Stereo-Vision Based Control of a Car using Fast Line-Segment Extraction

Brian McKinnon, John Anderson, and Jacky Baltes

Autonomous Agents Laboratory, Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada R3T2N2

Introduction

- Inexpensive equipment is an important element in domains where expendability is crucial (e.g. USAR); it also lends itself to a focus on strong AI based solutions rather than relying on specialized H/W
- Stereo vision is an important form of sensing for control: it provides depth information with low cost (two webcams). Processing stereo vision pushes the limits of today's embedded systems, however
- Most previous work on stereo matching is based on point features, and requires a guarantee of consistent camera calibration. Inexpensive setups, however, allow cameras to shake independently due to robot movement
- This work describes our efforts at using line-segment-based stereo matching, allowing stereo vision to be employed without assumptions of consistent camera calibration
- Robustness to camera jitter means knowledge of epipolar lines cannot be assumed, nor rectified stereo frames. Cheap equipment also renders color calibration unreliable. The result is a more robustly-applicable approach to stereo vision because of a lack of reliance on these traditional assumptions

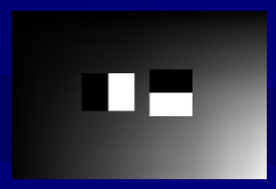
Related Work

- Line segment matching is considered to be more difficult than interest point mapping, because line end points tend to be very unstable due to noise. Matching is usually done by topology [Hollinghurst97] or color around the line segments [Bay06]
- [Zhang95] measured the amount of overlapping area in a calibration, to deal with unstable endpoints. He began with an unknown calibration, and estimated the essential matrix by using rotated points on an icosahedron. His approach does not describe the line-segment matching problem, and it appears that pre-matched line segments were used. This makes it difficult to predict the applicability of his approach

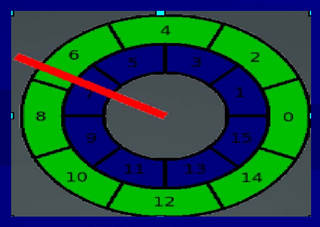
Line Segment Extraction

- Our approach involves two algorithms for line segment extraction and matching. Line segment extraction requires the following steps:

1. Integral image Generation. The value stored in each pixel in the integral image is the sum of all intensities in the area above and to the left of the pixel: the sum of any rectangular region can then be extracted by four table lookups and three additions/subtractions
2. Haar feature extraction. The integral image can be used to extract arbitrarily sized Haar features from an image [Viola]. Two vertical gradient features are extracted at kernel sizes of 4x4, 8x8, and 12x12 pixels, and are cached in a lookup-table
3. Gradient Thinning. Based on the Canny method with the addition of Haar feature gradients and an efficient data structure. This step extracts peak gradients by keeping only the strongest values along the gradient orientation. One of the cached kernel sizes from the previous step is selected for thinning, a threshold based on the magnitude of the gradient is applied to the pixels, and those exceeding the threshold activate an edge cell (EC)



Haar Features

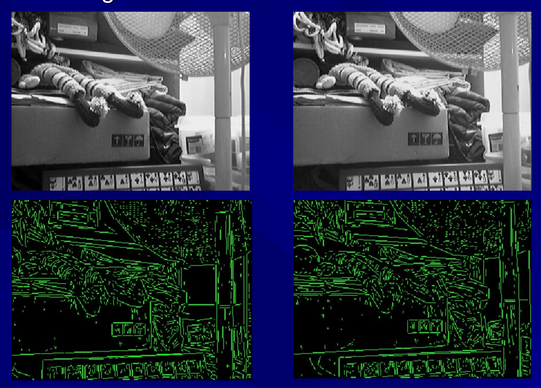


EC Data Structure

This is an efficient data structure for storing gradient orientations, allowing binary comparison. Each is a dual ring layer discretized into 8 cells, allowing π/4 per ring, with the layers offset by a distance of π/8. The gradient orientation of each pixel activates one cell in each layer of the pixel's EC. Using this activation, the EC can be tested for orientation with a binary logical operator. Thinning direction is determined by EC activation

Line Segment Extraction

4. Binary Segmentation. For each active pixel, the EC data structure is separated into 2 Split ECs (SECs): inner ring activation and the outer ring activation. 8-neighbor binary segmentation tests each of the SECs separately against neighbors, returning true if the neighbor EC has one cell or more overlap to the current SEC. Each pixel is ultimately a member of two thinned binary regions (line segments). We remove any segment contained in a longer segment, by allowing each pixel to cast a vote for the longer of those it belongs to. An optimal cut point is found for any partially overlapping line segments, and any segment less than a threshold length is discarded



Sample stereo pair, and the results of this line segment extraction algorithm

Line Segment Matching

- We use a dynamic programming solution. The dynamic programming table consists of points from the source segment (S1) making up the row header, and points from the matching line segment (S2) making up the column header). The match for 2 point feature vectors V1, V2 is:

$$M = \sum(sign(v1_i) == sign(v2_i)) * \min(v1_i, v2_i)/\max(v1_i, v2_i)$$

- Each insertion into the table involves selecting a previous match value, adding the current match value and incrementing the number of points contributing to the final result

Table Assumptions

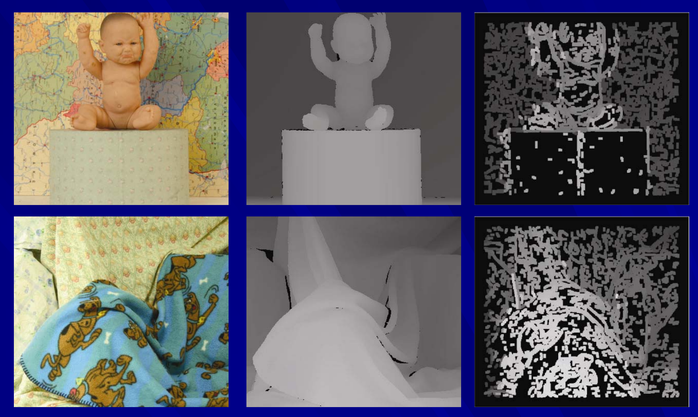
- The table requires two assumptions to prevent it from becoming degenerative:

1. *If a line segment diverges from the center line, it cannot converge back to the center line (prevents oscillation in the line match)*
2. *No point in the first line can match more than two points in the second (prevents the table from repeatedly using one good feature to compute all matches)*

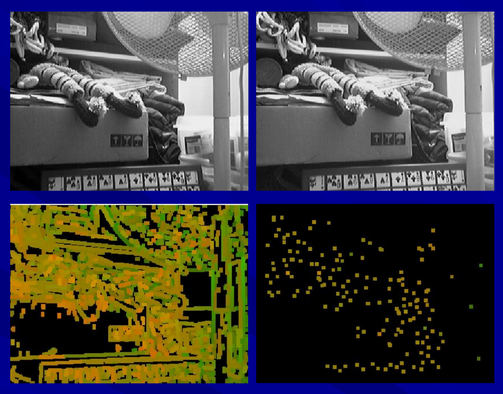
- The best value is checked in the last column for all rows with an index >= the length of line segment 2 (ls2), or for any entries in the point count table with a count = the size of line segment 1 (ls1)
- the best match value is recalculated using the disparity generated by the linear regression line. The matching process is repeated for all ls2 segments in the potential match set
- To reduce the number of line segments compared, we employ a few optimizations: we compare only line segments that have similar edge cell activations, and apply two different thresholds to the maximum search space. The first places a bounding rectangle extended by the search space around ls1, with a second rectangle around ls2: if the bounding rectangles overlap, the comparison continues. The second is point-by-point, with points outside the search space receiving a match value of 0

Evaluation

- We performed initial testing using the Middlebury 2005 and 2006 datasets. For these 27 image pairs, our approach has a matching accuracy of 71.8% +/- 11.5% for matches within 1 pixel of error. If only unoccluded pixels are considered (i.e. assuming occlusion detection was added to the algorithm), the accuracy is 78.8% +/- 10%
- Image, ground truth depth map, and depth map generated from stereo images by our line segment matching approach, for two sample Middlebury Images:



- Our goal is to apply this to robot control, so performance in unstructured domains using low quality image capture is more significant. While min/max disparity in images under these conditions in unknown, for the sake of execution time we assumed a +/- 80 pixel horizontal and 20 pixel vertical disparity in a 320x240 image
- Under these conditions, a depth map of the previous stereo pair (left) can be compared qualitatively with an implementation of SURF [Bay06] (right), which uses point-based matching:



- Line segments cover an average of 7% +/- 2.4% of the image: far more density than point match-based approaches
- Depth map of the previous stereo pair (left) under these conditions, and comparison to the with the results of We also illustrate the use of this algorithm on a sequence of images taken while controlling an autonomous vehicle

- Immediate future work in this area involves integrating this approach to vision with our prior USAR software (RoboCup 2004, 2005) form a robust platform for this application
- The extraction algorithm is very robust; future work will focus on merging line segments split by noise and extending end points to meet at junctions
- The matching method must be improved to address inaccuracies when dealing with repeated patterns, and occlusion boundaries