Full Speed Jackal

A project of the 2016 Robotics Course of the School of Information Science and Technology (SIST) of ShanghaiTech University

Instructor: Prof. Sören Schwertfeger · https://robotics.shanghaitech.edu.cn/teaching/robotics2016

Jiawei Hou, Zhiming Li, Yanlin Zha,

***Abstract*—We applied two ROS packages for an autonomous robot, Jackal, driving itself along a sub-optimal trajectory to a given terminal with avoiding obstacles. We tried to formulate the problem via combining simultaneously locating and mapping (SLAM) and mixed-integer optimal control programming (MIOCP) and then applied move_base package to make local plan. The Jackal identifies obstacles, locates itself via laser sensor. Then, environmental information is passed to move_base as a low-level controller with a given goal to make local plan. These local goals are computed by the high-level optimal controller that regards the jackal as a linear system and obstacles as integer constraints to enforce collision avoidance and sub-optimality.**

Currently, we have finished the low-level controller and high-level controller independently. Moreover, we have made a demo to show the low-level controller performs well in avoiding obstacles. It is worth to note that all computation are finished in real-time and this method also allows Jackel to avoid moving obstacles. This project will not terminate at present and we will continue to combing low-level and high-level controller to achieve automatic driving.

***Index Terms*—ROS, move_base, velodyne, MIOCP.**

I. Introduction

These years are nearly dawn of the era of automatic driving, since relevant theories (e.g. optimization in real-time), algorithms (e.g. simultaneously locating and mapping), and hardware resources, are all mature. Currently how to combine them and implement is the main challenge of this technique. In robotics point-of-view, autonomous vehicles are naturally regarded as autonomous mobile robots, those who is capable to go to some place with building and refreshing maps via detecting obstacles in real time. Hence, Jackal, that is such an autonomous mobile robot, is an ideal platform to verify automatic driving technique.

In this project, *Ful Speed Jackal*, we would like to exploit the algorithm of trajectory planning with collision avoidance and implement a Jackal controller which are able to drive Jackal automatically in its full speed (2m/s). Furthermore, the Jackal should be able to find a sub-optimal trajectory to the given terminal with avoiding static obstacles at least. In order to achieve this goal, we would like to propose an approach combining the state-of-the-art techniques of ROS and Mixed Integer Optimal Control Programming (MIOCP) and verify it with several experiments.

The structure of this proposal is as follows. The sate-of-the-art situation is discussed in Sect.2. Relevant theories, algorithms, open-source-ROS packages, as well as similar research achievements are introduced. In Sect.3, we describe our idea and proposed system in detail. Then, system evaluation are presented in Sect.4. Finally, a brief summary to this project is in Sect.5.

II. State of the Art

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. There are considerable works related to autonomous driving. In this section, these relevant works are introduced in these three subsections: mapping, locating, and controlling.

A. Obstacles Detection

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Obstacles Detection is a very important part in this project, including using laser sensor to build the cloud points and detecting obstacles from the cloud points. We use the package (velodyne) to receive the sensor's data and publish the environment point cloud around the Jackal to the topic (\velodyne\points). After receive the environment data, we detect obstacles (as circles and segments) in cloud points and estimate their positions and size. This process achieved by existing open-source package (obstacle_detector[1]).

For dynamic obstacle, there is a paper[2] introducing a method to detect dynamic obstructions, which could be applied to urban autonomous vehicles. In order to predict the movement of objects and avoid collisions, researchers estimate the local trajectory of the oncoming vehicle and evaluate each trajectory to select collision-free trajectories. In the *Full Speed Jackel* project, we would like to employ the approaches to predict the movement of obstacles, and select feasible trajectories by solving MIOCP problem.

For detection of terrain, which also plays a key role, this paper[3] describes the terrain detection by detecting step and slope. In order to detect and track static objects, researchers proposed a robust algorithm[4], where they built a system, based on laser range finder data[5], [5], [6], to implement the detection and tracking of moving objects (DATMO). Furthermore, they used a lot of space on how to fit model for obstacles to detect which obstacle points belong to the same obstacle, which is applicable and robust when velodyne (a laser scanner) is utilized. This laser scanner gives the scan segments in the 2D plane, i.e. a set of ordered points with a constant step size. The first step is segment processing, which adds points to an existing cluster. If the distance from point to the cluster is less than the threshold, then add the point to the cluster. Next, it extracts the features from segments that are stable to view. Fit the model as a corner or a line, in order to let the fit be robustness to the outliers, a test should be performed for fitting. Last step is the object tracking. In order to track the object, first it should associate datas between scans, mainly by detecting the the overlap area of the prediction of object area and the current object area. In addition, this algorithm is applicable not only to the detection of static obstacles, but also to the detection of dynamic obstacles. This allows us to detect some fixed objects while also detecting pedestrians or other obstacles that can not be predicted by advance mapping, which is more suitable for autonomous vehicles.

In this part, we use velodyne [7] to get the data of the environment around the Jackal. Velodyne is a sensor can map 360 3D point clouds, which is often used in 3D mapping and autonomous navigation. First we use the ROS Package velodyne[8] to get the raw data from velodyne. The package receives raw velodyne data from the device, then publish the topic \velodyne_points, which includes the information of the 3d point cloud of the environment. Besides, the package includes a calibration file, which can be launched easily by us to calibrate the sensor.

We use obstacle_detector to find points on obstacles. It subscribes the topic \velodyne_points, and detects obstacles as two kinds, circle and segments, according to the threshold the user set, and publishes to the topic \obstacles.

B. Locating

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Localization of a mobile robot is to identify the robot's current position when moving or change directions in the environment. There are several methods has been developed to detect the position and the velocity of the mobile robot. The result can be different depending on the environment. A wheel encoder has been widely used for localization. However, when the wheels slipping or the plane moving on is uneven, the performance would be poor[1]. By using GPS and sonar, localization can be done well in the outside environment. However, it is restricted by the signal receiving and in the inner environment, GPS cannot be used to localize the mobile robot[2].

In this project, inertial measurement unit (IMU) is applied in our system to calculate the position and the orientation of our mobile robot by the numerical integration of the acceleration and angular velocity measurements from an IMU. However, errors may be increased due to noise, bias error of the sensor outputs. So when IMU is used as a position sensor, simultaneously using rotary encoders can improve the position accuracy. A filter method is introduced to lower the errors generated in calculating the position and the orientation by using Kalman Filtering and Particle Filtering. In this paper[3], researchers investigated the effects on the different orientation errors by conducting three simulations in 6-DoF. In the simulation, the position error reduces significantly by using Kalman Filter and by using the postponed filtering, the error can be even smaller; As to the orientation error, there is no difference between the Kalman filtering and the gyro integration method, and when postponed filtering is applied, the orientation error is significantly reduced.[3] Another problem is the differential drive. Our robot is a robot whose wheels are installed on each side, and the motion control is achieved by changing the speed of each wheels. When the robots need to change the direction, the velocity of the wheels on both sides is different. To tackle the problems discussed above, we introduce an open-source ROS package: differential_drive,xsens_driver[9], [10].

The xsens_driver provides a driver for the Xsens IMU devices. It publishes orientation, angular velocity, linear velocity, linear acceleration, altitude, latitude and longitude (covariances are not yet supported). As a ROS node, it only forwards the data streamed onto ROS topics. There is a few Xsens IMU devices, however, compared to the other MTi drivers (lse_xsens_mti and xsens_mti), this one can handle other configurations than the default and the GPS module of the MTi-G.

The differential_drive can receive wheel rotary encoder messages from the hardware and generate the tf transform messages, then transmit them to the navigation stacks. It can also transfer the twist into target velocity of each wheels, then controls the velocity of the wheels by basic PID velocity controllers for each wheels. This package consists of the following nodes we might use:

diff_tf;.
pid_velocity
twist_to_motors

Finally, it it worth to note that even through there are several publications related to sensor fusion (e.g. [11], [12]), sensor fusion is still a challenge and we do not find a practical ROS package to combine location data from multi-sensors such as GPS, IMU, encoder, as well as scanner. We will try the strategy discussed above firstly, and write our own program to implement that if necessary.

C. Controlling

1) High-level Controller: The considered high-level controller is similar as [13], where the authors applied mixed-integer quadratic programming to generate optimal trajectory with collision avoidance. Here combined with the model predictive control scheme (MPC), MIOCP is introduced with its formulation.

The main idea of MIOCP is formulating conditions of collision avoidance as constrains with binary variables. The optimal trajectory is generated via solving the MIOCP. This method is able to deal with complex environment, e.g. over two moving or static obstacles with appropriate speed is closing to the Jackel. However, the limitation of this kind of approaches is hard to compute sometimes even NP-hard[14]. Now, thanks to the developments of computational ability, the MIOCP is able to be solver in real-time as long as the binary variables are not too much (e.g. over 1,000) which is enough for Jackel. Actually, around 2005, Richards and coworkers wrote a series of paper about combining MIP with MPC summarized in[15] to generate trajectory for their vehicles. Later, Sager investigated Mixed Integer Optimal Control Problems(MIOCP) systematically, e.g. [16], [17], [18]. Recently, Kumar et.al. applied the MIQP in quadrocopters[13]. They generated the map firstly, then apply A* algorithm to find a pre-defind trajectory. Next, the formulated their system as a MIQP and solved MIQP in real time to generate the optimal trajectory with its control sequence. In this project, we also formulate our system as MIOCP, and combine the solution of MIOCP with MPC sheme, which means the Jackal need to solve refreshed MIOCP in each step to obtain the newest first one of the control sequence to execute , which will lead to optimal trajectory, until the Jackal arrive at the given terminal.

There exist a few of MIP solvers such as Gurobi, CPLEX, and MUSCOD-II which is specialized for MIOCP. In this project, we would like to apply Gurobi.

Gurobi is a commercial solver to deal with MIOCP problem. The academical version is free for us and can be installed in Linux environment. It also provides Python, C++, Julia interface for simulation and utilization. The latest version is 7.0, which solve MIOCP by branch-and-cut with heuristic algorithms. Basic branch-and-cut algorithm can be described as follows.

Firstly, remove all of the integrality restrictions. The resulting OCP is called the OCP relaxation of the original MIOCP. We can then solve this OCP by common solver e.g. qpOASES. If the result happens to satisfy all of the integrality restrictions,

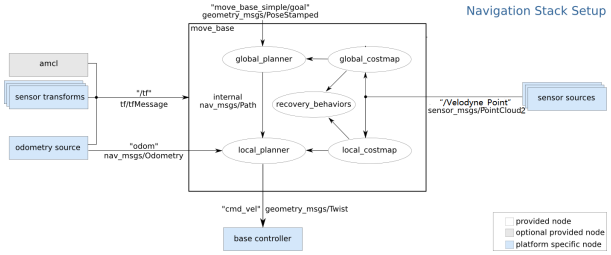

Fig. 1. Framework of move_base

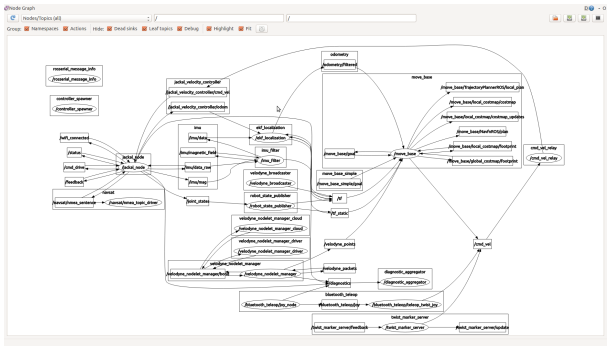

Fig. 2. Topics in move_base

even though these were not explicitly imposed, then we have been quite lucky. This solution is an optimal solution of the original MIOCP, and we can stop. If not, as is usually the case, then the normal procedure is to pick some variable that is restricted to be integer, but whose value in the OCP relaxation is fractional. Next divide this OCP to two or more parallel subproblems, and iterate this process until the solution is found.

Here, we need to write our own ROS package to utilize Gurobi to achieve our goal.

2) Low-level Controller: We apply the move_base package as the low-level controller. The move_base package provides an implementation of an action that, given a goal in the world, will attempt to reach it with a mobile base. The move_base node links together a global and local planner to accomplish its global navigation task. It supports any global planner adhering to the nav_core::BaseGlobalPlanner interface specified in the nav_core package and any local planner adhering to the nav_core::BaseLocalPlanner interface specified in the nav_core package. The move_base node also maintains two costmaps, one for the global planner, and one for a local planner (see the costmap_2d package) that are used to accomplish navigation tasks.

The move_base node provides a ROS interface for configuring, running, and interacting with the navigation stack on a robot. A high-level view of the move_base node and its interaction with other components is shown. The blue vary based on the robot platform, the gray are optional but are provided for all systems, and the white nodes are required but also provided for all systems.

The move_base node basically consists of 5 parts: global_costmap, global_planner, recovery_behaviors, local_costmap and local_planner. In the real robot, topic /tf

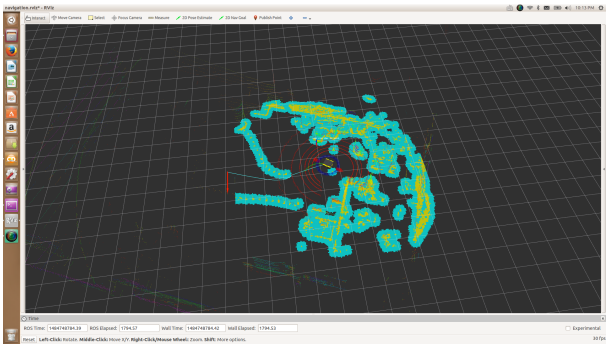

Fig. 3. Path planning

and /odom would be published by the robot, and the sensor(in our project, we use Velodyne VLP-16) would publish the information of environment around the robot, the move_base node will subscribe all the topics and according to the topic move_base_simple/goal to plan both the global path and the local path, which is formed by the global planner and the local_planner. In this experiment, numbers of parameters need to be set to make the real robot move smoothly and accuracy. These parameters are set in the .yaml file, which was listed below:

- base_local_planner_params.yaml
- costmap_common_params.yaml
- global_costmap_params.yaml
- local_costmap_params.yaml

In base_local_planner_params.yaml, which concludes parameters about velocity of both dx and dy, $dtheta$ velocities, the path length local planner will make, and the mark system for pathplanning. The local_planner will set this parameters and plan the local path to avoid obstacles and choose the best one. In costmap_common_params.ymal, some parameters on sensor such as message type, topic, obstacle height ranges are set. Global_planner will form a global path from the start to the terminal by using Obstacle::Layer to form obstacle models and Inflation Layer to inflate the obstacles.

The demo graph is shown in the Figure 3, the blue part is the obstacles inflation, and the yellow part are the obstacles. The green line is the global path from the robot to the end, and the deep green is the path local planner makes.

By the time all the parameters are set, and all the information or the topics are published, the robot will do obstacle detection and path planning, and moving in a full speed.

III. System Description:

As it stated above, our ideas are summarized here. We would like to apply Velodyne, Gurobi, move_base and the Jackal Odometry in our system. The overall system is consisted by three parts: detecting, locating, and controlling.

For detecting, we would like to utilize the ROS package velodyne, to obtain the environment information and build as well as refresh point cloud in real time. For location, we subscribe the topic (\odometry\filtered) published by Jackal itself to estimating the Jackal's location in map in real time. As to controlling, we would like to apply move_base as low-level controller and write our own package to utilize Gurobi to find a optimal trajectory with enforcing collision avoidance.

IV. System Evaluation:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. We use the Rviz to make sure that the obstacles we detect from the velodyne points cloud are good enough (the position and size of the obstacles are reasonable). First, move\base plan a global path to the goal. We can see the velodyne points cloud and move\base inflated the points with setting radius as obstacles on the Rviz. After getting detected obstacles, move\base plans a local path to avoid the obstacles.

Moreover, we also timed the speed of Jackal, which is 1.45m/s. In addition the Jackal runs at a speed of 0.7 meter per second stably and smoothly. It doesn't crash any obstacle and can arrive at the setting goal exactly.

V. Conclusion

The proposed project is about automatic driving. We would like to propose an algorithm which can lead Jackal drive itself. The overall system is divided into three parts: mapping, locating, and controlling. We plan to combine the state-of-art works to accomplish obstacles detection, simultaneously locating and mapping with sensor fusion, as well as model predictive control via mix-integer programming. In addition we would also keep implementing the combination of low-level controller and high-level controller.

References

[1] https://github.com/tysik/obstacle_detector.
[2] D. Ferguson, M. Darms, C. Urmson, and S. Kolski, "Detection, prediction, and avoidance of dynamic obstacles in urban environments," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 1149–1154.
[3] J. Larson and M. Trivedi, "Lidar based off-road negative obstacle detection and analysis," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 192–197.
[4] C. Mertz, L. E. Navarro-Serment, R. MacLachlan, P. Rybski, A. Steinfeld, A. Suppe, C. Urmson, N. Vandapel, M. Hebert, C. Thorpe *et al.*, "Moving object detection with laser scanners," *Journal of Field Robotics*, vol. 30, no. 1, pp. 17–43, 2013.
[5] R. MacLachlan and C. Mertz, "Tracking of moving objects from a moving vehicle using a scanning laser rangefinder," in *2006 IEEE Intelligent Transportation Systems Conference*. IEEE, 2006, pp. 301–306.
[6] L. E. Navarro-Serment, C. Mertz, N. Vandapel, and M. Hebert, "Ladar-based pedestrian detection and tracking," *Robotics Institute*, p. 343, 2008.

[7] http://velodynelidar.com/vlp-16.html.
[8] http://wiki.ros.org/velodyne_pointcloud.
[9] http://wiki.ros.org/differential_drive.
[10] http://wiki.ros.org/xsens_driver.
[11] F. Caron, E. Duflos, D. Pomorski, and P. Vanheeghe, "Gps/imu data fusion using multisensor kalman filtering: introduction of contextual aspects," *Information fusion*, vol. 7, no. 2, pp. 221–230, 2006.
[12] A. Martinelli, "Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 44–60, 2012.
[13] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, May 2012.
[14] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surveys in Operations Research and Management Science*, vol. 17, no. 2, pp. 97 – 106, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1876735412000037
[15] A. Richards and J. How, "Mixed-integer programming for control," in *Proceedings of the 2005, American Control Conference, 2005.*, Jun. 2005, pp. 2676–2683 vol. 4, 00000.
[16] S. Sager, H. G. Bock, and M. Diehl, "The integer approximation error in mixed-integer optimal control," *Mathematical Programming*, vol. 133, no. 1-2, pp. 1–23, 2012.
[17] S. Sager, M. Claeys, and F. Messine, "Efficient upper and lower bounds for global mixed-integer optimal control," *Journal of Global Optimization*, vol. 61, no. 4, pp. 721–743, Jun. 2014.
[18] S. Sager, *Numerical methods for mixed-integer optimal control problems*. Der andere Verlag Tönning, Lübeck, Marburg, 2005.