Overtaking Vehicle Detection Using Implicit Optical Flow

Parag H. Batavia Dean A. Pomerleau Charles E. Thorpe
Robotics Institute - Carnegie Mellon University
Pittsburgh, PA, USA
Keywords - Optical Flow, Vehicle Detection, Blindspot

ABSTRACT
We describe an optical flow based obstacle detection system for use in detecting vehicles approaching the blind spot of a car on highways and city streets. The system runs at near frame rate (8-15 frames/second) on PC hardware. We will discuss the prediction of a camera image given an implicit optical flow field and comparison with the actual camera image. The advantage to this approach is that we never explicitly calculate optical flow. We will also present results on digitized highway images, and video taken from Navlab 5 while driving on a Pittsburgh highway.

INTRODUCTION
Many active methods of obstacle detection such laser, radar, and sonar may not be possible in the Automated Highway System due to interference between vehicles. For instance, two automated vehicles operating near each other may have to coordinate radar frequencies to avoid interfering with each other. While these problems can be solved, they present a great challenge. Therefore, it is worthwhile to investigate completely passive approaches, such as vision, for obstacle detection. While vision generally cannot provide the accuracy, resolution, and speed of a high end laser range finder, an effective single camera vision system can be built cheaply and operate with relatively minimal computational resources.

We have developed a system which is optimized for detecting obstacles in the blind spot of an intelligent vehicle operating on city streets and highways. By blind spot, we mean the area behind and to the left and right of the vehicle, which is normally not visible in the rear view and side view mirrors. Generally, the blind spot is only viewable when the driver turns his head.

The principal approach is to make certain assumptions about the operating environment which let us solve the inverse perspective problem to compute a predicted image given camera parameters and vehicle velocity. This predicted image is then compared against an actual image, and any area which does not match our prediction is presumed to be caused by an obstacle. One advantage is that although we use the notion of optical flow to generate the predicted image, we never have to explicitly calculate an optical flow field, which can be an expensive operation. In addition, our method is sensitive to vertical obstacles, but ignores lane markers and shadows cast by stationary objects. We present previous work, a system description, experimental results, and conclusion.

PREVIOUS WORK
Many people have investigated explicitly calculating optical flow for obstacle detection. Bohrer et al [4] looked at the idea of detecting vertical objects using optical flow. They present a useful analysis of the influence of camera parameter estimation errors. De Micheli and Verri [6] detect obstacles using vehicle angular velocity around an axis orthogonal to the road, and generalized time to contact. They show that motion recovery becomes a linear problem if the image plane is orthogonal to the road and the optical axis points along the direction of travel. Their method, like ours, seems to be sensitive to calculated camera parameters. However, their method for calculating the angular velocity requires horizontal texture along the vertical line (in the image plane) passing through the image plane origin. This means that it is not optimal for road scenes, in which that line may consist mostly of untextured road.

Enkelmann [9] calculates a predicted optical flow field (OFF) given camera parameters, focus of expansion, and camera motion. His results, while preliminary, showed the feasibility of using a planar parallax model to generate a predicted optical flow field to compare to an actual flow field. We use this approach, but get around calculating the actual optical flow by using the predicted flow to warp the current image, and compare images.

Graefe [10] uses a distance dependent subsampling along with vehicle templates to look for vehicles approaching the ego-vehicle from the rear. Their system runs in real time on specialized hardware.

Zhao and Yuta [16] use stereo vision to predict the image seen by the right camera given the left image, assuming flat earth. They use inverse perspective mapping to transform every point in the left image to world coordinates, and reproject them back onto the right image, and compare against the actual right image. They are able to pick out contours of objects above the ground plane, while rejecting stationary markings on the road.

Bertozzi and Broggi [2] use stereo vision for generic obstacle detection, but instead of warping the left image into the right image, they compute the inverse perspective map of both images, and compare those. They find objects that are not on the ground plane, and use this to determine the free space in front of a vehicle. They run on *Paprica*, a massively parallel SIMD machine.

Dickmanns [7] uses parallel transputers to attack obstacle detection and avoidance from different angles. They use recursive estimation techniques to extract obstacle vehicle state using monocular vision, and knowledge based systems for 3-D obstacle recognition while dealing with occlusions. Their VaMoRs-P vehicle drives on highways at speeds of up to 130km/h.

Betke et al [3] use a custom hard real time operating system running on a PC to look for sudden changes in brightness over a few frames to indicate a possible passing vehicle. This is verified using model based template matching, which requires a model set consisting of the front and rear views of several different types of vehicles. To recognize distant cars, they use a feature based approach, looking for horizontal and vertical edges. Their system is also sensitive to contrast changes caused by shadows.

SYSTEM DESCRIPTION

To detect vehicles, we do the following: first, we sample the image, perform an edge detection, and use our planar parallax model to predict what that edge image will look like after travelling a certain distance. Next, we capture an image after travelling our assumed distance, and compare it to the prediction. For each edge point in the predicted image, we verify that there is a corresponding edge point in the actual image. If there is a match, then our prediction (based on a flat earth assumption) is verified. Otherwise, we know that the cause of the horizontal line in the predicted image was an obstacle (i.e., above the ground plane). There are 4 components to the system - sampling and preprocessing, dynamic image stabilization, model-based prediction, and obstacle detection. Each is described below.

Sampling and Preprocessing

The first key to operating at near real time is to not process the entire image. A full image of 640x480 pixels results in 9,216,000 pixels that need to be processed each second (assuming 30 frames per second). Because we are only interested in monitoring the blind spots, we do not need to process the entire image. Instead, we specify one or two windows (depending on the number of lanes), each centered on the lanes surrounding the lane the vehicle is in. Currently, those locations are set by hand. However, the RALPH lane tracking system [14] can provide lane curvature information which can be used to determine the location of the adjacent lanes. So, rather than processing over 9 million pixels per image, we process two windows of 100x100 pixels. This is a total of 20,000 pixels per frame. This number is further reduced, as we only compute our prediction on edge pixels. By reducing the amount of data, we can handle 15 frames per second on a 166 MHz Pentium running the QNX real time operating system.

Two stages of preprocessing are required - smoothing and edge detection. The smoothing is done by a 3x3 Gaussian window, run over each of the sampling windows. The edge detection is required, as mentioned above, as we only compute the prediction on edge pixels. A standard sobel edge detection is used [11]. A prototype implementation of this system, done by Alban Pobla [13] used the Canny edge detector [5]. However, this required the edge images to be computed off-line.

Dynamic Image Stabilization

The ultimate goal is generation of a predicted edge image at time *t* for comparison with an actual image taken at time *t*. While the model described in the next section can generate this predicted image, it does not take vehicle vibration into account. When a car or van is driving down a road, it vibrates a great deal vertically, and this needs to be accounted for. We tried using a camcorder with dynamic image stabilization, but it was tuned to dampen vibrations at a much lower frequency than those generated by a road.

Our solution is to compute 1-dimensional optic flow (in the vertical direction). For speed and simplicity, we take 10 column-wise slices out of each image, and find the largest magnitude edge in each slice using a 1-D sobel filter. Ideally, these edges will correspond to features in the background of the scene, such as a treeline or overpass. The slices extend from the top of the image to our calculated horizon - this usually prevents us from finding edges due to moving vehicles. These edges are then searched for in the next image, giving us 10 estimates of global vertical flow. A weighted average is computed, based on correlation score. This global flow is used in adjusting the prediction model described in the next section. During normal operation, however, there was only a 1-2 pixel ‘bounce’ due to vehicle vibration.

Model based prediction

Once the edge images are calculated, a prediction of the next image is generated using inverse perspective mapping [12]. In regular perspective projection, 2-D image coordinates are computed from real world 3-D coordinated. The problem with the inverse projection is that it is a mapping from R^2 to R^3. This means that the solution is a ray passing through both the image point and the focus. However, by assuming that we are travelling on a flat road and that everything we see lies on that road, the world coordinate of an image point is constrained to be the intersection of the ray with the plane formed by the road. In other words, by fixing the world Y coordinate to be 0, we end up transforming from one 2-D plane to another. For a more complete description, see Appendix A of [1], which is a tech. report containing this paper’s text, along with additional results and a detailed explanation of the prediction equations.

Solving this constrained inverse perspective mapping tells us that given an edge pixel and the flat earth assumption, we can calculate its coordinates in a coordinate frame relative to the vehicle. If we also know the vehicle velocity (which is available via on-board GPS) and time between successive frames, we can calculate how far the vehicle has travelled. This means that we can compute the new coordinates of each edge pixel in the image. This new coordinate can be projected back into the image plane to give us a prediction of what an image would look like after travelling a certain distance. The global vertical flow described in the previous section is added to the prediction to account for vehicle vibration. The prediction equations, taken from Duda and Hart [8], are given in Appendix A of [1].

This method is most sensitive along the edges of the visual field. The predicted location of a point directly in front of the camera does not vary much with its height above the ground (unless the distance between images is very large). Along the edges, however, an object at the height of an average car causes an easily detectable prediction error. This is because when an object translates along a given direction, the magnitude of the optical flow field near the focus of expansion is much less than at the edges. Therefore, the closer the approaching vehicle is to one side of the image, the larger the prediction error. Since we are concentrating on the blind spot, this is an ideal sensitivity profile.

Obstacle Detection

The predicted image that was generated in the previous section is based on the assumption that any objects in the image lie flat on the road plane. If, in fact, they do not lie on the road plane, they will project to slightly different image coordinates than predicted. It is this discrepancy that we search for. There is a certain amount of inherent noise in this procedure due to camera calibration error, so simply subtracting the two images does not work. Instead, for every edge pixel in the predicted image, we search a 3x3 neighborhood of the actual next image for an edge pixel. If we find an edge pixel in this area, we assume the prediction is satisfied - i.e, that edge point was caused by an object (such as a shadow) on the road plane. If an edge pixel in the predicted image is *not* found in the actual next image, we assume that it projected to some point outside our 3x3 window, in which case it was caused by an object *not* on the ground plane. One problem with this search is that predictions can be falsely satisfied if edges are very close together. However, for detecting vehicles, this is not too bad of a problem. Although it may seem counter-intuitive that *not* seeing something means it's an obstacle, a feature which is on the ground plane would be accounted for by our prediction, and therefore would have been found (i.e., 'seen'). Any such object which is not seen is therefore assumed to be an obstacle.

Even with the above steps, there is still some noise present in both the prediction and actual next images, which cause false positive and false negatives to affect the results. Therefore, we make one more assumption, which is that any vehicle that we are looking for has horizontal edges. So, for any prediction failure to be considered an obstacle vehicle, it must be part of a larger set of horizontally aligned prediction failures. Once we generate the full obstacle image, we sum the image horizontally, and look for peaks which exceed a preset threshold, which depends on sampling window size. If we find a peak, we then know that there is a horizontal edge in the predicted image which is *not* in its predicted location in the actual image. Therefore, it was generated by an object which is not on the ground plane, and therefore must be a vehicle. We assume any horizontal line which *does* match its prediction is caused by a shadow or other stationary feature on the ground.

EXPERIMENTS

Real World Digitized Images

Figures 1-4 show a set of images taken from a camera mounted on the rear window of Navlab 5. We drove NL5 along Interstate 279 in Pittsburgh, while maintaining a speed of 55-65 m.p.h. We were generally driving a bit slower than traffic, so we could get video of vehicles passing us from the rear. As these experiments were done off-line, we processed the entire image.

For camera calibration, we measured camera height by hand, as it is possible to do this within a centimeter, which was less than 1% of the actual camera height. The pitch, however, was calibrated using a method which relied on knowing the width of a lane. This produced better results. However, camera yaw was still a problem. Not wanting to resort to an explicit calibration method like Tsai's method [15], we tuned the yaw by hand, until a good match was found. For our camera setup, this turned out to be 1.5 degrees. In the future, this can, and should be calculated on-line.

Figure 1 shows an image taken from NL5 with two vehicles approaching us. Figure 2 shows that same situation, 120 mS after Fig. 1. Figure 3 shows the prediction of Figure 1, with the flat earth assumption, overlaid over the real edge image of Figure 2. Note that the lane marker are aligned almost precisely. However, the vehicles are not aligned. The hill in the background isn't matched perfectly either, because it is above the ground plane. Figure 4 shows the final obstacle image. The histogram on the left side is a row-sum of the image. The nearer vehicle is easy to detect, as indicated by the spikes in the histogram. The dashes on the top of the images are VITC time code.

Live Video

We tested our system using the same video from which we digitized Figs. 1-24. Everything worked qualitatively the same using live video, as it did with the digitized images. A few minor problems

Fig. 1 - Rear view road image

Fig. 2 - Same image after 120 mS.

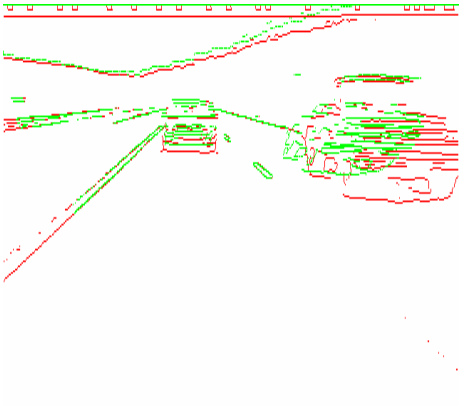

Fig. 3 - Difference Image. The red overlay is the edge image of Fig. 2, the green overlay is the predicted Fig. 2.

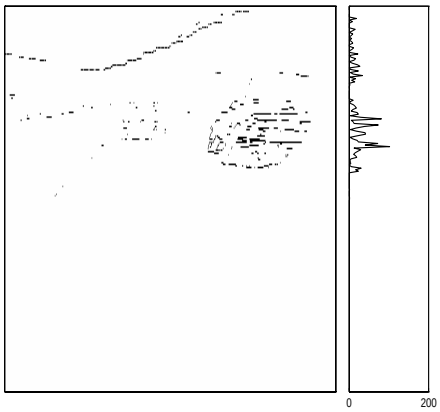

Fig. 4 - Obstacle image. The vehicle on the right is easily detectable.

added some noise to the system, though. First, our digitizer hardware wasn't capable of directly reading the time code on the video tape, so we depended on the system timer to calculate how many frames have passed between runs. The system timer has a resolution of 10 mS., which is high compared to a frame time of 33 mS. Second, as we were not running directly on Navlab 5, we didn't have access to the GPS provided speed, so we hard set it at 60 m.p.h. Regardless, the results were very encouraging, and comparable to the off-line version. A demonstration is available on the web at http://hoagie.ius.cs.cmu.edu:8000/~parag/research/optic_flow/optic_flow.html.

FUTURE WORK

Future work will concentrate on using curvature information provided by RALPH to modify the prediction to handle curved roads. An onboard pitch sensor can also be used to modify the model to allow operation on varying terrain.

We are also looking at the possibility of using custom hardware to explicitly calculate the actual optical flow field over the entire image. This would free us from the limitation of looking only at relatively small image windows, and would allow us to look for aberrant flow over the entire road scene.

Camera calibration is also an issue which needs to be addressed. We will be investigating possible methods for calibration which will *not* require 40-50 calibration points, which would be infeasible for an AHS type application.

CONCLUSIONS

We have presented a system which solves a restricted inverse perspective mapping problem to warp a camera image to how it would appear after travelling a distance *d*, given our planar assumptions. This warped image is then compared to the actual image taken at distance *d*, and an obstacle image is generated. The strengths of this approach are that it is computationally cheap (due to not computing an actual optical flow field), can run on standard PCs, and is not affected by shadows and other non-moving objects which lie in the ground plane. The disadvantages to this approach were discussed in the future work section. These initial experiments show promise, and we believe that this is a potentially viable way to detect blind-spot impingement using a cheap, passive sensor.

ACKNOWLEDGMENTS

This material is based upon work supported in part by a National Science Foundation Graduate Research Fellowship, and in part by the USDOT under Cooperative Agreement Number DTFH61-94-X-00001 as part of the National Automated Highway System Consortium. The authors would also like to thank Peter Rander for his help in explaining the problems in *really* knowing what you're digitizing.

REFERENCES

[1] Batavia, Parag, Pomerleau, Dean, and Thorpe, Chuck, "Detecting Overtaking Vehicles With Implicit Optical Flow", *CMU RI Technical Report, CMU-RI-TR-97-28,* 1997.

[2] Bertozzi, Massimo, and Broggi, Alberto, "Read-Time Lane and Obstacle Detection on the GOLD System," *Proc. Intelligent Vehicles,* 1996.

[3] Betke, Margrit, Haritaoglu, Esin, and Davis, Larry, "Multiple Vehicle Detection in Hard Real-Time," *Proc. Intelligent Vehicles,* 1996

[4] Bohrer, Stefan, Brauckmann, Michael, and von Seelen, Werner, "Visual Obstacle Detection by a Geometrically Simplified Optical Flow Approach," *10th European Conference on Artificial Intelligence,* B. Neumann, Ed., 1992.

[5] Canny, John, "A Computational Approach to Edge Detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, no. 6, p.679-98.

[6] De Micheli, E., Verri, A., "Vehicle Guidance from One Dimensional Optical Flow," *Proc. IEEE Intelligent Vehicles Symposium*, 183-189, 1993

[7] Dickmanns, E.D., "Performance Improvements for Autonomous Road Vehicles," *Int. Conf. on Intelligent Autonomous Systems*, 1995.

[8] Duda, Richard, and Hart, Peter, *Pattern Classification and Scene Analysis*, John Wiley Publishers, New York, 1973.

[9] Enkelmann, Wilfried, "Obstacle Detection by Evaluation of Optical Flow Fields from Image Sequences," *Proc. First European Conference on Computer Vision,* O. Faugeras, Ed., 1991.

[10] Graefe, Volker, and Efenberger, Wolfgang, "A Novel Approach for the Detection of Vehicles on Freeways by Real-Time Vision," *Proc. Intelligent Vehicles*, 1996.

[11] Horn, B.K.P., *Robot Vision*, MIT Press, Cambridge, 1986.

[12] Mallot, A.H., Buelthoff, J.J., and Bohrer, S., "Inverse Perspective Mapping Simplifies Optical Flow Computation and Obstacle Detection," *Biological Cybernetics* 64:177-185, 1991

[13] Pobla, Alban, "Obstacle Detection Using Optical Flow: a Faster than Real-Time Implementation," CMU Internal Document

[14] Pomerleau, Dean, "RALPH: Rapidly Adapting Lateral Position Handler," *Proc. IEEE Symposium on Intelligent Vehicles*, 1995

[15] Tsai, Roger, "An Efficient and Accurate Camera Calibration Technique for 3d Machine Vision," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* 1986

[16] Zhao, Guo-Wei, and Yuta, Shin'ichi, "Obstacle Detection by Vision System for Autonomous Vehicle." *Proc. Intelligent Vehicles*, 31-36, 1993.