Learning Fast Walking Patterns with Reliable Odometry Information for Four-Legged Robots

Matthias Hebbel, Ingo Dahm, Denis Fisseler and Walter Nistico
Institute for Robot Research
Section Information Technology
Dortmund University
Otto-Hahn-Str.8, 44221 Dortmund
Email: *forename.surname*@uni-dortmund.de

Abstract

In this paper we describe a way to control and learn walking patters for a four-legged robot which result in very fast and stable omnidirectional walks with accurate odometry information. The fastest forward walk which was learned on a Sony Aibo ERS 7 with this approach reaches more than 50 cm/s. This is more than 25% faster than the fastest published walk found by any RoboCup team so far. The fast and manoeuvrable walk contributed a lot to the good overall performance of our team and helped to win all attended RoboCup competitions in 2005.

I. Introduction

Legged locomotion presents distinct advantages compared to the wheeled one, due to its ability to overcome obstacles such as stairs, and operate on rough surfaces and terrains. Further, it mimicks the natural locomotion of biological systems, and there are several robotic applications where a certain degree of anthropomorphism is desirable, because such robots have to interact with humans and their environment. On the other hand, the control systems required for this kind of locomotion are highly complex and represent a real challenge due to the high dimensionality of the control space, resulting from the multiple degrees of freedom which a leg offers, and the added problem of stability. A good opportunity to research in this field is offered by the commercially available robot "Aibo" from Sony, which is used as the official hardware platform for the RoboCup Four-Legged League, due to its wide availability and moderate cost. Robot soccer represents an excellent test environment and benchmark for applications of this nature, due to its competitive nature where the speed, maneuverability and stability of the robots' locomotion play a fundamental role.

A. The Platform

The development platform used for our research is the robot Sony Aibo ERS-7. The Sony Aibo is a fully autonomous quadruped robot which comes equipped with a CMOS-camera as the only exteroceptive sensor. Its legs have each 3 degrees of freedom: hip abduction, hip flexion and knee flexion joints. The robot can perform cognitive and reactive computations fully on board, thanks to a 64 bit 576 MHz MIPS processor and 64MByte of RAM; For communication the robot is equipped with a WLAN 802.11 wireless network card.

All coordinates mentioned in this text are in a robot centric reference system and are aligned as shown in Figure 1. The x-axis points to the forward direction, the y-axis points to the left side of the robot and the z-axis points upwards.

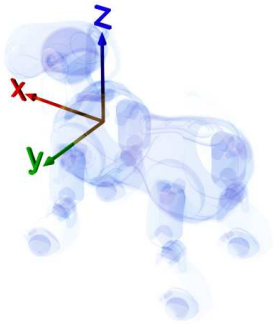


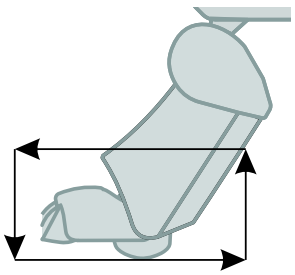
Fig. 1. Coordinates system of the robot

B. Background

The first significant achievement on this robotic platform has been presented in 2001 with the so-called "PWalk" [1], which allows omnidirectional locomotion of the robot by treating the legs as wheels, and achieves a very good stability using a particular posture of the front legs, where the contact point with the ground is located on the forearm instead of the paw. This approach moves each robots' leg on a 2-dimensional squared locus. The required joint angles for the movement of the legs are then calculated using inverse kinematics. For omnidirectional movements the loci of the feet are rotated like wheels accordingly to the desired direction. Figure 2 shows the rotation of these loci about the z-axis for different walk requests.

(a) Forward (b) Diagonal (c) Rotation

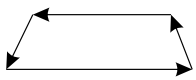
Fig. 2. The rotation of the leg locus for different walk requests



(a) The controlled locus

(b) Rectangular locus

(c) Half-elliptical locus



(d) Trapezoidal locus

Fig. 3. Different trajectories to move a foot

Following a trot gait, the two diagonal opposite legs are always moved at the same time, e.g. two legs are always in the air while the other two legs remain in contact with the ground. Theoretically this walk is only dynamically stable, but in practice it has shown to be very stable even during a soccer match where a robot is often pushed by opponents or teammates, this because the whole forearm can be approximately considered as a contact area.

The loci where the feet move on are represented in a parameterized form [2]–[4]. A lot of efforts have been spent in order to determine an optimal locus shape and parameterization: UT Austin Villa used 17 parameters [5] and an half-elliptical locus, the GermanTeam 14 parameters [4] and a squared locus, and the University of Pennsylvania used 19 parameters [6] to describe free-form quads (see Figure 3).

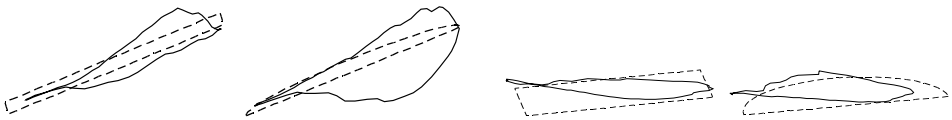
To optimize such high-dimensional parameter spaces, automatic learning has been shown to be the best approach. In [7] four common optimization strategies have been compared on a 12 parameters walking engine. To estimate the fitness of a parameter set, the robots were walking straight between two known landmarks and the resulting speed was measured. The best result achieved was 291 mm/s on the Sony Aibo ERS-210. Another paper focused on reducing the time to evaluate the fitness of a parameter set by making use of proprioceptive touch sensors in the robots' feet [8]. This way the robot speed was measurable within a few seconds, hence a much bigger search space could be examined and the resulting walk achieved a forward speed of 311 mm/s on the Sony Aibo ERS-210 and a forward speed of 400 mm/s on the more powerful Sony Aibo ERS-7. This approach to measure the speed has the disadvantage that it constraints the hind legs to move always in a way that the touch sensors stay in contact with the ground during the ground phase of a step.

All of the mentioned approaches have in common that only a single parameter set has been used for walking (tuned for fast forward walking) and the loci where the feet move are two-dimensional. As we will show, removing such constraints can yield greatly improved performance for omnidirectional walking.

II. Controlling the Legs

The control engine for the Sony Aibo in all the approaches described in the literature is based on static inverse kinematics. This because a complete dynamic model for this robot is not available, and it's also not computationally feasible for the on-board resources, which have to be shared with other processes such as cognition and image processing. As a result, there are significative differences between the controlled locus and the real locus that a foot describes during a walk.

Figure 4 shows such phenomenon for two different shapes of the controlled loci in the xz-plane for the fore and hind legs; the loci in the Figures 4(a) and 4(b) describe a rectangular path while the loci in Figure 4(c) and 4(d) follow a half-elliptical path. All other gait parameters like timing, step lengths etc. do not differ. Despite the small differences between the mentioned controlled paths, in our experiments with the rectangular locus the robot was walking smoothly, while in case of the semi-elliptical one, it was stumbling and slipping. By measuring the real locus in both cases, using data recorded from the joint sensors of the legs during the walk, we found out that the real trajectories were differing much more than what we would have expected from the differences in the controlled values, as can be seen from the solid cuves in Figure 4(a) – 4(d). Another important characteristic which emerged from our observations of the real trajectory of the feet in the yz-plane was that the real locus for front and hind legs doesn't lie on a plane, but describes a tri-dimensional path.



(a) Fore leg rectangular (b) Fore leg half-elliptical (c) Hind leg rectangular (d) Hind leg half-elliptical

Fig. 4. The controlled (dashed) and real (solid) loci of the fore and hind legs for different path models

A. Parameters Defining the Gait

Due to the mentioned observations on the real loci it appeared evident that giving more flexibility in the shape of the loci could give better results in terms of faster walks, particulary we introduced three dimensional polygons instead of the common two dimensional shapes.

As mentioned in section I-B, keeping low the overall number of parameters is highly desirable, this because hand tuning them is not feasible, and the time to optimize the parameters with machine learning approaches greatly profits from a smaller search space, especially considering the long times required to evaluate the fitness.

So, we decided to use 3 dimensional polygons with n vertices $P_1, \ldots, P_n$. Additionally n timing parameters are needed to specify the amount of time needed for the foot to travel from vertex P_i to P_{i+1}, for $1 \leq i < n$, and from P_n to P_1. Further we restricted the walking gait to a trot gait, so two diagonally opposite legs are at the same time in the air, while the other remain on the ground. This results in a total number of parameters for a single leg of $3n + n$. Considering the symmetry between the left and right side of the robot, for a 4 legged robot we have a total of $8n$ parameters.

B. Improving Omnidirectional Walking

Unfortunately a parameter set optimized for a certain walking direction, for example fast forward, is not necessarily good for walking sideways or backwards or rotating. For example, the fastest walk described in [8] was only useful for straight forward walking with sidewards and rotational components almost null. Consequently, this parameter set was used only for "sprints" to the ball [4], while in the more common game situations the robot was switching to a slower but more neutral with respect to direction parameter set.

The problem with "hard" switching of parameter sets while walking is that the abrupt transition can cause stumbling and tripping of the robot, so we decided to implement a technique to smoothly interpolate among parameter sets. Thus, we can use optimized parameters for the main walking directions without incurring in the negative effects of stumbling or unwanted directional changes. To be able to interpolate between the parameter sets, we constrained them to have all the same number of vertices.

Figure 5 shows the matrix of 12 polygon sets which are used to generate by interpolation an optimal set for each walk request. Apart from the sets optimized for maximum speed in the main directions (fast rotation, fast forward, fast backward, fast sidewards and fast diagonal), we have also included special parameter sets which ensure a smooth motion with limited "head bobbing" at medium speeds. This gives the advantage of having only little motion blur in the camera images when walking at moderate speed, which happens in many game situations.

To calculate a polygon set for a certain walk request, at first a polygon set for group 1 (see Figure 5) is calculated by linear interpolation between two sets of this same group according to the x-component of the request. Similarly, a second polygon set is generated for group 2 or 3, depending on the y-component of the request. Out of

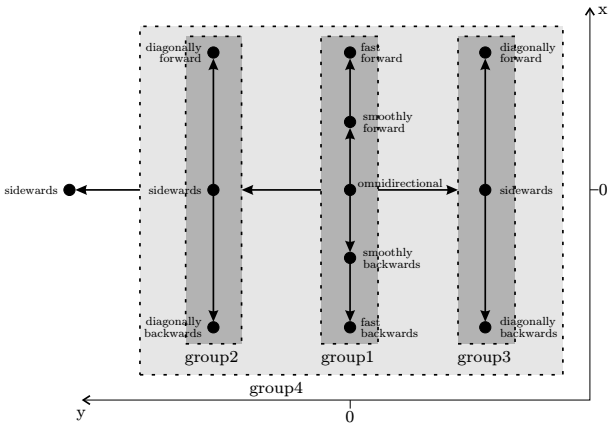


Fig. 5. The interpolation scheme for a walk request

these two resulting intermediate polygon sets, a final polygon set for group 4 can be calculated. This suffices for motion requests having only a translational component. In presence of an additional rotational component, this polygon set has finally to be interpolated with the special set for rotations.

III. Learning to Walk

Due to the large number of possible walking parameter combinations resulting from the high dimensional search space, the fact that the real trajectories of the feet differ significantly from the controlled trajectories and the lack of a physical simulator for the Sony Aibo robots, the only possibility to find adequate walking parameters is to use machine learning approaches on the physical robot.

A. The Learning Strategy

Since the search space is high–dimensional and expected to be non–linear and multimodal, we decided to use a $(\mu/\rho \stackrel{+}{,} \lambda)$ evolution strategy with self-adaption [9]. More simple approaches like e.g. hill-climbing are usually more determined but also get easily stuck in local optima. The $(\mu/\rho \stackrel{+}{,} \lambda)$ evolution strategy is inspired by the biological evolution process and is based on the paradigm *survival of the fittest*. The evolution strategy operates on populations of individuals. An individual contains assignments for the parameters which have to be optimized. The parent generation consists of μ individuals and the offspring generation consists of λ individuals. The offsprings are generated out of ρ randomly selected parents by using *recombination* and a following random *mutation*.

After the generation of the offsprings, their *fitness* has to be determined. Better assignments of the parameters of an offspring result in a better fitness. Having estimated the fitness of all offspring individuals, the next parent generation gets *selected*. In case of the $+$-strategy, the *selection*-operator elects the λ best individuals out of the offsprings *and* the parents while in the case of the ,-strategy only the best λ individuals out of the offsprings are chosen to be the parents of the next generation.

The strength of the *mutation* of an offspring has a big effect on the speed to approach the optimum. For too big mutations, the risk to overshoot the optimum rises, while too small mutation strengths can cause the danger to get stuck in a local optimum and prevent a fast progress speed toward the optimum. Due to the importance of the mutation strengths, we decided to use a self-adapting approach to control the mutation strength [10]. The basic idea of this approach is to let the mutation strengths be part of

the individuals. Due to the inclusion of the strategy parameters in the individual, they are selected and inherited together with the individual's assignments for the parameters which have to be optimized. Thus, they have a higher probability of survival when they "encode" object parameter variations that produce fitter object parameters [11].

B. Estimating the fitness of an individual

The major goal of the required walking patterns is the maximum speed. In robot soccer the team with the faster robots has an invaluable advantage: the faster robots reach the ball at first and can keep it under control while the opponent is automatically in a defensive position. Other side requirements for the walk are stability, to be robust against pushing of other robots, and smoothness, to prevent the camera from making blurry images.

As a first approach we decided to use only the maximum speed of the walk resulting out of the parameter assignments of an individual as its fitness. To determine the fitness, all previous approaches that make use of the sensors of the robot have important shortcomings, such as the expected imprecise measurements (based on vision only), missing or imprecise odometry information (based on self localization) or constraints to the possible walking patterns (based on proprioception) [8]. Excepting the proprioception based approach, all other approaches also have the disadvantage that measuring the speed of a walking pattern takes a lot of time. To be able to measure the robots' speed reliably and quickly, we mounted a camera at the ceiling observing the robot on the playing field. The camera is connected to a server which processes the images with a frame rate of 25 Hz and broadcasts the position of the detected robot into the network. Since the shape of the robot makes it difficult to observe it accurately with a camera from top, we equipped the robot with lightweight and colored markers on the back, which did not affect the walk. Figure 6 shows a picture taken with the ceiling camera of one half of the playing field with several debug drawings. The robot with its red and blue marker can be seen in the upper left corner. With this configuration we achieved a variance in the position accuracy below 1 cm for a standing robot.

Fig. 6. One half of soccer field observed by the ceiling camera.

C. Learning fast walking patterns

Due to the big search space, we expected a high number of walking attempts of the robot to be necessary to find an optimal parameter set, so the major goal of the learning behavior of the robot was to achieve as much autonomy as possible. Since the robot can always receive its accurate position over the integrated wireless lan card from

the ceiling camera, the whole process can run without user intervention. The learning behavior is as follows: at first the robot positions in a way that it will not leave the field and the observation area of the ceiling camera when walking into the planned direction. To evaluate the fitness of an individual, the robot starts to walk with the walking pattern resulting out of the parameters of the individual at maximum speed. To exclude acceleration effects from the speed measurement, we allow 2 seconds walking, before starting to measure the speed. After this "warm up" phase, the robot stores its current position and after another 2 seconds its achieved position. By dividing the covered distance between the two taken points by 2, we get the average speed of the last 2 seconds' walk.

IV. Enhancing Odometry Information

The commonly used self localization approaches make intensive use of the odometry of the walk [12], [13]. More accurate information about the covered distance results in a better tracking of the position. When the robot can trust more on its odometry data, this also has a positive effect on its behavior: it can focus its attention more often to the ball or other game relevant objects instead of looking to landmarks to localize.

For the walking gaits described in section II the theoretical odometry data for a certain walk request can be calculated straight forward. The length of the ground phase of the feet is equivalent to the movement of the robot. In practice, due to slip on the ground and the fact that the real loci of the feet differ from the controlled ones, the real odometry differs from the theoretical odometry.

The problem of the non–linear correlation between the controlled speed and the real speed of a Sony Aibo ERS-7 robot has been discussed in [4], [14]. The graphs in Figure 7 illustrate this for a robot which is moving on a circular path with different velocities. For slow walking, the measured speed follows the controlled speed with no big deviation until a point where the real speed can not follow the requested speed anymore.

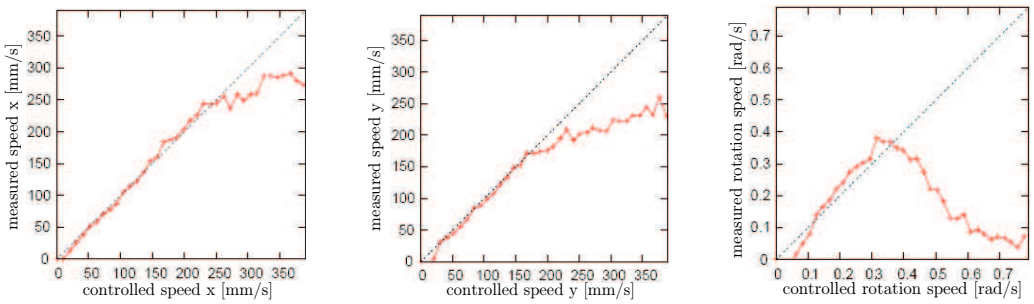


Fig. 7. Odometry Errors for different speeds

The introduction of three dimensional polygons as loci for the feet brings up a new problem for the odometry. While in the 2 dimensional, rectangular case, the ground phase of the feet and therefore the theoretical odometry was easy to measure, this approach fails for the case of 3 dimensional polygons. The ground phase of the feet can not be identified, this is especially true for more than four vertices per polygon. Due to this fact, in a first approach we calculated the longest distance in x and y direction of the polygon and cut it at the maximum and minimum points into two halves, consisting of according segments of the polygon. The trajectory of these two halves of the polygon which is nearer to the ground was used as the ground phase and therefore used for calculating the odometry. The resulting odometry of this approach was

poorer in accuracy than the ones of the walks which are based on the two dimensional polygons and the faster speed of the walk did not countervail the disadvantage of the less precise odometry information.

With the ceiling camera infrastructure, we have an excellent tool to measure the speed of a robot within several seconds. This led us to the idea to create odometry correction tables which map a theoretical velocity to a real precisely measured odometry. The tables are 3 dimensional because of the ability of the robot to combine walking in x and y direction as well as rotation. Calibrating the x, y and rotational component in three separate tables would result in much less measurements but combining the three velocity components does not result in the same linear odometry combination. As a consequence we put a grid in the 3 dimensional control space, measured the real speed and therefore the odometry data at the nodes of the grid. Data between the nodes of the grids are interpolated at runtime.

V. Results

The described enhancements to the walking model and the according odometry information result in a big improvement of the overall performance of our robots.

The chosen evolution strategy in combination with the ceiling camera based fitness measurement worked well in terms of fast convergence toward an optimum. For the population size we encountered best results with $\mu = 6$ parents and $\lambda = 24$ offsprings. The tracking of the robot with the ceiling cam worked reliably but problems occured when during the walk evolution persons or other robots were moving under the ceiling camera and misleadingly detected as the walking robot. This was a problem because in case of a wrongly detected fast speed and therefore a good fitness, the according individual is in case of the $+$-strategy part of the next parent generation and might remain there forever, hampering the evolution process. The ,-strategy which selects only offsprings for the next parent generation was not vulnerable to wrongly detected too good fitness because a wrongly selected parent dies automatically in the next generation, however, the convergence speed of the ,-strategy was slower than the convergence speed of the $+$-strategy. Due to this fact, we decided to combine the advantages of both strategies and selected the parents for the next generation out of the former parents and the offsprings, but limited their life to two generations. This compromise led to fast convergence without being vulnerable to mismeasurements. Finding the fastest forward walk with this approach needed only 45 minutes of training. The progression of the individuals' fitness during this evolution is shown in Figure 8.

After several evolution runs, we found out, that 4 vertices per polygon seem to be sufficient and flexible enough to find promising walks. The search space was small enough to find optima for walks in all directions less than one hour. The found parameter sets for all walking directions result in a faster speed than all walks on the Aibo ERS-7 published so far. The maximum reached speeds are shown in comparison to the walking engine described in [2] in Figure 9. After these results we found out that a fitness based on maximum speed only is sufficient because most of the fast walks automatically tend to be smooth without shaking the robots' camera much, additionally, as shown in Figure 8, a lot of walking parameters with a very good fitness are generated during the evolution so that in the end we manually selected the walking parameters with the best overall properties. One parameter set resulted in a very fast forward speed of 51 cm/s but was quite unstable and shaking the robot while walking. Due to this fact we chose this parameter set (boost) only for certain situations in the game, like running for a fast distance toward the ball.

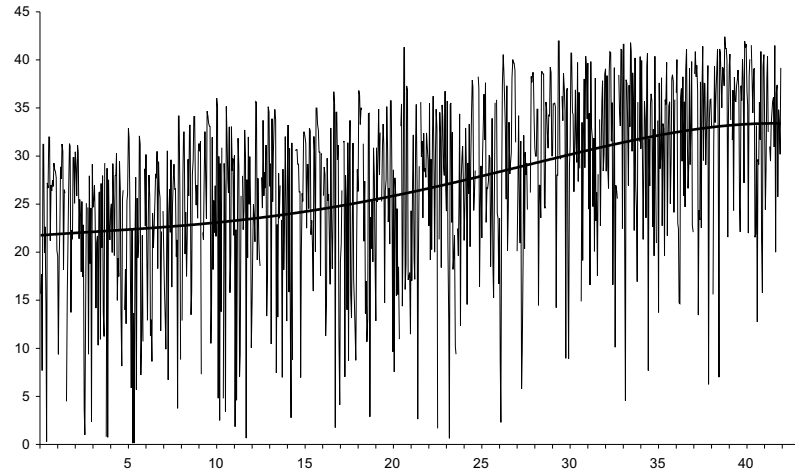
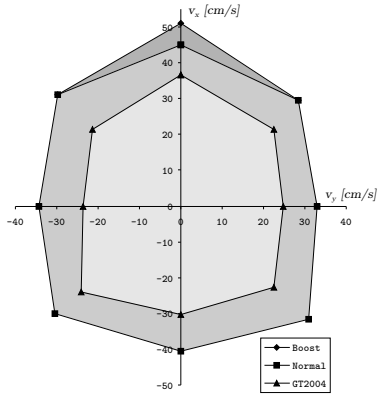


Fig. 8. The fitness of all offspring individuals during an evolution run. The bold line shows the trend of the fitness.



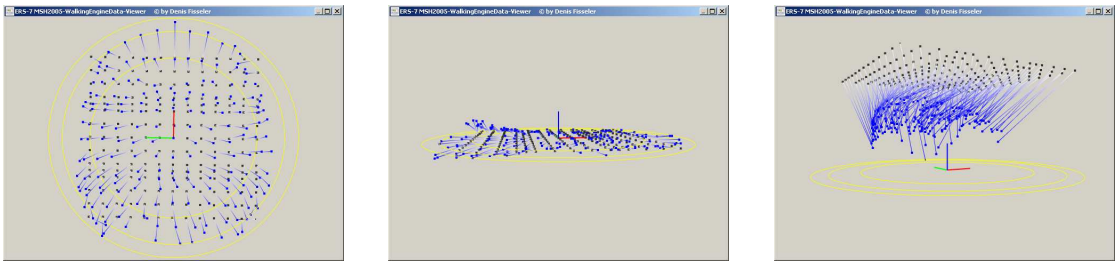
forward boost	51.0 cm/s
forward normal	45.1 cm/s
backwards	40.5 cm/s
sideways	34.4 cm/s
diagonal forward	42.1 cm/s
diagonal backwards	43.5 cm/s
rotation	200 °/s

Fig. 9. The achieved speeds in comparison to the speeds of the GermanTeam's walk from 2004.

Also the results of the odometry correction were better than the reference system. Due to this fact, our self localization could rely more on the odometry, making it possible to focus the attention of the camera more on game relevant objects like the ball rather than objects needed to localize. Some layers of the odometry correction tables are shown in Figure 10. The black dots represent the nodes of the grid, the blue dots show the according speed. Especially figure 10(c) visualizes the big difference between the theoretical and real odometry. The apparent disadvantage that the robots' real walking speed may differ from its controlled speed and only the odometry feedback is corrected using the look up tables is negligible because our closed loop level behavior control continuously readjusts the walk requests to the current situation.

VI. Conclusion

In this paper we presented both a more flexible approach for the leg control including a learning strategy for fast walks and a method for accurate odometry information. The learned walks with the described approach were in all directions more than 25% faster than existing walks, e.g. the walk of the German Team from 2004. The self localization profits from the accuracy improvements to the odometry information. Due to the fact, that the tracking of the robots' position can make more use of the odometry, the attention of the robot can focus more on game relevant objects like the ball or opponents rather than looking to landmarks to localize. The here described enhancements to the existing approaches were also one of the reasons why our four-legged, the Microsoft Hellhounds,



(a) Top view of the layer with translation movement only

(b) Side view of the layer with translation movement only

(c) Side view of the layer with translation movement with rotation component of 200 °/s

Fig. 10. Visualization of single layers of the odometry correction tables

has shown a very good overall performance in this year and is in the whole season 2005 undefeated.

ACKNOWLEDGMENT

The authors would like to thank the students of the project group 468 for their valuable work on the robots in the winter term 04/05 and the summer term 05. Further we would like to thank Microsoft MSDNAA for their financial support and the other universities of the GermanTeam for fruitful discussions and support.

REFERENCES

[1] B. Hengst, D. Ibbotson, S. B. Pham, and C. Sammut, "Omnidirectional locomotion for quadruped robots," in *RoboCup 2001 Robot Soccer World Cup V, A. Birk, S. Coradeschi, S. Tadokoro (Eds.)*, ser. Lecture Notes in Computer Science, no. 2377. Springer, 2002, pp. 368–373.

[2] U. Düffert and J. Hoffmann, "Reliable and precise gait modeling for a quadruped robot," in *RoboCup 2005*, ser. LNAI. Springer, 2005.

[3] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[4] T. Röfer, T. Laue, H. Burkhard, J. Hoffmann, M. Jüngel, D. Göhring, M. Lötzsch, M. Spranger, B. Altmeyer, V. Goetzke, O. von Stryk, R. Brunn, M. Dassler, M. Kunz, M. Risler, M. Stelzer, D. Thomas, S. Uhrig, U. Schwiegelshohn, I. Dahm, M. Hebbel, W. Nistico, C. Schumann, and M. Wachter, "German Team Report 2004," HU Berlin, TU Bremen, TU Darmstadt and University of Dortmund, Tech. Rep., 2004.

[5] P. Stone, K. Dresner, P. Fidelman, N. K. Jong, N. Kohl, G. Kuhlmann, M. Sridharan, and D. Stronger, "The UT Austin Villa 2004 RoboCup four-legged team: Coming of age," The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, Tech. Rep. UT-AI-TR-04-313, October 2004.

[6] D. Cohen, Y. H. Ooi, P. Vernaza, and D. D. Lee, "Robocup 2004 Legged Soccer Team," University of Pennsylvania, Tech. Rep., 2004.

[7] N. Kohl and P. Stone, "Machine learning for fast quadrupedal locomotion," in *The Nineteenth National Conference on Artificial Intelligence*, July 2004, pp. 611–616.

[8] T. Röfer, "Evolutionary gait-optimization using a fitness function based on proprioception," in *RoboCup 2004*, ser. LNAI. Springer, 2004.

[9] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[10] H.-P. Schwefel, *Evolution and Optimum Seeking*, ser. Sixth-Generation Computer Technology. New York: Wiley Interscience, 1995, p. 116 f.

[11] H.-G. Beyer, "Toward a theory of evolution strategies: Self-adaptation," *Evolutionary Computation*, vol. 3, no. 3, pp. 311–347, 1996.

[12] M. Sridharan, G. Kuhlmann, and P. Stone, "Practical vision-based monte carlo localization on a legged robot," in *IEEE International Conference on Robotics and Automation*, April 2005.

[13] T. Röfer and M. Jüngel, "Vision-based fast and reactive monte-carlo localization," in *IEEE International Conference on Robotics and Automation*, 2003.

[14] U. Düffert, "Quadruped walking: Modeling and optimization of robot movements," 2004.