Temporally Coherent Stereo: Improving Performance Through Knowledge of Motion

Vladimir Tucakov, David G. Lowe
Dept. of Computer Science, University of British Columbia
Vancouver, B.C., Canada
{tucakov,lowe}@cs.ubc.ca

Abstract

This paper introduces the idea of temporally extending results of a stereo algorithm in order to improve the algorithm's performance. This approach anticipates the changes between two consecutive depth maps resulting from the motion of the cameras. Uncertainties in motion are accounted for by computation of an ambiguity area and a resulting disparity range for each pixel. The computation is used to verify and refine the anticipated values, rather than calculate them without prior knowledge. The paper compares the performance of the algorithm under different constraints on motion. Speedups of up to 400% are achieved without significant errors.

1 Introduction

Stereo vision is one of the most common and robust vision algorithms used in mobile robot navigation. It has been used for mapping, localization and obstacle avoidance [8], [10], [11]. While performance of stereo has increased with growing computer power, the technique is still limited by the high computational cost of the algorithm. One reason for the lack of performance is that stereo is performed fully in each iteration of the perceptual cycle. This is wasteful because the algorithm is not taking advantage of the coherence between depth maps obtained in consecutive time intervals. We propose a method of coherent stereo as a way of increasing the speed of the stereo algorithm by using the information from the previous iterations of the algorithm and the constraints on the robot motion.

The thrust of our approach is to anticipate the changes between two consecutive stereo pairs of images. The computation is used to verify and refine the anticipated values, rather than calculate them without prior knowledge. The main computational gain is achieved by reducing the search space needed to find the correct disparity for each pixel in the image.

The changes in the stereo depth information between two time intervals can be attributed to motion of the robot and motion of objects in the environment. Our algorithm assumes a static environment, however we suggest ways of extending the method to dynamic environments.

The following is an outline of the algorithm:

1. *Compute stereo over the full range of disparities, creating a depth map*
2. *The robot moves*
3. *Project the old depth map to get a new map that reflects the estimated robot motion*
4. *Use the new depth values to compute uncertainty ranges in disparity for search in the next iteration of the algorithm*
5. *Go to step 2*

At the beginning of the process stereo is computed fully because there is no prior knowledge about the scene. After the initial depth map is obtained the search space is reduced according to knowledge of the robot motion. The speedup of the algorithm depends on how accurately it is possible to determine the new depth model.

The accuracy of the new depth map depends on the error in measuring the new position of the robot. The most general constraint on robot motion is the amount it can possibly move in one time interval. If the time interval is short the change in what is observed may be very small.

The constraints on motion can be refined by knowledge of the general direction in which the robot is moving. For example, if it is known that the robot is moving forward, it should be expected that distances are getting smaller. Therefore, the search space is further reduced.

Finally, the amount of relative motion can be determined, quite accurately, by the use of odometry.

In this case the uncertainty in the robot's position comes from the odometry readings, which in general are quite accurate over short distances. It should be noted that dead-reckoning suffers from accumulating errors over long periods of time [2]. Our approach, however, does not require absolute odometry readings; Rather it uses the relative change in the readings between two closely-spaced time intervals.

This paper presents the approach taken in speeding up the stereo algorithm when the constraints on motion of the robot are known. The paper compares the increases of performance achieved depending on what is known about the robot's motion.

2 Related Work

Speeding up stereo algorithms has been an ongoing problem in computer vision. Much research has been done in developing specialized hardware that implements the stereo algorithm in parallel [7]. Our algorithm could exploit special hardware, but it is particularly well suited for implementation on sequential computers, which are currently much less expensive and easier to program.

Our approach is related to the coarse-to-fine stereo algorithm [15] which takes advantage of the results obtained at lower resolutions to predict the results at larger resolutions. This approach is related to our work because the coarse-to-fine algorithm is designed to take initial values for stereo search in order to speed up the computation. In the coarse-to-fine approach, information is propagated between resolutions, while our approach propagates information temporally.

The idea of temporally propagating knowledge about the scene in order to speed up the execution of an algorithm is not a novel one. The concept of temporal coherence is used in computer graphics to propagate the scene structure through time. For example, temporal and structural coherence is used in accelerating the calculation of animation sequences [5]. While the concepts used in graphics are similar, the overall goal is different. Computer graphics generates images given the scene structure, and the stereo algorithm produces depth maps given the images.

Recent work in view synthesis is also related to our work. View synthesis is concerned with generating realistic-looking images of a scene from a novel viewpoint, given one or more images of the scene. Work done by Scharstein [13] uses a stereo image pair to generate views from new viewpoints. Our approach is similar to this work because our algorithm needs the depth maps from new viewpoints. We share the problem of obtaining new disparity maps given sparse information. The difference, however, is in the application of the new depth maps. While view synthesis is concerned with reproducing the images from the new depth maps, we use the new depth maps to improve the performance of the stereo algorithms.

There has been much research in results of stereo algorithms over time [12], [1], [14]. The bulk of this research uses only results of the stereo algorithm, without altering its performance. To the best of our knowledge, the idea of temporally extending the stereo results in order to accelerate the algorithm is a novel one.

3 Approach

We have implemented a correlation-based stereo algorithm, following the approach taken by Fua [4]. The algorithm computes similarity scores for every pixel in the image by taking a fixed window in the left image and shifting it along the epipolar line in the right image. The scores are determined using the normalized mean-squared difference of gray levels:

$$s(i,j,d) = \frac{\sum((I_L - \bar{I_L}) - (I_R - \bar{I_R}))^2}{\sqrt{\sum(I_L - \bar{I_L})^2 \sum(I_R - \bar{I_R})^2}}$$

Where s is the score of the correlation. The summations are performed over all pixels in the correlation window. I_L and I_R are pixels from the left and right correlation window respectively, $\bar{I_L}$ and $\bar{I_R}$ are their average values over the correlation window, i and j are the coordinates of the correlation window in the left image, and d is the disparity at which the comparison is made.

The desired disparity at the given pixel is then the one that provides the minimum correlation score:

$$D(i,j) = d \in [d_{min}, d_{max}] \left| s(i,j,d) = \min_{x=d_{min}}^{d_{max}} s(i,j,x) \right.$$

Where D is the disparity map and d_{min}, d_{max} is the disparity search range.

The disparity map can be interpreted as the distance from the robot to the objects in the viewed scene, under the assumption of parallel camera image planes. Each disparity is inversely proportional to the distance of the object along the line of sight of each pixel [6]:

$$z(i,j) = b * \frac{f}{D(i,j)}$$

where b is the baseline distance between cameras and f is the focal length of the camera.

The performance of the stereo algorithm is improved by temporally extending the results of the stereo algorithm. The speedup is achieved by reducing the amount of searching done along the epipolar lines. In general stereo algorithms search for the best match within a fixed disparity range, $[d_{min}, d_{max}]$. Our algorithm accepts different disparity search ranges, $[d_{min}(i,j), d_{max}(i,j)]$, for each pixel in the image. The disparity ranges provided are less then or equal to the full disparity range. Therefore, the amount of searching by the algorithm is reduced.

The disparity ranges are computed from the previous disparity map and the constraints on the motion of the robot. The first step in computing the disparity ranges is to determine how much each pixel can move in the scene, given the constraints on the motion of the robot. The area of the image to which the pixel can move will be referred to as the *ambiguity area*, A.

Once the ambiguity areas are computed for each pixel, we have determined all points in the scene that the pixels can possibly see. By scanning the ambiguity area in the disparity map it is possible to determine the minimum and maximum disparity that the pixel may have in the next time interval. This disparity range is determined for each pixel, and provided to the next iteration of the stereo algorithm as input.

3.1 Ambiguity area

Image ambiguity area is part of the image that a pixel may move to given the constraints on motion of the camera. We are interested in determining the boundaries of this area because it contains the minimum and maximum disparity that the pixel will have in the next iteration of the algorithm.

In Figure 1 we present the pinhole model of one of the cameras in the stereo camera setup. The oval represents the lens of the camera. The x_i and z_i axis define the camera coordinate system. The x_i is on the image plane, the z_i points towards the scene. The curved line at the top of the figure represents the scene viewed by the camera. The focal length of the lens is labeled with the letter f. The point X on the image plane is the projection of a point X' in the scene. For simplicity, we will consider only a two dimensional motion of the camera. The motion of the camera is parameterized by the possible translations along the x and z axes $\delta x_{min}, \delta x_{max}$ and $\delta z_{min}, \delta z_{max}$, and rotation around the pinhole of the camera by $\delta\theta_{min}, \delta\theta_{max}$. Figure 2 shows the extreme possible positions of the camera after motion. The shaded rectangular area represents the possible positions of the camera relative to the current position of the camera. The new positions of the camera are chosen to reach the farthest visible point in the scene

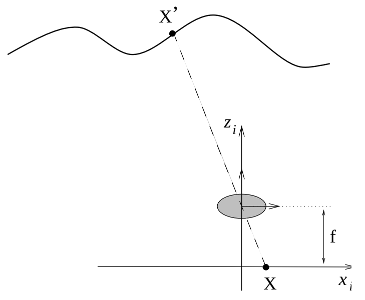


Figure 1: Pinhole model of the camera

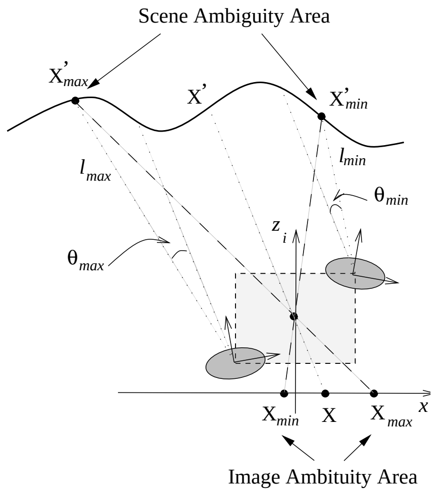


Figure 2: Computing the ambiguity area

given the constraints on its motion. We consider the point X in the image. We are interested in calculating the position of points X_{min} and X_{max} that define the image ambiguity area. The point X is the projection of a point X' in the scene. After the camera moves, a number of points in the scene can project back onto point X. We are interested in calculating the position of these points in the current image. This is done by considering the most extreme positions of the camera. The dotted lines, l_{min} and l_{max}, represent the line of sight from point X at the extreme position of the camera after motion. The intersection of the dotted lines and the scene are the left most point X'_{max} and the right most point X'_{min} that can project onto pixel X after the motion of the camera.

When the points X'_{min} and X'_{max} are projected back onto the image plane we obtain the points X_{min} and X_{max}. The range between these two points contains the pixel that point X will see after the camera has moved. Therefore the ambiguity area is $A_{min}(i) = X_{min} - X$ and $A_{max}(i) = X_{max} - X$.

The extreme lines of sight for the point X, l_{min}

and l_{max}, are a function the robot motion:

$$l = F(x, f)$$

$$l_{min} = F_{min}(x, f, \delta x_{min}, \delta z, \delta\theta_{min})$$

$$l_{max} = F_{max1}(x, f, \delta x_{max}, \delta z, \delta\theta_{max})$$

F, F_{min} and F_{max} can be easily determined using simple geometric transformations. Once the extreme lines of sight are determined, the positions X'_{min} and X'_{max} is a function of the scene structure. The structure of the scene is defined by the disparity map $D(i)$ in the one dimensional case considered in this example.

The position of points X'_min and X'_max is determined by searching for the point that either lies on the lines l_{min}, l_{max} or is closer to the image plane.

$$X'_{max} = \min_{x=X}^{max(x)} \Big| W(D(x), x) \ \ is\ left\ of\ \ l_{min}$$

$$X'_{min} = \max_{x=X}^{min(x)} \Big| W(D(x), x) \ \ is\ right\ of\ \ l_{min}$$

Where $W(D(x), x)$ is a function that determines the location of the point in the scene, given the disparity and the position of its projection, $min(x)$ and $max(x)$ are the coordinates of the left and right most pixel in the image.

This example considers two dimensions. The derivations are fully applicable in three dimensions and will not be discussed in detail.

3.2 Disparity range

Once the image ambiguity area is determined it is possible to determine the disparity ranges $d_{min}(i,j)$ and $d_{max}(i,j)$. This is done by searching the disparity map in the image ambiguity area and determining the minimum and maximum disparity values.

$$d_{min}(i,j) = \min D(x,y) \ \ | \ \ x, y \in A(i,j)$$

$$d_{max}(i,j) = \max D(x,y) \ \ | \ \ x, y \in A(i,j)$$

Here D is the disparity map, A is the ambiguity area for each pixel, and d_{min}, d_{max} represent the disparity search range. They are inversely proportional to the closest and farthest distance in the scene from the current viewpoint. In order to have the disparities represent the distances from the new point view we need to account for the translation the robot will do. The final disparity range is determined by accounting for the maximum amount of translation the camera can do towards and away from these points.

In some cases the image ambiguity area will be outside of the existing disparity map. This will occur when the robot rotates or moves backward. The pixels that may not see parts of the seen already observed have to search of the full disparity range.

If the ambiguity area includes invalid points then the disparity range will be set by the values of valid points. However, if the area is completely invalid then no computation will be done and that pixel will be invalid ahead of time.

4 Implementation

The algorithm implements a validity check suggested by Fua [4]. Validation is done by doing correlation twice by reversing the roles of two images. Valid matches are considered to be only the ones for which the disparities are equal.

An important modification to Fua's validation approach was to improve its performance for small disparity ranges. The problem with Fua's approach was that it found many invalid disparities to be valid when the disparity range was small. The reason for this is seen by an empirical observation that when matching is done in both directions the resulting match is often at the extreme points of the range. The validation process would often identify these disparities as valid while the the full algorithm would identify them as invalid.

The solution to this problem was to discard the pixels that pass Fua's validation procedure, but fall either on the minimum or maximum on the reduced disparity range. Another way of justifying this approach is that the expected disparity is more likely to be in the middle of the reduced disparity range if it is valid. In other words, if the model that produces the disparity range is correct and the range is sufficiently wide, than the found disparity should not fall at the extreme locations of the range.

The algorithm makes assumptions about the incoming images. It assumes that the epipolar lines are parallel with the image scan-lines and the epipolar lines are at the same y coordinate in the image. This is achieved by a calibration process described by Lenz and Tsai [9] . The position of the camera relative to each other, their focal lengths, and radial distortions are determined and used to correct the obtained images.

In order fairly judge the efficiency of the full algorithm the stereo algorithm was implemented recursively [3]. This means that an effort was made to reduce the amount of necessary repetition of computation. The reduction in processing was achieved by storing the correlation results of one pixel and reusing them to get the results for the neighboring pixels.

First, the correlation scores were computed for correlation windows that correspond to one scan line of the correlation window. These scores are noted as $s_{horiz}(i,j,d)$ where the i,j are the coordinate of the center pixel and d is the disparity at which the score is computed.

The horizontal scores are summed to produce the total correlation score of the window:

$$s(i,j,d) = \sum_{x=-w/2}^{w/2} s_{horiz}(i+x,j,d)$$

where w is the size of the window.

This summation was done only for the first row. On the next row the information from above window was used to compute the correlation score without having to do the full summation:

$$s(i,j,d) = s(i,j-1,d) - upper + lower$$

Figure 3 shows how computing the correlation score can be sped up by taking advantage of the results obtained in the previous scan line. The area that is subtracted is $upper = s_{horiz}(i,j-w/2-1,d)$. The area added is $lower = s_{horiz}(i,j+w/2,d)$.

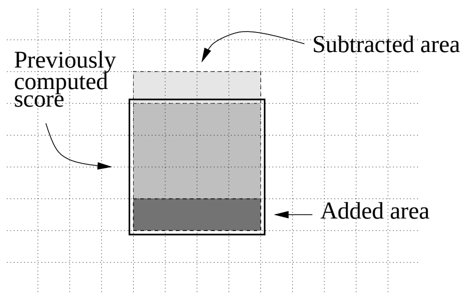


Figure 3: Recursive correlation

Figure 3 illustrates only the recursion in the vertical direction. The recursion is also done horizontally such that it takes advantage of the scores already computed for on the same scan line.

The recursive implementation gives an advantage to the full execution of the algorithm. The advantage comes from the fact that all the necessary information for recursion is available. In the case of the coherent algorithm the scores for the neighboring pixels may not be available. The lack of information is due to the difference in the disparity search ranges of the pixels. The coherent algorithm is therefore disadvantaged in two ways: first, it needs to do non recursive computations when the information is not available, and second, it has to check what information is available before any computation is done.

5 Expected results

Assessing the performance of the full algorithm is straight forward. The speed at which the full algorithm executes depends on the resolution of the image, the size of the correlation mask and, most importantly, the size of the fixed disparity range. The execution time of the full algorithm is linear to the number of pixels and disparity range and constant with respect to the size of the correlation mask (due to recursion).

The performance of the temporally coherent stereo algorithm on the other hand is quite complex. The complexity arises from the fact that the performance depends on the additional information available to the algorithm.

The first factor is the knowledge of motion. If the motion is known, then it is possible to accurately estimate the disparity search ranges. If the disparity ranges are small then the stereo search will be done faster.

The second factor is the structure of the scene. If the scene consists mostly of flat surfaces, then the predicted disparity ranges will be around the previous values. If, on the other hand, the scene has many discontinuities, then the search at the discontinuities will have to include both ranges which may be far apart.

The third factor is the number of valid and invalid pixels in the image. The algorithm processes only parts of the image that may move to a valid part of the image. If the image has large areas of invalid pixels and the motion is well known or small, then it is possible to determine areas of the image that do not need to be processed. This means that images that lead to large numbers of invalid disparities will execute faster with the coherent stereo algorithm.

Another important point is that even if the disparity ranges can be determined precisely, the program can still perform inefficiently. The inefficiency is due to the recursive nature of the algorithm. In other words, if one correlation can not benefit from the previously done work, the computation is done inefficiently. Scattered invalid points and small areas of different disparities can cause this effect.

Finally, close objects cause greater change in the disparity values when the robot moves towards or away from them. Therefore the algorithm will have poorer performance when objects are in close proximity.

The performance of the stereo algorithm, when given the disparity ranges, is one part of the computational cost of the whole algorithm. The other part is determining the disparity ranges. The time required to compute the disparity ranges depends mainly on the amount of robot motion. In general the less the knowledge of motion is constrained the longer it takes to compute the disparity ranges. Loose constraints on motion mean that larger ambiguity areas, which result in more searching for the disparity range.

6 Experimental results

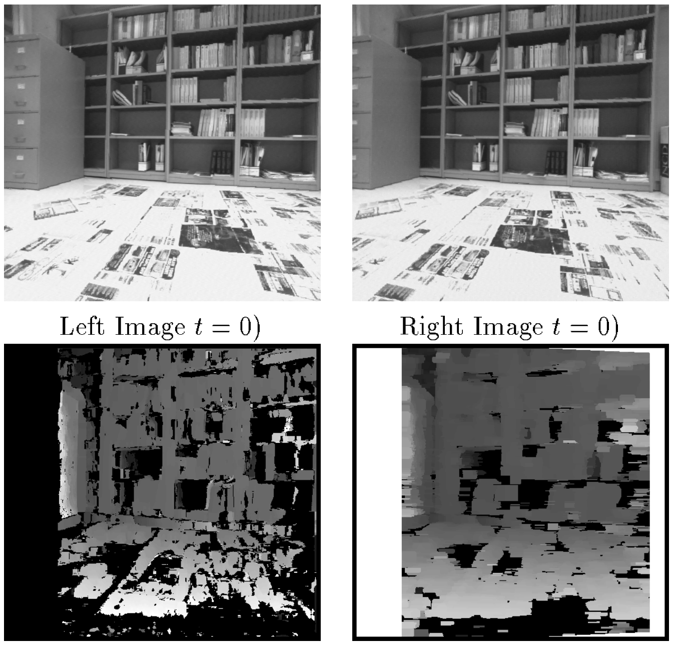
The execution time of our algorithm was compared to the execution time of the full algorithm in order to get a measure of the relative speedup. Both algorithms were provided with a sequence of images obtained from a mobile robot (Spinoza at Laboratory for Computational Intelligence at the University of British Columbia). The robot has two black and white cameras mounted on it. A sequence of 512×480 gray-scale images was obtained while the robot was moving through the laboratory. The robot was programmed to translate forward 5cm, capture a set of stereo images, rotate 3 degrees, and capture another set of stereo images. The robot repeated this motion twenty times.

The amount of information about the motion of the robot was broken down into three types;

- Type A: robot moves ± 6 cm and rotates $\pm 3.5^\circ$
- Type B: robot moves $[0..6] cm$ or rotates $[0..3.5]^\circ$
- Type C: robot moves 5 ± 1 cm or rotates $3^\circ \pm .5$

Figure 6 presents the information at the beginning of the sequence. The images shown are the views form the left and right camera. The disparity maps shown are the result of processing with the full algorithm. The shades of gray represent the valid disparities. The brighter shades of gray represent points in the scene that are closer to the viewer. The black areas of the image represent invalid points. The image labeled with *maximum disparity range* is the upper bound on the the disparity range for all pixels, given that the robot has rotated to the right anywhere between 0 and 3.5°. The shades of gray represent the upper bound of the disparity search range and black pixels represent points that are believed to be invalid. The white areas of the image represent the lack of information from the previous image. The right part of the image therefore has a white vertical strip, because the robot had moved to the right.

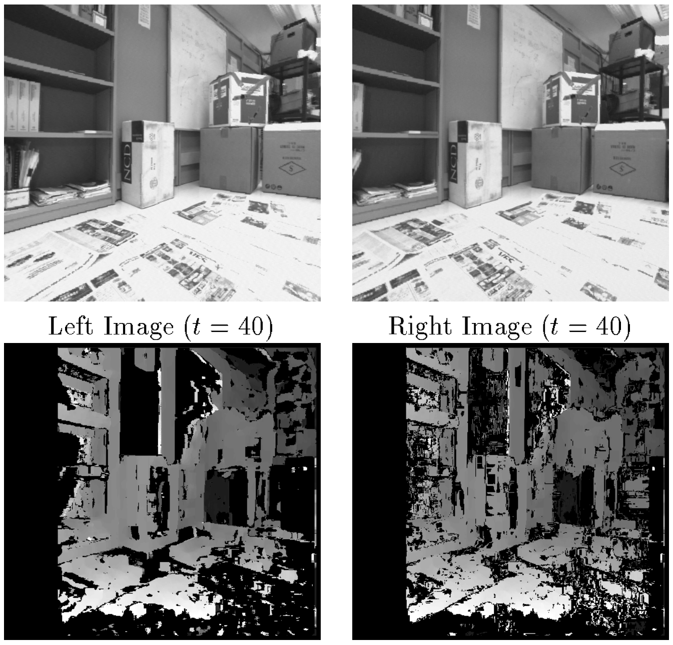
Figure 6 displays the last stereo pair of images in the sequence and the results obtained both by the full algorithm as well as the result of the coherent stereo algorithm. The result of the coherent stereo algorithm was obtained given that the general direction of robot motion is known.



Left Image $t = 0$) Right Image $t = 0$)

Optimal disparity map ($t = 0$) Maximum disparity image ($t = 0$)

Figure 4: Processing done when the robot turns to the right



Left Image ($t = 40$) Right Image ($t = 40$)

Optimal disparity map ($t = 40$) Temporally coherent stereo ($t = 40$)

Figure 5: Processing done at the end of the robot motion

constraint type	speedup (%)	valid matches (%)	average error (pixels)	error over 1 (%)
A	121	92.3	0.157	0.50
B	342	91.6	0.693	0.85
C	414	88.9	0.622	0.13

Table 1: Comparison of performance between the full and coherent stereo algorithm

The performance of the algorithm is analyzed by a number of criteria presented in Table 6. The *speedup* is calculated as the ratio of the CPU time used by the full algorithm over the CPU time used by the coherent stereo algorithm. The time spent in computing the disparity ranges is included in the time of the coherent algorithm. The CPU time spent on calibrating images is not considered for either algorithm. The column *valid matches* represents the percentage of the valid disparities correctly identified by the coherent stereo algorithm. The *average error* column presents the average difference between the disparity values found by the full algorithm and values found by the coherent algorithm. The *error over 1* column presents the percentage of pixels that are different from the correct result by more than 1.

7 Discussion and Conclusion

It can be seen from the experimental data, that the temporally coherent algorithm can decrease the processing time needed to produce acceptable results. The results suggest that as the amount of knowledge about the robot motion increases, so does the speed of the algorithm. It should be noted that the significant jump in the performance can be observed when the general direction of the camera motion is specified. This is important because the accurate odometry readings may not be available on all robots, but it is likely that the robot knows that it is moving in a particular direction.

The accuracy of the depth map can be as good or better when using the temporally coherent algorithm. We have presented the number of matches of valid disparities between the full and coherent algorithm. The number of matched valid disparities is above 85%. The coherent algorithm does find pixels to be valid even though the full algorithm finds them invalid. The number of pixels found to be valid only by the coherent algorithm are in the range of 10 to 15 % of all pixels, for more constrained motion. The explanation for the this phenomenon is that the results of stereo are temporally extended when the search range is limited. Therefore the algorithm is still able to identify the disparity as valid. If the disparity range was increased the algorithm would find the disparity invalid.

In the case when the motion of the camera is known, the disparity ranges are reduced to less than 10% of the full disparity range. This would lead us to believe that the algorithm should run 1000% faster. The speedup however is only in the range of 400%. A part of the reason for this unexpected performance is the time spent producing the disparity ranges. However, more important is the fact that the algorithm is trying to take advantage of recursion. The coherent algorithm needs to check if the necessary information is available. By doing this it executes many conditional jumps which are inherently expensive on sequential computers.

The disparity maps presented in this paper have on average 60% of all pixels valid. This is quite high considering that no interpolation was done. This was done on purpose in order to force the coherent algorithm to perform computation. Greater speedups are possible if the obtained disparity maps are very sparse. In this case the algorithm may choose not do process disparities that are believed to be invalid in the next iteration. In this case the speedups can go as high as 1000%. The problem with sparse depth maps is that there must be a mechanism for introducing valid points in the regions where invalid disparities are expected. Otherwise the whole image could possibly turn invalid. As our future work we propose two methods as a solution to this problem: statistical verifying of results and/or use of additional knowledge.

The statistical verification means selecting a number of random pixels and processing them over the full range of disparities. The obtained results are then compared with results of the coherent stereo. If the results are different then that part of the image can be processed with a larger disparity range. The statistical verification can be used for introducing valid points to an area of the image that was previously invalid. It could also be used for detecting dynamic objects in the environment.

The second approach to solving the problem of introducing valid points can be solved by using additional knowledge such as the image content. For example, it is well known that correlation-based stereo algorithms perform poorly on texture-less surfaces. Therefore, the checks for texture in parts of the image where it is necessary can help in deciding on whether computation should be done or not.

Additional information can also be useful depending on the task of the robot. For example, a well calibrated robot can determine disparity ranges that correspond to a particular part of the environment. The floor would be ideal to ignore, given that there are disparities that correspond to points below the floor. On the other hand, the robot may be particularly interested in holes in the ground. In that case the disparities should be tuned to find points below the floor level.

References

[1] Y. Aloimonos and J.-Y. Herve. Correspondence stereo and motion; planar surfaces. *IEEE Trans. Patt. Anal. Mach. Intel.*, 12:504–510, 1990.

[2] T. Dahlin and D. Krantz. Low-cost, medium-accuracy land navigation system. *Sensors*, February 1988.

[3] R. Deriche. Fast algorithms for low-level vision. *IEEE-PAMI*, 12(1):78–87, 1990.

[4] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. In *Machine Vision and Applications*, pages 35–49, 1993.

[5] E. Groller and W. Purgathofer. Using temporal and spatial coherence for accelerating the calculation of animation sequences. In Werner Purgathofer, editor, *Eurographics '91*, pages 103–113. North-Holland, September 1991.

[6] B. K. P. Horn. *Robot Vision.* MIT Press/McGraw-Hill, Cambridge, MA, 1986.

[7] A. Kazuo O. Kano H. Kanade, T. Yoshida and M. Tanaka. A stereo machine for video-rate dense depth mapping and its new applications. In *Proceedings CVPR '96*, pages 196–202. IEEE, San Francisco, CA, June 18-20, 1996.

[8] E. Kreigman, D.J. Triendl and T.O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Transactions on Robotics and Automation*, pages 792–803, June 18-20 1989.

[9] R. K. Lenz and R. Y. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3-d machine vision metrology. 10:713–720, 1988.

[10] H.P. Moravec. The stanford cart and the cmu rover. In *Proceedings of IEEE*, 1983.

[11] T. Nakamura and M. Asada. Stereo sketch: Stereo vision-based target reaching behavior acquisition with occlusion detection and avoidance. In *Proceedings CVPR '96*, pages 1314–1319. IEEE, San Francisco, CA, June 18-20, 1996.

[12] W. Richards. Structure from stereo and motion. *J. Opt. Soc. Am.*, 2:343–349, February 1985.

[13] D. Scharstein. Stereo vision for view synthesis. In *Proceedings CVPR '96*, pages 852–857. IEEE, San Francisco, CA, June 18-20, 1996.

[14] A.M. Waxman and J.H. Duncan. Binocular image flows: Steps towards stereo-motion fusion. *IEEE Trans. Patt. Anal. Mach. Intel.*, 8:715–729, Nov. 1986.

[15] L. R. Williams and P. Anandan. A coarse-to-fine control strategy for stereo and motion on a mesh-connected computer. In *CVPR-86*, 1986.