Autonomous Humanoid Navigation Using Laser and Odometry Data

Ricardo Tellez, Francesco Ferro, Dario Mora, Daniel Pinyol and Davide Faconti

***Abstract*— In this paper we present a novel approach to legged humanoid navigation on indoor environments using classical probabilistic SLAM methods based on odometry information and laser measurements. We use two small lasers installed in the robot feet to capture the laser data. Odometry is obtained by calculating the position of each feet at every time step. The SLAM problem is solved by using a multi-laser SLAM solution together with a holonomic motion model. Navigation skills also include a path planning module with obstacle avoidance for autonomous navigation in indoor environments, and the whole process is performed without external computational power. Optionally, localization robustness is increased by adding the detection of landmarks using a camera. Results obtained are presented for the 1.5 m tall Reem-B humanoid robot.**

I. INTRODUCTION

Simultaneous Localization and Mapping in legged humanoid robots is a difficult issue, mainly due to the fact that traditional SLAM techniques based on laser and odometry data cannot be used. The main reason is that laser devices are very heavy and cannot be mounted onboard a humanoid. On top of that, in some cases where researchers have achieved to include lasers on a humanoid [1], [2], the large number of degrees of freedom makes very difficult to take stabilized measurements of laser-odometry.

For those reasons, SLAM implementations in humanoids are mainly based on the use of vision. Instead of using laser devices, cameras are used to identify landmarks, converting the SLAM problem into a vision one [3], [4], [5], [6]. For instance, the QRIO robot uses a stereo camera to self-localize and identify possible paths to goals [7]. The HRP robot achieved to use a small laser mounted on the robot's mouth [1], [2] in addition to the use of the camera to create complex 3D representations of the environment. However, due to its inability to correctly measure the laser position, its use was merely relegated to the identification of obstacles.

In this paper, a novel approach is proposed based only on laser and odometry data. Two small lasers are mounted on the feet of the robot and used to navigate. Using this setup, the robot is able to solve the whole problem of navigation, that is, generate maps in real time, localize on it, and even autonomously navigate using path planning and obstacle avoidance. Additionally, and as an optional feature, the robot makes use of its stereo vision camera to increase the localization robustness and allow quick recovery in lost situations. The paper is divided as follows: in section II, a description of the humanoid platform is provided including the hardware setup required to perform navigation. Section III describes the algorithm used to perform mapping. Section IV describes the localization system. In section V, there is a description of the path planning method used, together with the obstacle avoidance module. Each section includes the results obtained when applied to the real robot Reem-B.

Pal Technology Robotics
Paris 175, 4-1, Barcelona, Spain, tellezatwork@gmail.com

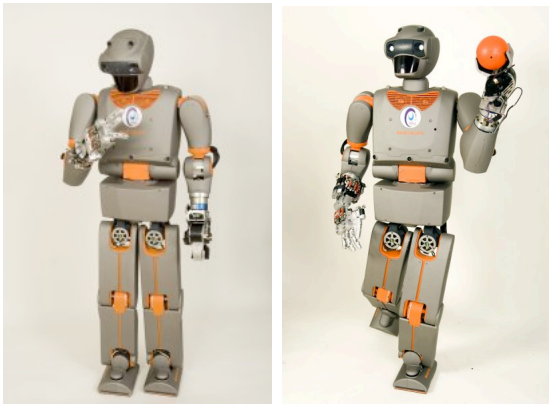

Fig. 1. Reem B robot used for the experiments

II. HUMANOID PLATFORM

The humanoid robot used is called Reem-B, a humanoid robot of 1.5 meters tall built by Pal Technology. Its weight is about 60 Kg with an autonomy of about 120 minutes. Among other abilities, Reem-B can carry loads as heavy as 12 Kg, and walk at a maximum speed of 1.5 Km/h with a great stability. The walking algorithm is based on a zero moment point approximation [8].

In order to feed the SLAM algorithm with sensor data, the robot is equipped with two small Hokuyo lasers [9] mounted on its feet (figure 2). Each laser is configured to scan 180 degrees on an angle that is rotated 30 degrees from the foot center. This rotation allows the detection of obstacles behind the robot, and avoids self-detection of the other leg. Angle resolution is of one degree, what makes a total of 360 measurements to be processed by the algorithm at every time step.

To provide a stabilized data measurement, the movement of the feet when stepping is maintained parallel to the ground by the walking algorithm. This fact, assures that the measurements made by a laser in movement corresponds to a unique inclination. Measurements taken in this way are surprisingly very stable, as can be observed in figure 3. This situation allows the use of the laser scanned data on a regular basis with independence of movement status of the foot. In order to use this information it is assumed that there are no obstacles with a height between the laser height when the

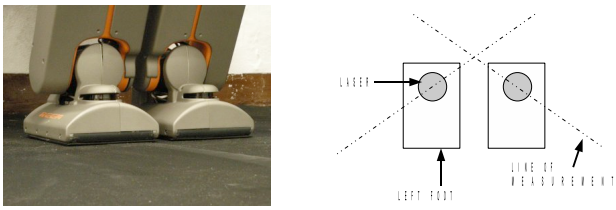

Fig. 2. Image of the two lasers mounted on the feet (left), and diagram of the scanned range of each laser seen from top (right)

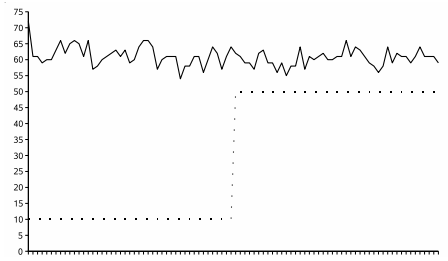

Fig. 3. A measurement made by the left laser foot at 180 degrees, when the robot looks parallel to a wall, at 6 cm aprox. At some point in time, the foot is moved up and forward (showed by the dotted line). The graphic shows that there is no notable difference between the measurement with the foot on the ground or with the foot on the air.

feet is on the ground and when the feet is at its maximum height (see section VI for a discussion on that).

Odometry is obtained with an inverse kinematic computation by the walking algorithm, providing at each step the (x,y,z) position and (θ, φ, ϕ) angles of each foot within a global coordinates axis, in a similar way as provided for a 2D mobile base. For our algorithm, only the (x,y) coordinates and the θ angle (as projected over the (x,y) axis) are used to indicate the orientation of each laser, in a similar way as in typical SLAM with wheeled robots. The z coordinate is not used in the algorithm, and the other two angles (φ, ϕ), which are the angles projected over the axis (x,z) and (y,z) respectively, are maintained constant during the whole walking movement.

The robot is equipped with two PC104 stack based computers, which provide all the computational power required to perform all the navigation and control tasks. This means that all the algorithms required to perform mapping, localization, path planning and obstacle avoidance are executed inside the robot. Only the visualization tools used to take pictures of the maps and observe localization status where run on external computers to the robot itself. This fact makes the Reem-B one of the most autonomous robots of the world of its size.

III. SLAM ALGORITHM

Localization and mapping abilities of the robot are based on the use of particle filters. For the implementation of the SLAM feature, we have selected an algorithm called DP-SLAM [10], [11]. This is a very interesting algorithm, since it allows an almost real-time generation of maps, without requiring posterior post-processing. This means that our robot creates its environment map at the same time that it walks around it, and, if correctly tuned, no further processing is required. Once the map is completed, the robot can directly change its operation mode to localization (see section IV), and start using the map for localization and path planning, without requiring any additional step.

A. DP-SLAM algorithm

The DP-SLAM algorithm uses a particle filter to maintain a joint probability over robot pose and maps. This implies that the algorithm is able to solve map ambiguities during the particle filter execution. Furthermore, it doesn't need any additional phase to produce the map and/or close loops. The difference with similar algorithms [12], [13], [14] is the technique it uses to maintain the large number of maps efficiently by exploiting redundancies between maps. The authors have developed a method called distributed particle mapping. Instead of explicitly maintaining multiple maps, the algorithm maintains a single map matrix that represents the grid map, and includes in each grid the observations made by different particles.

DP-SLAM maintains an ancestry-tree of particles, where current particles are the leaves. The parent of each node is the particle of the previous iteration before resampling. This ancestry tree is used to retrieve the exact lineage of a given particle at any time, and then reconstruct the best current map.

We have made a C++ adaptation of the algorithm described in [11], which is an improved version of a previous algorithm described in [10]. We implemented the algorithm with a modification to the particle filter for its use with the humanoid by using a multi-laser approach.

B. Multi-laser particle filter

The laser data required to localize the robot is generated by two laser devices. Each laser is localized on one of the robot feet, providing a 181 lecture data points. Even if the position of the two feet are directly related by the mechanical structure that ties them up, it is possible for each foot to have a different position and orientation. This means that the relative position of each laser respect to the other is not maintained over time, but it changes as the robot moves. Since the generation of a single robot map should accommodate the lectures of both lasers, a special method to combine them into the same probabilistic process has been adopted.

For the resolution of this problem, we take a multi-laser approach. We see the robot as a single holonomic wheeled robot, where particles represent the position of the left feet, as centered in the robot center. Right feet laser position is obtained by the combination of the current relative position to left-laser and a gaussian distribution of noise. By doing this, it is possible to estimate the posterior probability over one robot trajectory (the one of the left feet) and one single map.

Observations provided by each laser are introduced into the particle filter as if they where of a single robot, where

each observation correspond to a determined estimated position of the related laser.

Then, each particle of the filter will contain as probabilistic values the pose of the left-foot foot, and the joint map $\left\langle x_t^{left},\, m_t,\, w_t^{left} \right\rangle$.

Given an observation tuple $(z_t^{left},\, u_t^{left},\, z_t^{right},\, u_t^{right})$ the update rule for one step is for each particle *i*:

$$x_{it}^{l} = A(u_{t-1}^{left}, x_{t-1}^{left}) \qquad x_{it}^{right} = A(u_{t-1}^{right}, x_{t-1}^{right})$$

$$m_{it} = M(z_t^{left}, x_{it}^{left}, z_t^{right}, x_{it}^{right}) + m_{it-1}$$

$$\omega_{it}^{left} = S(z_t^{left}, x_{it}^{left}, m_{it-1}) S(z_t^{right}, x_{it}^{right}, m_{it-1}) \omega_{it-1}^{left}$$

where $\left\langle z_t^{left}, x_{it}^{left} \right\rangle$, is an observation tuple for laser left at time *t*, A is the actual model, M is the motion model, S is the sensor model, and ω_i is the weight for particle *i*, and m_{it} is the map guess of particle *i* at time *t*.

C. Motion model

The motion model is in charge of describing the relationship between the odometry lecture and the actual pose of the robot, taking into account the previous step position of the robot. For our particle filter, the motion model is based on odometry measurements, and it is defined for an holonomic robot. This model is applied only to the left laser in order to obtain the next step position of that foot. Laser right position is then calculated directly from the odometry related to the obtained position of the left foot, and an additional gaussian noise.

The motion of left foot is decomposed into two principle components. D will represent the movement through the major movement direction, and C will represent the shift experimented by the foot in the orthogonal direction to the major direction.

$$x_t = x_{t-1} + Dcos(\theta_{t-1} + \tfrac{T}{2}) + Ccos(\theta_{t-1} + \tfrac{T+\pi}{2})$$

$$y_t = y_{t-1} + Dsin(\theta_{t-1} + \tfrac{T}{2}) + Csin(\theta_{t-1} + \tfrac{T+\pi}{2})$$

$$\theta_t = \theta_{t-1} + T \; mod \, 2\pi$$

where θ_t is the facing angle of the robot foot at time *t*, and T is the actual turn performed. Hence, we approximate the major axis direction at time *t* by $(\theta_t + \frac{T}{2})$ and the minor by $(\theta_t + \frac{T+\pi}{2})$.

Parameters of the motion model have been tuned using the algorithm described in [15], where the parameters of the model are obtained by a learning method from recorded laser-odometry data of the robot. Basically, it consists of using an expectation maximization algorithm (EM) to estimate the model parameters. The expectation step is provided by the execution of the DP-SLAM algorithm on recorded data, starting with some random initial parameters. The possible trajectories obtained are then used in the maximization phase to create a set of parameters which best describe the motions represented by those trajectories. We refer the reader to Eliazar's paper for further information about how this optimization algorithm works.

Parameters	Value
Num. particles	1000
Grid size	3 cm

TABLE I

TABLE OF PARAMETERS USED FOR THE DP-SLAM ALGORITHM.

D. Sensor model

The sensor model is the same as described in [11]. It is a very complete model which takes into account occlusions in the ray, and depends on the distance traveled through a grid square. As counterpart, the model uses far more CPU than the typical raytracing algorithms. The model is based on the concept of *opacity* of map grids. The opacity term basically describes the probability that a laser crosses the grid, and is calculated from the number of rays that reach the grid divided by the distance travelled through that grid. The opacity of a grid defines the probability that a laser ray is interrupted on that grid (Pg), after it travels a distance x through that grid of opacity ρ. This behavior is modeled by an exponential distribution:

$$Pg(x, \rho) = 1 - e^{-x/\rho}$$

Then the probability that a laser stops at square *j* is calculated as the probability that the laser travels up to grid *j-1*, and then stops at grid *j*:

$$P(stop = j) = P_g(x_j, \rho_j)(1 - P_g(x_{1:j-1}, \rho_{1:j-1}))$$

The probability of a measurement is then the sum, over all grid squares in the range of the laser, of the product of the conditional probability of the measurement given that the beam has stopped, and the probability that the beam stopped in each square. This sensor model allows for the traveling of non uniform distances over different grids.

E. Map generator

The generation of the map is not performed using a simple ray-tracing algorithm. It is based on the special computation performed by the sensor model described in section III-D. For each grid, the algorithm maintains a computation of the total distance travelled through that square by a laser beam, and the number of times that a laser ray has stopped in the square. The ρ estimation is then provided by the quotient between the distances and the stops. ρ is used to indicate the opacity of that grid in the map construction.

F. Results

The maps generated by our robot are shown in figure 4. They were generated in realtime while the robot was driven by a joystick. The quality of the maps is impressive even when the robot moves at its highest speed (1.5 km/h) and the update rate to the particle filter is produced only twice per second[1]. Those good results may be explained by the fact that two lasers are used and when one laser is moving

[1]Due to the limited computational power onboard the robot

Fig. 4. Some map examples created by Reem-B

the other one is static. The staticity of one laser over the same group of particles may decrease the particles diversity, but on the other side, the low update rate of the filter may increase the diversity, creating a kind of compensation effect. This statement will be confirmed in future work.

IV. Localization

Once a map has been created by the SLAM algorithm, the robot can change its status to localization mode, via a vocal or GUI command, and start localizing on the generated map with no additional (post-processing) steps. Localization over that map is performed using a typical MonteCarlo particle filter approach.

A. Particle filter description

The localization module of Reem-B uses the same particle filter and motion model as for mapping (described in sections III-B and III-C). However, in order to improve the speed of the algorithm with more particles, a simpler sensor model for localization has been used. A typical sensor model based on a distance map is used (as described in [13]), which does not include the detection of occlussions. The practical results showed that the model is good enough to localize the robot with quite good stability in typical office setups.

Particles are created for one single feet and then replicated for the second one with added gaussian noise. The particle filter then maintains only one single group of particles as described in section III-B. The simplification of the sensor model allows the increase of particles used for localization, which is a very important factor specially when global localization or kidnapping are required.

In the case that the measurement of the matching decreases below a certain level, new gaussian noise is added to the pose estimation. Whenever the diversity of the particles reaches a threshold, particles are randomly distributed over the whole map, and the filter started again. This mechanism allows for a correct recuperation of the robot localization when starts up or when it is kidnapped.

B. Results

In an office environment, the robot localizes itself quite quickly and maintains its localization quite stable even when small obstacles and a few people is around it, which were not included into the original map. However, when larger obstacles are present, the localization quality decreases rapidly. Is work for future to implement on the robot some filters [16] that improve localization under those circumstances.

C. Improved global localization

In order to improve the localization abilities of the robot, an additional localization mechanism has been implemented. It consists of using visual information to re-localize the robot when the localization status decreases below a certain threshold of confidence.

During the map creation phase, the robot is allowed to acquire visual landmarks whenever required. Whenever the process of acquisition of visual landmark is activated, by a voice command, the map will include a snapshot of the current visual field, and attach this snapshot to the position detected by the robot. The distance of the snapshot is calculated from laser information. Due to this issue, the snapshot feature requires the robot to be stopped while capturing the landmark, and is limited to its use for landmarks on walls and with free space from the feet to the wall.

Whenever the localization confidence of the robot decreases a certain level, then it will automatically start a process to identify on its visual field any of the snapshots that are stored in the map. If one of the snapshots is identified, then the particles are spread around the landmark with an amount of gaussian noise.

Landmark distance to robot is calculated by the left-foot laser only during map creation, since the laser is more precise and the environment can be easily controlled during map creation. Instead, when the robot is left alone, distance to landmarks has to be performed using vision, since it cannot be assured that no obstacle will interfere the laser. Because of that, the recognition of landmarks is not affected very much by movement of the robot or the existence of small obstacles between the robot and the landmark. As drawbacks, we have a big difficulty recognizing objects in ambiance with bad light, and that the distance calculation from a stereo image is not very precise.

During the experiments performed, we observed that in indoor office-like environments, the particle filter based on laser was very stable and did not require the use of the vision add-on. Only in cases with crowded rooms the localization confidence decreased considerably and the visual localization system activated.

V. Path Planning with obstacle avoidance

The path planning module is integrated with the localization and mapping system. Using the maps created and the current estimated robot position, the path planning module is able to calculate a safe path to a requested goal position in the map. Then the module commands the robot from current position to the goal position following the path planned while avoiding obstacles. Destination points are requested via voice (for predefined locations) or through a control GUI which

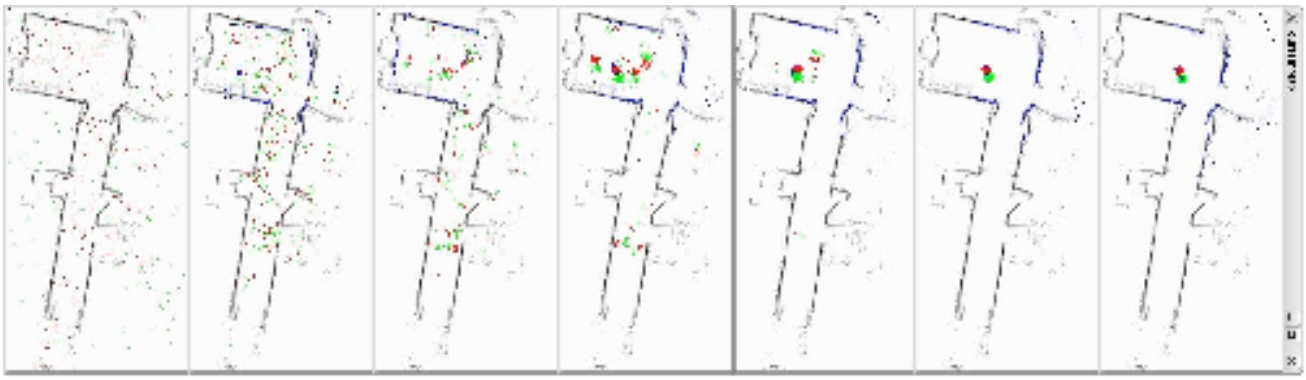

Fig. 5. Sequence of global localization. The robot is switched on at the left most picture. Then it is moved with a joystick and localization improves with it. Red dots represent the left foot position guess, and green dots the right one. Blue dots are the currently sensed obstacles by the lasers. They must match the map structure when correctly localized (as in the right-most picture).

shows the current map on an external computer and allows the selection of the destination point.

The autonomous navigation ability of Reem-B is decomposed in three different parts [17]: motion planning, motion control and obstacle avoidance.

A. Motion planning

Motion planning is performed in three phases:

On the first phase, the currently sensed obstacles by the lasers are included on a temporal copy of the current map, resulting in what we call an *updated map* (figure 6-left). Second, a *distances map* is generated from the updated map (figure 6-center). The distances map provides the places in the map that are away enough from obstacles (safe places for the robot), given the action radius of the robot. The trajectory is then calculated over the distances map. Only the free points in the map are taken into account on the calculation of the trajectory.

The trajectory is calculated using the A* algorithm. The computed path follows the points in the map that are safe enough for the robot, and provide the shortest trajectory possible. The trajectory provided by the A* algorithm (*raw trajectory*) is very abrupt which makes very difficult for the robot to follow (see pink line in the trajectory plot in figure 6-right). The trajectory is then translated to a set of straight lines (*filtered trajectory*) that best fit the raw trajectory (blue lines in figure 6-right). The filtered trajectory calculates the straight lines that best fit in the raw trajectory. The final step of the motion planning module calculates from the filtered trajectory the required *steps* for the robot to follow that trajectory. Steps indicate in local coordinates of the robot where the feet must be in order to follow the trajectory.

The calculated steps are sent to the walking server, which translates foot position into the required angles for each motor, performing the actual walking movement of the robot.

B. Motion control

Once the trajectory has been calculated over the map, it has to be performed by the robot. This is a different problem from motion planning which is called motion control. Imperfections on localization, motion of the robot, and the map,

Fig. 6. The left most figure shows the map used for navigation. The centered figure shows the calculated distance map, over which, a trajectory will be calculated. The right most figure shows the trajectory generated by the path planning algorithm from the robot (represented by two dots in red and green), and some free point in the map. The trajectory is composed of two different lines: first one, in violet, is the raw trajectory calculated by the A* algorithm. The blue line is the smoothed trajectory. Is this smoothed trajectory the one that the robot will follow.

generate differences between the trajectory planned and the real trajectory followed by the robot.

To detect those errors, the robot is equiped with a control of motion system which calculates at every time step a measurement of the difference between the trajectory path and the actual robot position. Difference is measured in terms of position and direction of movement. The direction measurement indicates how different is the current orientation of the robot related to the orientation it should have to follow the calculated line of motion. The position measurement indicates how far from the current line of motion is the robot. Both differences are obtained through a direct connection with the localization module which indicates where the robot actually is in the map.

The motion control module modifies the angle of the steps in case of errors. Whenever the orientation of the robot is above a minimal angle, steps are corrected without replanning. If for any reason, the position of the robot goes outside the route planned by a certain threshold, a complete replanning is performed.

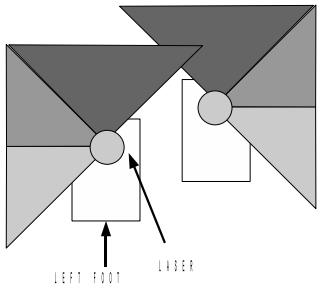

Fig. 7. Security zones of movement of the robot for the feet. Figure shows three different zones per foot, one for front movements, one for diagonal movements and another for lateral and backwards movements. There is some overlap between the front zones of both feet.

C. Obstacle detector with replanning

In order to avoid obstacles in the direction of movement, Reem-B is equipped with a module that detects obstacles in the path and replans if necessary. The robot has six security zones defined (see figure 7). Those zones are the secure movement space for the robot. If an obstacle enters one of the zones, and the direction of movement of the robot is within that zone, then the obstacle detector will signal this and stop the robot. Then a replan of the trajectory to goal is performed, taking into account this new obstacle.

The distance at which the robot detects obstacles is tuned by two parameters that specify the maximal front and lateral distances at which the obstacles will be detected. At present, those distances are set to 1.5 meters and 50 cm respectively. Obstacles detected within the security zones will only invoque replanning when the direction of movement of the robot goes in the direction of the obstacle.

Future work includes the real time modification of walking steps to dynamically avoid obstacles without replanning, by using a potential field method.

D. Results

The tests performed allowed the robot to move autonomously in office ambients. The test performed show that the robot is capable to move coherently in spaces big enough to allow the robot move freely. In smaller spaces the movement becomes less fluid. The results show that the robot does replan just in a few cases, mostly because of errors in the movement due to sleep movements in sleepy grounds. Figure 8 shows one of those cases. Figure 9 shows a change in trajectory when the obstacle detector detects an error.

VI. Discussion

On the previous section, the results presented show that, when a map is available, Reem-B can autonomously and safely move inside and office environment. However, we must point that in fact there are two modes of actuation for Reem. On the first one, the robot is joystick driven while generating the map. On the second one, while moving through the map using trajectories generated by the path planning module, the robot is completely autonomous.

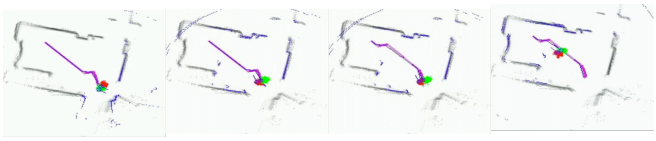

Fig. 9. Sequence of path planning-2. At the left-most picture, the robot generates a trajectory to a goal point, that is not looking at (the robot is oriented to the opposite side). Once the trajectory is planned, the robot starts moving (rotating) towards that goal while avoiding obstacles. When rotation is completed, the robot detects an obstacle just on its trajectory. Then a replan is made, taking into account the newly detected object. The new trajectory completely avoids the obstacle detected.

When in autonomous movement mode, the robot achieves with no problem to move around the obstacles and reach the desired position on the map. However, the movement of the robot is not fluid at all and *fluidity* decreases as far as the space becomes more cluttered. This is a limitation of the current path planning approach taken which does not define the trajectory as a continuos one, but as a set of straight lines that the robot has to follow. At the end of each line, there is a turn required to face the direction of the next line. This turn, stops the robot from its current forward motion. When the space is full of obstacles, the trajectory is composed of large groups of small lines, what produces the effect of low fluidity. This will be changed in the future, by planning the steps directly through the trajectory provided by the A* algorithm.

The mechanism presented can be improved in several points, as has been indicated along the text, but even on its present state, it is the first time a humanoid robot of its size uses classical laser-based techniques to guide a human-size humanoid robot in all the navigation skills required for an autonomous robot.

VII. Conclusions

Complete navigation abilities for a human size humanoid robot have been shown[2]. Even if with some limitations, this is the first time that a humanoid robot of human size is able to generate a complete map of its environment, localize on it and move autonomously on it. Those navigation abilities together with the characteristic that the robot has long lasting batteries (more than two hours), and all the computational resources required to control it are onboard, make of Reem-B one of the most autonomous humanoids in the world of its size.

References

[1] S. Thompson, S. Kagami, and K. Nishiwaki, "Localisation for autonomous humanoid navigation," in *IEEE-RAS International Conference on Humanoid Robots*, 2006.

[2] S. Thompson, Y. Kida, A. Miyazaki, and S. Kagami, "Realtime autonomous navigation with a 3d laser range sensor," in *Proceedings of 3rd Int. Conf. on Autonomous Robots and Agents*, 2006.

[3] K. Okada, M. Inaba, and H. Inoue, "Integration of real-time binocular stereo vision and whole body information for dynamic walking navigation of humanoid robot," in *Proceedings of Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, 2003.

[2] A video showing all the navigation skills described in this paper can be found at www.pal-robotics.com/media.html

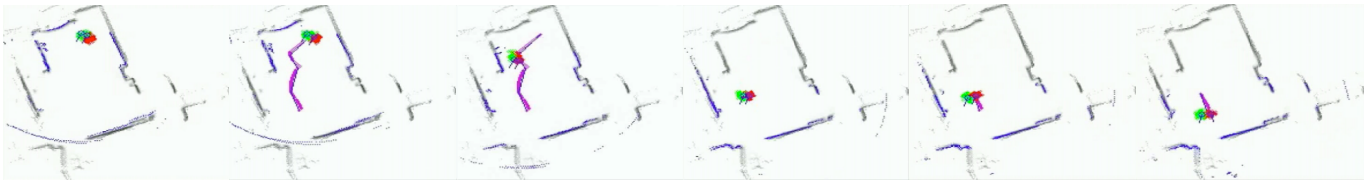

Fig. 8. Sequence of path planning-1. At the left-most picture, the robot is located at one corner of the room. Then it generates a trajectory to the opposite corner. Once the trajectory is defined, the robot moves towards the goal point while avoiding obstacles. At some point in the trajectory, the robot moves too far from the planned trajectory. Then a replan is made, and the robot follows the new trajectory until the end goal position.

[4] N. Karlsson, E. di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. Munich, "The vslam algorithm for robust localization and mapping," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005.

[5] A. J. Davison, O. Stasse, and K. Yokoi, "Vision based slam for a humanoid robot," in *International Conference on Robotics and Automation*, April 18-22 2005.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, 2007.

[7] J. Gutman, M. Fukuchi, and M. Fujita, "Real-time path planning for humanoid robot navigation," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2005, pp. 1232–1238.

[8] T. Sugihara, Y. Nakamura, and H. Inoue, "Realtime humanoid motion generation through zmp manipulation based on inverted pendulum control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[9] M. Toshihiro, H. Masanori, and Y. Shin'Ichi, "Development of laser area sensor by measuring laser pulse reflecting time," *Nippon Robotto Gakkai Gakujutsu Koenkai Yokoshu*, vol. 24, 2006.

[10] A. Eliazar and R. Parr, "Dp-slam: Fast, robust simultanious localization and mapping without predetermined landmarks," in *Proceeding of the IJCAI 2003*, 2003.

[11] ——, "Dp-slam 2.0," 2006.

[12] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (slam): Part i the essential algorithms," *Robotics and Automation Magazine*, vol. June, 2006.

[13] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, 2001.

[14] M. Kopicki, "Monte carlo localisation for mobile robots," Master's thesis, University of Birmingham, School of Computer Science, 2004.

[15] A. Eliazar and R. Parr, "Learning probabilistic motion models for mobile robots," in *proceedings of the International Conference on Machine Learning*, 2004.

[16] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of artificial intelligence research*, vol. 11, pp. 391–427, 1999.

[17] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005.