Developer Documentation: Representation and relative positioning from visual information

Realized by:
ASMA BRAZI

Proposed by:
CDRIC HERPSON

Laboratory of Computer Sciences, Paris 6
Sorbonne University - Faculty of Science and Engineering

June-July 2019

Contents

Chapter 1

Introduction

This document represents a guide for developers. The project has been built so that it can be extremely adaptable. So, it defines precisely each function developed. Also, it describes how some features can be improved.

We precise that this document covers only the main developed and updated features. For more details about the other features developed previously, please visit the project at `https://gitlab.com/asmabrz/pandroide`

The development of the project is divided into small applications.

Chapter 2

Code organization

The project is composed of two independent parts communicating with each others. The first part runs on a computer and it concerns the exploration strategy. The second one runs on a Raspberry-Pi and it concerns the image processing and the robot's movements. In what follows, the two parts are developed.

2.1 Computer side

When the robot collects visual information, It sends them to the computer. According to those visual information, the next move is decided from the exploration strategy, and sent to the robot. The code on the computer covering the exploration strategy and the communication between the robot and the computer is written in Java. We wrote the code on Java because the previous works were written in **Java** (`https://www.java.com`).

The visual information received by the computer may be walls or obstacles. This knowledge is represented in real time on a 3D map using **jMonkey Engine** at `http://jmonkeyengine.org`. The map is updated every time the robot explores a new element of the environment.

The most important functions developed are:

- **void simpleInitApp():** Initialize the 3D render.
- **void simpleUpdate(float tpf):** Update the 3D render periodically.
- **sendToPC(String objToReceive,String dir):** Receive on the computer an object sent by the robot.
- **void initPi():** Send the database of objects and scripts for image processing and robot movements to the robot.
- **void executeCommandRPi(String command,boolean getResult):** Order to the robot to execute the command mentioned in arguments. the parameter **getResult** specifies whether this command returns a result or not.
- **void rotate(double angle):** Order to the robot to rotate with a specific angle. If the angle is negative, the robot rotates left.

- **float move(double distanceRob):** Order to the robot to travel the specified distance. If the distance is negative, the robot moves back.
- **float updatePosition() :** After a movement, the robot's position changes. this function updates the robot's position accoding to the distance traveled.
- **List<ArrayList<Float>> captureData():** Order the robot to collect visual information and to send them to the computer.
- **int explore(int target):** Order to the robot to explore the environment and to search for a target.
- **float pixToCmY(float x):** Convert to centimeters a distance measured in pixel between the robot and a specified object.
- **float pixToCmX(float x1,float x2):** Convert to centimeters an object width according to its width measured in pixels and the real distance from the robot in centimeters.

2.2 Raspberry-Pi side

From the start of the project, this part has been mainly written in Python, and we kept that choice. The principal modules used in the development are:

- OpenCV for image processing (Hough lines, LSD, Canny...etc).
- Picamera to manipulate a Raspberry-Pi camera (Take a picture).
- Aseba to manipulate a Thymio-II robot (Rotation, movement, feedback from sensors...etc).

The most important functions developed are:

- **takePicture(r1=600,r2=400):** The robot takes a picture with the Raspberry-Pi. The resolution me bay specified but not required.
- **HoughLinesP(im='v.jpg',maxLG=350):** Returns the longest vertical line of an image. The maximum length can be specified with the parameter **maxLG**.
- **recognition(img,nbRefs=1):** Verify if any element of the database of objects figures on the image specified in parameters.
- **newLineMerging(l1,l2):** Merge the two lines **l1** and **l2** into one line.
- **mergeLines(lines,threshold=30):** Given a set of lines, the nearest lines are merged according to a fixed threshold.
- **LSDDetection(im):** Apply LSD algorithm on a picture to detect vertical, horizontal and diagonal lines.
- **moveForward(distCM):** Order the robot to move forward. If an obstacle is detected and the robot travels less than the distance requested, the real distance traveled is returned.