OpenCV IDP Final Report and Failure Analysis

#flow
Categories: OpenCV Cambridge Engineering

OpenCV Report and FA

Decision to abandon line following

We made the decision to attempt to use OpenCV for the following reasons:

1. To allow the robot to traverse the ramp, significant, potentially breaking changes would need to be made to the location of the line sensors
2. The electrical connection to the line sensors have been rather unreliable

OpenCV control architecture

The main control loop brings together the following 2 modules to complete the tasks set by CUED:

Remote control over WiFi

A http request parser was implemented on the Arduino onboard the robot, which allows for a master-slave style communication protocol.

The Arduino is connected to the main control PC via the PC's mobile hotspot. The PC can then send commands over to the Arduino, each consisting of an instruction string and an optional parameter string. For example, the robot can be commanded to **move forward** for *1000ms*. After each command, the Arduino sends back a HTTP 200 OK response, with a reading of the LDR value, which enables the detection of the block type.

Video acquisition and processing

The video acquisition module takes care of tasks like acquiring real time video from the overhead camera, correcting for distortion and locating the robot.

The `arena` class located within this module also contains calibrated waypoints for navigation across the arena, which include all the positions where blocks are to be collected or dropped, as well as tunnel and ramp endpoints.

Control loop behavior

At the beginning of the routine, a calibration is performed where the distance and anglular displacement per unit time is recorded, and used for all subsequent movement commands.

Then, the robot simply navigates to predetermined waypoints and performs the requisite actions to complete the tasks set for us.

Scheme to improve reliability

Due to the script losing track of the marker intermittently when the robot is at the peripheral areas of the camera, and by extension the very edges of the arena, we devised the

following scheme to attempt to regain a lock on the AruCo marker:

When the main control script requests a reading of the current position of the robot, if the target locating script does not manage to find the target in the current frame, it will try again by acquiring a new frame and running all the detection routines on the new frames, for a maximum of `max-retries` times. This helps to solve issues related to missing the marker due to noise.

Should the script still fail to locate the marker, the robot performs a manoevure to attempt to go to an area of the arena where the target has a better chance of being detected. This is achieved in the final script by reversing the robot for 1000 ms and commanding it to go forward for 2000 ms. During our testing, it was found that this reverse before moving forward scheme was particularly effective if the robot gets stuck inside the tunnel.

Key decisions in OpenCV Architecture

Waypoint based navigation

Initially, a complete set of functions to automatically detect features within the arena and intelligently pathfind around was developed. Despite having a working solution, we chose to abandon it in favour of a simpler, pre-recorded waypoint based navigation solution due to difficulties testing the smart feature detection based solution as it requires the arena to be completely clear of unexpected objects, and it was impossible to clear the arena as other teams also had to use it.

Time based control

As the decision to use OpenCV was made rather late, it was not possible to realise the initial idea of having optical encoders on the wheels for absolute position sensing. This made the task of accurately navigating the robot extremely challenging due to the inconsistent performance of the DC brushed motors being controlled by PWM in an open loop manner. However, it was the only option given the time constrains.

Use of AruCo markers

Initially, the plan was to use a round target to serve as the main detection target, as a large round object would be very easy to detect since it will be a unique feature within the field of view of the camera. However, upon testing, the size of target needed exceeded our expectations, and 3D printing a target large enough would not have been possible within the time we had. So, the decision was made to use AruCo markers which had the added benefit of also providing information of the heading of the robot.

Only attempting one lap

Given the constrains of having to slow down the motors to reduce errors caused by inconsistent motor drive strength, we had to reduce the motor speed to about 80% of maximum, which did not allow us to comfortably navigate around multiple times within 5 mins. Hence, it was decided that we would prioritise getting back to the home point to confidently gain the 20 points associated.

Failure Analysis

Unfortunately, despite all the measures described above attempting to increase the reliablilty of the system, it failed to complete a single lap around the arena, which only allowed us to score 10 points out of the (conservatively estimated) 70 points.

The main cause of failure was that the video processing script lost track of the marker, and all of the logic implemented to attempt a recovery failed to do so.

It was particularly perplexing because when the failure occured, the robot was in an area of the arena that had extremely reliable marker detection in all of the prior testing (usually, the marker would be lost near the tunnel edge, but this time the failure occured at the ramp end, which was well within the field of view).

Without extensive testing and validation, it would be hard to conclude what exactly caused this unrecoverable failure. However, by observing the recorded video feed of past successful attempts, and the live video feed as the competition was going on, we surmise the following possible causes:

Change in lighting

We did not expect this to be an issue, as the entire idea of using the monochromatic AruCo marker was that it would be robust against changes in lighting. However, upon scrutiny, the reflections on the tape residue on the arena was different in the recorded and live footage. Considering the marker was printed with a laser jet printer which produces black areas which are glossy, it is possible that somehow light from unexpected sources was reflecting off the marker, causing too much loss of contrast.

The fix for this would be trivial: print the pattern using an inkjet printer, which produces matte prints

Recovery manouvre was ineffective

The recovery manouvre described was great in getting the robot unstuck from under the tunnel, and sometimes effective in enabling the script to regain a lock on the marker. However, in hindsight, the occasions where recovery was achieved when the robot was not stuck were probably flukes, since in the case of the competition, it so happened that the manoeuvre caused the robot to move *away* from the center where detection was most reliable.

A much better way to implement such a recovery measure would be to keep a log of the past commands sent to the robot, particularly those telling it to translate, and send the scaled down reverse of those commands, essentially partially undo-ing the past command that caused the robot to veer out of the zone where detection can be done successfully.

Imbalance in left and right wheels

This was a known issue since the start of development, and a mitigation was attempted by manually tuning the power sent to each motor, the calibration procedure at the start of the routine, and separating translation and rotation commands (because wheel speed imbalances primarily cause unexpected rotation). However, that still proved to be insufficient.

The best and most straightforward way to solve this problem would be to implement optical encoders on the wheel, as another group did, and as planned for ourselves. This would

have the added benefit of increased speed and accuracy, since the movement of the robot would be precisely known, and the slow calibration and on-the-fly heading correction can both be eliminated.

Poor choice of pattern

Looking at another group which managed good results with computer vision, we suspect that the particular pattern index we chose, and thus the generated pattern, was too complex.

Scrutinising the image from the camera revealed that at the periphery of the field-of-view, due to *comatic aberation*, the square cells of the pattern were no longer square, and there was significant spillover of the white areas into the black areas. We suspect that this was the most direct reason the detection was unreliable at the edges of the arena.

This is significant since the marker identification function implemented by the OpenCV library rejects markers that have a certain percentage pixels within cells (squares) which are the wrong colour.

The fix for this would be to choose a marker which is more blob-like, i.e. a marker that contains a large contiguous shape, as the other group had, would probably be more tolerant in terms of this failure mode, than the one we had, which was significantly more checkerboard-like.

Ultimately, these are just educated guesses at why the robot failed the way it did. Despite that, these speculations should serve as excellent jumping off points for any subsequent debugging efforts.

Probably a good idea to include that this is written on 2022-11-30 at 23:25

References: