Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Integrated Motion Planning for a Hexapod Robot Walking on Rough Terrain

Dominik Belter * **Piotr Skrzypczyński** *

* *Institute of Control and Information Engineering, Poznan University of Technology (e-mail: {Dominik.Belter, Piotr.Skrzypczynski}@put.poznan.pl).*

Abstract: Missions of walking robots in distant areas require use of the teleoperation mode. However, the capabilities of a human operator to sense the terrain and to control the robot are limited. Thus, a walking robot should have enough autonomy to take an advantage of its high locomotion capabilities in spite of a limited feedback from the remote operator. This paper presents a method for real-time motion planning on a rugged terrain. The proposed method employs several modules for planning the robot's path and trajectories of the feet, foothold selection, collision avoidance, and stability analysis. By using this method the robot can autonomously find a path to the desired position and discriminate between traversable and non-traversable areas. The rapidly exploring random trees concept is used as a backbone of the proposed solution. Results of simulations and experiments on the real robot are presented.

Keywords: walking robot, path planning, map building, robot navigation, robust walking

1. INTRODUCTION

It is not possible for a remote operator to control a walking robot by defining the movement of each leg. The robot should have a goal which can be defined by the operator, but gait generation and coordination of the legs have to be solved on-board. When the robot is walking on a rough, demanding terrain the task becomes much harder, and the control system of the robot should provide more autonomous decision capabilities. The main tasks of the control system are: foothold selection, generation of the feet trajectories, and global path planning for the robot's trunk. The obtained path should be executable, which means that the kinematic constraints, the stability maintenance criteria, and any possible collisions between parts of the robot have to be taken into account.

The number of combinations of the possible robot's postures and legs' movements in a multi-legged robot makes the motion planning task difficult and computationally expensive. Therefore, some researchers try to simplify the problem. Such a radical simplification can be achieved by adapting existing 2D planning techniques to operate in 3D terrain, and then by following the obtained path using a reactive controller (Wooden et al. (2010)). Similarly, Bai and Low (2001) mark obstacles on a legged robot's path as inaccessible or traversable and use a potential field algorithm for path planning. These approaches may generate quite conservative motion plans, which do not take full advantage of the locomotion capabilities of a legged robot. Another way of addressing the issue of computational complexity is presented by Kolter et al. (2008), who at first plan a path of the LittleDog robot's trunk without considering the trajectories of its feet. Then, the footholds are planned to follow the initial path while considering appropriate constraints. This approach restricts the huge search space, but also may result in motion plans that do not consider all possible paths, as some solutions could be eliminated in an early stage of planning.

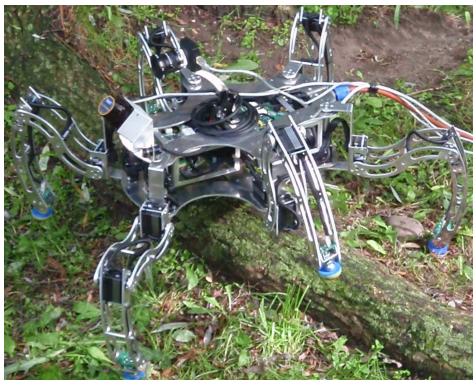

Fig. 1. The Messor robot

If more computing power is available, more complicated motion planning strategies may be considered. A recent work (Vernaza et al. (2009)) presents a search-based planning approach for a quadruped walking on rough terrain. The motion planning problem is decomposed into two main stages: an initial global planning stage, which results in a footstep trajectory, and an execution stage, which generates trajectories of the joints that enable the robot to follow the footstep trajectory. Because the space of footstep trajectory is still high-dimensional the R* algorithm, which combines aspects of deterministic and randomized search, is used for planning. Kalakrishnan et al. (2010) apply a yet more extended hierarchy of planning and optimization methods, which works well for a demanding case of a quadruped robot walking dynamically on a rough terrain. However, such an approach increases the complexity of the whole control architecture. The high-dimensionality issue in motion planning for legged robots

may be also tackled by employing a randomized planning algorithm. In particular, Hauser et al. (2006) successfully used the Probabilistic Roadmaps technique to plan motion of a six-legged lunar robot.

The approach presented in this paper falls into the same general category of randomized planners, using the Rapidly exploring Random Trees (RRT) concept (LaValle (1998)) as a general framework. The idea of RRT was successfully used abroad range of planning problems in robotics LaValle and Kuffner (2001). The proposed solution avoids decomposition of the planning problem into several stages. Instead, the issues specific to a walking robot, e.g. foothold selection and stability maintenance are solved by specialized sub-routines within the main RRT-based planner. Thus, this approach is called "integrated". The solution is verified on the six-legged Messor robot (Fig. 1), however, it can be used on any multi-legged robot that walks statically stable. The method allows for real-time planning of the movement on a rough terrain. A grid-based elevation map of the surrounding terrain is used to make a traversability assessment. The terrain profile in front of the robot is measured on-line by using the laser scanner URG-04LX. While moving the robot scans the terrain and updates a local elevation map which is then used to plan the further movements (Belter and Skrzypczyński (2010b)).

2. OUTLINE OF THE MOTION PLANNING ALGORITHM

The proposed method is based on the RRT-Connect algorithm (Kuffner and LaValle (2000)). The rapidly exploring random trees concept, and its RRT-Connect implementation are used here as a meta-algorithm, which invokes several lower-level planning modules (Alg. 1). The particular sub-procedures of this meta-algorithm are explained further in the paper. The RRT-Connect algorithm creates two random trees. Each node in a tree represents a kinematic configuration (pose and posture) of the robot. We impose only the kinematic constraints because the robot uses a static gait, however the RTT can handle kinodynamic constraints as well LaValle and Kuffner (2001). A child node is connected to the parent node if a transition between these two configurations is possible. The root configuration of the first tree T_a is located in the initial configuration of the robot. The root configuration of the second tree T_b defines a goal state of the robot. For the T_a tree the robot moves forward, whereas the direction in the T_b tree is inverted (the robot moves backward).

Algorithm 1.

```
      procedure RRT_Motion_Planning(q_init,q_goal)
1       T_a.Init(q_init); T_b.Init(q_init)
2       for k:=1 to K do
3         begin
4           q_rand:=Random_Config;
5           if not (q_new:=Extend(T_a,q_rand))=Trapped
6             if Extend(T_b,q_new)=Reached
7               return Path(T_a,T_b)
8           Swap(T_a,T_b);
9         end;
```

At first, the procedure Random_Config randomly selects a position of the robot q_{rand} (only the x and y coordinates) on the given elevation map. Then the Extend operation extends the tree and tries to build new branch in the direction of q_{rand}. If it is possible, and the new node q_{new} is created, the algorithm extends the second tree in the direction of q_{new} node. In the next iteration of the algorithm the order of the trees is swapped (Swap procedure). As a result the T_b tree is extended at the beginning of the next iteration. The maximal number of iterations is defined by K.

When the two trees are connected and the Extend procedure returns "Reached" value the algorithm finds the shortest path between q_{init} and q_{goal}. The q_{new} configuration is common for the two trees. The Path procedure finds the path from q_{new} to the root nodes in both trees. The sequence of the nodes for T_b tree should be inverted. Finally, the sequence of configurations which allows to move from the initial to the final location is obtained.

3. MAIN MODULES OF THE MOTION PLANNER

3.1 Planning the Posture of the Robot

For a wheeled or tracked robot the given kinematic configuration is achievable on a rough terrain if the inclination of the robot is inside the defined range – e.g. the maximal pitch angle is defined as 30°. As a result, the robot avoids paths on slopes bigger than the defined maximum. Although such a simple approach was used with the RRT concept, e.g. on the Lunar and Mars rovers (Kubota et al. (2001)), for a walking robot the mentioned conditions are not sufficient. First, to assume that the desired position of a walking robot is achievable the robot should find appropriate footholds there. Second, a certain clearance between the robot's trunk and the ground should be maintained to avoid collisions. Therefore, the local shape of the terrain implies the configuration of the robot.

The module responsible for maintaining an acceptable posture creates the robot's configuration at the given (x, y) location on the map. It computes the elevation and inclination of the robot's trunk. These data are defined in the robot's coordinates. The pitch angle of the trunk is defined by the local slope of the terrain along the y axis, and the roll angle is defined by the slope of the terrain along the x axis. The slope is computed by using the elevation map describing the surface under the platform of the robot. To compute the roll and pitch angles a robust regression procedure is applied. As a result, the average inclination of the ground along the x and y axes is obtained. The trunk's inclination is taken into account when computing the distance of the robot's platform to the ground. The minimal distance from the bottom of the trunk to the highest point of the terrain under the robot should fulfill the requirement given as:

$$D^{\Pi}_{\min} = d^{\text{body}}_{\max}, \tag{1}$$

where: $D^{\Pi}_{\min}$ is the minimal distance between the plane representing the trunk and the ground, and $d^{\text{body}}_{\max}$ is the acceptable clearance parameter for the robot's body.

3.2 Foothold Selection Algorithm

At a given posture of the body the robot has to find appropriate footholds. The algorithm proposed by Belter

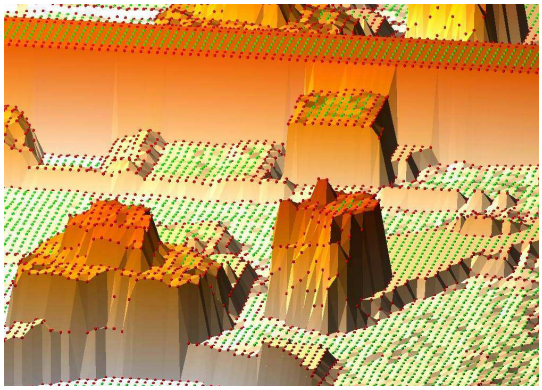

Fig. 2. Results obtained by using the proposed foothold selection algorithm (green – acceptable footholds, red – unacceptable footholds)

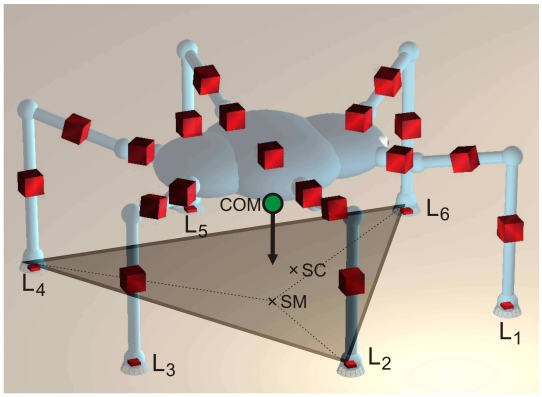

Fig. 3. Static stability checking procedure

et al. (2010) is used. The aim of this algorithm is to find an analytical relation between the K_1 and K_2 coefficients, being some simple characteristics of the local shape of the terrain, and the predicted slippage of the robot's feet. The predicted slippage is a slippage which is expected when the robot choses the considered point of the elevation map.

As shown in (Belter et al. (2010)) the robot learns unsupervised how to determine the footholds. Then it exerts the obtained experience to predict the slippage and minimize it by choosing proper points from the map. It takes 15 ms to find the best foothold for a single leg by using the obtained analytical relation.

In the presented system the algorithm from (Belter et al. (2010)) is slightly modified. The final quality Q_{final} of a candidate foothold is defined by the predicted slippage Q of several neighboring points:

$$Q_{\text{final}}(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} Q_{i+k,j+l}, \tag{2}$$

where $z_{i,j}$ and $z_{i+k,j+l}$ are cells in the elevation map. As a result, the robot prefers points surrounded by neighbors which are also assessed as good footholds. The equation (2) is used to minimize the influence of the localization errors. When the measured position differs from the real position the robot actually places its feet on points located next to the selected footholds. When the neighboring points are also good footholds, the risk of falling down decreases.

Finally, to determine if the considered footholds are achievable the robot uses information about the workspace of the legs. It prefers the points which are located far from the border of the given leg's workspace. This strategy prevents deadlocks – situations when the robot can't move its legs due to kinematic constraints. Example results from the foothold selection module are presented in Fig. 2.

3.3 Static Stability Checking Procedure

A stability criterion is rarely used for motion planning of wheeled robots. Keeping the vehicle's inclination below a certain value is sufficient to preserve stability of a conventional wheeled robot. However, the static stability is crucial for legged robots. Therefore a static stability checking procedure is implemented to preserve balance and avoid unexpected falls.

There is a number of stability criteria for legged robots e.g. stability margin, tumble stability margin, gradient, stability margin, energy stability margin, etc. (Hirose et al. (2001)). There is also a criterion which takes into account the friction coefficient between robot's feet and the ground (Bretl and Lall (2008)). Although it allows to precisely define the support polygon it can't be used for motion planning. The procedure is iterative and the computation cost is high. Because of that a fast, basic stability criterion defined by McGhee and Iswandhi (1979) is used in the presented motion planning system to check the balance for every planned posture of the robot.

The stability checking procedure is shown in Fig. 3. At the beginning the centers of mass of all the robot's joints are computed. To compute the center of mass (COM) of the robot the instantaneous configuration of the legs and the trunk is used. The robot posture is taken into account during COM computation because a significant part of the robot's mass is allocated in the legs. Thus, a modification of the position of the legs significantly changes the COM position. Then a projection of the COM on the plane (SM) is computed. If the robot is statically stable, the SM point is located inside the convex hull formed by the contact points of the legs being in the stance phase.

To check if the SM point is located inside the support polygon with L_2, L_4, L_6 vertexes (an example for the tripod gait, cf. Fig. 3) the areas of the component triangles ($\triangle L_2L_4SM, \triangle L_4L_6SM, \triangle L_6L_2SM,$) are computed. If the sum of areas equals the area of the support polygon, the SM point is inside the support polygon, and the robot is statically stable. If the sum of the areas is bigger than the area of the support polygon the SM point is outside the support polygon and the robot is not statically stable.

This static stability checking procedure is fast but approximate. In practice, to avoid risky postures, the area of the support polygon is reduced by relocating the leg contact points towards the center of the support polygon.

3.4 Determining the Workspace of the Legs

To check if a desired foothold is reachable, the robot has to verify if the position of the considered foot is located inside the workspace of the given leg. There are known

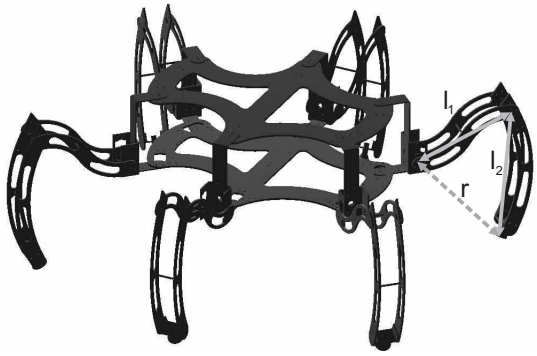

Fig. 4. CAD model of the Messor robot used for collision detection

methods to determine shape of the workspace of a robot's leg (Cavusoglu et al. (2001)). These procedures are time-consuming, hence a simpler solution is used here, which is however specific to the configuration of the Messor robot.

The kinematic configuration of a leg of the Messor robot is shown in Fig. 4. If the desired point is outside the workspace the distance r is bigger than the sum $l_1 + l_2$. As a result, the inverse kinematic computations fails yielding $|cos(\theta_3)| > 1$ or trying to compute $\sqrt{x}$ for $x < 0$. Such a situation is easy to detect and it indicates that the desired foot position is outside the workspace.

3.5 Collision Detection Module

The module which detects movements that are outside the leg's workspace is not sufficient for the proper motion planning. Legs of the robot can collide if a selected foot position is inside of the workspace of the other legs. Moreover, the trunk of the robot or the legs might collide with the ground. To detect such situations the robot is equipped with a special module for collision detection. This module uses a CAD model of the robot's frame, which is shown in Fig. 4.

The CAD model is constructed from a number of triangles. When two bodies collide, there is a line which is common for two triangles. To detect a collision the triangle-triangle intersection test for oriented bounding box (OBB) algorithm (Möller (1997)) is used. It is fast and reliable when it is used to detect collision between robot's parts (the model is fixed, only its configuration may vary). However, collision detection between parts of the robot and the ground is rather slow, because the ground model changes during walking on an uneven terrain. On the other hand, the foothold selection module is in most cases sufficient to prevent collisions with the ground. Eventually, the collision detection module is used only to prevent collisions between parts of the robot.

4. TRANSITION BETWEEN STATES IN THE MOTION PLANNING

4.1 Trajectory Planner for the Feet

In the RRT algorithm (cf. Alg. 1) the EXTEND operation checks if a transition between two given states (i.e. configurations of the robot) is possible. For a wheeled robot it is sufficient to check the change of terrain elevation, and

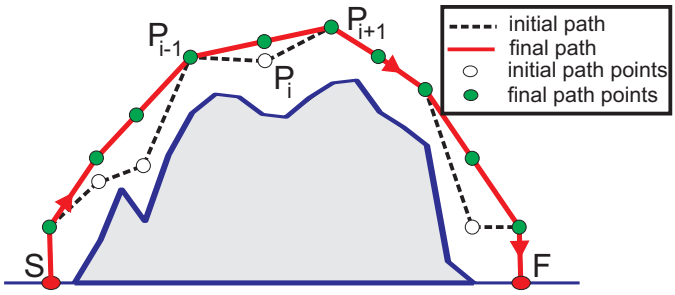

Fig. 5. Procedure for planning the foot trajectory

the change of inclination of the vehicle in order to determine if a robot can reach the desired position (Ettlin and Bleuler (2006)). However, for a legged robot checking these conditions is insufficient. The algorithm has to determine if the robot is able to move its legs in such a way, that the goal state is reached, e.g. by moving the legs above an obstacle. Therefore, a fast trajectory planner for the feet of the robot is required (Fig. 5). The trajectory is planned for a single foot. The initial position of the foot is S, the desired foothold is marked as F. At the beginning an initial trajectory is created. It represents the shape of the terrain on the straight path from S to F. The clearance between the via-points of this trajectory and the ground is set to $d_{\min}^{\text{foot}}$ to ensure a safety margin. However, the initial trajectory is typically quite long, because it unnecessary follows the shape of the terrain profile in all depressions.

The trajectory can be shortened by using a simple algorithm which creates a convex hull over the initial path. The algorithm finds a straight line which intersect the first point of the initial trajectory and the following points, and has the largest inclination. The line that is bounded by the two end-points of the trajectory is a part of a new trajectory. This procedure is repeated for the last point of the new trajectory until the point F is added.

In the transition between two states the path of the robot's trunk is determined as a straight line. The orientation (yaw, pitch and roll angles) changes linearly as well. When the trunk's path and the trajectories of the feet are determined, the result is verified by using the collision detection module, the static stability checking procedure, and the unit for determining the workspace of the legs. If the desired positions of the legs are outside their workspaces or the desired posture is not achievable because of collisions the transition is not possible.

4.2 The Extend Operation

Finally, the EXTEND operation in the presented version of the RRT-Connect algorithm (Alg. 2) is defined. At the beginning the NEAREST_NEIGHBOUR operation finds the existing node which is the closest one to the new, random position q. Then, a new configuration is created by using the module which selects footholds and creates a posture (CREATE_POSTURE procedure). Then, the procedure CHECK_TRAVERSE verifies if the transition from q to q_{near} is possible. At first, trajectories for the feet and a path for the trunk are created. Then, the robot checks if the desired sequence of the robot's postures is achievable (lack of collisions, feet positions inside the workspaces) and safe (robot maintains the static stability). If a transition to the

position is impossible, another position q which is however closer to the q_{near} is created by the REDUCE_DISTANCE procedure. The maximal number of iterations is N. To make the EXTEND procedure faster the Cartesian distance d_q between q and q_{new} is computed. If the distance d_q is bigger than the maximal possible step $d_{\max}$ of the robot, the distance d_q is initially reduced to $d_{\max}$.

Algorithm 2.

```
      procedure EXTEND(T,q)
1       q_near:=NEAREST_NEIGHBOUR(q)
2       for k:=1 to K do
3         begin
4           CREATE_POSTURE(q);
5           if CHECK_TRAVERSE(q,q_near)=true
6             if k=1
7               return q, Reached;
8             else
9               return q, Advanced;
10          REDUCE_DISTANCE(q,q_near)
11        end;
12      return Trapped;
```

The described planning procedure results in a sequence of configurations of the robot. In each node of this sequence the robot uses basic kinematic relations to generate a movement that results in the next desired configuration. However, using only the position-based controller is risky on a rugged terrain, when internal representation of the environment is not accurate. Therefore, the robot uses the force sensors in its feet to make the legs complaint. They are used to detect collisions with the ground and to stop the movement when support of the terrain is sufficient. This approach makes execution of the movement obtained from the RRT_MOTION_PLANNING procedure much more robust to any inaccuracies in perception and modelling of the terrain, as well as to the unavoidable uncertainty in actuation of the legs.

5. RESULTS

The described motion planning method has been verified in a realistic simulator (Belter and Skrzypczyński (2010a)) of the Messor robot. A specially prepared terrain model has been used. It includes obstacles similar to scattered flagstones, extensive hills with mild slopes, small and pointed hills, depressions and stones. This terrain model includes also a map of a real rocky terrain acquired using the URG-04LX sensor (Belter and Skrzypczyński (2010b)). Additionally, it includes a non-traversable hill and two walls 35 cm and 15 cm high. The second wall is traversable to the Messor robot.

An example of path obtained with the presented motion planning method is shown in Fig. 6. A path to the goal position is found. Some of the RRT branches explore parts of the terrain that could be labelled as "dead ends". Eventually, these routes are omitted, because of the non-traversable obstacles at the end of the path. What is important, the robot passes by the sloppy hill, but it finds a path which allows to traverse the 15 cm high step. Generation of the whole motion sequence takes about 20s. The time consumed for path generation highly depends on the terrain shape. If there are no high obstacles and "dead ends", the result is obtained in few seconds. To check the

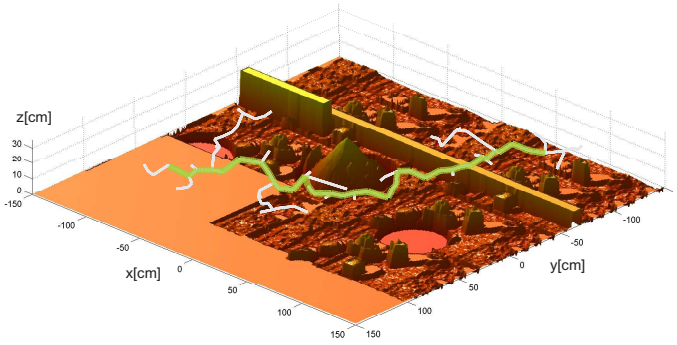

Fig. 6. Path obtained in the experiment (green – final path; white – RRT branches)

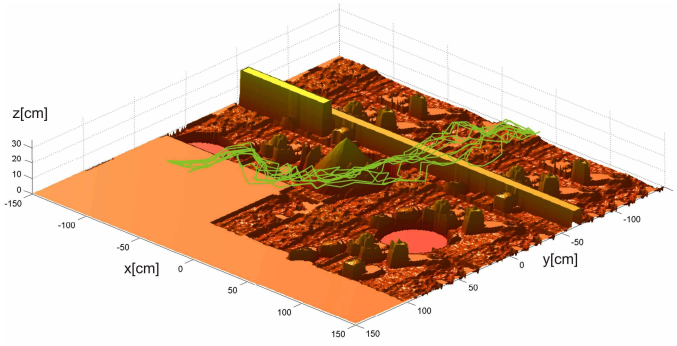

Fig. 7. Results of a series of experiments

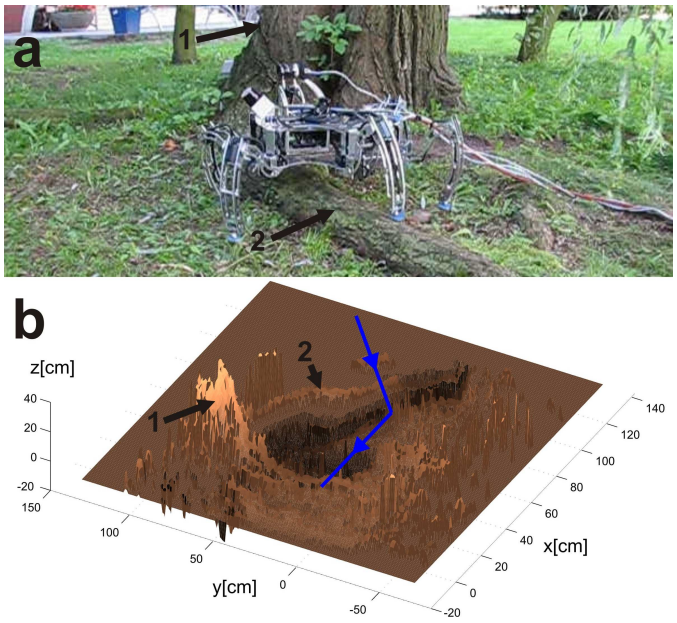

Fig. 8. Outdoor experiment (a) and the obtained elevation map (b)

effectiveness of the algorithm a series of ten experiments has been conducted. In all the experiments the appropriate path was found. The results are shown in Fig 7.

Some verification experiments were also conducted on a rough terrain mockup and outdoors. In these experiments the aim was to test the particular motion planning modules, without the RRT-based meta-algorithm, so the robot was tele-operated. The desired path of the robot's trunk

was defined by a remote operator. The robot scanned the terrain and created an elevation map. Then, it used the previously described modules to plan the footholds, find the trajectories of the feet, and verify achievability of the movements. All these modules were working in real time. The outdoor experiment and the obtained elevation map are shown in Fig. 8.

6. CONCLUSIONS

This paper describes a method for planning motion of a six-legged walking robot. The presented procedures allow to determine the traversable areas, and to find the appropriate path to the goal position automatically. Collisions be5tween parts of the robot are detected. The robot avoids collisions with the ground, deadlocks, unstable postures and carefully selects footholds. The obtained path is executed in a robust way by using the compliance of the legs. While walking the robot creates an elevation map of the terrain, which is then used in further planning. The presented method is implemented as modular software, allowing for modification of the particular modules within the RRT-based framework. For example various stability criteria or different foothold selection algorithms can be used. The results are shown by a movie clip available at http://lrm.cie.put.poznan.pl/ifac2010DBPS.wmv.

Although the proposed method is used mainly with wave and tripod gaits, it allows to plan the movement using any kind of cyclic gait. The sequence of swing and stance phase is defined in the procedure for planning trajectories of the feet. In further experiments it will be shown that the method can be used with a free gait as well.

Currently, the robot is using legged odometry to obtain a position estimate. Further improvements include use of a visual SLAM module for robot localization. It should improve quality of the obtained elevation map. The URG-04LX sensor on Messor scans the terrain about 80 cm in front of the robot. This distance is too small to plan an effective path of the robot's trunk on-line, while perceiving the terrain. Hence, it is planned to use a stereo camera to create a more extended representation of the terrain, which should allow for planning in a much distant horizon.

ACKNOWLEDGEMENTS

The authors would like to thank Miłosz Matelski for his work on collision detection module, Krzysztof Walas for his support in experiments on the real robot & the CAD model, and Tomasz Gajewski for his initial work on the RRT implementation.

REFERENCES

Bai, S. and Low, K. (2001). Terrain evaluation and its application to path planning for walking machines. *Advanced Robotics*, 15(9), 729–748.

Belter, D., Łabecki, P., and Skrzypczyński, P. (2010). Map-based adaptive foothold planning for unstructured terrain walking. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 5256–5261. Anchorage, USA.

Belter, D. and Skrzypczyński, P. (2010a). A biologically inspired approach to feasible gait learning for a hexapod robot. *Int. Journal of Applied Mathematics and Computer Science*, 20(1), 69–84.

Belter, D. and Skrzypczyński, P. (2010b). Rough terrain mapping and classification for foothold selection in a walking robot. In *Proc. IEEE Int. Workshop on Safety, Security and Rescue Robotics*, CD–ROM. Bremen, Germany.

Bretl, T. and Lall, S. (2008). Testing static equilibrium for legged robots. *IEEE Transactions on Robotics*, 24(4), 794–807.

Cavusoglu, M.C., Villanueva, I., and Tendick, F. (2001). Workspace analysis of robotic manipulators for a teleoperated suturing task. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2234–2239. Maui, USA.

Ettlin, A. and Bleuler, H. (2006). Randomised rough-terrain robot motion planning. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 5798–5803. Beijing, China.

Hauser, K., Bretl, T., Latombe, J.C., and Wilcox, D. (2006). Motion planning for a six-legged lunar robot. In *7th Int. Workshop on the Algorithmic Foundations of Robotics*, 301–316. New York, USA.

Hirose, S., Tsukagoshi, H., and Yoneda, K. (2001). Normalized energy stability margin and its contour of walking vehicles on rough terrain. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 181–186. Seoul, Korea.

Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., and Schaal, S. (2010). Fast, robust quadruped locomotion over challenging terrain. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 1050–4729. Anchorage, USA.

Kolter, J.Z., Rodgers, M.P., and Ng, A.Y. (2008). A control architecture for quadruped locomotion over rough terrain. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 811–818. Pasadena, USA.

Kubota, T., Kuroda, Y., Kunii, Y., and Yoshimitsu, T. (2001). Path planning for newly developed microrover. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 3710–3715. Seoul, Korea.

Kuffner, J.J. and LaValle, S.M. (2000). RRT-Connect: An efficient approach to single-query path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 995–1001. San Francisco, USA.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Iowa State University, USA.

LaValle, S.M. and Kuffner, J.J. (2001). Rapidly-exploring random trees: Progress and prospects. *Algorithmic and Computational Robotics: New Directions,* B. R. Donald et al. (Eds.), A K Peters, Wellesley, 293–308.

McGhee, R.B. and Iswandhi, G. (1979). Adaptive locomotion for a multilegged robot over rough terrain. *IEEE Trans. on Syst., Man, and Cyb.*, SMC-9(4), 176–182.

Möller, T. (1997). A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2, 25–30.

Vernaza, P., Likhachev, M., Bhattacharya, S., Chitta, S., Kushleyev, A., and Lee, D. (2009). Search-based planning for a legged robot over rough terrain. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2380–2387. Kobe, Japan.

Wooden, D., Malchano, M., Blankenspoor, K., Howard, A., Rizzi, A., and Raibert, M. (2010). Autonomous navigation for BigDog. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 4736–4741. Anchorage, USA.