2020 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 15-16, Ponta Delgada, Portugal

Charlotte: Low-cost Open-source Semi-Autonomous Quadruped Robot

Facundo García-Cárdenas, Nelson Soberón, Oscar E. Ramos, and Ruth Canahuire
Department of Electrical and Mechatronic Engineering, Universidad de Ingenieria y Tecnologia - UTEC, Lima, Peru
Email: {facundo.garcia, nelson.soberon, oramos, rcanahuire}@utec.edu.pe

***Abstract*—The use of legged robots in non-uniform terrains is a deeply studied subject, as they are capable of performing activities such as exploration, monitoring, and collection in places where other solutions present serious disadvantages. However, most of these robots are too expensive and complex for young researchers and students to develop their algorithms and test them in a standardized model. In contrast, we propose an open-source low-cost quadruped robot that employs a ROS framework to give the learner the necessary tools to develop a basic autonomous robot, capable of mapping its surroundings while avoiding obstacles. This paper presents the details of the whole development process, which includes the motion of the robot, achieved using quadratic optimization and static stability of the center of gravity, a LiDAR-based SLAM system, and path planning using the A* algorithm. Simulation and testing show that the robot is capable of successfully achieving the tasks with responsive and natural movements.**

***Index Terms*—Legged Robot, A*, Static Stability, SLAM**

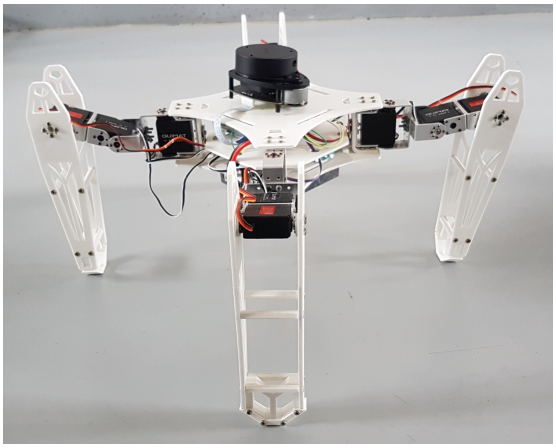

Fig. 1: Quadruped robot developed at UTEC.

I. Introduction

The implementation of robots in areas such as supervision and exploration is an ongoing subject of great interest to the field robotics community. Although well developed current applications involve wheeled robots and drones, they have been proved inefficient when dealing with challenging environments such as rainforests, cliffs, and caves [1]. To overcome the difficulties of these environments, multi-limbed robots can be used as an alternative due to their ability to move on complex terrains while maintaining stability, mobility, and dexterity. In this context, many companies decided to develop their quadruped robots, but most of these are not easily accessible to the market and do not present an open-source framework [2], [3]. Therefore, the field lacks a standardized model for researchers and students to test their algorithms. To deal with this issue, some open-source alternatives have recently appeared. These implementations involve dog-like robots that can perform robust dynamic locomotion, such as walking and trotting gaits [4], mobilize in difficult terrain [5], and exert dynamic maneuvers [6]. Nonetheless, these models are often too expensive and complex for new researchers to work upon, as they depend solely on their dynamic behavior and vary greatly depending on the structure employed.

For this purpose, this work presents Charlotte, a low-cost semi-autonomous robot with the capability of mapping its surroundings, locating itself in space, and making decisions based on its environment and its assigned task. This robot employs an arachnid-like structure, which, given its static stability, does not need a complex control, presents an omnidirectional motion, and allows for a low-cost design that does not require high-torque motors or gear mechanisms to function properly. Moreover, all the schematics and codes required for Charlotte, are published in our Github repository [7]. The model presented in this paper allows young researchers and students, which may not have access to large funds, to focus on developing complex locomotion or autonomous navigation at an affordable price of approximately $540. In comparison, the majority of research-oriented robots cost over $2000. Even though some recent publications have presented cheaper options, they achieve this by limiting the robot capabilities or fabricating their components, which is outside the scope of this paper. A detailed list of the components employed for this model is shown in the repository, which includes alternatives to lower its cost by replacing the most expensive parts of the robot (e.g., computer or servomotors) for cheaper but less capable options. The framework proposed in this paper uses an algorithm capable of mapping unknown environments and locating the robot when odometric data is not reliable [8]. This method can provide a highly detailed map of its surroundings, just by using a laser scanner, and is periodically run to deal with dynamic environments. Once the map is known, the A* algorithm is used as a pathfinding tool to calculate an optimal route to the desired position. Then, a gait planning algorithm computes the omnidirectional movement of the robot. Finally, the user can monitor and control the behavior of the robot via an intuitive Human-Machine Interface (HMI).

978-1-7281-7078-7/20/$31.00 ©2020 IEEE

II. Development of the Robot

A. Design of the Robot

The mechanical structure of the quadruped robot consists of a square base with four 3-DoF legs attached to the edges, which generate a wider work-space and avoid singularities. Also, four legs make the robot more stable and ease its locomotion on rough terrains. The structure was 3D-printed using polylactic acid (PLA), and then assembled at UTEC. The length and weight of each part are described in Table I.

TABLE I: Characteristics of each piece of the structure

Piece	Size (m)	Weight (kg)	Quantity
Base	0.195×0.195×0.065	0.286	1
Leg	0.27×0.05×0.06	0.086	4
Motors	0.11×0.08×0.025	0.1	12

To control the 12 servomotors, a PCA9685 servo driver was used together with an optocoupler plate to protect the main controller board. For the tasks of autonomous mapping, navigation, and locomotion, an RPLIDAR-A1M8 with an MPU-9250 IMU was attached on top of the mechanical chassis. All the subsystems were controlled using an onboard NVIDIA Jetson Nano. The resulting robot is shown in Fig. 1.

B. Simulation

To test different motion control algorithms, a Gazebo simulation was developed, as shown in Fig. 2. This dynamic simulation takes into consideration the weight distribution, inertia tensors, and the collision blocks of the body to perform a more realistic behavior. Each joint is simulated using damping and friction models to imitate the performance of the servomotors, and a PID position controller was implemented for each motor. Joint limits for the position, velocity, and effort of each joint were also set in the URDF model to ensure that the generated motion is feasible using the real robot constraints.

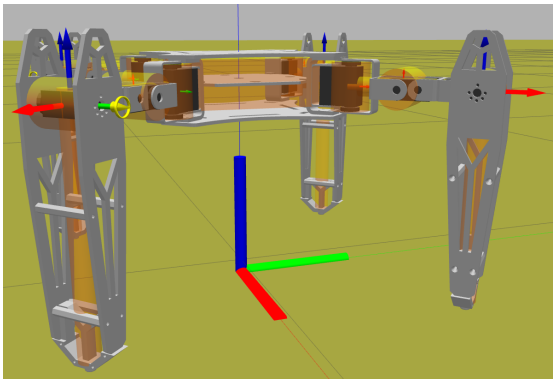

Fig. 2: Gazebo simulation for the quadruped robot.

C. Human-Machine Interface

To avoid possible accidents and help identify pragmatic errors, an intuitive interface was developed, as shown in Fig. 3. The information displayed includes position feedback and simulation, video transmission, and 3D mapping of a designated area. The main function of this HMI is to provide user control over the current position and tasks of the robot, described in both joint and task spaces. Also, a simulation tab can be used to emulate the robot's behavior in a controlled virtual environment. This view guarantees the accurate performance of the robot and helps identify issues such as incorrect data transmission and mechanical failures. Moreover, there is an option to modify the kinematic control parameters to vary the behavior of the robot, depending on the current situation. In regards to the development, the interface was built primarily using Python for the back-end functioning and PyQt for the front-end appearance.

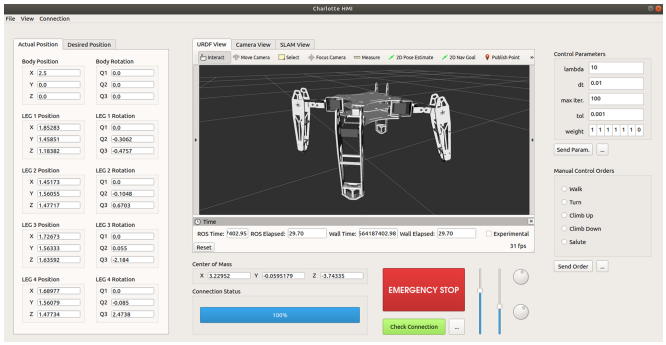

Fig. 3: Human-Machine Interface displaying the URDF model of the robot.

III. Kinematic Motion Control of the Quadruped

A. Task-space Kinematic Model

For motion control, since actuators operate in the joint space, but tasks (and their associated constraints) cannot be naturally described in this space, the map between the joint space and the task space is needed. For the proposed quadruped robot, the operational points, for which the tasks are specified in, are located at the end tip of each limb. Thus, the task space of each operational point is associated with its respective position and orientation, represented using Cartesian coordinates and a quaternion. The kinematic model for each operational point is obtained using the standard Denavit-Hartenberg (D-H) convention [9].

Given that the robot is composed of 4 identical legs attached to a base, it is possible to assign the same D-H parameters to each of them and then use a homogeneous transformation matrix to adjust their position. Considering that the inertial frame of reference is located at the center of the square base, to achieve symmetry, the modeled parameters are obtained as shown in Table II. The D-H parameters are used to obtain the homogeneous transformation matrices $\mathbf{T}(q_i)$ that define the pose of each i-th operational point with respect to the robot base. A well-known mapping from a rotation matrix to a quaternion is then applied to represent the orientation. However, in order to consider the effect of the floating base on the pose of each operational point, as described in [10], it is necessary to use the generalized joint space as $\mathbf{q} = [\mathbf{x}_b \ \mathbf{q}_a]^T$, where $\mathbf{x}_b \in \mathrm{SE}(3)$ represents the pose of the base, and $\mathbf{q}_a \in \mathbb{R}^n$ the actuated joint configuration, with $n = 12$.

TABLE II: D-H Parameters for each 3-DOF leg

Link	a_i	α_i	d_i	θ_i	Home
1	0.055	$\frac{\pi}{2}$	0	θ_1	0
2	0.075	π	0	θ_2	$-\frac{\pi}{2}$
3	0.225	0	0	θ_3	0

B. Kinematic Pose Task

To control the pose of an operational point, it is necessary to first define the error $\mathbf{e}$ between the current and the desired pose. For the position $\mathbf{x}_p \in \mathbb{R}^3$, the error is defined as a term by term difference $\mathbf{e}_p = \mathbf{x}_p - \mathbf{x}_p^*$, where $\mathbf{x}_p^*$ is the desired position. For the orientation, if $\mathbf{x}_o = Q$ is a quaternion $Q = (w, \epsilon_x, \epsilon_y, \epsilon_z) \in \mathcal{H}$, where $\mathcal{H}$ is the Hamiltonian space, the quaternion error is defined as $Q_e = Q_d Q^{-1}$, where Q_d is the desired orientation. This error can be expanded as $Q_e = (w_d w + \epsilon_d^T \epsilon_d, -w_d \epsilon + w\epsilon_d - \epsilon_d \times \epsilon)$, and the orientation error is obtained with respect to the identity as $\mathbf{e}_o = [w_e - 1, \epsilon_e]$.

For the i-th operational point, the differential relation between the task space and the joint space is given by the task Jacobian $\mathbf{J}_i = \frac{\partial \mathbf{x}_i}{\partial \mathbf{q}}$. Since $\dot{\mathbf{x}} = \dot{\mathbf{e}}$, assuming a constant or slowly moving desired pose, the derivative of the generalized joint configuration can be found as

$$\dot{\mathbf{q}} = J_i^{\#} \dot{\mathbf{e}}_i \tag{1}$$

where $J_i^{\#}$ is the pseudo-inverse of J_i. To ensure the convergence of the system, the control law $\dot{\mathbf{e}}^* = -\lambda \mathbf{e}$ is set using a kinematic task gain $\lambda > 0$. To achieve faster convergence, this gain can be given by

$$\lambda = (\lambda_{max} - \lambda_{min}) e^{-k\|\mathbf{e}\|} + \lambda_{min} \tag{2}$$

where λ_{max} and λ_{min} are hyperparameters that define the convergence rate, and must be empirically chosen.

C. Kinematic Control

Since the quadruped robot presents an inherently redundant structure, the approach for its whole-body control uses an optimization-based differential kinematic solution. The method consists of finding the generalized joint velocity vector $\dot{\mathbf{q}}$ using a weighted quadratic optimizer of t tasks defining the desired motion. Each task has a weight $w_i > 0$, and the functional constraints characterize the degree of dexterity of the redundant structure [11]. This scheme is defined as:

$$\min_{\dot{\mathbf{q}}} \left\{ \dot{\mathbf{q}}^T W \dot{\mathbf{q}} + \dot{\mathbf{q}}^T \mathbf{p} \right\}$$
$$\text{s.t.} \tag{3}$$
$$\max \left\{ \frac{\underline{\mathbf{q}} - \mathbf{q}}{\Delta t}, \underline{\dot{\mathbf{q}}} \right\} \leq \dot{\mathbf{q}} \leq \min \left\{ \frac{\overline{\mathbf{q}} - \mathbf{q}}{\Delta t}, \overline{\dot{\mathbf{q}}} \right\}$$

where

$$W = \sum_{i=1}^{t} w_i J_i^T J_i, \quad \mathbf{p} = -2 \sum_{i=1}^{t} w_i J_i^T \dot{\mathbf{e}}_i,$$

$\underline{\mathbf{q}}$, $\bar{\mathbf{q}}$ are the lower and upper joint angular limits, and Δt is the time between each control signal [12]. With this approach, a larger weight is related to a higher importance of the task, but there is always a compromise amongst all the tasks [13]. Since the motion of the quadruped involves always keeping the balance, a task that considers the Jacobian of the CoG was added with a very high priority.

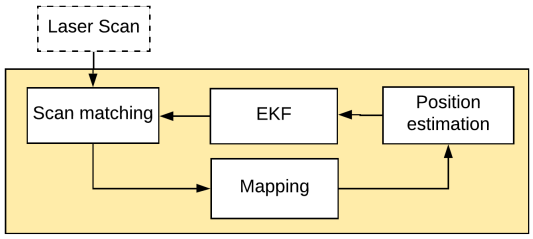

Fig. 4: Overview of the SLAM process.

IV. High-level Motion Generation

A. Simultaneous Localization and Mapping (SLAM)

In order to detect obstacles and generate a map for localization, it is necessary to implement SLAM, and we used the Hector SLAM algorithm [14]. The scan measurements were obtained using an RPLIDAR-A1M8 module, which is a 360° laser scanner with a maximum range up to 12 meters. This LiDAR was chosen due to cost-effective reasons and its wide range, which is enough for mapping a room or a corridor.

The algorithm in our system works by combining scan matches with attitude estimations, and is capable of locating the robot using only the LiDAR readings. Since the IMU alone is not reliable for obtaining the robot position, this feature is indispensable. The required data is obtained using the pose updates, based on the analysis of the laser scan at each timestamp. Fig. 4 shows an overview of the process: by applying an Extended Kalman Filter (EKF), the current laser scan is compared to the map generated during the previous timestamps. Here, static objects such as walls and columns are used as reference features. This method results in a fast and accurate SLAM system, which does not require additional sensors or high computational resources.

B. Path planning algorithm

Given that the robot will be located in unknown environments, the system requires an algorithm that incentives exploring and, based on the collected information, computes the best route to the target position. Even though a variety of algorithms are capable of dealing with this problem (e.g., Artificial Potential Fields or Focussed D*), the implementation of a complex algorithm is not in the scope of this paper. Therefore, the A* algorithm was chosen due to its simplicity and efficiency. This approach requires a map consisting of weighted values in a fixed two-dimensional space grid, which is provided by the SLAM. These values, called nodes p, correspond to each point of the map and can either be zero or one, representing walkable or non-walkable terrain, respectively. Furthermore, as the map is periodically updated, the trajectory is recalculated to deal with changes in the environment.

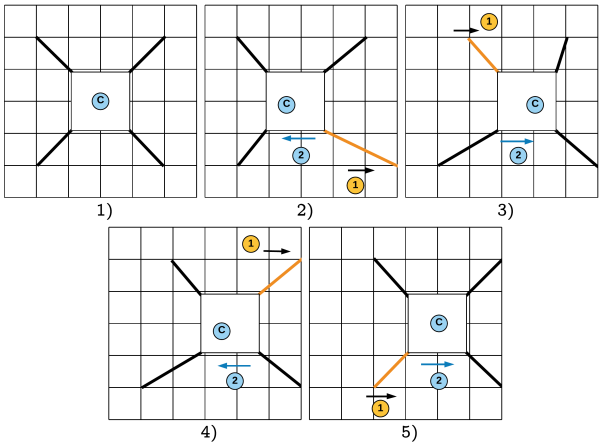

Fig. 5: Walking pattern using the principle of static stability.

C. Gait Planning

The proposed walking pattern allows the robot to move in any direction without needing to rotate its base. This sequence is based on static stability; that is, for the robot to be stable in each step, the center of gravity (CoG) must always be kept within the support polygon formed by the contact points of the remaining legs on the ground [15]. To ensure the omnidirectional movement of the robot, all parameters are expressed based on the angle formed by the direction vector and the inertial x-axis. Once the angle is found, the Euclidean distance between the initial and desired position is computed.

On the other hand, to find the necessary number of steps, every step is assumed to be located inside the primary work-space of the robot. As a result, the number of steps is equal to the quotient between the total distance and the work-space limits. Finally, considering the start and end points given by the A* algorithm, the director vector (x, y) is computed, an it selects the working quadrant and the leg that starts the pattern. A summary of the proposed method is presented in Fig. 5, where each step of the walking pattern ensures the static stability of the system.

V. Results

The tests applied to the robot suggested that the weighted quadratic optimizer, for the task-space kinematic model, is a versatile solution that allowed the robot to achieve omni-directional motion and freely change the pose of its base. Fig. 7 shows a dynamic Gazebo simulation, where the robot was commanded to reach a point outside of its working area, allowing to test the capabilities of the gait planning algorithm. The simulation achieved the desired position with a mean error of 0.0321m, in a computational time of 41ms per task, using an Intel Core i7-8550U CPU @ 1.80GHz, running Ubuntu 18.04. In this example, the model proved to be stable, since the robot's position converged once the task was completed. Besides, the robot was also capable of adequately controlling the CoG, with generated gaits that kept the CoG inside the

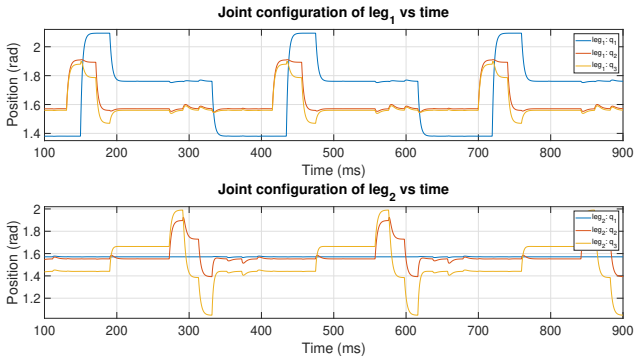

Fig. 6: Diagonal walk pattern in the joint space of the quadruped robot.

support polygon. On average, each movement of the CoG presented an error of 0.4 mm.

To accomplish the aforementioned task, the robot was commanded to move diagonally. The behavior of the robot in the joint space, for legs 1 and 2, is presented in Fig. 6, where it is possible to observe the periodicity of the walk, as well as the asymptotic stability and low error of the control law. It can be also inferred that there is a compromise between the tasks, in which a joint may surpass its desired position to allow other tasks to converge, proving the multitasking capabilities of this method. Moreover, each joint of the robot was able to effectively maintain its desired position while using the necessary torque to withstand the weight of the structure. This proves that the PID control of the servomotors was adequate, as it presented a fast and stable response.

To test the dexterity and accuracy achieved by the task-space kinematic control, a joystick controller was connected to the quadruped robot. Then, the robot was commanded to follow a circular trajectory using its base, which was easily obtained with a mean error of 0.011 m during this task. On the other hand, the rotation of the base was also tested using this method. Here, the robot was ordered to rotate its base in the range of $[-\frac{2\pi}{5}, \frac{2\pi}{5}]$, around the z-axis. Then, the same task was replicated in the range of $[-\frac{\pi}{6}, \frac{\pi}{6}]$ around the x-axis and y-axis. For the first task, the robot achieved a mean error of 0.213°, while the analysis of the other tasks presented a mean error of just 0.411° and 0.376°, respectively. This proved that the quaternion method of orientation control, and the conversion to Euler angles - to achieve a more intuitive user control - were both successful. Once the robot proved to be stable inside the simulated tests, the algorithm was implemented in the prototype robot, as shown in Fig. 8.

For all of these tests, the selected value for Δt was 0.01s and the kinematic gain limits were assigned as $[1; 30]$. It is important to select these values based on the task and its relationship with the joint limits of the robot, as a high kinematic gain allows to achieve the desired position faster. However, if the objective is outside of the working area, the robot may present a risky behavior where the motors employ a high torque or the robot collisions with itself. On the other hand, it is worth mentioning that each of these values indirectly controls the number of iterations needed to reach the objective,

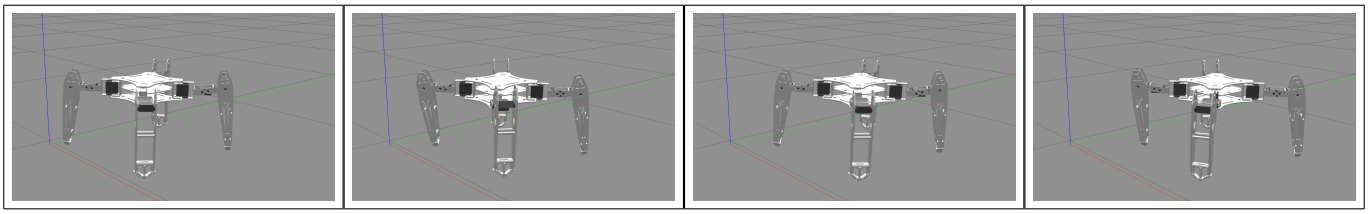

Fig. 7: Gazebo simulation of the diagonal walk pattern of the quadruped robot.

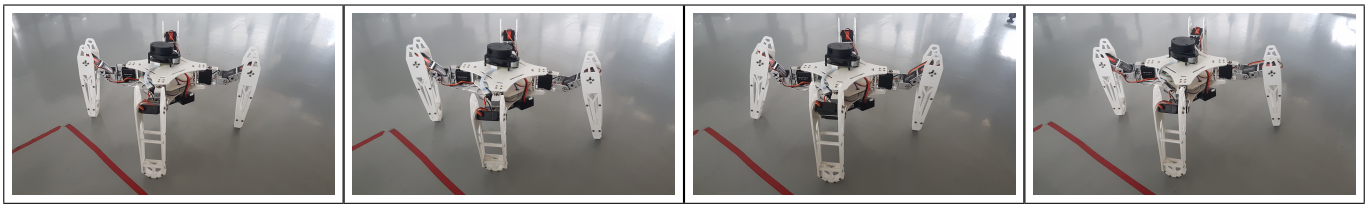

Fig. 8: Implementation of the diagonal walk pattern of the quadruped robot.

as well as the precision of the task. Additionally, the choice of the λ parameter resulted in a notorious upgrade from the previous control law [16]. The reason behind this is presented by solving the differential equation $\dot{\mathbf{x}} = \dot{\mathbf{e}}$,

$$e(t) = e^{-\lambda(t-t_0)+\ln(e_0)} \tag{4}$$

As shown, this control law ensures the asymptotic stability of the system, where the kinematic gain λ controls how fast it converges. However, (2) modifies this relationship by generating a dynamic kinematic gain, which increases the further away the current configuration is from the objective. This generates a behavior in which the movement of the robot is faster at the beginning to shorten the task time, but then lowers to ensure the desired precision.

On the other hand, the Hector SLAM algorithm proved to be efficient at mapping a small corridor, as shown in Fig. 9. The information provided by the laser scanner alone was accurate enough for the algorithm to work. To obtain a highly-detailed map, the point cloud data was updated at a frequency of 10 Hz. Nonetheless, once the readings were processed, the map presented an average error of 0.013 meters. This was mainly caused by the ~1% error in the laser scanner readings, combined with the fast motion of the robot's base generated during the tasks. Moreover, it was noted that some corners were curved after some time, as the algorithm could not make up for the error in orientation. Additionally, this algorithm fails when there are not enough reference points to locate the robot. However, it is possible to correct this issue by fusing the data provided by the Inertial Measurement Unit (IMU), which presented a mean error of 1.87° for the orientation of the robot.

To further examine the behavior of the SLAM system, and its integration to the A* algorithm, the following 3 sensors were added to the simulation: IMU, camera, LIDAR. As shown in Fig. 10, the robot was able to freely move around

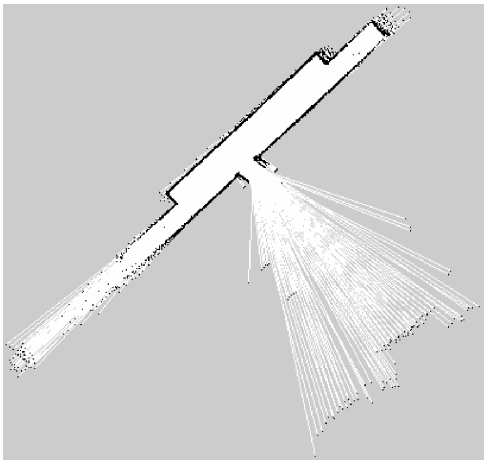

Fig. 9: Map generated by the Hector SLAM algorithm.

a simulated garage, in which it was tasked to map its surroundings and avoid obstacles. The generated map presented similar features to the one described during the implementation phase. However, even though some white noise was added to the LIDAR readings, the map presented a lower error in comparison. This allowed testing the capabilities of the A* algorithm without worrying about the abovementioned error.

The map generated by the SLAM system was divided into a grid, in which each pixel had a corresponding weight; where obstacles and inaccessible - or unknown - areas weighted 1, while the accessible areas weighted 0. Additionally, unnecessary gray pixels were cropped to improve performance. The resulting map formed a weighted grid of 380×400 pixels. Subsequently, the start and finish points were selected. As shown in Fig. 11, the algorithm proved to be efficient at reaching the desired position, while avoiding a wall and a piece of furniture, in a computational time of 150ms. During both simulation and implementation, the A* algorithm presented

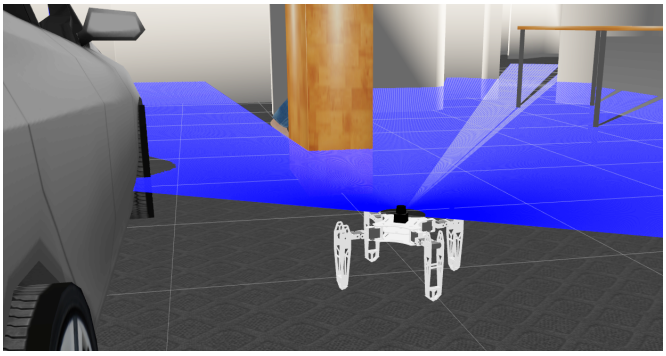

Fig. 10: Laser scan readings during the simulation inside the garage world.

fast and accurate results. Nonetheless, this method does not produce the shortest path for all scenarios, as it relies heavily on approximations to calculate the heuristic function $h(p)$. Furthermore, it sometimes collides with obstacles, as it does not consider the size of the robot. To solve this, an alternative using Artificial Potential Fields (APF) is proposed.

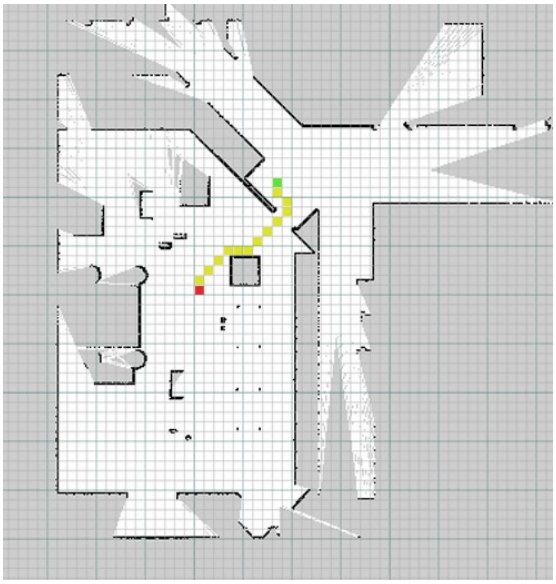

Fig. 11: Visual representation of the route generated by the A* algorithm.

Finally, the HMI resulted in a very intuitive way to control the robot and troubleshoot some errors. Nevertheless, given the amount of information being sent at the same time, there was a noticeable delay in the communication between the robot's processor and the working station. To solve this, using a UDP protocol for video footage and mapping; as well as reducing their data frequency, allowed to increase the communication speed overall.

VI. Conclusion

The implementation of Charlotte resulted in a low-cost, open-source, and efficient framework for young researchers who have an interest in developing autonomous robots. The structure of the ROS system allows the learners to modify each section of the control algorithm, separately. The quadratic optimizer resulted in a precise and stable solution for the kinematic redundancy of the robot, which allows it to obtain whole-body motion control and perform multitasking. Moreover, this method allowed to control the center of mass of the robot with a high degree of precision and, as a consequence, generate a walking pattern that follows the principle of static stability. Added to this, the trajectory planning by A*, mixed with Hector Mapping, resulted in an efficient solution to locate in space and avoid objects. Although the computational time of the A* algorithm is high, route retracing is only needed when new obstacles are detected in the map; thus, it does not significantly affect the total performance of the system. In future work, it is suggested to make a sensor array to improve the quality of the maps generated by the SLAM. Additionally, the inclusion of an MPC controller is being discussed to increase the accuracy of the walking patterns. Finally, an improved architecture is currently in development to allow for a greater degree of flexibility, a lower weight and facilitate the integration of new and upgraded sensors.

References

[1] A. Zhilenkov and I. Epifantsev, "System of autonomous navigation of the drone in difficult conditions of the forest trails," *IEEE Conf of Russian Young Researchers in Electrical and Electronic Engineering*, 2018.

[2] S. Seok, A. Wang, Meng Yee Chuah, D. Otten, J. Lang, and S. Kim, "Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 3307–3312.

[3] M. B. R. Playter and M. Raibert, "Bigdog," *Proc. SPIE 6230, Unmanned Systems Technology VIII, 62302O*, 05 2006.

[4] A. T. Spröwitz, A. Tuleu *et al.*, "Oncilla robot: a versatile open-source quadruped research robot with compliant pantograph legs," *Frontiers in Robotics and AI*, vol. 5, p. 67, 2018.

[5] N. Kau, A. Schultz, N. Ferrante, and P. Slade, "Stanford doggo: An open-source, quasi-direct-drive quadruped," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6309–6315.

[6] C. Semini, N. Tsagarakis *et al.*, "Design of hyq -a hydraulically and electrically actuated quadruped robot," *Journal of Systems and Control Engineering*, vol. 225, pp. 831–849, 08 2011.

[7] F. Garcia-Cardenas and N. Soberon, "Open-source Quadruped Robot Repository," Oct. 2019. [Online]. Available: https://github.com/fgarciacardenas/Charlotte

[8] M.-Y. Ju, Y.-J. Chen, and W.-C. Jiang, "Implementation of odometry with ekf in hector slam methods," *International Journal of Automation and Smart Technology*, vol. 8, no. 1, pp. 9–18, 2018.

[9] P. I. Corke, "A simple and systematic approach to assigning denavit-hartenberg parameters," *in IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 590–594, June 2007.

[10] O. E. Ramos, "Step-by-step kinematic modeling and control of a robot with a free-floating base," in *IEEE ANDESCON*, 2018.

[11] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal of Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[12] O. Ramos, "A kinematic whole-body control system for highly redundant robots," *IEEE Andescon*, 2016.

[13] P. Baerlocher and R. Boulic, "An inverse kinematics architecture enforcing an arbitrary number of strict priority levels," *The visual computer*, vol. 20, no. 6, pp. 402–417, 2004.

[14] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," *IEEE International Symposium on Safety, Security and Rescue Robotics*, November 2011.

[15] T. K. Kumar, R. Kumar, S. Mogily *et al.*, "Implementation of gaits for achieving omnidirectional walking in a quadruped robot," in *Proceedings of the 2015 Conference on Advances In Robotics*, 2015, pp. 1–6.

[16] F. García-Cárdenas, O.E.Ramos, and R. Canahuire, "Task-space kinematic control of a quadruped robot with a floating base," *IEEE Colombian Conference on Robotics and Automation*, 2018.