β controls the convergence speeds as in the Levenberg–Marquardt iteration. Step 2) is necessary because, when we update $\hat{\mathbf{T}}$ in Step 3), we use the newly updated $\hat{\mathbf{R}}$ of Step 1). However, when we skip Step 2), $\hat{\mathbf{R}}$ and $\hat{\mathbf{x}}_i$ will be inconsistent in Step 3). See Section IV for the comparison between with and without separating the update of $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$.

B. Motion Detection

Once the ego motion is estimated, we detect object motion by finding inconsistent corner motions. We use $\hat{\mathbf{x}}_i$ (obtained from the ego-motion estimation) to detect inconsistent motion. The ith point match is inconsistent with the estimated ego motion when $e(\mathbf{x}_i, \mathbf{x}_i', \hat{\mathbf{x}}_i) > \mathrm{sigmoid}(1.0; \mu)$ [cf., (8)].

The inconsistency of a point motion can come from three possible reasons: 1) object motion; 2) matching error; or 3) ego-motion estimation error. Note that the inconsistency caused by the object motion is distinguished from the others because it is consistent over the frames. Therefore, we applied a simple tracking algorithm to check this "consistent inconsistency." Once a corner is determined to be inconsistent, its matches are found for the next seven or more frames and we report it as a moving object when its motion is consistently inconsistent to the ego motion.

IV. Experiments

Currently, only a small volume of video data is available for a preliminary test. We focus our experiment on the ego-motion estimation.

We first compare the proposed optimization method and the reprojection error method. Fig. 5 shows the convergence properties of two example pairs of frames. When there was a small number of false matches [for example, no visible false match was found in the frame pair for Fig. 5(a)], the difference between two methods was small. However, when there were false matches [for example, at least three visible false matches were found in the frame pair of Fig. 5(b)], the proposed method showed much better performance (6.58 versus 11.98).

The comparison is based on the objective function of (8). However, even when we compared the reprojection error in (7), the proposed methods showed superior performance when there were false matches [for example, 5.90 versus 21.27 for Fig. 5(b), both resulted in two outliers]. At the same time, the differences were not significant when there was no false match [for example, 7.34 versus 7.31 for Fig. 5(a)]. In fact, the proposed method showed even better performances on many examples with no false matches. From this result, we find that applying a robust estimation method reduces the risk of falling into a local minimum caused by false matches.

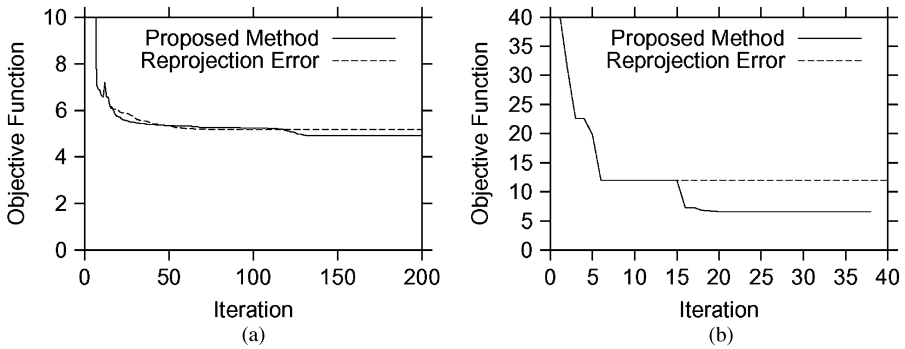

Fig. 5. Comparison between the proposed optimization method and the reprojection error method, when (a) there is no visible false match and (b) several false matches were found.

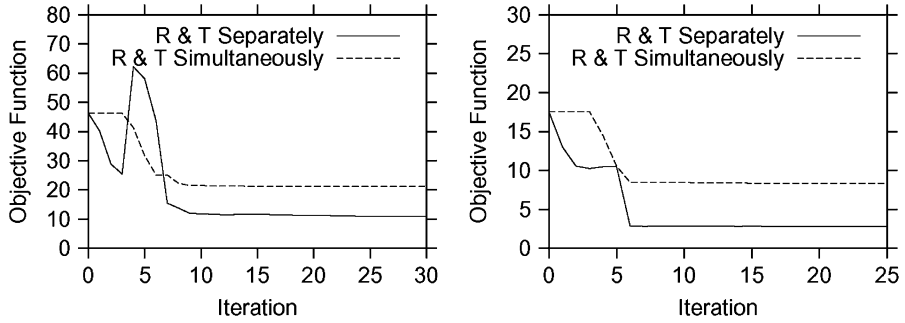

Fig. 6. We get better estimation when we update $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$ separately (see Section III-A).

Fig. 7. Example video sequence (first frame) and train trajectory reconstructed from the suggested ego-motion estimation.

Fig. 8. RANSAC with a seven-point algorithm does not work well. RANSAC's scoring fuction says that 52 out of 68 correspondences are consistent with the resulting essential matrix, but the reprojection error tells that only 29 are consistent (white boxes).

We also performed an experiment to justify the separate update of $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$. The comparison result is shown in Fig. 6. We find that the simultaneous update of $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$ caused poor performance because of the different convergence characteristics of $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$.

A motion-reconstruction example is shown in Fig. 7. We observe that the train trajectory is nicely reconstructed. We also tested random sample consensus (RANSAC), a popular pseudoreal-time (or real time with a pre-emptive scoring function, [11]) motion-estimation algorithm for the same video. We applied a seven-point algorithm in which hundreds of essential matrix hypotheses are generated from random sets of seven points and evaluated by the epipolar constraint. The essential matrix with the best score is chosen, which determines the rotation and translation. However, scoring essential matrix does not give an accurate result. An example is shown in Fig. 8. While RANSAC's scoring function says that 52 out of 68 correspondences are consistent with the resulting essential matrix, the reprojection error tells that only 29 of them are consistent.

Finally, the moving-object detection result is shown in Fig. 9. Five corners of the moving object were detected without any false alarm. The amount of computation depended on the number of the corners processed and the convergence characteristics on the motion. The experiment was performed on Pentium III (1-GHz) personal computer

Fig. 9. Detected moving object with the trajectories of its corner features.

and, for most cases, approximately 50–150 ms were spent for the corner detection and matching and 20–500 ms (but mostly under 150 ms) were spent for the optimization.

V. Conclusion and Future Work

We presented a moving-object detection algorithm for railroad-crossing safety. We proposed a fast triangulation method and a robust optimization algorithm to meet our application's need. We presented preliminary experiments that showed that the proposed approach is promising. A more thorough evaluation will be performed when a large volume of video data is collected. We will generate a learning data set by showing video clips to human and compare the detection result with our algorithm.

Our algorithm works at pseudoreal time (about three to four frames/s on a Pentium III personal computer). We expect that, in the near future (with the development of computer hardware), it can be applied to real-time applications such as an in-vehicle collision warning system.

Appendix
Proof of (9)

We define the object function

$$\mathrm{Err}(x', y', x, y, \hat{Z}) = \{(\hat{x'}(x, y, \hat{Z}) - x')^2 + (\hat{y'}(x, y, \hat{Z}) - y')^2\}$$

to be minimized with regard to $\hat{Z}$. From (3) and (4)

$$\hat{x'}(x, y, \hat{Z}) = \frac{\hat{X}'(x, y, \hat{Z})}{\hat{Z}'(x, y, \hat{Z})} = \frac{\hat{Z} r_x + \hat{T}_X}{\hat{Z} r_z + \hat{T}_Z}$$

and

$$\hat{y'}(x, y, \hat{Z}) = \frac{\hat{Y}'(x, y, \hat{Z})}{\hat{Z}'(x, y, \hat{Z})} = \frac{\hat{Z} r_y + \hat{T}_Y}{\hat{Z} r_z + \hat{T}_Z}$$

where $r_x = \hat{R}_{1,1}x + \hat{R}_{1,2}y + \hat{R}_{1,3}$, $r_y = \hat{R}_{2,1}x + \hat{R}_{2,2}y + \hat{R}_{2,3}$, and $r_z = \hat{R}_{3,1}x + \hat{R}_{3,2}y + \hat{R}_{3,3}$.

We get $\hat{Z}$ that minimizes $\mathrm{Err}()$ by taking its partial derivative with regard to $\hat{Z}$

$$\frac{\partial \mathrm{Err}}{\partial \hat{Z}} = \frac{-2}{(\hat{Z} r_z + \hat{T}_Z)^3}[(r_x \hat{T}_Z - \hat{T}_X)\{(x' - r_x)\hat{Z} + (x'\hat{T}_Z - \hat{T}_X)\} + (r_y \hat{T}_Z - \hat{T}_Y)\{(y' - r_y)\hat{Z} + (y'\hat{T}_Z - \hat{T}_Y)\}] = 0.$$

$$\hat{Z} = \frac{(r_x \hat{T}_Z - \hat{T}_X)(x'\hat{T}_Z - \hat{T}_X) + (r_y \hat{T}_Z - \hat{T}_Y)(y'\hat{T}_Z - \hat{T}_Y)}{(r_x \hat{T}_Z - \hat{T}_X)(r_x - x') + (r_y \hat{T}_Z - \hat{T}_Y)(r_y - y')}.$$

This is the only extremum of $\mathrm{Err}()$ with regard to $\hat{Z}$, which is the global minimum.

Acknowledgment

The authors would like to thank R. Vidal for his helpful comments on the optimization methods.

References

[1] Railroad safety statistics annual report. Office Safety Anal., Fed. Railroad Admin., U.S. Dept. Transport. [Online]. Available: http://safetydata.fra.dot.gov/officeofsafety/

[2] O. Shakernia, Y. Ma, T. J. Koo, and S. Sastry, "Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control," *Asian J. Control*, vol. 1, no. 3, pp. 128–145, 1999.

[3] J. Oliensis, "A multi-frame structure-from-motion algorithm under perspective projection," *Int. J. Comp. Vision*, vol. 34, no. 2, pp. 163–192, 1999.

[4] M. Han and T. Kanade, "Creating 3-D models with uncalibrated cameras," in *Proc. 5th IEEE Workshop Applications of Computer Vision*, 2000, pp. 178–185.

[5] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[6] B. K. Horn. (1999) Projective geometry considered harmful [Online]. Available: http://www.ai.mit.edu/people/bkph/

[7] W. Forstner and E. Gulch, "A fast operator for detection and precise location of distinct points, corners, and centers of circular features," in *Proc. Int. Conf. Fast Processing of Photogrammetric Data*, 1987, pp. 281–305.

[8] R. Vidal and S. Sastry, "Optimal segmentation of dynamic scenes from two perspective views," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 281–286.

[9] K. Daniilidis and H.-H. Nagel, "Analytical results on error sensitivity of motion estimation from two views," *Image Vision Comput.*, vol. 8, pp. 297–303, 1990.

[10] R. I. Hartley and P. Sturm, "Triangulation," in *Proc. Conf. Computer Analysis of Images and Patterns*, 1995, pp. 190–197.

[11] D. Nistér, "Preemptive ransac for live structure and motion estimation," in *Proc. IEEE Int. Conf. Computer Vision*, vol. 1, 2003, pp. 199–206.

Stochastic Car Tracking With Line- and Color-Based Features

Tao Xiong and Christian Debrunner

***Abstract*—Color- and edge-based trackers can often be "distracted," causing them to track the wrong object. Many researchers have dealt with this problem by using multiple features, as it is unlikely that all will be distracted at the same time. It is also important for the tracker to maintain multiple hypotheses for the state; sequential Monte Carlo filters have been shown to be a convenient and straightforward means of maintaining multiple hypotheses. In this paper, we improve the accuracy and robustness of real-time tracking by combining a color histogram feature with an edge-gradient-based shape feature under a sequential Monte Carlo framework.**

***Index Terms*—Color-based tracking, condensation, edge-based tracking, feature integration, Monte Carlo filter, multiple hypotheses, particle filter.**

I. Introduction

In recent years, researchers have developed the concept of driver-assistance systems as a means of reducing the number of traffic accidents and increasing driver comfort. Different types of sensors, such as radar, laser, and acoustic, have been considered for sensing in this application. Thanks to the increasingly powerful computer systems and the less-expensive high-performance video cameras that have become available in the past few years, the use of computer vision technology as a sensor in driver-assistance systems has become more common and has led to increased performance. Vision sensors can provide rich information about the vehicle's surroundings and also have the advantage over active sensors (such as laser rangers or radars) of not causing intervehicle interference.

Manuscript received December 1, 2003; revised July 15, 2004 and August 1, 2004. The Associate Editor for this paper was F.-Y. Wang.

T. Xiong is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: txiong@ece.umn.edu).

C. Debrunner is with the Engineering Division, Colorado School of Mines, Golden, CO 80401 USA (e-mail: cdebrunn@mines.edu).

Digital Object Identifier 10.1109/TITS.2004.838192