S.O.S VR 2016 Team Description Paper for RoboCup 2016 Rescue Virtual Robot League

Mahdi Taherahmadi[1], Reyhaneh Pahlevan[2], Narges Sayyah, Sajad Azami

Department of Computer Engineering and Information Technology, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Hafez Ave, No. 424, Iran
14taher@gmail.com,
http://SOSVR.github.io

Abstract. This paper describes the current state of CEIT Department of Amirkabir University of Technologys submission to the 2016 RoboCup Rescue Virtual Robot competition with team S.O.S VR. We address our system architecture and software structure of the robot which based on ROS Framework. Along with the algorithms involved in the vital components of the system architecture. We also discuss high-level architectural interaction of the components, and provide relevant flow diagrams. ...

Keywords: Robotic, RoboCup, Virtual Robot Rescue, ROS, SLAM, Human Recognition, Multi Agent

1 Introduction

Maybe its time for humankind to take advantage of robots aid in tough situations such as earthquakes, floods, natural disasters and urban accidents. In these cases, even if smart and automatic robots save a single persons life, science has reached its goal. In the ahead challenge regardless of hardware implementation, we are focused on achieving Accurate perception of the (simulated) world from sensors input, make a right decision, plan considering the global goal, behave and perform the best. We aim to develop basic modules for the challenge which later can form a framework where they interact with each other. This document briefly explains these modules and the framework that we are prototyping to optimize them, with respect to the tasks in the challenge. Therefore,as discussed in the future of robot rescue simulation workshop, our team has designed a new structure based on ros framework and gazebo simulation environment that were going to explain its details in the following. We the S.O.S VR are participating in RoboCup Rescue Virtual Robot for the first time, yet our team has already been the winner of agent simulation league in two tournaments, also other team members have participated in real rescue, Humanoid, SPL leagues, earned achievements and experienced working with ROS framework beside other robotic challenges.

2 Background and Related Works

S.O.S team is well known for its great performance in rescue simulation league[1]. We have achieved lots of trophies in this league. In the last two years, we achieved the 1st place in the national robocup competition 2014 Brazil and 2015 china. We have achieved a very stable situation in our base codes by developing them for years by old team members and our main strategy has been shaped during these years in different competitions. This year, we -people with different proficiencies in Image processing, multi agent, robotics, ROS framework and Cognitive science- are gathered together in a new team and got participated in virtual robot rescue league to examine our knowledge and ability in this field and do more research and practical work.

3 System Architecture

We will use Kenaf, PA3T and The Air vehicle in the simulation. Our robots system directly performs on ROS and Gazebo, robots will interact with WCS (Wireless Communication Simulator) which is a model for local wireless lan written in ROS. The architecture designed to be modular and expansible besides algorithm improvements and future developments, also we have decided to publish our project on teams github repository[2].

4 Code Structure

4.1 Cognition

Robots cognition is about doing robotics that deals with cognitive phenomena such as perception, attention, anticipation, planning, memory, learning, and reasoning. This section of code consist of three main parts, Perception, Mapping and Localization and Human Recognition.

Perception Recognizing and perceiving the surroundings is the most important task that the robot should do after starting its activity in the (simulated) world. Regarding to the rules it is done via camera/sonar/laser range scanner sensors. In our idea, perception is receiving the environment data, processing them and extracting useful information. Output of these nodes are the input for other ROS nodes.

SLAM Simultaneous Localization and Mapping (SLAM) is one of the most critical challenges in Rescue Simulation. SLAM's objective is to explore the surrounding world, by creating maps, finding itself in them and planning to navigate through which path, until we find a victim, then we will decide what to do next.

for these purpose we used this combination of this packages from ROS Navigation Stack.

1. G-Mapping
   For Mapping, we have used G-Mapping package of "ROS". G-Mapping is a ROS wrapper for OpenSlam's Gmapping.[3] Using slam_gmapping, you can create a 2-D occupancy grid map (like a building floorplan) with a highly efficient Rao-Blackwellized particle filer from laser range scanner and pose data collected by robot.
2. AMCL
   For Localization part, we are working to reach an optimized and fast solution, for this purpose, we are using amcl, that is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox)[4], which uses a particle filter to track the pose of a robot against a known map.
3. Hector
   hector_mapping[5] is a SLAM approach that can be used without odometry as well as on platforms that exhibit roll/pitch motion (of the sensor, the platform or both). It leverages the high update rate of modern LIDAR systems like the Hokuyo UTM-30LX and provides 2D pose estimates at scan rate of the sensors (40Hz for the UTM-30LX). While the system does not provide explicit loop closing ability, it is sufficiently accurate for many real world scenarios. The system has successfully been used on Unmanned Ground Robots, Unmanned Surface Vehicles, Handheld Mapping Devices and logged data from quadrotor UAVs.

Human Recognition the most critical task of this league is human detection. Without detecting victims and localizing them, our other efforts will be quite inconclusive. The problem is detecting human victims using monocular moving camera. We use OpenCV HOG pedestrian detector[6] and HAAR face detector[7] OpenCV Implementation in ROS for this purpose. This package contains several packages for detecting and identifying faces, and detecting standing pedestrians in the image.

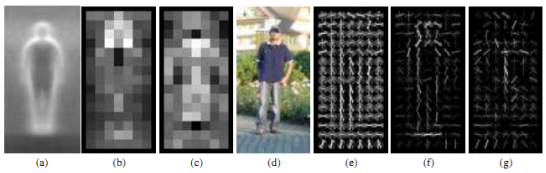

Fig. 1. our HOG detectors cue mainly on silhouette contours (especially the head, shoulders and feet). The most active blocks are centered on the image background just outside the contour.

(a) The average gradient image over the training examples. (b) each "pixel" shows the maximum positive SVM weight in the block centered on pixel. (c) Likewise for the negative SVM weights. (d) test image. (e) It's computed R-HOG descriptor. (f,g) The R-HOG descriptor weighted by respectively the positive and negative SVM weights.[12]

This method with an edge direction histogram, scale invariant feature transform and shape context method has a lot of similarities. But they differ are: HOG calculations are based on density matrix uniform space to improve accuracy. That is: in a dense grid of uniformly sized cell unit basis, and in order to improve performance, but also the use of overlapping local contrast normalization techniques .

HOG features and SVM classifier in combination, have been widely used in image recognition, especially in pedestrian detection.[13]

HOG feature extraction methods: the image is first divided into small regional connectivity, we call it cell unit. Then direction histogram acquisition unit cell in each pixel gradients or edges. Finally, the combination of these together can form a histogram feature descriptor.

The R-HOG lt-HOG: lt-section substantially the HOG few square lattice, it can be characterized by three parameters: the number of cells per unit interval, the number of pixels in each unit cell, each cell histogram channel number. R-HOG descriptors look very similar, but its difference is: R-HOG under a single scale, lower inner dense grid, there is no sort of direction the situation is calculated.

4.2 Behavior

A behavior is anything your robot does: turning on a single motor is a behavior, moving forward is a behavior, tracking a line is a behavior, navigating a maze is a behavior.

SMACH We use the SMACH state machine system for our main Behavioral reaction system. Each process is managed from a main node which handles robots status. SMACH is a task-level architecture for rapidly creating complex robot behaviors[8]. At its core, SMACH is a ROS-independent Python library to build hierarchical state machines.

World Exploration Contains nodes taking robot_state and grid_map and deciding which route leads to overall faster and optimized world exploration.

- Path Planning and Navigation
  Using a map, the planner creates a kinematic trajectory for the robot to get from a start to a goal location. Along the way, the planner creates, at least locally around the robot, a value function, represented as a grid map. This value function encodes the costs of traversing through the grid cells[9].

Multi-Agent We use a multi-agent system, a main agent whose task is to find the victim and the other one which we call Explorer agent We thought it is better to use an air robot as explorer agent because in addition to helping us to control the main agent manually, it can give us a good view from our main agent by sending us images from higher attitudes. The main strategy is for one group of robots (explorer) to scan the map as quick as possible and others to check the places more likely for victims to be there, a bit slower, in order to focus on detecting victims. We can extend this method by using more Explorer agents which can have much more tasks except forwarding. Of course, it is trade-off between this chaining method and cost (number of agents).

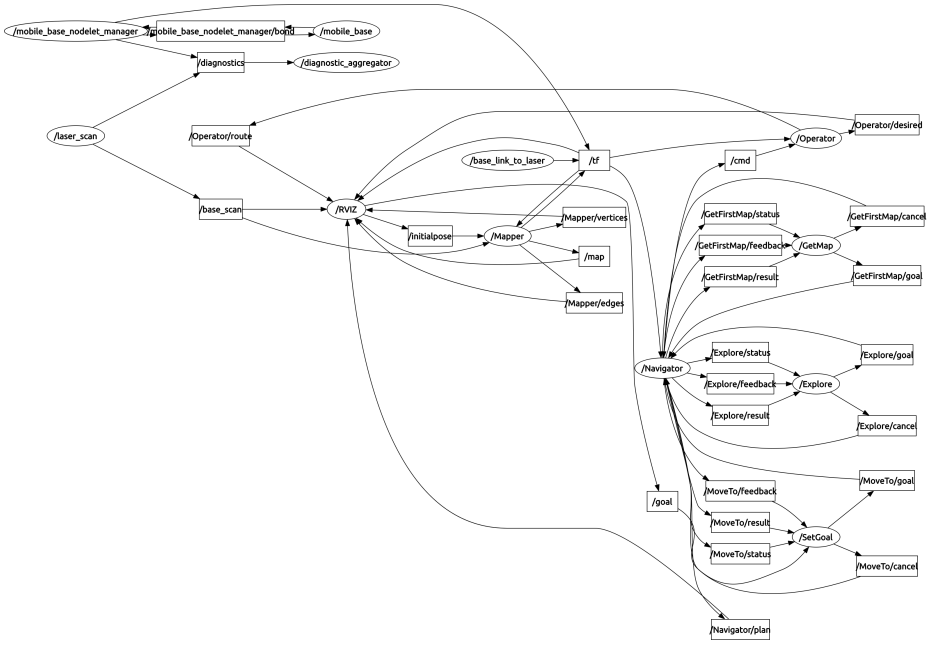

Fig. 2. ROS Node Computation Graph of our code structure

4.3 Motion

Move_base The move_base package which is a major component of the navigation stack 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[10] 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 that are used to accomplish navigation tasks.

Odometry Node which publishing Odometry information that gives the local planner the current speed of the robot. The velocity information in this message is assumed to be in the same coordinate frame as the robot_base_frame of the costmap[11] contained within the TrajectoryPlannerROS object

Controller The controller's job is to use this value function to determine dx, dy, dtheta velocities to send to the robot.

4.4 Utility

We also developed some utility softwares that focuses on helping running, Debugging, visualizing, configure settings and interact with controller more efficiently, beside using powerful ROS applications and features like Rviz and rqt.

Visualizer Dashboard is a Name for our Visualizer which is a plugin for rqt that allows us to see robot(s)_state, grid_map, topics published by specific nodes, state machine flow chart and current state, teleoperator and controller pane.

4.5 Wireless Communication

4.6 Bring Up System

ROS Launch Global bringup system made of roslaunch launch files for convenient startup the modules, packages and their dependencies together.

- roslaunch is a tool for easily launching multiple ROS nodes locally and remotely via SSH, as well as setting parameters on the Parameter Server. It includes options to automatically respawn processes that have already died. Roslaunch takes in one or more XML configuration files (with the .launch extension) that specify the parameters to set and nodes to launch, as well as the machines that they should be run on.

5 Conclusion and Future Work

We present an overview of our systems that will be built for the RoboCup Robot Rescue 2016. At the time of writing this document we are progressing in developing the basic modules. We will continue to advance our systems to compete in RoboCup, to complete our tasks. Our developments will be posted in our webpage and our source code will be released after competition on the team's github repository.

References

1. https://www.ida.liu.se/ TDDD10/papers/rsl-agents/rsl2013_submission_19.pdfArch. Rat.
2. S.O.S VR github repository https://github.com/SOSVR
3. http://openslam.org/gmapping.html
4. https://www.ri.cmu.edu/pub_files/pub1/fox_dieter_1999_1/fox_dieter_1999_1.pdf
5. http://www.sim.informatik.tu-darmstadt.de/publ/download/2013_RoboCup_Kohlbrecher_open_source_tools.pdf
6. http://docs.opencv.org/modules/gpu/doc/object_detection.html
7. http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html
8. https://www.willowgarage.com/sites/default/files/ICRA11_1089_FI.pdf
9. http://cs.mtu.edu/ jwwalker/files/cs5881.pdf
10. http://wiki.ros.org/base_local_planner
11. http://wiki.ros.org/costmap_2d
12. http://blog.csdn.net/songzitea/article/details/17025149
13. https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf