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

Brian McKinnon, Jacky Baltes, and John Anderson

Autonomous Agents Laboratory, University of Manitoba, Winnipeg, MB, Canada, R3T 2N2
jacky,andersj@cs.umanitoba.ca,
http://aalab.cs.umanitoba.ca

Abstract. This paper describes our work on applying stereo vision to the control of a car or car-like mobile robot, using cheap, low-quality cameras. Our approach is based on line segments, since those provide significant information about the environment, provide more depth information than point features, and are robust to image noise and colour variation. However, stereo matching with line segments is a difficult problem, due to poorly localized end points and perspective distortion. Our algorithm uses integral images and Haar features for line segment extraction. Dynamic programming is used in the line segment matching phase. The resulting line segments track accurately from one frame to the next, even in the presence of noise.

1 Introduction

This paper describes our approach to stereo vision processing for the control of a car-like mobile robot using low-quality cameras. The end goal of our work is a mobile platform that can be viewed as expendable for applications such as Urban Search and Rescue (USAR), necessitating inexpensive equipment. We are working on making stereo vision an inexpensive and robust alternative to the more expensive forms of sensing that are currently employed for robotic mapping and exploration (e.g. high-resolution laser scanners). A focus on inexpensive equipment also leads to the development of more broadly applicable intelligent software solutions, compared to relying on expensive hardware to deal with difficulties such as image noise.

The goal of all stereo vision processing is to produce dense depth maps that can be used to accurately reconstruct a 3-D environment. However, our target application imposes two additional constraints that make this problem more difficult. First, since the stereo system is intended for a mobile robot, the approach must support real-time depth map extraction. Second, since we are using inexpensive equipment, the approach must be robust to errors such as noise in the acquired images and the calibration of the cameras.

Most of the focus in stereo vision processing to date has been on point feature-based matching. The most notable feature extraction methods are corner detection [1, 2], SIFT [3], and SURF [4]. Most point feature based stereo matching relies on the epipolar constraint [5] to reduce the search space of the feature matcher to a 1-D line. The epipolar constraint can be applied using the Essential or Fundamental matrix, which can be determined using several methods [5].

The problem encountered in attempting to employ these mechanisms under conditions typical of USAR with inexpensive equipment is that cameras tend to shake as the robot moves. Unlike the heavy equipment used in events such as the Grand Challenge, inexpensive robots designed to be deployed in large groups in expendable settings must be based on far less expensive materials. In our USAR work, for example, we employ very cheap ($10) webcams mounted on 3mm aluminum sheet-metal tie, connected to a servo so that the stereo vision mechanism can be panned. Under these conditions, even small changes in the position of one camera relative to the second require the recalculation of the epipolar lines, which can be an expensive and error-prone task during the operation of the robot. While a more elaborate hardware setup would be another solution, forcing the work to be performed on simple hardware not only makes the solution more widely deployable, it demands a more robust and generally applicable solution that serves to advance the state of the art. In this case, solving this problem using an inexpensive vision system demands a more robust feature set that would be trackable over a larger 2-D search space.

These weaknesses led us to explore region-based stereo matching [6] to provide a more robust feature set. However, region segmentation proved to be a difficult task in complex scenes under real-time constraints. In addition, regions would not always form in a similar manner, making the hull signature matching approach we employed prone to errors. The most important piece of information gleaned from this work was that most of the information in a region was contained on the boundary between two regions. This intuitively led to a shift in our approach towards the extraction and matching of boundaries - in particular, line segments. Though several authors have proposed methods for line segment matching [7, 8], none could achieve the desired speed and accuracy needed for our mobile robotics applications.

The line segment-based approach taken in this paper emphasizes the dual requirements of real-time processing and robustness to camera jitter due to robot movement. The latter requirement means that knowledge of the epipolar lines in particular cannot be assumed, nor can we assume rectified stereo frames. In addition, we allow the off-the-shelf webcams in our implementation to freely modify the brightness and contrast of an image (typical of low-priced hardware), and so we also cannot assume colour calibration.

We describe two algorithms designed for these conditions, involving fast line segment extraction and two-frame line segment matching. The fast line segment extraction makes use of integral images to improve the performance of gradient extraction, as well as provide an edge representation suitable for binary image segmentation. Line segment matching is achieved using a dynamic programming algorithm, with integral images providing fast feature vector extraction.

We begin by introducing existing methods for line segment based stereo processing, and then describe our algorithms in detail. This is followed by some initial results that have been generated using the described line segment based method.

2 Related Work

This section describes existing line segment extraction and matching algorithms. The term *Line segment extraction* refers to the process of identifying boundaries between image regions, and grouping the points into a line segment feature. *Matching* is the process of identifying the new location of a previously observed feature given some change in the position of the cameras.

The extraction of line segments can be approached using either global or local methods [9]. Both methods rely on identifying boundary pixels using one of several methods of edge detection (for more information refer to [10]).

Most global line segment extraction algorithms are extensions of the generalized Hough transform [11]. In the generalized Hough transform each boundary pixel votes for candidate lines in the Hough space accumulator. Once all boundary pixels are processed, a search is done in the accumulator to find peaks that exceed a threshold number of votes. This Hough transform is used to extract infinite lines, but many methods have been presented [11, 12] to deal with the extraction of line segments with start and end points. The primary difficulty involved with global line segment extraction algorithms is that they are computationally expensive, and can generate imaginary lines formed by textured surfaces.

The simplest form of local line segment extraction uses chain coding [13]. Chain code values represent the location of the next pixel in the chain using either a 4 or 8 connected neighbourhood. The boundary is followed starting from an initial edge point (generally the top-leftmost point in the chain) and followed until the chain returns to the start. Noise can be filtered from the chain code using a median filter over the connection direction. The final chain code is then examined for line segments by finding runs of equal value neighbour connections. Local line segment methods are more sensitive to noise in the image, and so most recent work focuses on joining small segments into larger segments [11, 9].

Line segment matching is generally considered to be a more difficult problem than interest point matching. Although it seems simple to view line segments as simply a connected start and end point, the problem is that end points on line segments tend to be very unstable due to noise in the image [8]. The two primary features of a line segment that are used for matching are the colour information around the line segment, and the topology of line segments. Bay et al. [8] used colour information from points three pixels to the left and right of a line segment to generate histograms. The histograms from the two candidate lines were normalized, and the distance between them was calculated using the Euclidean distance between histogram bins in the defined colour space. The colour information produced soft matches [8], reducing the number of potential matches. By applying the sidedness constraint [8], the incorrect matches were filtered out to produce the final set of matched lines. The use of colour information was limited to situations where the capture devices produced very consistent colour output. In the more common situation, colour information varies greatly between the images [14] due to automatic brightness and contrast corrections differing between the capture devices.

If colour information is ignored, then matching must be based on the topology of the images. Hollinghurst [14] defined a set of rules for topological matching that includes initial matching and refinement operations. Using geometric constraints, initial matches are generated based on overlap, length ratio, orientation difference, and disparity limit. The initial matches are refined by applying constraints that either support or suppress a potential match. Matches are suppressed if they match multiple line segments, and if they violate the sidedness constraint. Support is generated by collinearity and connectivity, including parallelism, junctions at endpoints, and T-junctions. These constraints are applied to the initial matches until only supported matches remain.

To deal with the problem of unstable endpoints, Zhang [7] measured the amount of overlapping area in a given calibration. In his experiments, the extrinsic calibration for the two cameras was unknown, and instead the essential matrix was estimated by using rotated points on an icosahedron. He found that given a set of matched line segments, the essential matrix could be estimated as accurately as a calibrated trinocular system. The system did show that matched line segments could be used to generate a depth map of the image. One problem is that there is no discussion of the line segment matching problem, and it appears that pre-matched (possibly by a human operator) line segments were used. As a result, it is difficult to predict how line segment-based stereo vision will work in the presence of noise found in real-world images.

3 Implementation

This section describes our algorithms for line segment extraction and matching. For simplicity and grounding, the method described in this paper will be applied to 8-bit gray-scale stereo images of dimensions 320x240. The sample images used to describe our work are shown in Fig. 1.

3.1 Line Segment Extraction

Our algorithm for line segment extraction is divided into five steps:

1. Integral Image Generation
2. Haar Feature Extraction
3. Gradient Thinning
4. Binary Segmentation
5. Overlap Clean-up

The integral image is an efficient data structure for extracting large amounts of data from images in constant time. In an integral image, the value stored in each pixel is the sum of all intensities in the area above and to the left of

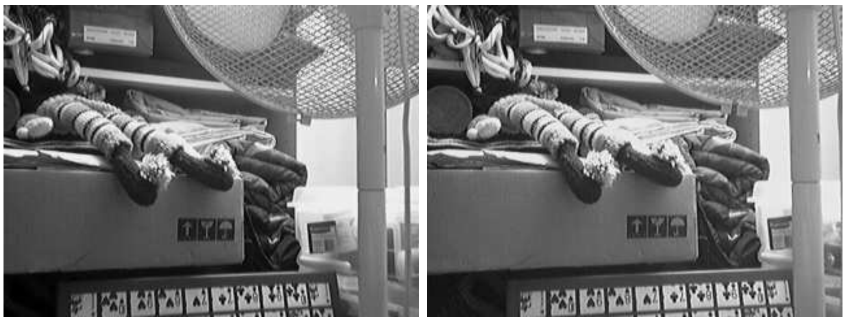

Fig. 1. The left and right input images used in the description of the algorithm.

that pixel. This means that the sum of any rectangular subregion inside the integral image can be extracted with only four table look-ups and three addition/subtraction operations.

The most practical application of integral images is to extract arbitrarily sized Haar features from an image [15]. While there are several Haar features that can be extracted from an image, this algorithm focuses on the vertical and horizontal gradient features. During the Haar feature extraction stage of the algorithm, the two gradient features are extracted at a kernel size of 4x4, 8x8, and 12x12 pixels. The values are cached into a look-up table to improve performance during the matching stage. These features then provide the input for both gradient thinning and dynamic program matching.

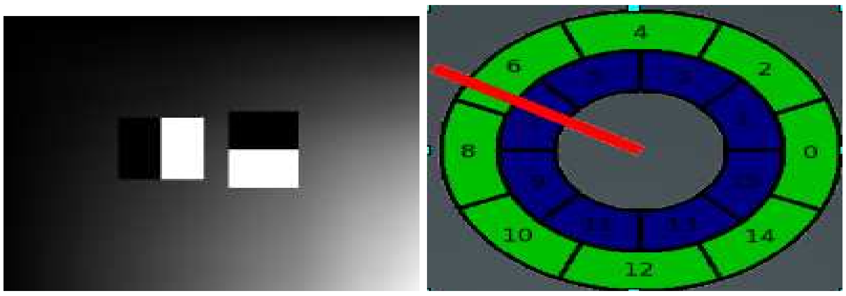

Fig. 2. The two main data structures used by the algorithm. Left, two Haar features overlayed over the integral image from Fig. 1. Right, the encoding of the gradient of a line using an edge cell.

The gradient thinning algorithm is based on the Canny method, with the addition of the Haar feature gradients and an efficient data structure. The goal of this method is to extract the peak gradients by keeping only the strongest values along the gradient orientation. From the cached Haar features, one kernel size is selected for thinning. A threshold based on the magnitude of the gradient is initially applied to the pixels. Pixels exceeding the defined threshold are used to activate an edge cell (EC).

The EC is an efficient data structure for storing gradient orientations, since it allows binary comparisons of the gradient orientation. Each EC is a dual layer ring discretized into 8 cells, allowing a resolution of $\pi/4$ per ring. The inner layer is offset from the outer layer by a distance of $\pi/8$, and the combined layers produce a $\pi/8$ resolution for the EC. The gradient orientation of each pixel activates one cell in each layer of the pixel's EC. Using this activation, the EC can be quickly tested for a horizontal, vertical, or diagonal orientation with a binary logic operation. ECs can be compared against each other using either an equal operator, which returns true if both cell layers overlap, or a similar operator, which returns true if at least one cell layer overlaps. The thinning direction is determined by the EC activation, and only the strongest gradient pixels along the thinning orientation are processed by the binary segmentation.

Binary segmentation is the core of the line segment extraction, and makes use of the EC data structure described above. For each active pixel, the EC is separated into two Split ECs (SEC) with the first containing the inner ring activation, and second containing the outer ring activation. The 8-neighbour binary segmentation then tests each of the two SECs separately against its neighbours. The binary test for the neighbour returns true if that neighbour EC is similar (at least one cell overlaps) to the current SEC. The final result of the segmentation is that each pixel is a member of two thinned binary regions, or line segments as they will be referred to from now on. The overlapped line segments are useful, but we generally prefer that each pixel only belong to a single line segment.

There are two cases of overlap that need to be cleaned up to produce the final result. The first clean-up step is the removal of any line segments contained largely within another longer line segment. This is achieved by simply having each pixel cast a vote for the longer of the two line segments for which it is a member. Any line segment with a number a votes below a threshold is discarded. The second clean-up step involves finding an optimal cut point for any partially overlapping line segments. This is achieved by finding the point in the overlapping area that maximizes the area of a triangle formed between the start of the first line segment, the cut point, and the end point of the second line segment. The overlapping line segments are then trimmed back to the cut point to produce the final line segment. Any line segments smaller than a defined threshold are discarded at the end of this stage.

Fig. 3 shows the output of our line extraction algorithm. The result are good line segments for stereo vision. However, more post processing could be performed to improve the line segments and further reduce noise. A few examples include merging line segments separated by noise, or extending end points to junctions. These features await future implementation.

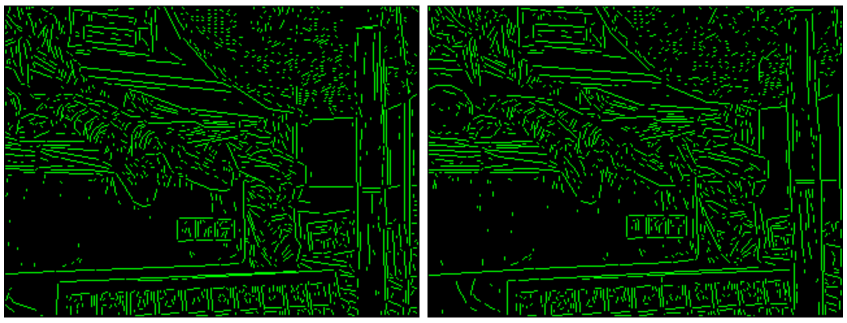

Fig. 3. The final line segments extracted by the described algorithm.

3.2 Dynamic Matching

The matching of line segments is a difficult task, since the end points are generally poorly localized and the segments can be broken into multiple pieces due to noise in the image. In addition, some line segments may not be completely visible in all images due to occlusion. To address these problems, a dynamic programming solution is applied to the line segment matching problem. The dynamic programming table consists of points from the source line segment, $ls1$, making up the row header, and the points from the matching line segment, $ls2$, making up the column header. The goal is to find the overlapping area that maximizes the match value between the points of the two line segments. To compare the points of two line segments, we return to the Haar features extracted earlier. The match value for two point feature vectors, $v1$ and $v2$, is calculated as:

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

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

The insertion of the match values into the dynamic programming table requires certain assumptions to prevent the table from becoming degenerative. The assumptions are defined algorithmically using the variables, x for the current column, y for the current row, Dp for the dynamic programming table, St for the match sum table, Ct for the point count table, and M for the current match value. The Dp value is generated from St/Ct and for simplicity any assignment $Dp[x][y] = Dp[x-1][y-1]$ is actually $St[x][y] = St[x-1][y-1] + M$ and $Ct[x][y] = Ct[x-1][y-1] + 1$. With this in mind, the assumptions used to generate the Dp table are:

1. If a line segment diverges from the center line, it cannot converge back to the center line. This prevents oscillation in the line match, and is enforced using the rules:

   – if $x = y$ then
     $Dp[x][y] = Dp[x-1][y-1]$
   – if $x > y$ then
     $Dp[x][y] = best(Dp[x-1][y-1], Dp[x-1][y])$
   – if $x < y$ then
     $Dp[x][y] = best(Dp[x-1][y-1], Dp[x][y-1])$
2. No point in the first line can match more than two points in the second line. This prevents the table from repeatedly using one good feature to compute all matches, and is enforced by defining the behaviour of the *best* function as:
   – if $Ct[x1][y1] > Ct[x2][y2]$ then
     $best(Dp[x1][y1], Dp[x2][y2]) = Dp[x2][y2]$
   – else if $Ct[x1][y1] < Ct[x2][y2]$ then
     $best(Dp[x1][y1], Dp[x2][y2]) = Dp[x1][y1]$
   – else
     $best(Dp[x1][y1], Dp[x2][y2]) = \max(Dp[x1][y1], Dp[x2][y2])$

The best match value is checked in the last column for all rows with an index greater than or equal to the length of $ls2$, or for any entries in Ct with a count equal to the size of $ls1$. Once a best match value is found, a back trace is done through the table to find the point matched that produced the best match. The position difference between the matched points in $ls1$ and $ls2$ is used to determine the disparity of the matched pixels. Linear regression is used to find a slope for the disparity for all points in $ls1$. The best match value is then recalculated using the disparity generated by the linear regression line, since the best match value should be based on the final disparity match. The matching process in repeated for all $ls2$ segments in the potential match set.

To reduce the number of line segments that are compared, a few simple optimizations are performed. The primary filtering method is to only compare line segments that have similar edge cell activations. A secondary filtering method is to apply two different thresholds to the maximum search space. The first threshold involves placing a bounding rectangle extended by the size of the search space around $ls1$, with a second rectangle placed around $ls2$. If the two bounding rectangles overlap, then the comparison continues. The second threshold is done on a point-by-point basis, with points outside the search space receiving a match value of zero. These simple optimizations greatly increase the speed of the algorithm, making real-time performance achievable.

4 Experiments

To evaluate these algorithms, we performed intial testing using the well-known Middlebury 2005 and 2006 datasets [16].

Using the 27 image pairs from the 2005 and 2006 Middlebury test set, the algorithm described above has a matching accuracy of $71.8\% +/- 11.5\%$ for matches within one pixel of error (See Fig. 4. If only unoccluded pixels are

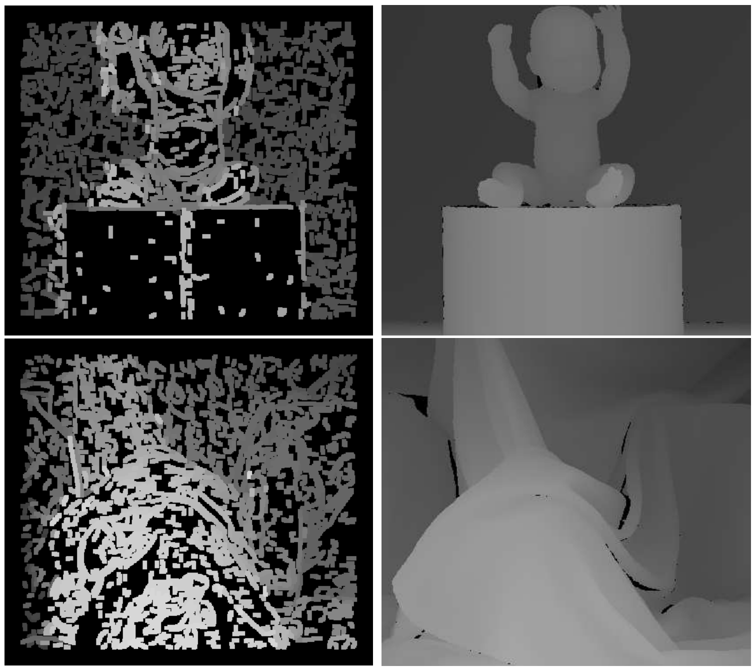

Fig. 4. Left, depth map generated by the line segment matching algorithm. Right, ground truth depth map from the Middlebury set. Depth maps have a small dilation applied to improve visibility

considered (assuming that some form of occlusion detection has been added to the algorithm) the accuracy rises to $78.8\% + / - 10.0\%$. In addition, the line segments cover an average of $7.0\% + / - 2.4\%$ of the image, providing far more density than point matches.

The most important results, however, are observed in the environments in which we intend to apply this work: unstructured robot control domains using low quality webcams. In such environments, the minimum and maximum disparity in the horizontal and vertical directions are unknown. For the sake of execution time, a $+/-$ 80 pixel horizontal and $+/-$ 20 pixel vertical disparity limit are applied to 320x240 webcam images. The results are compared qualitatively against a SURF [4] implementation, by examining the disparities generated at neighbouring points.

Fig. 5 illustrates a single stereo image under these conditions. To demonstrate performance on a sequence of images, we equipped an inexpensive robotic platform (a large-scale plastic RC car) with a stereo-vision system consisting of two webcams mounted on an aluminum bar, which was itself mounted on a servo to allow camera panning. The image sequence in Fig. 6 shows the output of our algorithm. The extracted line segments are robust and are particularly

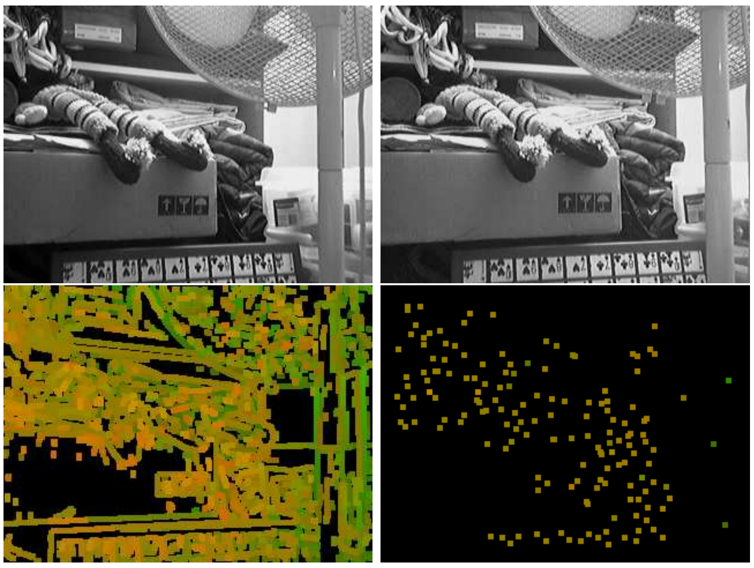

Fig. 5. Top, left and right stereo web cam images. Bottom, line segment matching disparity (left), SURF feature matching (right).

well-suited as matching features for stereo vision in this domain, since apart from a depth map these also incorporate the first stage of object recognition.

The image sequence shows that the extracted line information includes crucial areas of the image such as road markings or other vehicles.

5 Discussion

The method of line segment extraction described in this paper has been shown in demonstration to produce good quality matches, and has proven its suitability to real-world image sequences such as the ones from autonomous driving vehicles. By combining integral images and dynamic programming it has been shown that real-time processing rates are achievable using commodity hardware. The matching algorithm also shows that good quality matching can be achieved without the epipolar constraint. If exact calibrations are not required then it is possible to build stereo camera rigs with the ability to converge the cameras and focus on specific distances.

The immediate step in applying this work is to integrate it with our previous implementations of stereo vision-based USAR [17] to form a more robust platform for this application.

We are also currently performing a larger empirical comparison using a mobile platform under the real-world conditions depicted in Fig. 6. There are also several improvements to both the line segment extraction and matching algo-

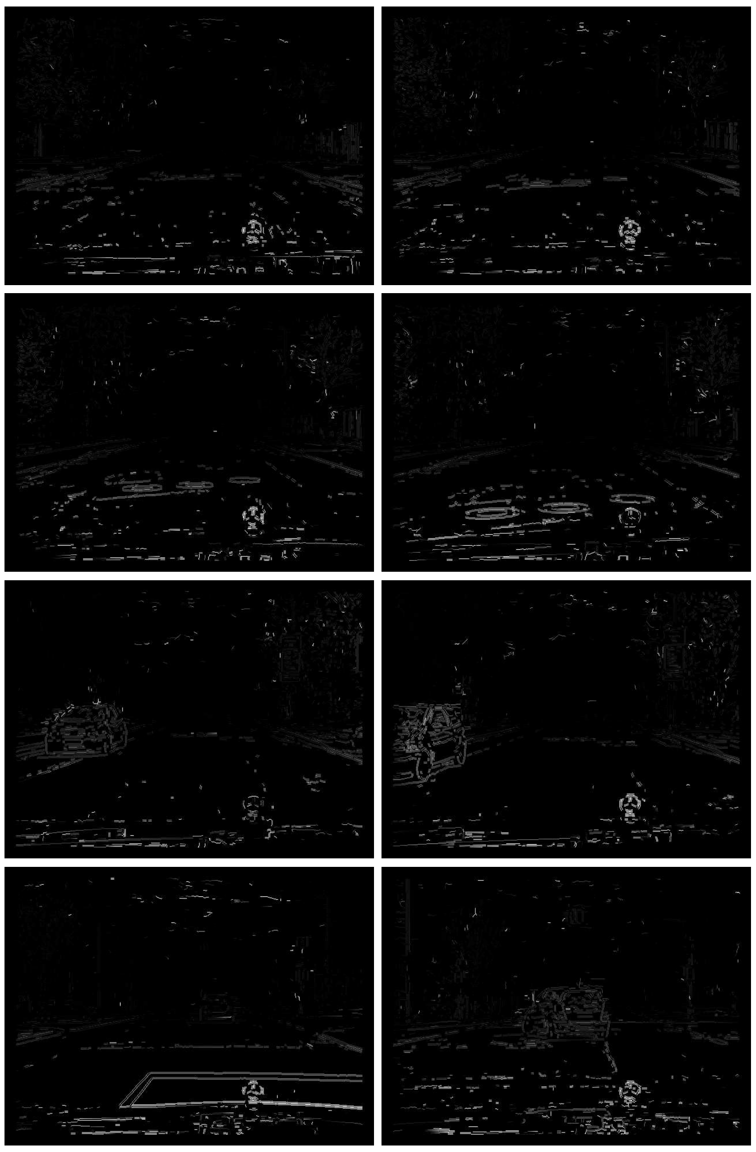

Fig. 6. Output of our algorithm on a sequence from an vehicle equipped with stereo vision cameras

rithms being proposed for future development. These include improved gradient extraction, using one of the methods evaluated by Heath et al [10]. The extraction algorithm is very robust, and therefore future development 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. Repeated patterns will require more of a global optimization over the matching values, rather than the greedy method currently used.

References

1. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition. (1994)
2. Tomasi, C., Kanade, T.: Detection and tracking of point features. In: Carnegie Mellon University Technical Report CMU-CS-91-132. (1991)
3. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision ICCV, Corfu. (1999) 1150–1157
4. Bay, H., Tuytelaars, T., Gool, L.V.: Surf: Speeded up robust features. In: Proceedings of the ninth European Conference on Computer Vision. (2006)
5. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2004)
6. McKinnon, B., Baltes, J., Anderson, J.: A region-based approach to stereo matching for usar. In: Proceedings of RoboCup-2005, Osaka (2005)
7. Zhang, Z.: Estimating motion and structure from correspondences of line segments between two perspective images. IEEE Trans. Pattern Analysis and Machine Intelligence **17** (1995) 1129–1139
8. Bay, H., Ferrari, V., Gool, L.V.: Wide-baseline stereo matching with line segments. In: IEEE Conference on Computer Vision and Pattern Recognition. Volume 1. (2005) 329–336
9. Jang, J.H., Hong, K.S.: Fast line segment grouping method for finding globally more favorable line segments. Pattern Recognition **35** (2002) 2235–2247
10. Heath, M., Sarkar, S., Sanocki, T., Bowyer, K.: A robust visual method for assessing the relative performance of edge-detection algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997) 1338–1359
11. Kim, E., Haseyama, M., Kitajima, H.: Fast line extraction from digital images using line segments. Systems and Computers in Japan **34** (2003)
12. Mirmehdi, M., Palmer, P.L., Kittler, J.: Robust line-segment extraction using genetic algorithms. In: 6th IEEE International Conference on Image Processing and Its Applications. (1997) 141–145
13. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Addison-Wesley Longman, Boston, MA, USA (2001)
14. Hollinghurst, N.J.: Uncalibrated stereo hand-eye coordination. PhD thesis, University of Cambridge (1997)
15. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Volume 1. (2001)
16. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision **47** (2002)
17. Baltes, J., Anderson, J., McKinnon, B., Schaerer, S.: The keystone fire brigade 2005. In: Proceedings of RoboCup-2005, Osaka (2005)