



MTRN4110 22T2 Phase C Task Description

(Week 7-9)

Written By Leo Wu

Released 11/7/2022

1. Overview of the Course Project:

The main project of MTRN4110 22T2 is a simulation-based project adapted from the [Micromouse](#) competition. [Webots](#) will be used as the simulation platform throughout the project. 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 60% to your final mark in this course.

The project consists of four sequential phases which are connected, but attempting one phase is not dependent on the completion of another:

- Phase A: Driving and Perception (Week 1-3, 14%, individual)
- Phase B: Path Planning (Week 4-6, 14%, individual)
- Phase C: Computer Vision (Week 7-9, 14%, individual)
- Phase D: Integration and Improvement (Week 10-12, 18%, group)

This document will describe the tasks of **Phase C**.



2. Overview of Phase C – Computer Vision:

Phase C aims to develop a vision program to build a map for the maze-solving robot. You are required to complete the tasks of this phase by **13:00 Monday, Week 10 (1 August)**.

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;
- detect the position of a target; and
- generate a map of the maze with a robot and a target 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

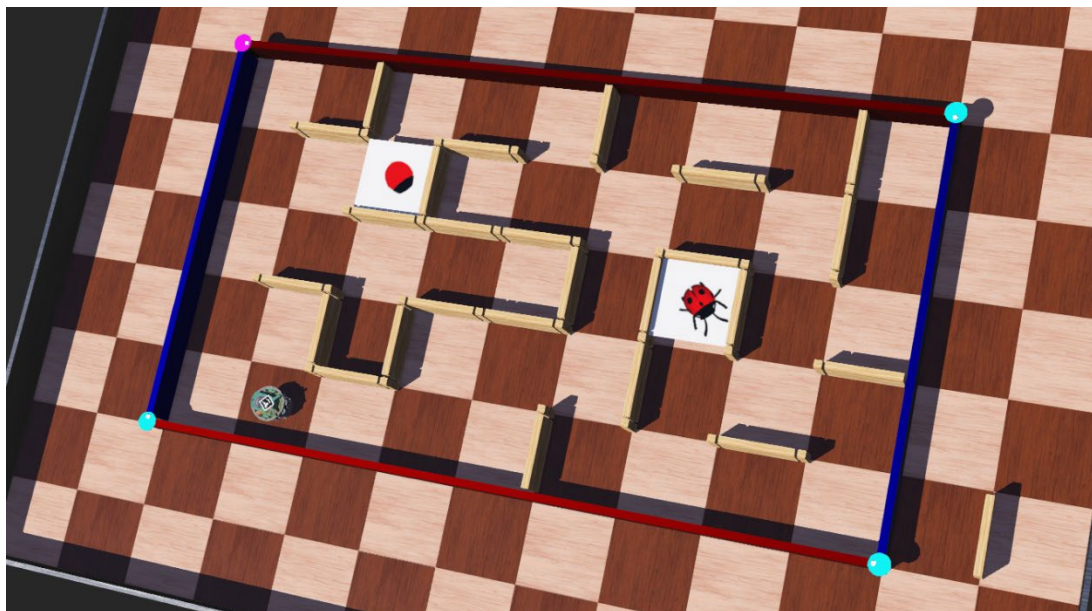


Fig. 1'. (Example 2) Maze.png



Fig. 2. (Example 1) Robot.png

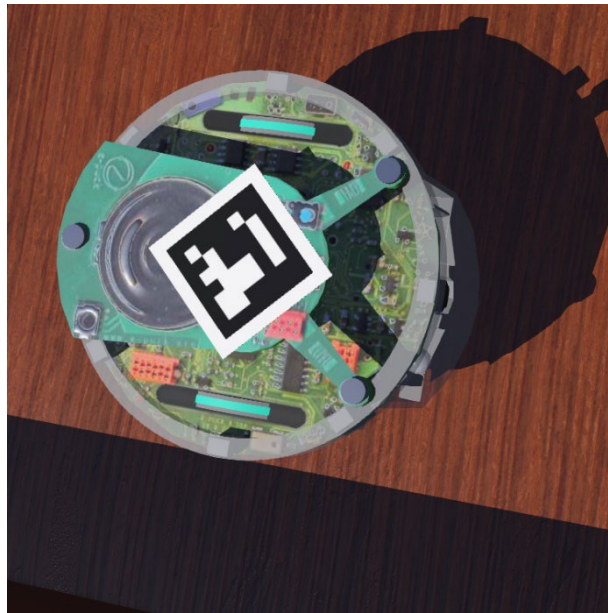


Fig. 2'. (Example 2) Robot.png



Fig. 3. (Example 1 & 2) Ladybug_small.png

3.2. Find the four ordered cornerstones of the maze

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

Below is an example indication (highly recommended). Other ways of indication are also acceptable if the following conditions are met:

- The markers are **easily** recognisable;
- The **centre** of the marker is **within** the top surface of each cornerstone;
- The markers for the **three cyan** cornerstones are in the **same** colour;
- The marker for the **pink** cornerstone is in a **different** colour from the other three.

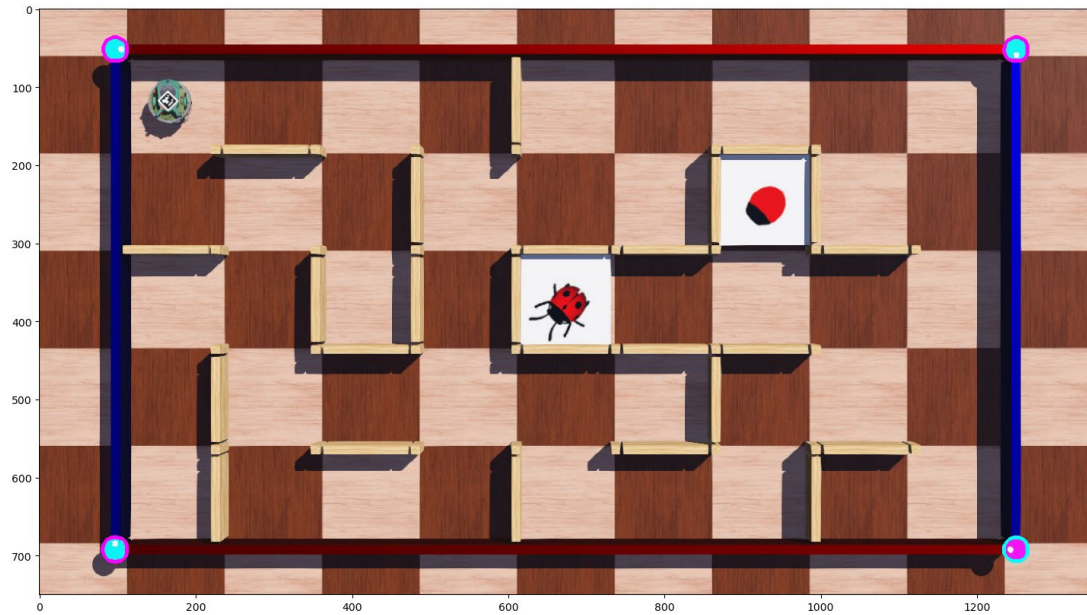


Fig. 4. (Example 1) Indication of the four ordered cornerstones

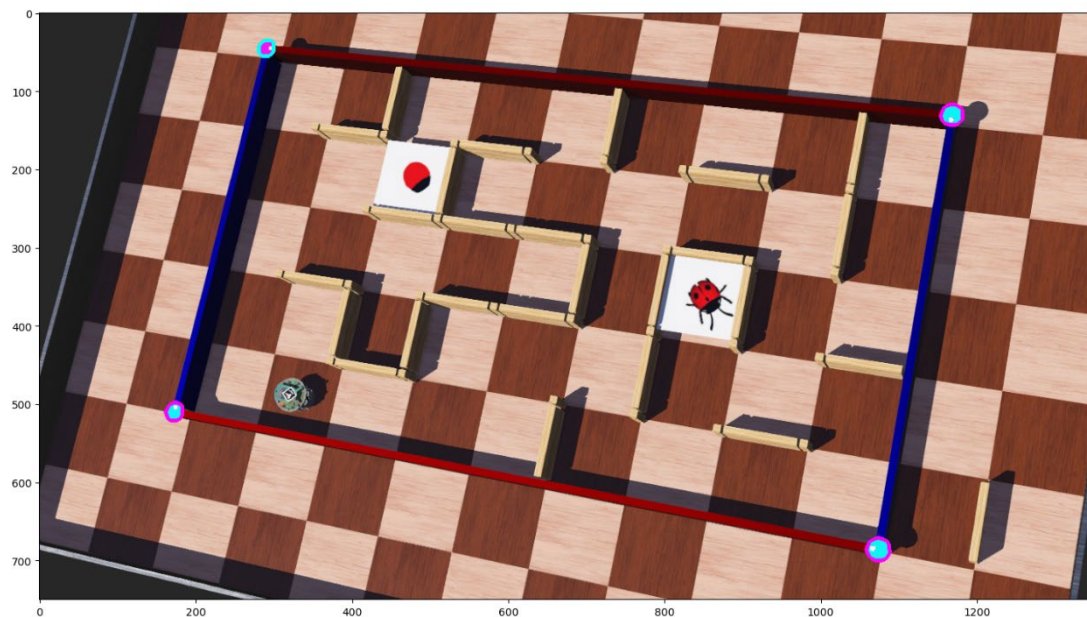


Fig. 4'. (Example 2) Indication of the four ordered cornerstones

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

Use perspective transformation to convert the maze from the original image into a rectangle image **whose four vertices are the four cornerstones**. The following conditions should be met:

- The **width:height** ratio of the transformed image should be **9:5**;
- The **bottom-right** corner of the transformed image should be the **pink** cornerstone;
- The transformed image should **not** be distorted.

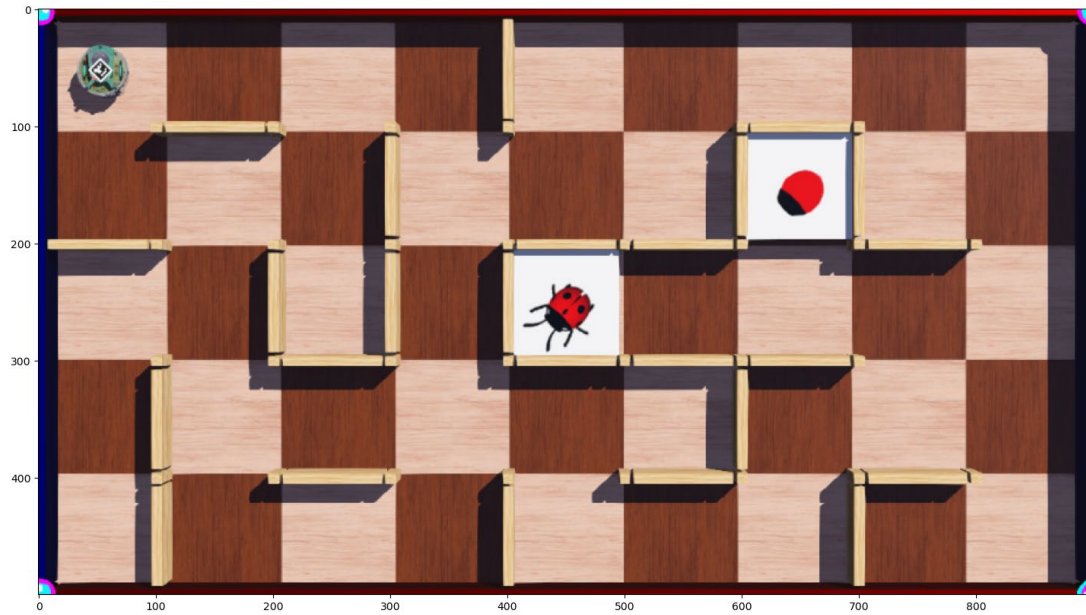


Fig. 5. (Example 1) Image after perspective transform

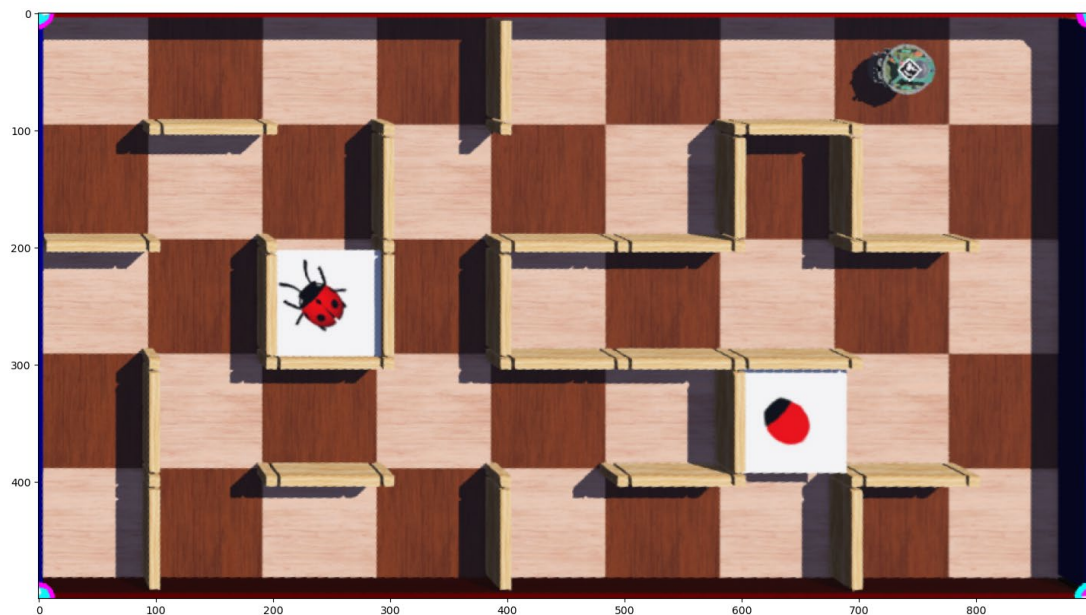


Fig. 5'. (Example 2) Image after perspective transform

3.4. Detect all the internal walls

Detect all the **internal** walls and indicate them on the **transformed** image.

Below is an example indication (highly recommended). Other ways of indication are also acceptable if the following conditions are met:

- The markers are **easily** recognisable;
- Only the **top surface** of each wall is marked (the side surfaces should **not** be marked);

- The **centre** of the marker is **within** the top surface of each wall;
- External walls can be marked but will **not** be considered.
- Places, where there is not a wall, should **not** be marked.

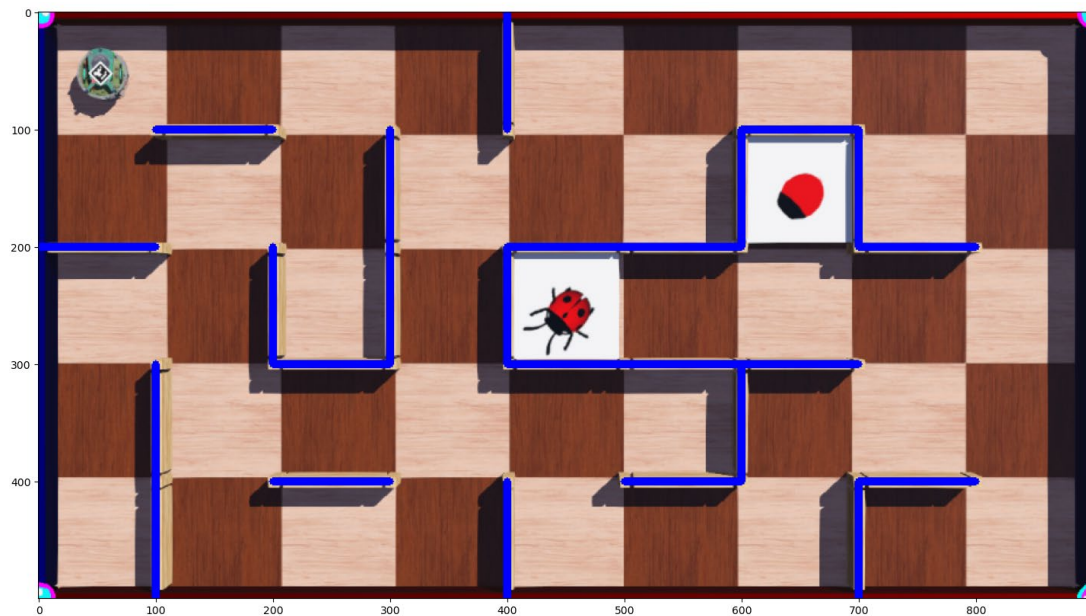


Fig. 6. (Example 1) Indication of internal walls

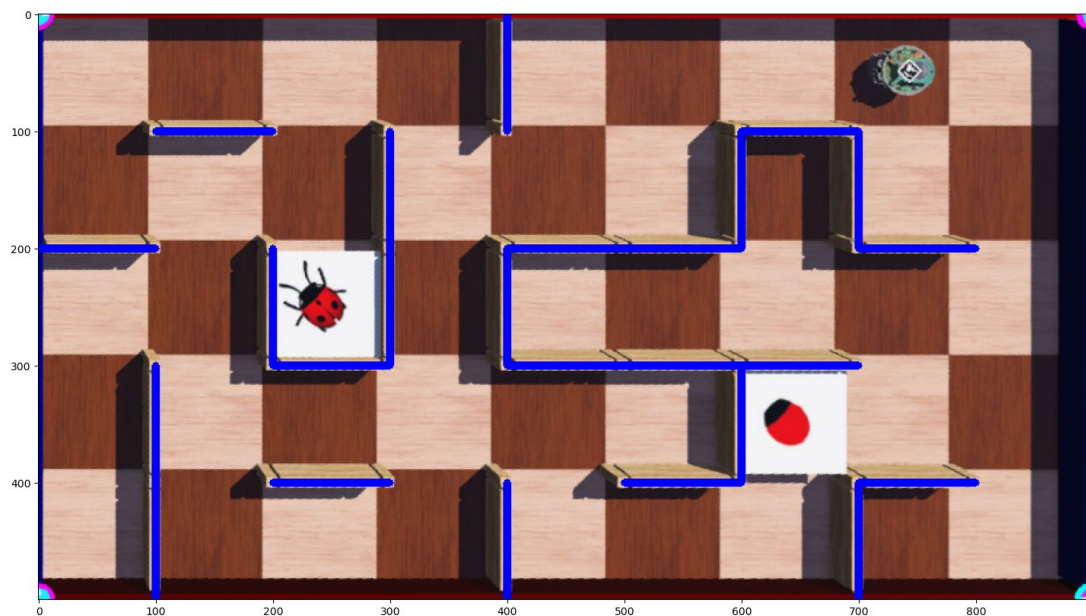


Fig. 6'. (Example 2) Indication of internal walls

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.

Below is an example indication (highly recommended). Other ways of indication are also acceptable if the following conditions are met:

- The markers are **easily** recognisable;
- The **centre** of the marker representing the **location** is **within** the cell that the robot is at;
- The **centre** of the marker representing the **heading** is also **within** the cell that the robot is at;
- The marker representing the **heading** must use “^”, “>”, “v” and “<” to describe “N”, “E”, “S” and “W”, respectively, as defined in Phase B. **N/E/S/W should be referenced when the pink cornerstone is on the bottom-right of the maze.**

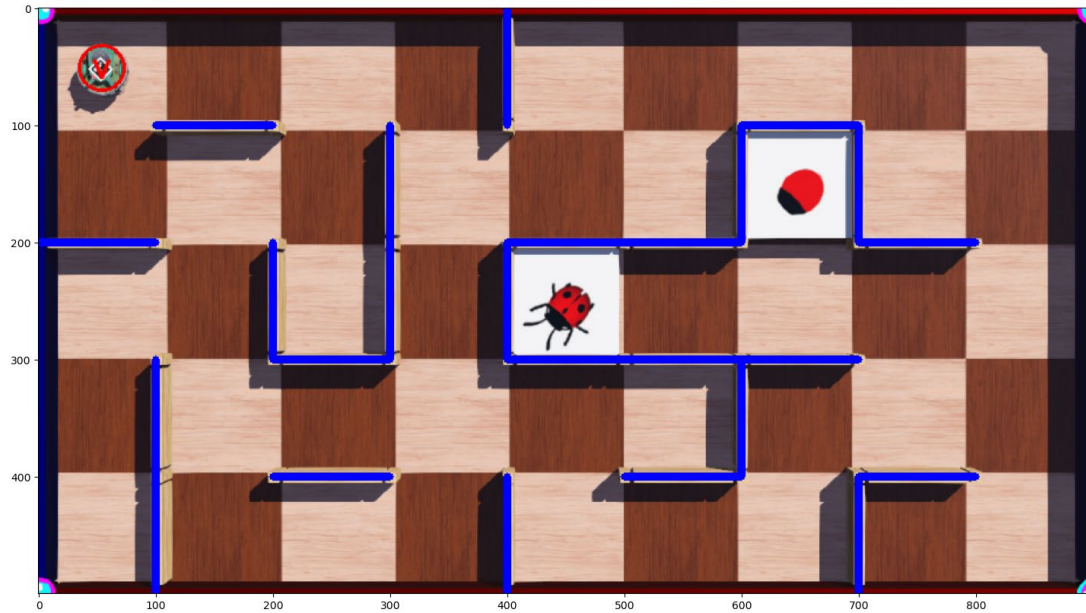


Fig. 7. (Example 1) Indication of location and heading of robot

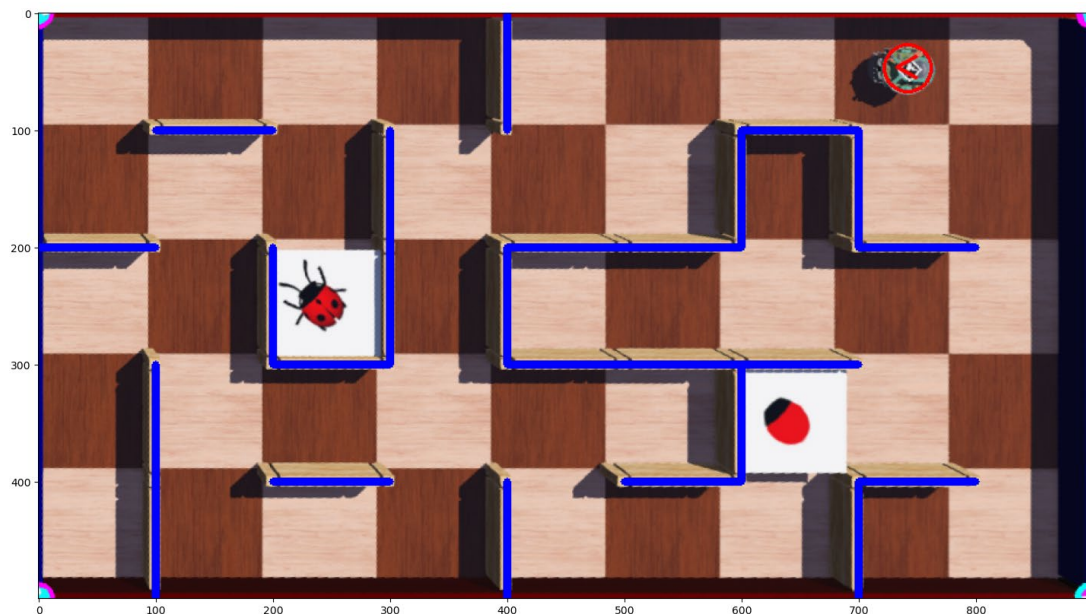


Fig. 7'. (Example 2) Indication of location and heading of robot

3.6. Detect the position of the true target

Detect the **position** of the **true** target and indicate it on the **transformed** image.

Below is an example indication (highly recommended). Other ways of indication are also acceptable if the following conditions are met:

- The markers are **easily** recognisable;
- The **centre** of the marker representing the **position** is **within** the cell that the true target is in;
- The marker representing the **position** must contain an “x”, as defined in Phase B;
- Only the **true** target (the Webots ladybug) should be marked; the **fake** target (the hand-drawn ladybug) should **not** be marked.
- The orientation of the target does **not** need to be indicated.

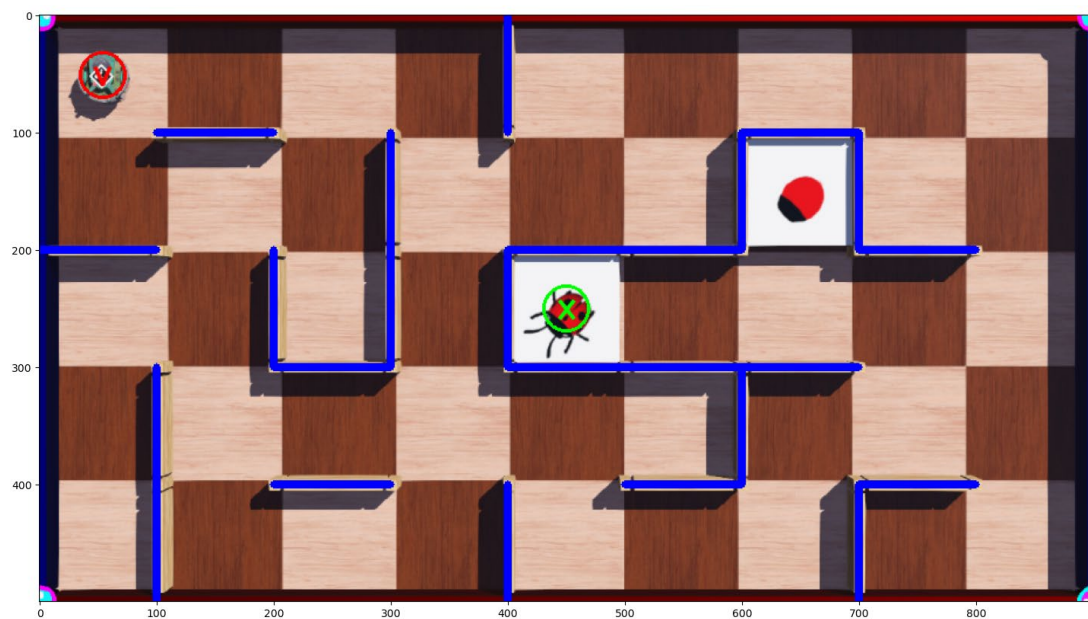
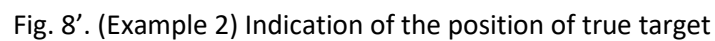


Fig. 8. (Example 1) Indication of the position of the true target



Based on the processed image, generate a map containing **all the walls (both internal and external)**, the **location** and **heading** of the **robot**, and the **position** of the **true target**. Write the map into a text file named **"MapBuilt.txt"**.

[illegible]

Fig. 8. (Example 1) Map found by the program

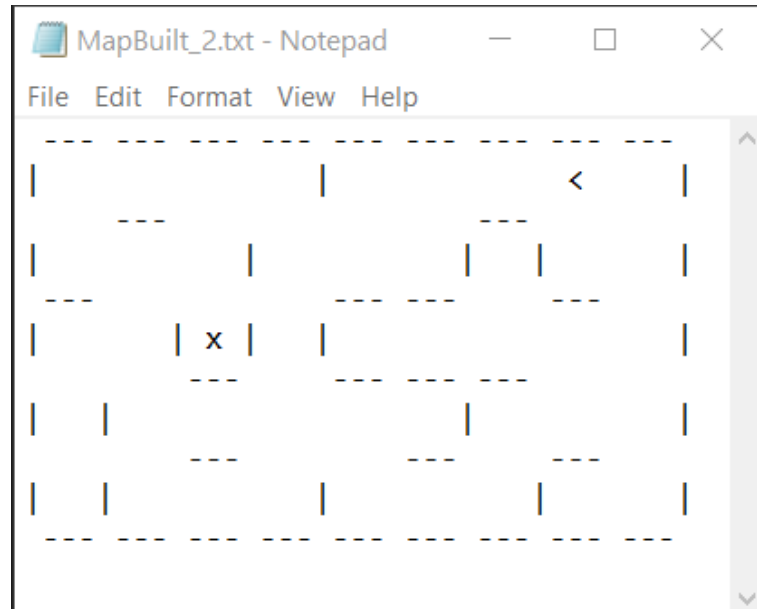


Fig. 8'. (Example 2) Map found by the program

3.8. Task summary:

Task	Description
1	Read in an image and display it in RGB mode
2	Find the four ordered 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	Detect the position of the true target
7	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. You will also be given an image named "**Ladybug_small.png**" shared by the two examples.
3. These are for your practice. For assessment, you may be tested with **different** images in which the maze layout, the initial location and heading of the robot, and the positions of the true and fake targets may be **different**.
4. A world file of Webots "**z1234567_MTRN4110_PhaseC.wbt**" containing a maze, a robot, a true target, and a fake target will be provided to you so that you can generate different images for your testing.
5. Another world file of Webots "**z1234567_MTRN4110_PhaseC_2.wbt**" will also be provided to you to show you how example 2 images are generated.
6. The size of the "**Maze.png**" will always be **1350 x 750** in pixels.
7. The size of the "**Robot.png**" will always be **750 x 750** in pixels.
8. The size of the "**Ladybug_small.png**" will always be **100 x 100** in pixels.
9. The maze will always be the same size (**5** rows by **9** columns) with **closed** borders.
10. The initial location of the robot will **always** be at the **centre** of a cell.
11. The initial heading of the robot will **always** be towards one of the **four** directions (North, East, South, West).
12. The layout of the maze for assessment may be **different** from the example.
13. The colours of the objects will always be the **same** as in the example.
14. The lighting will be the same as in the example world file of Webots, but it may appear **differently** with different perspectives.
15. The perspective of the image may be **different** from the example. However, the four cornerstones will be restricted within the following four regions (A – D), respectively.
16. The location of the **pink** cornerstone will either be in **Region A** or **Region D**, but **not** in Region B or Region C.
17. The viewpoint of the "**Maze.png**" will vary between tests, but will always ensure the top surfaces of the cornerstones, walls, robot, and target are clearly visible.

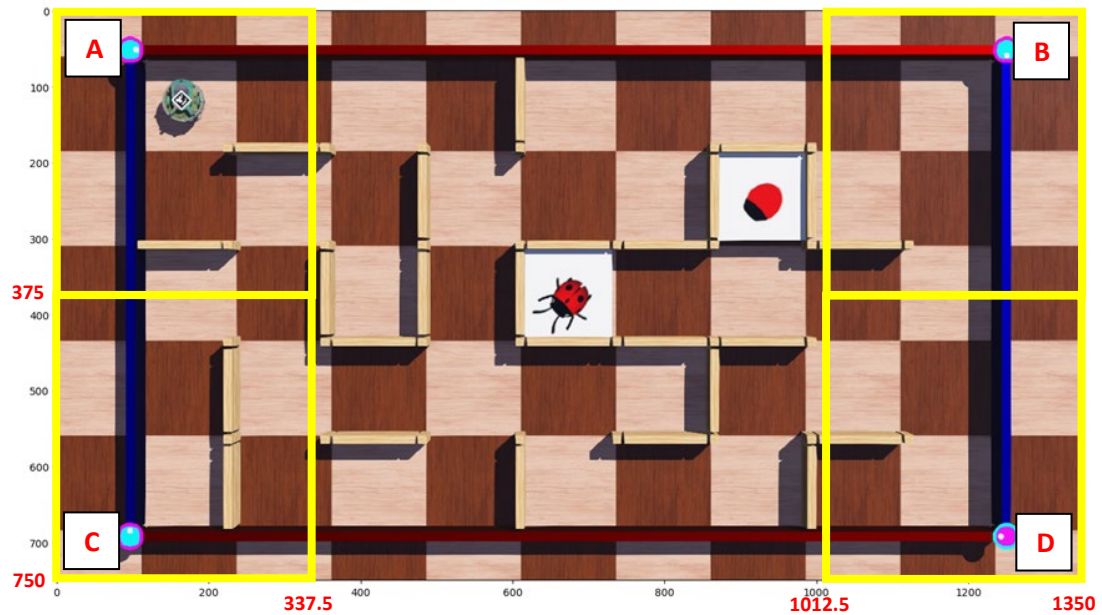


Fig. 9. (Example 1) Location of the four cornerstones will be in the four regions, respectively

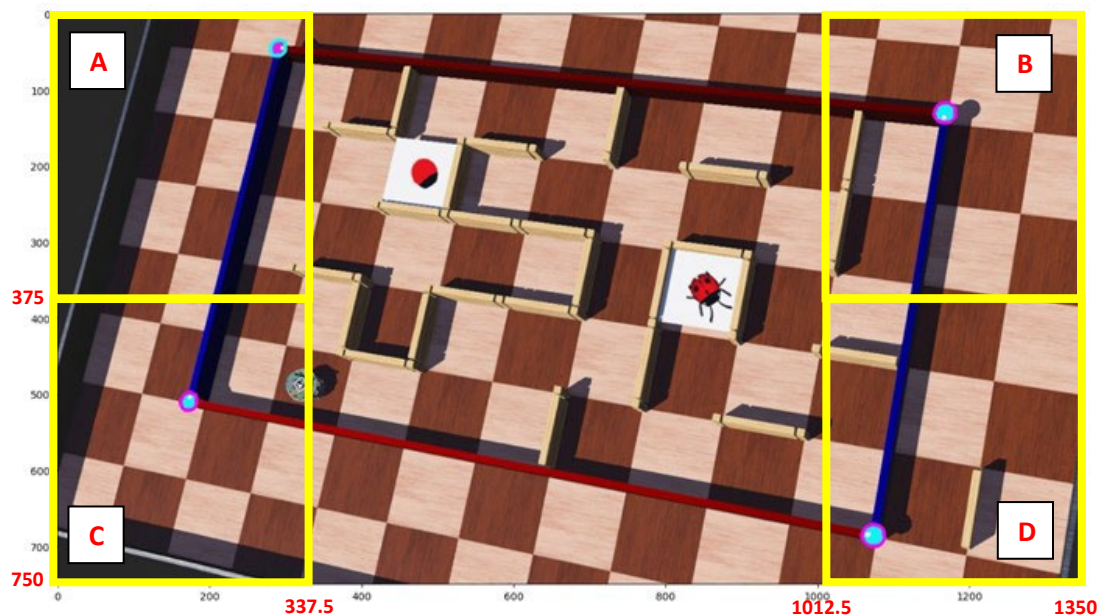


Fig. 9'. (Example 2) Location of the four cornerstones will be in the four regions, respectively

Robot and Target:

18. You should **not** make any changes to the appearance of the robot in the world file of Webots for testing.
19. You **can** change the location or heading of the robot for testing.
20. The closeup image of the robot and the original image of the maze will share the **same** orientation in the **viewpoint** setting of the world file in Webots.
21. The orientation of the true target can be rotated for 0°, 90°, 180°, or 270° in the world file in Webots.



22. The orientation of the fake target will be the same as the true target in the world file in Webots.

Implementation:

23. You **must** use **Python + OpenCV** for this assignment.
24. **No** other languages are allowed for this phase.
25. Tutorials and support will **only** be provided for **Python + OpenCV** implementation.
26. You **must** implement your program using **Jupyter Notebook (*.ipynb)** for the sake of assessment.
27. You **must** use the following versions of packages to avoid version-related issues in assessment (you can create a virtual environment for this following the tutorial):
- `python==3.7.10`
 - `matplotlib==3.2.2,`
 - `opencv-python==3.4.2.17`
 - `opencv-contrib-python==3.4.2.17`
 - `ipykernel==5.3.2`
 - `notebook==6.4.1`
- You **must** have a “requirements.txt” specifying **all** the libraries you used that are not installed along with python installation. **A penalty will be applied if an error occurs due to wrong library usage.**
28. 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 the relative path `../Maze.png` will allow you to access the file **if** the folder structure specified in Section 5.1 is followed.
29. 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 the relative path `../Robot.png` will allow you to access the file **if** the folder structure specified in Section 5.1 is followed.
30. The image file of the ladybug should be named “**Ladybug_small.png**”. You should define a global variable at the beginning of your program indicating the path of this image file. E.g.,
- `IMAGE_LADYBUG_FILE_NAME = ‘../Ladybug_small.png’`
- where the relative path `../Ladybug_small.png` will allow you to access the file **if** the folder structure specified in Section 5.1 is followed.
31. The text file storing the built map should be named “**MapBuilt.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 = “../MapBuilt.txt”`
- where the relative path `../MapBuilt.txt` will allow you to access the file **if** the folder structure specified in Section 5.1 is followed.
32. The generated map **must** follow the convention defined in Phase B. The bottom-right of the maze **must** correspond to the pink cornerstone. **A map using a different convention, even if essentially correct, will not get the marks associated with the task.**



33. You **should** show the **intermediate** results step by step (like figures 1 & 4-8) **rather than** showing all the results in one single image (figure 8). We will mark the tasks based on the best results we see from running your program. **Showing intermediate results allows you to get partial marks if not all the results are displayed correctly. If you only show one final image, you risk getting zero marks as the assessor cannot see any intermediate results if your last image is not output correctly.**
34. You should **explicitly** define a variable for the **transformed image** after perspective transform. The tutor may replace it with a correct image for the rest marking if your perspective transform was wrong, provided this variable can be easily found.

4.2. Hints:

1. Consult the lecturer/demonstrators if you are unclear about anything.
2. You can use a snipping tool (e.g., Screenshot Captor) with customisable resolutions (e.g., 1350 x 750, 750 x 750) for generating the testing images.
3. You can also get a snapshot in Webots first and then crop the image into the desired resolution.
4. Thresholding can be used as a basic step for many of the tasks.
5. A priori knowledge about the maze, the robot, and the target can be used for some tasks.
6. Follow the first [Python Tutorial](#) to install and configure your python and the virtual environment. Develop your code in the virtual environment. If packages become messy during development, create a new virtual environment with the minimum needed packages installed to tidy up.



5. Assessment:

5.1. Submission of your work

You should zip your project folder and name it as “z*****_MTRN4110_PhaseC.zip”, where z***** is your zID. Submit this zip file to Moodle.

You should include two folders in the folder: the <worlds> folder and the <programs> folder.

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

In the <programs> folder, you should include your vision program, named as “z*****_MTRN4110_PhaseC.ipynb”, where z***** 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.zip
|-- z*****_MTRN4110_PhaseC
    |--programs
    |     |--z*****_MTRN4110_PhaseC.ipynb
    |     |--requirements.txt
    |--worlds
    |     |--z*****_MTRN4110_PhaseC.wbt
    |     |--z*****_MTRN4110_PhaseC_2.wbt
    |     |--ladybug.png
    |     |--ladybug_fake.png
    |     |--marker1.png
    |-- Ladybug_small.png
    |-- MapBuilt.txt
    |-- MapBuilt_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 have other files if needed, but you should not change the name/location of these folders or files. Besides, only the finalised version of your code should be included. It is your responsibility to make sure your submission is self-contained and up-to-date.

As we have developed an automatic tool to do the plagiarism check on your code, it is CRUCIAL that you strictly follow the file structure specified above. Any violation in the submission format would incur a 5% penalty (as if one-day late), as that adds much more work to the assessors (we will still do the plagiarism check!).



5.2. Marking criteria:

This assignment will contribute **14%** to your final mark (including 1% from Progress Check).

You will be assessed **six** times with different setups. Among them, two tests will be on the examples given to you for practice. Your final mark will be calculated as the following:

$$\text{mark}_{\text{final}} = (\text{mark}_{\text{example1}} + \text{mark}_{\text{example2}} + \text{mark}_{\text{newtest1}} + \text{mark}_{\text{newtest2}} + \text{mark}_{\text{newtest3}} + \text{mark}_{\text{newtest4}}) / 6$$

Each attempt will be assessed by using the following criteria.

Task	Description	Marking (out of 100%)
1	Read in an image and display it in RGB mode	+5% if all correct, otherwise no marks
2	Find the four ordered cornerstones of the maze	+20% if all correct ¹ , otherwise <ul style="list-style-type: none">+5% each for correct¹ indication of one cornerstone
3	Perspective transform the maze from 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 <ul style="list-style-type: none">+25% - X% where X is the number of incorrect⁴ indications
5	Detect the location and heading of the robot	+20% if all correct ⁵ , otherwise <ul style="list-style-type: none">+10% for correct indication of the location of the robot+10% for correct indication of the heading of the robot
6	Detect the position of the true target	+10% if all correct ⁶
7	Generate a map and write it to a text file	+10% if all correct ⁷ , otherwise <ul style="list-style-type: none">(8% maximum)<ul style="list-style-type: none">+2% each for correctly writing a line (excluding the outer walls)

¹Correct means the requirements listed are all met (if the criteria on colour are not satisfied while the other conditions are met, +15% maximum)

²Correct means the requirements listed are all met

³Correct means the requirements listed are all met

⁴Incorrect means an existing wall is not correctly indicated, or a non-existing wall is indicated

⁵Correct means the requirements listed are all met

⁶Correct means the requirements listed are all met

⁷Correct means the map written to the file is correctly representing the given file and in the required format

5.3. Deadline

The submission will be open from **13:00 AEST 25 July 2022 (Monday Week 9)** and the deadline is **13:00 AEST 1 August 2022 (Monday Week 10)**.

We will apply a university-wide late policy that has been specified by UNSW and MME (refer to the course outline):



Late policy

Work submitted late without an approved extension by the course coordinator or delegated authority is subject to a late penalty of five percent (5%) of the maximum mark possible for that assessment item, per calendar day.

The late penalty is applied per calendar day (including weekends and public holidays) that the assessment is overdue. There is no pro-rata of the late penalty for submissions made part way through a day. This is for all assessments where a penalty applies.

Work submitted after five days (120 hours) will not be accepted and a mark of zero will be awarded for that assessment item.

For example:

- Your course has an assessment task worth a total of 100 marks.
- You submit the assessment 2 days (or part thereof) late (i.e. from 24-48 hours after the deadline).
- The submission is graded and awarded a mark of 65/100.
- A late penalty of 10 marks is deducted from your awarded mark (2 days @ 5% of 100 marks).
- Your adjusted final score is 55/100.

Note the 5% penalty will be multiplied by the maximum possible mark and applied directly to the **awarded mark** (refer to the example).

Students are expected to manage their time to meet deadlines or request extensions (apply for special consideration) as early as possible before the deadline.

5.4. Progress Check

You will have your progress checked with your demonstrator in a **5 min** meeting in your **Week 8 workshop session**.

During the session, please show your progress to your demonstrator in person. For the online workshop, please share your screen with your demonstrator. If you don't know how to do this, please watch this short video: [How to share your screen in a Microsoft Teams meeting](#)
The progress check makes up **1%** of the overall course mark (**included in the 14%**).

To pass the progress check, you must demonstrate that you can **correctly detect all the four ordered cornerstones and indicate them following the requirements** for both given examples.

5.5. Plagiarism

If you are unclear about the definition of plagiarism, please refer to [What is Plagiarism? | UNSW Current Students](#).

You would get **zero marks** for the assignment if you were found:

- Knowingly providing your work to anyone and it was subsequently submitted (by anyone), or
- Copying or submitting any other persons' work, **including code from previous students of this course** (except general public open-source libraries/code, for which you should explicitly cite the source wherever applicable).

You will be notified and allowed to justify your case before such a penalty is applied.



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_Python_Tutorials
- Vision I Tutorial: https://github.com/drliaowu/MTRN4110_Lecture_Vision_I
- Vision II Tutorial: https://github.com/drliaowu/MTRN4110_Lecture_Vision_II
- OpenCV-Python Tutorials: https://docs.opencv.org/3.4.2/d6/d00/tutorial_py_root.html
- Python File I/O: https://www.tutorialspoint.com/python/python_files_io.htm