B User Manual

B.1 Installation Guidelines

1. ***(Pre-requisite)*** The system shall be installed with "ROS Noetic", and "Python version 3.7.6 or above", thereby meeting all their installation requirements (including any additional requirements by "Tensorflow version 2.8.0", if any). The recommended OS is Linux, Ubuntu 20.04, and Python 3.7.6.
2. Install the ROS perception package by following the given below instructions:
   (a) Setup a new catkin workspace by following the instructions provided at `http://wiki.ros.org/catkin/Tutorials/create_a_workspace`.
   (b) Create a new catkin package named "perception_pkg" with dependencies "std_msgs", "rospy", "roscpp", "message_runtime", and "message_generation" by following the instructions provided at `http://wiki.ros.org/ROS/Tutorials/CreatingPackage`.
   (c) Copy and replace all the files from the provided "perception_pkg" folder to the now-created "perception_pkg" folder.
   (d) Complete the creation of the catkin package "perception_pkg" by following the instructions provided in 2b, to build the package in the catkin workspace using "catkin_make" command.
   (e) Change the directory to the folder "perception_pkg/src/scripts/developer_scripts/src/", and run the script "make_codes_executable.py". This is to convert all the necessary scripts to executable scripts within the ROS package.
3. Install all the libraries mentioned in the "requirements.txt" for the ROS perception package and the object detection pipeline software. This can be achieved by following the given below instructions for both the software packages individually:
   (a) Change the directory path to the directory where "requirements.txt" is located.
   (b) If required, activate the virtual-environment.
   (c) Run the following command in a new shell/terminal:
   "pip3 install -r requirements.txt"

B.2 Functionalities

B.2.1 System Activation

The ROS "perception_pkg" is designed in a way that the system can be activated at three levels - system level (distribution center), unit level (vehicle, infrastructure), sensor level (LiDAR, camera, opti-track, odometry, IMU) according to the user-inputs. One can activate the whole system, or just

an unit of the system, or just a sensor of an unit of the system. This can be achieved by following the given below instructions with the appropriate "id" numbers for system, unit, and sensor, respectively.

Prior to the activation, the user-inputs have to be provided in the file: "perception_pkg/src/scripts/user_scripts/user_config/system_definition/dc_1_system_definition.yaml". Instructions for filling the user-inputs have been provided as comments in the file.

System-Level:

1. For "Distribution Center" activation of id number 'i', run the following command in a new shell/terminal:
   "rosrun perception_pkg distributioncenter_activation.py --dc_id 'i' ".

   (For instance, to activate the distribution center of id '1', the command is as follows:
   "rosrun perception_pkg distributioncenter_activation.py --dc_id 1 ").

Unit-Level:

1. For "Vehicle" activation of id number 'j' of distribution center of id number 'i', run the following command in a new shell/terminal:
   "rosrun perception_pkg vehicle_activation.py --dc_id 'i' --unit_type v --unit_id 'j' ".
2. For "Infrastructure" activation of id number 'j' of distribution center of id number 'i', run the following command in a new shell/terminal:
   "rosrun perception_pkg infrastructure_activation.py --dc_id 'i' --unit_type i --unit_id 'j' ".

Sensor-Level:

1. For "LiDAR" activation of id number 'k' of vehicle of id number 'j' of distribution center of id number 'i', run the following command in a new shell/terminal:
   "rosrun perception_pkg lidar_activation.py --dc_id 'i' --unit_type v --unit_id 'j' --lidar_id 'k' ".
2. For "Camera" activation of id number 'k' of vehicle of id number 'j' of distribution center of id number 'i', run the following command in a new shell/terminal:
   "rosrun perception_pkg camera_activation.py --dc_id 'i' --unit_type v --unit_id 'j' --camera_id 'k' ".
3. For "Odometry" activation of vehicle of id number 'j' of distribution center of id number 'i', run the following command in a new shell/terminal:
   "rosrun perception_pkg odometry_activation.py --dc_id 'i' --unit_type v --unit_id 'j' ".
4. For "IMU" activation of vehicle of id number 'j' of distribution center of id number 'i', run the following command in a new shell/terminal:
   "rosrun perception_pkg imu_activation.py --dc_id 'i' --unit_type v --unit_id 'j' ".
5. The "Opti-track" sensor is part of the infrastructure, but is activated for vehicles individually. For "Opti-track" activation of vehicle of id number 'j' of distribution center of id number 'i',

run the following command in a new shell/terminal:
"rosrun perception_pkg optitrack_activation.py --dc_id 'i' --unit_type v --unit_id 'j' ".

Please note that if the LiDAR/camera is a part of the infrastructure, then change the argument "unit_type v" to "unit_type i".

B.2.2 Obstacle Detection

Prior to the activation of obstacle detection and collision alert, the user-inputs have to be provided in the file:
"perception_pkg/src/scripts/user_scripts/user_config/features/obstacle_detection/obstacle_detection.yaml".
Instructions for filling the user-inputs have been provided as comments in the file.

1. Activate the LiDAR-of-interest of id number 'k' either as part of the system-level activation or unit-level activation or sensor-level activation, as described in the section B.2.1.
2. For activation of obstacle detection of "LiDAR" id number 'k' of vehicle of id number 'j' of distribution center of id number 'i', run the following command in a new shell/terminal:
   "rosrun perception_pkg lidar_obstacle_detection_activation.py --dc_id 'i' --unit_type v --unit_id 'j' --lidar_id 'k' ".
3. For activation of the "collision alert" feature of the same LiDAR, after step 2 run the following command in a new shell/terminal:
   "rosrun perception_pkg lidar_collision_alert_activation.py --dc_id 'i' --unit_type v --unit_id 'j' --lidar_id 'k' ".

B.2.3 Object Detection

Prior to the activation of object detection, the user-inputs have to be provided in the file:
"perception_pkg/src/scripts/user_scripts/user_config/features/object_detection/object_detection.yaml".
Instructions for filling the user-inputs have been provided as comments in the file.

1. Follow section B.2.2 to activate obstacle detection of "LiDAR" id number 'k' of vehicle of id number 'j' of distribution center of id number 'i'.
2. For activation of the "object detection", after step 1 run the following command in a new shell/terminal:
   "rosrun perception_pkg lidar_object_detection_activation.py --dc_id 'i' --unit_type v --unit_id 'j' --lidar_id 'k' ".

B.2.4 Data Collection

Prior to the activation of data collection, the user-inputs have to be provided in the file:
"perception_pkg/src/scripts/user_scripts/user_config/data_collection/datacollection.yaml". Instructions for filling the user-inputs have been provided as comments in the file.

1. For activation of the “data collection” of distribution center of id number ’i’, run the following command in a new shell/terminal:
   “rosrun perception_pkg data_collector.py −−dc_id ’i’”.
2. The collected data shall be stored as a ROS bag file in the folder location provided as input by the user in the “datacollection.yaml” file.

B.3 Additional Functionalities

This section provides instructions to capture object-shape data, utilize them to generate training data, and train the neural network model for object detection.

This section falls into the “developer view”, and requires a basic understanding of python programming language to be executed.

B.3.1 Data Capture

1. To capture the obstacle-pattern of an object of name ’NAME’, place the LiDAR sensor at different poses around (but facing) the object-of-interest.
2. At each pose (or capture number ’m’) activate the “data capture” feature of “LiDAR” id number ’k’ of vehicle of id number ’j’ of distribution center of id number ’i’ by running the following command in a new shell/terminal:
   “rosrun perception_pkg data_capture_at_a_pose.py −−dc_id ’i’
   −−unit_type v −−unit_id ’j’ −−lidar_id ’k’ −−obj_name ’NAME’ −−cap_num ’m’”.

   For instance, to take the fourth capture of an object named ’Semitrailer’ using the LiDAR id number ’1’ of vehicle of id number ’2’ of distribution center of id number ’1’, the command shall look like:
   “rosrun perception_pkg data_capture_at_a_pose.py −−dc_id 1
   −−unit_type v −−unit_id 2 −−lidar_id 1 −−obj_name Semitrailer −−cap_num 4”.

   Repeat this step for an user-desired number of captures, say ’M’. In addition, please note down the approximate distance ’d’ of the object’s location from the LiDAR sensor (For instance, d=1.5m).
3. After capturing the total ’M’ number of captures of the object named ’NAME’, run the following command to process the captured data:
   “rosrun perception_pkg data_read_bag_and_update_captures.py −−dc_id ’i’
   −−unit_type v −−unit_id ’j’ −−lidar_id ’k’ −−obj_name ’NAME’ −−tot_cap ’M’”.

   In the instance here (say total number of captures M=10), the command shall look like:
   “rosrun perception_pkg data_read_bag_and_update_captures.py −−dc_id 1
   −−unit_type v −−unit_id 2 −−lidar_id 1 −−obj_name Semitrailer −−tot_cap 10”.
4. The processed data shall be stored as “’object_name’.csv” files in the folder: “perception_pkg/src/scripts/data/data_captured_for_training_data/processed_data_files”

B.3.2 Training Data Generation

1. Copy all the files from the processed data folder of step 4 of section B.3.1, and paste them in the folder:

   "Object_Detection_Training_Pipeline/Data_Generation/scene_details
   /lidar_capture_data_to_generate_map"

2. Run the following script to create map formats (intermediate step-1):

   "Object_Detection_Training_Pipeline/Data_Generation/scene_details/**create_map_formats.py**"
   The distance noted in the section B.3.1 could be given as input for the "rad_of_interest", if required.
3. Run the following script to pre-process the created map formats for scene simulation (intermediate step-2):

   "Object_Detection_Training_Pipeline/Data_Generation/scene_details/**map_preprocess.py**"
4. Run the following script to generate the training data (user-inputs can be provided under the script-section "USER CHOICES"):

   "Object_Detection_Training_Pipeline/Data_Generation/**generate_data.py**"
5. The generated data shall be stored in the folder:

   "Object_Detection_Training_Pipeline/Data_Generation/generated_data"

B.3.3 Training the Neural Network

1. Copy all the class names of the current object detection feature of the ROS perception package from the file:

   "perception_pkg/src/scripts/developer_scripts/src/features/object_detection/1_sensor/lidar/
   trucklab_object_detection/**trucklab_classes.names**"

   and paste them into the file in the object detection pipeline:

   "Object_Detection_Training_Pipeline/TrainModel_TruckLab/training_data/Dataset_names_old.txt"
   (This is done to enable transfer training from the previous neural network model).
2. Copy all the contents from:

   "Object_Detection_Training_Pipeline/Data_Generation/**generated_data**"

   ('Dataset_test', 'Dataset_train', 'Dataset_test.txt', 'Dataset_test.txt', 'Dataset_names.txt') from section B.3.2, and paste them in the folder:

   "Object_Detection_Training_Pipeline/TrainModel_TruckLab/**training_data**"

3. Provide the hyper-parameters for the training pipeline in the configuration file:

   "Object_Detection_Training_Pipeline/TrainModel_TruckLab/yolov3_files/**configs.py**"

4. Run the following script to train the neural network model:

   "Object_Detection_Training_Pipeline/TrainModel_TruckLab/**train.py**"

   Please note that the training process can take a very long time depending on the training data and hyper-parameters.

5. After training is done, copy all the files from the folder:

   "Object_Detection_Training_Pipeline/TrainModel_TruckLab/**model_saved_after_training**"

   to the ROS perception package's folder:

   "perception_pkg/src/scripts/data/model_data/**model_current**"

6. Finally, copy all the class names of the trained neural network from the file:

   "Object_Detection_Training_Pipeline/TrainModel_TruckLab/training_data/**Dataset_names.txt**"

   and paste them into the following file in the ROS perception package:

   "perception_pkg/src/scripts/developer_scripts/src/features/object_detection/1_sensor/lidar/trucklab_object_detection/**trucklab_classes.names**"