Three-camera stereo vision for intelligent transportation systems

Jason Bergendahl, Ichiro Masaki, Berthold K. P. Horn
Three-camera stereo vision for intelligent transportation systems

J. Bergendahl, I. Masaki, B.K.P. Horn

{bergenda, masaki, bkph}@ai.mit.edu
Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, MA 02139 USA

ABSTRACT

A major obstacle in the application of stereo vision to intelligent transportation systems is high computational cost. In this paper, a PC based three-camera stereo vision system constructed with off-the-shelf components is described. The system serves as a tool for developing and testing robust algorithms which approach real-time performance. We present an edge based, subpixel stereo algorithm which is adapted to permit accurate distance measurements to objects in the field of view using a compact camera assembly. Once computed, the 3-D scene information may be directly applied to a number of in-vehicle applications, such as adaptive cruise control, obstacle detection, and lane tracking. Moreover, since the largest computational cost is incurred in generating the 3-D scene information, multiple applications that leverage this information can be implemented in a single system with minimal cost. On-road applications, such as vehicle counting and incident detection, are also possible. Preliminary in-vehicle road trial results are presented.

Keywords: stereo vision, feature matching, subpixel precision, intelligent transportation systems

1 INTRODUCTION

Intelligent transportation systems (ITS) have been the focus of considerable research effort in both academic and industrial settings over the past several years. Technologies currently under investigation for the detection of objects in the path of a moving vehicle include millimeter wave radar, optical distancers, and machine vision. The merits of machine vision relative to alternative technologies for ITS are discussed at length in the literature. To summarize, vision systems (1) are passive in nature, (2) have potential for low cost through large scale production in a modern CMOS processes, and (3) can perform multiple applications in a single system.

The feasible applications of a vision system applied to ITS include in-vehicle systems for obstacle detection, lane tracking, and adaptive cruise control (ACC). Systems incorporated into the transportation infrastructure are also possible. One such on-road application is automatic traffic monitoring for congestion management or incident detection. In the specific case of ACC, in which we desire to maintain a safe following distance to a vehicle in front, depth clues are essential. While not strictly necessary for the other applications, depth clues can be applied to improve performance or accuracy. Stereo is a useful technique for recovering depth from 2-D images. Unfortunately, most existing stereo systems are either too slow or use a large amount of processing hardware. Our goal is to build a system that achieves an acceptable level of performance using a modest amount of hardware.
To this end, we have constructed a three-camera stereo vision system using off-the-shelf components. The rest of this paper describes the implementation of this system, a sample algorithm, and results from on-road trials.

2 BACKGROUND

Our approach is to recover the depth to features in the 3-D environment using a stereo triple acquired from three cameras with a special geometry. We find point correspondences among features in the images, and compute depth from disparity.

2.1 Stereo vision

The key problem in stereo vision is finding conjugate points, i.e., points in each view which correspond to the same point in the 3-D environment. To find conjugate points the camera geometry must be known, specifically their relative orientation and position. Ideally, we should also have knowledge of the internal camera parameters. A rigorous approach to finding conjugate points involves determining the internal calibration parameters of each camera individually, followed by a determination of the coordinate system transformations among them (decomposed as a rotation followed by a translation). This point transformation may then be applied to compare points in the different views.

Some approaches apply area correlation of small patches of the original greyscale image to solve the correspondence problem, in some cases calculating disparity to subpixel resolutions. Instead of finding conjugate points, this technique searches for conjugate patches among the different views which have similar brightness patterns. While producing a dense depth map, area correlation methods usually require massive parallelism to perform the required computation at acceptable frame rates, and with reasonable disparity search range. The computational load can be reduced by matching sparse features among the views.

In the simple case of two cameras with optical axes aligned and a baseline oriented in the x (horizontal) direction, distance may be computed directly from disparity for corresponding image points as:

\[ z = b \frac{f}{x' - x'_r} \]  

where \( b \) is the length of the baseline, \( f \) is the focal length, and \( x'_l \) and \( x'_r \) are the x-coordinates in the left and right images respectively. As we decrease the length of the baseline, the uncertainty in the distance measurement increases due to quantization error in the difference \( (x'_l - x'_r) \).

2.2 Our stereo setup

The method of directly applying the internal and external camera calibration parameters amounts to a matrix multiplication, which adds significant overhead to the task of comparing points in different views. We set aside this approach in favor of constructing a special camera geometry which allows us to reduce the computational load. In particular, we physically align the optical axes of the left and right cameras, and we make allowances for residual alignment errors in our algorithm. With the cameras positioned on a horizontal line, the epipolar lines correspond to horizontal pixel rows in the images, facilitating direct comparison of points among the images with little overhead.
The process of finding conjugate patches with area correlation methods requires a large number of pixel comparisons, followed by computation of a metric for estimating patch similarity, typically the sum of squared differences (SSD). We elect to find point correspondences between features instead of patches. Although feature extraction typically requires a convolution, off-the-shelf hardware exists to perform this computationally expensive operation at high speed. One notable disadvantage of the feature based approach is sparser depth maps; we are trading density of the depth map for speed. However, even area correlation methods give unreliable depth estimates in uniform, "featureless" regions.

The particular features that we use are vertical edges since they are characteristic of vehicles and roadway markings. This choice of feature orientation requires the cameras to lie on a horizontal line. If the baseline orientation deviates from true horizontal, we will begin to lose the vertical edges and accuracy (the effective baseline length is decreasing). Any such deviations should be slight however, since the cameras are rigidly attached to the test vehicle.

By adding a third camera, we can further reduce the computational load of finding corresponding feature points among the views. Given a candidate correspondence between two features in the left and right image, we can calculate the expected position of the feature in the center image and check that it indeed exists there. We again use a special geometry to minimize the computational load of this check. We simply position the center camera at the midpoint of the baseline between the left and right cameras, with its optical axis aligned. The center camera in our setup is used solely to disambiguate matches, not to provide multiple binocular views, as others have done.

Visions systems specific to ITS require a short baseline so that it may be installed in an automobile or other vehicle. To overcome the trade-off between short baseline and accuracy in the distance measurement, we adopt a subpixel resolution approach in localizing features. This approach will be described in more detail below.

3 IMPLEMENTATION

Our system has three major components: the camera assembly, the PC based image processing hardware, and the PC itself. A block diagram of the system architecture is shown in Figure 1. The PC is present to both serve as host to the image processing hardware and to perform the high level functions of the algorithm. Power for the system is obtained exclusively from the electrical supply of the test vehicle (through the cigarette lighter).
3.1 Camera assembly

The system uses three standard CCD cameras. The cameras are equally spaced, with their optical axes aligned. We require that the cameras lie on a short baseline. In our experimental setup, the left and right cameras are 252mm apart, with the center camera located midway between them. The cameras are secured to a metal baseplate, machined to assist in physical alignment. After mounting, the alignment of the cameras was fine-tuned, and the cameras locked into position. However, some small misalignment invariably remains. The lenses have focal length 35mm, yielding a horizontal field of view of 10.4° and vertical field of view of 7.9° when used with our cameras.

3.2 PC-based image processing hardware

A PC serves as the host for the system. The PC houses off-the-shelf image processing hardware, which has three principle components: (1) an RGB acquisition module, (2) three image buffers, and (3) a convolver. The acquisition module synchronizes the cameras and is capable of capturing each of the three monochrome channels simultaneously. The acquisition module and image buffers taken together comprise a standard frame grabber. The remaining component of the image processing hardware is a convolver which performs edge detection (described below). The convolver was selected for its capability of performing a 3 x 3 convolution at a pixel rate of 40 MHz. The image processing hardware exploits the high bandwidth of the PCI bus to transmit the result of the convolution to host memory at high data rates (45 Mbytes/sec measured).

3.3 CPU

The 100 MHz Pentium processor used in this setup represents somewhat less than the state of the art in processor performance. Its function is to perform the higher level functions of the algorithm (everything after the convolution) within a frame time less the time required for edge detection in the image processing hardware and transmittal.

4 ALGORITHM

4.1 Overview

The input to the algorithm is a stereo triple. The first step is to generate the vertical edge gradient for each image, which is accomplished in the image processing hardware as a convolution. The PC utilizes the edge gradients to start executing the remainder of the algorithm. Edge points in the left and right images are matched using the center camera to help eliminate false correspondences. For corresponding pixels, edge positions in the left and right images are estimated to subpixel resolutions and used to arrive at a subpixel disparity. We use the subpixel disparities to build an edge depth map. Figure 2 shows a flow chart of our algorithm.

4.2 Acquisition

The images are acquired simultaneously from all three cameras. Starting with an original 512 x 480 interlaced camera image, we immediately restrict our attention to a region of interest of size 512 x 160 located at the center.
of the original camera image. This portion of the camera image typically contains the information that we are most interested in, specifically the position of the other vehicles in the roadway and lane markings, whereas it is likely that the discarded portions of the original camera image contain only sky and empty road. Using a region of interest of this size also allows us to complete the requisite processing in a reasonable amount of time.

The timing offset between the even and odd fields in an interlaced camera can produce serrated images when there is rapid motion of the image in the camera’s image plane. Hence, the region of interest is further reduced by subsampling in the vertical direction, i.e., either the even or odd field is discarded, and an input image of size 512 x 80 results. These steps are summarized in Figure 3. Since we are interested in vertical edges, we are able sacrifice vertical resolution without affecting the accuracy of our depth estimates.

4.3 Feature extraction

To extract edge features, we apply a two-dimensional gradient operator to each of the images. This is accomplished in the image processing hardware as a convolution using the Sobel operator with a 3 x 3 neighborhood (gradient in the horizontal direction only):
We take the absolute value of the magnitude of the gradient, i.e., we ignore the direction of the gradient. We do this to increase the total number of edges in the field of view. The gradients are also scaled by a constant factor (typically less than one) in the image processing hardware to prevent saturation in the amplitude of the gradient. This is necessary to minimize ambiguities in edge localization for edges with a very strong edge response.

4.4 Localizing edges

Once the gradients are computed, the remainder of the algorithm is executed by the PC’s CPU. To facilitate comparison of edges among the images, we perform what is essentially a thinning operation on the gradient images by identifying the discrete positions of local maxima. Specifically, a pixel at position \((i, j)\) is classified as an edge if the gradient magnitude at that point, \(G(i, j)\), exceeds some threshold, \(T\), and the following condition holds:

\[
G(i - 1, j) \leq G(i, j) \geq G(i + 1, j)
\]

\(T\) is typically set to approximately 20% of full scale to eliminate spurious edges. Notice that in the case of saturation, \(G(i - 1, j) = G(i, j) = G(i + 1, j)\), we would observe a continuous string of pixels classified as edges. We localize the edges for the left, center and right images.

4.5 Edge correspondences

The localized edge points in the left and right images must be matched in order to extract depth information. Due to the special camera geometry of the test setup, this can be done in a simple, computationally efficient way. Consider a single row of pixels in the left edge map, \(E_l(1 \ldots i, j)\), and the corresponding row in the right edge map, \(E_r(1 \ldots i, j)\). We compare every edge point in \(E_l(1 \ldots i, j)\) with every candidate edge point in \(E_r(1 \ldots i, j)\). If a pair of edges correspond to the same edge, we expect to find an edge point along the corresponding row in center image, \(E_c(1 \ldots i, j)\), midway between the pair. For example, to check for correspondence between the edge points \(E_l(m, j)\) and \(E_r(n, j)\), we check for an edge point at \(E_c((m+n)/2, j)\).

Since the cameras are uncalibrated, it is possible that for corresponding edges there will not be an edge where we expect to find one in the center edge map. Hence, instead of specifying a single point in the center image, we specify a range. For the example above, we would allow \(E_c((m+n)/2 \pm \Delta, j)\), where \(\Delta\) is typically one or two pixels. Using a large value of \(\Delta\) would likely lead to an increase in the number of false correspondences.

4.6 Localizing edges to subpixel accuracy

For greater accuracy in depth computation, disparity can be computed to subpixel accuracy. A number of subpixel approaches have been evaluated. Quadratic interpolation was selected for its quality and ease of computation. Given three gradient amplitudes, quadratic interpolation is performed as shown in Figure 4 to arrive at a subpixel edge location.
4.7 Depth map and further analysis

From each subpixel disparity, the depth may be calculated directly from Equation (1). This produces a generic depth map for edges in the field of view. A histogram of the number of edge points versus disparity (or depth) can be constructed, and after some bin averaging, the first peak at high disparity can be taken as the closest object.

5 RESULTS

An on-road trial was conducted on a local highway to evaluate our approach and algorithm. Although the system has not yet been tested against ground truth data, the results are qualitatively reasonable.

The motion of the test vehicle caused noticeable difficulty in image acquisition. If the vertical motion of the cameras is rapid enough (easily the case with a large bump), the even and odd fields in the interlaced image will reflect the motion of the image in the image plane. To illustrate this effect, a zoomed region of an interlaced camera image is shown in Figure 5.

Starting with an original 512 x 480 interlaced camera image, we immediately restrict our attention to a region of interest of size 512 x 160 located at the center of the original camera image, as discussed previously. The region of interest is then subsampled in the vertical direction to reduce sensitivity to vertical camera motion. This results in an input image of size 512 x 80. One such input image taken from each of the right, center and...
left cameras produces an input triple like the one shown in Figure 6. This triple is taken as the input to the algorithm. Following the flow of our algorithm, after applying the gradient operator, the edges in this input triple are localized as shown in Figure 7.

Figure 6: Sample input triple.

Figure 7: Localized edges of input triple.

After correlation, we are able to compute the edge depth map, as viewed from the center camera, shown in Figure 8. Edges that are close to the test vehicle have lighter shading.

Figure 8: Depth map.
The configuration and associated performance benchmarks of our system are summarized in the tables below. While the disparity search range is nominally 0 - 400, the greatest measurable disparity is limited by the amount of overlap between the left and right images (more overlap with shorter baseline). The PC processing time will increase as the number of edges in the scene grows, due to the increased load in computing edge point correspondences. The PC processing time shown in the table reflects typical highway scenes, like the one shown in Figure 6.

<table>
<thead>
<tr>
<th>Function</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image acquisition</td>
<td>33</td>
</tr>
<tr>
<td>Preprocessing in special hardware</td>
<td>10</td>
</tr>
<tr>
<td>Processing in PC</td>
<td>19</td>
</tr>
</tbody>
</table>

Since the preprocessing in the image processing hardware and PC processing can together be accomplished within a frame time (33ms), the system can be made real-time by pipelining the image acquisition and processing steps.

Due to the dependence of our algorithm on a special camera geometry, the algorithm exhibits sensitivity to misalignments of the cameras. The algorithm is also susceptible to periodic edge patterns, which can generate a large number of false correspondences. To guarantee performance under a wider range of lighting conditions, auto iris lenses can be used for maximum possible contrast while working outdoors.

6 CONCLUSION AND FUTURE WORK

We described a PC based three-camera stereo vision constructed with off-the-shelf components, and presented an edge based, subpixel stereo algorithm capable of making accurate distance measurements using only a short baseline. The performance benchmarks from on-road trials indicate that real-time depth recovery is realizable using this system.

An obvious next step is to carefully investigate the robustness of the depth map estimation. In addition, some confidence metric for the distance estimates is highly desirable so that we may determine when they can be relied upon. Simple scoring schemes will not contribute significantly to the computational load. Finally, the generic depth map data will be applied to specific ITS applications.

7 ACKNOWLEDGMENTS

Thanks to Marcelo Mizuki for his assistance during the road trials, and to Tina Kapur for reviewing an early draft. Partial support for this research from the National Science Foundation under Grant No. MIP-9423221 is gratefully acknowledged.
8 REFERENCES


