Use of Simultaneous Localization and Mapping Algorithm and Tracking-Learning-Detection Algorithm in Botball Competition

Yimo Xu
Qingdao No.2 Middle School

0 Introduction

Botball 2017's difficulty is much higher than before. Water towers and blue poms may moving when touching the ground, make robots very hard to obtain them. Also, when climbing the slope, robots may fall down if the black duct tape is not precisely followed. To finish different tasks steadily, robots need algorithms that can track objects' position, and determine its own position due to various reasons such as falling down or motor error.

This paper focus on use of vision tracking algorithms and feasibility.

1 Problems

As we mentioned above, some high-valued objects are moving and thus they are very hard to be obtained, since if we use a low mechanical hand, robots can only obtain them when facing directly to them in a certain distance. Also, if robots cannot determine their position if something bad happen, they will score less and even interfere another robots' operation.

2 Preparation for the Solution

2.1 Selection of Algorithms

The author is responsible for the small robot. By searching the Internet and reading related document, author found that two algorithms may be used in competition: SLAM, which can determine current position of the robot after the robot is randomly placed, and TLD, an algorithm that tracks moving objects, such as water tower in the competition. OpenCV provides several basic libraries related to these algorithms.

2.2 Building the Platform

1.3.1 Software

We were decided to use Wallaby controller to run these algorithms on the robot. Unfortunately, although we can plug in several accessories on the controller, we are

always unable to solve OpenCV's dependencies indicated in cmake. But we are able to know how the camera mounted on Wallaby works, so we set an environment on Ubuntu 16.02 LTS to simulate Wallaby's UNIX-like system testing two algorithms only.

1.3.2 Hardware

SLAM algorithm needs an RGB-Depth camera like Kinect, which is not possible for us to mount on Wallaby. Instead, we used the webcam provided along with Botball parts and two servos with the IR sensor to solve the RGB-Depth scenario. Two servos will move horizontally and vertically, making the IR sensor cover all parts that the webcam covers. We used trigonometric functions to solve the actual distance.

3 Finishing the Solution

As for the TLD algorithm, we used OpenTLD, which is a C++ implementation of TLD algorithm published on GitHub under GNU GPLv3 License. To simulate the environment on Wallaby controller, we finished a tiny program that capture 6 pictures of the Botball playground in one second and consolidate them into a .mpg video file that can be recognized by OpenTLD.

We cannot simulate all environment for the SLAM algorithm because it needs some operations that cannot be done in our simulation environment such as controlling the servo. Thus, we recorded all depth data that will be used in this algorithm and input in by hand. We also solved the scaling factor of the IR sensor, which is needed if we need to convert the IR sensor's output into distance.

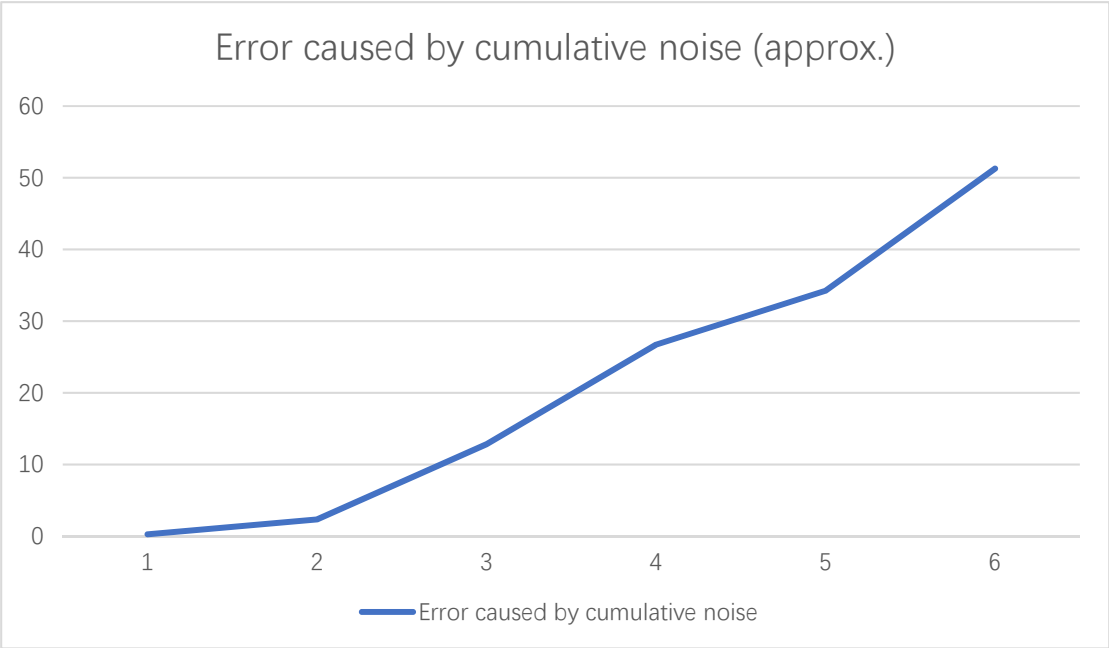
The most interesting part of this algorithm must be solving where the robot is. We first move the robot (camera when testing) and let it find some points that is unique in the playground, such as the intersection point of duct tapes. Then, we move the robot, simulating falling or other situation. Iterative Closest Point Algorithm (provided in OpenCV) is used to get more key points in the image during the whole process. Using function SolvePnPRANSA(); in OpenCV building on Random Sample Consensus architecture, we will be able to solve the robot's displacement, and using the unique points we get, we will be able to solve the position of the robot.

4 Result

The TLD algorithm can successfully solve the displacement of the water tower, but since the Wallaby's capability limited the video to a maximum of 6 FPS, it is very hard to track other objects that move very fast.

SLAM algorithm can solve current robot's position, but since the resolution of images

is not very high, the cumulative noise becomes unacceptable high after a short period of time.



5 Feasibility

Two algorithms do have finished and open-source project available online, but it will be very hard to make them adopt Wallaby Controller, as well as that the Wallaby Controller may become insufferable slow if running both OpenCV library and competition program. Since the competition does not allow us to attach other accessories to the Wallaby, it is nearly impossible to use these algorithms during competition.

6 Conclusion

Although it is impossible to finish these tasks during competition by using these two algorithms, using these algorithms will definitely improve the precision of the robot. We will try to make these two algorithms faster by deleting useless steps in the competition or improving the image-capturing process using of the Wallaby defined in /usr/include/wallaby/camera.h.