ITAndroids 3D Soccer Simulation Team Description 2013

Eric Muxagata, Fábio Mello, and Marcos Maximo

Technological Institute of Aeronautics,
São José dos Campos, São Paulo, Brazil
fabio.mello50@yahoo.com.br
{ericgmconrado,maximo.marcos}@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 were 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. Now, the team is recovering fast and has already placed 1st in 2D and 3D Soccer Simulation leagues in Latin American Robotics Competition (LARC) 2012. This paper describes our latest developments in 3D Soccer Simulation, including development of an omnidirectional walking mechanism, application of Delaunay Triangulation to positioning and use of Monte Carlo localization to improve the accuracy of our localization system. Moreover, we discuss our plans for future development.

1 Introduction

ITAndroids were created in 2005 by Jackson Paul Matsuura, who was a graduate student at the Technological Institute of Aeronautics (ITA) at the time. The group rapidly consolidated itself in Brazil and Latin American, winning several competitions, including brazilian versions of 2D and 3D Soccer Simulation leagues. However, due to a personal decision of Matsuura to focus his efforts on building up the whole brazilian robotics competitions scenario, the team faded with time and dismantled.

By mid-2011, a group of undergraduate students reestablished ITAndroids 2D Soccer Simulation. The team evolved fast and was able to qualify for RoboCup Mexico 2012 where it placed 10th, the best position ever achieved by a brazilian soccer simulation team in RoboCup. Motivated to study low-level robotics problems, like humanoid locomotion, and to apply the high-level knowledge acquired in 2D Soccer Simulation in a more realistic environment, some members of ITAndroids 2D team decided to start working with 3D Soccer Simulation league by early-2012.

On October 2012, ITAndroids participated in Latin American Robotics Competition (LARC) 2012, where it placed 1st in both 2D and 3D leagues. Despite being our first 3D competition, our 3D team won every match it played, scoring a total of 22 goals and not suffering any in the whole competition.

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. The omnidirectional walk mechanism developed by our team was based on UT Austin Villa [3]. Also, we adapted a positioning mechanism developed by the 2D Soccer Simulation team HELIOS [5]. Furthermore, we have implemented a particle filter that considers the history of sensors measurements and actions in order to provide a better localization estimate, instead of relying solely on the latest visual perception.

This paper presents our recent development efforts. Sec. 2 describes our most important attempts in building a fast and stable walk. Sec. 3 presents a positioning method based on Delaunay Triangulation (DT) [6] that was adapted from the 2D Soccer Simulation team HELIOS. Sec. 4 explains how we implemented Monte Carlo localization (MCL) [7] in our team. Finally, Sec. 5 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, some of which are described in this paper.

2.1 Parametric Omnidirectional Walk

One of the walking methods tested by our team was based on [3]. 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.

During 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, an 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 Truncated Fourier Series

Since a humanoid walking is a periodic movement, it is reasonable that the angular trajectories of joints can be written as Fourier Series. Actually, [8] proved that Truncated Fourier Series (TFS) can be used to generate angular trajectories that yield a valid gait by ZMP criterion [9].

In a latter series of works [10–12], TFS was used to actually generate stable gaits for the simulated Nao from SimSpark. In these works, the most complex gait model uses the following joints from both body sides: hip-pitch, knee, foot-pitch, shoulder-pitch and hip-roll. All joints trajectories are generated by TFS, except for foot-pitch joints, whose angles are set to compensate hip-pitch and knee angles in a way that keeps the feet always parallel to the ground. Observing actual trajectories of a human gait, [12] assumes that joints angles should follow the equations:

$$\theta_{hip-pitch}(t) = \begin{cases} O_{hp} + A\sin\left(\dfrac{2\pi t}{T}\right), & t \in \left[kT, \dfrac{T}{2} + kT\right), k \in \mathbb{Z} \\ \\ O_{hp} + B\sin\left(\dfrac{2\pi t}{T}\right), & t \in \left[\dfrac{T}{2} + kT, (k+1)T\right), k \in \mathbb{Z} \end{cases} \tag{1}$$

$$\theta_{knee}(t) = \begin{cases} O_k + C\sin\left(\dfrac{2\pi(t-t_2)}{T}\right), & t \in \left[kT, \dfrac{T}{2} + kT\right), k \in \mathbb{Z} \\ \\ O_k, & t \in \left[\dfrac{T}{2} + kT, (k+1)T\right), k \in \mathbb{Z} \end{cases} \tag{2}$$

$$\theta_{shoulder}(t) = -D \sin\left(\frac{2\pi t}{T}\right) \quad (3)$$

$$\theta_{hip-roll}(t) = \begin{cases} E \sin\left(\frac{2\pi t)}{T}\right), & t \in \left[kT, \frac{T}{2} + kT\right), k \in \mathbb{Z} \\ \\ 0, & t \in \left[\frac{T}{2} + kT, (k+1)T\right), k \in \mathbb{Z} \end{cases} \quad (4)$$

Where O_h, O_k, A, B, C, D, E, t_2 and T are parameters. Therefore, the problem of describing a gait by joints angular trajectories is reduced to specifying these 9 parameters. The only difference in our approach from the one described above is that we assume that shoulder movement may have different amplitudes when swinging the arm forward or backwards, so the model we used has 10 parameters.

One approach to find the set of parameters that yield the best gait is to use an optimization algorithm. We decided to use Particle Swarm Optimization (PSO) [19] for this due to the success in using it in our 2D Soccer Simulation team. Note that to create a phase difference between $\theta_{hip-pitch}$ and θ_{knee}, one may assume the constraint $0 < t_2 < T$.

After the optimization process, the walk learnt happened to be much faster than the base team walk. To compare this walk against magma-AF's walk, we ran 10 experiments with each walking method, where the robot was commanded to move as fast as possible towards positive x direction during 15 seconds. At the end of each run, the distance traveled on x direction was measured. Table 1 summarizes the results.

The drawback is that the walk provided by this method is not omnidirectional, so it is very inconvenient for soccer domain, where a robot often needs to change direction of movement. We are still analyzing if switching to this walk when fast forward speed is needed may improve team performance.

Walking Method	Mean	Standard Deviation
Optimized TFS-based walk	7.41	0.55
magma-AF	3.93	0.4

Table 1. Comparison between the optimized TFS-based walk and the original magma-AF walk. Results show distances traveled on x direction during 15 seconds, averaged over 10 runs.

2.3 Variational Calculus

Another method tested used variational calculus to calculate the trajectory that minimizes a given cost functional. In order to achieve dynamic equilibrium and to keep the angles of the joints within its physical constraints, the functional used corresponded to the integral over time of a function of the joint angles and its two first derivatives that tends to infinity when the configuration reaches either a state close to the physical constraints of the joints or a state in which the robot is in the eminence of falling according to the ZMP criterion [9].

In order to compute the trajectory, it was used the Euler-Lagrange equation derived from the chosen functional, which is a fourth order diferential equation system, since the functional depends on the the two first derivatives of the joint angles. Therefore, in order to specify a given trajectory, it is necessary to have the initial joint angles and its four first derivatives. Since the dynamics of the system only specifies the joint angles and their two first derivatives, the other initial conditions were used in order to specify the desired final state of the robot.

Altough the cost function used tends to infinity when the robot aproachs a configuration where it is in the eminence of falling, it is not garanteed that the cost functional is not integrable. Thus, there might be trajectories in which the robot exceed the borders in which it can move. Indeed, after testing the trajectories generated using a random set of initial conditions, there were cases in which the robot reached non-equillibrium states.

In order to choose the best initial conditions it is necessary to predict both position in which the foot of the robot would reach the ground and whether or not the trajectory would lead to a state in which the robot is not in equilibrium. Both of these predictions were made using feedforward neural networks. However, the predictions made were not accurate enough to determine a good set of initial conditions of the robot.

3 Positioning Using Delaunay Triangulation

A technique popularized by Helios in the 2D Soccer Simulation League, the Delaunay Triangulation Positioning [20] consists of hard-coding the playerss positions for a certain number of ball positions, and then computing the teams 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 assigned. 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 [18]. 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 technique to predict the positioning of other teams as already done by some teams in the 2D Soccer Simulation League [20, 21] and already implemented in our 2D Soccer Simulation team.

4 Localization

One of the biggest challenges in Mobile Robotics is Localization, i.e. have the robot to localize itself in the world. In 3D Soccer Simulation, this means to have each agent deduce its global coordinates relative to the soccer field. To solve this problem, a common approach in modern Mobile Robotics is to use a Bayes filter, which iteratively incorporates sensors information and actions to construct a probabilistic estimate of the robot position. Since implementing this technique directly is not feasible computationally, approximated techniques, such as Kalman filter [16] or particle filter [17], are often employed. We decided to use Monte Carlo localization (MCL) [7], which uses a particle filter to solve the Localization problem, because it is one the most efficient methods [15] and we know that some teams in 3D had success in using it [3, 4].

Our MCL implementation was greatly inspired by the work explained in [13]. Each particle mantains a pose estimate represented by a 3-dimensional vector (x, y, θ), where x and y are global field coordinates and θ is the horizontal angle the body of the robot is heading.

In the sensing phase, we currenly use only landmarks observations. Denote d_j and $\hat{d}_{ij}$ as the measured (using vision perceptor) and expected (considering the pose of particle p_i) distances to the visible landmark l_j (θ_j and $\hat{\theta}_{ij}$ have analogous meanings), then a score (weight) is given to each particle following (5), where the product iterates over all visible landmarks. If no landmark is seen in the current cycle, no sensing update is done. Note that since we use only visual information to update our filter, the sensing phase is run every 3 cycles. To improve the filter performance, we expect to incorporate line observations in future developments.

$$w_i = \prod_j \left\{ \exp\left[-\left(\frac{d_j - \hat{d}_{ij}}{\sigma_d} \right)^2 \right] \cdot \exp\left[-\left(\frac{\theta_j - \hat{\theta}_{ij}}{\sigma_\theta} \right)^2 \right] \right\} \tag{5}$$

For motion update, our first attempt was to use the velocities commanded to our walking engine directly. However, it soon became clear that our walking engine was performing slower than expected, specially close to maximum velocities. So, we tried to fix this by simply saying that the actual velocities were $(v_{x'}^{act}, v_{y'}^{act}, v_{\theta'}^{act}) = (\alpha \cdot v_{x'}^{com}, \beta \cdot v_{y'}^{com}, \gamma \cdot v_{\theta'}^{com})$ (x', y' and θ' are relative to the robot's coordinates frame) and manually tweaking α, β and γ. Still, we are not satisfied with the motion model performance, specially with the fact that we are not modeling noise. Therefore, we are preparing experiments to determine more accurately how actual velocities relate to the commanded ones. Moreover, we expect to model actuation noise using gaussian models, as recommended in [15]. Note that to avoid having more motion than sensing phases, we also run a motion phase every 3 cycles.

To avoid the kidnapped robot problem, which happens in 3D domain when the server teleports the agent, every update our filter substitute the s worst particles by particles drawn at random from the soccer field. Also, for each particle p_i a random walk displacement is added proportionally to $(1 - w_i)$, similarly to the strategy described in [13].

Finally, our MCL component determines the estimated agent pose by a weighted average of the particles. Moreover, a weighted standard deviation is computed as an estimate to how well the agent is localized.

For future implementation, besides the improvements that we discussed to sensing and motion updates, we plan to do a smarter particle resetting using landmarks observations and a Adaptive-MCL strategy to select how many particles to reset [14, 15], so our localization may better recover from a kidnap. Furthermore, we want to run an optimization algorithm (e.g. PSO) with an adequate evaluation function to optimize the filter parameters, since the current values were tweaked manually by visually comparing the robot real position with the filter's pose estimate drawn in Roboviz software [2].

5 Conclusions and Future Work

This paper presented the latest efforts of team ITAndroids 3D. Despite being a new team in 3D Soccer Simulation, we have already won Latin American Robotics Competition 2012.

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 [3, 5, 10–14].

For immediate work, we expect to focus on the following: build a optimization scenario to optimize our walking engine, optimize specific low-level skills (e.g. getting up), develop better kicks and refine our localization system. We expect to have these modifications done and integrated in our team until RoboCup.

References

1. magmaOffenburg Agent-Framework, 2011, online, avaliable at: `http://robocup.hs-offenburg.de/downloads/magma3D-2011Release.tar.gz`, consulted on February 2012.
2. Roboviz, 2011, online, avaliable at: `https://sites.google.com/site/umroboviz/`, consulted on February 2012.

3. 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)
4. 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)
5. Akiyama, H., Shimora, H.: HELIOS2010 Team Description (2010)
6. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applicaitions. Springer-Verlag (2008)
7. 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)
8. 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)
9. 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)
10. 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)
11. 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)
12. 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)
13. Hester, T., Stone, P.: Negative Information and Line Observations for Monte Carlo Localization. In: IEEE International Conference on Robotics and Automation (ICRA'08) (2008)
14. 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)
15. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005)
16. Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME–Journal of Basic Engineering 82, pp. 35–45 (1960)
17. Smith, A.F.M., Gelfand, A.E.: Bayesian Statistics Without Tears: A Sampling-Resampling Perspective. American Statistician 46, pp. 84–88 (1992)
18. agent2d, `http://pt.sourceforge.jp/projects/rctools/downloads/51943/agent2d-3.1.0.tar.gz/`
19. Eberhart, R.C.; Kennedy, J.: A New Optimizer Using Particle Swarm Theory. In: Sixth International Symposium on Micro Machine and Human Science (MHS'95). Nagoya, Japan. (1995)
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.