ITAndroids 3D Soccer Simulation Team Description Paper for RoboCup 2015

Fábio Mello, Marcos Maximo, Mateus Coelho, and Samuel Pinto

Technological Institute of Aeronautics,
São José dos Campos, São Paulo, Brazil
fabio.mello50@yahoo.com.br
{maximo.marcos,mateuscoelho2009,sacepi}@gmail.com
itandroids-soccer3d@googlegroups.com
http://www.itandroids.com/

Abstract. ITAndroids was reestablished in mid-2011 by undergraduate students at the Technological Institute of Aeronautics. In the past, ITAndroids was a successful robotics competition group, winning several competitions in Brazil and Latin America. Unfortunately, the team dismantled and the expertise was lost over years of inactivity. After reestablished, the team has already won several competitions, especially in Latin America. This paper describes our developments in 3D Soccer Simulation, including development of an ZMP based omnidirectional walking engine, a kick algorithm, a positioning mechanism based on Delaunay Triangulation and use of particle filters for robot localization. Moreover, we discuss our plans for future development.

1 Introduction

ITAndroids is a robotics research group at Technological Institute of Aeronautics. The group was founded in 2006 by Jackson Matsuura. As required by a complete endeavor in robotics, the group is multidisciplinary and contains about 30 students from different undergraduate engineering courses.

In the last 3 years, we have achieved good results in competitions, especially in Latin America:

- 10th place in RoboCup 2D Soccer Simulation in RoboCup 2012;
- 1st place in RoboCup 2D Soccer Simulation in Latin American Robotics Competition (LARC) 2012;
- 2nd place in RoboCup 3D Soccer Simulation in LARC 2012;
- 3rd place in IEEE Humanoid Robot Racing in LARC 2012;
- 12th place in RoboCup 2D Soccer Simulation in RoboCup 2013
- Top 12 in RoboCup 3D Soccer Simulation in RoboCup 2013;
- Top 12 in RoboCup 3D Soccer Simulation in RoboCup 2013;
- 1st place in RoboCup 2D Soccer Simulation in Brazilian Robotics Competition (CBR) 2013;
- 2nd place in RoboCup 3D Soccer Simulation in CBR 2013;

- 1st place in RoboCup 2D Soccer Simulation in LARC 2014;
- 2nd place in RoboCup 3D Soccer Simulation in LARC 2014;
- 3rd place in RoboCup Humanoid KidSize in LARC 2014.

Our progress was largely supported by RoboCup community. Our current code is based on magmaOffenburg Agent-Framework (magma-AF) [1]. Many of our ideas were inspired by other teams work. Our first omnidirectional walk mechanism developed by our team was based on UT Austin Villa [2]. Also, we adapted a positioning mechanism developed by the 2D Soccer Simulation team HELIOS [20]. Furthermore, we have implemented a particle filter for robot localization.

This paper presents our recent development efforts. Sec. 2 describes our most important attempts in building a fast and stable walk. Sec. 3 presents an algorithm for kicking where we use the same balancing strategy that we use for walking. Sec. 4 presents a positioning method based on Delaunay Triangulation (DT) [24] that was adapted from the 2D Soccer Simulation team HELIOS. Sec. 5 explains how our robot localizes itself in the field. Finally, Sec. 6 concludes the paper and shares our vision for future implementation.

2 Walking

In 3D Soccer Simulation league, most actions of the robots are highly dependent on its ability to walk. Therefore, a great amount of our team efforts was focused on walking. In order to find a good walking method, several ideas were tested. Our lastest walking models are described in this section.

2.1 Parametric Omnidirectional Walk

One of the walking methods tested by our team was based on [2]. The walking engine developed used a similar parametrization for the trajectory. However, our development was focused on being able to achieve a fast and stable walking without the need of using much computational resources during the optimization process.

On the optimization of the parameters of a walking trajectory, one of the biggest problems is the need of adapting the parameters to different goals. The two main goals in this task are walking as fast as possible and being as stable as possible. Since the combination of more than one goal on the same evaluation function commonly does not provide a good tradeoff between these goals, the problem was divided in two coupled optimization problems with different sets of parameters to be optimized and different goals for each problem.

The first problem was maximizing the stability of the movement keeping the speed constant. However, in order to take into account the influence of the size of the step in the movement stability, we kept the ratio between the size and the duration of the step constant and not both of these parameters. The second problem was to increase the speed of the robot as much as possible without making the movement too unstable.

The major advantage of this aproach is that, using each parameter to optimize the feature of the walking that is more influenced by it, the influence of a change on a parameter is noticed earlier; thus, allowing a faster optimization process. For instance, the height to which the moving foot of the robot is lifted during a step influences the stability of the movement, but it does not influence the speed of the movement unless the feet of the robot are sliding on the floor. Therefore, a evaluation function that considers both the speed and the stability of the movement might fail to notice this change in the stability while it would be much easier to notice it if considering only the stability of the robot.

In the end, the optimization was composed of two alternating steps, one making the movement as stable as possible while keeping the speed constant and the other one increasing the speed as much as possible while keeping the other parameters constant. Using this strategy, it was possible to manually tweak the parameters and achieve a reasonable fast and stable motion. In the future, we intend to adapt this strategy to a method for automatically setting the parameters without the need of much computational effort.

2.2 ZMP Based Omnidirectional Walking Engine

In general terms, the walking engine follows the flux presented on Figure 1. The input to the algorithm is the desired velocity $\mathbf{v} = [v_x, v_y, v_\psi]^T$ with respect to the local coordinate system of the robot. Then, at the beginning of a new step, poses for the torso and the swing foot are selected for achieving the expected displacement at the end of the step. So, a trajectory for the center of mass (CoM) that keeps the Zero Moment Point (ZMP) at the center of the support foot is computed by using an analytic solution of the 3D-LIPM equation. We approximate the CoM by a fixed position in the torso. The trajectory of the swing foot is obtained by interpolating between the initial and final poses of this foot. Finally, joints angles are calculated through Inverse Kinematics (IK) considering the poses of the support and swing feet. Note that the module "Next Torso and Swing Poses Selector" is called once for step, while the others are executed at the update rate of the joints.

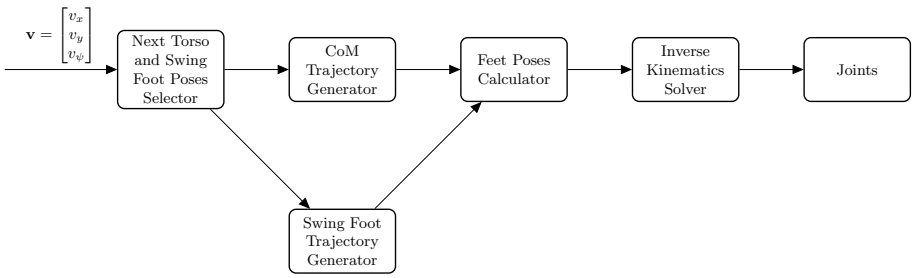


Fig. 1. Walking Engine overview.

A humanoid robot must satisfy dynamical constraints to remain stable. Moreover, the robot geometry imposes constraints: leg reachability is limited by leg physical dimensions and we also do not want movements where the legs collide. To achieve dynamic stability, our strategy is to generate a CoM trajectory that keeps the ZMP at the center of the support foot during single support. We may at least restrict the initial and final positions of this CoM trajectory, as will be explained below. Hence, our requirement is to have the robot matches the omnidirectional model only at the beginning and at the end of the step. Assuming constant $\mathbf{v}$, we may compute the expected pose after a step duration T:

$$\begin{bmatrix} x(t+T) \\ y(t+T) \\ \psi(t+T) \end{bmatrix} = \begin{bmatrix} x(t) + 2\frac{v_x}{v_\psi}\sin\left(\frac{v_\psi T}{2}\right)\cos\left(\psi(t)+\frac{v_\psi T}{2}\right) - 2\frac{v_y}{v_\psi}\sin\left(\frac{v_\psi T}{2}\right)\sin\left(\psi(t)+\frac{v_\psi T}{2}\right) \\ y(t) + 2\frac{v_x}{v_\psi}\sin\left(\frac{v_\psi T}{2}\right)\sin\left(\psi(t)+\frac{v_\psi T}{2}\right) + 2\frac{v_y}{v_\psi}\sin\left(\frac{v_\psi T}{2}\right)\cos\left(\psi(t)+\frac{v_\psi T}{2}\right) \\ \psi(t) + v_\psi T \end{bmatrix} \tag{1}$$

For our multibody humanoid robot, it is convenient to select a body part as representative of the whole robot motion: we choose the torso for this. Thus, at the beginning of a new step, given the current torso and swing foot poses, an algorithm plans the torso and swing foot poses at the end of the step to make the torso arrive at the pose dictated by Equation (1) while trying to satisfy geometric constraints. This algorithm is based mainly on heuristics.

We still need to move the robot without losing balance. To reason about the robot dynamics, we approximate it using the 3D Linear Inverted Pendulum Model (3D-LIPM) [12]:

$$\mathbf{x}_{ZMP} = \mathbf{x}_{CoM} - \frac{z_{CoM}}{g}\ddot{\mathbf{x}}_{CoM} \tag{2}$$

Where $\mathbf{x}_{ZMP} = [x_{ZMP}, y_{ZMP}]^T$ is the ZMP position, $\mathbf{x}_{CoM} = [x_{CoM}, y_{CoM}]^T$ is the CoM position, z_{CoM} is the CoM height, and g is the acceleration of gravity. The ZMP is kept at the center of the support foot during single support and moves it from the current support foot to the next one during double support.

However, the difference between the multibody humanoid robot dynamics and 3D-LIPM and perturbations, such as external forces and uneven terrain, will prevent the ZMP to match the reference. The robot is able to accommodate ZMP error up to the margins of the support polygon without tipping, which is often sufficient to allow open-loop walking if no strong perturbations are present. Nevertheless, closed-loop balancing strategies are useful to make the walking more robust. We have tried using the angular velocities measured from the gyrometer to stabilize the walk, which proved effective.

Unfortunately, due to lack of time, our current code still uses the walking engine described in the previous subsection. Nevertheless, we have developed the walking engine explained in this section for our Humanoid KidSize robot. This engine have proved to work adequately both in simulation and in the real robot. Thus, we expect to have this new walking engine integrated with our Soccer 3D code long before the competition.

3 Kick

We consider that kicking is a motion where the biped starts in a stand position, kicks the ball and returns to the same stand position. Moreover, during kicking, one foot is taken off the ground, henceforth referred as kicking foot, while the other one is kept on the ground as support foot. This description suggests breaking the motion in phases, thus we divided it in the following 5 phases:

- Phase A: the robot moves the ZMP to the center of the support foot to allow the kicking foot to be taken off the ground in the next phase without balance loss.
- Phase B: the robot takes the foot off the ground and position it to prepare for kicking the ball.
- Phase C: the robot kicks the ball.
- Phase D: the robot places the kicking foot on the ground.
- Phase E: the robot goes back to the stand position (ZMP is moved to the torso projection on the ground).

During phases B, C and D, the robot is in single support, so stability is of concern. Based on this perception, we use the same algorithm we used to balance walking for balancing kicking. Again, we constraint the CoM to maintain a constant height z_{CoM}, so the dynamics becomes linear.

The current kick integrated in our code is the one from the base team (magma). We expect to implement the algorithm explained in this section soon.

4 Positioning Using Delaunay Trangulation

A technique popularized by Helios in the 2D Soccer Simulation League, the Delaunay Triangulation Positioning [20] consists of hard-coding the players's positions for a certain number of ball positions, and then computing the team's positioning for any ball position through a 2D linear reduction based on the formations for the known ball positions. Our team adapted this idea to the 3D Simulation.

More specifically, the triangulation works on a set of possible positions for the ball, each of which containing the ideal positions for the players if the ball happens to be there. The Delaunay Triangulation is such that all circumcircles are empty, the resulting grid of triangles produces a graph covering the whole field. In order to compute the triangulation, we used the library [22].

After the triangulation is done, we can calculate the linearization parameters in order to smoothly adjust the positions in which the player will be. To do so, we use for each triangle a linear funtion of the ball position to determine the position in which a given player will be if the ball is inside that triangle. In order to determine the linear function, it is stated that, in the vertices, all players should position as asigned. This way, it is possible to smoothly interpolate the positions assigned to each player to an arbitrary ball position.

Until now, our efforts were mainly focused on constructing a parser compatible to the formation editor released by Helios in the 2D Soccer Simulation League [23] and implementing the linear functions to perform the interpolation. Therefore, we are currently using the same formation configuration used in agent 2d [6]. In the future, the team intends to adapt the positioning in order to better fit the specific aspects of the 3D Soccer Simulation League. Also, the team aims to use this thecnique to predict the positioning of other teams as already done by some teams in the 2D Soccer Simulation League [20, 21].

5 Localization

5.1 Stochastic Modeling

The robot state $s = [x\ y\ \psi]$ can be written in as a function of its previous state, as in following:

$$\begin{bmatrix} x_k \\ y_k \\ \psi_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \psi_{k-1} \end{bmatrix} + \begin{bmatrix} u_{k_x} \\ u_{k_y} \\ u_{k_\psi} \end{bmatrix} \tag{3}$$

In which $u_k^{\mathbf{T}}$ is a multivariate gaussian vector with mean $[\mu_x\ \mu_y\ \mu_\psi]$, given by the odometry information, and covariance matrix σ_k.

The vision returns information about landmarks observed. The probability density function associated to the robot observation of the point a associated to the landmark m at time k, $p^m_{a,k}(z_0|s_{0:k})$, is a multivariate gaussian function with mean $\mu^m_{point\ a,k}$ and covariance matrix $\sigma^m_{point\ a,k}$.

5.2 Particle filters

The localization is responsible for recursively estimating the robot‘s state in the field, i.e. the vector $s = [x\ y\ \psi]$. However, the robot is not able to observe directly its state. Therefore, in order to estimate its state, informations from the odometry and from the computational vision are merged using a stochastic algorithm known as particle filter. This algorithm aims to approximate the probability density distribution function by the density of a set of discrete hipothesys named particles. Each particle j can be understood as a hypothesys $[x^{(i)}\ y^{(i)}\ \psi^{(i)}]$. This algorithm was implemented as described in [11] and is explained in algorithm 1.

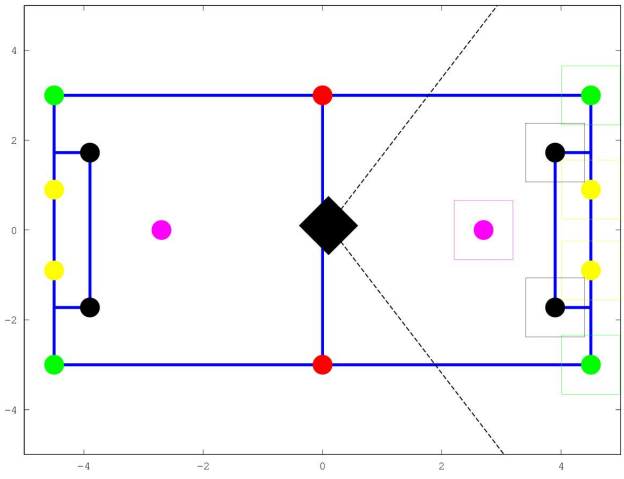
The mathematical model described in the Stochastic Modeling subsection was simulated in a simulator built using Matlab. This simulator is a tool for debuging localization algorithms. It was implemented to simulate the behavior of the localizations signals considering a 2D environment. It simulates the robot's kinematics and observations, including the randomness. The simulator also has a MEX interface, therefore it can be used to test both MATLAB localization code and C++ code. The fig. 5.2 shows the simulator.

Then, the obtained signals were processed using the techniques previously described. 200 particles were used and the results have shown that the algorithm delivered very accurated results, as shown in fig. 5.2.

begin
  for $i = 1, ..., N_p$ **do**
    $Draw\ s_k^{(i)}\ with\ p(s_0|z_0)$;
    $w_k^{(i)} \leftarrow \frac{1}{N_p}$;
  end
  for *every* k **do**
    for $i = 1, ..., N_p$ **do**
      $Draw\ s_k^{(i)}\ with\ p(s_k|s_{k-1}^{(i)})$;
      $\hat{w}_k^{(i)} \leftarrow w_{k-1}^{(i)} p(z_k|s_k^{(i)})$;
    end
    $B \leftarrow \sum_{k=1}^{N_p} \hat{w}_k^{(i)}$;
    for $i = 1, ..., N_p$ **do**
      $w_k^{(i)} \leftarrow \frac{\hat{w}_k^{(i)}}{B}$;
    end
    $N_{eff} \leftarrow \frac{1}{\sum_{k=1}^{N_p} {w_k^{(i)}}^2}$;
    if $N_{eff} > 0.4$ **then**
      $Ressample$;
    end
  end
end

Algorithm 1: Localization Algorithm.

Fig. 2. Localization simulator

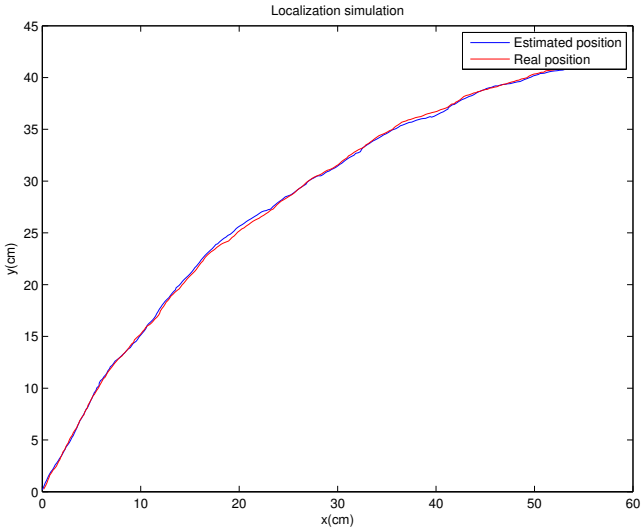


6 Conclusions and Future Work

This paper presented the latest efforts of team ITAndroids 3D. In the last 3 years, we have won several competitions, especially in Latin America.

Our fast progress was highly supported by the RoboCup community: our current code is based on magma-AF base team [1] and most of our implementation was greatly inspired by other teams work [2, 20, 15–19].

Fig. 3. Localization results.



For immediate work, we expect to focus on implementing and tuning some of the methods described in this paper. We expect to accomplish this long before RoboCup. Moreover, since we are reaching a state where our low level skills are reliable, we intend to start focusing in our decision making, which is currently very simple.

References

1. magmaOffenburg Agent-Framework, 2011, online, avaliable at: `http://robocup.hs-offenburg.de/downloads/magma3D-2011Release.tar.gz`, consulted on February 2012.
2. MacAlpine P., Urieli, D., Barrett, S., Kalyanakrishnan, S., Barrera, F., Lopez-Mobilia, A., Ştiurcă, N., Vu, V., Stone, P.: UT Austin Villa 2011 3D Simulation Team Report. Technical Report, The University of Texas at Austin, Department of Computer Science, AI Laboratory (2011)
3. Haider, S., W., M.-A., Raza, S., Johnston, B., Abidi, S., Sharif, U., Raza, A.: Karachi Koalas3D Simulation Soccer Team, Team Description Paper for World RoboCup 2012 (2012)
4. Yang, L., Chew, C.-M., Zielinska, T., Poo, A.-N.: A Uniform Biped Gait Generator with Offline Optimization and Online Adjustable Parameters. In: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4435–4440 (2006)
5. Vukobratovic, M., Borovac, B., Surdilovic, D.: Zero-Moment Point Proper Interpretation and New Applications. In: Proceedings of the 2nd IEEE-RAS International Conference on Humanoid Robots, pp. 237–244 (2001)
6. agent2d, `http://pt.sourceforge.jp/projects/rctools/downloads/51943/agent2d-3.1.0.tar.gz/`

7. Akiyama, H., Shimora, H.: HELIOS2010 Team Description (2010)
8. Bai, A., Zhang, H., Lu, G., Jiang, M., Chen, X.: WrightEagle 2D Soccer Simulation Team Description 2012 (2012)
9. http://wiki.processing.org/w/Triangulation
10. fedit2-0.0.0, 2010, online, avaliable at: http://pt.sourceforge.jp/projects/rctools/downloads/48791/fedit2-0.0.0.tar.gz/, consulted on February 2012.
11. M. G. S. Bruno. Sequential Monte Carlo Methods for Nonlinear Discrete-Time Filtering. Technological Institute of Aeronautics.
12. S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3D Linear Inverted Pendulum Mode: A Simple Modeling for a Biped Walking Pattern Generation. In Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2001.
13. Mello, F., Muxagata, E., Maximo, M.: ITAndroids 3D Soccer Simulation Team Description 2013 (2013).
14. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte Carlo Localization for Mobile Robots. In: IEEE International Conference on Robotics and Automation (ICRA'99) (1999)
15. Shafii, N., Aslani, S., Nezami, O.M., Shiry, S.: Evolution of Biped Walking Using Truncated Fourier Series and Particle Swarm Optimization. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009, pp. 344–354 (2010)
16. Shafii, N., Khorsandian, A., Abdolmaleki, A., Jozi, B.: An Optimized Gait Generator Based on Fourier Series Toward Fast and Robust Biped Locomotion Involving Arms Swing. In: Proceedings of the IEEE International Conference on Automation and Logistics (2009)
17. Shafii, N., Reis, L.P., Lau, N.: Biped Walking Using Coronal and Sagittal Movements Based on Truncated Fourier Series. In: RoboCup 2010: Robot Soccer World Cup XIV, Lecture Notes in Computer Science, pp. 324–335 (2010)
18. Hester, T., Stone, P.: Negative Information and Line Observations for Monte Carlo Localization. In: IEEE International Conference on Robotics and Automation (ICRA'08) (2008)
19. Coltin, B., Veloso, M.M.: Multi-Observation Sensor Resetting Localization with Ambiguous Landmarks. In: Proceedings of AAAI'11, the Twenty-Fifth Conference on Artificial Intelligence, San Francisco, CA (2011)
20. Akiyama, H., Shimora, H.: HELIOS2010 Team Description (2010)
21. Bai, A., Zhang, H., Lu, G., Jiang, M., Chen, X.: WrightEagle 2D Soccer Simulation Team Description 2012 (2012)
22. http://wiki.processing.org/w/Triangulation
23. fedit2-0.0.0, 2010, online, avaliable at: http://pt.sourceforge.jp/projects/rctools/downloads/48791/fedit2-0.0.0.tar.gz/, consulted on February 2012.
24. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applicaitions. Springer-Verlag (2008)