2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Map-based Adaptive Foothold Planning for Unstructured Terrain Walking

Dominik Belter, Przemysław Łabęcki and Piotr Skrzypczyński

Abstract— **This paper presents an adaptive foothold planning method for a hexapod walking robot. A local terrain map acquired with an inexpensive structured light sensor is exploited as the information source for the planning algorithm, which uses a polynomial-based approximation method to create a decision surface. The robot learns from simulations, therefore no *a priori* knowledge is required. The results show that the method is general enough to work on various types of terrain. The planned footholds enable the robot to walk more stable, avoiding slippages and fall-downs.**

I. Introduction

In the recent years walking robots are in the area of high interest because of their ability to access unstructured terrain – they are able to climb rocks, curbs, and thresholds. However, to exhibit an autonomy in an unstructured terrain a legged robot requires a relatively complicated control algorithm. There are several problems to deal with in such an algorithm: terrain model acquisition, foothold selection, static and dynamic stability, and path planning. Terrain perception/mapping and foothold selection are crucial, because whenever the feet are placed improperly on the ground the probability of slippage and the risk of fall increase rapidly. On the other hand, reliable terrain mapping and careful foothold selection simplify maintenance of the stability.

To achieve an autonomy, the robot requires a sensing system that perceives the terrain profile ahead of it, and enables to build a local map, which serves the purpose of safe/optimal foot placement. However, the limited payload and energy affect the sensing abilities and on-board computing power of a walking robot. Considering these technological limitations, and a limited budget of our small-scale Ragno hexapod robot project [15], we have decided to equip the robot with a low-cost monocular vision system supported by a laser stripe projector, forming together a structured light sensor [11].

Once a terrain map is available, a foothold selection method can be implemented. Although the robotics literature describes many walking robots with correctly working foothold selection systems, these approaches are usually heavily based on the domain knowledge, with a large number of parameters that should be set by a human expert [12]. Hence, they are hardly adaptable to various gaits and different types of terrain. In contrast, the algorithm proposed in this paper allows to create a ground scorer which learns by using results of experiments conducted on the real robot or its simulated model. A human expert is not necessary, as the robot learns unsupervised how to walk on the rough terrain. This concept, together with the terrain map acquisition system, gives the robot a possibility to self-adapt the foothold selection method to various terrain types.

This work is funded by MNiSW grant N514 294635 in years 2008–2010

D. Belter, P. Łabęcki and P. Skrzypczyński are with Institute of Control and Information Engineering, Poznan University of Technology, 60-965 Poznan, Poland {dominik.belter,przemyslaw.labecki, piotr.skrzypczynski}@put.poznan.pl

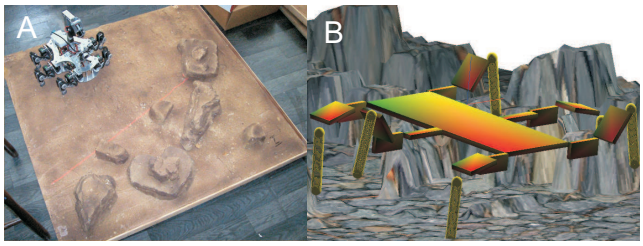

Fig. 1. Ragno the robot on a rough terrain mockup (A) and simulated robot (B)

II. Related work

The problem of terrain mapping has been considered many times in the robotics literature. Ambler planetary rover [8] was one of the first walking robots being able to map their surroundings by means of a 3D laser scanner. The terrain representation used was the elevation map, a grid-type $2\frac{1}{2}$D map, where each cell holds a value that represents the height of the object at that cell. Other terrain representations used with legged robots include an inference grid used by the Lauron III robot [6], and a recent probabilistic technique [9], which builds maps based on Gaussian process models, and does not assume a fixed discretization of the space.

The Lauron robot is an example of a walking machine, which uses occupancy and credibility map to make decision about the appropriate footholds [13]. The chosen best position of a foot depends on the value of credibility and the distance from the center of the local map (which is close to a foothold for the "normal" walking behavior). Also [4] uses a special map characterizing the perception uncertainty to plan the motion of the robot on a rough terrain. Based on this information a gait is planned by using ordinal optimization. The quadruped LittleDog robot has been equipped with a terrain scorer/classifier, which evaluates the potential foot placement at a given point as acceptable or unacceptable, taking into account the height of the considered point and the adjacent points [12]. The scorer rejects points which are located on too large slopes, are too close to the top edge or the base of a cliff, or are inside of a hole.

Although a detailed terrain map gives the walking robot much flexibility in taking the decisions about its further steps,

including the possibility of larger obstacle negotiation and feet trajectory planning, the foothold selection itself may be based on a different principle. For example, a controller that adapts to feet slippages by using force control has been proposed in simulation by [10]. The adaptive controller of a robot for walking on irregular terrain can be also based on the Central Pattern Generators (CPG) and a system of reflexes [5].

III. The Robot and the Simulator

The hexapod walking robot Ragno (cf. Fig. 1A) is used in this research. Each leg has three joints that are driven by integrated servomotors. They allow to set the desired angles in joints and to generate movement of the robot. The robot is relatively small. It weights 2.155kg without batteries and can fit in box of 33×30cm. Its mechanical structure allows to walk with the trunk from 15 to about 70 mm over the ground (the maximal height of the trunk is 120 mm, however it is the end of robot's workspace and the horizontal movement is not possible). The robot has been equipped with various types of sensors including the MEMS accelerometers that form an Inertial Measurement Unit (IMU) to measure the θ (pitch) and ϕ (roll) angles [15].

The simulator is based on the Open Dynamics Engine (ODE). This library allows to simulate rigid body dynamics. It has methods for modeling various joints and for detecting collisions with friction and softness [14]. The ODE solver was set to be non-deterministic. As a result the same simulations give slightly different results if repeated, what makes the simulation more realistic. For object visualization the OpenGL routines have been used. The robot is represented by simple rigid objects connected by rotary joints (cf. Fig. 1B). The model of the robot, which has been successfully used to evolve feasible gaits [2], is designed to be simple because of the simulation speed. Servomotors are simulated as simple proportional controllers. They receive reference values α_r on input, what generates an appropriate torque to drive the joint. An error between the desired foot position and the actual foot position achieved, which is observed in the real robot, is also present in the simulator.

IV. Terrain Description and Decision Making

A. Terrain mapping method

To select footholds for a walking robot it is necessary to build a model of the environment. For that purpose we prefer a grid-type map that is simple in implementation and easy to update at a high rate, and may be directly used to select proper footholds for the robot. However, a classic elevation map has its drawbacks: it does not provide information about the elevation uncertainty that can be utilized by the motion planner [9], and it provides no means to compensate such undesirable effects as missing data and range measurement artifacts. Such effects occur in all types of range sensors, but for the considered structured light sensor they may particularly deteriorate the mapping performance, due to the high probability of occlusions (missing data), and vulnerability of

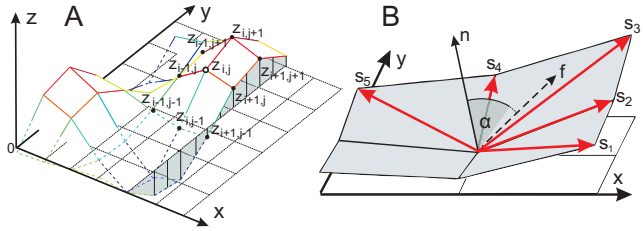

Fig. 2. Computation of K1 and K2 (A) and K3 (B) coefficients

the sensor to the environmental conditions (e.g. artifacts due to light spots) [11].

Taking into account these considerations we have decided to implement for our robot the CAS (Certainty Assisted Spatial) filter based version of the elevation grid map proposed by Ye and Borenstein [3]. In this approach the terrain map consists of two grids of the same size: an elevation grid and a certainty grid. The elevation grid holds values that estimate the height of the terrain, while each cell in the certainty map holds a value that describes the accuracy of the corresponding cell's estimate of the elevation. We use robot-centered local grids with a cell size of 5×5 mm. The CAS-filter uses physical constraints on motion continuity of the robot and spatial continuity of the terrain to identify corrupted and missing data in the elevation grid. As in [3], we assume that an estimate of the 6-dof robot pose is available. Although for a legged robot such an estimate is not easy to obtain, for small local maps centered in the robot coordinates proprioceptive sensing is enough [7], while for more challenging scenarios visual odometry may be used. The quality of the resulting terrain map was discussed in more detail in [11]. Although we have no ground truth data (i.e. a perfect map) for the rocky terrain mockup we used in the foothold selection experiments, mapping objects of known size we have found that the errors are below 10% of the measured height of an object in the worst case.

B. Control strategy

While walking on the rough terrain the robot uses a tripod gait [2]. This gait has been chosen because it is the fastest statically stable gait for a hexapod walking robot. During the stance phase the robot maintains a constant average height of the trunk above the ground. This average height is computed by using the last six footholds. To stabilize the robot orientation during walking the information about θ and ϕ angles from the IMU is used.

At the end of the swing phase of the gait the robot searches for three new footholds. This simple strategy enables to use the same swing trajectory of a foot that is used in the nominal gait, but it sporadically results in a necessity to move a foot to a different position after its swing. We are working on an adaptive foot trajectory planner that considers the selected footholds in advance. With the new planner, foothold selection will take place before the swing phase. Starting the search the robot checks if the potential footholds are in the workspace of the considered leg. The robot raises or lowers

body, when no appropriate foothold is found. If there are no appropriate footholds with the reach of one of the legs, the height of the robot body is changed to reach a proper support point. If this strategy fails the control system informs the path planner module that the planned movement is not achievable. The problem is being solved by the higher layers of the control system, what is out of the scope of this paper. However, in the reported experiments and simulations (cf. Section VI) the reactions which change robot's posture were sufficient to deal with the problem.

C. Ground properties

The presented algorithm starts with the elevation map built from the structured light sensor data, and computes four coefficients to describe the potential footholds [1]. The K_1 coefficient for the i and j grid co-ordinates is defined as:

$$K_1(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} (z_{i,j} - z_{i+k,j+l}), \tag{1}$$

where $z_{i,j}$ and $z_{i+k,j+l}$ are terrain levels of the corresponding points of the grid map (Fig. 2A). This coefficient allows to detect a top edge or a hole. For a top edge K_1 is positive and for a hole K_1 is negative. The value of K_1 gives information about the local terrain extreme, however K_1 is ambiguous when its value is 0. Flat terrains as well as a terrain with constant slope give the same results.

The K_2 coefficient for i and j grid references is defined as:

$$K_2(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} |z_{i,j} - z_{i+k,j+l}|. \tag{2}$$

The K_2 coefficient provides an information about the slope of a terrain. It returns different results for a constant slope and for a flat terrain. On the other hand, top edges and holes give the same results. Thus, K_1 and K_2 are ambiguous separately, but together they give a fundamental information about the structure of the terrain.

The K_3 coefficient is defined by an angle α between a vector $\mathbf{f}$ describing the foot movement and a vector $\mathbf{n}$ normal to the surface. The normal vector $\mathbf{n}$ is computed only for a part of the surface as it was shown in Fig. 2B. It depends on the projection of the vector $\mathbf{f}$ on the surface xy. Tangent vectors $\mathbf{s}_1, ..., \mathbf{s}_5$ are computed for points which are neighbors of the quadrant where projection of the vector $\mathbf{f}$ is located. Next, the vector $\mathbf{n}$ is computed as follows:

$$\mathbf{n} = \sum_{i=1}^{4} \mathbf{s}_i \times \mathbf{s}_{i+1}. \tag{3}$$

The angle α between the vectors $\mathbf{n}$ and $\mathbf{f}$ is computed as:

$$\alpha = K_3 = \arccos\left(\frac{\mathbf{n} \cdot \mathbf{f}}{|\mathbf{n}||\mathbf{f}|}\right). \tag{4}$$

The K_4 coefficient is computed as a distance between the considered point and the point being the center of the local map (foothold for the "normal" walking behavior (x_0,y_0,z_0)):

$$K_4(i,j) = \sqrt{(x_0 - x_{i,j})^2 + (y_0 - y_{i,j})^2}. \tag{5}$$

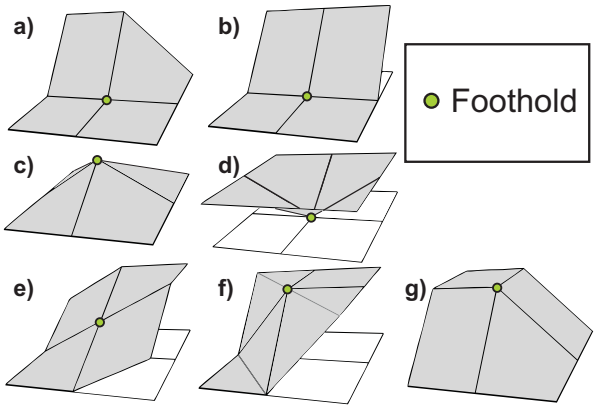

Fig. 3. The ground primitives

V. Foothold selection algorithm

The aim of the algorithm is to evaluate potential footholds and to choose the best one. The best point for a foot stance should minimize a risk of slippage. Slippage is defined by the difference between the position of a foot at the beginning and at the end of the stance phase. If the foot does not slip at all, these positions should be the same. Thus, i-th foot slippage r_i is computed as $r_i = d_i/|\mathbf{f}_i|$, where d_i is the distance between the initial and the final position of this foot during the stance phase, and $|\mathbf{f}_i|$ (used as a normalizing factor) is length of the vector that defines the intended i-th leg movement with regard to the body. The algorithm consists of three stages: (i) collecting data, (ii) learning – approximation, (iii) regular operation.

A. Collecting data

At first, the robot collects the appropriate data for learning. Two strategies of data acquisition were investigated. In the first one the robot is walking on the rough terrain, randomly selects footholds, saves coefficients corresponding to these points and as a result of this selection – a foot slippage measured after the stance phase. In the second approach the robot tests seven previously prepared "ground primitives" (shown in Fig. 3). Each of the robot's foot is placed on the same primitive. The robot executes a movement, and after that a slippage of each foot is registered. This simulation is repeated for all primitives. To test different ground types the height of the primitives is changed during the acquisition phase in the range of 4 cm, with a step of 1 cm, and the primitives are rotated with the step of $45°$. After this stage the robot has an appropriate data set to build the decision surface and distinguish between the good and the poor footholds.

The point cloud shown in Fig. 4 was obtained in a situation when only two coefficients were used (K_1 and K_2), and when the primitives were used for data acquisition. Whenever the input space has a higher dimensionality it is not possible to show the results graphically.

B. Learning stage

In the second phase the robot forms its decision surface. It is difficult to deduce information about usefulness of

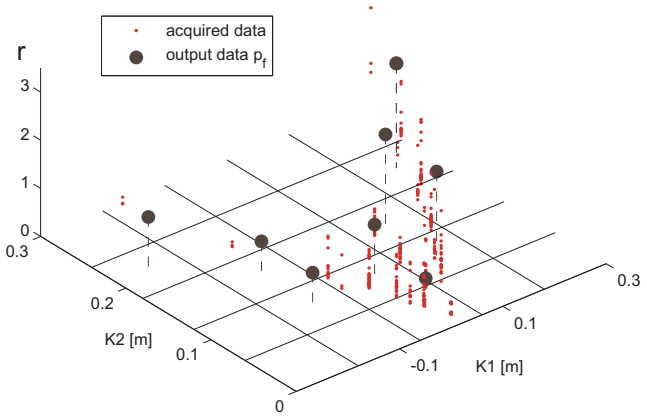

Fig. 4. Collected data and discretization effect

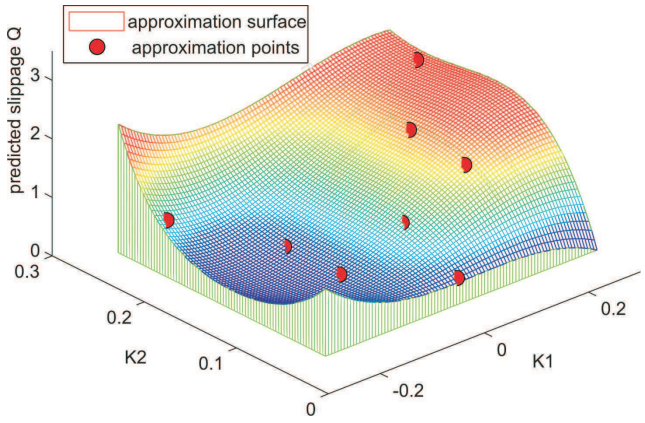

Fig. 5. Decision surface

footholds from a raw point cloud. The data are very often ambiguous. Selection of the same points (defined by the same coordinates K_1 and K_2) gives different slippage values. To deduce useful information from the point cloud a discretization in a fixed grid is used. The input space K_1, K_2 and K_3 is divided into regular grid (Fig. 4 for situation when only K_1 and K_2 are used). For points whose projection lies in the considered grid cell a mean value is computed. The result is placed in the center of the considered grid cell. After this discretization a group of points $\mathbf{p}_f = [p_{f1}, ..., p_{fk}]^T$ which represent general trend is obtained.

A foothold selection method should be general, but it is not possible to test all types of terrain. However, the algorithm should have the ability to judge the usefulness of the ground points, which haven't been tested explicitly in the data acquisition stage. To this end a least-squares polynomial approximation is used [16]. The points are no longer needed, because the knowledge is stored in the approximation polynomial. The approximation assignment requires a selection of an appropriate polynomial base:

$$P(K_1, K_2, ..., K_n) = \sum_{i=0}^{m} c_i \cdot \phi_i(K_1, K_2, ..., K_n), \qquad (6)$$

where $K_1, K_2, ..., K_n$ are coefficients which characterize the terrain and m is a number of elements in the polynomial. The polynomial P allows to assess potential footholds represented by values of the $K_1, ..., K_n$ coefficients. To determine the c_i values, the Least-Squares Fitting method is applied on the set of $\mathbf{p}_f$ points obtained in the data acquisition stage. The polynomial coefficients $\mathbf{c}$ are computed as follows:

$$\mathbf{c} = (\mathbf{V}^T \cdot \mathbf{V})^{-1} \cdot \mathbf{V}^T \cdot \mathbf{r}, \qquad (7)$$

where

$$\mathbf{V} = \begin{bmatrix} \phi_1(p_{f1}) & \phi_2(p_{f1}) & ... & \phi_m(p_{f1}) \\ ... & ... & ... & ... \\ \phi_1(p_{fk}) & \phi_2(p_{fk}) & ... & \phi_m(p_{fk}) \end{bmatrix}, \qquad (8)$$

is a Vandermonde matrix, and $\mathbf{r}$ is a vector of slippages corresponding to the respective $\mathbf{p}_f$ points.

The obtained decision surface and approximation points $\mathbf{p}_f$ are shown in Fig. 5 (only K_1 and K_2 are used, for the sake of clarity).

C. Regular operation

During the third phase the robot uses the approximated polynomial to assess the potential footholds. Then, these results are used to find the best place to put the foot on it.

The first three coefficients are used as an input to compute the polynomial (6). The fourth one (K_4) is used to exclude points of the ground which are too close to the boundaries of the robot's leg workspace. The final $Q(i,j)$ coefficient (interpreted as a prediction of slippage), which describes the usefulness of the potential $[i,j]$ foothold described by K_1,..,K_4 features is given as:

$$Q(i,j) = P(K_1(i,j), K_2(i,j), K_3(i,j)) + k \cdot K_4(i,j). \qquad (9)$$

The tuning constant k was set to 8. When this value is too big the robot chooses points which are in the center of the local map. When it is too small, there is a risk that the robot will choose points which are very close to the boundaries of the leg's workspace, what might render the movement impossible.

VI. Results

During walking the robot uses the standard gait for walking on the flat terrain. This gait gives an expected foothold for each leg of the robot. Around the expected foothold a local grid map is defined. Its size is set to 15×15 cells (7.5×7.5 cm). The computed polynomial is then used to evaluate all the points of this local map. An example of the local map with the assessment indicated is shown in Fig. 6. There are cells which are excluded from the set of the potential footholds because they are out of the range of the robot's legs. Cells are also excluded when the value of Q is too big (such cells give a weak support to the leg's tip) or a cell is described by K_i values which are outside the boundaries defined in the learning stage (properties of this cell are unknown). The foothold selection algorithm searches

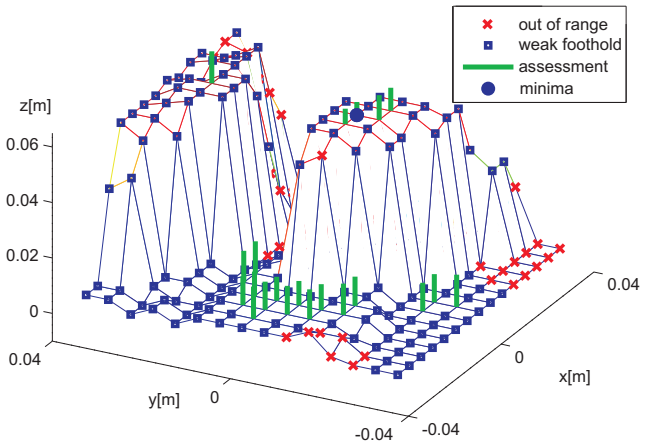

Fig. 6. Example terrain and assessment results (see text)

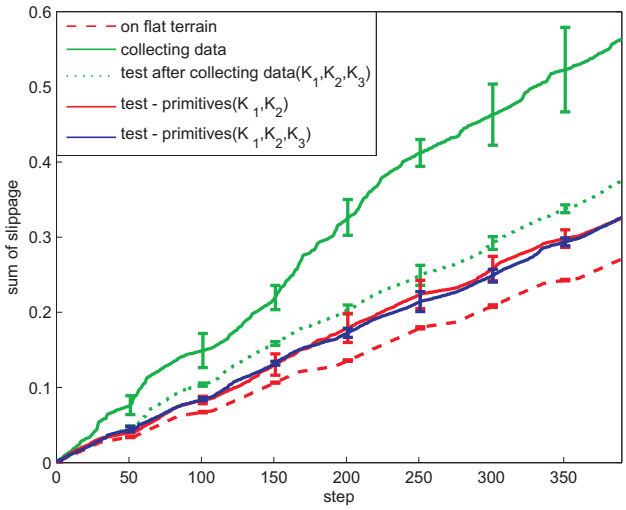

Fig. 7. Comparison of the average slippage values resulting from the simulation

for the minimum among the rest of the cells. The shorter length of the bar (shown in Fig. 6) the more useful is the given cell as a foothold. After a decision about foothold selection the control system of the robot modifies trajectories of the feet, and places them on the selected points of the terrain.

Slippages during the whole simulation run have been recorded to show results given by the proposed algorithm (Fig. 7). Each simulation consists of a series of three tests. Mean value of the accumulated slippage and its standard deviation are shown. These results are compared to the results of a simulation on a flat terrain, and a simulation when the robot randomly selects footholds. Better results were obtained when a point cloud acquired using the prepared primitives was used. The results when three coefficients were used for decision making are slightly better. When only K_1 and K_2 coefficients were used the robot had problems with extensive hills of mild slope.

The algorithm has been verified in simulation, but using a real map of a rocky terrain (an indoor mockup was used)

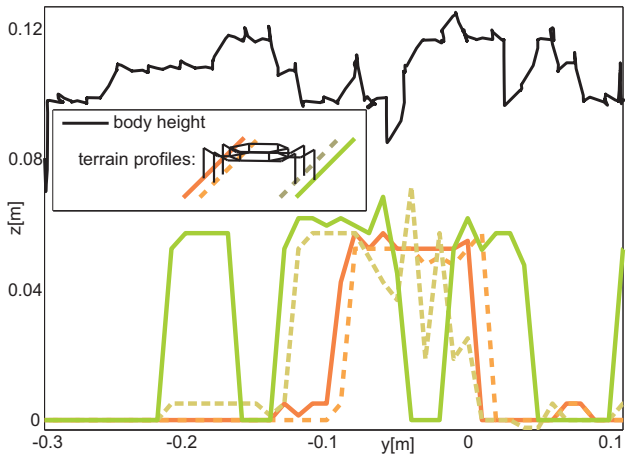

Fig. 8. Body height variations resulting from the simulation

obtained by the Ragno robot equipped with the structured light sensor. The map contains obstacles more than 13 cm high, however the robot passes by the highest rocks and steps obstacles only 6.5 cm high. During the simulation the robot walked 0.5 m forward, traversed 0.06 m left, turned $45°$ left, and then went 0.1 m ahead.

The changes of the trunk height with regard to the start pose, and motion profiles for the feet are shown in Fig. 8. The height of the robot's trunk changes according to the terrain profile. When the robot selects footholds which are located on the obstacles it moves the trunk up. During walking down the hills the robot moves its legs down, and it searches for appropriate footholds. Usually the ground is out of the front leg's workspace and as a result the robot lowers the trunk to find proper contact points.

The robot's paths during the simulations have been shown in Fig. 9. In the first case the robot does not use any algorithm for foothold selection. As a result it places feet on the edges of cliffs or on large slopes. When even one leg has an insufficient support, the robot loses the stability and falls down. It rapidly changes its orientation (here measured as a module of error for two angles: $\sqrt{\theta^2 + \phi^2}$) as it is shown in Fig. 10). Finally, the orientation in the horizontal plane is $-64°$ while the desired angle was $45°$. When the control algorithm uses the proposed foothold planning algorithm the trajectory of the robot's trunk is much smoother (cf. Fig 9B). The robot changes its position to adapt itself to the terrain level. There are only small orientation changes during walking. They are caused by few leg's collisions with the ground during the swing phase. The orientation error in z axis after the simulation is only $5°$.

VII. Conclusions

This paper presents a foothold selection system for a small, multi-legged walking robot. The system uses a terrain map acquired by using an inexpensive structured light sensor, and learns automatically without any *a priori* expert knowledge. The robot acquires the ground surface characteristics by walking on a known example terrain or by testing

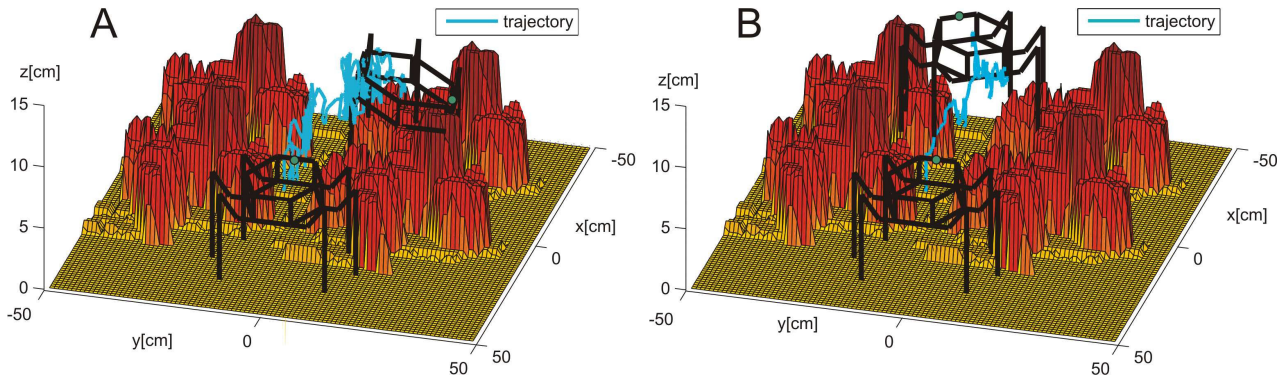

Fig. 9. Results – trajectories without (A) and with the foothold selection algorithm (B)

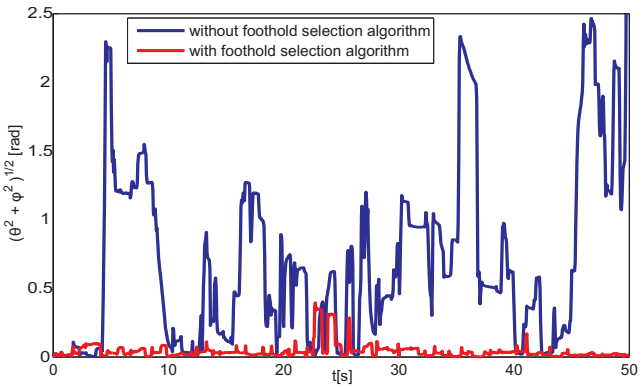

Fig. 10. Results – orientation during simulation

ground primitives. Then it exploits this knowledge to build a decision surface, which is used for assessment of the candidate footholds. The decision surface reflects only geometric characteristics of candidate footholds, because these characteristics are the only information provided by the range sensor employed in our system. However, if any information about other ground properties, e.g. terrain compressibility is available, it may be included by adding coefficients to (6).

The simulations show that after learning the robot is able to select the appropriate footholds to prevent slippages. The same rules work on various types of terrain. The developed decision unit has been tested on a map of the real terrain and verified on the real robot Ragno[1].

As a further development we plan to test it on our new, bigger hexapod robot Messor, which is able to carry a Hokuyo URG-04LX laser scanner.

REFERENCES

[1] D. Belter, "Adaptive foothold selection for a hexapod robot walking on rough terrain", *7th Workshop on Advanced Control and Diagnosis*, Zielona Gora, Poland, 2009, CD-ROM.

[2] D. Belter, A. Kasiński, P. Skrzypczyński,"Evolving feasible gaits for a hexapod robot by reducing the space of possible solutions", *in Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, Nice, France, 2008, pp. 2673–2678.

[3] C. Ye, J. Borenstein, "A Novel filter for terrain mapping with laser rangefinders", *IEEE Trans. Robot. and Automat.*, 20(5), 2004, pp. 913–921.

[4] C.H. Chen, V. Kumar, "Motion planning of walking robots in environments with uncertainty", *in Proc. IEEE Int. Conf. on Robotics and Automation*, Minneapolis, USA, 1996, pp. 3277–3282.

[5] Y. Fukuoka, H. Kimura, A.H. Cohen, "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts", *Int. Journal of Robotics Research*, vol. 22(3–4), 2003, pp. 187–202.

[6] B. Gaßmann, L. Frommberger, R. Dillmann, K. Berns, "Real-time 3D map building for local navigation of a walking robot in unstructured terrain", *in Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, Las Vegas, USA, 2003, pp. 2185–2190.

[7] B. Gaßmann, J. M. Zöllner, R. Dillmann, "Navigation of walking robots: localisation by odometry", *Climbing and Walking Robots VIII*, Springer, Berlin, 2005, pp. 953–960.

[8] E. Krotkov, R. Hoffman, "Terrain mapping for a walking planetary rover", *IEEE Trans. Robot. and Automat.*, 10(6), 1994, pp. 728–739.

[9] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, W. Burgard, "Learning predictive terrain models for legged robot locomotion", *in Proc. IEEE/RSJ Int. Conf. Intell. Robots and Systems*, Nice, France, 2008, pp. 3545–3552.

[10] R. Prajoux, L. de S. F. Martins, "A walk supervisor architecture for autonomous four-legged robots embedding real-time decisionmaking", *in Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, Osaka, Japan, 1996, pp. 200–207.

[11] P. Łabęcki, A. Łopatowski, P. Skrzypczyński, "Terrain perception for a walking robot with a low-cost structured light sensor", *in Proc. 4th European Conf. on Mobile Robots*, Dubrovnik, Croatia, 2009, pp. 199–204.

[12] J. R. Rebula, P. D. Neuhaus, B. V. Bonnlander, M. J. Johnson, J. E. Pratt, "A controller for the LittleDog quadruped walking on rough terrain", *in Proc. IEEE Int. Conf. on Robotics and Automation*, Roma, Italy, 2007, pp. 1050–4729.

[13] A. Roennau, T. Kerscher, M. Ziegenmeyer, J.M. Zoellner, R. Dillmann, "Six-legged walking in rough terrain based on foot point planning", in: *Mobile Robotics: Solutions and Challenges* (O. Tosun *et al.*, eds.), Singapore, World Scientific, 2009, pp. 591–598.

[14] R. Smith, *Open dynamics engine*, `www.ode.org`. September 2009.

[15] K. Walas, D. Belter, A. Kasiński, "Control and environment sensing system for a six-legged robot", *Journal of Automation, Mobile Robotics & Intelligent Systems*, vol. 2(3), 2008, pp. 26–31.

[16] E. Weisstein, "Least Squares Fitting–Polynomial", *From MathWorld – A Wolfram Web Resource*. http://mathworld.wolfram.com.

[1] A video is available on http://lrm.cie.put.poznan.pl/icra2010.avi