TDP

1 Introduction

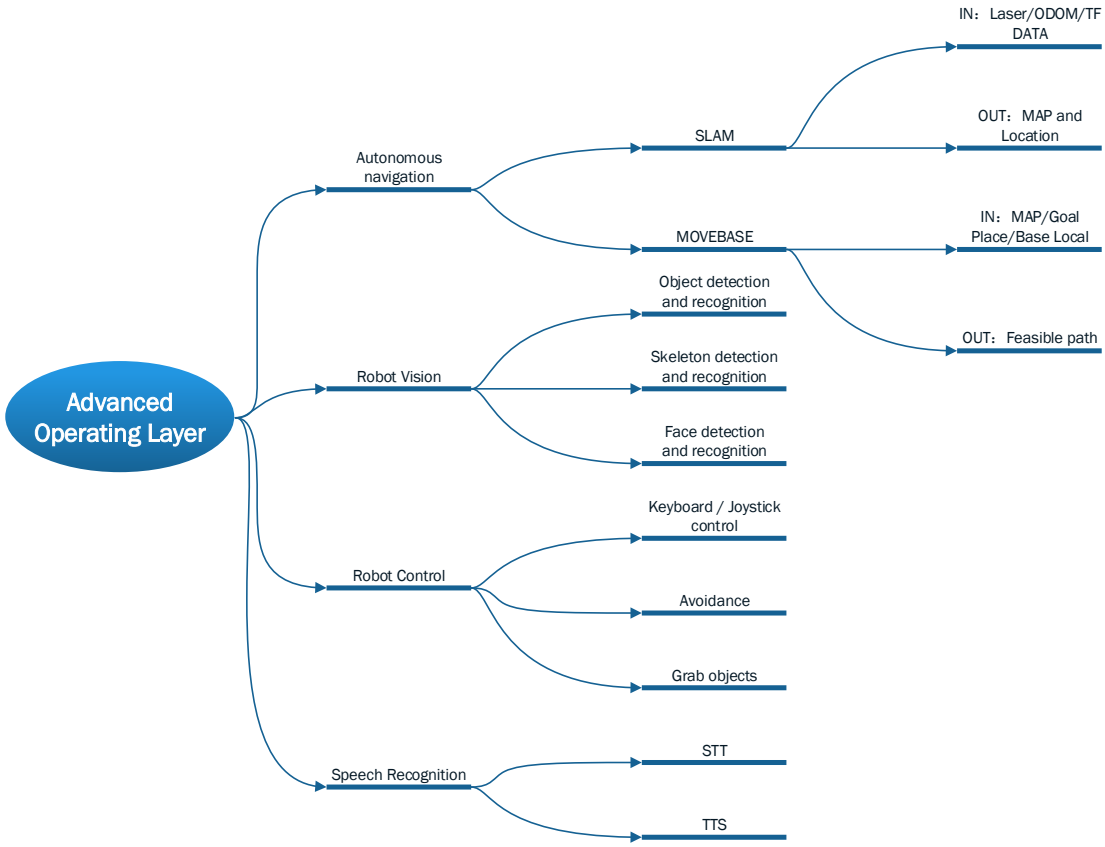
The RoboCup@Home league aims to develop service and assistive robot technology with high relevance for future personal domestic applications. It is the largest international annual competition for autonomous service robots and is part of the RoboCup initiative. A set of benchmark tests is used to evaluate the robots' abilities and performance in a realistic non-standardized home environment setting. Focus lies on the following domains : Human-Robot-Interaction and Cooperation, Navigation and Mapping in dynamic environments, Computer Vision and Object Recognition under natural light conditions, Object Manipulation, Adaptive Behaviors, Behavior Integration, Ambient Intelligence, Standardization and System Integration.

The motivation of developing our robot BangBang is twofold. First, we want to build an intelligent robot integrated with advanced AI techniques, such as natural language processing , hierarchical task planning and knowledge acquisition. Second, by participating RoboCup@Home League, all these techniques could be tested in real-world like scenarios, which in return helps the development of such techniques. Other demo videos are available on our website.

2 Hardware

The robot BangBang is designed to manipulate a lot of objects indoor. Robot is based on a three-wheels driving. A lifting system is mounted on the chassis, attached with the robot's upper body. Robot has two arms, with three degrees-of-freedom. The robot's power is supplied by two 20Ah batteries. A laptop is used to meet the computation needs. The power provides the robot works about two hours. Our robot is equipped with a Kinect camera, a SICK sensors and a microphone.

3 Software



a. Self-Localization and Navigation

In this chapter, we will cover the three essential ROS packages that make up the core of the Navigation Stack:

•`move_base` for moving the robot to a goal pose within a given reference frame

•`gmapping` for creating a map from laser scan data (or simulated laser data from a depth camera)

•`amcl` for localization using an existing map

When we are finished, we will be able to command the robot to go to any location or series of locations within the map, all while avoiding obstacles. Before going further, it is highly recommended that the reader check out the Navigation Robot Setup tutorial on the ROS Wiki. This tutorial provides an excellent overview of the ROS navigation stack. For an even better understanding, check out all of the Navigation Tutorials. And for a superb introduction to the mathematics underlying SLAM, check out Sebastian Thrun's online Artificial Intelligence course on Udacity.

b. Vision

Person recognition

Introduction

In follow me test, the robot is required to tracking and recognizing a locked person who is previously unknown to the robot. Our solution is use OpenNI to get the skeleton data of the persons in the vision from Kinect, then loop all the person recognized by OpenNI, match the main color with the locked person. If matches, the robot recognizes the person as the locked and follow him.

Recognition Algorithm:

Use skeleton information to cut the body image from the vision frame
Split the body image into 4 parts
Use the RGB information to calculate a finger print for each parts
Compare the finger print of the locked person and the finger print of the body image of the recognized skeleton
If the confidence exceeded the threshold then consider the skeleton as the locked person

c. Move Control

（1）Three wheel drive

Minho team decided to use a three-wheel drive because of the rich manoeuvrability and also due to the simple control. This type of wheels has small rollers to allow the wheels to move freely on any direction. They move along the primary diameter, just as any other wheel. Though, the smaller rollers along the outside of this diameter allow free rotation along an orthogonal direction to the powered rotation. The mechanics were even simplified in this case, because the previous built robot platform used three chains to reduce the speed of each motor by a ratio of 1/48. With this new platform the motors are coupled directly to the omni wheels simplifying the mechanics. The traction reduction is slightly compensated by the third wheel and therefore the traction loss is partly compensated. The mechanical construction is shown in Figure 2.

On the left image, the grey circle represents the robotic platform, and the three motors coupled to the Omni wheels are mounted with 120 degree between them, aligned like in an equilateral triangle so that their axis intersect at the robot

centre. In the centre it can be seen the specially built encoders coupled to each motor axis.

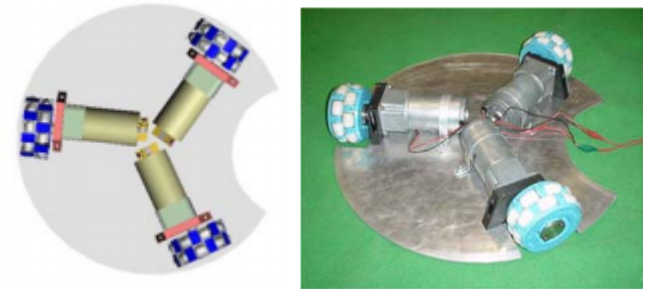


Figure 2: Three-Wheel drive mechanical construction (design and physical)

（2）Kinematics of Minho Robotic Platform

The inverse kinematics model is simple. It was considered that the representative coordinates of the robot were located in its centre. Each wheel is placed in such orientation that its axis of rotation points towards the centre of the robot and there is an angle of 120º between the wheels. The velocity vector generated by each wheel is represented on Figure 1-b by an arrow and their direction relative to the Yr coordinate (or robot front direction) are 150º, 30º and 270º respectively.

1） Linear Movement

For this type of configuration, the total platform displacement is achieved by summing up all the three vectors contributions, given by:

$$\vec{F}_T = \vec{F}_A + \vec{F}_B + \vec{F}_C \qquad (1)$$

A software simulator was built and is depicted in Figure 1. The user inputs three variables (linear speed, linear direction and angular speed) and the program outputs each motor contribution. For now, only linear speed is described.

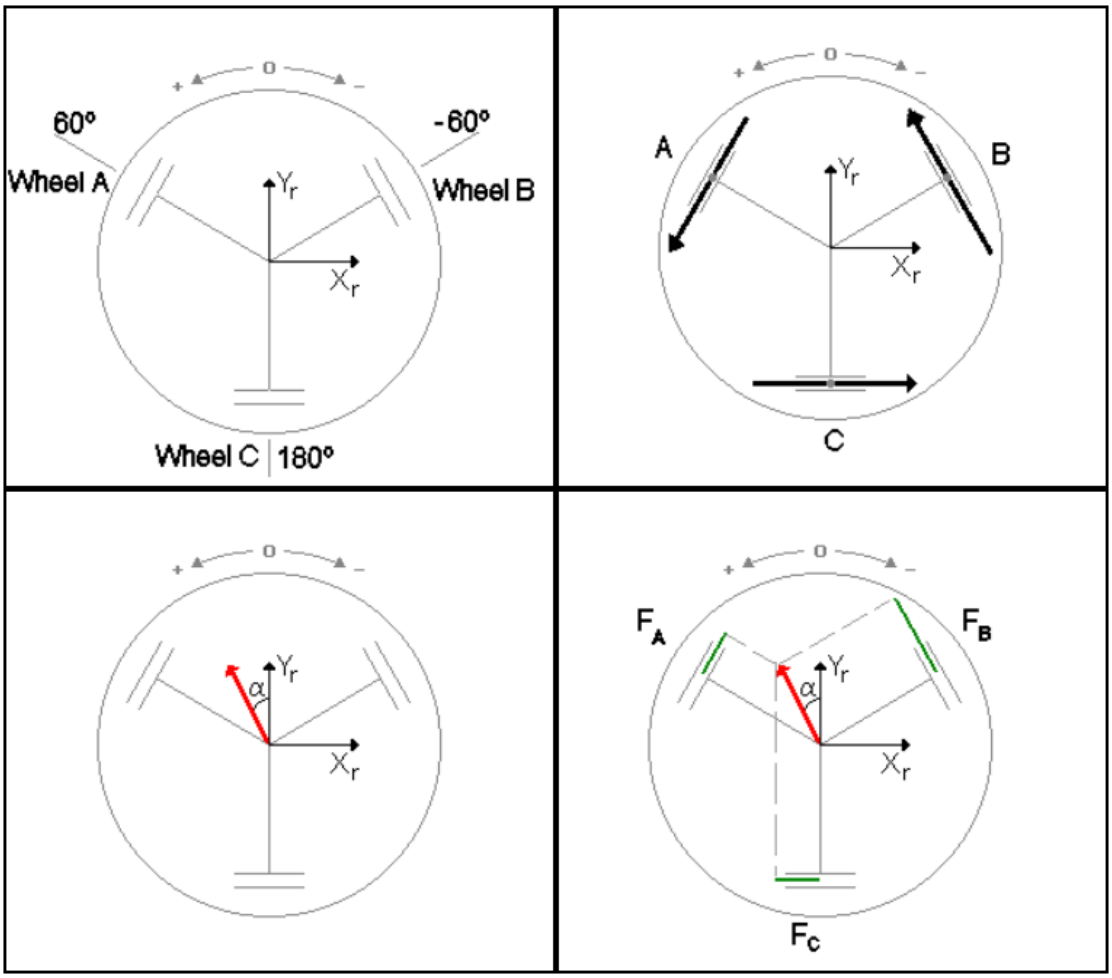


Figure 1: Graphical representation of motor contribution; a) platform motors/wheels distribution; b) wheels driving axis; c) desired movement; d) motor contributions

First of all, some definitions need to be considered. Figure 1-a) represents the diagram of the mobile robot platform with the three wheels. It was assumed that the front of the robot represents 0 degrees direction, and the positive side to its left. The three wheels coupled to the motors are mounted at angle position +60, -60 and +180 degrees respectively. It is important to remember that the wheel driving direction is perpendicular to the motor axis (therefore 90 degrees more). The line of movement for each wheel (when driven by the motor and ignoring sliding forces) is represented in Figure 1-b) by the segments A, B and C. The arrow indicates positive direction contribution. The total platform displacement is the sum of three vector components (one per motor) and is represented as a vector in the platform body centre. In Figure 1-c) it is depicted a vector representing the desired movement; the angle α represents the direction and the vector length represents the velocity. In order to find out the three independent motor contributions, this vector is projected on A, B and C axis representing the line of movement of each wheel. Figure 1-d) shows the projections that represent the three vector components of the contributions. The vectors can have a positive or

negative direction which represents the direction in which the motor has to move (forward or backwards respectively).

Since the robot forward direction is represented by Yr, each motor contribution consists of the cosine of the angle α (DesiredDirection) projected on each wheel drive direction, multiplied by the velocity, given by:

$$F_n = velocity \cdot \cos(WheelDriveDirection_n - DesiredDirection) \quad (2)$$

Considering now that the three wheels driving directions of this robot are 150, 30 and 270 degrees respectively, the contribution for each motor for linear velocity is given by:

$$F_A = velocity \cdot \cos(150 - DesiredDirection) \quad (3)$$

$$F_B = velocity \cdot \cos(30 - DesiredDirection) \quad (4)$$

$$F_C = velocity \cdot \cos(270 - DesiredDirection) \quad (5)$$

Where: F - is the motor vector contribution

A, B, C – represent the motors

Velocity - is the linear velocity the robot should move

DesiredDirection - is the angle α of the desired movement

Figure 2 shows each motor contribution according to the desired direction from 0 to 360 degrees.

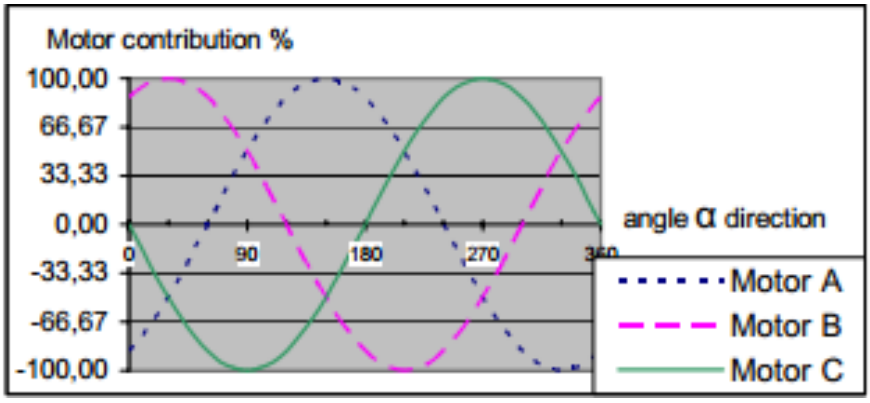


Figure 2: Motors contribution for robot platform movement

d. Dialogue Understanding

Pre-inserted sentences recognition

Introduction

In *basic function* test, the robot will be asked 3 ~ 10 pre-announced questions and should provide an appropriate answer. Our solution to the test is use STT programs to transform the sentence into text format, compare the text with the pre-inserted sentences to find out which is this question and give the answer to the question with TTS programs. As the recognition rate is limited, it is rarely for the programs to transform the sentence into text perfectly. As a consequence, a strategy should be used to ensure the program can find the correct sentence with efficiency. Our solution is loop all the questions and calculate the length of LCS between the text and question, choose the one which has the biggest match percentage as the question asked.

LCS (Longest Common Subsequence)

What is LCS

The LCS (Longest Common Subsequence) algorithm, briefly speaking, is to find the similarity of the two given strings. Different from directly matching which can only match the strings consecutively, the LCS algorithm can carry out inconsecutive matching on the original string and find the longest common subsequence of the original string.

Why LCS

As the recognition rate is limited, sometimes the STT programs will recognize one word as another one or more words. Under this circumstance, if we want to find the similarity of the two sentences, we should "pass" the mismatched words and find the maximum of the common words in the order of the original sentence. But the direct matching can only solve the consecutive strings, so if there is a mismatched word, it should go over the left sentence to find a new place to start with. Obviously, if we support the length of the two

sentences is both n, the total complexity should be more than $O(n3)$, but to LCS, it is only $O(n2)$.

How to do with LCS

The LCS algorithm is a Dynamic Programming, so the importance is its dynamic programming function.

Let two sequences be defined as X=($x1$,$x2$...xm) and Y=($y1$,$y2$...yn). The prefixes of X are $X1$,2,...m; the prefixes of Y are $Y1$,2,...n. Let LCS(Xi,Yj) represent the set of longest common subsequence of prefixes Xi and Yj. Let $LCS(i,j)$ represent the length of longest common subsequence of prefixes $Xi$and Yj.

The LCS function is defined as follows:

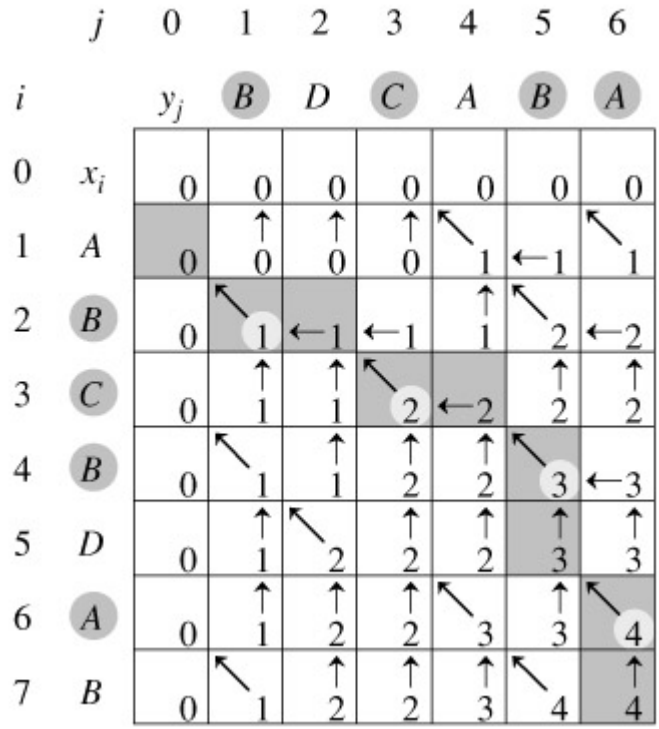
$$LCS(X_i, Y_j) = \begin{cases} \emptyset & if\ i = 0\ or\ j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \cup \{x_i\} & if\ x_i = y_j \\ longest(LCS(X_i, Y_{j-1}),\ LCS(X_{i-1}, Y_j)) & if\ x_i \neq y_j \end{cases}$$

If we just want to find the length of LCS, then the function can be defined as follows:

$$LCS(i,\ j) = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ LCS(i-1,\ j-1) + 1 & if\ x_i = y_j \\ max(LCS(i,\ j-1),\ LCS(i-1,\ j)) & if\ x_i \neq y_j \end{cases}$$

Instance

Let letters in bigger case represent a word in the sentence and remove all the spaces. Let the original sentence be `ABCBDAB` and the transformed text be `BDCABA`. The result matrix is shown as follows:



4. Bibliography

[1] Software source. http://wiki.ros.org

[2] Move Control source. ROS_by_example_groovy___volume_1