Sapienza Università di Roma

Dipartimento di Informatica e Sistemistica Antonio Ruberti

Homework : Laser Mapping

Giorgio Grisetti

Tiziano Guadagnino

Exercise

- Fill the laser callback in mapper.cpp, which have to :
  - Get the transformation between the laser frame and the odometry frame
  - Extract the point cloud from the laser scan message
  - Transform the points in the odometry frame (hints in the code)
  - Paint the corresponding pixels on the canvas

Homework

- Create a new package which color the pixel for both the points perceived by the scanner and the positions of the robot.
- You can use/modify the classes used in the exercise

Canvas API

```
#pragma once
#include <opencv2/opencv.hpp>
#include <iostream>
#include <Eigen/Dense>

namespace utils{

  class Canvas{
  public:
    Canvas(const size_t& rows_, const size_t& cols_, const float& resolution_);
    ~Canvas();
    void resize(const size_t& rows_, const size_t& cols_);
    void colorPoint(const Eigen::Vector2f& point_, const cv::Vec3b& color=cv::Vec3b(0,0,0));
    inline void show(){
      cv::imshow("map",*_img);
      cv::waitKey(0.1);
    }
  protected:
    cv::Mat* _img;
    float _resolution;
  };

}
```

Mapper API

```
#pragma once
#include "ros/ros.h"
#include "sensor_msgs/LaserScan.h"
#include "tf/transform_listener.h"
#include "geometry_utils_fd.h"
#include "canvas.h"

class LaserMapper{
public:
  LaserMapper(ros::NodeHandle& nh_);
  ~LaserMapper();
  inline void setOdomFrameId(const std::string& odom_frame_id_){_odom_frame_id=odom_frame_id_;}
  inline void setLaserTopic(const std::string& laser_topic_){_laser_topic=laser_topic_;}
  inline void setCanvas(utils::Canvas& canvas_){_canvas=&canvas_;}
  void laserCallback(const sensor_msgs::LaserScan::ConstPtr& msg_);
  void subscribe();
protected:
  std::string _odom_frame_id;
  std::string _laser_topic;
  utils::Canvas* _canvas;
  tf::TransformListener* _listener;
  ros::NodeHandle& _nh;
  ros::Subscriber _laser_sub;
};
```

Convert raw scan to point cloud

sensor_msgs/LaserScan

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

$$p = \begin{bmatrix} ranges[i]\cos(angle) \\ ranges[i]\sin(angle) \end{bmatrix}$$

$$angle = angle\_min + i \cdot angle\_increment$$