Dynamic Programming Stereo Vision Algorithm for Robotic Applications

Rafael C. González, Jose A. Cancelas, Juan C. Alvarez, Jose A. Fernández, Ignacio Alvarez

Area de Ingeniería de Sistemas y Automática

Universidad de Oviedo

Campus de Viesques, s/n, 33204 Gijón

corsino@hecate.etsiig.uniovi.es

Abstract

Autonomous navigation applications demand sensors with a low sample time to be able to increase speed. Stereo vision algorithms that produce a dense disparity map present a slow time response (half a minute per frame at high resolution). We have developed a new cost function for dynamic programming stereo algorithms, capable to deliver dense disparity maps for single, high-resolution scanlines at high speed (40 ms/line), even for wide disparity rages (>100). We have tested the algorithm with both synthetic and real image, and we have compared its practical performance with other dynamic programming algorithms. Our cost function is based on a weighted sum of the squared intensity errors. Weight factors are based on gradient values. The occlusion cost is not constant for the whole image, instead we modify it depending on gradient values of matched points. In this paper we present this cost function, and compare its performance for autonomous navigation applications with other dynamic programming solutions.

1 Introduction

Autonomous navigation applications demand low sample time sensors to be able to increase speed. Traditionally, stereo vision algorithms presented high computation times due to the complexity of the algorithms. Some real-time applications have been reported, which provides depth calculation at real-time, but they are usually based on highly specialized hardware, meaning a high price. We have tried to find an algorithm capable to provide reasonably good depth estimations, in real-time, and for large disparity ranges.

The key problem in stereo vision is to establish the correct set of correspondences among selected features in two or more images taken at the same time. Correctness is measured, in general, by a cost function that incorporates terms for similarity and smoothness constraints. The matching problem is therefore stated as a constrained minimization problem. Many different solutions have been proposed. They can be classified according to the characteristics of the optimization algorithm, the kind of image features selected, the way they are described and whether they are multiresolution or not. In addition, some algorithms integrate different kind of image descriptions and even they integrate different depth cues (vergence, depth from focus).

Among this variety of algorithms, those based in dynamic programming present nice characteristics for real-time applications. Dynamic programming is an efficient way to minimize a function of many discrete variables. Epipolar geometry also contributes to speed up the search for an optimum set of matches because it reduces the search space to one dimension. Joining both tools, and assuming ordering constraints, search for correct matches becomes a minimum cost path-finding problem. Even in this case, algorithm performance will depend on the set of image features selected, the way to establish the search and the shape of the cost function.

Cox et al. have proposed a maximum likelihood algorithm independent of selected matching primitives. In their experiments, they use individual pixel intensities for the matching process. Resulting algorithm provides a dense disparity map without requiring feature extraction, and avoids the adaptive windowing problem of area-based correlation methods. Resulting cost function is very simple but presents multiple global minima. To deal with this problem, they introduce cohesivity constraints by searching for the solution that minimizes the number of intra- and inter-scanlines disparity discontinuities.

Intille and Bobick proposed a different approach. They based their algorithm on a data structure called Disparity-Space Image, and used it to look for matches and

occlusions simultaneously. To avoid multiple global minima, they forced the correct path to pass trough some points called Ground Control Points (GCP). Those points are correct matches determined, by correlation, in a previous step to the dynamic programming algorithm. The cost of avoiding a GCP is infinitum. Other algorithms use cost functions based on correlation factors, but as out main interest was to obtain a fast algorithm, we looked for simple, fast cost functions, but still able to produce good disparity estimations.

Section 2 of the paper describes our cost function. First we introduce the general schema of DP algorithm, and describe simple cost functions previously published. Then we criticize those functions and propose a new cost function. Section 3 presents the results of our tests, comparing our cost function with others. Section 4 summarizes our conclusions.

2 Cost Function Definition.

2.1 Schema of a generic DP Algorithm

To apply a dynamic programming (dp) algorithm, a cost function to measure the quality of the solution has to be defined. Of course, for a given pair of images, optimum solution (minimum cost) will vary for different cost functions. Search space may be viewed as a square matrix, where each node Nij represents the cost of matching the first i features of the left image with the j first features of the right one. Assuming that the ordering constraint apply, the cost of adding a new node can be:

1) when feature i matches feature j: cost will be that of the match.
2) when feature i doesn't appear among the j first features of right image: i is a left occlusion feature.
3) when feature j doesn't appear among the i first features of left image: j is a right occlusion feature.

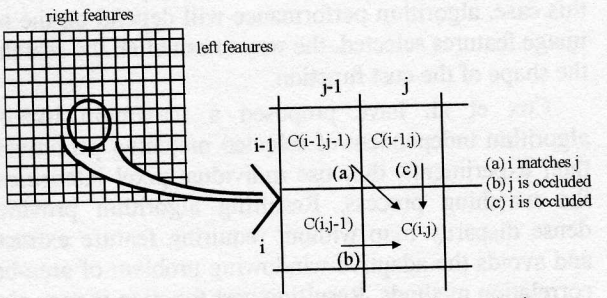

Figure 1: Possible paths and costs under ordering constraint assumption.

Those different cost values define the shape of the search space. Some authors have selected edges as primitives to be matched. Although edges are more reliable candidates for matches, this kind of algorithms will produce sparse depth estimations. On the other hand, matching pixel intensities is more sensitive to noise in the image.

Working with epipolar images, correct matches are restricted to lie in the same scanline. Resulting complexity of the algorithm is O(n m Δd), where n, m are image dimensions and Δd is the disparity range. In general, as Δd is small compared to image dimensions, is neglected and complexity is proportional to the number of pixels in the image. In real-time applications it is important to determine the proportionality factor because, it will affect directly the final execution time.

Cox et al. derives a cost function based on pixel intensity differences. They assume that pixel intensity follows a normal distribution of known variance. Cost of a match takes the form of a sum of squared errors. Occlusion cost is constant for the whole image, and its value depends on gray distribution variance and estimation of the number of occluded points in the image. They also apply a cohesivity constraint, implementing a search for the solution with fewer discontinuities in disparity values.

Intille and Bobick faced the problem in a different way. First they built the Disparity-Space Image, based of pixel intensity differences for a set of disparity values. To reduce the effect of noise, computation is done using a window centered in the pixel. As flat areas in the image produce flat areas in the search space, they introduce the concept of Ground Control Points, restricting valid solutions to go through these points. Ground Control Points are supposed to be correct matches and they are determined in a previous step by correlation.

2.2 An adaptive cost function

A cost function has two different parts to be defined:

a) The incremental cost of a match.
b) The cost of occluded regions.

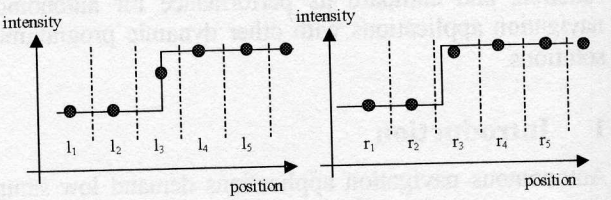

Figure 2: Quantization noise effect

In general, simple cost functions doesn't include any factor related to the confidence level for the match. Figure 2 presents a contour point (l_3) that is to be matched. Due to quantization noise, intensity difference between l_4 and r_3 is less than between l_3 and r_3, which should be the correct match. If we weight de intensity error inversely to gradient value, the match (l_3,r_3) will became less expensive than (l_4,r_3). To achieve this goal, we introduce use the measure of evidence defined by Scharstein. Evidence includes in a

single value similarity and confidence. Given a pixel on each image, with gradients $\vec{g}_l(x_l,y_l)$ and $\vec{g}_r(x_l+\delta_x,y_l+\delta_y)$, the evidence for that match is:

$$E(x_l,y_l,\vec{\delta})=\frac{|\vec{g}_l|+|\vec{g}_r|}{2}-\alpha|\vec{g}_l-\vec{g}_r|, \quad (1)$$

where $\vec{\delta}$ is the disparity and α is a parameter to balance both terms, the first one accounting for the confidence level, and the second one for similarity. We change this definition to obtain an inverse measure of evidence. We call this new measure Modified Evidence:

$$ME(x_l,y_l,\vec{\delta})=\frac{255-E(x_l,y_l,\vec{\delta})}{510}, \quad (2)$$

Figure 3 shows the contour plot of the modified evidence measurement. ME ranges from 0 for two points with absolute gradient values of 255, and 1 for opposite gradient values. Points in planar regions have a modified evidence of 0.5. A point with low modified evidence will produce always a low increase in the cost of the path going through that node. In fact, ME is only a function of the gradient value, so we will use the following representation:

$$ME(\vec{g}_l(r,i),\vec{g}_r(r,j))=\frac{255-\frac{(|\vec{g}_l|+|\vec{g}_r|)}{2}+|\vec{g}_l-\vec{g}_r|}{510}, \quad (3)$$

where r is the row been studied, and i,j are the position of candidates along the scan line. Due to this characteristic, ME can be tabulated easily.

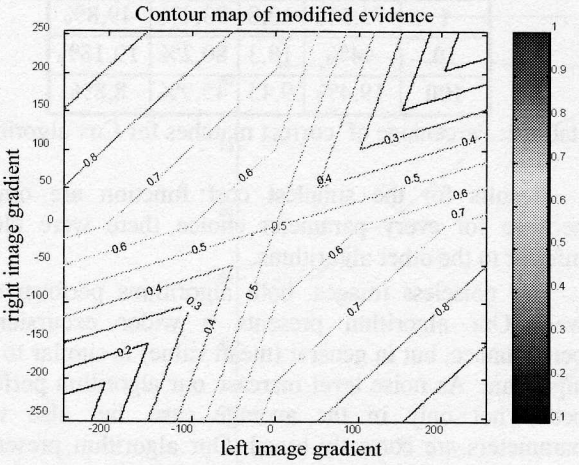

Figure 3: Contour map of modified evidence

The other key factor of the cost function is the cost for an occlusion. If this cost is set to 0, then a trivial solution occluding every point in the image will always be found. The cost of a known occluded point should be zero, because it is a good match. On the other hand the cost of wrong occlusion point should be infinitum because it wouldn't be allowed. If the cost is kept constant, we have refused to identify occluded points.

It is very difficult to identify occlusion points in an image. Several rules can be applied. Occlusion points are related to disparity discontinuities, and they appear in the contour of objects, so there is a relation between occlusions and gradient values. This relation is not straightforward because not every change in the intensity is caused by an object boundary. In fact, the majority of the occlusions follow the following patterns:

a) Left occlusions: Start inside an object (a relatively flat area of the image) and finish in an object boundary (not necessarily a point with high gradient).
b) Right occlusions: Start in the contour of an object and finish in relatively flat area of the image (inside another object).

Slanted surfaces introduce a different kind of occlusion points, those derived a continuously changing depth. These points are not real occlusions, but a discrete algorithm such as dynamic programming has to introduce an occlusion point to allow a jump in disparity.

In DP algorithms it is not possible to take a decision based on future ones, so it is impossible to know where a left occlusion starts or ends. In any case, and trying to introduce the knowledge we have at each moment about occlusions, we have modeled the cost of occlusions as a function of the modified evidence. If a candidate match has a low modified evidence value, it is probably a correct one, and therefore occlusion cost should be very high. On the other hand, if the modified evidence is small there is no information to determine if the match is correct or if it is an occlusion, apart from the intensity difference between both points. The form of our cost function for occluded points is:

$$OC(\vec{g}_l(r,i),\vec{g}_r(r,j))=K_1\left(1+K_2\,e^{\left(\frac{ME(\vec{g}_l,\vec{g}_r)}{K_3}\right)}\right), \quad (4)$$

figure 4 shows the shape of this function for different values. Occlusion cost can also be easily tabulated to reduce execution time. The overall cost function for our algorithm presents 3 parameters:

- K_1: represents the minimum cost of an occlusion. This should be slightly bigger than the maximum allowed difference in gray level. For example, for a maximum change in intensity of 10 levels, assuming gradient zero for both points, K_1 should be 101 (10^2+1).
- K_2: represents a general factor to increase occlusion cost.
- K_3: determines the rate of change in the variable part of occlusion cost.

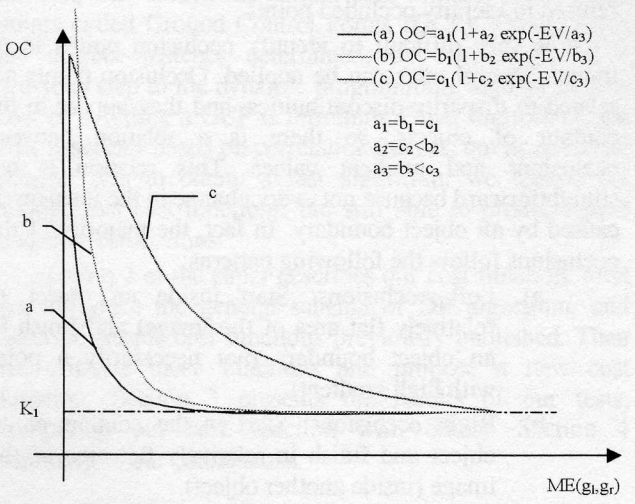

Figure 4: Shape of Occlusion Cost function.

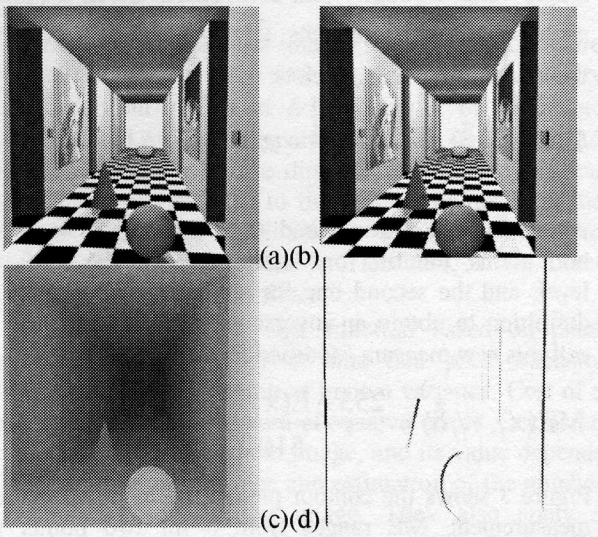

Figure 5: Fröhlinghaus synthetic stereo pair. (a) left image, (b) right image, (c) expanded (x21.25) left disparity map. (d) Left oclusion map.

3 Experimental Results

As our claim is basically related with the practical application of the algorithm, its validation depends on the experimental results achieved. We tested this algorithm extensively using both synthetic and real images. We measured its sensitivity to the parameters, to noise and to increasing disparity ranges. We also compared this algorithm with Cox algorithm, and with the simplest cost function, the sum of squared intensity differences, with constant occlusion cost. We didn't check Intille's algorithms because some aspects of its implementation were not clear, basically, the actual way to determine GCP. Cox algorithm implementation is available on the net, and only modified that implementation to reduce the amount of dynamic memory allocation, replacing dynamic allocation with static arrays.

The main difficulty in testing stereo algorithms is the lack of ground truth information. In general only printed images are available. So it is very usual to obtain similar disparity maps without been able to state which one is more accurate. Lately, there is an increasing interest in providing full information test images. Fröhlinghaus gives public access to a synthetic stereo including dense ground truth information and occlusion maps for both left and right images. In addition, the same stereo pair is available with different noise levels.

We have run the algorithm for 126 different combinations of the parameters, $K_1 \in [37,401]$ (6 to 20 in gray difference), $K_2 \in [1,30]$ and $K_3 \in [0.01,0.5]$, for every noise level. We also run Cox algorithms for 20 different parameter configurations. Tables 1 and 2 resume the results of the tests and figures 6 and 7 provides a graphical representation of the data.

noise	Mean	std	max.	min.
0	79 %	3,7	85,4%	68,9%
1	79 %	3,1	83,6%	69,1%
10	77 %	2,4	81%	69,8%
100	66.2 %	11,3	77,5%	36,3%

table 1: Percentage of correct matches for our algorithm.

noise	Mean	std	max.	min.
0	82,5%	0.82	84%	80,9%
1	73,3%	10,8	83,4%	49,8%
10	44%	18,3	80,2%	19,18%
100	19,4%	9,45	45,9%	8,8%

table 2: Percentage of correct matches for Cox algorithm.

Results for the simplest cost function are omitted because for every parameter choice there were clearly inferior to the other algorithms.

For noiseless images, both algorithms perform quite well. Our algorithm presents a wider excursion in performance, but in general (mean value) is similar to Cox algorithm. As noise level increase our algorithm performs better not only in the average case, but also when parameters are correctly tuned. Our algorithm presents a small excess of occlusion points. In noiseless images this excess ranges from 0.2% to a maximum of 3.8%, with a mean value of 1.57%. In the worst case (noise level 100), the minimum percentage is 0.34%, the maximum 16,76, and the mean value is 3,6%. For Cox algorithm, this value is bigger, ranging from 3,42% to 14,3% without noise and from 12,4% to 59,6%. Those values are calculated with respect to the number of occluded points in the image.

Our algorithm, including gradient calculation for both images, performs 2.46 times faster than Cox algorithm. In particular, average time for our algorithm was 923 ms, while average time for Cox was 2275 ms. Differences among different runs where always less than 50 ms. Noise level doesn't affect execution time. Average execution time for simplest cost function was 802 ms. All runs were made for a disparity range of 20, in a Pentium MMX at 200 Mhz, with 128Mb of RAM.

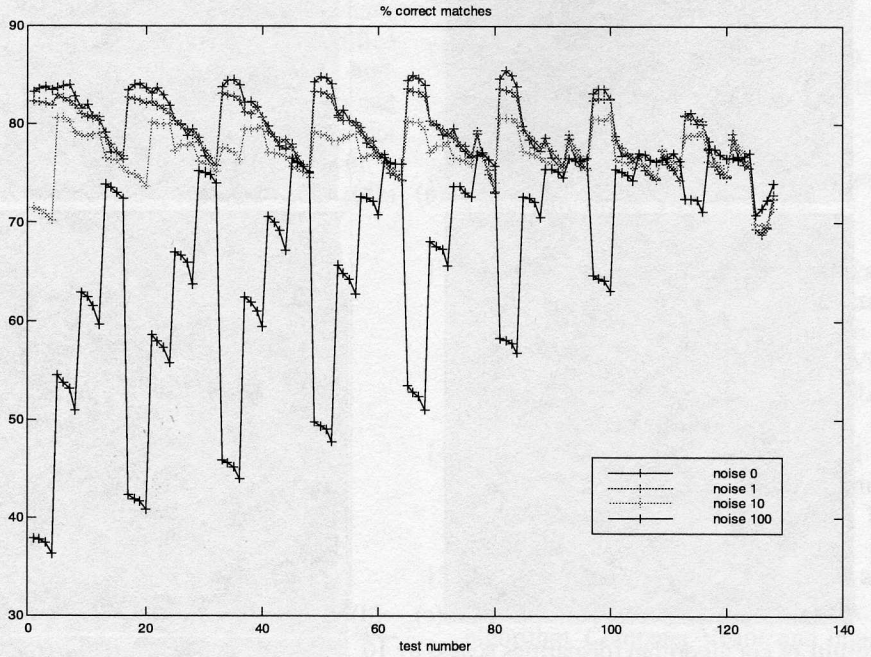

Figure 6 : Correct disparity values in our algorithm for differen noise levels and parameter configurarions

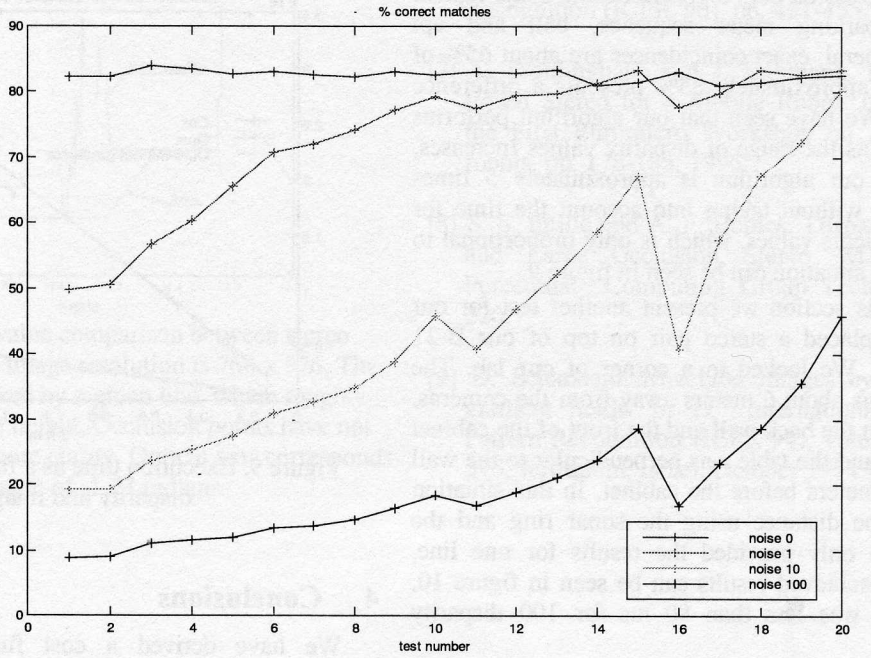

Figure 7: Correct disparity values in Cox algorithm for different noise levels and parameter configurations.

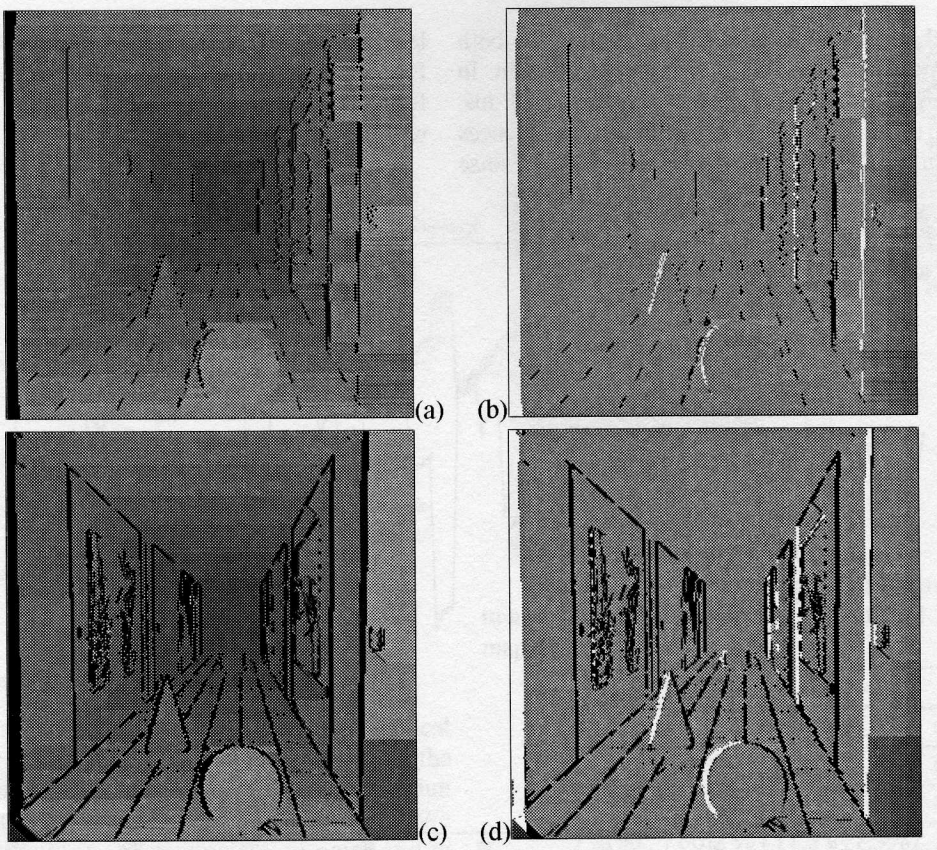

Figure 8: (a) Results of our algorithm (disparities scaled by 10, and with a bias of 5). (b) Error meaused: white means correct occlusion points, black means wrong occlusion points, 127 is for disparity error 0. Other erros are represented as 127±(10 error), according to error sign. (c) and (d) are the same as (a) and (b) for Cox algorithm.

We have also tested both algorithm using some typical real images: parking meter sequence, ball and epi sequence. In general, exact coincidences are about 65% of the image, and approximately 85% presents a difference bigger than 1. We have seen that our algorithm performs faster than Cox as the range of disparity values increases, this is because our algorithm is approximately 5 times faster than Cox without taking into account the time for calculating gradients values, which is only proportional to image size. This situation can be seen in figure 9.

To finish this section we present another test for our algorithm. We placed a stereo pair on top of our B-21 robot ('Trasgu'). We looked to a corner of our lab. The filing cabinet was about 6 meters away from the cameras, distance between the back wall and the front of the cabinet was 1.4 meters and the table was perpendicular to the wall and begun 1.2 meters before the cabinet. In this situation we measured the distance using the sonar ring and the stereo pair. We only provided the results for one line, comparation of achieved results can be seen in figure 10. Execution time was less than 80 ms for 100 disparity values.

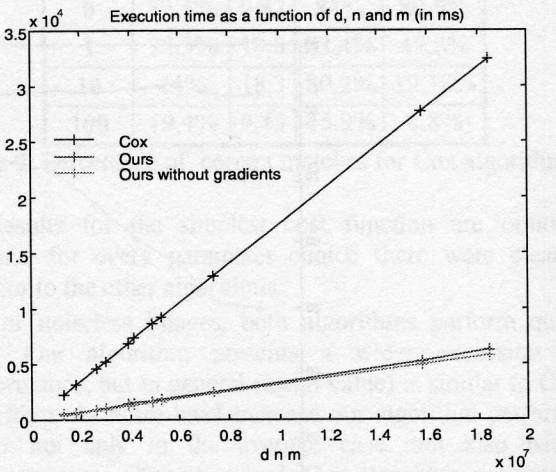

Figure 9: Execution time as a function of maximum disparity and image size

4 Conclusions

We have derived a cost function for a dynamic programming stereo algorithm, which presents fairly good disparity estimations for a wide range of images. Execution

time is low enough to apply the algorithm in autonomous navigation applications. For distant objects this algorithm may represent a valuable sensor for the robot as sonar decrease their performance. We have achieved these results by weighting the squared intensity difference between candidate matches. The weighting factor is an adaptation of Scharstein measure of evidence. We have inverted and normalized values, so that our modified evidences ranges from 0 for a candidate mach with high and equal gradient values, and 1 for points with high and opposite gradients. In addition occlusion penalty is not constant for the entire image. Candidate matches with a low modified evidence value are penalized with extra occlusion cost, as they are probably a good match.

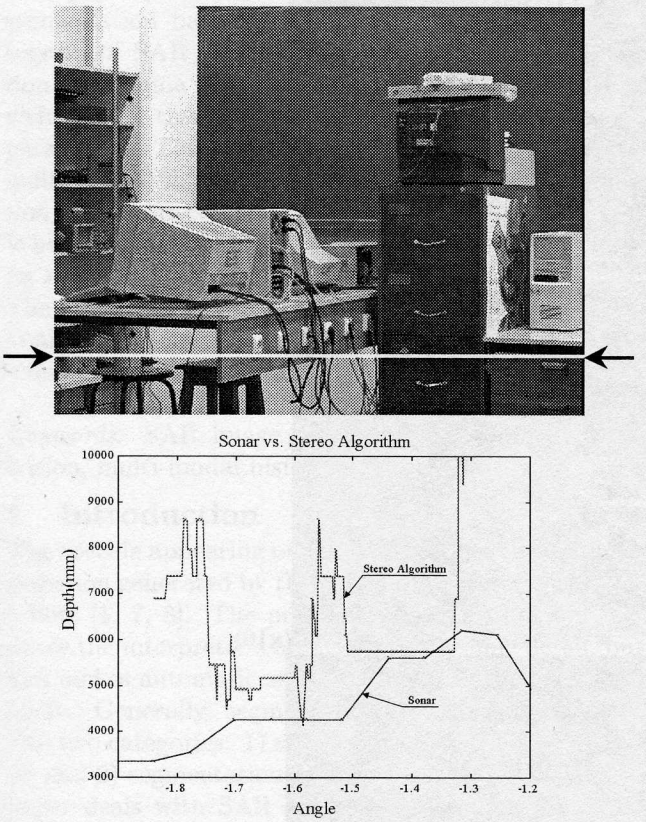

Figure 10: Depth value comparison between stereo algorithm and sonar. Image resolution is 768 x 576. The selected row is marked by a green line, which roughly corresponds to sonar height. Occlusion points have not been represented for more clarity. Optical axis corresponds to an angle of -1.57 radians.

We have extensively tested the resulting cost function. We have shown that parameters are easy to tune, and the resulting algorithm is quite robust in the presence of noise.

References

[1] R. Bolles, H. Baker, and M. Hannah. The JISCT stereo evaluation. In Proc. Image Understanding Workshop, pages 263-274, 1993

[2] R. E. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[3] I. J. Cox, S. Hingorani, B. M. Maggs, ans S. B. Rao. Stereo without disparity gradient smoothing: a Bayesian sensor fusion solution. In D. Hogg and R. Boyle, editors, British Machine Vision Conference, pages 337-346. Springer-Verlag, 1992

[4] I.J. Cox. A maximum likelihood N-camera stereo algorithm. In IEEE Conf. On Computer Vision and Pattern Recognition, pages 733-739, 1994.

[5] I. J. Cox and Sunita L. Hingorani and Satish B. Rao and Bruce M. Maggs. Maximum likelihood stereo algorithm. Computer Vision and Image Understanding: CVIU, 63(3), pp. 542-567, May 1996

[6] T. Fröhlinghaus and J. Buhmann. Regularizing Phase-Based Stereo. in: International Conference on Pattern Recognition (ICPR '96), pp. 451-455, Vienna, 1996.

[7] T. Fröhlinghaus and J. Buhmann. Real-Time Phase-Based Stereo for a Mobile Robot. in: Proceedings of the First Euromicro Workshop on Advanced Mobile robots. pp. 178-185, 1996.

[8] S. S. Intille and A. F. Bobick. Disparity-Space Images and Large Occlusion Stereo. M.I.T. Media Lab Perceptual Computing Group Technical Report No. 220.

[9] D. Scharstein. Matching images by comparing their gradient fields. In 12th International Conference on Pattern Recognition (ICPR'94), volume 1, pages 572-575, Jerusalem, Israel, October 1994

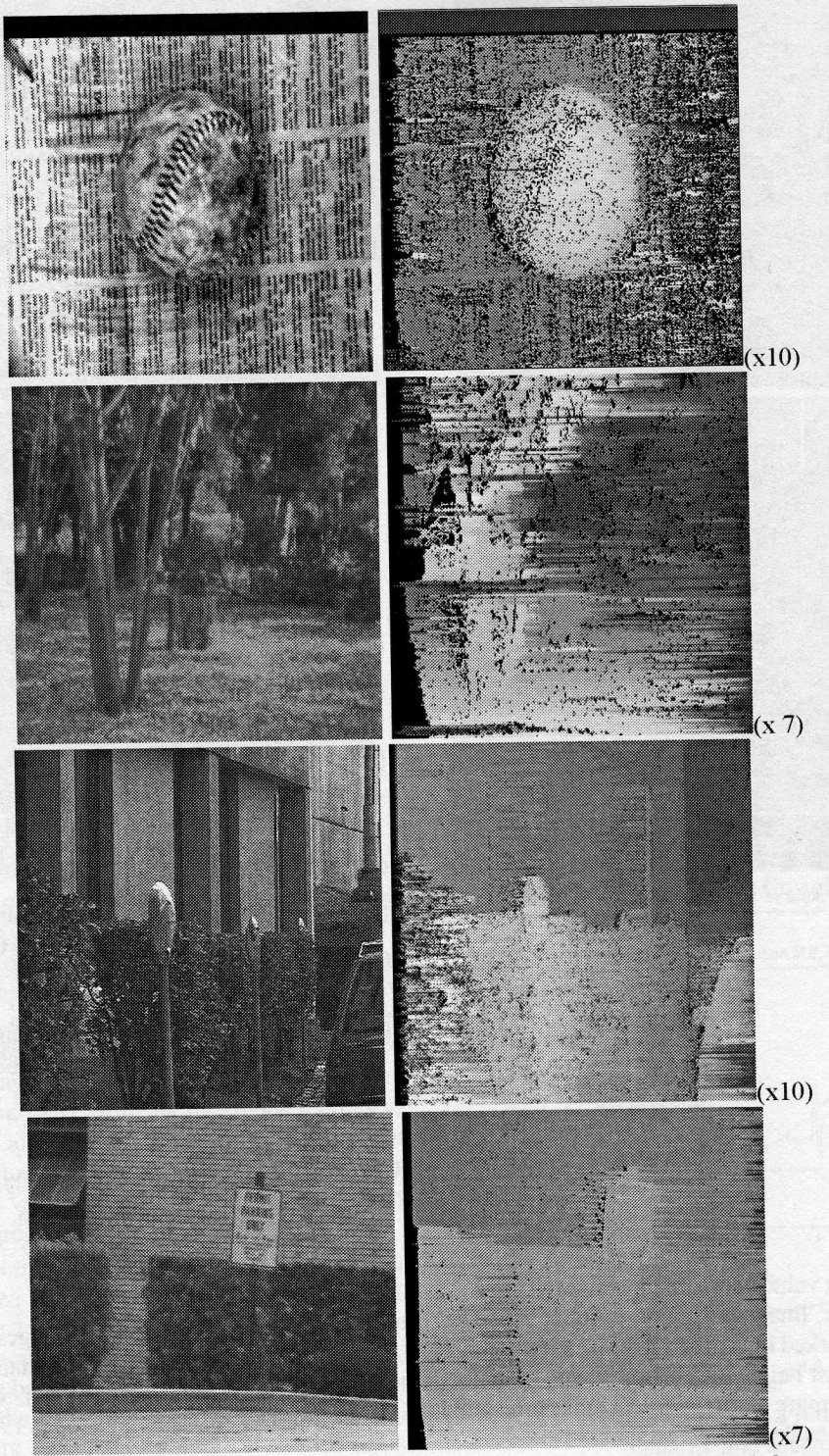

Figure 11: Some disparity images obtained with our stereo algorithm. The number at the left represent the scaling of image. All disparity values has an initial offset of 5 (0 means occlusion)