BRNO UNIVERSITY OF TECHNOLOGY

LGV report

Data Bridge, Path Planning, Decision Making

Jaroslav Rozman
Radim Luza

2013

Report

1 Introduction

This software package contains two independent components: the Path Planner and the Decision Maker. Both components are implemented as ROS nodes. The Path Planner component provides a proper path that avoids obstacles according to particular avoiding path request. The Decision Maker component is capable of making decisions about robot behaviour. The Decision Maker follows some global goal like reaching given destination or go somewhere, grasp something and return back and it is able to react to unexpected obstacles in robot's path. By unexpected obstacle it is meant another robot, static object like box laying in the path or person walking into the robot's path. The Decision Maker can decide if the obstacle should be avoided, if the robot should sound a horn or if the robot should take another action.

2 Path Planning

The aim of this node is to find avoiding path for the robot. There are two possible algorithms for using in this node - first is based on the potential fields and second on the Bug II algorithm. They are wrapped by code, that is providing interface between them and ROS. This means that there can be used more algorithm for path planning. They just have to keep the input and output that is provided by ROS interface.
This interface is described in detail later in this document in the chapter called Interface . Namely it needs

- original path (from topic *data_bridge_trajectory_in*)
- current robot position (from topic *data_bridge_abs_position*)
- positions and bounding boxes of obstacles (from topic *data_bridge_objdet*)
- map of the surroundings of the robot (from topic *map*, provided by gmapping)

The node provides

- avoiding path with some additional info (published to the topic *data_bridge_trajectory_out*)
- avoiding path for visualization in the RVIZ (published to the topic *SafePathToShow*)

The node is started by sending the request to the *data_bridge_trajectory_in* topic. From the IDs in this topic we can find positions and bounding boxes of the obstacles (these are in the *data_bridge_objdet* topic). These bounding boxes are placed in the map from the *map* topic. Starting point is the current position of the robot and the goal point is computed on the original path on the certain distance from the current position of the robot. Since it is not safe to send robot to the unknown space (behind the obstacle), only a portion of the planned path is sended. Sending is done through *data_bridge_trajectory_out* topic that provides also some extra information. Data from this topic can not be visualized in the RVIZ, so the path of type *nav_msgs::Path* is also sended to the *SafePathToShow* topic. After sending safe avoiding path the node waits for another request.

2.1 Potential Field Path Planning

Potential field path planning is well known method for robotic path planning. Here the harmonic potential field is used.

The grid for computation is obtained from the topic *map*, or it may be created artificially, as we can assume, that the surroundings of the robot is empty, except of the obstacles. In the case of obtaining from the *map*, the values of +100 (obstacle) and -1 (unknown space) are internally converted to 0.0 and values 0.0 (free space) to 0.00001. The dimensions of the grid are also loaded from the *map* or in the case of artificial creation, it has to

be set as parameters, including the resolution. The obstacles are then enlarged by certain size (should be the radius of the robot) and the path is computed as maxima in the eight surrounding cells until value 1.0 (goal) is reached.
To make the path smoother, the angle is computed as average of five surrounding values.

Parameters to set

- Distance of the goal on the original path from the current (place where robot stops) robot position. The distance should not be too short because of safety, has to be inside the grid for potential field computing.
- Length of the sended part of the path. The destination point of the path should be placed to the point, that is visible from the place, where robot stops. The length depends on the distance of the robot from the obstacle - for example if this distance is 2 meters, this parameter should be e.g. 3 - 4 meters.
- Enlargement of the obstacles. Value of this parameter should be at least diameter of the circumscribed circle of the robot.
- Width, height and resolution of the artificial map (dimensions are in the mm or mm/pix). This should be large enough to succesfully plan the path. For the robot of size 2x4m and obstacle of the same size it should be at least 17m (17 000mm) long with the appropriate width. The recommended resolution is from 50mm to 200-250mm. The smaller the resolution, the denser the sampling of the path will be, but also the computional time will be higher.
- Number of iterations of the potential field path planning. This parameter should be at least the number of cells in the diagonal of the map. More iterations means better path but also more computational time.

3 Decision Making

This component is used for making decisions about obstacle avoidance and also it controlls robot behaviour during traveling from one point to another. It is based on the SMACH, which is ROS implementation of State MACHine. Decision making based on SMACH consists typically of two parts, first is the state machine itself, this part is written in Python. Second part is the implementation of particular states - this part is done by ROS Actionlib written in C++ and some of them are written in the Python). The design of the state machine consists of 7 states. They are

- WAITFORGOAL - robot is waiting for the coordination of the goal set by user (or by higher level controller).
- PATHPLANNING - computes the path from a current robot position to the goal position given by the user′s request.
- RUN - following the planned path.
- AVOIDANCEDECISION - deciding if the obstacle should be avoided and how critical is the situation, decisions are made according to the table.
- AVOIDANCEPLANNING - making the plan for avoiding the obstacle - calculating a new path.
- AVOID - avoiding the obstacle - this state is similar to the RUN.
- WAIT - waiting for obstacle to leave the robot′s way - for example waiting for a person to walk away.

There can also be the eighth state, CALLHELP, for cases, where no avoiding path can be computed. Once we have all states created, it is extremely easy to change connections between states according to our needs.

AVOIDANCEDECISION is one of the states implemented in Python. It implements table that determines the behaviour of the robot in case it detects various kinds of obstacles in the warning or danger zone.

4 Interface

This chapter describes communication interfaces of Path Planner and Decision Maker components. There are described inputs, outputs and message formats used in communication with these components.

4.1 Path Planner

The message formats used for communication are partially taken from another component - the Data Bridge. This software package does not include the Data Bridge itself, but it includes an interface of the Data Bridge.

Inputs:
ROS topics:

- map (not in the Data Bridge interface)
  - DESCRIPTION: Standard topic from gmapping node.
  - MESSAGE TYPE: nav_msgs/OccupancyGrid
  - MESSAGE FREQUENCY: 1Hz.
- data_bridge_trajectory_in
  - DESCRIPTION: Request for computation of path avoiding given obstacles - sent by robot's decision making system.
  - MESSAGE TYPE: data_bridge/DTB_Trajectory
  - MESSAGE FREQUENCY: once - request.
- data_bridge_abs_position
  - DESCRIPTION: Input of an absolute position of the robot.
  - MESSAGE TYPE: data_bridge/DTB_AbsPosition
  - MESSAGE FREQUENCY: 5Hz minimum.
- data_bridge_objdet
  - DESCRIPTION: Report of objects in robot's surroundings detected by sensors of robot.
  - MESSAGE TYPE: data_bridge/DTB_ObjReport
  - MESSAGE FREQUENCY: 30Hz by specification, 5Hz minimum.

Outputs:
ROS topics:

- data_bridge_trajectory_out
  - DESCRIPTION: Response from Path planner to robot's decision making system - extended form with additional information.
  - MESSAGE TYPE: data_bridge/DTB_Trajectory
  - MESSAGE FREQUENCY: once - response.
- SafePathToShow
  - DESCRIPTION: Response from Path Planner to robot's decision making system - standard ROS form.

  – MESSAGE TYPE: nav_msgs/Path
  – MESSAGE FREQUENCY: once - response.

Message types:
This is description of message types used that differ from the standard message types available in ROS. Standard message types include nav_msgs/OccupancyGrid and nav_msgs/Path.

- data_bridge/DTB_AbsPosition
  This message type describes absolute position and orientation of the robot in 2D plane of floor. It contains common header and geometry_msgs/Pose2D structure describing position of the robot on the plane of floor - x,y and orientation of the robot - theta (rotation about Z axis).

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose2D abs_position
  float64 x
  float64 y
  float64 theta
```

- data_bridge/DTB_Trajectory
  This message type describes an extended trajectory. The trajectory itself is described by array of geometry_msgs/Pose2D points. Each point represents position of robot - x,y on the floor plane and orientation of the robot - theta (rotation about Z axis). Furhter the mesage contains an array of object IDs that are being avoided or that needs to be avoided according to particular use of the message. The message also contains mode and proc_mode values. The mode item may contain values: 'r' - request for path avoidance, 'a' - avoidance path - succesfull response or 'f' - failure of finding the avoidance path. The procedure_mode describes current state of avoiding obstacles. Value '0' is the default state, value '1' says that the robot is leaving its original path, value '2' means that robot is in avoidance mode - it is avoiding obstacles and value '3' means that robot is returning back to the original path and avoiding maneuver is over. The trajectory message also contains a common header.

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose2D[] traj_points
  float64 x
  float64 y
  float64 theta
int32[] object_IDs
int32 mode
int32 proc_mode
```

- data_bridge/DTB_ObjReport
  The object report message is an array of objects with common header. Each object is described by its ID - the m_id item, class meaning person, robot, box, other obstacle, etc. - the m_class item, zone in which the object occurs including safe zone (far enough), warning zone (close to robot) and dangerous (very close) - the m_zone item and also speed class saying if the object is static, slow moving or fast

moving - the m_speed_class item. The particular speed vector of object is described by m_speed three-dimensional vector. Another useful information is position and orientation of each object described by geometry_msgs/Pose item in m_pose_2D item of objects structure. Bouding box of object is described by four vertices of surrounding polygon in m_bb item. The rest of items of object structure contains the data about detection: a detection score - m_score, a size of bitmap covering the detected object in input image - image_size_x, image_size_y and bitmap mask of detected object in rectangular segment of image described of given size. The bitmap is described by m_mask.

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
data_bridge/DTB_Object[] objects
  int32 m_id
  int32 m_class
  int32 m_zone
  int32 m_speed_class
  float32 m_score
  geometry_msgs/Pose m_pose_2D
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
  geometry_msgs/Point[4] m_bb
    float64 x
    float64 y
    float64 z
  geometry_msgs/Point m_speed
    float64 x
    float64 y
    float64 z
  int32 image_size_x
  int32 image_size_y
  std_msgs/Bool[] m_mask
    bool data
```

4.2 Decision Maker

The Decision Maker component uses the interface of the Data Bridge. It receives report about object in robot surroundings and according to this report it decides about action consequentive action. The types of messages used in this section are the same as in subsection above called Path Planner.

Inputs:
ROS topics:

- data_bridge_trajectory_out

- DESCRIPTION: Response from robot's path planning system. Response may contain path that avoids given obstacles or it may contain only information about failure of finding the path.
- MESSAGE TYPE: data_bridge/DTB_Trajectory
- MESSAGE FREQUENCY: once - response.

- data_bridge_objdet
  - DESCRIPTION: Report of objects in robot's surroundings detected by sensors of robot.
  - MESSAGE TYPE: data_bridge/DTB_ObjReport
  - MESSAGE FREQUENCY: 30Hz by specification, 5Hz minimum.

Outputs:
ROS topics:

- data_bridge_trajectory_in
  - DESCRIPTION: Request for a new path that avoids given obstacles. Request is sent to robot's path planning system and after sending it Decision Maker waits until response arrives.
  - MESSAGE TYPE: data_bridge/DTB_Trajectory
  - MESSAGE FREQUENCY: once - request.
- cmd_vel
  - DESCRIPTION: Output message for velocity driven base of robot. This message type is used with Turtle Bot for example.
  - MESSAGE TYPE: geometry_msgs/Twist
  - MESSAGE FREQUENCY: 30Hz minimum.

4.3 Time diagram of communication

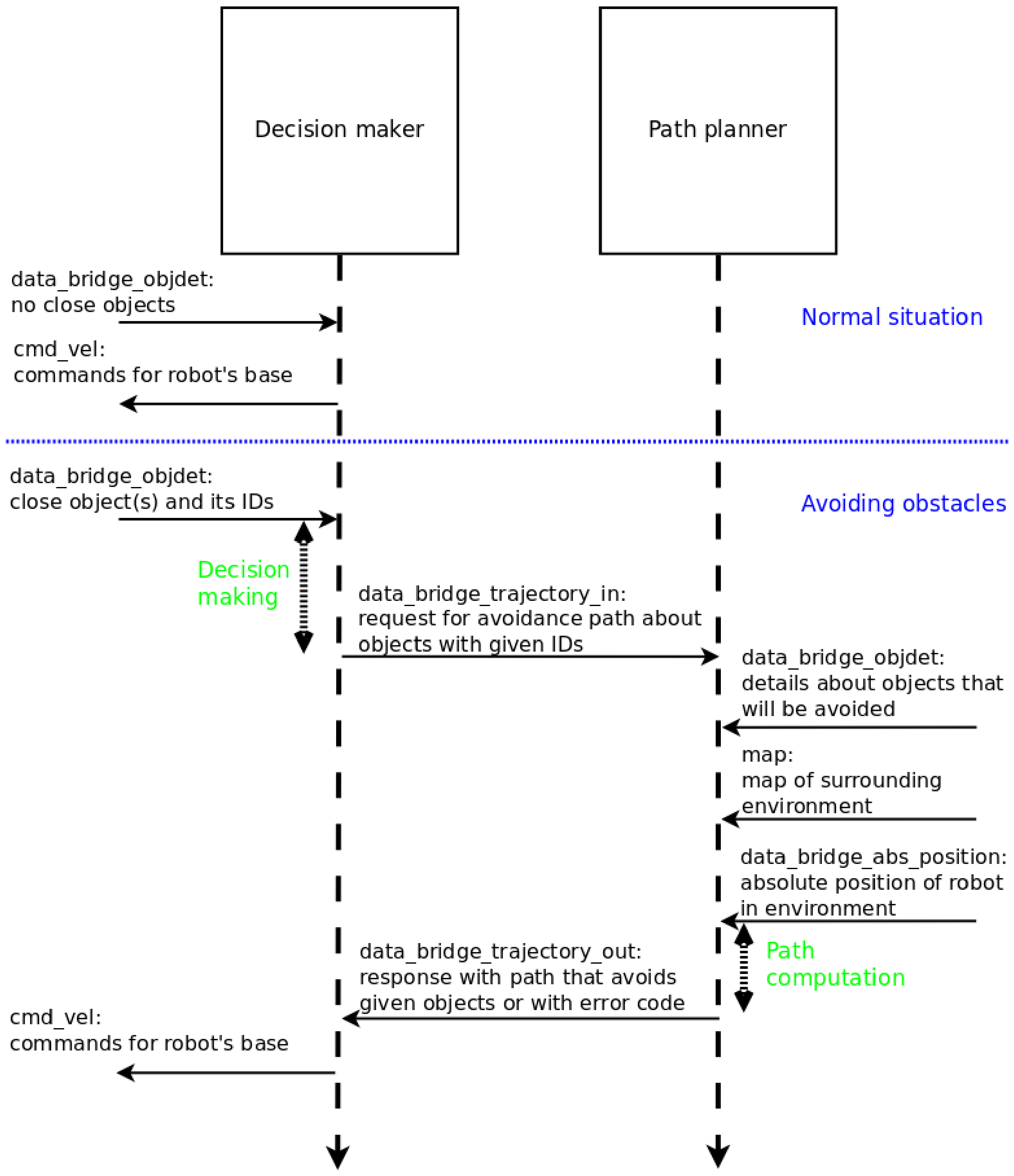
Decision maker
Path planner
data_bridge_objdet:
no close objects
Normal situation
cmd_vel:
commands for robot's base
data_bridge_objdet:
close object(s) and its IDs
Avoiding obstacles
Decision
making
data_bridge_trajectory_in:
request for avoidance path about
objects with given IDs
data_bridge_objdet:
details about objects that
will be avoided
map:
map of surrounding
environment
data_bridge_abs_position:
absolute position of robot
in environment
data_bridge_trajectory_out:
response with path that avoids
given objects or with error code
Path
computation
cmd_vel:
commands for robot's base

Figure 1: Time diagram of communication flow between Decision Maker, Path Planner and the rest of robot control system.