Planning and control of EduMIP among obstacles

Site Hu，Han Shi

May 14, 2018

1. Introduction

EduMIP is a self-balancing robot built around the BeagleBone Blue embeded microprocessor board. It can be treated as a unicycle (see Fig.1).

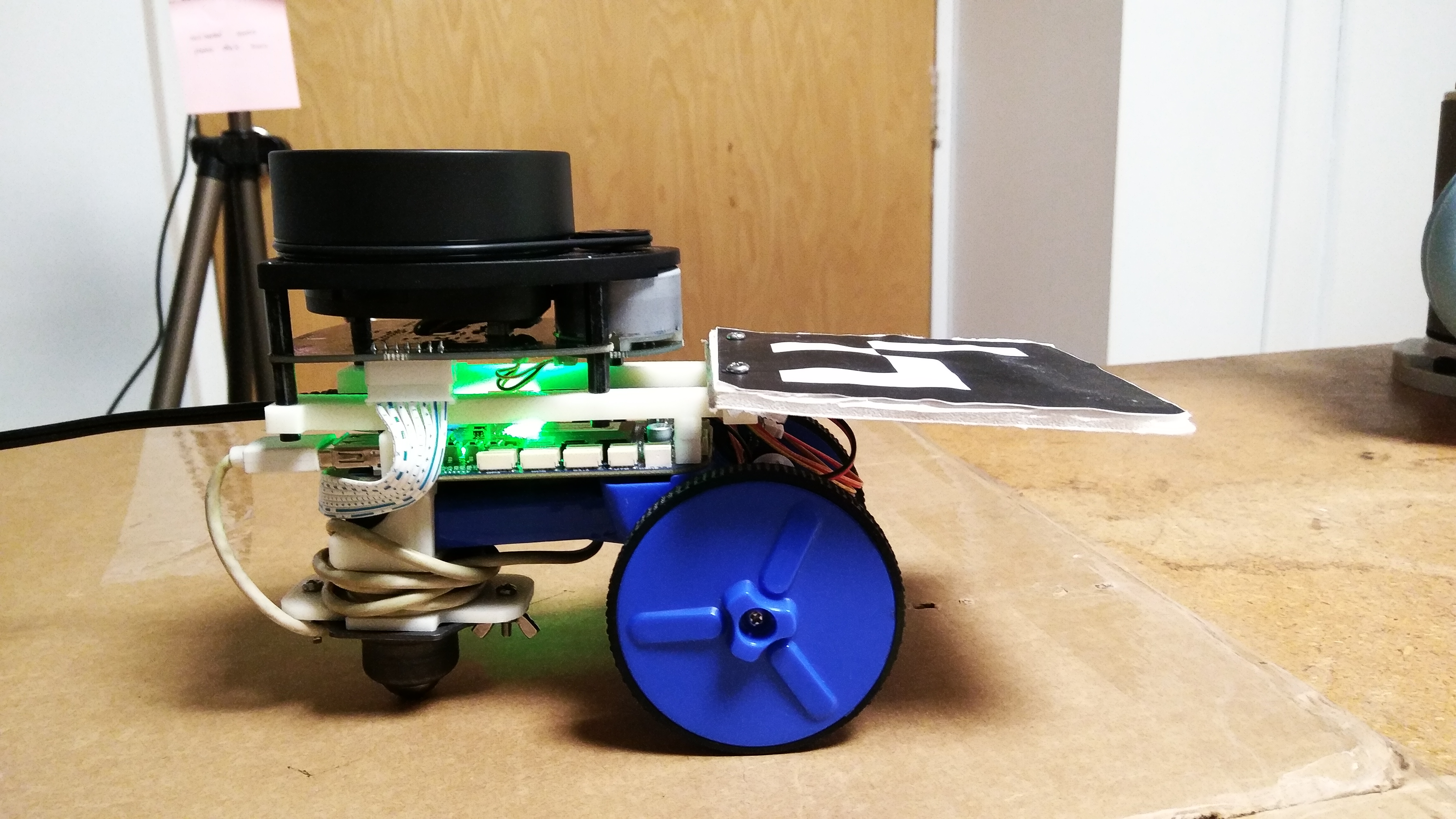


Figure 1: an EduMIP robot with RPLIDAR A1 Laser sensor

Our goal is to develop a motion planning algorithm based on RRT which computes a desired trajectory to a given desired goal region, such that: (a) the trajectory minimizes the distance; b) EduMIP does not collide with obstacles, which are based on the map generated by Simultaneous Localization and Mapping (SLAM). Then we will develop trajectory tracking control laws to follow the computed trajectory via feedback linearization. Finally, we will use MATLAB to demonstrate the result.

2. RRT blossom

RRT algorithm is designed to search high dimensional spaces within configuration space by randomly building tree. It composes of creating randomly points, testing collision then creating desired vertex, and so on till it reaches the goal.

RRT blossom is a new variation of the RRT planner which demonstrates good performance on both loosely constrained and highly-constrained environments [1].

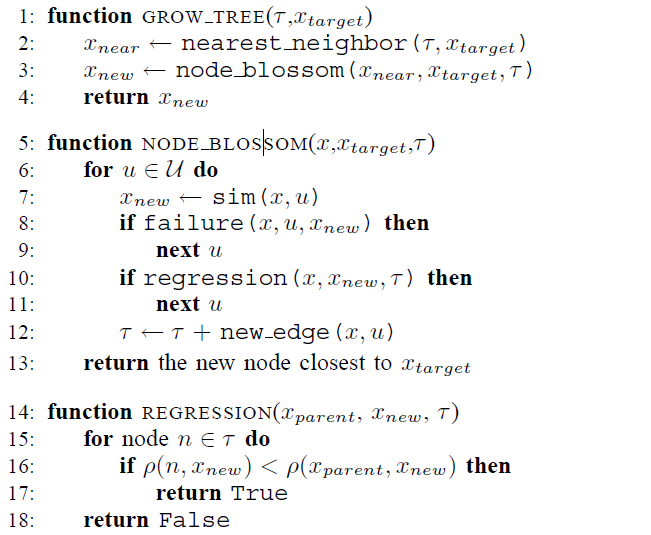


Figure 2: pseudocode for RRT-blossom

This algorithm handles unsuccessful expansions efficiently, by adding a regression function to make sure that they are never re-attempted.

3. MATLAB functions

Here is short preview to our RRT\_EduMIP file with functions:

3.1 existing library: Librobotics

Librobotics is a small but useful library of MATLAB functions frequently used in robotics for plotting. It offers functions to plot different robots, transforms, reference frames, scalable text, error ellipses, or compound operations. We use this to draw the robot.

It can be downloaded from <http://srl.informatik.uni-freiburg.de/downloads>.

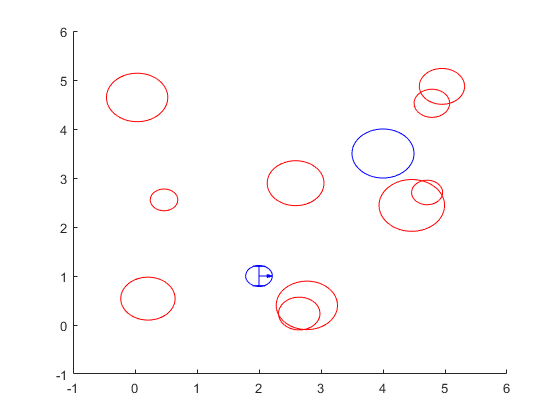


Figure 3: robot and obstacles drawn by Librobotics

3.2 initialization

Define tree and add initial point to the tree. Define obstacles.

% Define Tree

% Tree consists of vertex

% v.id -- ID of the vertex

% v.pose -- Pose of the vertex

% v.edgeq -- edge where to save intermediate configurations

% v.edgeu -- edge where to save intermediate controls u

% v.pId -- parent ID

%% Define obstacles: consider obstacles as ellipses

% obs(i).position

% obs(i).width

% obs(i).heigth

3.3 RRT

RRT procedure:

1. Generate Random Configuration and check for collision
2. Find Nearest Vertex
3. Extend the tree
4. If Goal return path

while(goalFound<1)

%% Generate Random Configuration

qrand=randomstate(N,dim);

% Check

if(isempty(qrand))

break;

end

if(checkcollision(qrand(1:2)',obs)>0)

disp('wrong qrand')

continue

end

if(PRINTSAMPLE)

figure(1),plot(qrand(1),qrand(2),'\*g')

end

%% Find Nearest Vertex

v\_near=NearestTree(RRT\_tree,qrand);

figure(1)

hv=plot(v\_near.pose(1),v\_near.pose(2),'\*r');

%% Extend the tree

[v\_new, RRT\_tree]=extendblossom(RRT\_tree,obs,v\_near,qrand,motionPrimitiveCommandArray,delta);

if(isempty(v\_new.edgeq))

continue;

end

%% If Goal return path

if(norm(v\_new.pose(1:2)-qgoal(1:2))<radiusGoal)

disp('GOAL!! reached!!!')

goalFound=1;

p=getPath(v\_new,RRT\_tree);

u=getControls(v\_new,RRT\_tree);

break;

end

end

3.4 Regression

Figure 4 shows the possible expansions for a particular node; all the red expansions (dashed) are regressing since a foreign node is closer than the parent (indicated with loops). Only the single edge in green is suitable for instantiation.



Figure 4: regression

credit: reference [1]

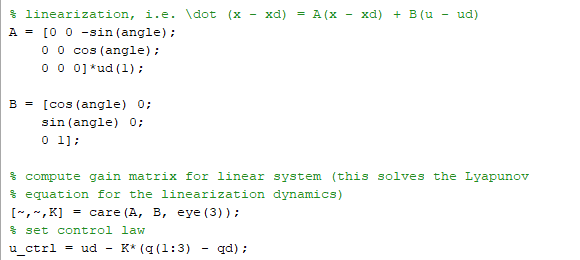
* 1. Trajectory Tracking

　 First, get desired path and controls from the tree.

Then use feedback linearization to find the controls

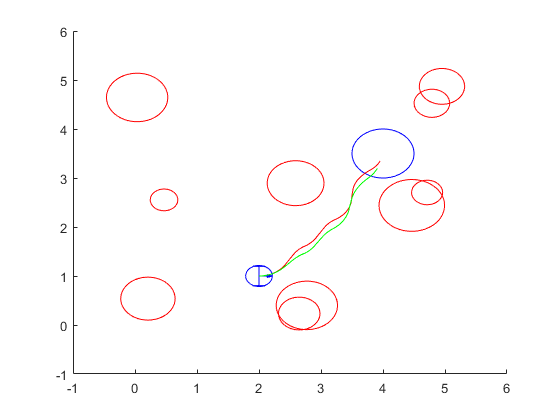
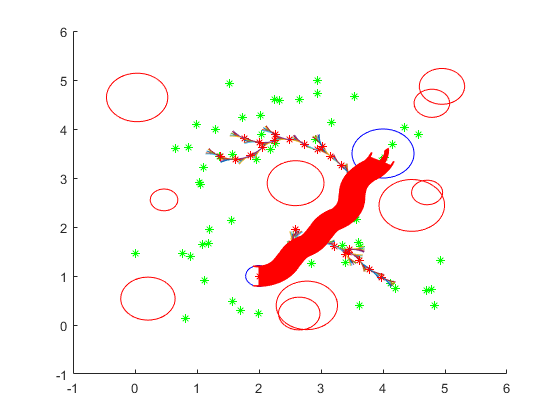
function p=getPath(vchild,RRT\_tree)

function u=getControls(vchild,RRT\_tree)



4. results

Red line represents desired path and green line is trajectory tracking.



5. reference

[1]. M.Kalisiak and M. van de Panne, "RRT-blossom: RRT with a local flood-fillbehavior," Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., Orlando, FL, 2006, pp. 1237-1242.

[2].<http://srl.informatik.uni-freiburg.de/downloads>