Submitted to ICRA-04

A Vision-Based Tracking System for a Street-Crossing Robot

Michael Baker
Computer Science Department
University of Massachusetts Lowell
Lowell, MA
mbaker@cs.uml.edu

Holly A. Yanco
Computer Science Department
University of Massachusetts Lowell
Lowell, MA
holly@cs.uml.edu

Abstract— **This paper describes progress toward a street-crossing system for an outdoor mobile robot. The system can detect and track vehicles in real time. It reasons about extracted motion regions to decide when it is safe to cross.**

Keywords-mobile robots, street crossing, computer vision, vehicle tracking.

I. INTRODUCTION

Why did the robot cross the road? To get to the other side.

In the past, delivery robots have been confined to a single building. Imagine a robot that could travel autonomously across campus to pick up a book at the library, and then bring it back to you in your office. There are systems that are able to drive safely on sidewalks (for example, [1]), avoiding obstacles while staying on a path, but none are able to cross the street. Developing a robot that can cross a street autonomously would allow for delivery robots that could cover a number of buildings, robotic wheelchairs that could drive their users safely across intersections, and sentry robots that could patrol multiple buildings.

Mori [2-5] has explored algorithms for tracking cars in the context of a robotic travel aid for the blind, but he does not address the street crossing problem explicitly. His vehicle detection and tracking results are quite impressive, but his tracking results depict a single oncoming vehicle. In fact, his algorithm explicitly expects to detect at most two vehicles in a frame.

Mori's algorithm needs to detect road boundaries in its initial setup. The algorithm uses dynamic tracking windows overlying a street lane at known distances from the robot. His distance, width, velocity, and collision time estimates are based on a known world distance to a pixel coordinate in the image plane. We believe that this fixed camera constraint is not appropriate for a robot trying to make its way across the street.

A street-crossing mobile robot must be able to detect and track all moving and stationary vehicles in the visible scene in real time, as the robot itself moves through the world. The system must be able to determine the "collision time" of all tracked vehicles. Collision time is the time it will take a vehicle to reach the robot given its current speed and a constant velocity assumption. Being able to determine collision time implies knowing distance and tracking distance over time.

As a safety precaution, human street crossers often make eye contact with drivers. A person's pose and gaze indicate a desire to cross. A person may become impatient or frustrated with a busy street. Some people assume drivers will stop if they walk out in front of their cars.

A robot can't make eye contact and must signal its intention to cross in some nonhuman fashion. The street-crossing robot we are developing is infinitely patient, and will be a safe and conservative street crosser. The robot will only attempt to cross the street when the minimum collision time of all tracked vehicles is greater than the robot's crossing time by some margin of safety.

In this initial work, we focused on crossing the street in marked crosswalks across a one-way or two-way road with no more than one lane in either direction. (See Fig. 1 for a diagram of the robot's camera view at the curb. Fig. 2 shows an image from the robot's left-facing camera.) Marked crosswalks are useful since they are likely to have curb cuts that will allow our wheeled robot to move safely into the street. From a curbside perspective, it is particularly difficult to detect and track vehicles in a multilane scenario due to the ways that vehicle occlusions can occur. Furthermore, although it is illegal to pass a stopped car at a crosswalk, this is a frequent cause of pedestrian accidents at multilane crossings. Statistically, intersections are very dangerous places to cross the street, so we have chosen to start with a safer place first.[1]

Tracking vehicles over time is fundamental to a street-crossing algorithm. Tracking is needed to determine vehicle speed and what a vehicle is doing: accelerating or decelerating, approaching or leaving. Since vehicles are being tracked using motion, slow or stopped vehicles can "disappear." The robot's sonars and SICK laser scanner can detect vehicles in close proximity to the crosswalk. Additionally, since all vehicles are being tracked, we expect to see as many vehicles leaving on the right as entered from the left and vice versa.

[1] Note that the work is being done in Massachusetts, where no place is truly safe to cross.

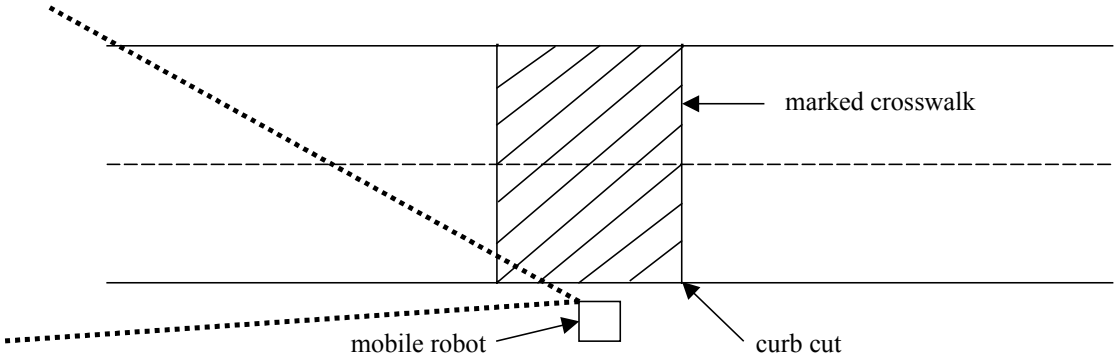

Figure 1. Robot's field of view from the left-facing camera (overhead view).

Figure 2. Robot's curbside perspective. Image taken with the left-facing camera.

The system first begins tracking at the curb, in a stationary position. While determining whether it is safe to cross, the street-crossing system will devote equal processing to the left and right looking image streams. Vehicles approaching from either direction represent danger to the robot.

Once the robot has entered the crosswalk, tracking is done while moving. Processing the left image stream is still necessary, but more processing is shifted to the right stream, which represents more danger to the crossing robot. Should the right side (far lane) become unsafe before the robot completes the street-crossing maneuver, the robot will stop short of the median strip and wait for vehicles in the far lane to yield or pass.

This paper presents the algorithms developed for the vehicle tracking system.

II. Related Work

Systems that use computer vision to track vehicles in real time generally come under two major headings: automated driving systems [6–12] and systems that monitor and analyze traffic scenes [13–21]. Unfortunately, most methods from these domains are unsuitable or inappropriate for the street-crossing problem.

The vast majority of computer vision-based vehicle tracking algorithms take a model based [8, 9, 12, 20, 21] or template matching based approach [7, 10, 11, 22]. Use of a Kalman filter or Extended Kalman filter is quite common for predictive tracking [8, 10, 11, 12, 15, 16, 20, 21]. The Kalman filter assumes a unimodal Gaussian distribution of the correlation feature and a constant velocity model, which limits its usefulness [22]. The use of models and/or templates almost certainly implies special off-line initializations. Typically, models and templates need to be trained before they can be used to track vehicles reliably. This, of course, depends on some cooperation from the real world. Poor training data will cause poor initial results. The off-line requirements and inherent computational overhead due to on-line updating make these approaches undesirable for the street-crossing algorithm. Domain-specific assumptions built into existing motion and scene models do not hold in a typical street-crossing scenario. We have a lower camera angle and do not have the relatively structured or uniform appearance of a typical highway scene.

Some traffic analysis systems use a homography or projective transform between image and world coordinates [15, 16, 19]. This transform is useful for gathering traffic statistics such as vehicle count, speed and congestion. It requires off-line camera initialization or road markers to support on-line camera calibration. This is a reasonable approach in the context of fixed camera traffic surveillance systems, but invalid for a mobile robot.

Some vehicle tracking systems exploit symmetries that arise from the viewing angle. For example, the rear view of a car from an autonomous vehicle is generally symmetric. These systems use models based on vehicle gray level or color symmetries and vehicle edge symmetries or aspect ratios [6, 7, 9]. From the vantage point of a robot waiting to cross the street, the symmetry of an approaching or passing car is ever-changing. Another type of symmetry is based on the relatively uniform palette of gray values on the road surface [23, 24]. Here is another assumption that is generally true of highways and generally false in a typical street-crossing scene. The road model needs to stay current with changing lighting conditions.

Some systems use stereo vision for depth measurement [6]. To use stereo, we would need to mount four cameras on our robot to get depth readings in both directions, making it

impractical. A single camera can compute depth maps using optical flow, but optical flow has been rejected by a number of researchers as being too expensive to compute in real time.

Few vehicle tracking systems deal with shadow removal and/or vehicle occlusions. In the street crossing problem, shadows and occlusions are especially problematic due to the curbside camera angle. Some systems that measure vehicle speed depend on finding the geometric centers of tracked vehicles [14, 18], which implies the ability to segment vehicles perfectly; otherwise, the speed calculations would suffer. For example, a motion region could include overlapping vehicles or a vehicle and its shadow. This problem is less severe in the traffic analysis domain due to the overhead camera view, but an effective street crossing system must solve these issues.

The ground plane constraint is used in a number of vehicle tracking systems [2, 3, 4, 5, 15, 16, 20]. This assumption is exploited in two ways in our tracking algorithm. Since we do not expect to detect vehicles above the horizon line, we do not need to raster scan for motion regions above the horizon line. This represents a significant data reduction without doing any processing. Lipton [22] uses the ground plane constraint to define a "depth ordering" which he uses to reason about the way vehicles can occlude one another. While our algorithm does not currently include explicit occlusion reasoning or disambiguation, we plan to add this in the future. The current algorithm implicitly deals with partial occlusions due to the way the bounding boxes are computed.

A laser range finder has been used for measuring distance on the Navlab5 autonomous vehicle test bed [12]. Our research robot has a forward looking SICK laser, but it is already delegated to the task of finding the curb cut on the far side of the street. To measure distance in both directions simultaneously, we would need two SICK lasers. Even if our research platform could support the weight and size of another laser, they are quite expensive. Radar is another possibility, but our robot is not currently equipped with a radar sensor.

Almost all of the algorithms presented in the literature include some kind of lane masking or road boundary detection to limit the search space for vehicles. In the context of traffic analysis systems, highway lanes are generally straight and relatively easy to detect given the overhead camera view. In the context of autonomous highway vehicles, lane detection is fundamental to following the curvature of the road. Again, the camera view is conducive to detecting lane boundaries reliably as long as the road does not have sharp bends. The street-crossing robot has a sidelong view of the roadway at street level. Perfect detection of the road boundaries requires some cooperation from the real world. The street must be generally flat, straight, and free of traffic at the time of detection; for this application, explicit road boundary detection is an unrealistic expectation. Implicitly, road boundaries are being detected as the left and right limits of motion in the scene. As a practical matter, it is unrealistic to assume that vehicles remain wholly within their lane at all times.

III. Vehicle Detection and Tracking Algorithm

To detect vehicles, a number of processing steps are taken. Movement in the image is found by differencing successive frames of the scene. The differenced image is then filtered using a 3x3 median filter to remove noise and background motion. Edges are extracted using a Sobel edge detector. The highest edge points are candidates for a line marking the top of a car. The bottom of the car is found using Mori's sign pattern.

A. Image Differencing

The principal tracking feature of the algorithm is motion. Image differencing is a common technique used for extracting scene motion. There are two general methods: reference frame differencing and interframe or temporal differencing. Both methods are sensitive to background and camera motion, but the reference frame (or background subtraction) method is unsuitable for a street-crossing robot. A reference frame must be grabbed when the scene is stationary and free of objects that could move while the tracking process is active. Alternatively, a reference frame could be computed by observing the constant parts of the image over time. Next, the reference frame needs to be updated on-line to stay current with lighting changes. Finally, whenever the robot moves, the reference frame becomes invalid. It is highly unlikely that the world will cooperate whenever the algorithm needs a clean reference frame.

Our vehicle detection algorithm uses simple interframe differencing to extract motion regions from frame to frame. Some researchers have used a variation on simple image subtraction called "double differencing" (see, for example, [17, 18]). The double difference method takes three consecutive frames, performs two subtractions against the middle frame, then ANDs the subtraction results to obtain a single motion image. Empirically, results were similar using double differencing and interframe differencing, so the faster method was selected.

B. Noise Filtering

A 3x3 median filter is used to remove camera noise and tiny flutters due to background motion. Most motion due to wind on trees and telephone wires is ignored by the assumption that vehicles stay in contact with the road plane. Filtering the difference image is more efficient than cleaning the raw camera images separately.

C. Edge Extraction and Thresholding

A Sobel edge detector is used to delineate the motion edges. Sobel is used frequently in the literature because it is efficient and provides generally good results. In the context of vehicle tracking, we expect to find mostly horizontal and vertical edges due to the typically rectangular contour of vehicles.

The edge detected image is thresholded to extract the motion regions and facilitate raster scanning of the image. Experiments have used a fixed threshold value that worked well for the lighting condition. Histogram thresholding did not achieve a noticeably better result. Since (near-)optimal extraction of motion regions is essential to the success of this algorithm, work is continuing towards the inclusion of an adaptive thresholding method.

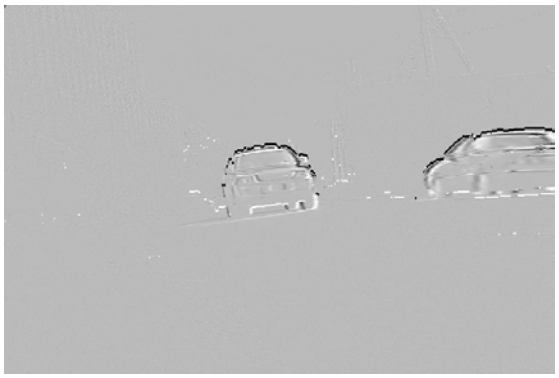

Figure 3. Result of Mori (white pixels) and roofline (black pixels) raster scans. Notice how the Mori scan detects the underneath of the vehicle without detecting the shadow region (which can be seen in the lighter gray color underneath the left car).

D. Mori Scan

Mori [2–5] describes a "sign pattern" for vehicle detection that is invariant to weather (lighting) and holds for wet and dry roads. Stated succinctly, the shadow underneath a vehicle is darker than any other point on the road. As Mori explains, if the shadow cast by a vehicle appears to be as dark as the shadow underneath a vehicle, this is "an illusion of brightness constancy."

Mori's result has been used in a highway driving context [24]. Our algorithm uses Mori's result to scan for vehicle bottoms in a single raw image (see Figure). The sign pattern is very useful for detecting vehicles while ignoring shadows. Before this was incorporated into the algorithm, shadows were handled in an ad hoc way, by eliminating short, wide bounding boxes as being caused by shadows. This simple rule eliminated most shadows effectively, but some shadows remained, which were sometimes grouped with a trailing vehicle and resulted in oversize bounding boxes. Use of the Mori result obviates the need for explicit shadow removal.

E. Data Reduction and Abstract Data Types

To support the vehicle detection algorithm, we define three abstract data types: lines, boxes, and (processed) frames. A line corresponds to the roofline of a vehicle. The fully segmented image is raster scanned from the horizon line downward to extract the highest motion pixels in the image plane. This pixel array is mined for lines; lines are defined as a minimum number of consecutive pixels such that the row coordinate of adjacent pixels does not differ by more than a specified value. The line finding algorithm tolerates a specified percentage of outlier pixels. If too many lines are rejected during the line-mining process, either because they're too short or have too many outlier pixels, the whole processed frame is rejected. A high number of rejected lines indicates that a frame is too noisy to be useful for tracking information.

Fig. 3 shows the result of the processing once the roofline and Mori scans have been completed. An efficient downward and outward scan starting at either end of a roofline finds the vehicle sides. Finally, the data from the Mori scan is added to completely determine a vehicle's bounding box. The computed bounding boxes are sanity checked for things like negative height and unrealistic aspect ratios. Frame statistics such as number of vehicles (boxes) detected are stored in a frame object.

F. History Tracking

The algorithm includes a history component that supports the actual tracking of detected vehicles. Assuming a fast frame rate, a tracked vehicle does not move very far in the image plane from frame to frame. Segmentation can be flawed, due to camera noise and extraneous motion in the scene, but we can use a vehicle's history to smooth the results. For example, a vehicle that has been approaching the robot for a number of frames can not spontaneously reverse direction and go the other way, even if the bounding box suggests otherwise.

IV. RESULTS

Vehicle detection results are shown in Fig. 4. This figure shows six frames from a video sequence from the robot's left-facing camera. The white boxes drawn on the images represent the bounding boxes of the tracked vehicles found by our algorithm. The algorithm is able to track multiple cars, both approaching and driving away from the robot. Cars that are very far in the distance, such as the second approaching car in the first two frames shown, are not detected because their roofs do not have enough pixels to mark the line as a roofline. However, once the car gets a bit closer, it is detected. When the second approaching car is detected in the third frame of the tracking sequence, it is still approximately 150 feet away from the camera.

The sequence also shows the occlusion of the passing car in the far lane by the approaching SUV. In the first two frames of the sequence, the approaching SUV and passing car are computed as separate bounding boxes. In the third frame, the two cars are merged into one box. Looking closely at the image, you can see that the top of the passing car is almost at the same height as the SUV's hood. In the next two images, the passing car is completely occluded by the SUV. Finally, in the last image, the algorithm has redetected the passing car as it emerges from behind the SUV.

The image sequence is only displaying the bounding boxes found, not the system's hypothesis about the direction of each vehicle computed by the history tracking algorithm. When deciding if it is safe to cross, direction information would be used.

The current system does not run at frame rate, but we are currently running on large images (720 x 480 pixels). As development continues, we will use smaller image sizes. Additionally, we are optimizing the algorithm.

V. CONCLUSIONS AND FUTURE WORK

The current tracking algorithm is able to identify vehicles in two lanes, one moving towards and one moving away from the robot's camera. Multiple vehicles can be found traveling in both lanes.

Figure 4. Six frames from a video sequence of approaching and passing vehicles from the robot's left-facing camera. The white boxes show the cars found by the tracking algorithm. The sequence shows the detection of a second approaching car as it gets closer. It also shows a passing car being tracked, then occluded by the oncoming vehicle, then finally tracked again as it emerges from behind the oncoming vehicle.

We are starting to use the tracking results for actual crossings. The robot will need to choose a safe starting time. Then, as it moves across the street, it will need to continue tracking vehicles to update its safety measure.

Extensions to the current system could include multilane and intersection crossings. For these extensions, other scene features could be used as clues to find safe crossing locations and safe crossing times. For example, the robot could decide to follow a pedestrian across the street. Traffic light changes could also be used to help the system reason about safe crossing times.

REFERENCES

[1] H. A. Yanco, "Shared user-computer control of a robotic wheelchair system," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, September 2000.

[2] N. M. Charkari, K. Ishii, and H. Mori, "Proper selection of sonar and visual sensors for vehicle detection and avoidance," Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, Vol. 2 , 12-16 Sept. 1994, pp. 1110 –1117.

[3] N. M. Charkari and H. Mori, "A new approach for real time moving vehicle detection," Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1, 26-30 July 1993, pp. 273 – 278.

[4] H. Mori and N. M. Charkari, "Shadow and rhythm as sign patterns of obstacle detection," International Symposium on Industrial Electronics, 1993, pp. 271--277.

[5] H. Mori, N. M. Charkari, T. Matsushita, "On-line vehicle and pedestrian detection based on sign pattern," IEEE Trans. on Industrial Electronics, Vol. 41, No. 4, pp. 384-391, Aug. 1994.

[6] M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele, "Stereo vision-based vehicle detection," IEEE Intelligent Vehicles Symposium, Detroit, MI, October 2000, pp. 39-44.

[7] M. Betke, E. Haritaoglu, and L. Davis, "Highway scene analysis in hard real-time," Intelligent Transportation Systems, IEEE, July 1997.

[8] F. Dellaert and C. Thorpe, "Robust car tracking using Kalman filtering and Bayesian templates," Proc. of the SPIE - Int. Soc. Opt. Eng., Vol. 3207, October 1997.

[9] U. Regensburger and V. Graefe, "Visual recognition of obstacles on roads," Proc. IROS '94, pp. 980–987, Munich, Germany, 1994.

[10] M. B. van Leeuwen and F. C. A. Groen, "A platform for robust real-time vehicle tracking using decomposed templates," Intelligent Autonomous Systems Group, Faculty of Science, University of Amsterdam, Amsterdam, The Netherlands, unpublished.

[11] M.B. van Leeuwen and F.C.A. Groen, "Vehicle detection with a mobile camera," Intelligent Autonomous Systems Group, Faculty of Science, University of Amsterdam, Amsterdam, The Netherlands, unpublished.

[12] L. Zhao and C. Thorpe. "Qualitative and quantitative car tracking from a range image sequence," Computer Vision and Pattern Recognition, pages 496-501, 1998.

[13] L. A. Alexandre and A. C. Campilho, "A 2D image motion detection method using a stationary camera," RECPAD98, 10th Portuguese Conference on Pattern Recognition, Lisbon, Portugal, 1998.

[14] E. Atkociunas and M. Kazimianec, "Aspects in traffic control system development," Vilnius University, Faculty of Mathematics and Informatics, Jyvaskyla, 2002.

[15] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. "A real-time computer vision system for vehicle tracking and traffic surveillance," Transportation Research: Part C, vol 6, no 4, 1998, pp 271-288.

[16] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A realtime computer vision system for measuring traffic parameters," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1997.

[17] R. Cucchiara, M. Piccardi, A. Prati, and N. Scarabottolo, "Real-time detection of moving vehicles," Proc International Conference on Image Analysis and Processing, Venice, Italy, pp. 618 – 623, September 1999.

[18] D.J. Dailey and L. Li, "Video image processing to create a speed sensor," ITS Research Program, Final Research Report, College of Engineering, University of Washington, Seattle, Washington, March 1999.

[19] N. Ferrier, S. Rowe, and A. Blake, "Real-time traffic monitoring," Proc. 2nd IEEE Workshop on Applications of Computer Vision, pp. 81–88, 1994.

[20] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, S. Russel, "Towards robust automatic traffic scene analysis in real-time," Proc. Int'l Conf. Pattern Recognition, pp. 126–131, 1994.

[21] D. R. Magee, “Tracking multiple vehicles using foreground, background and motion models,” University of Leeds, School of Computer Studies, Research Report Series, (Submitted to European Conference on Computer Vision, May 2002), December 2001.

[22] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, “Moving target classification and tracking from real time video,” IEEE Workshop on Application of Computer Vision, pp. 8–14, 1998.

[23] M. Betke and H. Nguyen, “Highway scene analysis from a moving vehicle under reduced visibility conditions,” Proc. of the International Conference on Intelligent Vehicles, IEEE Industrial Electronics Society, Stuttgart, Germany pp. 131–136. Oct. 1998.

[24] C. Tzomakas and W. von Seelen, “Vehicle detection in traffic scenes using shadows,” Internal Report IRINI 98-06, Institut fur Neuroinformatik, Ruhr-Universitat Bochum, Germany, August 1998.