CATBot Localization

Part 1: Dead-Reckoning

Mahmoud Abdul Galil

Tutorial-6, Wednesday October 26, 2016

CATBot Encoder

- In order to perform dead-reckoning using the encoder, we need to measure the angular velocities of the wheels periodically, and have the dimensions concerning the geometry of the diff drive robot (Wheel separation and Wheel diameter)
- To simulate that in Gazebo, we will use the information published by gazebo-ros-control regarding the published information of the joints that we attached controllers to. This information is published under the name space */catbot* on the topic /joint_states

CATBot Encoder

- We will write a node that subscribes to */catbot/joint_states* and publishes Pose information on a topic with a name of our choice (let's call it */catbot/odometry_pose*)
- Message type of */catbot/joint_states* is sensor_msgs/JointState
- Message type of */catbot/odometry_pose* is geometry_msgs/PoseStamped

PublisherSubscriber.h

```
#ifndef PUBLISHER_SUBSCRIBER_H
#define PUBLISHER_SUBSCRIBER_H

#include <ros/ros.h>
#include <string>
#include <tf/transform_broadcaster.h>

template<typename PublishT,typename SubscribeT>
class PublisherSubscriber
{
public:
  PublisherSubscriber() {}
  PublisherSubscriber(std::string publishTopicName, std::string subscribeTopicName, int queueSize)
  {
    publisherObject  = nH.advertise<PublishT>(publishTopicName,queueSize);
    subscriberObject = nH.subscribe<SubscribeT>(subscribeTopicName,queueSize,&PublisherSubscriber::subscriberCallback,this);
  }
  void subscriberCallback(const typename SubscribeT::ConstPtr& recievedMsg);

protected:

  ros::Subscriber subscriberObject;
  ros::Publisher  publisherObject;
  ros::NodeHandle nH;
  tf::TransformBroadcaster tf_br;
};

#endif
```

Ground-Truth Pose

- If we want to generate ground truth information to compare the encoder output, we need to extract these information from the simulation
- Gazebo publishes there information on a topic named */gazebo/model_states*
- Message type of */gazebo/model_states* is gazebo_msgs/ModelStates
- We'll write a node that extracts information regarding catbot model state and republishes them as a Pose message on a separate topic.
- Let's call it /catbot/ground_truth_pose

ModelStateToPoseStamped.h

```
#ifndef MODELSTATETOPOSESTAMPED_H
#define MODELSTATETOPOSESTAMPED_H

#include "PublisherSubscriber.h"
#include <ros/ros.h>
#include <geometry_msgs/PoseStamped.h>
#include <gazebo_msgs/ModelStates.h>

template<>
PublisherSubscriber<geometry_msgs::PoseStamped,gazebo_msgs::ModelStates>::PublisherSubscriber() {}

class PoseGenerator : protected PublisherSubscriber<geometry_msgs::PoseStamped,gazebo_msgs::ModelStates>
{
public:
  PoseGenerator(std::string publishTopicName, std::string subscribeTopicName, int queueSize, std::string frame_name)
  {
    publisherObject  = nH.advertise<geometry_msgs::PoseStamped>(publishTopicName,queueSize);
    subscriberObject = nH.subscribe<gazebo_msgs::ModelStates>(subscribeTopicName,queueSize,&PoseGenerator::subscriberCallback,this);
    frame_id         = frame_name;
  }

  void subscriberCallback(const gazebo_msgs::ModelStates::ConstPtr& receivedMsg)
  {
    geometry_msgs::PoseStamped poseMsg;

    for(size_t i = 0; i < receivedMsg->name.size(); ++i)
    {
      if (receivedMsg->name[i]=="catbot")
      {
        poseMsg.pose  = receivedMsg->pose [i];
      }
    }

    poseMsg.header.frame_id    = frame_id;
    poseMsg.header.stamp       = ros::Time::now();

    publisherObject.publish(poseMsg);
  }
protected:
  std::string frame_id;
};


#endif // MODELSTATETOPOSESTAMPED_H
```

PoseStampedToPath.h

- Next, we need to collect the stamped poses in a path message in order to be able to display it in Rviz.
- We'll write a node that subcribes to /catbot/ground_truth_pose and publishes to /catbot/ground_truth_path
- The node will collect pose information, save them in a container with a specified size, and updates them whenever a new pose is published on the /catbot/ground_truth_pose

PoseStampedToPath.h

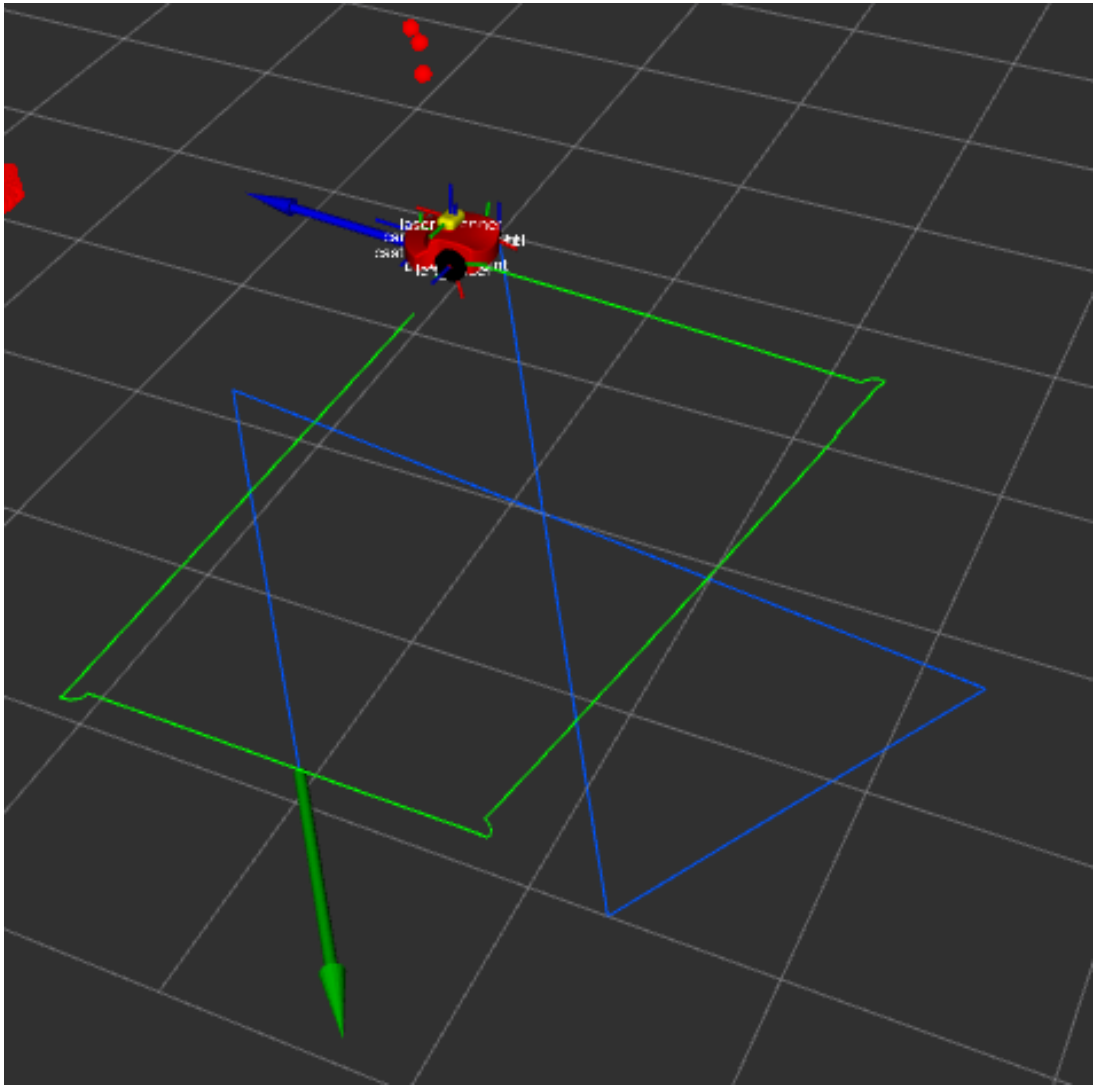
```
#ifndef POSETOPATH_H
#define POSETOPATH_H
#include "PublisherSubscriber.h"
#include <ros/ros.h>
#include <geometry_msgs/PoseStamped.h>
#include <nav_msgs/Path.h>

template<>
PublisherSubscriber<nav_msgs::Path,geometry_msgs::PoseStamped>::PublisherSubscriber() {}

class PathGenerator : protected PublisherSubscriber<nav_msgs::Path,geometry_msgs::PoseStamped>
{
public:
  PathGenerator(std::string publishTopicName, std::string subscribeTopicName, int queueSize, std::string frame_name)
  {
    publisherObject  = nH.advertise<nav_msgs::Path>(publishTopicName,queueSize);
    subscriberObject = nH.subscribe<geometry_msgs::PoseStamped>(subscribeTopicName,queueSize,&PathGenerator::subscriberCallback,this);
    frame_id         = frame_name;
  }

  void subscriberCallback(const geometry_msgs::PoseStamped::ConstPtr& receivedMsg)
  {
    geometry_msgs::PoseStamped poseMsg;

    poseMsg.pose            = receivedMsg->pose;
    poseMsg.header          = receivedMsg->header;

    const geometry_msgs::PoseStamped constPoseMsg = poseMsg;
    if (pathMsg.poses.size() < 5000)
    {
      pathMsg.poses.push_back(constPoseMsg);
    }
    else
    {
      pathMsg.poses.erase(pathMsg.poses.begin());
      pathMsg.poses.push_back(constPoseMsg);
    }

    pathMsg.header.frame_id = frame_id;
    publisherObject.publish(pathMsg);
  }
protected:
  nav_msgs::Path pathMsg;
  std::string frame_id;
};

#endif // POSETOPATH_H
```

Launch File

```
<launch>
        <arg name="rviz_gui" default="False" />
        <arg name="paused" default="false"/>
        <arg name="use_sim_time" default="true"/>
        <arg name="gazebo_gui" default="true"/>
        <arg name="headless" default="false"/>
        <arg name="debug" default="false"/>

        <param name="robot_description" textfile="$(find catbot_description)/urdf/diff_catbot_perception.urdf"/>
        <param name="use_gui" value="$(arg rviz_gui)" />

        <rosparam file="$(find catbot_control)/config/diff_catbot_kinematics.yaml" command="load"/>
        <include file="$(find catbot_localization)/launch/catbot_world.launch" />

        <node name="spawn_model" pkg="gazebo_ros" type="spawn_model" args="-param robot_description -urdf -model catbot -z 0.13" output="screen" />
        <node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false" output="screen" ns="/catbot"
            args= "left_motor_controller right_motor_controller joint_state_controller"/>
        <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" respawn="false" output="screen">
                <remap from="/joint_states" to="/catbot/joint_states" />
        </node>
        <node name="rviz" pkg="rviz" type="rviz" args="-d $(find catbot_localization)/rviz/encoder_world_config.rviz" />
        <node name="kinematic_model_teleop" pkg="catbot_control" type="kinematic_model" output="screen"/>
        <node name="ground_truth" pkg="catbot_localization" type="ground_truth" output="screen" />
        <node name="ground_truth_pose_generator" pkg="catbot_localization" type = "ground_truth_pose_generator" />
        <node name="ground_truth_path_generator" pkg="catbot_localization" type = "ground_truth_path_generator" />
        <node name="odometry_pose_generator" pkg="catbot_localization" type = "odometry_pose_generator" />
        <node name="odometry_path_generator" pkg="catbot_localization" type = "odometry_path_generator" />
</launch>
```

Ground-Truth vs Encoder

- Blue line is Encoder
- Green line is ground-truth
- Huge difference due to huge model errors



References

1) http://wiki.ros.org/tf
2) http://wiki.ros.org/roslaunch/XML
3) https://www.youtube.com/watch?v=2gVo06HR2Tc
4) https://www.youtube.com/watch?v=g9WHxOpAUns
5) https://www.youtube.com/watch?v=W0aoAm6eYSk
6) https://www.youtube.com/watch?v=U2QvTsMvWmM