Int. J. Appl. Math. Comput. Sci., 2011, Vol. 21, No. 2, 363–377
DOI: 10.2478/v10006-011-0027-9

SUPPORTING LOCOMOTIVE FUNCTIONS OF A SIX–LEGGED WALKING ROBOT

KRZYSZTOF WALAS, DOMINIK BELTER

Institute of Control and Information Engineering
Poznań University of Technology, ul. Piotrowo 3A, 60–965 Poznań, Poland
e-mail:{Krzysztof.Walas,Dominik.Belter}@put.poznan.pl

This paper presents a method for building a foothold selection module as well as methods for the stability check for a multi-legged walking robot. The foothold selection decision maker is shaped automatically, without expert knowledge. The robot learns how to select appropriate footholds by walking on rough terrain or by testing ground primitives. The gathered knowledge is then used to find a relation between slippages and the obtained local shape of the terrain, which is further employed to assess potential footholds. A new approach to function approximation is proposed. It uses the least-squares fitting method, the Kolmogorov theorem and population-based optimization algorithms. A strategy for re-learning is proposed. The role of the decision support unit in the control system of the robot is presented. The importance of the stability check procedure is shown. A method of finding the stability region is described. Further improvements in the stability check procedure due to taking into account kinematic correction are reported. A description of the system for calculating static stability on-line is given. Methods for measuring stance forces are described. The measurement of stance forces facilitates the extended stability check procedure. The correctness of the method is proved by results obtained in a real environment on a real robot.

Keywords: robotics, walking robots, foothold placement, static stability.

1. Introduction

1.1. Taxonomy of locomotion tasks for a walking robot. A six-legged walking robot, due to its stability and kinematic redundancy, has better locomotive abilities than wheeled mobile robots and also better performance compared with walking robots with two or four legs. A six-legged robot is able to walk on rough terrain with obstacles as high as its leg's last segment. Thanks to its kinematic redundancy, it is also able to climb obstacles which exhibit the local height difference as high as the robot's stretched leg. The movement on rough terrain should be fast. The problem of foot placement has to be solved in order to avoid slippages or being stuck in local terrain concavities. Traversing high obstacles is similar to climbing strategies and thus has deliberate character. Planning each next step is strongly dependent on the current robot pose and the prediction of robot stability in the target pose. As can be seen, two locomotion tasks of different character are involved. Each of them requires different planning methods. The locomotion system architecture for a walking robot can be seen in Fig. 1

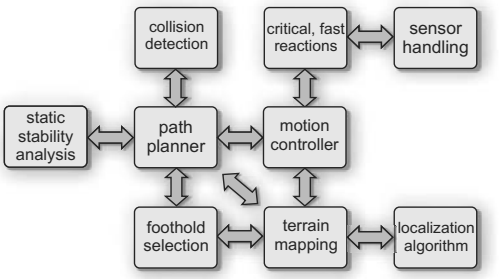

Fig. 1. Control architecture.

1.2. State of the art. To create an efficient control algorithm for a robot walking on rough terrain, one should deal with many substantial problems. At the beginning, the robot should build a proper model of the environment. The Lauron robot is an example of a walking machine which is able to acquire a map of an unstructured environment (Gassmann *et al.*, 2003; Roennau *et al.*, 2009). It uses both occupancy and credibility maps in order to make decisions about appropriate footholds. The chosen

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 expected for "normal" walking behavior). Another robot which can acquire a map of the surrounding terrain is LittleDog (Kolter *et al*., 2009). Here a different approach to the foothold selection problem was implemented (Rebula *et al*., 2007). The robot was equipped with a terrain scorer/classifier. It judges a potential foot placement at a given point as acceptable or unacceptable, by taking into account the height of the point considered and the height of the neighboring ones. The scorer rejects the points which are located on too excessive slopes or are too close to the top edge or the base of a cliff, or are inside of a hole. The robot uses a previously acquired map of the surroundings to compute a fine grid of the scored terrain. An interesting approach to the foothold selection problem was presented by Kalakrishnan *et al*. (2009). They proposed to use terrain templates and supervised learning from expert demonstrations. This allows us to obtain a complex ranking function which improves the walking performance. This function is generic and can be applied to a previously unvisited terrain.

Another problem which should be taken into account is the role of the foothold selection algorithm in the control system architecture. The foothold planner can work as part of a higher-level planner (Kolter *et al*., 2008). It produces a set of footsteps on the terrain. The low-level planner takes into account the foothold sequence to generate desired paths of the legs and of the robot's center of gravity.

The appropriate structure of the control system should contain an efficient motion controller unit. The obtained footstep sequence is used for motion planning of the robot (Vernaza *et al*., 2009). Afterwards, the robot executes the joints trajectories to follow the desired footstep sequence. Robustness during movement execution can be provided by using force control (Schmucker *et al*., 2003). Compliant legs of a robot yield automatic adaptation to small irregularities on the surface while walking on a rough terrain.

A slightly different problem is posed by the negotiation of higher obstacles such as stairs. Much research in this field was conducted for biped robots. The use of force sensors and accelerometers while climbing stairs was described by Li *et al*. (2007). Another approach was reported by Gutmann *et al*. (2004), who used a stereo vision system as the main state sensor. Bipedal robots are able to move in the horizontal direction, but a big problem is their poor stability while walking, as they can easily fall down if they encounter an obstacle on the stairs.

Walking robots with six or more legs exhibit a considerable static stability in transient states, which prevents them from falling down in the case of obstacles on a staircase. Nevertheless, there is still a need for performing the stability check procedure which is based on the position of the robot Center of Mass (CM). Early research in this field was done in the 1980s and 1990s, and it is described by Kumar and Waldron (1988) as well as Barghava and Waldron (1988). Further results in this area were reported by Gonzalez *et al*. (2005), whose stability margin check included joints torques and limited power. The latest research is focused on testing the static equilibrium on-line by using optimization methods (Bretl and Lall, 2006; 2008).

The solution proposed by Bretl and Lall (2008) was taken as the starting point in our research. The method of finding a support region was adapted to the case of a six-legged walking robot. The specification of the problem led to the simplification of computations, thus enabling a real time operation on a real robot.

The modules presented in the paper are part of a locomotion system architecture. They support the main module in motion planning of the robot body. Terrain evaluation and the robot mobility should be taken into account during trajectory planning and gait generation (Bai and Low, 2001). The efficiency of walking on a rough terrain can be improved by appropriate gait generation (Bai *et al*., 1999). During the execution of the planned movement the robot should distribute the body force to the feet to prevent leg slippages. This could be achieved by applying active force control (Zhou *et al*., 2000).

2. Adaptive foothold selection problem

2.1. Adapting a walk to uneven terrain. Walking on an uneven terrain requires selecting an appropriate gait type. For the proposed task, the tripod gait was chosen (Belter *et al.*, 2008). However, the results are applicable for wave and free gait. We decided to verify the algorithm on the most demanding type of gait. The tripod gait is characterized by the smallest support region. As a result, walking on rough terrain is more demanding according to the stability criterion. On the other hand, the robot has the potential to cover a particular distance in shorter time, which is important in search and rescue missions. The wave and free gaits were rejected due to low maximal speeds of walking in these modes.

The problem of walking on rough terrain is similar to that of multi-finger grasping. The are two approaches to this problem. The first one is based on the form-closure principle, while the second one exploits constraints resulting from closure forces. Form-closure constraints result from geometric properties of a set of contact constraints between feet and the terrain. Force-closure is due to the forces which are transmitted through the contact points. The use of force-closure constraints requires the knowledge of a friction coefficient. It is difficult to deduce such information. Additionally, the friction of the terrain may differ locally while walking on a rocky ground. For that reason, the robot controller uses only geometrical properties of the terrain to plan the footholds. Moreover, such a

method is independent of the friction coefficient value. It works properly for terrain with varying values of friction coefficients but does not take into account dynamic terrain such as moving rocks.

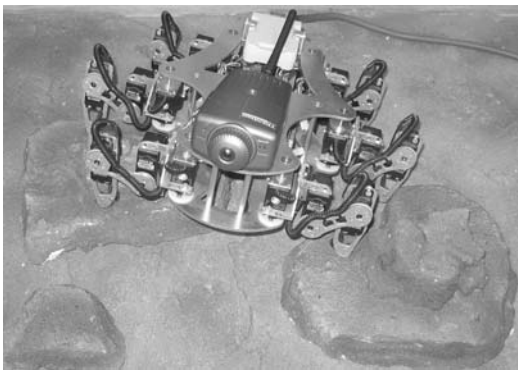

Fig. 2. Ragno robot during tests on a rough terrain mockup.

2.2. Robot and the simulator. The Ragno hexapod walking robot (cf. Fig. 2) is used in this research. Each leg has three joints that are driven by integrated servomotors. The robot weighs 2.155 kg without batteries and fits in a box of 33×30 cm. Its mechanical structure allows for walking with the trunk elevated from 15 to 70 mm over the ground. The robot is equipped with various types of sensors including an Inertial Measurements Unit (IMU) to measure the roll and pitch angles. The reader can find more information in the paper by Walas *et al*. (2008). What is important, the results are also verified on a bigger robot, Messor, with a similar feet shape.

The simulator of the Ragno robot is based on the Open Dynamics Engine (ODE). This library allows for simulating rigid body dynamics. It provides methods for modeling various joints with friction and softness, and for detecting collisions (Smith, 2007). The parameters of the simulated model are optimized by using the evolutionary algorithm, which minimizes discrepancies between the real and the simulated robot (Belter and Skrzypczyński, 2009). This simulator has been already successfully used to evolve feasible gaits for the Ragno robot (Belter *et al.*, 2008).

2.3. Control strategy for walking on uneven terrain. During the stance phase the robot maintains a constant average height of the trunk above the ground. This average height is computed by using position coordinates of the last six footholds coordinates. To stabilize the trunk orientation during the walk, the information about pitch and roll angles from the IMU is used.

At the end of the swing phase of the supporting legs, the robot searches for the best positions of three new footholds. At first, it checks whether the potential footholds

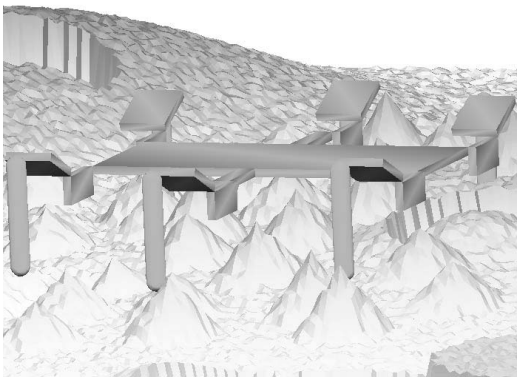

Fig. 3. Simulated robot.

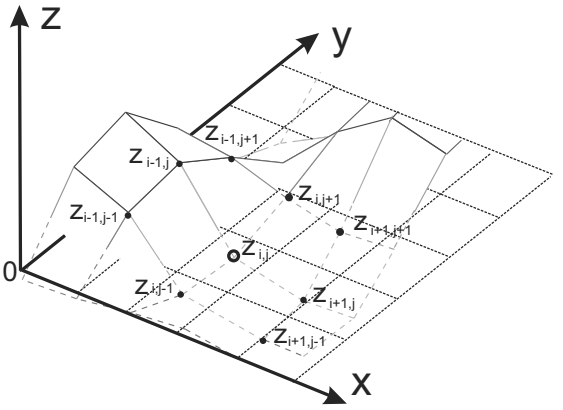

Fig. 4. Computation of the coefficients K_1 and K_2.

are in the workspace of the given leg. If there is no feasible foothold in the reach of one of the legs, the robot changes the height of its trunk to reach a proper support point. If this strategy fails, the control system informs the path planner module that the planned movement is not feasible. The problem is solved by higher layers of the control system, which are not considered in this paper. However, in the experiments presented further, the reactions which change the robot's posture were sufficient to deal with the problem.

2.4. Ground properties. The presented algorithm uses a known grid map of the surroundings and computes four coefficients which describe the geometric properties of potential footholds. For the $[i, j]$ grid coordinates, the coefficient K_1 is defined as follows:

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

where $z_{i,j}$ and $z_{i+k,j+l}$ are terrain elevations of the corresponding points from the grid map (Fig. 4). This coefficient allows for detecting a top edge or a hole. For a top edge, K_1 is positive, and for a hole, K_1 is negative. The

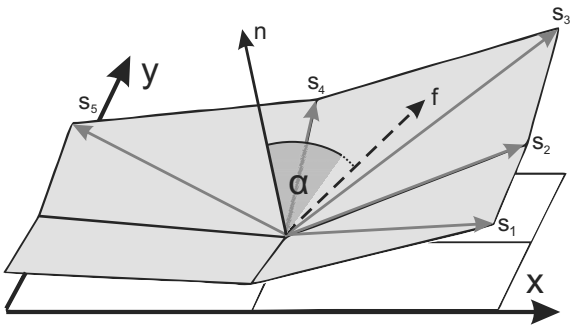

Fig. 5. Computation of the K_3 coefficient.

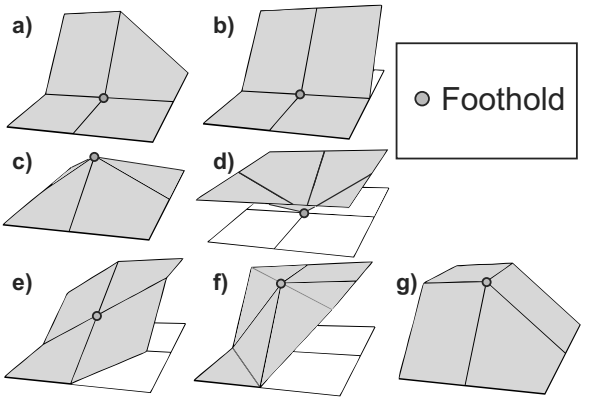

Fig. 6. Ground primitives.

value of K_1 provides information about the local terrain extremes, although K_1 is ambiguous when its value is zero. A flat terrain as well as a terrain with a constant slope give the same results for K_1.

The coefficient K_2 for $[i, j]$ grid coordinates is defined as follows:

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

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

The coefficient K_3 is defined by the angle α between a vector $\mathbf{f}$ describing the foot movement with respect to the body and a vector $\mathbf{n}$ normal to the surface. The normal vector $\mathbf{n}$ is computed only for a piece of the surface as shown in Fig. 5. It depends on the projection of the vector $\mathbf{f}$ on the plane xy. Tangent vectors $\mathbf{s}_1, \ldots, \mathbf{s}_5$ are computed for points which are neighbors to the quadrant where the 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 follows:

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

Whenever the angle α is close to π, the robot's foot has good support for the movement. When α is small, then there is a high probability of a slippage.

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

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

3. Algorithm

The aim of the algorithm is to evaluate potential footholds and to select the best one. The best point for a foot stance should minimize the risk of a slippage. The slippage is defined by the difference between the positions 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, the 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 the length of the vector that defines the intended i-th leg movement with respect to the body.

To evaluate potential foot placements, a grid map of the terrain is used with 5 mm×5 mm grid cells. In the presented simulations and experiments the terrain map is given *a priori*, but this method can be used with an online terrain mapping system utilizing a laser scanner or a structured light sensor (Łabecki *et al.*, 2009). For the points of the map considered, the coefficients K_1, K_2, K_3, and K_4 are computed. The foothold selection algorithm consists of four steps (Belter, 2009):

1. collecting data,
2. learning (approximation),
3. regular operation,
4. re-learning.

3.1. Collecting data that identify the ground properties. At first, the robot collects the appropriate data for learning. Two strategies of data acquisition were investigated. In the first one, the robot walks on rough terrain,

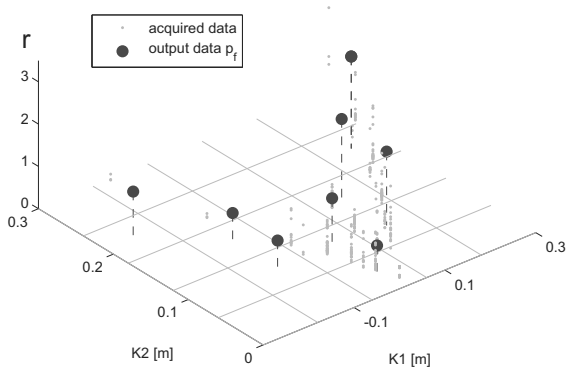

Fig. 7. Collected data and the filtration effect.

randomly selecting footholds, saving coefficients corresponding to these points and, as a result of a particular step selection, a foot slippage measured after the stance phase. In the second approach, the robot tests seven types of previously prepared "ground primitives" (shown in Fig. 6). Each of the robot's foot is placed on the same primitive. The robot executes a movement, and after that, the slippage of each foot is registered. This experiment is repeated with all primitives. To test different ground types, the height of the primitives is changed during the acquisition phase within the range of 4 cm, with a step of 1 cm. The primitives are also rotated with a step of 45°. After this stage, the robot has an appropriate data set to build a relation between the slippages and the local shape of the terrain and to distinguish between the good and the poor footholds. In most supervised learning systems the robot is taught on positive examples. The teacher shows the points of the terrain which are appropriate for a foot stance. In this system, negative examples have the same importance.

The point cloud shown in Fig. 7 was obtained in a situation when only two coefficients were used (K_1 and K_2). The primitives were used for data acquisition. Whenever the parameter space has a higher dimensionality, it is not possible to show the results graphically. It is difficult to deduce information about the usefulness of the footholds from a cloud of raw points. Very often the data are ambiguous. Selecting the same points (defined by the same coordinates K_1 and K_2) results in different slippage values. This is due to the closed kinematic chains made by the robot's legs with the ground. Footholds are evaluated locally, leg by leg, but the slippage at the selected point depends also on footholds of the remaining legs. If even a single foot skids, the forces are transmitted through the robot body, which can cause a slippage in the contact points of other feet.

3.2. Learning the decision support unit. In the second phase, the robot forms its decision support module. Learning is understood here in a broader sense, as the development of a decision support unit on the basis of the terrain characteristics. The learning phase provides an adaptation mechanism, which incorporates the knowledge gathered by the robot into its control system. To deduce useful information from the cloud of points, a discretization using a fixed grid is used. The input space K_1, K_2 and K_3 is divided into a regular grid (Fig. 7 for a situation when only K_1 and K_2 are used). For points that have projections located in the examined grid cell, a mean value is computed. The result is placed in the center of the grid cell considered. After this discretization, a group of input points $\mathbf{p}_f = [\mathbf{p}_{f1}, \ldots, \mathbf{p}_{fk}]^T$ is obtained. The output vector $\mathbf{r} = [r_1, \ldots, r_k]^T$ contains the slippage values which correspond to the respective $\mathbf{p}_{fi}$ vector of input values. For further purposes, only these points are stored. The previous point cloud is removed from the memory.

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 have not been tested explicitly in the data acquisition stage. To this end, a least-squares polynomial approximation is used (Dahlquist and Bjorck, 1974). The approximation assignment requires the selection of an appropriate polynomial base:

$$P(K_1, K_2, \ldots, K_n) = \sum_{q=0}^{m} c_q \cdot \phi_q(K_1, K_2, \ldots, K_n), \tag{6}$$

where $K_1, K_2, \ldots, K_n$ are coefficients which characterize the terrain. The polynomial P allows assessing the potential footholds represented by values of the $K_1, \ldots, K_n$ coefficients. To determine the vector $\mathbf{c}$ of c_q values, the least-squares fitting method is applied on the set of points $\mathbf{p}_f$ obtained in the data acquisition stage. The method uses the Gram matrix $\mathbf{G}$:

$$\mathbf{G} = \mathbf{V}^T\mathbf{V}, \tag{7}$$

$$\mathbf{c} = \mathbf{G}^{-1}\mathbf{V}^T\mathbf{r}, \tag{8}$$

where

$$\mathbf{V} = \begin{bmatrix} \phi_1(\mathbf{p}_{f1}) & \phi_2(\mathbf{p}_{f1}) & \ldots & \phi_m(\mathbf{p}_{f1}) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \phi_1(\mathbf{p}_{fk}) & \phi_2(\mathbf{p}_{fk}) & \ldots & \phi_m(\mathbf{p}_{fk}) \end{bmatrix} \tag{9}$$

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

Unfortunately, there are no general rules for polynomial base selection. Knowledge about the approximated phenomenon is very useful, although in this case it is not available. One possibility is to define the components ϕ_q of the polynomial base as

$$\phi_q(K_1, K_2, \ldots, K_n) = f_1(K_1) \cdot f_2(K_2) \cdot \cdots \cdot f_n(K_n), \tag{10}$$

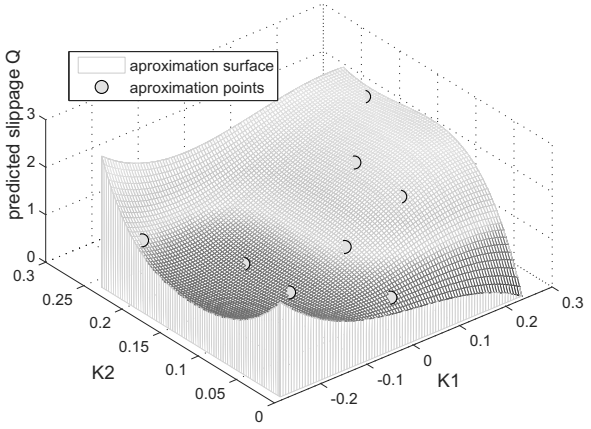

Fig. 8. Relation between slippages and the local shape of the terrain obtained by using approximation with elementary functions.

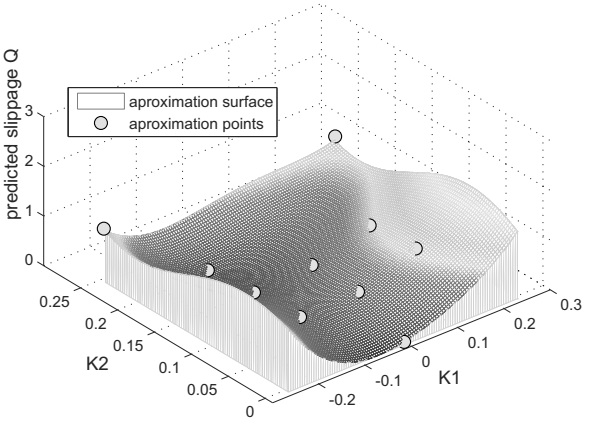

Fig. 9. Relation between slippages and the local shape of the terrain obtained by using approximation with the Gaussian function.

where the functions $f_1,\ldots,f_n$ are chosen from the following candidate elementary functions:

$$(K_i - a_{iq})^{\mathrm{round}(\lambda_{iq})}, \tag{11}$$

$$\sin(\lambda_{iq} \cdot K_i - a_{iq}), \tag{12}$$

$$e^{-\lambda_{iq}(K_i - a_{iq})}, \tag{13}$$

where the $\mathrm{round}()$ operator rounds a real number to the nearest integer.

Another possibility for the construction of the polynomial base is suggested by the Kolmogorov theorem. In this approach, only Gaussian functions are used. According to the Kolmogorov theorem, any continuous functions of several variables can be represented by the superposition of functions of one variable and by the sum of functions (Kolmogorov, 1957). Further generalizations introduced by Sprecher and Lorentz (Lorentz, 1986) led to the following form.

There exist n constants $0 < \lambda_i \leq 1$, $i = 1,\ldots,n$ and $2n+1$ functions $\phi_q(x)$, $q = 0,\ldots,2n$ defined on I and with values in I, which have the following properties:

- the functions ϕ_q are strictly increasing and belong to a class Lip(α), $\alpha > 0$;
- for each continuous function f defined on S one can find a continuous function $g(x)$, $0 \leq x \leq n$ such that

$$f(x_1,\ldots,x_n) = \sum_{q=0}^{2n} g(\lambda_1\phi_q(x_1) + \cdots + \lambda_n\phi_q(x_n)), \tag{14}$$

where $I = [0,1]$, S is an n-dimensional cube and $0 \leq x_i, \leq 1$ for $i = 1,\ldots,n$.

After substitution $\phi_q = (x_i - a_{iq})^2$ and $g(x) = c_q \cdot e^x$, the approximation polynomial (6) is defined as follows:

$$P(K_1,\ldots,K_n) = \sum_{q=0}^{m} c_q \cdot e^{\left(\sum_{i=1}^{n} \lambda_{iq}(K_i - a_{iq})^2\right)}. \tag{15}$$

To find an approximation of an n-dimensional continuous function, the algorithm searches for an appropriate sum of n-dimensional Gaussian functions. This approach gives better results than the method which uses the sum of products of elementary functions. To verify this algorithm and present the quality of approximation, the following commonly used three-input nonlinear test function was tested (Kosiński and Weigl, 1998):

$$P(K_1,\ldots,K_n) = \left(1 + \sqrt{K_1} + \frac{1}{K_2} + K_3^{-1.5}\right)^2. \tag{16}$$

For comparison, the Average Percentage Error (APE) was used:

$$APE = \frac{1}{N}\sum_{p=1}^{N} \frac{|y - \hat{y}|}{y} \cdot 100\%, \tag{17}$$

where y is the desired output and $\hat{y}$ is the real output from the system. The APE for the training data set is 0.069%, and for test data it is 0.079%.

Computational intelligence optimization methods can be applied in order to obtain the components λ_{iq} and a_{iq} of the polynomial base. In the context of this research, two of such methods were selected and tested: the population based Evolutionary Algorithm (EA) (Annunziato and Pizzuti, 2000), and the Particle Swarm Optimization (PSO) algorithm (Kennedy and Eberhart, 1995). The EA is used because it is able to self-adapt its own parameters during the search, which minimizes the number of parameters that have to be set manually. PSO is used because it works simultaneously in all dimensions.

The number of elements m in the polynomial (6) is a design parameter. It could be determined automatically, e.g., by using the Akaike information criterion (Burnhamn and Anderson, 2002). However, it is not used in this case. A fixed m makes the implementation of the EA and PSO much simpler, while the EA and PSO themselves eliminate unnecessary elements of the polynomial by setting their corresponding parameters λ_{iq} to zero. In the EA, the number of elements m in Eqn. (6) defines the number of genes in the chromosome. A single gene contains information about one function (10). When a single gene is mutated, all values of the a_{iq} and λ_{iq} parameters are randomly changed and the type of function f_i is chosen from among (11)–(13). For reproduction, the two-point crossover is used. When the optimization starts, only the boundaries for a_{iq} and λ_{iq} along with the number of elements m have to be defined. The experiments show that the parameter m should be small to ensure a proper quality of approximation and model complexity. The search space boundaries for particular a_{iq} are set taking into account the values of its corresponding parameter K_i. The fitness function in the EA and PSO is defined as the sum of distances between points $\mathbf{p}_f$ and the corresponding points of the obtained surface.

The obtained approximation polynomial and approximation points $\mathbf{p}_f$ are shown in Fig. 8 (only K_1 and K_2 in input). According to the obtained relation between slippages and the local shape of the terrain, the best potential footholds are holes (negative K_1 and a high, positive value of K_2). Flat surfaces are also evaluated as useful. The system learns to avoid local peaks (high, positive values of K_1 and K_2). Deep depressions are not good for foot support either.

The use of the Gaussian function for the approximation by using the set of points shown in Fig. 7 gives a similar approximation surface (cf. Fig. 8). To obtain better results, the number of input data was increased. The experiments conducted by using primitives give points which lie on the plane $K_2 = |K_1|$ or $K_2 = 0$. To increase the number of input data, the primitives were additionally modified by using Gaussian noise. It provides a higher variety of tested primitives. The obtained relation between slippages and the local shape of the terrain is shown in Fig. 9.

3.3. Control in regular mode. During the third phase of the algorithm, the robot uses an approximated polynomial to assess the potential footholds. Then, these results are used to find the best place to put the foot on. Such a decision support unit allows covering successfully difficult and rough terrain while avoiding slippages.

The first three coefficients are used as input to compute the polynomial (6). The fourth one (K_4) is used to exclude points on the ground which are too close to the boundaries of the robot's leg workspace. The final coefficient $Q(i,j)$ (interpreted as a prediction of a slippage),

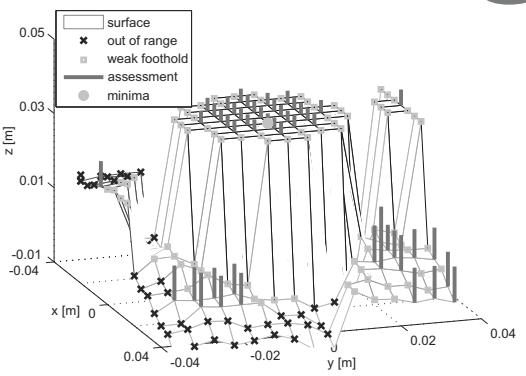

Fig. 10. Terrain and assessment results.

which describes the usefulness of the potential $[i,j]$ foothold described by K_1,...,K_4 features, is given as

$$Q(i,j) = P(K_1(i,j), K_2(i,j), K_3(i,j)) + k \cdot K_4(i,j). \tag{18}$$

The adjustable constant k has to be set as 8. When this value is too big, the robot selects 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, which might render the movement impossible.

3.4. Results: Regular mode. The algorithm was tested in a simulator on a specially prepared terrain map. It includes obstacles similar to scattered flagstones, extensive hills with mild slopes, small and pointed hills, depressions and stones. The highest point of the terrain is 8.7 cm high, and the deepest depression is -0.35 cm.

During the walk, the robot uses the standard gait for walking on 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 $\times$ 15 cells (7.5 $\times$ 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 visualized assessments is shown in Fig. 10. There are some 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 while the value of Q is too big (such cells give weak support to a leg's tip), or when a cell is described by values of K_i which are outside the boundaries defined in the learning phase (as properties of this cell are unknown). The foothold selection algorithm searches for the minimum among the rest of the cells. The shorter the bar shown in Fig. 10, the more useful the given cell as a foothold. When a decision about foothold selection is made, the control system of the robot modifies the trajectories of the feet and places them at selected points.

Slippages during the whole simulation run were recorded to show the results produced by the proposed al-

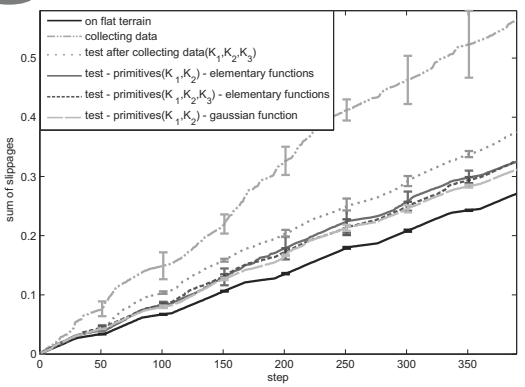

Fig. 11. Results: slippages obtained during verification experiments.

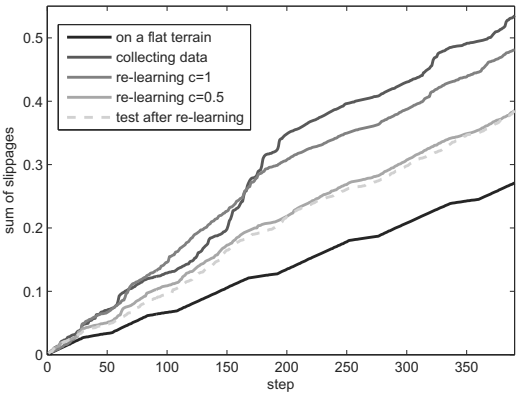

Fig. 12. Results of a re-learning procedure.

gorithm (Fig. 11). Each experiment consists of a series of three tests. The mean value of the accumulated slippages and its standard deviation are shown. The sum of slippages after n steps is defined as the sum of slippages from step $i = 0$ to $i = n$. These results are compared with the results of a simulation on flat terrain, and a simulation in which the robot randomly selected footholds. Better results are obtained when a point cloud acquired by using pre-prepared primitives is applied. Data obtained during experiments on real terrain are often ambiguous and disturbed. The results when all three coefficients are used for decision making are slightly better. When only coefficients K_1 and K_2 are used, the robot has a problem with traversing extensive hills with mild slopes. The best results are obtained in the experiment where the Gaussian approximation is used.

3.5. *A posteriori* modification of the decision support unit: Re-learning. For learning systems, there is always a problem with learning data sets. The knowledge which is used for shaping the system should cover the whole possible input space. The problem is when it is not possible to provide all necessary data. The robot is then taught using incomplete data. The decision support unit, which is shaped with these data, works properly only when it assesses known points on the terrain. There is a risk that previously unknown points of the terrain are assessed improperly. Then the robot should have a possibility to examine the unknown points of the terrain, to collect information about them and to merge the new knowledge with that stored in the old decision unit. As a result, a new, more general decision support unit is created. This procedure is called re-learning. It works in an iterative way.

To modify and adapt the decision support, the learning strategy was modified. During the normal operation stage, the robot additionally collects new data about the terrain. New information can be used to modify the existing approximation polynomial. The new point cloud is filtered and, as a result, a new group of points $\mathbf{p_f}$ is obtained. To extend the knowledge represented by the old group of points $\mathbf{p}_f^{\text{old}}$ and to compute a new group of points $\mathbf{p}_f^{\text{new}}$, a weighted mean is used:

$$\mathbf{p}_{f_i}^{\text{new}} = \frac{\mathbf{p}_{f_i}^{\text{old}} + c \cdot \mathbf{p}_{f_i}}{1 + c}, \tag{19}$$

where $\mathbf{p}_{f_i}^{\text{old}}$ and $\mathbf{p}_{f_i}$ are the points of old and newly collected groups of points respectively. New points which do not have their counterpart in the old group are simply added to the new group. The constant c is a weight (the value should be between 0 and 1) given to the new knowledge represented by a group of points $\mathbf{p}_f$. The value of c set to 1 means that newly collected data and the old ones are equally important. Generally, if the robot has a correctly working decision system, and it only aims to fine-tune it, the constant c should be small ($c < 0.1$).

To show properties of the re-learning procedure, a series of experiments was conducted. Results are shown in Fig. 12. At the beginning, the robot collects learning data by walking on rough terrain and by randomly selecting footholds. Next, the relation between slippages and the local relief of the terrain is shaped. Because the robot collects data randomly and the experiment is short, there is always a risk that the gathered knowledge is not complete. The robot does not have appropriate knowledge about some points of the terrain. In this experiment, the robot did not collect information on the sharp edges of the cliffs. As a result, the decision unit works improperly.

Because there were gaps concerning the knowledge related to sharp edges, the robot classifies such points as potentially suitable for footholds. As a result, the slippages in the experiment involving a relation between slippages and the local shape of the terrain are similar to those appearing while collecting data. What is important, during this experiment the robot collects new knowledge about previously unknown points. The new information is then added to the old knowledge according to (19). The value of c was set to 1. After this step, a significant improve-

ment was observed. Then the re-learning procedure was repeated again for c set to 0.5. After the last iteration, the slippages decreased slightly, which ended the re-learning procedure.

4. Static stability control of a six-legged walking robot

4.1. Importance of the stability region in motion planning. A six-legged walking robot while climbing obstacles is prone to falling down due to the lack of static stability. This could end up in mission failure. In order to avoid such a situation, a stability check is required. As climbing is a deliberate process, it is possible to plan it a few steps ahead. The robot can stop and check its supporting polygon in a current position and in planned positions in order to avoid overturning. To test a static equilibrium condition of the robot, there is a need to establish its support region. In many cases, a convex hull described by the contact points of the robot's supporting feet is taken as a good approximation of this region. But it was shown by Bretl and Lall (2008) that it is an insufficient assumption in the free multi-leg gaits. The CM of the robot must satisfy constraints represented by Eqns. (20)–(22) to preserve its stability:

$$\sum_{i=1}^{n} \mathbf{F}_i + m\mathbf{g} = 0, \tag{20}$$

$$\sum_{i=1}^{n} \mathbf{r}_i \times \mathbf{F}_i + \mathbf{c} \times m\mathbf{g} = 0, \tag{21}$$

$$\|(\mathbf{I} - \mathbf{v}_i v_i^{\mathbf{T}})\mathbf{F}_i\| \leq \mu \mathbf{v}_i^T \mathbf{F}_i, \tag{22}$$

where $\mathbf{F}_i$ is the vector of a reaction force at each leg (indexed by i), $\mathbf{r}_i$ is the position vector of the i-th leg contact point, $\mathbf{c}$ is the position vector of the CM, m is the mass of the robot, $\mathbf{g}$ is the gravity force vector, $\mathbf{v}_i$ is the vector normal to the surface at the i-th contact point, μ is a static friction coefficient. All vectors are expressed in the global coordinate system x_0, y_0, z_0.

A solution to the robot static equilibrium problem requires a projection of a nonlinear convex set onto a two-dimensional subspace as described by Bretl and Lall (2008). This can be done by using a Second Order Cone Program (SOCP) (Lobo *et al.*, 1998) specified in (23). These computations produce the extremal position of the CM in a chosen direction for each given set of the contact points of the legs. The optimal solution is found subject to a linear equality and a second order cone constraint. The method was selected because the constraints for the force and the torque balance are linear and the static friction is modeled as a second order cone. The solutions from the SOCP are used to approximate the support region by a polygon:

$$\begin{aligned} &\text{maximize} && \mathbf{a}^T\mathbf{y}, \\ &\text{subject to} && \mathbf{A}_1\mathbf{F} + \mathbf{A}_2\mathbf{y} = \mathbf{t}, \\ & && \| \mathbf{BF} \| \leq \mathbf{u}^T\mathbf{F}_i, \end{aligned} \tag{23}$$

where $\mathbf{y} \in \mathbb{R}^2$ is the position of a CM, $\mathbf{a} \in \mathbb{R}^2$ is the direction of optimization

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_1 \\ \vdots \\ \mathbf{F}_N \end{bmatrix}, \quad \mathbf{y} = \mathbf{P}\mathbf{c},$$

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Here $\mathbf{T}(\mathbf{r}_1)$ is a skew-symmetric matrix and $\mathbf{T}(\mathbf{r}_1)\mathbf{F}_1 = \mathbf{r}_i \times \mathbf{F}_i$,

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{T}(\mathbf{r}_1) & \cdots & \mathbf{T}(\mathbf{r}_N) \end{bmatrix},$$

$\mathbf{T}(m\mathbf{g})$ being a skew-symmetric matrix and $-\mathbf{T}(m\mathbf{g})\mathbf{c} = \mathbf{c} \times m\mathbf{g}$,

$$\mathbf{A}_2 = \begin{bmatrix} 0 \\ -\mathbf{T}(m\mathbf{g})\mathbf{P}^T \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} -m\mathbf{g} \\ 0 \end{bmatrix},$$

$$\mathbf{B} = \operatorname{diag}(\mathbf{I} - \mathbf{v}_1\mathbf{v}_1^T, \ldots, \mathbf{I} - \mathbf{v}_n\mathbf{v}_n^T),$$

$$\mathbf{u} = \begin{bmatrix} \mu_1\mathbf{v}_1 \\ \vdots \\ \mu_n\mathbf{v}_n \end{bmatrix}.$$

Bretl and Lall (2008) use an adaptive algorithm called PROJECT to approximate the support region by the inscribed and prescribed polygon, having the number of edges dependent on the complexity of the convex set geometry. In our work, the approximation of the support region with a 16-edge polygon is used (Walas, 2009).

4.2. Displacement of the CM. The walking robot consists of a trunk and legs. The trunk is not actuated and its CM position is constant with respect to the local robot coordinates system. However, any movement of the legs influences the position of the CM of the robot complete body. This change in the CM position could easily be calculated from the robot kinematic model. The kinematics of the Ragno robot are described by Walas *et al.* (2008) and Belter *et al.* (2008). Its frame assignment is shown in Figs. 13 and 14. Using the appropriate transformation matrices, the CM of each leg segment is described in the leg local coordinate system as follows:

$$\mathbf{CM}_{\text{coxa}} = \mathbf{A}_1\mathbf{r}_{\text{coxa}}, \tag{24}$$

$$\mathbf{CM}_{\text{femur}} = \mathbf{A}_1\mathbf{A}_2\mathbf{r}_{\text{femur}}, \tag{25}$$

$$\mathbf{CM}_{\text{tibia}} = \mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\mathbf{r}_{\text{tibia}}, \tag{26}$$

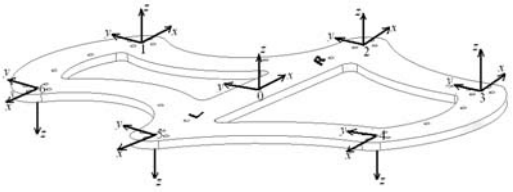

Fig. 13. Coordinates systems for the walking robot Ragno.

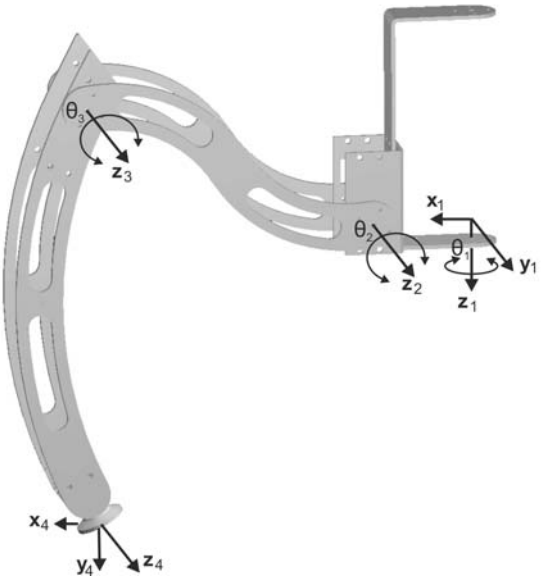

Fig. 14. Coordinates systems for the leg of the Ragno robot.

where $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$ are the transformation matrices from joint coordinates to the leg coordinate system, $\mathbf{r}_{\text{coxa}}, \mathbf{r}_{\text{femur}}, \mathbf{r}_{\text{tibia}}$ are the coordinates of the CM for each segment in the joint coordinate system. Having this, we could calculate the CM position for the whole leg by using

$$\begin{aligned}\mathbf{CM}_{\text{leg}} = &\frac{\mathbf{CM}_{\text{coxa}} \cdot m_{\text{coxa}}}{m_{\text{coxa}} + m_{\text{femur}} + m_{\text{tibia}}} \\ &+ \frac{\mathbf{CM}_{\text{femur}} \cdot m_{\text{femur}}}{m_{\text{coxa}} + m_{\text{femur}} + m_{\text{tibia}}} \\ &+ \frac{\mathbf{CM}_{\text{tibia}} \cdot m_{\text{tibia}}}{m_{\text{coxa}} + m_{\text{femur}} + m_{\text{tibia}}},\end{aligned} \tag{27}$$

where m_{coxa}, m_{femur}, m_{tibia} are the masses of each segment of the leg. Using the appropriate constant transformation matrices, we can express the CM of each leg in a robot local coordinate frame.

The CM position for the whole robot with its legs in a neutral configuration is shown in Fig. 15, marked with a dashed line. It was assumed that the neutral position of the leg is the following configuration: $\theta_1 = 0°$, $\theta_2 = 24°$, $\theta_3 = -114°$. This corresponds to the biological insect-like prototype. The arrows in the figure are the vectors of the CM position of each leg described in

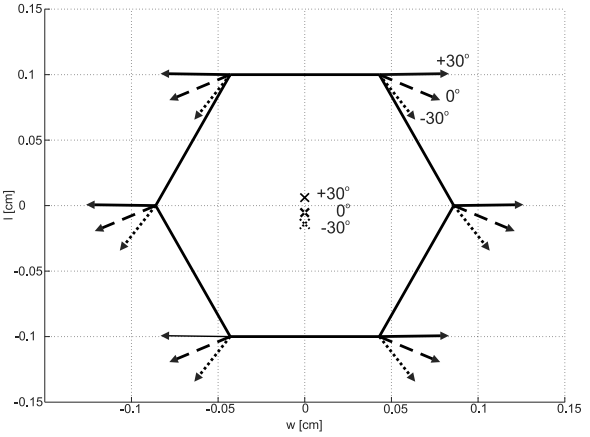

Fig. 15. Center of mass for the walking robot Ragno with legs in the neutral position and moved 30° forward and backward.

the leg's local coordinate system (on the xy plane). While the legs are rotated by 30° forward (in a positive direction around the z axis), the CM of the robot moves forward by 1.1 [cm] (marked with a continuous line). For the backward movement of 30°, the deviation from the neutral position is by 0.9 [cm] (marked with a dotted line). Therefore, the total shift between the extreme positions is 2.0 [cm]. When compared with the longitude of the robot trunk, which is 20.0 [cm], the relative change is of about 10%. The range of the movement of the robot CM depends on the length and the weight of the legs. In the case of Ragno, the mass of the legs constitutes half the weight of the whole machine.

4.3. Calculating static stability on-line. The proposed three-phase on-line stability test system for a walking machine is composed of two subsystems which were described in Sections 4.1 and 4.2. The description of each phase is as follows:

- In the first step, the SOCP problem is solved. For the stability check procedure, one performs a single SOCP computation in a direction planned beforehand and according to the movement vector $\mathbf{r}_M$ as shown in Fig. 17. After this step, one obtains $\mathbf{r}_{\text{SOCP}}$ (the extremal CM position in that direction). The entire 16 vertices approximation results in a complete region of stability. For a specific task, the optimization problem with constraints pointing towards a particular direction is solved. The output of this step is $\mathbf{CM}_{\text{SOCP}}$,$\mathbf{r}_{\text{SOCP}}$.

- In the second step, the change in the CM position for the whole body of the robot ($\mathbf{r}_{\text{KIN}}$) is found. The shift is due to the movement of the legs while performing the planned displacement in a direction described by the vector $\mathbf{r}_M$ (without a change of contact points).

The solution is found by using the kinematics of the robot as described in Section 4.2. The output of this step is $\mathbf{CM}_{\mathrm{KIN}}$ and $\mathbf{r}_{\mathrm{KIN}}$ (used in the third step).

- In the third step, the difference between the vectors $\mathbf{r}_M$ and $\mathbf{r}_{\mathrm{KIN}}$ is calculated. The result of this operation gives the real position of the center of mass $\mathbf{CM}_{\mathrm{KM}}$ after the planned displacement. It is the position after kinematic correction. The obtained $\mathbf{CM}_{\mathrm{KM}}$ position is described by the vector $\mathbf{r}_{\mathrm{KM}}$. Equation (28) allows evaluating the real Geometrical Stability Margin (GSM) for the movement of the robot in direction $\mathbf{r}_M$. All described vectors are attached to the point defining the robot's center of mass in a neutral position for the legs configuration as shown in Fig. 17:

$$\|\mathbf{r}_{\mathrm{SOCP}} - \mathbf{r}_M - \mathbf{r}_{\mathrm{KIN}}\| = \mathbf{GSM}. \qquad (28)$$

A schematic description of this approach is shown in Fig. 16. The whole procedure for the specific case is described below. Let us consider a walking robot climbing against a 30° slope. Its fore- and middle-legs are on the slope and hind-legs are on the flat surface beneath the slope. The body of the robot is parallel to the flat surface beneath. The result of the stability check procedure for this example is shown in Fig. 17. The support region obtained using the SOCP (marked as a continuous line) is smaller here than the one obtained with the classical approach (a convex hull described by contact points of the robot's supporting feet, marked as a dotted line). The robot is about to move 4.3 cm forward in the horizontal direction; this change is described by the vector $\mathbf{r}_M$. The second and the third step of the algorithm presented in Section 4.3 are shown using the position vectors $\mathbf{r}_{\mathrm{KIN}}$ and $\mathbf{r}_{\mathrm{KM}}$. The difference between GSM in the given direction obtained using the SOCP and by using Kinematic Correction (KM) is evident. The procedure described in Section 4.3 is appropriate for all walking robots having static stability in transient states. The implementation of the stability check on a real machine requires some customization, because it depends on the kinematics of a particular robot. In most cases, walking robot legs have an anthropomorphic structure. Thus the only change needed to implement the procedure presented here for the Ragno robot on another machine is the change in the dimensions of each leg segment.

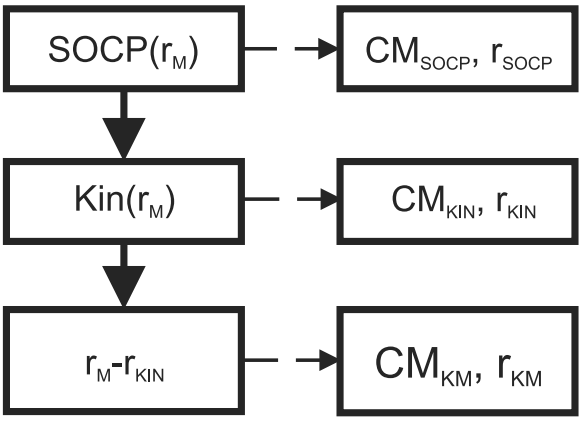

Fig. 16. Stability check procedure.

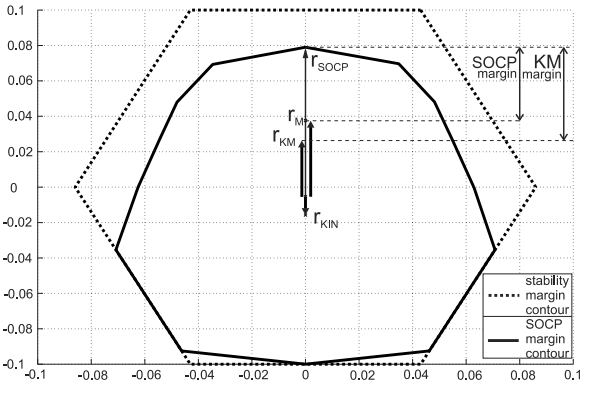

Fig. 17. Stability analysis result for the walking robot Ragno.

4.4. Reaction forces and stability analysis. The system for calculating static stability on-line described in Section 4.3 works well assuming that the exact position of the CM of the robot is known. We can calculate a CM of the whole robot off-line by using CM position of each leg segment and the trunk as described in Section 4.2. But it is insufficient, as the robot's CM could change its location during the task. For example, it is possible to place some load on the back of the robot or to carry some package by the middle legs and to walk on the remaining four. In the mentioned examples, there was a need for establishing the CM position for the complete robot on-line. This could be done by measuring the reaction forces exerted on the ground. But the problem is to measure the forces. Hereafter we propose two methods of doing this.

4.5. Measuring stance forces. It is possible to split those methods into direct and indirect ones. The direct method is based on measurements with a force sensor which is shown in Fig. 18. It is a resistive sensor whose characteristics are non-linear (in the required range from 0.5 to 2 kg could be linearized). We obtain a voltage on the resistor dependent on the reaction force. Using the A/D converter, an exact value of a voltage representing the force is obtained. One gets the module of the force normal to the ground in the contact point. To obtain the direction of the reaction force vector, we use direct kinematics of the leg and information on the orientation of the ground. Thus the information on the reaction force is complete. The sensor

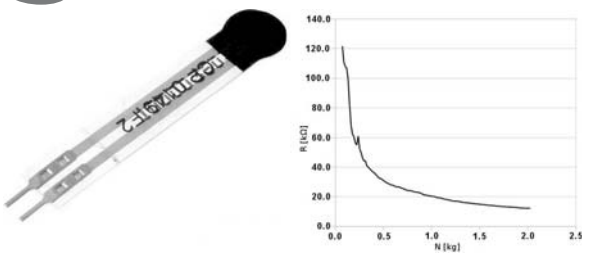

Fig. 18. Pressure sensor and its characteristics.

installation on the Messor robot is shown in Fig. 19. The robot is the next generation of Ragno. Both robots have a similar mechanical structure, but Messor is approximately three times bigger than Ragno. Moreover, the Messor robot is equipped with more sensors than the other one.

The second method is indirect and it is based on measuring the torque in each joint of the robot's leg. To be precise, the current of the motor in each joint is measured. According to Eqn. (29), a torque in each joint is obtained. To measure the value of the current, a transducer is used. The sensor has linear current/voltage characteristics. Measurements are based on the Hall effect, so the circuit current does not change:

$$M(t) = k_\phi \cdot i(t), \tag{29}$$

where k_ϕ= 0.845 Nm/A.

With a torque in each joint measured, it is possible to calculate the reaction force for each leg. This is done by using.

$$\mathbf{F} = (\mathbf{J^T})^{-1}\mathbf{Q}, \tag{30}$$

where $\mathbf{F}$ is the reaction force vector $[3 \times 1]$, $\mathbf{J}$ is the geometric Jacobian for the leg $[3 \times 3]$, $\mathbf{Q}$ is the torque in each joint vector $[3 \times 1]$. We use an inverse geometric Jacobian to obtain the force at the contact point at the tip of the leg.

An example of the characteristics of the change in the force at the tip of the leg of the robot while performing tripod gait is shown in Fig. 20. The measurements are based on the current ones and are calculated using Eqn. (30).

Fig. 19. Example application of the pressure sensor on the Messor robot.

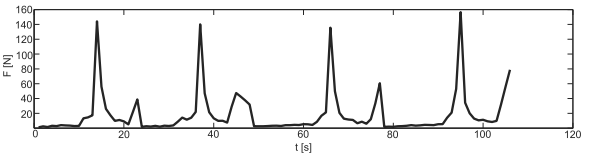

Fig. 20. Example characteristic of the force at the tip of the robot leg.

Both methods return the force in a robot local coordinates system. To find the forces described in a global frame, we have to know the orientation of the local robot frame in a global frame. It could be established by using the IMU sensor. In the Messor robot, ADIS 16350 is used. It is a six-DOF sensor which provides information about acceleration along each axis and about angular velocity around each axis. Each of the methods has its flaws. The first one is direct. It is based on the contact sensor and has measurements inaccuracies which depend on the shape of the contact of the foot with the sensor. The shape and the area of the contact play an important role in this case. The second method, which is indirect and based on the current measurements is sensitive to small changes of the current which, could be due to measurement noise. Simultaneous use of both methods could give better results.

With the full information about forces and the robot's orientation, it is possible to obtain the real position of the CM of the robot in the xy-plane. This is achieved by the following equation, which was derived from (23):

$$\mathbf{y} = \mathbf{A}_2^+\mathbf{t} - \mathbf{A}_2^+\mathbf{A}_1\mathbf{F}, \tag{31}$$

where $\mathbf{y}$ is the position of the CM $[2 \times 1]$, $\mathbf{t}$ is the gravity vector $[6 \times 1]$, $\mathbf{F}$ is the force vector $[18 \times 1]$, $\mathbf{A_1}$ is the reaction forces matrix $[6 \times 18]$, $\mathbf{A_2}$ is the gravity forces matrix $[6 \times 2]$.

4.6. Modified stability check procedure for a walking robot. In Section 4.3, the basic stability check procedure was described. As could be seen, the step where the current CM position is measured is missing. In an extended version of the stability check shown in Fig. 21 there is one more step included where the real CM position of the robot is established. It allows the robot to plan future movements by relying on the current knowledge of its CM position in the support region. The description of the functions of each step is as follows:

- The first step gives the approximated support region for the walking robot.
- The second step gives the real position of the CM of the robot and its distances to the borders of the support region. The function *Torque to CM position (T2CM)* takes as the argument a vector of torques in each leg (τ) and returns the real position of the CM of the robot.

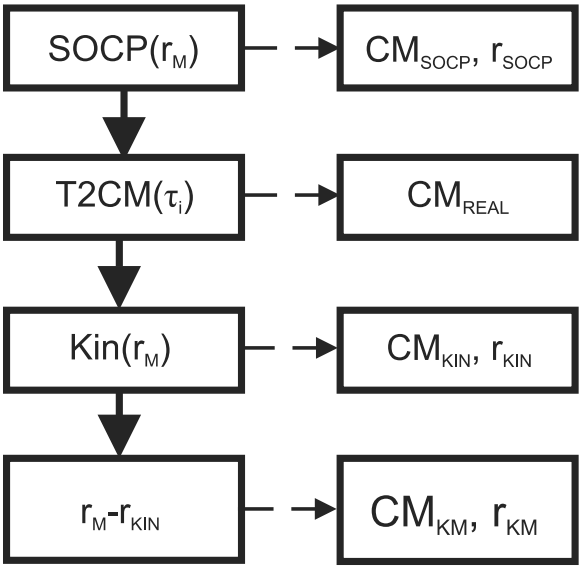

Fig. 21. Extended stability check procedure.

- The third step gives the displacement of the CM of the robot due to the planned movement (with kinematic correction).
- The fourth step returns the real geometric stability margin for a six-legged walking robot.

4.7. Calibration procedure and results of the check. At the beginning, the calculations of the position of the CM of the robot were obtained in the simulator. This created the possibility to check if the assumptions were correct. Thus we had a known position of the CM and also the force distribution for a given position of the CM obtained with SOCP method. We passed the force vectors to Eqn. (31). The resulting position of the CM of the robot was exactly the same as the one obtained with the SOCP. In this way, it was proved that our assumptions were correct.

The next step was to validate if the method works in a real environment on a real walking robot. First, the preliminary calibration of the system was performed. In this procedure robot moved 10 cm forward and then 10 cm backward. The resulting CM position along the Y-axis was 15.27 cm for the forward movement and 11.32 cm for the backward movement. By dividing the performed movement 20 cm by the sum of the above mentioned distances 26.59 cm, one obtains a real value of a gain of a motors equal to k_ϕ=0.752 Nm/A. The positions of the CM of the robot while moving forward and backward are shown in Figs. 22 and 23, respectively. Having all this, the measurement of the real position of the CM of the robot could be done. We measured the currents and from that we obtained the force distribution for the walking robot at the contact points. The force vectors are marked with arrows as shown in Fig. 24. In this figure, the measured CM position of the robot is represented as a cross. This experiment was performed with the Messor robot standing in its leg's neutral position, $\theta_2 = 24°$, $\theta_3 = -114°$. As could be seen, the established CM of the robot lies in the middle of the convex hull described by the contact points of legs. The CM of the robot is at the point $(-4.47$ cm,1.07 cm).

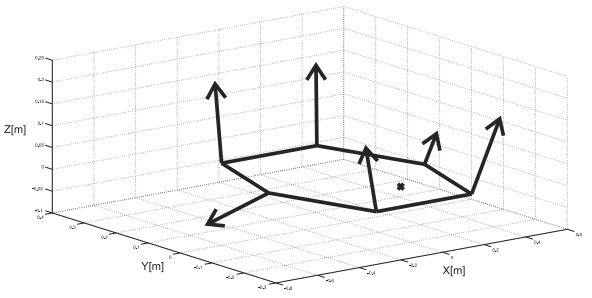

Fig. 22. Backward movement of the robot.

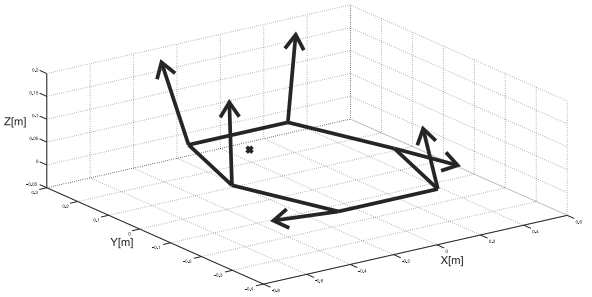

Fig. 23. Forward movement of the robot.

5. Conclusions and future work

This paper presented a method for creating a foothold selection system for a multi-legged walking robot and a static stability control system for such a machine.

The system of foothold selection learns automatically without any expert knowledge. The robot acquires the ground surface characteristics by walking on example terrain or by testing ground primitives. Then it exploits this knowledge to find a relation between slippages and the local shape of the terrain, which is used for assessment of the candidate footholds.

The simulations show that after learning the robot is able to select the appropriate footholds to prevent slippages. The robot learned to avoid peaks of the terrain and to select such points on the ground which give appropriate support. This behavior is general. The same rules work on different types of terrain. Additionally, the decision support unit can be modified by using a re-learning procedure. It offers a possibility to extend the knowledge stored in the approximation polynomial.

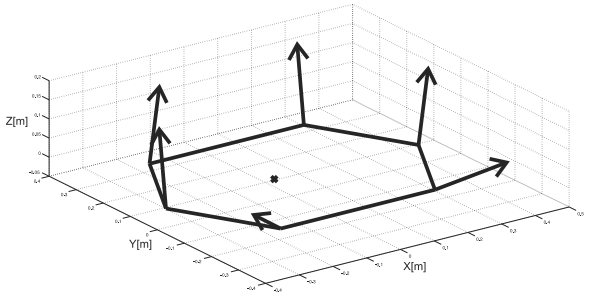

Fig. 24. CM position obtained from measurements of the real Messor robot.

The system of static stability control determines a real position of the CM of the robot and allows predicting its future position after making a planned step. The availability of such a system is vital for stable climbing of a walking robot. The results obtained on a real robot demonstrated that the method is feasible but there is still a need for some refinement. Especially signal conditioning is necessary to achieve smooth measurement results in each configuration of the robot. Random fluctuations at a steady state are due to digital servomotors operation. Efficient methods of reducing this noise are needed.

Acknowledgment

This work has been funded by the grant no. N514 294635 for the years 2008–2010 of the Polish Ministry of Science and Higher Education. Dominik Belter is a scholarship holder within the project *Scholarship support for Ph.D. students specializing in majors strategic for Wielkopolska development*, Sub-measure 8.2.2: Human Capital Operational Programme, co-financed by the European Union under the European Social Fund.

References

Annunziato, M. and Pizzuti, S. (2000). Adaptive parameterization of evolutionary algorithms driven by reproduction and competition, *Proceedings of ESIT 2000, Aachen, Germany*, Vol. 1, pp. 31–35.

Bai, S. and Low, K.H. (2001). Terrain evaluation and its application to path planning for walking machines, *Advanced Robotics* **15**(1): 729–748.

Bai, S., Low, K.H. and Zielińska, T. (1999). A new free gait generation for quadrupeds based on primary/secondary gait, *Proceedings of the IEEE International Conference on Robotics and Automation, Detroit, MI, USA*, pp. 1371–1376.

Barghava, S. and Waldron, K. (1988). Stability analysis of the walking beam vehicle, *Proceedings of the International Advanced Robotics Conference, Pisa, Italy*, pp. 114–119.

Belter, D. (2009). Adaptive foothold selection for a hexapod robot walking on rough terrain, *7th Workshop on Advanced Control and Diagnosis, Zielona Góra, Poland*, (on CD-ROM).

Belter, D., Kasiński, A. and Skrzypczyński, P. (2008). Evolving feasible gaits for a hexapod robot by reducing the space of possible solutions, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Nice, France*, pp. 2673–2678.

Belter, D. and Skrzypczyński, P. (2009). Efficient gait learning in simulation: Crossing the reality gap by evolutionary model identification, *in* O. Tosun, H.L. Akin, M.O. Tokhi and G.S.Virk (Eds.), *Mobile Robotics: Solutions and Challenges*, World Scientific, Singapore, pp. 861–868.

Belter, D., Walas, K. and Kasiński, A. (2008). Distributed control system of DC servomotors for six legged walking robot, *Proceedings of the International Power Electronics and Motion Control Conference, EPE-PEMC 2008, Poznań, Poland*, pp. 1044–1049.

Bretl, T. and Lall, S. (2006). A fast and adaptive test of static equilibrium for legged robots, *Proceedings of the International Robotics and Automation Conference, Orlando, FL, USA*, pp. 1109–1116.

Bretl, T. and Lall, S. (2008). Testing static equilibrium for legged robots, *IEEE Transactions on Robotics* **24**(4): 794–807, DOI: 10.1109/TRO.2008.2001360.

Burnhamn, K. and Anderson, D. (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretical Approach*, Springer-Verlag, New York, NY.

Dahlquist, G. and Bjorck, A. (1974). *Numerical Methods*, Prentice Hall, Englewood Cliffs, NJ.

Gassmann, B., Frommberger, L., Dillmann, R. and Berns, K. (2003). Real-time 3d map building for local navigation of a walking robot in unstructured terrain, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA*, pp. 2185–2190.

Gonzalez, P., Estremera, J., Garcia, E. and Armada, M. (2005). Force distribution in closed kinematic chains, *Autonomous Robots* **18**(1): 43–57, DOI: 10.1023/B:AURO.0000047288.23401.5c.

Gutmann, J.-S., Fukuchi, M. and Fujita, M. (2004). Stair-climbing control of humanoid robot using force and accelerometer sensors, *Proceedings of the International Intelligent Robots and Systems Conference, Sendai, Japan*, pp. 1407–1413.

Kalakrishnan, M., Buchli, J., Pastor, P. and Schaal, S. (2009). Learning locomotion over rough terrain using terrain templates, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems, St. Louis, MO, USA*, pp. 167–172.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks, Piscataway, Australia*, pp. 1942–1948.

Kolmogorov, A. (1957). On the representation of continous function of several variables by superpositions of continous functions of one variable and addition, *Doklady Akademii Nauk SSSR* **114**(4): 953–956.

Kolter, J., Rodgers, M. and Ng, A. (2008). A control architecture for quadruped locomotion over rough terrain, *Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, CA, USA*, pp. 811–818.

Kolter, J., Youngjun, K. and Ng, A. (2009). Stereo vision and terrain modeling for quadruped robots, *Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan*, pp. 1557–1564.

Kosiński, W. and Weigl, M. (1998). General mapping approximation problems solving by neural networks and fuzzy inference systems, *Systems Analysis Modelling Simulation* **30**(1): 11–28.

Kumar, V. and Waldron, K. (1988). Force distribution in closed kinematic chains, *Proceedings of the International Robotics and Automation Conference, Philadelphia, PA, USA*, pp. 114–119.

Łabecki, P., Łopatowski, A. and Skrzypczyński, P. (2009). Terrain perception for a walking robot with a low-cost structured light sensor, *Proceedings of the 4th European Conference on Moblie Robots, Dubrovnik, Croatia*, pp. 199–204.

Li, T.-H., Su, Y.-T., Kuo, C.-H., Chen, C.-Y., Hsu, C.-L. and Lu, M.-F. (2007). Stair-climbing control of humanoid robot using force and accelerometer sensors, *Proceedings of the SICE Annual Conference, Takamatsu, Japan*, pp. 2115–2120.

Lobo, M., Vandenberghe, L., S.Boyd and Lebret, H. (1998). Applications of second-order cone programming, *Linear Algebra and Its Applications* **284**(1–3): 193–228, DOI: 10.1016/S0024-3795(98)10032-0.

Lorentz, G. (1986). *Approximation of Functions*, American Mathematical Society, New York, NY.

Rebula, J., Neuhaus, P., Bonnlander, B., Johnson, M. and Pratt, J. (2007). A controller for the littledog quadruped walking on rough terrain, *Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy*, pp. 1467–1473.

Roennau, A., Kerscher, T., Ziegenmeyer, M., Zoellner, J. and Dillmann, R. (2009). Six-legged walking in rough terrain based on foot point planning, *in* O. Tosun, H.L. Akin, M.O. Tokhi and G.S. Virk (Eds.) *Mobile Robotics: Solutions and Challenges*, World Scientific, Singapore, pp. 591–698.

Schmucker, U., Schneider, A. and Rusin, V. (2003). Interactive Virtual Simulator (IVS) of six-legged robot Katharina, *Proceedings of the IEEE International Conference on Climbing and Walking Robots, Catania, Italy*, pp. 327–332.

Smith, R. (2007). Open dynamics engine, `www.ode.org`.

Vernaza, P., Likhachev, M., Bhattacharya, S., Chitta, S. and Kushleyev, A. Lee, D. (2009). Search-based planning for a legged robot over rough terrain, *Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan*, pp. 2380–2387.

Walas, K. (2009). Static equilibrium condition for a multi-leg, stairs climbing walking robot, *in* K.R. Kozlowski (Ed.), *Robot Motion and Control 2009*, Lecture Notes in Control and Information Sciences, Vol. 396, Springer-Verlag, Berlin/Heidelberg, pp. 197–206, DOI: 10.1007/978-1-84882-985-5.

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

Zhou, D., Low, K. and Zielińska, T. (2000). An efficient foot-force distribution algorithm for quadruped walking robots, *Robotica* **18**(4): 403–413.

Krzysztof Walas received the M.Sc. degree in control engineering and robotics from the Poznań University of Technology in 2007. Since then he has been pursuing his Ph.D. in robotics, working as a research assistant at the Institute of Control and Information Engineering of the same university. His research interests include walking robots control, multisensor environment perception for navigation purposes and machine vision.

Dominik Belter received the M.Sc. degree in control engineering and robotics from the Poznań University of Technology in 2007. Since then he has been pursuing his Ph.D. in robotics, working as a research assistant at the Institute of Control and Information Engineering of the same university. His research interests include the control of walking robots, machine learning, and soft computing.

Received: 15 January 2010
Revised: 22 June 2010
Re-revised: 16 November 2010