MTRN4110 20T2 Project Phase C Task Descriptions (week 7-9)

(Updated 12/07/2020)

Changelog

- 12/07: First release

1. Overview of the Course Project:

The main project of MTRN4110 20T2 is a simulation-based project adapted from the Micromouse competition. Webots will be used as the simulation platform throughout the course. You will design a mobile robot and implement a controller and a vision program to negotiate a maze autonomously in Webots. The project will contribute 55% to your final mark of this course.

The project consists of four sequential phases:

- Phase A: Driving and Perception (week 1-3, 15%, individual)
- Phase B: Path Planning (week 4-6, 15%, individual)
- Phase C: Vision (week 7-9, 15%, individual)
- Phase D: Integration and Improvement (week 10-11, 10%, group)

This document will describe the tasks of Phase C.

2. Overview of Phase C – Vision:

The purpose of Phase C is to develop a vision program to create a map for the maze-solving robot. You are required to complete the tasks of this phase by the end of week 9.

2.1. Expectations:

By the end of Phase C, you are expected to have been able to:

- perform basic image processing and computer vision techniques for image analysis;
- detect the four cornerstones of a maze from an image and perform perspective transform;
- detect all the internal walls of a maze;
- detect the location and heading of a robot;
- generate a map of the maze with a robot in it and write the map into a text file.

2.2. Learning Outcomes Associated with this Assignment:

- **LO2**: Apply computer vision techniques for feature/object detection and tracking in complicated environments
- **LO3**: Demonstrate practical skills in mechatronics design, fabrication, and implementation

3. Phase C Task Descriptions:

In this phase, you are supposed to implement a vision program for the maze-solving robot. The program should be able to generate a map that can be used by the path planning module developed in Phase B.

Unlike Phase A and B, you do not need to implement the program in Webots for Phase C. Instead, you are strongly recommended to implement it with Python + OpenCV in a separate IDE (e.g., Jupyter Notebook in Browser or Visual Studio Code). If you are not comfortable with using Python, you are also allowed to use MATLAB. Note that, however, tutorials and support will only be provided for Python + OpenCV but not for MATLAB implementation. No other languages are allowed for the sake of assessment. (see Section 4.1 for more details about implementation.)

The program should complete the following tasks once started. (We will illustrate two examples in this document but only refer to the first one for conciseness; in implementation your program only needs to deal with one setup in each run.)

3.1. Read an image into the program and display it

You will be given two image files, one named "Maze.png" that captures the maze and the robot, as shown in Fig. 1, and one named "Robot.png" which is a closeup of the robot in "Maze.png", as shown in Fig. 2.

The results of all the tasks should be mainly illustrated based on "Maze.png" in your program; "Robot.png" could be used to help with some tasks, but use of "Robot.png" is optional.

For this task, you only need to read "Maze.png" into your program, and display it in RGB mode (original mode).

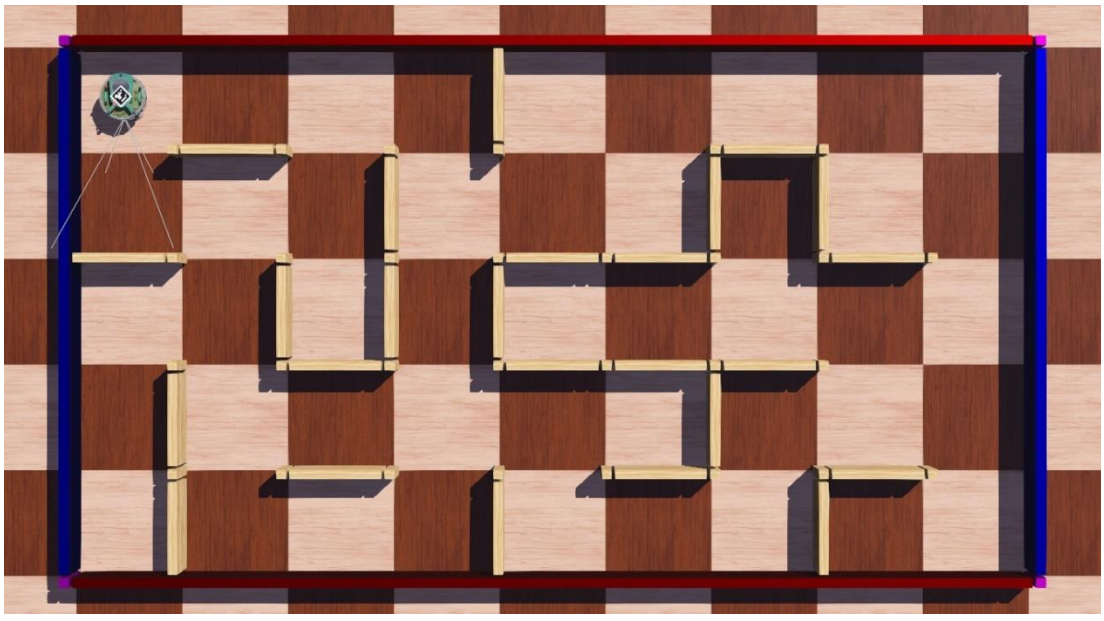

Fig. 1. (Example 1) Maze.png

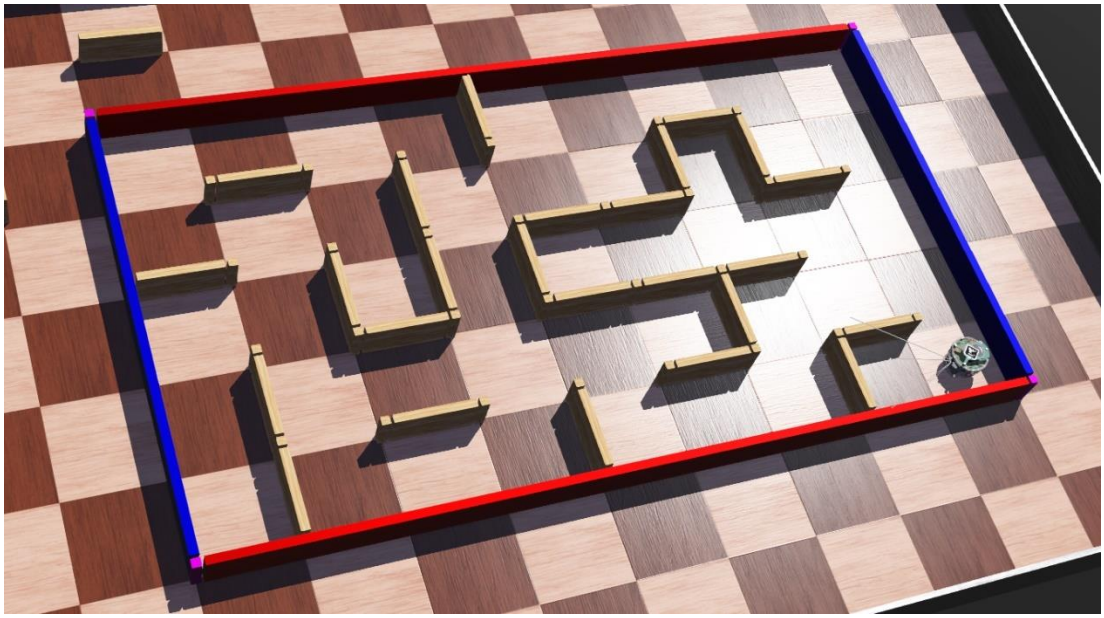

Fig. 1’. (Example 2) Maze.png

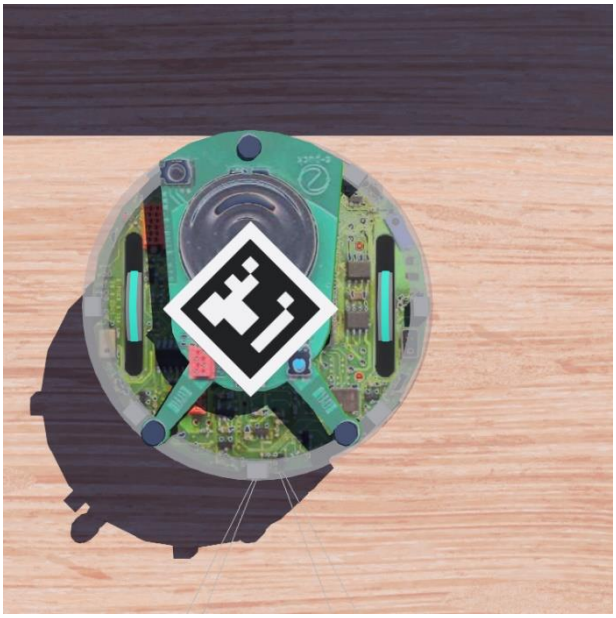

Fig. 2. (Example 1) Robot.png

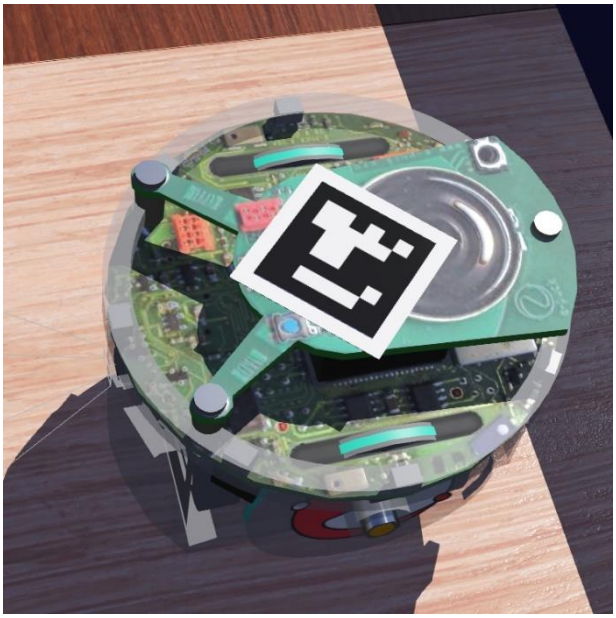

Fig. 2’. (Example 2) Robot.png

3.2. Find the four cornerstones of the maze

Detect the positions of the four cornerstones from “Maze.png” and indicate them on the image.

Below is an example indication. Other ways of indication are also acceptable if the positions of the cornerstones are clearly indicated (the centre of your indication should be placed within the top surfaces of the cornerstones).

Coordinates beside the image are not required in all the tasks.

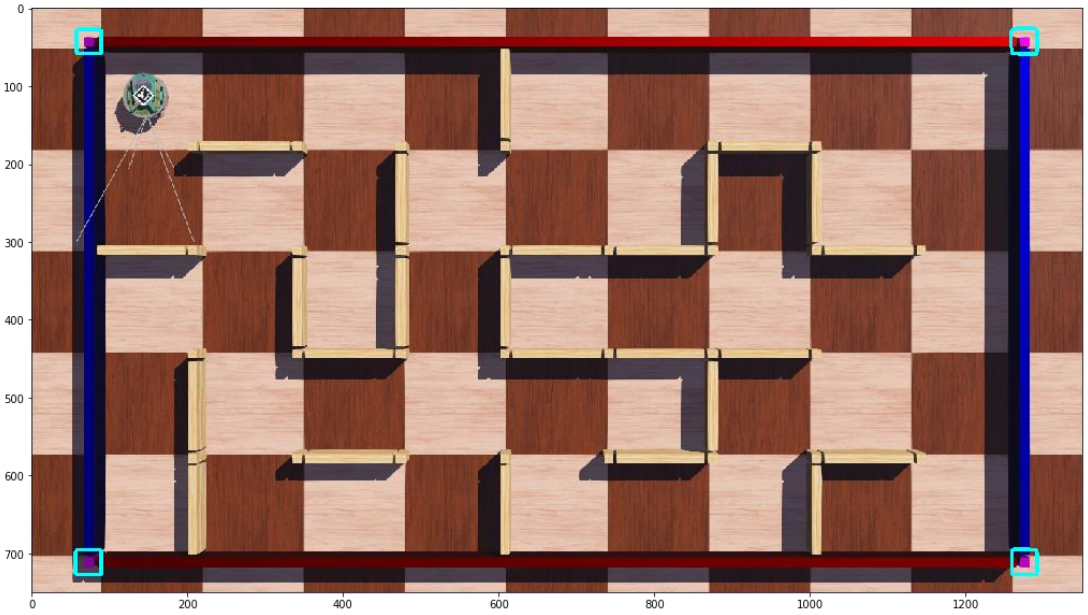

Fig. 3. (Example 1) Indication of the four cornerstones

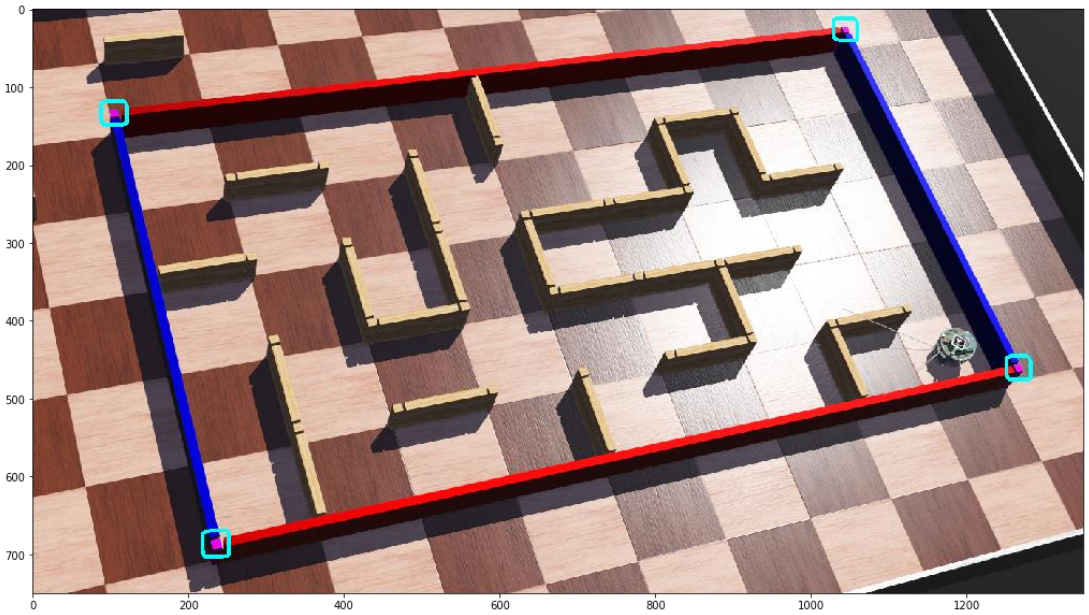

Fig. 3’. (Example 2) Indication of the four cornerstones

3.3. Perspective transform the maze from the original image to a rectangle image

Use perspective transformation to transform the maze from the original image to a rectangle image whose four vertices are the four cornerstones. The height:width ratio of the transformed image should be 9:5.

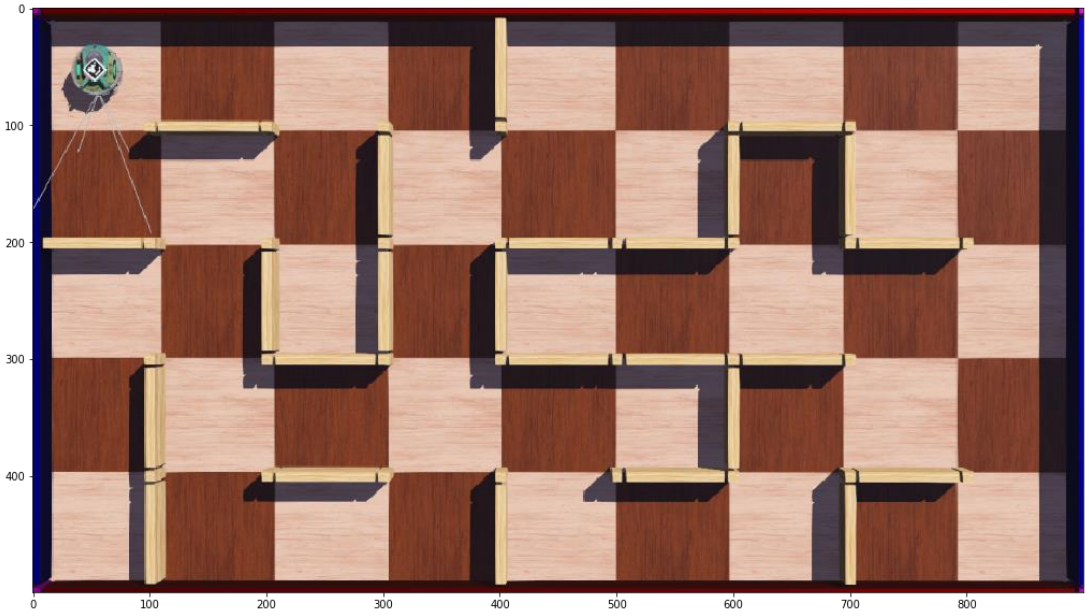

Fig. 4. (Example 1) Image after perspective transform

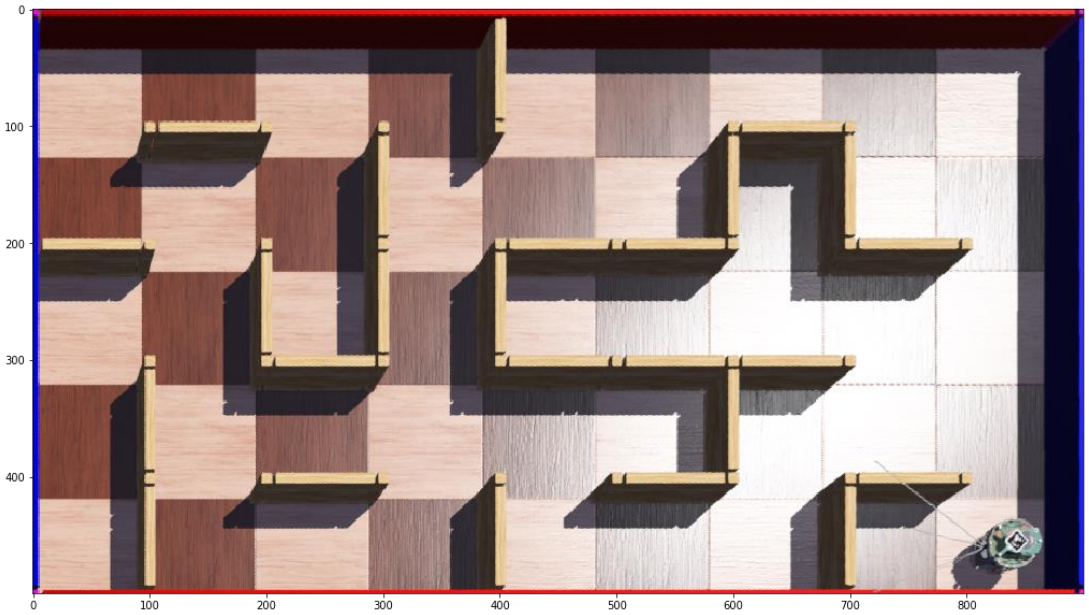

Fig. 4'. (Example 2) Image after perspective transform

3.4. Detect all the internal walls

Detect all the internal walls and indicate them on the transformed image (or the original image as shown in Fig. 1).

Below is an example indication. Other ways of indication are also acceptable if the walls are clearly indicated (the centre of your indication should be placed within the top surface of the walls).

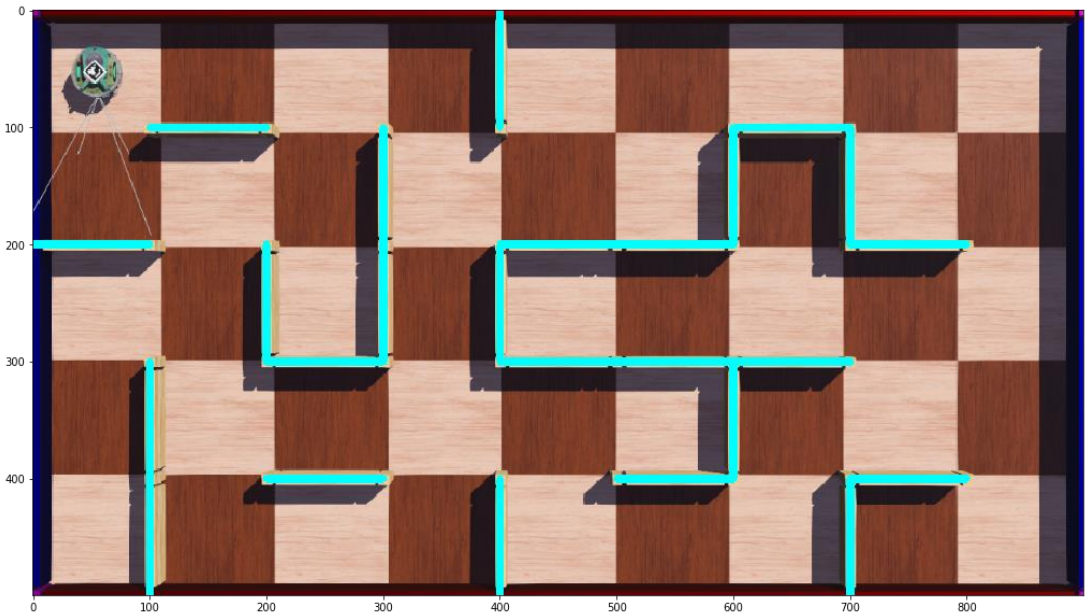

Fig. 5. (Example 1) Indication of internal walls

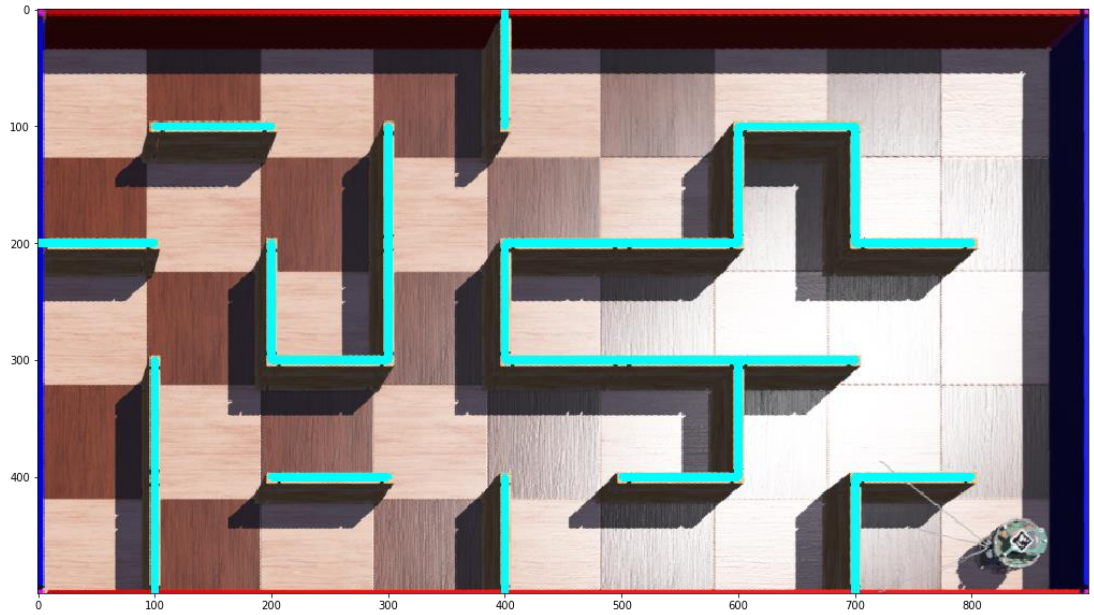

Fig. 5’. (Example 2) Indication of internal walls

3.5. Detect the location and heading of the robot

Detect the location and heading of the robot and indicate them on the transformed image (or the original image as shown in Fig. 1).

Below is an example indication. Other ways of indication are also acceptable if the location and heading of the robot are clearly illustrated.

The indication of the location should place its centre within the surface of the robot.

The indication of the heading should place its centre within the surface of the robot as well, and follow the rules defined in Phase B, i.e., use “^”, “>”, “v” and “<” to represent “N”, “E”, “S” and “W”.

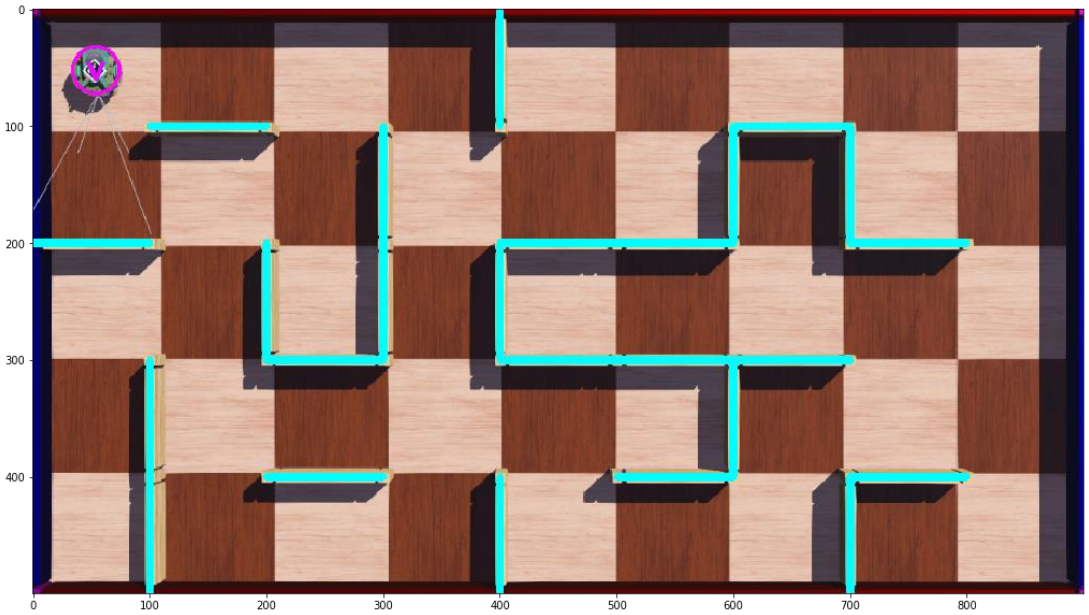

Fig. 6. (Example 1) Indication of location and heading of robot

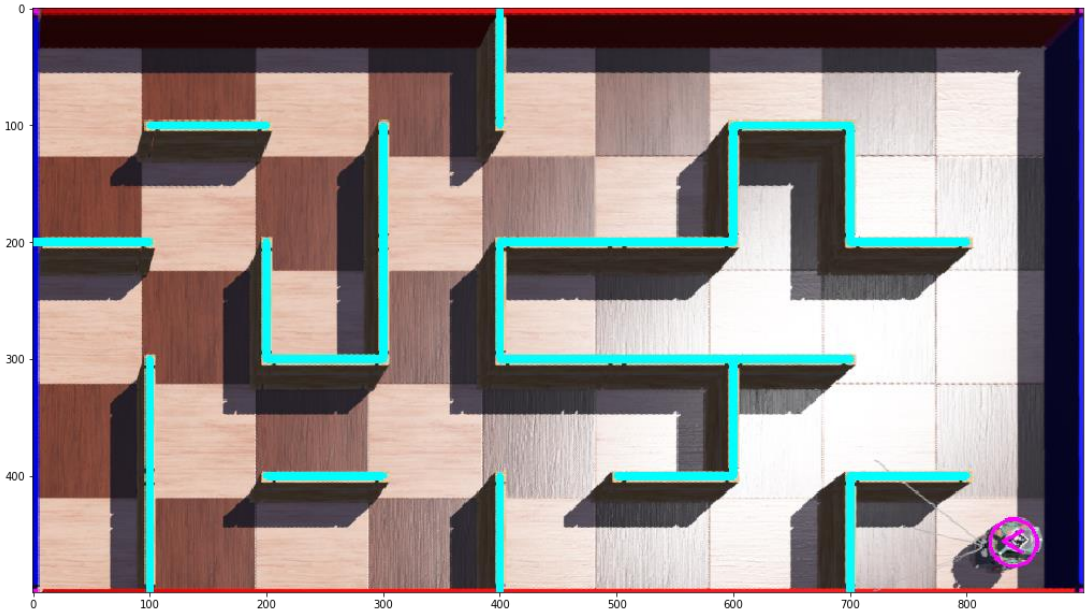

Fig. 6'. (Example 2) Indication of location and heading of robot

3.6. Generate a map and write it to a text file

Based on the processed image, generate a map containing all the walls and the location and heading of the robot. Write the map into a text file named "MapFound.txt".

The generated map must follow the convention defined in Phase B. A map using a different convention, even if essentially correct, will not get the marks associated with this task.

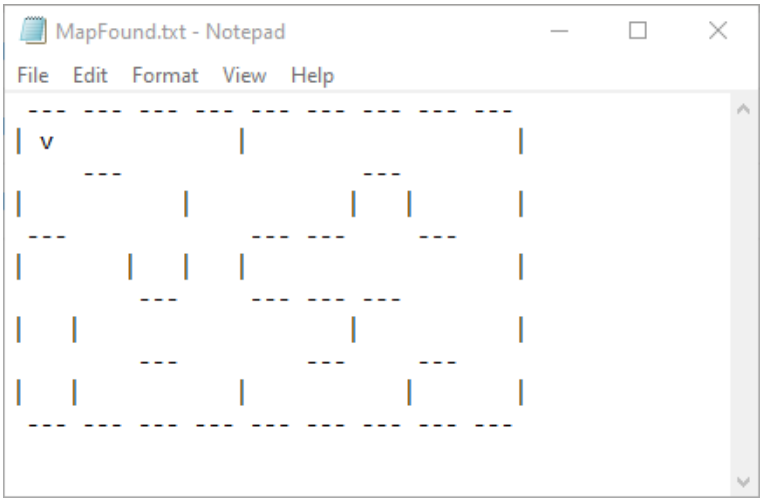

Fig. 7. (Example 1) Map found by the program

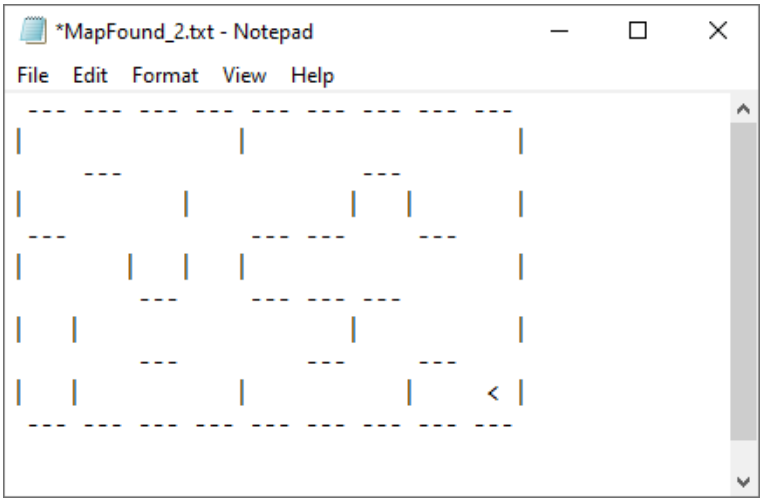

Fig. 7'. (Example 2) Map found by the program

3.7. Task summary:

Task	Description
1	Read an image into the program and display it
2	Find the four cornerstones of the maze
3	Perspective transform the maze from the original image to a rectangle image
4	Detect all the internal walls
5	Detect the location and heading of the robot
6	Generate a map and write it to a text file

4. Specifications and Hints:

4.1. Specifications:

Maze:

1. At the beginning of Phase C, you will be given two sets of examples named ("Maze.png", "Robot.png") and ("Maze_2.png", "Robot_2.png").
2. These are for your practice. For assessment, you may be tested with different images in which the maze layout and the initial location and heading of the robot may be different.
3. A world file of Webots "z1234567_MTRN4110_PhaseC.wbt" containing a maze and a robot will be provided to you so that you can generate different images for test.
4. Another world file of Webots "z1234567_MTRN4110_PhaseC_2.wbt" will also be provided to you to show you how example 2 image are generated.
5. The size of the "Maze.png" will always be 1350 x 750 in pixels.
6. The size of the "Robot.png" will always be 750 x 750 in pixels.
7. The maze will always be the same size (5 rows by 9 columns) with closed borders.
8. The initial position of the robot will always be one of the four corners.
9. The initial heading of the robot will always be towards an opening.
10. The layout of the maze for assessment may be different from the example.
11. The colours of the objects will always be the same as in the example.
12. The lighting will be the same as in the example world file of Webots, but it may appear differently with different perspectives.
13. The perspective of the image may be different from the example. However, the four cornerstones will be restricted within the following four regions, respectively.

Fig. 8. (Example 1) Location of the four cornerstones will be in the four regions, respectively

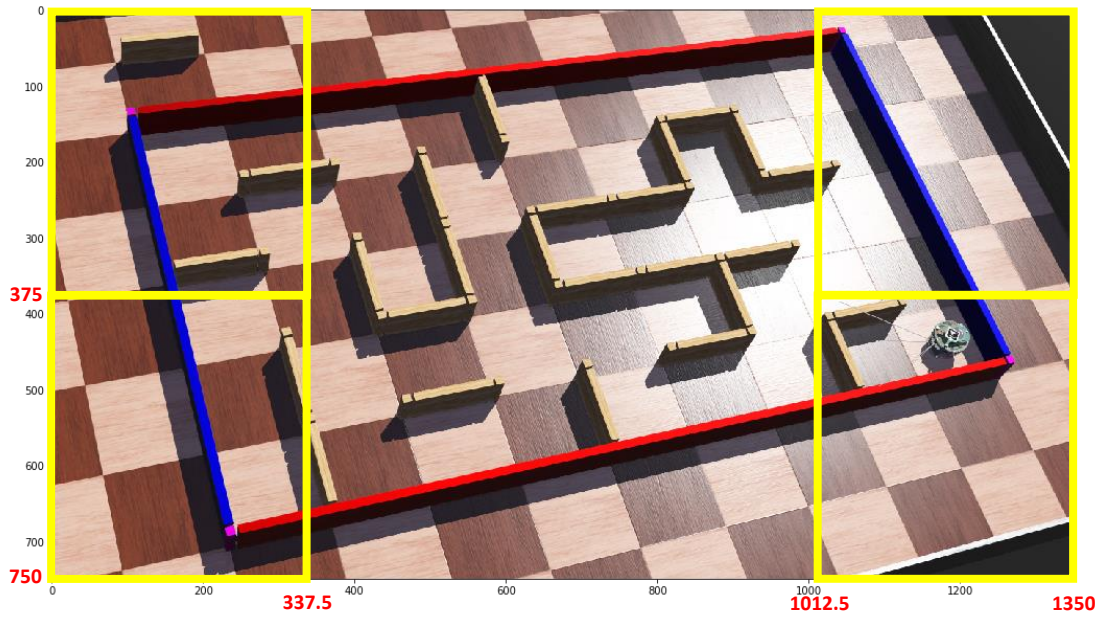

Fig. 8'. (Example 2) Location of the four cornerstones will be in the four regions, respectively

Robot:

14. You should not make any changes to the appearance of the robot in the world file of Webots for test.
15. You can change the location or heading of the robot for test.
16. The closeup image of the robot and the image of the maze will share the same orientation of viewpoint in the world file of Webots.

Implementation:

17. You are strongly recommended to use Python and OpenCV for this assignment.
18. Alternatively, you can use MATLAB for the implementation if you are not comfortable with Python (tutorials and support will only be provided for Python + OpenCV implementation).
19. If you use Python:
    - You must implement your program using Jupyter Notebook (*.ipynb) for the sake of assessment.
    - You must use python=3.7.7, opencv-python=3.4.2.17, and opencv-contrib-python=3.4.2.17 to avoid issues with different versions.
    - You must have a "requirements.txt" specifying all the libraries you used that are not installed along with python installation.
20. If you use MATLAB:
    - You must implement your program using Live Scripts (*.mlx) for the sake of assessment.
    - You must use MATLAB R2019b and should not use any libraries other than from the Image Processing Toolbox or the Computer Vision Toolbox.
21. The image file of the maze should be named "Maze.png". You should define a global variable at the beginning of your program indicating the path of this image file. E.g.,

    MAZE_FILE_NAME = "**/Maze.png"

where ** is the Relative path of the image file stored on your computer.

22. The image file of the robot should be named "Robot.png". You should define a global variable at the beginning of your program indicating the path of this image file. E.g.,

    ROBOT_FILE_NAME = "**/Robot.png"

    where ** is the Relative path of the image file stored on your computer.
23. The text file storing the found map should be named "MapFound.txt". You should define a global variable at the beginning of your controller file indicating the path of this text file. E.g.,

    MAP_FILE_NAME = "**/MapFound.txt"

    where ** is the Relative path of the text file stored on your computer.
24. The generated map must follow the convention defined in Phase B. A map using a different convention, even if essentially correct, will not get the marks associated with this task.
25. You can show all the results in one image or show them in separate images; we will mark based on the best results we see from running your program.

4.2. Hints:

1. Consult the lecturer/demonstrators if you are unclear about anything.
2. You can use a snipping tool with customisable resolutions (e.g., 1350 x 750, 750 x 750) for generating the images for test.
3. Thresholding can be used as a basic step for many of the tasks.
4. A priori knowledge can be used to make your program more robust.
5. You should use forward slash / or double backward slash \\ instead of single backward slash \ to define the path of the files.

5. Assessment:

5.1. Submission of your work

You should zip your project folder and rename it as "z*******_MTRN4110_PhaseC.zip" where ******* is your zID. Submit this zip file to Moodle.

In the folder, you should include two folders: the <worlds> folder and the <programs> folder.

In the <worlds> folder, you should include your world file, named as "z*******_MTRN4110_PhaseC.wbt" where ******* is your zID.

In the <programs> folder, you should include your vision program, named as "z*******_MTRN4110_PhaseC.ipynb" (for Python implementation) or "z*******_MTRN4110_PhaseC.mlx" (for MATLAB implementation) where ******* is your zID.

In the <programs> folder, you should also include the "requirements.txt" file specifying the libraries you use.

You should have the following folder structure in your submission:

```
Z*******_MTRN4110_PhaseC
  |--programs
  |    |--z*******_MTRN4110_PhaseC.ipynb(.mlx)
  |    |--requirements.txt
  |--worlds
  |    |--z*******_MTRN4110_PhaseC.wbt
  |    |--z*******_MTRN4110_PhaseC_2.wbt
  |    |--marker1.png
  |-- MapFound.txt
  |-- MapFound_2.txt
  |-- Maze.png
  |-- Maze_2.png
  |-- Robot.png
  |-- Robot_2.png
```

These are the essential folders and files you should include. You can also include other folders/files if needed but you should not change the name/location of these essential folders and files. It is your responsibility to make sure your submission is self-contained.

5.2. Marking criteria:

This assignment will contribute 15% to your final mark.
You will be assessed four times with different setups. Your final mark will be the average of the marks of the four attempts., i.e.,

$$mark_{final} = 25\% * mark_{attempt\_1} + 25\% * mark_{attempt\_2} + 25\% * mark_{attempt\_3} + 25\% * mark_{attempt\_4}$$

Each attempt will be assessed by using the following criteria.

Task	Description	Marking (out of 100%)
1	Read an image into the program and display it	+5% if all correct, otherwise no marks
2	Find the four cornerstones of the maze	+20% if all correct[1], otherwise • +5% each for correct[1] indication of one cornerstone
3	Perspective transform the maze of the original image to a rectangle image	+10% if all correct[2], otherwise no marks
4	Detect all the internal walls	+25% if all correct[3], otherwise • +25% - X% where X is the number of incorrect[4] indications
5	Detect the location and heading of the robot	+30% if all correct, otherwise • +10% for correct[5] indication of the location of the robot • +20% for correct[6] indication of the heading of the robot ○ For heading 10% can be given if the use of "^", ">", "v" and "<" is correct but its centre is not within the surface of the robot
6	Generate a map and write it to a text file	+10% if all correct, otherwise • (8% maximum) ○ +2% each for correctly writing a line (excluding the outer walls)

[1]Correct means the centre of the indication is within the top surface of the cornerstone
[2]Correct means the perspective transformed image is consistent with the area defined by the four cornerstones found
[3]Correct means the centre of the indication is within the top surface of the wall
[4]Incorrect means an existing wall is not correctly indicated or a non-existing wall is indicated
[5]Correct means the centre of the indication is within the surface of the robot
[6]Correct means the centre of the indication is within the surface of the robot and the use of "^", ">", "v" and "<" is corresponding to the true heading of the robot

5.3. Deadline

The submission deadline is 17:00 AEST 2 August 2020 (Sunday Week 9).

If your assignment is submitted after this date, each 1 hour it is late reduces the maximum mark it can achieve by 1%. For example if an assignment worth 74% were submitted 10 hours late, the late submission would have no effect. If the same assignment were submitted 30 hours late, it would be awarded 70%, the maximum mark it can achieve at that time.

5.4. Progress Check

You will have your progress checked with your demonstrator in a 5 min meeting on the afternoon of Thursday Week 8 (or another time on weekdays Week 8 by appointment).

To pass the progress check, you must demonstrate that you are able to correctly detect at least one internal wall from the given image.

6. Additional Resources:

- Webots user guide: https://cyberbotics.com/doc/guide/index
- Webots reference manual: https://cyberbotics.com/doc/reference/index
- Python Tutorials: https://github.com/drliaowu/MTRN4110_20T2_Python_Tutorials
- Vision I Tutorial: https://github.com/drliaowu/MTRN4110_20T2_Lecture_Vision_I
- Vision II Tutorial: https://github.com/drliaowu/MTRN4110_20T2_Lecture_Vision_II
- OpenCV-Python Tutorials: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html
- Python File I/O: https://www.tutorialspoint.com/python/python_files_io.htm