Motion Planning for a Legged Vehicle Based on Optical Sensor Information

Richard Bade[1], André Herms[1], and Thomas Ihme[2]

[1] University of Magdeburg, Unipl. 2, 39106 Magdeburg, Germany
{ribade,aherms}@ivs.cs.uni-magdeburg.de

[2] University of Applied Sciences Mannheim, Windeckstr. 110, 68163 Mannheim, Germany t.ihme@fh-mannheim.de

Summary. This paper describes the motion planning for a walking robot based on environment information. The planning algorithm is based on random sampling. The environment information are generated by a stereo vision algorithm that has been modified to meet real-time requirements.

1 Introduction

Legged vehicles can help to get access to many parts of earth surface which wheeled vehicles can not cope. Different concepts are used to control legged locomotion in unstructured environment. Simple control concepts use gait pattern, which define a fixed sequence of footholds and are suitable for quite flat terrain. Reactive control components like the elevator reflex [8] are added to improve walking capabilities and terrain adaptation. But problems arise if the exception handling dominates the normal walking process. Path planning can help to minimize exception handling occurring. It uses some environment information to calculate movements avoiding exceptions. At the same it should result in good movements. This leads to the use of an optimization approach as described in section 4.

For getting the needed environment information we use stereo vision. The usual algorithms do not meet our real-time constraints or require special hardware. So we developed an anytime algorithm [3] that returns imprecise but sufficient results if aborted. It is described in chapter 3.

2 Related Work

There exist many control systems for walking robots. Gait pattern as used in [7, 17] are stable but not reliable on uneven ground. Even reactive control components like used in [8] have the mentioned drawbacks. Motion planning in [12] is only used to control gait parameters like stroke height. Planning as described in [20] switches between proper gait pattern, but these are limited. In [5] the whole motion of the robot is planned. Genetic algorithms and back-tracking is used for this. But there are no real-time assertions. The ordinal optimization approach used in [2] cannot guaranty that a solution is found.

The real-time stereo vision solutions used in [13, 14, 16] require special hardware or set constraints to the target environment [4]. As we focus on a more general approach, we developed an algorithmic solution.

3 Stereo Vision as Anytime Algorithm

Stereo vision algorithms are generally time consuming. To use them within a real-time system, an implementation with a predictable runtime is required and so an anytime algorithm is preferred. This class of algorithms requires that a result is available no matter when the algorithm is stopped.

To get an anytime algorithm we first have to choose a stereo vision method, which we can implement as such an algorithm. Our choice for this was the block-matching method. It is a representative for area based stereo [4]. We decided to use area based stereo, because it can be used either for all pixels of an image or for selected areas [21]. This provides the opportunity to extract features in the image and compute the displacement vectors for these features only. In the future, one can use this opportunity to decrease computation time, if only parts of the image are of interest.

The main idea of the block matching algorithm is a similarity measure between two equal sized blocks ($n \times m$-matrices) in the stereo images. This means, a block in the left image will be compared with all possible blocks of the right image, which have the same size. A standard method to describe the similarity is the mean square error (MSE) of the pixel values inside the respective blocks. For further details on Stereo Vision using the standard block matching method see [6].

But the standard block-matching method can only provide results after the algorithm is finished. To transform it into an anytime algorithm means, changing it to get a first result as soon as possible, which can be gradually improved afterwards. We have solved this problem by using a pyramid model approach.

3.1 Hierarchical Stereo Vision

Tanimoto and Pavlidis [18] introduced the use of pyramid models for image processing. We use a four–level pyramid to enhance the block matching method.

The correspondence problem will be solved on each level of the image pyramid starting with the coarsest level. The results of the previous level are used to initialize the disparity computation of the more detailed level.

Therefore, the search for corresponding blocks starts at the disparities, which were found in the coarse levels. This allows to decrease the search space in horizontal direction. A minimal *MSE* can be found fast. For further details on the algorithm see [11].

The main idea is to be able to use the results of every level as computation result, if the algorithm is stopped in between because of timing constraints. Missing results of one level are completed with results of the previous level. This ensures that the result improves steadily. Therefore, a first result, though imprecise, is available in short time. Because imprecise results are better than no results they can still be used, e.g. to avoid collisions.

Now we show how to speed up our method to faster get the highly detailed results. A serious problem of the block-matching method is the size of the search space. If the epipolar constraint (see [6]) can not be fulfilled the search space must be spread to several rows. Another assumption is made about the order of the pixels in both images. It is assumed that pixels of one epipolar line in one of the stereo images can be found in the same order in the corresponding epipolar line in the other image. But this assumption is violated, if the depth distance between objects in the observed scene is too large. We considered that, as we do not want to restrict the environment. So the block-matching method was changed to allow for a search of corresponding blocks in horizontal and vertical direction. But to limit the search space we use an initialized disparity value (from previous pyramid level) so that definite false results will not be considered. Additionally, the search space in horizontal direction can be limited due to the vision parameters of the cameras. The closer an object to the cameras is, the larger the disparity values of the corresponding pixels will be. If one assumes a minimum distance, e.g. by the stereo camera configuration on the robot, the maximum horizontal search space can be defined exactly.

However, not only the search space defines speed and quality of the results. An important parameter is the block size. We have tested different block sizes and concluded that the ratio between block height and block width should be 2:3, if the epipolar constraint is used further on. In our tests, a block size of 12×8 pixels yields the best results. The hierarchical structure of the algorithm and the available provisional results lead to an anytime algorithm. With this, we have a method that provides depth information from stereo images in real-time.

4 Walking as Optimization Problem

The goal of motion planning is to calculate a good movement based on environment information and application demands. By the claim for a *good* solution the problems may be treated as an optimization problem: Find a valid solution (movement) from the solution space (all possible movements) that is optimal regarding given objectives. To solve the optimization problem, we need a formal definition and an algorithm for solving it.

4.1 Definition

A formal optimization problem is a triple (U, S, ϕ). The input U parameterizes the set of valid solutions $S(U)$. The weighting function ϕ defines some value by mapping every elements to a real number:

$$\phi : S(U) \mapsto \mathbb{R} \tag{1}$$

This triple has to be defined for our given problem. The input consists of the source and destination position and the environment information generated from the sensor data. The positions are defined by two-dimensional world coordinates. The environment information is given by a height map which defines the height of a point every 100 mm.

The solution space represents all valid movements. It should include *every* possible solution. Otherwise a good one could already be excluded by the definition. The whole movement is composed by the motion of the individual legs and the robot body. Each of them is described by a reference point. For the legs this is the foot, for the body the center of gravity. Their movement is described by a list of *events*. An event defines a linear movement between two points in a given time. By concatenation it can be used for linear approximation with arbitrary precision. So we have seven lists of such events for describing the movement of the robot.

The weighting function assigns a value to reflect the quality of a movement. We considered several criteria for this. They are combined to define the final value. The most important criterion is the speed. We use the direct distance to the destination divided by the time. The faster, the better the movement is. An additional point we considered is the stability of the robot. This is affected by the ground and the arrangement of the legs. For the latter case the stability margin (as described in [15]) gives a good measurement. We calculate the minimal value for the whole movement. The bigger, the lower is the risk to tip over. For the stability of the ground we considered steer areas. The robot can loose it's grip on this. We use the maximal gradient of the ground under all steps. The lower it is, the better the movement. All these conditions can be incorporated into a weighting function $\psi_i : S(U) \mapsto \mathbb{R}$. These terms are linearly combined to the final value: $\phi(s) = \sum_i \lambda_i \psi_i(s)$. The λ_i allow to adjust the influence of the criteria.

4.2 Possible Heuristics

Due to the complexity of the problem no exact solving algorithm is known. So we concentrate on finding a good heuristic. Some of the commonly used methods to solve optimization problems were investigated regarding their practicability for our problem. At first this is only done theoretically by checking common criteria. The results are listed in table 1. Their description can be found in [10, 1, 19].

heuristic	*anytime algorithm*	*parallel computation*	*required memory*	*general applicability*
greedy	no	no	low	bad
branch and bound	yes	yes	low	bad
local search	yes	multi start	low	good
tabu search	yes	multi start	high	good
random sampling	yes	yes	low	very good
simulated annealing	yes	multi start	low	good
genetic algorithms	yes	yes	high	good

Table 1. selection of possible heuristics

We want some anytime algorithm [3] that allows us to stop the calculation at any time giving a valid (maybe suboptimal) result. Furthermore, to enable the possible use of all processors in the robot, parallel computation should be possible and the memory requirements should not be too high. The most important requirement is the general applicability to the problem. This ensures that we can use it for our problem.

As a result, three possible heuristics are chosen: *local search*, *random sampling* and *simulated annealing.*

4.3 Implementation

For the three heuristics we need some way of generating random valid movements. This appears to be a nontrivial task. No efficient algorithm is known for only determining, if a valid movement exists [5]. So we lower our requirement to an algorithm which generates random movements that are valid in most cases. The result is a multistage process as described in [9]. For proper handling of invalid solutions we use a penalty value. They are treated as valid but really bad. So they should never occur as optimum.

For local search and simulated annealing a neighborhood definition is needed. Here a neighbor is defined by a small difference in one of the movement parameters, e.g. a foot position.

The three heuristics are applied to our optimization problem. The resulting algorithms follow the usual description. For testing we use a virtual environment. The world model is created manually at the moment. It is represented by a raster image where the green channel of a pixel corresponds to the height. This allows us to create various scenarios. The movement of the robot is planned regarding the given terrain. To verify the results we developed a visualization (see figure 1).

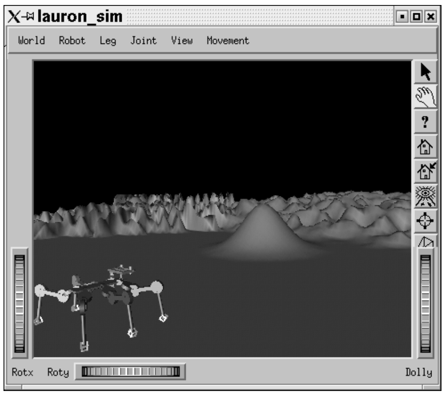

Fig. 1. robot visiualization

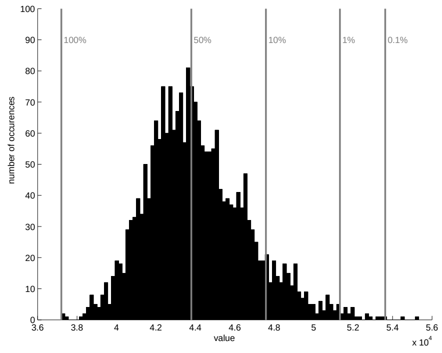

Fig. 2. results: flat ground

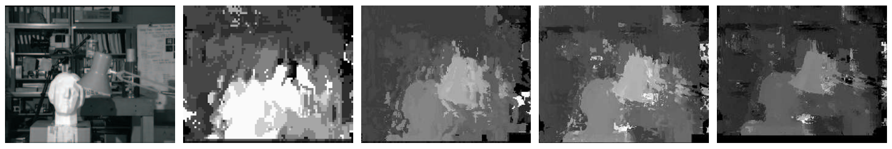

Fig. 3. original image on the left and the results after every pyramid step

5 Experimental Results

5.1 Results from Stereo Vision

We found out that our anytime stereo vision algorithm provides results much faster than the unmodified stereo vision algorithm we used as basis, if more imprecise results are acceptable. In figure 3 one can see that the results after every pyramid step becomes better. This is, the more time is available for the computation, the more detailed results are produced. For motion planning, an earlier result could be used as big structures and objects are already visible. This is sufficient for avoiding large obstacles.

5.2 Results from Motion Planning

Our tests with the different motion planning algorithms showed that only the random sampling approach is suitable for practical use. Instead of the other ones its computation time meets our requirements.

As random sampling is a probabilistic algorithm we had to run it several times for evaluation. Figure 2 shows the valid results for walking on even terrain. We got 2 259 valid movements for 10 000 iterations. So the probability for a valid solution per iteration is $p = \frac{2259}{10000} = 0.2259$. The chance of finding at least one valid solution depends on the number of iterations n. it

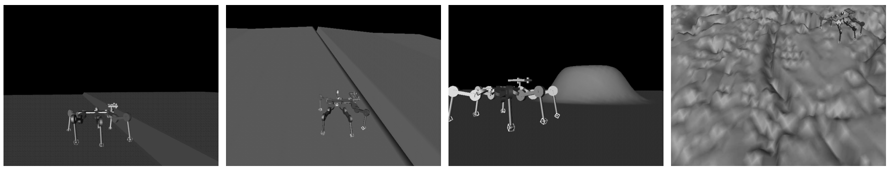

Fig. 4. scenarios: step, trench, unsurmountable obstacle, irregular terrain

can be computed by $P(\text{n times}) = 1 - (1 - p)^n$. For $n = 100$ it is already 0.999 999 999 992 419. So you can assume that after a certain number of iterations the chance of finding a valid one is nearly one. Figure 2 shows the distribution of the solutions. To get a solution of the best 1% more iterations are required. In this example there is a probability of 0.89 for 1 000 iterations. This is typical for a calculation time of 30 seconds.

The same way other scenarios are tested: *trench, step, unsurmountable obstacle,* and *irregular terrain.* All of them are solved sufficiently. The benchmarks *trench* and *step* are constructed in a way that normal regular gait pattern could not handle them. The robot has to choose the right distance to the obstacle and a special order of moving the feet. Our motion planner is able to handle this. The scenario *unsurmountable obstacles* demonstrates that the motion planner is even able to correct wrong guidelines from the application. The robot should walk to point straight ahead. But in direct line there is an obstacle it can not overcome. The planner handles this situation by walking around. The scenario of *irregular terrain* represents a typical case. It contains many steer areas the robot should not step on. The planner is also able to handle this.

6 Conclusion

Concluding, we want to point out that we dealt with the problem of motion planning based on information provided by optical sensors. This problem includes two main subproblems. First, obstacles had to be detected under real-time conditions. Second, this data were used for motion planning. Obstacle detection was solved with an anytime version of a stereo vision algorithm. This allows to comply with the real-time constraints. The motion planning has been treated as an optimization problem, which allows solving it with a good heuristic.

References

1. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer Verlag, 2000.

2. C.-H. Chen, V. Kumar, and Y.-C. Luo. Motion planning of walking robots in environments with uncertainty. In *Journal for Robotic Systems*, volume 16, pages 527–545. 1999.
3. T. L. Dean. Intrancibility and time-dependent planning. In *Workshop on Reasoning about Actions and Plans*, 1987.
4. E. D. Dickmanns and V. Graefe. Dynamic monocular machine vision. *Machine Vision and Applications*, 1, 1988. Springer International.
5. C. Eldershaw. *Heuristic algorithms for motion planing.* D.Phil. Thesis, University of Oxford, 2001.
6. O. Faugeras et al. Real time correlation-based stereo: algorithm, implementations and applications. Technical Report 2013, Institut national de recherche en informatique et en automatique, 1993.
7. Cynthia Ferrell. A comparison of three insect-inspired locomotion controllers. *Robotics and Autonomous Systems*, 16:135–159, 1995.
8. B. Gaßmann, K.-U. Scholl, and K. Berns. Locomotion of LAURON III in rough terrain. In *International Conference on Advanced Intelligent Mechatronics*, volume 2, pages 959–964, Como, Italy, July 2001.
9. A. Herms. Entwicklung eines verteilten Laufplaners basierend auf heuristischen Optimierungsverfahren. Diploma thesis, University of Magdeburg, 2004.
10. J. Hromkovič. *Algorithmics for hard problems: introduction to combinatorial optimization, ranomization, approximation and heuristics.* Springer Verlag, 2001.
11. T. Ihme and R. Bade. Method for hierarchical stereo vision for spatial environment modelling supported by motion information. In *Proceedings of Robotik 2004*. VDI-Verlag, June 17 – 18, 2004. Munich, Germany.
12. M. A. Jiménez and P. González de Santo. Terrain-adaptive gait for walking machines. *The International Journal of Robotics Research*, 16(3), June 1997.
13. K. Konolige. Small vision systems: Hardware and implementation. In *Eighth International Symposium on Robotics Research, Hayama, Japan*, pages 203–212. Springer Verlag, oct 1997.
14. A. Kelly L. Matthies and T. Litwin. Obstacle detection for unmanned ground vehicles: A progress report. In *International Symposium of Robotics Research, Munich, Germany*, oct 1995.
15. E. Papadopoulos and D. Rey. A new measure of tipover stability margin for mobile manipulators. In *IEEE International Conference On Robotics and Automation*, 1996.
16. T. Ohm R. Volpe, J. Balaram and R. Ivlev. The rocky 7 mars rover prototype. In *International Conference on Intelligent Robots and Systems, Osaka, Japan*, volume 3, pages 1558–1564, nov 1996.
17. Shin-Min Song and Kenneth J. Waldron. An anaytical approach for gait study and its applications on wave gaits. *The International Journal Of Robotics Research*, 6(2):60–71, Summer 1987.
18. S. Tanimoto and T. Pavlidis. A hierarchical data structure for picture processing. In *Computer Graphics and Image Processing*, volume 4, 1975.
19. P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications.* Kluwer Academic Publishers, 1987.
20. D. Wettergreen. *Robot walking in natural terrain.* PhD thesis, Carnegie Mellon University, 1995.
21. C. Zhang. A survey on stereo vision for mobile robots. Technical report, Dept. of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh.