Predictive Control for Behavior Generation of Omni-Directional Robots

João Cunha, Nuno Lau, João Rodrigues, Bernardo Cunha, José Luis Azevedo

IEETA / Department of Electronics, Telecommunications and Informatics
University of Aveiro, Portugal
{joao.cunha,nunolau,jmr,mbc,jla}@ua.pt

Abstract. This paper describes the approach developed by the CAMBADA robotic soccer team to address physical constraints regarding omni-directional motion control, with special focus on system delay. CAMBADA robots carry inherent delays which associated with discrete time control results in non-instant, non-continuous control degrading the performance over time. Besides a natural maximum velocity, CAMBADA robots have also a maximum acceleration limit implemented at software level to provide motion stability as well as current peaks avoidance on DC motors. Considering the previous constraints, such as the cycle time and the overall sensor-action delay, compensations can be made to improve the robot control. Since CAMBADA robots are among the slowest robots in the RoboCup Medium Size League, such compensations can help to improve both several behaviours as well as a better field coverage formation.

1 Introduction

CAMBADA[1] is the University of Aveiro's robotic soccer team. The project was started in 2003 by the IEETA[2] ATRI[3] research group which involves researchers from different scientific areas such as image analysis and processing, control, artificial intelligence, multi-agent coordination, sensor fusion.

CAMBADA currently participates in RoboCup's Middle Size League. RoboCup[4] is an international project focused on fostering innovation in robotics and related fields. In this league a team of 4 to 6 robots with maximum dimensions of $50cm \times 50cm \times 80cm$ and a maximum weight of 40kg play soccer in a $12m \times 18m$ field complying with robot-adapted official FIFA rules [1].

Given the dynamic environment of a soccer game, accurate robot movement plays a crucial role since most of the game duration the team's success depends

[1] CAMBADA is an acronym for Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture.

[2] Instituto de Engenharia Electrónica e Telemática de Aveiro - Aveiro's Institute of Electronic and Telematic Engineering

[3] Actividade Transversal em Robótica Inteligente - Transverse Activity on Intelligent Robotics

[4] http://www.robocup.org/

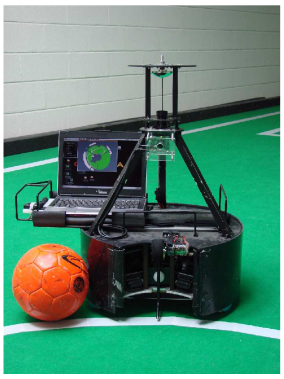


Fig. 1. A CAMBADA robot.

on placing the robot in the desired position. This paper explains some methods applied in the CAMBADA project to effectively control an omni-directional robot. In Section 2 an overview of the CAMBADA robot's motion and various physical constraints are presented. The predictive control implementation is discussed in Section 3. Section 4 describes the methods involved in the parameters estimation of the model and the respective results are in Section 5. Finally, Section 6 presents the conclusions.

2 Omni-Directional Robot Motion

As stated before, CAMBADA robots have an omni-directional motion. This is accomplished by a set of 3 swedish wheels placed at the periphery of the robot at angles that differ 120 degrees from each other. Such configuration enables a robot to move in any direction while facing any orientation. Since all degrees of freedom are controlable the robot's motion is holonomic.

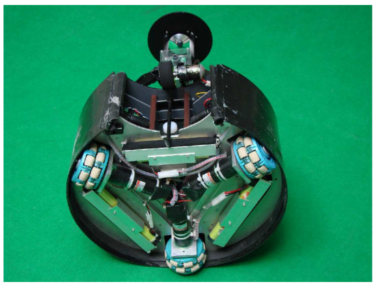


Fig. 2. Detail of the wheel configuration which enables an holonomic motion.

To successfully move a robot to a desired pose, setpoints to each of the robot's wheels must be provided at every control cycle. This is firstly done by translating the desired pose from field coordinates to robot coordinates which gives an offset relative to the robot's current pose. From this offset, error measurements are determined that are then applied to PID controllers to determine the linear and angular velocities to be applied at the robot's frame. These are in turn mapped to setpoints, the velocities to be applied at each wheel, using the robot's kinematic model [2], to achieve the desired movement.

Ideally the robot's linear and angular velocities would be mutually independent. This means that a robot could move in a straight line while rotating over itself. In practice, such control is negatively affected by a diverse array of factors that if ignored results in an undesired robot movement.

While rotating and moving at the same time, the robot deviates considerably from the ideal straight line. This deviation is consistently to the right when rotating clockwise and to the left when rotating counter-clockwise.

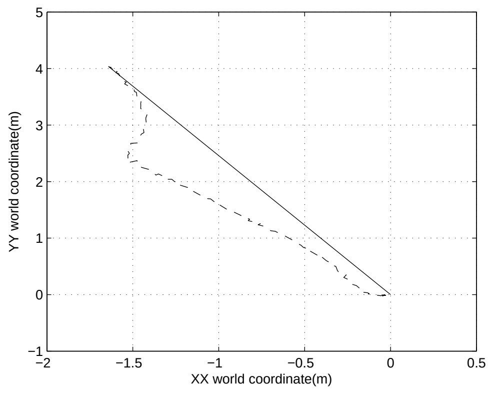


Fig. 3. Deviation in the robot movement when it rotates and moves simultaneously. The dashed line is the real robot path and the solid line would be the ideal straight path. The robot is rotating clockwise and moving towards (0,0).

An initial solution to minimize the deviation consists in rotating the robot at the initial position and then applying the linear velocity. This solution is not ideal since it doesn't take advantage of holonomic motion. Also, CAMBADA robots are among the slowest moving robot teams in the MSL [3][4][5][6], so this solution only emphasizes the speed difference.

The physical restraints affecting holonomic motion control dealt with in this paper are: discrete time control, maximum acceleration, actuator saturation, and control latency. Given the difficulty in quantification, wheel slippage and slacks are not accounted for the solution of the problem at this stage.

2.1 Discrete Time Control

Firstly, no digital computer closed loop control exists that can update its output on a continuous basis since these systems are discrete in nature. Also no real system cannot read their sensors measurements, process and apply commands instantaneously. Therefore discrete time control usually happens sequentially over a course of time defined as control cycle. This means that a command applied at any given instant is active until the command of the following cycle is applied.

This limitation prevents some robot movements as is the case of moving in a straight line while rotating over its center, as this movement would require continuous variations of the wheels speed setup. Using constant setups during the control cycle, the robot will describe an arc instead of moving over a straight line. The solution is to make the initial and final positions of the arc described in each control cycle lie on the pretended straight line. This can be obtained applying an angle offset to the linear velocity vector. Besides this angular offset, the speed needs to be increased since the length of an arc is longer than the length of its corresponding chord.

This problem can be addressed mathematically using the circle geometry and the chord-arc relationship. The direction deviation is given by the angle between the chord and the tangent to the arc in the chord's initial position. This shall be half the integral of the angular velocity. The chord-arc relationship states that given a chord and its central angle, the length of the arc connecting the initial and final positions of the chord is given by multiplying the chord's length by,

$$\frac{ArcLength}{ChordLength} = \frac{2\alpha r}{2\sin(\alpha) r} = \frac{\alpha}{\sin(\alpha)}$$

Since the angle used for correcting the direction is an inscribed angle, the arc's central angle shall be the double, as shown by Fig 4.

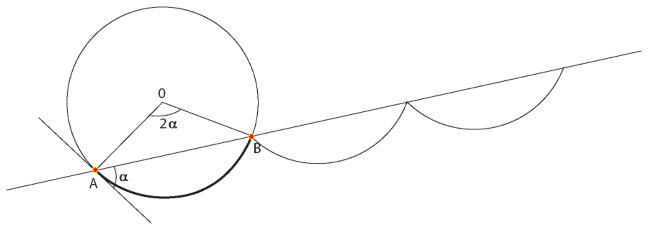


Fig. 4. While rotating and moving simultaneously the robot will describe a series of arcs along the straight line.

So, as mentioned before, $\alpha = \frac{(RotationalVelocity \times \Delta t)}{2}$.

In the robot, at every cycle, the length of the chord or straight line is calculated by integrating desired linear velocity to be applied over a cycle time. Then if the desired rotational velocity is not zero the length of the arc is calculated as mentioned above. The final velocity to be applied is obtained by deriving the arc's length, $v_{arc} = \frac{\alpha}{\sin(\alpha)} \times v_{line}$.

2.2 Acceleration

Other restrictions affecting omni-directional motion are the allowed values for the acceleration. In the RoboCup Small Size League some teams limit their robots acceleration by wheel traction only [7]. Given the Middle Size League robot's dimensions, this would result in a very dangerous behavior for the robot because if it is allowed to achieve, in a single cycle, a velocity with a different direction from the previous cycle there is a possibility of tipping off, severely damaging the robot's components. Given the electric nature of the used motors such unlimited acceleration can also cause large current peaks reducing the hardware lifetime and battery charge.

Therefore, the CAMBADA robots have an acceleration limiter at software level, which means that at each cycle the robot's velocity is allowed to varies only up to a certain amount around the previous cycle's velocity. The velocity then changes over time smoothly.

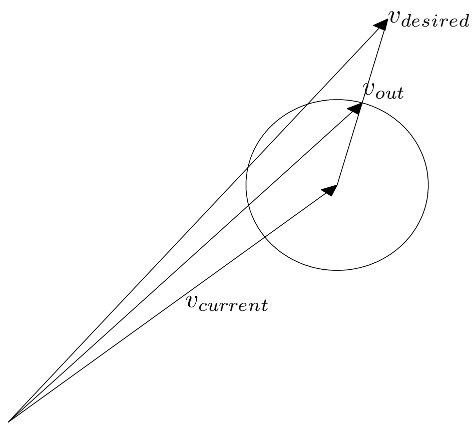


Fig. 5. Acceleration limiter. The circle represents all the velocities that can be achieved given the maximum acceleration.

2.3 Actuator Saturation

A factor adversely affecting an omni-directional robot control is the saturation of its different actuators. Of all the different actuators of CAMBADA robots, this paper will focus only on the motors controlling the omni-wheels. The saturation of the motors is an indirect indication of the maximum velocity a robot can achieve. In the CAMBADA case, simulation results point to a maximum speed of 2.2 m/s. This physical factor must not be overlooked. It affects not only the proportion between linear and angular velocities but also the proportion between the linear velocities components (v_x, v_y) diverting the robot from its path in case of actuator saturation.

Assuming a saturation value C, considering the 3 omni wheels, the velocities v_x, v_y and v_a, as the velocities along the robot's axis and angular velocity, respectively, the setpoints to apply at each wheel are calculated. Assuming that all setpoints exceed C, if no measure is taken the robot would rotate at maximum speed instead of moving according v_x, v_y and v_a. The solution involves determining a correction factor to be multiplied by all the velocities to be applied [8].

This correction factor is $\frac{C}{|S|}$, where S is the highest setpoint calculated.

This ensures that one of the setpoints is applied the maximum input possible and the others are in the original proportion before the saturation limitation. The resulting movement will be slower than desired but the direction of movement will be maintained while moving as fast as possible.

2.4 Delay

The final physical restraint mentioned in this paper is the system delay. In the RoboCup Middle Size League, some teams have already addressed this issue,as is the example of the Tribots team [9]. System delay results of the sums of all small delays in the control loop and greatly affects the robot control. A delay between the sensory input and the actuator output means a command calculated based on a worldstate A will be executed on a worldstate B that may or may not be the same as A. Considering the highly dynamic nature of a robotic soccer game, the worldstates will most likely not be the same.

The problem worsens as the delay increases not only because the error between the worldstates increases but also because as the delay overcomes the cycle time, more and more commands will affect the worldstate before the current command will start to have effects. This fact explains the hard deviation while moving and rotating simultaneously shown in Fig 3. The robot is stopped, and will try to move and rotate. Since the command calculated in the initial cycle will only reach the actuator some cycles later, all the control cycles in between them will calculate the same command since the robot will not move until the first command arrives. Ideally after the robot moved from the first command, a new one should be input considering the robot's new pose.

3 Predictive Control

Reflecting other teams solutions [7], a prediction module was implemented to determine the pose of a robot when the current command will reach the actuators, given the sensed pose and a buffer storing the commands that have already been issued but have yet executed due to the system delay. Thus, all commands are calculated without delay. The commands considered in the buffer of the prediction module do not reflect the desired velocities, but the actual input commands at the motors, after the saturation and acceleration limiters are taken in to account.

Considering the CAMBADA software structure [10], the desired commands are calculated on the agent process and transmitted to the low-level communication handler process, or HWcomm, through the RTDB[5], where they are processed through the saturation and acceleration limiters before being mapped to setpoints. In order to implement the prediction module in the agent process where the desired commands will be calculated accounting the predicted robot pose, the output commands of HWcomm must be sent back to the agent process, again through the RTDB. At the agent, as each command is active for an entire cycle and given the electric motors used, a purely uniform movement is assumed for each cycle. Then the predicted pose,$\hat{p}_{t_{pred}}$ for the robot is given by,

$$\hat{p}_{t_{pred}} = p_{t_{sensed}} + rem \cdot c_{t_{n-1}} + \sum_{i} c_{t_i} \varDelta t,\ i = sensed - n..sensed - 1$$

where

- $p_{t_{sensed}}$ is the self-localization determined pose,
- c_{t_i} is the command issued in the i-th control cycle,
- $n = \lfloor \frac{t_{delay}}{t_{cycle}} \rfloor$,
- $rem = delay - n \cdot \varDelta t$

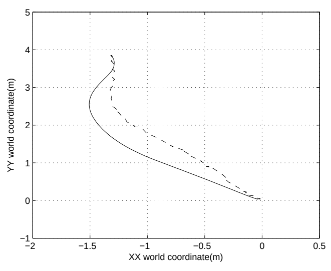
3.1 New Acceleration Limiter

Surprisingly the resulting path using predictive control was not the expected as the robot still doesn't move in a straight line. This is an undesired result from the current implementation of the acceleration limiter. As can be seen in Fig 5, if the desired velocity diverges too much from the current velocity, the output of the limiter will be a velocity that will neither be in the desired orientation nor in the desired norm. A solution to this problem is to increase the maximum acceleration of the robot, increasing the velocity variation every control cycle. However, simulation results showed that to compensate the large divergences between the velocities, the acceleration would have to be increased to values where the robot's motion stability would be compromised. These conditions when tested revealed another issue. While such high maximum acceleration allows for great velocity variations from cycle to cycle, the same high variations produce a considerable wheel slippage. Since this factor is not accounted for in the model, the resulting prediction will be very inaccurate. Therefore, the robot will not generate the anticipated path, as shown by Fig 6.

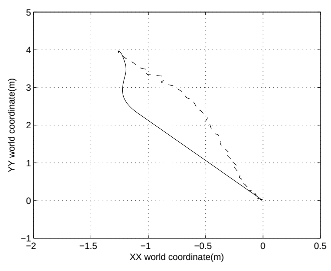
In this section, a new algorithm to update the velocity while complying with a maximum acceleration is suggested which prioritizes the velocity's direction over its norm.

The solution involves reducing the robot speed, in order to successfully rotate, similarly to what we humans do when driving. In order to reduce the speed by a minimum necessary the following situations are considered:

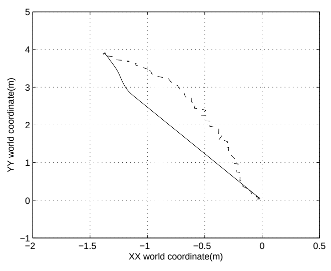
[5] Real-Time Data Base



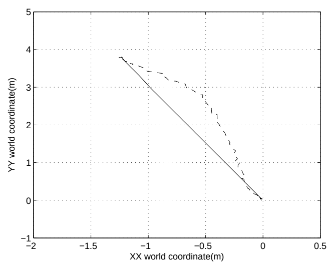
a)



b)



c)



d)

Fig. 6. Robot path with different values of maximum acceleration. The dashed line represents the real robot path while the solid line represents the simulated robot path. **a)** Maximum acceleration $3m/s^2$. **b)** Maximum acceleration $5m/s^2$. **c)** Maximum acceleration $7m/s^2$. **d)** Maximum acceleration $10m/s^2$.

Firstly, the closest point from the current velocity to the desired velocity vector is calculated. Uniting this point to the current velocity always creates a perpendicular line to the desired velocity. So the distance,d , between the current velocity and the closest point will be,

$d = sin(\Theta) \times |currentVelocity|$,

where Θ is the angle difference between the desired and current velocities.

Then if the distance is larger than the maximum velocity variation allowed, it means the robot doesn't have enough acceleration to reach the desired velocity direction. So to minimize the direction error, the output velocity of the acceleration limiter will be in the direction of the closest point.

If the distance is smaller, that means the robot can achieve the desired direction of the velocity even if not at the same norm. So the interception point between a circumference centered at the current velocity and with a radius of maximum velocity variation and the desired velocity vector is calculated. This point represents the maximum velocity the robot can achieve given the maximum acceleration limit while moving at the desired direction.

An exception must be made when the angle difference between the desired velocity and the current velocity is greater than 90 degrees. The resulting velocity using the perpendicular method would point in the opposite direction of the

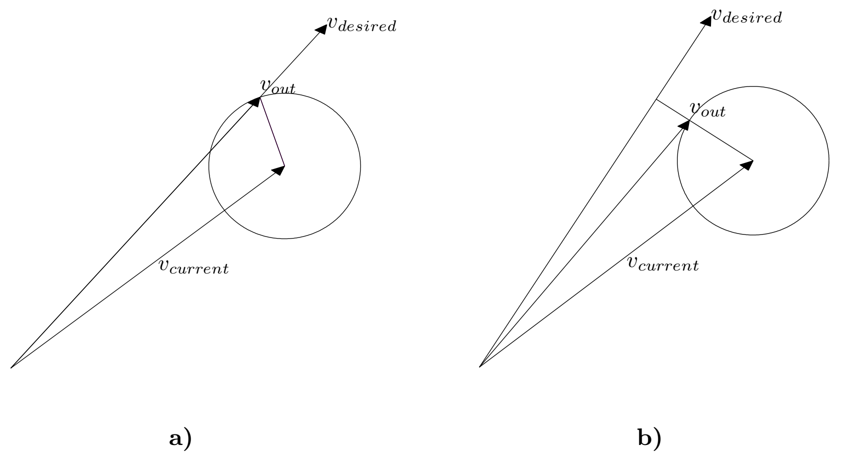


Fig. 7. New implementation of the acceleration limiter. Left **a)**: Priority is given to the direction of the desired velocity over its norm. **b)**: When the desired velocity can't be achieved given the maximum acceleration the output will be in the perpendicular line to the desired velocity that passes in the current velocity point.

desired velocity. In this case the original implementation of the acceleration limiter is used.

4 Estimating Model Parameters

The next step is to determine the control delay. For this purpose some captures were made while the robot moved. In these captures, the robot pose determined by self-localization and the commands sent to the actuators were recorded. Since the capture produces results every cycle, the delay is determined by the difference between the cycle corresponding to the first command sent and the cycle corresponding to the robot movement. This way, it means that when the robot pose changed the first command arrived at the motors. However, since the framerate of the camera used is 25 fps, it means the delay will be in an interval of $\frac{1}{25} = 0.040$ seconds. Using this method the delay affecting the CAMBADA robots would be in $[0.160; 0.200[$ seconds. Experiments were conducted to determine the delay with more precision, which tried to minimize the error between the simulated robot path and the robot real path. However at this stage none have proven trustworthy. So, as last resort, different delay values, between 0.160 and 0.200 seconds, were tested while the robot moved between the same two points. This method showed that when delay tends to 0.200 the robot path deviates further from the straight line. The chosen value for delay was 0.165 seconds, which provided the best resulting paths.

5 Results

Using the prediction module combined with the new acceleration limiter implementation the robot successfully moved in a straight line while rotating simultaneously as shown in Fig 8.

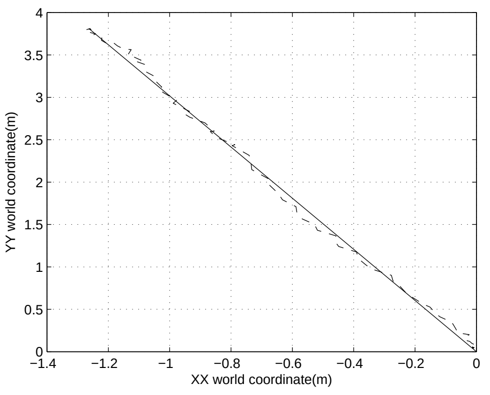


Fig. 8. The robot path using new acceleration limiter and robot pose prediction after delay. The delay value is 0.165 seconds. The robot is rotating counter-clockwise towards (0,0)

An experience was then conducted to compare the developed solution and the original solution, where the robot rotates first and moves afterwards. The experience consisted on moving the robot between two points using the previously mentioned solutions. The process was repeated ten times and the times were recorded for every run.

With the original solution, the robot took an average 3.62 seconds to move from one point to the other with a standard deviation of 0.11 seconds. With the developed solution, the robot took an average 3.28 seconds to move between the same two points with a standard deviation of 0.16 seconds.

This results in a mean time difference of 0.34 seconds. Although the improvement is not considerably large, given the fast pace of a soccer game, the increased speed might prove crucial in some game situations.

Finally an experience was conducted where the robot moved with a constant angular velocity. Even in this adverse condition the robot performance was acceptable, even despite not moving in a straight line and producing an overshoot.

6 Conclusion

An overview of different physical constraints affecting CAMBADA robots motion were presented with special focus on sensor-action delay. The various solutions

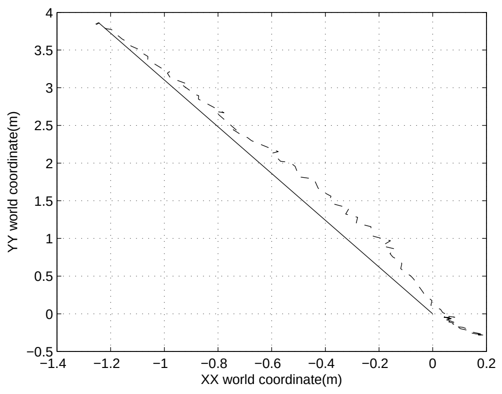


Fig. 9. The resulting robot path with constant angular velocity. The robot is rotating counter-clockwise towards (0,0).

developed to minimize the constraints' effects were described. A predictor module was implemented to address the delay affecting the robot along with an algorithm to update the robot velocity prioritizing the velocity direction. This initial implementation of predictive control improved robot motion and speed.

In the future the predictive control should be integrated with the existing behaviours of CAMBADA robots, enhancing ball handling, obstacle avoidance and team formation. For consistency a predictive model of the ball should also be implemented to predict its position. Since the teammate's position and velocity is shared between all robots the predictive model can be expanded predicting the teammates' positions. On the other hand since robot communication is not synchronized with the control cycle the resulting prediction might be very inaccurate. Also predicting the opponents position is impossible since at this date no opponent tracking is made being treated as stationary for obstacle avoidance purposes only.

Acknowledgments

This work was partially supported by project ACORD Adaptive Coordination of Robotic Teams, FCT/PTDC/EIA/70695/2006.

References

1. MSL Technical Committee 1997-2009: Middle Size Robot League Rules and Regulations for 2009 (2008).
2. Campion, G., Bastin, G., D'Andréa-Novel, B.: Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots, In: IEEE Transactions on Robotics and Automation, Vol. 12, No. 1, February 1996.

3. Oubbati, M., Schanz, M., Buchheim, T., Levi,P.: Velocity Control of an Omnidirectional RoboCup Player with Recurrent Neural Networks, A. Bredenfeld, et al., Eds., RoboCup 2005: Robot Soccer World Cup IX, LNCS, vol. 4020, 2006, 691-701.
4. Hafner, R., Lange S., Lauer, M., Riedmiller, M.: Brainstormers Tribots Team Description, RoboCup International Symposium 2008, CD Proc., Suzhou, China.
5. Sato, Y., Yamaguchi, S., et al.: Hibikino-Musashi Team Description Paper, RoboCup Int. Symposium 2008, CD Proc., Suzhou, China.
6. EtherCAT Robots win German Open, Press Release, EtherCAT Technology Group, 8 May 2008.
7. Behnke, S., Egorova, A., Gloye, A., Rojas, R., Simon, M.: Predicting away Robot Control Latency, Proc. of 7th RoboCup Int. Symposium, Padua, Italy, 2003
8. Plöger, P.-G.; Indiveri, G.; Paulus, J.: Motion Control of Swedish Wheeled Mobile Robots in the Presence of Actuator Saturation, RoboCup 2006 Symposium. Proceedings : Bremen, Germany, 19th and 20th of June 2006, Bremen, 2006, RoboCup International Symposium.
9. Lauer, M: Ego-Motion Estimation and Collision Detection for Omnidirectional Robots, In RoboCup 2006: Robot Soccer World Cup X, LNCS. Springer, 2006
10. A. J. R. Neves, J. L. Azevedo, M. B. Cunha, N. Lau, G. Corrente, F. Santos, A. Pereira, L. Almeida, L. S. Lopes, P. Pedreiras, J. Vieira, D. Martins, N. Figueiredo, J. Silva, N. Filipe, I. Pinheiro: CAMBADA'2009 Team Description.