Learning the Odometry on a Small Humanoid Robot

Quentin Rouxel, Grégoire Passault, Ludovic Hofer, Steve N'Guyen, Olivier Ly

***Abstract*— Odometry is an important element for the localization of mobile robots. For humanoid robots, it is very prone to integration errors, due to mechanical complexity, uncertainties and foot/ground contacts. Most of the time, a visual odometry is then used to encompass these problems. In this work we propose a method to compensate for odometry drifting using machine learning on a small size low-cost humanoid without vision. This method is tested on different ground conditions and exhibits a significant improvement in odometry accuracy.**

I. Introduction

Walking on uneven terrains is still an interesting challenge for humanoid robotics.

Our work is located in the context of the RoboCup competition, aiming at producing robots with human-like soccer level. Robots of humanoid leagues now have to walk on artificial grass. Along with the difficulties in walking and stabilizing on this kind of "soft" ground, artificial grass also causes more foot/ground slip and posture errors leading to more odometry errors.

The well known question of odometry – also named dead reckoning estimation – has been extensively studied on wheeled mobile robots. Kinematic models are built to summarize all known physical imperfections leading to systematic displacement errors. Open parameters are then calibrated with various geometrical [1] or statistical [2] approaches.

This process is far more difficult on humanoid robots – and in particular for low-cost humanoid robots – as in addition to foot slippery, ground contact constantly switches from single to double support foot phase and ground collisions are affecting the overall dynamics. Moreover, complex mechanical systems can never be perfect, especially in low-cost robots, making error modeling even more difficult.

This is why many recent methods are relaying on visual odometry estimated from robot's camera. In [3] a vision-based odometry is computed in real time using Simultaneous Localization and Mapping (SLAM) and Extended Kalman Filter (EKF) algorithms on a human sized robot. In [4] SLAM and an EKF are also used on a Nao robot, with the dead reckoning making use of Nao's pressure Force-Sensing Resistor (FSR) sensors to improve the support foot choice, which greatly influence the estimation accuracy.

An accurate a priori and online odometry is an essential basis for planning and robot navigation applications. For example, in [5], robot trajectories are planned (with classic A* and Markov Decision Process (MDP) framework) relaying on visual feature tracking to compensate for motion drift.

Visual odometry needs high computational power to analyze the camera information (which may be an obstacle for small robot embedded computers), is sensible to lighting and usually requires *controlled* environment conditions. Some works have tried other non visual sensors such as [6] which allows to correct odometry by measuring unwanted slippery with computer mice optical sensors placed on each foot. In [7] lasers where used for scanning the ground, helping SLAM localization in addition of foot contact sensors on an adult sized (non low-cost) robot.

Finally, quite a few works have proposed the use of regression techniques to improve odometry accuracy. On wheeled robots for instance, Angelova et al. [8] achieves to predict robot slippery from visual information on a planetary rover in a complex environment by using the machine learning Locally Weighted Projection Regression (LWPR) technique. On humanoid robots, Schmitz et al. [9] used a motion capture setup to learn the prediction of the next footstep in order to improve the control of a Central Pattern Generator (CPG) based walk engine. However, no odometry on the long run is presented and only an a priori (no sensor-based) footsteps prediction is learned.

In this work we propose a purely "internal" odometry system, i.e. without vision or laser sensor. Such a system is intended to be included as a component into a more global localization system, which in turn may include high level methods like vision or laser sensor.

We present a comprehensive comparison of an a priori and online odometry estimation using footsteps learning on both thin carpet and soft artificial grass. We then test the influence of both fully open-loop and simple closed-loop walk engine thanks to new low-cost foot pressure sensors mechanical integration. Using a kinematic model and fast prediction of learned regressions, operational real computation time is reached on the robot's embedded computer.

II. Humanoid Platform

A. Sigmaban Robot

Sigmaban [10] is a small and low-cost humanoid platform. The robot is 54cm height, weights 3.8Kg and has 20 degrees of freedom ; 6 per leg, 3 per arm and 2 for the head. The actuators are Dynamixel RX servo-motors with about 0.3° position encoder resolution. Finally, the main Central Processing Unit (CPU) is a small Intel Atom 1.6GHz processor.

A webcam is mounted on the head, providing visual feedback and an Inertia Measurement Unit (IMU) is giving accelerometers, gyroscopes and filtered pitch and roll

The authors are with the *Rhoban team*, LaBRI, University of Bordeaux, France. Emails: {quentin.rouxel, gregoire.passault, ludovic.hofer, steve.nguyen, olivier.ly}@labri.fr

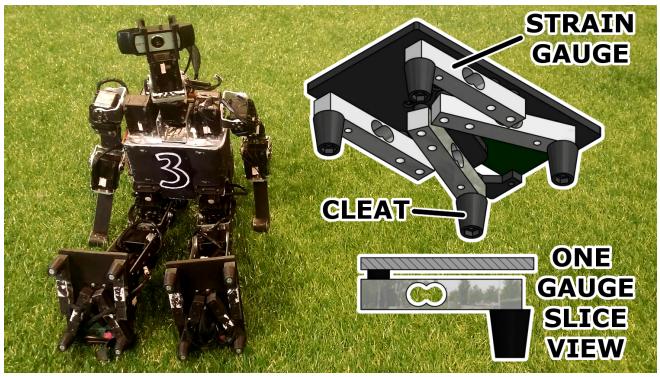

Fig. 1: On the left side, Sigmaban Kid-Size Humanoid Robot. On the right side, low-cost foot pressure sensors made of 4 strain gauges and measuring weight put on each cleat. This mechanical assembly is robust and well adapted to soft artificial grass, improving the stability of the robot

orientation. According to the RoboCup Humanoid league restrictions, the sensors of the robot are constrained to be "human possible". Therefore, we do not use 3D camera nor distance sensor.

Let us note that the small size of the robot makes experiments easier since the robot is more robust and can fall without breaking itself and without being dangerous for operators.

The robot is also equipped with foot pressure sensors[1] presented in [11] and detailed in [12] (see Fig. 1). Each foot has 4 cleats attached to small strain gauges which, once properly calibrated, allows us to measure foot contact forces on these 4 points. In our system the strain gauges are parts of the mechanical structure allowing to directly measure the foot deformations. This configuration alleviates the usual sensor/ground contact problems encountered when using other cheap sensors such as Force-Sensing Resistor (FSR). We can then compute the robot's weight on each foot along with an estimation of the center of pressure.

B. Walk Engine

The robot gait is controlled by the walk engine, mainly based on open-loop parametrized splines[2]. The walk engine controls the 12 servo-motors of the legs via inverse kinematics. Reference foot and trunk trajectories are defined in Cartesian (x, y, z) space using polynomial cubic splines. These splines are parametrized by a set of parameters such as the walk frequency, forward and lateral footstep length, footstep rotation angle, height of foot rise, height of the trunk from the ground, phase and amplitude of the trunk lateral oscillation, trunk pitch... Joints reference positions are then computed via inverse kinematics and sent to servo-motors. The walk scheme assumes no double support phase.

[1] Open source project available at `https://github.com/Rhoban/ForceFoot`

[2] Open source project available at `https://github.com/Rhoban/IKWalk`

The robot walking direction and velocity are controlled by adjusting the 3 dynamic parameters: footstep forward and lateral length, in addition to footstep rotation angle (gait commands). Mixing these inputs determines the foot position for the next step and therefore allows for an omni-directional motion. Note that these parameters are only updated on foot support exchange. All other parameters remain fixed during the walk sequences. They have been manually tuned by experiment to optimize robot's stability and speed on artificial grass.

On top of the open-loop motion, we added a simple closed-loop strategy in order to improve walk stability. A major cause of destabilization is the small desynchronizations between robot's dynamics and the open-loop timing. By using the foot pressure sensors to measure the weight distribution on each foot, the walk engine pauses the open-loop cycle when the foot about to rise has still a significant amount of weight on it. Doing so, it enforces a time-resynchronization between the model and the reality.

This walking scheme was used by the Sigmaban robots of the *Rhoban Football Club* team at Humanoid KidSize RoboCup 2015 competition in China.

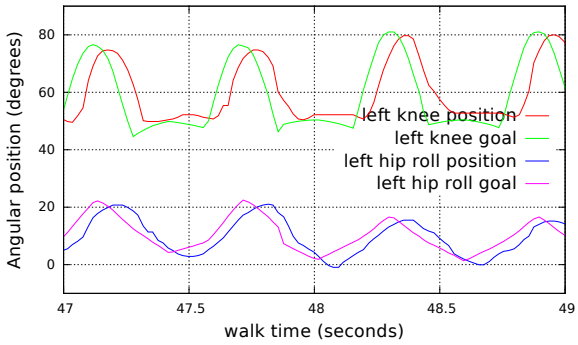

Fig. 2: Servo-motors control errors. Comparing left knee and left hip pitch reference positions to the read positions from encoders. In addition to the time lag, a max error of 3 to 5 degrees can be observed.

C. Robot Model

From the assembly of a Computer Assisted Design software, a complete dynamic model[3] has been created allowing for accurate geometric calculations. The model is represented as two kinematic trees whose roots are located on each foot of the robot. The used kinematic tree is imposed by the current support foot. Each time the current support foot is switched, the model is updated and the old flying foot becomes the new support foot. The odometry is then estimated by integrating the robot's head[4] position over time.

The model can then be updated in two ways. Without sensor, all the degrees of freedom goal positions are assigned

[3] The standard Unified Robot Description Format (URDF) export format is used

[4] The head is chosen to be the reference frame of the robot position because the motion capture reflective markers are placed on the robot's head

to servo-motors reference positions generated by the walk engine. The support foot is switched when the height of the flying foot about to land, goes below the floor level. In this manner, the robot's odometry can be simulated given the walk engine inputs.

When using sensors, all the degrees of freedom positions are set to the measured joint encoders. Then, the support foot is determined by the pressure sensors to be the foot measuring the more weight. The robot's yaw orientation is computed by integrating the raw yaw gyroscope which has proven to be more accurate than the integration of the kinematic model rotations. In addition to the mechanical backlash, when walking on soft surface such as artificial grass, the foot may not lie flat. Filtered measured pitch and roll from the IMU placed on the trunk are assumed to be correct. A rotation is then applied between the support foot and the ground in order for the trunk to match the measured orientation.

III. Odometry Learning

The absolute Cartesian position of the robot over time can be estimated in two major ways, *a priori* or *online* i.e. by taking into account only gait commands, equivalent to motor reference positions, or by using robot's sensors measured from the environment. In the first case, no actual physical action is needed, allowing to simulate future estimated positions given future gait commands. In the second case, sensors have to be recorded all along the traveled trajectory. The latter is expected to be obviously more accurate but can not be used for planning purposes.

The kinematic model of the robot is fed either with the motor goal positions or the complete measured sensors. In this way, it is able to compute the a priori or the online estimation of the robot's head odometry. In the following, a machine learning approach is proposed in order to improve the accuracy of these basic methods. During experiments, the ground truth absolute robot position is provided by a motion capture system.

A. Footstep Displacements

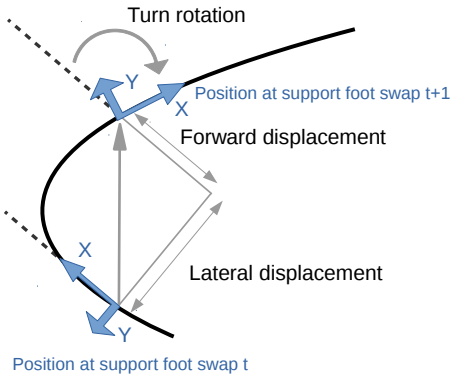

Fig. 3: Footstep displacement calculations on support foot swap

Instead of trying to predict the instantaneous velocity of the robot trajectory which is prone to high noise, we compute the difference of the robot's head pose between two consecutive support foot swaps. In addition, since the walk engine commands (forward and lateral step lengths, foot rotation) are only updated on support swap, same footsteps are expected to result in repeatable displacements.

During one walk cycle, two support foot swaps occur. Each time, the forward and lateral displacement, the angle rotation from the last support swap are computed in the robot's own reference frame as depicted on Fig. 3. Footstep displacements are calculated for both the left and right feet from the ground truth robot position, sensor-based odometry and simulation-based odometry.

B. Data Set

During learning experiments, a new data point is generated at each support foot swap time t. Each recorded data point contains the ground truth footstep displacements $\Delta(x, y, \theta)_{t,\ truth}$, the expected displacement from gait commands $\Delta(x, y, \theta)_{t,\ simulation\ based}$ and the displacement computed by the model from robot's sensors $\Delta(x, y, \theta)_{t,\ sensor\ based}$.

C. Learning with LWPR

The next phase is to learn to predict the ground truth footsteps with the odometry footsteps as input and so to build a corrective function improving the estimation precision. Without knowing the structure of the function to learn, we decided to use the Locally Weighted Projection Regression (LWPR) algorithm.

LWPR is a state of the art non parametric and non linear regression method (supervised learning) allowing for a fast prediction and incremental learning ([13]).

The input space is incrementally divided in several *receptive fields* in which a Partial Least Square (PLS) algorithm locally reduces the input space dimension and fits a linear model. Predictions are computed by averaging all local models weighted by the distance from each receptive field to the query point. Another state of the art non parametric family of regression algorithms are based on Gaussian Processes (GP). Even if the overall accuracy and convergence speed of GP are better ([14] compares LWPR with classic GP and with a proposed hybrid method), LWPR is the fastest technique for learning and prediction runtime.

LWPR has been chosen for its computational performance with the need of real time prediction on the small embedded robot's computer.

For both learning cases, 3 LWPR models are used to predict at support foot swap time t, the ground truth forward $\Delta x_{t,\ truth}$, lateral $\Delta y_{t,\ truth}$ and angle rotation $\Delta\theta_{t,\ truth}$ footstep displacements. This is a simplification ; the three ground truths $\Delta x_{t,\ truth}, \Delta y_{t,\ truth}, \Delta\theta_{t,\ truth}$ are assumed to be independent given the inputs.

The simulation-based odometry learning only uses gait commands as input. For each three ground truth displacement predictions, an independent function $\mathbb{R}^7 \longrightarrow \mathbb{R}^1$ is learned:

$$\begin{cases} \Delta(x,y,\theta)_{t,\ simulation\ based} \\ \Delta(x,y,\theta)_{t-1,\ simulation\ based} \\ support\_foot\ _{t,\ simulation\ based} \end{cases} \longmapsto \Delta x_{truth}$$

The sensor-based odometry, on the contrary, uses robot's sensors. The three learned functions $\mathbb{R}^9 \longrightarrow \mathbb{R}^1$ have the following inputs:

$$\begin{cases} \Delta(x,y,\theta)_{t,\ sensor\ based} \\ \Delta(x,y,\theta)_{t-1,\ sensor\ based} \\ support\_foot\ _{t,\ sensor\ based} \\ step\_duration\ _{t,\ sensor\ based} \\ step\_duration\ _{t-1,\ sensor\ based} \end{cases} \longmapsto \Delta x_{truth}$$

To take into account dynamical effects such as robot's accelerations, regressions are given an history of the last two footstep displacements as input as well as a binary variable indicating the current support foot. Since actual measured footsteps are not really periodic (especially on closed-loop walk engine), sensor-based odometry regressions are also given the step durations as input.

D. Overall Architecture

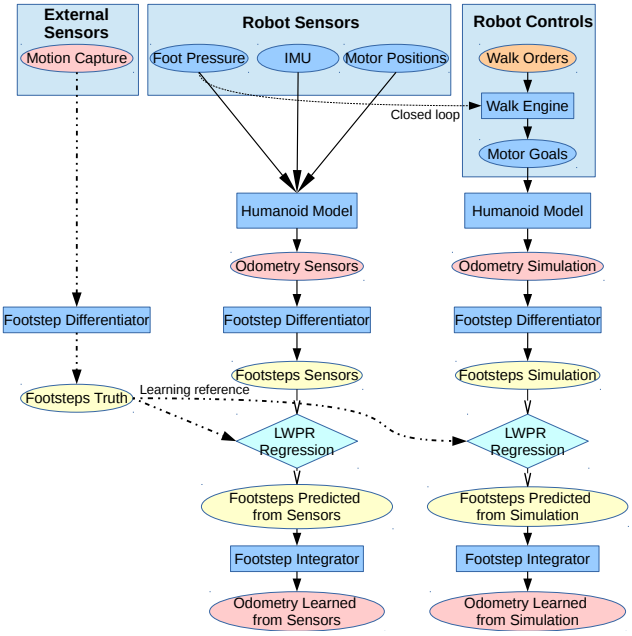

Fig. 4: System architecture. Ellipses represent values and rectangles represent abstract functions. All odometry estimations (red) are compared with ground truth motion capture.

Finally, Fig. 4 shows the global system architecture. Data are represented flowing from robot's internal sensors, external sensors and gait controls to the several odometry estimations. The walk engine is taking pressure sensors as input in the closed-loop version. After being predicted from the sensor-based and simulation-based odometry by LWPR regressions, footsteps are re-integrated (inverse operation than illustrated in Fig. 3) to get the corrected odometry estimate.

IV. Results

A short video can be found along with this paper and briefly presents the experimental setup and its results[5].

A. Experimental Setup

Fig. 5: The motion capture area on artificial grass (30mm thick) field and Sigmaban humanoid robot with reflective markers on the head

These experiments will compare results obtained on a regular smooth ground (a thin carpet) and a soft irregular ground (artificial grass of about 30mm thick)[6].

Each experiment run is composed of 4 or 5 walk sequences of about 5 minutes each. The walk engine is manually driven by a joystick controlling forward, lateral step length and rotation. The robot position is constrained to remain within the capture area while exploring the walk input space, mixing forward, lateral and turn gait commands. Walk stability is manually ensured by keeping gait commands and accelerations in acceptable range preventing the robot from falling.

During the experiments, the pose (position and orientation) of the robot's head is tracked by an external motion capture system[7] providing the absolute Cartesian pose in world frame at a frequency of 100Hz (see Fig. 5). All motor encoders, pressure sensors, gyroscopes and accelerometers are recorded at 50Hz.

B. Validation and Results

1) Computing Performances: Since no vision is used for the odometry calculations, good computation time

[5] The video is also available at: `https://youtu.be/9HT33KMtfLw`

[6] This is compliant to the RoboCup Humanoid KidSize League new rules introduced in 2015 (see `https://www.robocuphumanoid.org/materials/rules/`). This new constraint, from thin carpet to a soft ground, is a step toward Robotics soccer games against humans but also makes the biped walk much harder to stabilize.

[7] Six OptiTrack infrared cameras are used

Main control loop (50Hz) : 10.5ms
- Walk Engine: 0.36ms
- Logging Data: 1.6ms
- Odometry evaluation: 8.3ms
  - Model Geometric Computation: 2.8ms
  - LWPR prediction: 5.2ms

Fig. 6: Average computing time on robot's embedded CPU

can be reached on the embedded computer of the robot. LWPR regression allows for fast incremental learning and prediction time on the small Atom 1.6GHz robot's processor. Average computing time are depicted in Fig. 6. The whole code is implemented in C++. LWPR authors' implementation[8] is used for the regressions and the Rigid Body Dynamic Library (RBDL)[9] is used for all geometric and kinematic model computations.

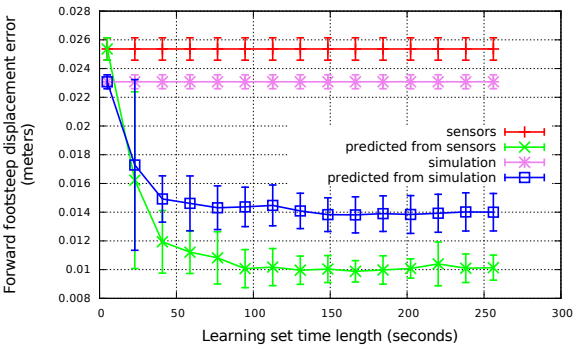

Fig. 7: Average distance between estimated footstep displacement in the robot's forward direction (x) and measured ground truth. Comparison of footstep displacements computed from motors goals (violet) and sensors (red) versus regression predictions (green and blue) under different learning time durations. Error bars give 95% confidence intervals evaluated from 5 logs.

2) Learning Convergence: These first results take place on artificial grass with a fully open-loop walk engine. 5 walk sequences from 345 seconds to 471 seconds long are recorded using the same motion capture setup. The first data log is divided in two sequences. The first half is used for LWPR regression as a learning set and the second half is used as a test set for tuning algorithm's meta parameters. Learned regressions are then applied to the 4 other logs.

Fig. 7 shows the convergence of prediction errors of LWPR regression compared with non-learning odometry approaches. For each data log, a learning sequence is extracted and regressions are trained. Then, footstep displacements are predicted on the remaining part of the log and the mean error between predicted footstep time series and recorded ground truth is computed.

Approximately two minutes of walk exploration of the footsteps space is needed to converge. Here, 120 seconds of learning sequence represents about 400 data points, since the walk frequency is 1.7Hz and that one walk cycle is composed of two steps. LWPR inspection reveals that the number of receptive fields used by the regressions is small, ranging only from 1 to 5 for the most complex function (rotation). This indicates that the learned function must be quite simple and almost linear ; explaining the little learning time.

The comparison between footstep displacements predicted from sensor-based and predicted from simulation-based odometry on Fig. 7 validates that measured displacements are better fitted by the regression using robot sensors as inputs than only gait commands simulation.

Note that a small footstep displacements error does not necessarily imply a good odometry tracking since a large error on left footstep could be compensated with a large opposite error on right footstep during displacement integration.

3) Comparison on Mid Term Trajectories: Once the footstep displacements are computed, the test logs are divided into non overlapping sequences of 20 seconds and the resulting odometry is reset and integrated over each small sequence. Some simple typical robot trajectories comparing the four methods are depicted in Fig. 8.

The odometry estimation from gait commands simulation is highly underestimating distances. Walk engine commands are directly linked to motor orders through inverse kinematics, but the purely feedback control algorithm of RX Dynamixel servo-motors (proportional controller) is unable to follow the desired joint trajectory under a such dynamic movement (see Fig. 2). Due to leg inertia, actual robot's steps are thus longer than expected. Note that our open-loop walk engine is tuned from its origin to deal with this behavior.

In a lesser way, the sensor-based odometry is also underestimating distances even when reading the motor position encoders. This is the result of some non measurable mechanical backlash increasing the real amplitude of the steps.

The relatively small length of the walk sequences (20s) used here is chosen in order to avoid statistical flukes on the odometry. This is explained by the small motion capture area (roughly a rectangle of 1.5 by 2 meters). Since the model odometry (without learning) underestimates distances but keeps a quite good orientation, when the robot goes back and forth (exploring footstep space), it often catches up the ground truth position by pure chance, which statically overestimates the odometry quality.

Finally, statistical results on Fig. 9 compare the overall odometry error of the four methods still using an open-loop walk engine on the artificial grass surface. In both sensor-based and simulation-based approaches, the learning technique significantly improves the odometry precision.

[8] http://wcms.inf.ed.ac.uk/ipab/slmc/research/software-lwpr

[9] https://bitbucket.org/rbdl/rbdl

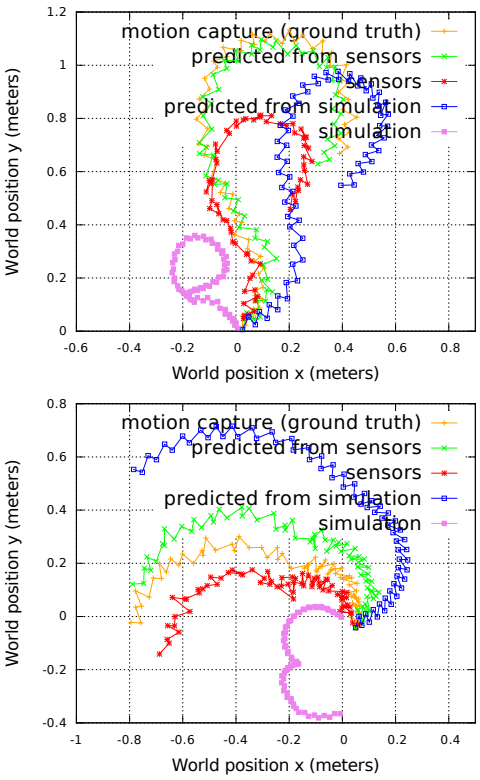

Fig. 8: The four odometry methods compared with motion capture measured ground truth on two typical walk sequences. The integrated robot's head trajectory is shown over a walk time length of 20 seconds starting from the origin $(0, 0)$ position. Forward steps, lateral steps and turn walk inputs are mixed. Due to control errors, a stationary or slightly negative forward order may results in a forward motion. On both trajectory beginnings, actual stationary turn is performed, misleading the simulation-based odometry which integrates backward displacements.

Since angular error computed from the model odometry is based on raw gyroscope integration, which is already fairly good with low drift on such time scale, learning technique only makes little gain.

Note that an important feature of simulation-based odometry is that it can be fully computed a priori, without any sensor feedback. Which means that before actually moving, the robot is able to predict its resulting position given a sequence of future walk engine inputs. Given the fact that due to the low-cost servo-motors control issues, raw kinematic footstep commands cannot be trusted. Then, the learning approach greatly enhance the prediction quality. This is an advantage for medium-term navigation planning (about 20s).

4) Ground Surfaces and Open/Closed Loop Comparison: Lastly, we compared the odometry quality on 4 different setups: artificial grass and carpet with open-loop and closed-loop walk engines.

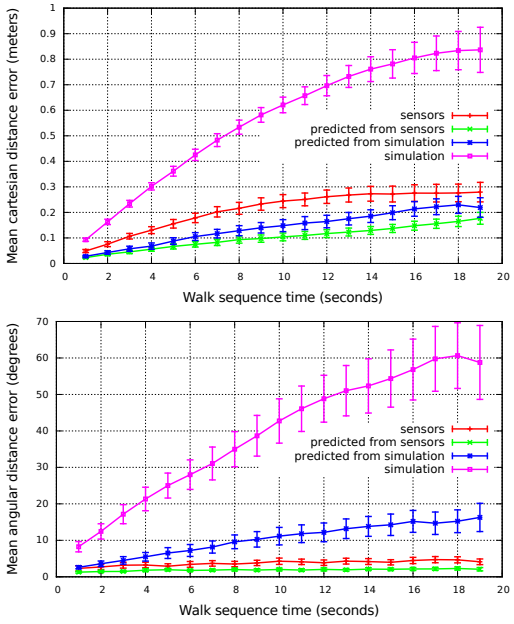

Fig. 9: Average distance of Cartesian and angular odometry from ground truth. Comparison of the four odometry methods statistics from 58 walk sequences of 20 seconds. Error bars give 95% confidence intervals on average.

Since the new RoboCup Humanoid rules (2015) introduced the artificial grass for soccer field, we compared this surface with the previous one, a thin carpet. The soft grass surface mainly tends to make the walk less stable and requires that the robot rises the flying foot higher due to grass frictions. The walk stability is thus ensured by decreasing the walk speed and switching from flat feet to adding 4 cleats integrated with pressure sensors.

Notice that the used walk engine parameters are the same in both conditions but the walk has been tuned for artificial grass. In addition, foot's cleats are clearly more adapted to the grass than to the carpet.

Odometry quality comparison on these setups is shown on Fig. 10. For each setup, 4 or 5 walk logs of about 5 minutes long are recorded. Learning takes place on the first log and statistics on small sequences of 20 seconds are then computed on other test logs. Several comments can be made:

- To begin, both simulation-based and sensor-based odometry errors are reduced by using a simple closed-loop walk engine over a fully open-loop one. One possible interpretation is that a more reactive control leads to a more stable gait, reducing the displacement noise affecting the robot trajectory when the system gets temporarily unstable.
- Then, switching from a carpet surface to a soft artificial grass increases simulation-based odometry errors but does not affect sensor-based odometry. This is a clue that due to grass friction on flying foot, the actual footstep displacements may be altered. Thus, not using the environment feedback increases the odometry errors.

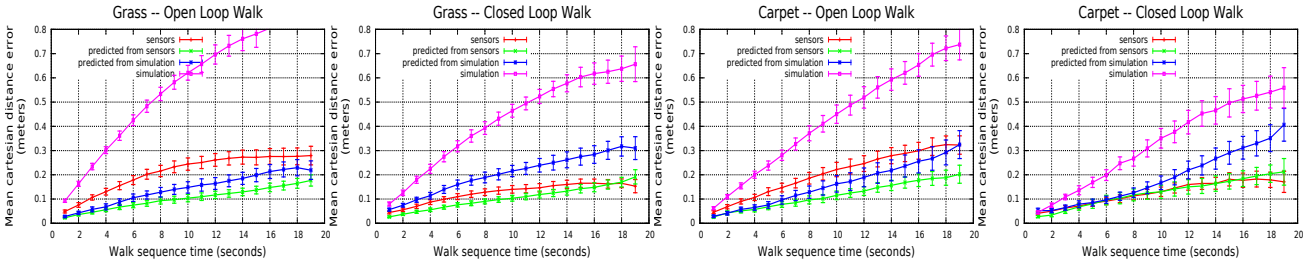

Fig. 10: Comparison of odometry error using closed-loop and open-loop walk engines and artificial grass and carpet surfaces. Each comparison is made of 45 to 72 walk sequences of 20 seconds.

On contrary, sensor-based odometry, taking into account motor encoders and sensors is not significantly affected by the grass.

- Finally, the errors of odometry predicted without sensors are higher using a closed-loop walk engine than using open-loop walk. By using a closed-loop walk, the actual motor orders and hence the footsteps are dependent of robot's sensors. The prediction of real footstep displacements from only gait commands is obviously lacking information to accurately fit the walk behavior. Using a closed-loop walk engine, the robot is more stable and non-learning approaches are more accurate than with open-loop but simulation-based odometry learning gets less precise as the robot movement is now depending on its sensors.

V. Conclusions and Future Works

In this paper we have demonstrated the possibility to use a machine learning technique to significantly enhance the internal odometry estimation on both simulation and online conditions of a small humanoid robot. On artificial grass and using an open-loop walk engine, the average online sensor-based odometry drift has been improved from 24.5cm to 10.3cm and the average off-line simulation-based odometry drift has been reduced from 62.0cm to 14.8cm after 10 seconds of walking.

This method has been compared in different setups, on a thin carpet and on soft artificial grass, with and without closed-loop correction. As expected, results indicate that the grass surface increases the average odometry error compared with the carpet, since the robot is destabilized by the soft ground and frictions. Moreover, the closed-loop walk engine tends to reduce the odometry error of non-learning methods (simulation and online using sensors) compared to the simple open-loop walk.

This work aims at improving the classic internal odometry component usually used in combination with external sensors such as vision or laser rangefinder in order to compute the localization of the robot. We thus expect an improvement in our localization system based on the integration by a particle filter of the odometry with monocular vision.

Since the drift of the simulation-based method has been significantly reduced by learning, the natural extension of this work is to use the improved prediction accuracy to develop enhanced mid-term step planning and navigation techniques.

References

[1] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 6, pp. 869–880, 1996.

[2] A. Kelly, "Fast and easy systematic and stochastic odometry calibration," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2004, pp. 3188–3194.

[3] S. Ahn, S. Yoon, S. Hyung, N. Kwak, and K. S. Roh, "On-board odometry estimation for 3d vision-based slam of humanoid robot," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4006–4012.

[4] G. Oriolo, A. Paolillo, L. Rosa, and M. Vendittelli, "Vision-based odometric localization for humanoids using a kinematic ekf," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*. IEEE, 2012, pp. 153–158.

[5] S. Osswald, A. Hornung, and M. Bennewitz, "Learning reliable and efficient navigation with a humanoid," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2375–2380.

[6] S. Czarnetzki, M. Hegele, and S. Kerner, "Odometry correction for humanoid robots using optical sensors," in *RoboCup 2010: Robot Soccer World Cup XIV*. Springer, 2011, pp. 48–59.

[7] J. Chestnutt, Y. Takaoka, K. Suga, K. Nishiwaki, J. Kuffner, and S. Kagami, "Biped navigation in rough environments using on-board sensing," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 3543–3548.

[8] A. Angelova, L. Matthies, D. Helmick, G. Sibley, and P. Perona, "Learning to predict slip for ground robots," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 3324–3331.

[9] A. Schmitz, M. Missura, and S. Behnke, "Learning footstep prediction from motion capture," in *RoboCup 2010: Robot Soccer World Cup XIV*. Springer, 2011, pp. 97–108.

[10] R. Fabre, H. Gimbert, L. Gondry, L. Hofer, O. Ly, S. N'Guyen, G. Passault, and Q. Rouxel, "Rhoban football club team – description paper," in *Humanoid KidSize League, Robocup 2015 Hefei*, 2015.

[11] G. Passault, Q. Rouxel, L. Hofer, S. N'Guyen, and O. Ly, "Low-cost force sensors for small size humanoid robot," in *2015 IEEE-RAS International Conference on Humanoid Robots (Video contribution)*, accepted.

[12] Q. Rouxel, G. Passault, L. Hofer, S. N'Guyen, and O. Ly, "Rhoban hardware and software open source contributions for robocup humanoids," in *Proceedings of 10th Workshop on Humanoid Soccer Robots, IEEE-RAS Int. Conference on Humanoid Robots, Seoul, Korea*, 2015.

[13] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[14] D. Nguyen-Tuong and J. Peters, "Local gaussian process regression for real-time model-based robot control," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 380–385.