2	Ц		Π	TI	Br
		Enginee	ering a	better	tomorrow



Rulebook		

Task 0

CoppeliaSim Tutorials

Welcome to NB theme!

1A - Explore OpenCV

1B - Detect and Encode Maze

1C - Design Ball Balance Platform

Task 2
Task 3
Task 4
Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

Task 1B

eYRC 2020-21: Nirikshak Bot (NB)

Detect and Encode Maze

[Last Updated on: 19th October 2020, 16:39 Hrs]

- 1. Understanding Mazes
 - Unique Encoding to represent Cells in a Maze
- 2. Implementation of Solution
 - Given
 - Problem Statement
 - Instructions
 - Running your solution
- 3. Testing the Solution
- Submission Instructions

The aim of this task is as follows:

- Detecting a maze from a given image using OpenCV techniques
- Encoding the maze as a matrix in a CSV file

This task is divided into **three** parts:

- 1. Understanding Mazes
- 2. Implementation of Solution
- 3. Testing the Solution

Make sure you go through the instructions thoroughly and in a sequential order. It will help you understand the tasks better.

Welcome to NB theme! Rulebook Task 0 Task 1 CoppeliaSim Tutorials 1A - Explore OpenCV 1B - Detect and Encode Maze 1C - Design Ball Balance Platform Task 2 Task 3 Task 4 Task 5 Practice Task Instructions for Task 6 Task 6 Scene Details Coding Standard Git and GitHub Live Session 1 - 24th October 2020 Live Session 2 - 21st November 2020 Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

1B - Detect and Encode Maze - eYRC 2020-21: Nirikshak Bot (NB)

1. Understanding Mazes

In this section, we will first learn about what mazes are and how they are constructed.

• We start with a **Grid** first. Shown below in Figure 1 is a representation of a **5 x 5** grid where *firs* number represent number of rows and the second number shows number of columns in the grid

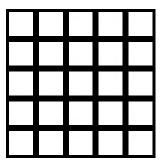


Figure 1: 5 x 5 Grid

- The smallest building element of a grid is defined as a Cell.
- A *Cell* is defined as a square with **four** sides where a *Wall* (represented by bold black lines around the cell) may or may not exist at each of the sides.
- A grid can be converted into a **Maze** by carving out passages through the grid. An example maze carved out of Figure 1 is shown in Figure 2.

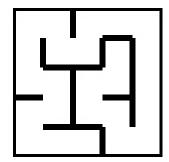


Figure 2: Example Maze from 5x5 Grid

- As we have converted grid of Figure 1 into a Maze in Figure 2, this maze also has dimensions c
 5 x 5, hence there are 25 cells in each of them.
- For example, consider the following cells in Figure 3.

Live Session 2 - 21st November 2020 Live Session 3 - 12th December 2020 Live Session 4 - 10th January 2021

Changelog

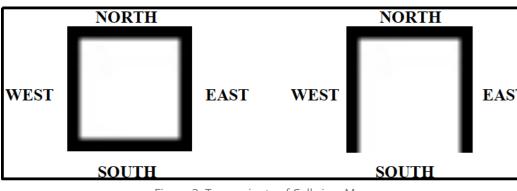


Figure 3: Two variants of Cells in a Maze

- The dark bands indicate presence of walls. **NORTH**, **SOUTH**, **EAST** and **WEST** are called **directions** in the cell.
- The first cell in Figure 3 has **no open** passages in either of the four directions of the cell while the second cell has an open passage in **SOUTH** and the passages to the other three directions are blocked by walls. Since each cell have four sides we can have a total of **4**² = **16** different ways in which a cell may be represented. Two of them are shown in Figure 3 and rest of the 14 configurations are shown in Figure 4.

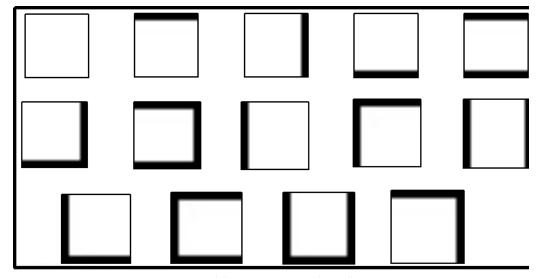


Figure 4: 14 different variants of a Cell in a Maze

• A maze as given in Figure 2, is nothing but a collection of different types of cells as given in Figure 4.

Welcome to NB theme!

Rulebook

Task 0

Task 1

CoppeliaSim Tutorials

1A - Explore OpenCV

1B - Detect and Encode Maze

1C - Design Ball Balance Platform

Task 2
Task 3
Task 4
Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

- As discussed above, there are a total of **16** different types of cells.
- Each cell can be represented using a unique *Cell Number* between **0** to **15**. This number is assigned on the basis of wall configuration of the cell.
- In Figure 5, we have assigned a binary weight to each of the directions of the cell. The cell
 number for each type of cell can be calculated by adding the weights for the directions in whice
 a wall exists and ignoring the weights of the directions where wall doesn't exist.
- For example, the *cell number* for cell in Figure 5 is Weight of North wall (W_N) + Weight of West wall (W_w) = 2 + 1 = 3.

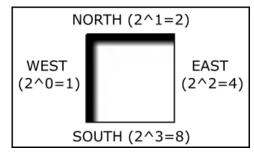


Figure 5: Cell Number calculation based on weights

• Similarly the *cell numbers* for the rest of the cell types can be calculated as given in Figure 6.

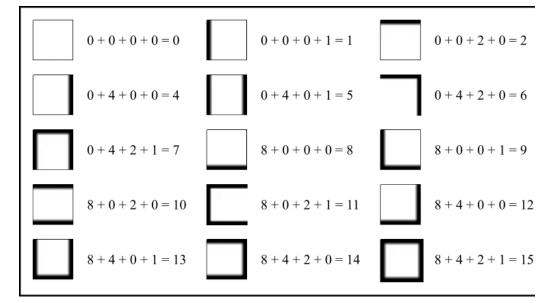


Figure 6: Cell Numbers for different types of cells

Welcome to NB theme! Rulebook Task 0 Task 1 CoppeliaSim Tutorials 1A - Explore OpenCV 1B - Detect and Encode Maze 1C - Design Ball Balance Platform Task 2 Task 3 Task 4 Task 5 Practice Task Instructions for Task 6 Task 6 Scene Details Coding Standard Git and GitHub Live Session 1 - 24th October 2020 Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

1B - Detect and Encode Maze - eYRC 2020-21: Nirikshak Bot (NB)

- You will have noticed that the cell numbers are completely unambiguous and unique.
- Cell numbers are significant because these allow us to represent a maze as a 2D array in whic
 each array element represents a cell of a maze by its corresponding cell number (as shown in
 Figure 7).

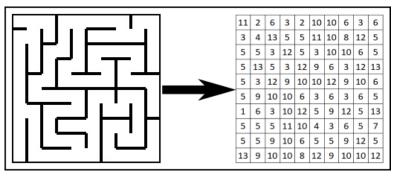


Figure 7: 2D array for a Maze using the Cell Numbers

2. Implementation of Solution

- Download the following zip file containing the files for Task 1B. Right-click on the hyperlink an select **Save Link As...** option to download.
 - task_1b_detect_and_encode_maze.zip
- You will find the following files/folder in the zip file:
 - o test_cases folder it contains all the maze images you will use to test your solution
 - o task 1b.py file you will implement your solution in this file
 - o test_task_1b.py file you will test your solution using this file
 - o task_1b_cardinal.pyc this file will help in testing the solution

Note: Do not tamper with the *task_1b_cardinal.pyc* file. It is required to be in the same directory / folder as the *test_task_1b.py* in order to test your solution.

Given

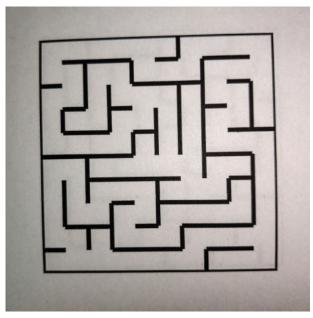
Welcome to NB theme! Rulebook Task 0 Task 1 CoppeliaSim Tutorials 1A - Explore OpenCV 1B - Detect and Encode Maze 1C - Design Ball Balance Platform Task 2 Task 3 Task 4 Task 5 Practice Task Instructions for Task 6 Task 6 Scene Details **Coding Standard** Git and GitHub Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog



A set of 10 maze images are given in the test_cases folder. An example image is shown in Figure 8.

Figure 8: Example Maze Image

Problem Statement

- Use OpenCV techniques to detect the maze in the above given image and convert it into a 2D maze array with appropriate Cell Numbers as shown in Figure 7.
- The maze_array is to be generated in the form of a nested list.
- For example, if the given maze image is as in Figure 7, the maze_array will be:

```
maze_array = [[11, 2, 6, 3, 2, 10, 10, 6, 3, 6],
              [3, 4, 13, 5, 5, 11, 10, 8, 12, 5],
              [5, 5, 3, 12, 5, 3, 10, 10, 6, 5],
              [5, 13, 5, 3, 12, 9, 6, 3, 12, 13],
              [5, 3, 12, 9, 10, 10, 12, 9, 10, 6],
              [5, 9, 10, 10, 6, 3, 6, 3, 6, 5],
              [1, 6, 3, 10, 12, 5, 9, 12, 5, 13],
              [5, 5, 5, 11, 10, 4, 3, 6, 5, 7],
              [5, 5, 9, 10, 6, 5, 5, 9, 12, 5],
              [13, 9, 10, 10, 8, 12, 9, 10, 10, 12]]
```

Welcome to NB theme!

Rulebook

Task 0

Task 1

CoppeliaSim Tutorials

1A - Explore OpenCV

1B - Detect and Encode Maze

1C - Design Ball Balance Platform

 Task 2
 2

 Task 3
 2

 Task 4
 2

 Task 5
 2

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

Instructions

- There are **two** functions pre-written in **task_1b.py** which you have to modify.
- The first function is applyPerspectiveTransform().

Function Name	applyPerspectiveTransform()
Purpose	Takes a maze test case image as input and applies a Perspective Transfrom on it to isolate the maze
Input Arguments	<pre>input_img : [numpy array] maze image in the form of a numpy array</pre>
Output Arguments	<pre>warped_img : [numpy array] resultant warped maze image after applying Perspective Transform</pre>
Example Call	warped_img = applyPerspectiveTransform(input_img)

• Figure 9 shows the **warped_img** output from the function **applyPerspectiveTransform** on an example maze image.

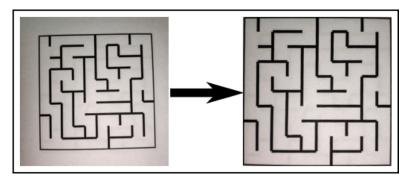


Figure 9: Output of applyPerspectiveTransform function

• The second function is **detectMaze()**.

Function Name	detectMaze()
Purpose	Takes the warped maze image as input and returns the maze encoded in form of a 2D array
Input Arguments	warped_img:[numpy array] resultant warped maze image after applying Perspective Transform

Welcome to NB theme!	
Rulebook	2
Task 0	7
Task 1	_
CoppeliaSim Tutorials	
1A - Explore OpenCV	

1B - Detect and Encode Maze

1C - Design Ball Balance Platform

Task 2		
Task 3		
Task 4		
Task 5		
Practice Task		

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

Function Name	detectMaze()		
Output Arguments	maze_array:[<i>nested list of lists</i>] encoded maze in the form of a 2D array		
Example Call	maze_array = detectMaze(warped_img)		

• Figure 10 shows the *maze_array* output from the function detectMaze for the *warped_img* input as in Figure 9.

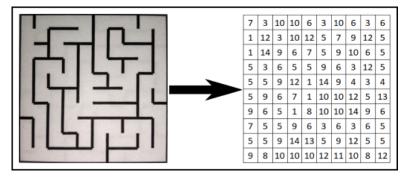


Figure 10: Output of detectMaze function

Running your solution

- To test and run your solution, do the following:
 - Open Anaconda Prompt or Terminal and navigate to the directory / folder task_1b_detect_and_encode_maze on your system.
 - 2. Activate the Conda environment created in Task 0.
 - 3. Run the command: python task_1b.py to execute your solution. Your should get an output similar to Figure 11 below.

Changelog

Welcome to NB theme!		
Rulebook		
Task 0	>	
Task 1	~	
CoppeliaSim Tutorials		
1A - Explore OpenCV		
1B - Detect and Encode Maze		
1C - Design Ball Balance Platform		
Task 2	>	
Task 3	>	
Task 4	>	
Task 5	>	
Practice Task		
Instructions for Task 6		
Task 6 Scene Details		
Coding Standard		
Git and GitHub		
Live Session 1 - 24th October 2020		
Live Session 2 - 21st November 2020		
Live Session 3 - 12th December 2020		
Live Session 4 - 10th January 2021		

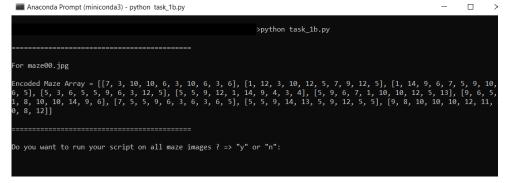


Figure 11: Executing task_1b.py

4. You can choose to run your script for rest **9 maze images** from *test_cases* folder by providing "y" as an input.

3. Testing the Solution

- The final step before submitting **Task 1B** is to test and evaluate your solution.
- For testing your solution, run the script **test_task_1b.py** in Anaconda Prompt / Terminal with the command: **python test_task_1b.py**.
- If your task_1b.py script executed without any errors, you will see the output resemble Figur 12.

Anaconda Prompt (miniconda3)

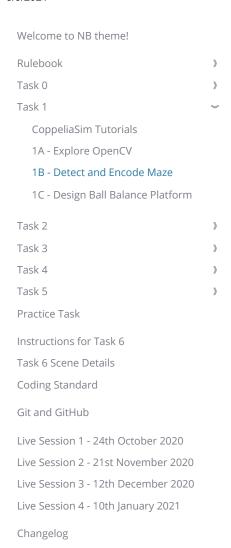


Figure 12: Executing test_task_1b.py

Note: Make sure that your code <code>task_1b.py</code> passes all the test case maze images given in the *test cases* folder.

If your code executed without errors, you will also see a new file task_1b_output.csv generate
in the task_1b_detect_and_encode_maze folder. This output file is encrypted and should
resemble Figure 13.

1B - Detect and Encode Maze - eYRC 2020-21: Nirikshak Bot (NB)

task_1b_output.csv 2 d· ta· 🖘 Pate* © Copy formate B // U III S A A S E E E E E Mega and Wasp Conferr Of Conferr Of Formate @ fx elE9QVVQT1RIYW0gSURdQTYnR2kt C D E F G H I J K L M N O P Q R S T 1 elE9QVVQT dWsoVkZAC VDJoeVw6S MCU7RF8vU1dpbmRvd3M/aEFicFc9 2 TSpEUG13SI IjQpdGI8MjEwMEE2JzFdSiQ= c3JwZmlMY aiN4fCMldMEJARWhrc1fl9Yjj9iVDEvaVkyeTA1V; Klg2NmtUZ Y2!LdikiNDE ITZIdChPNz; UIZ2ZUdIPz MySpbFw3RTZgXFVXfGdD U3YsKiMod Xyx2XXg6Uj bmRXdkp8U RUBheVlcc1JT9HJFkiXTE IXwnYFAwij JGR7akSqPj·cnp5N0d7JTYUkiKk4+cD LkUJiMjMDV+QXwnLCM5 emJeNSljaT cCtLRFxpXjETU5tOUtW/LEY0THJoQI KXBKNixeei Kl9dO1c7TT R0NU5VVPc OVhxJUtoei JztHOCRWT LyFzNTh1bDU2P3wtMj8h ZDITd0JQVC eDd3KjslTzN V2ppQVg9V NilWeCNLP YTUuPTEwN O08IYHFILTI N15UbydoC KjdFZSgrcTN T3BQISFQLj LEZrdHIWTzU8Lk5hbk0m KKMUY2AIPI WZIYTOROO KHY/W0B1c ImtiJWdaZT S3M7MV5Q ffxeaUE0UI ZzQuckZrSzi VFIPLkIwXz(TS9GSV5wC YmF7flZBWzQ7Wi4uMCpe RyVgJmtJZT dGslOCxWU azijNztQjjZv SDU3MTkiLj MDVhXFR+cJHNRajkyJD PSg8cS8iejE Sy4meXR3C QyxUZjVKRz Y2EsNTZqKzEzWzAsVXMyQ ZTBİB3BaRC citVelYiLDY; bCo9OzdhU Ti5ecnN1TT TFFhZi4zNTI X0ssfGemdT andWOvQvi VFp7dH44ci QW1ATZISC bktuNC1APiZSSUcnNSRe THIS-SERJUL GGBLGBOW P1px TXgtf1n CDJR/PMkU ZihmUmlml, RFwJWS42, KHxdckMT1; eOxgry9mal UQZJXXxdzij ZGBNLF9PbDV0ViolQkVZ QmM1JGJr1, UXRHZOBJKI X2IZXGNBQ bTt4VzUuSC PXNbejhNTI QDBzYJ9KM bZYvPjg8VT Z3dlZWovT SXM7YnhQx eGBIJmBCMjVBJEBdWiNL P11LRG9ES; VDp8R0FYQ bFFYW3AoL bGhdSmYwl lipZJ1M4bzt cUdXaHYml Xis8NmglTjt UFB4W2A/(Lm8wWmh QkRlP15aLDEyMGtiT31mVQ= Njo9XEkkW I3w8W14pRDk4MCxNbStdYQ== SEINKXSBcT, WmFiVilBez, Sil3cnsrQTJ\cnVockNrM d0FbQXoPR, aVYwRXdgP, dDB0KlxLM, MlwhW3Bx, QISFckI/Pzd, cns8ImJfMzcsNmY8QVBS MD5AOUpK UVAxYXE+d fk4hhnAoVT VIINWBByLTT SGxkNIIYAT I YOhDQmx8R R2xxY38UL "TXRIOCIzijh XCVBUjisXT U3NiYvpTZDQpTWBVNCxK
Y1FtWndFKI SmSSfUlCQ LmRcN29bJ OGBIS3p4YI dVUvMx2e+d bEpg5G99fC kzliXzpSkzA RTk/VjxEVJE Mjtvb0RVW QlRVWHMYvzEzdDk4VeZfQw 7 cZZ-digiTZf NVJ9T1V4Ri e2YwOkEqL cyhadFl3Njf bn8HXkdaIz STcIP3IzSDI: NWA+Njpke MGBFIXRcd KCE5RUUtY eipGSDVTdTdiZH11XEM7 18 LYYpU0oud: QWx8Xkc4c WjpXXIZVKC a0Ytdl84QD T05eeC0kST WXpnaUloIj Z3QwJFpGR J3ktt.J1JPDh c0VBRThgS1 fCskSyFKJTR5c3RPc2g9 OiN7MX58f buxILTleSTf Lmwofk0vT OFxRTS19N OF9cPnO1d eDZ+fntIKTf RVVQUiOhy Vne9PVUmt M3k6TzloU dvxfalsnaTEvPi1XcmMs ZnNYVnS3JTTU5bNU1hl bDNRSF9vV dnZyV3hsM: eUgyLDZ3M QDkqKjBESz Ri9RezJHQC Yj43LCEpaD c2UtezNHXj aFQzRFFHSDd1UXhrWSxd 1 IUISAXIIIDIV I ESERVI9MI MXANNOSOI YWYVIORWI MWRXVXh7 eGN1c1Vm) ZSIXOm5GN YIEOXED4W IUHsre28sO; XTRGW34vi zEvXE1uOUHERO--UHRFPUNyi M1Z9Sygyaj Y28xM0tiUj RWRdd1RH a3UsQydUN OG1CXllvSD Klshl1AwVT QCVgS3N4S KmZrYVJ0M VyE1N1QiNDE0QkNqSi WFE6VkR3c RWIJdForVC Skx8JCEjNTl Sz0uLUNGX_L1E3QydjRC NidDc1BRb; J1EtUHJDdz e2lOdSRISTl Z242bTVgTC VT9OIXt8STE0Q3orXkINXQ== bzZgQmhw(Pzkwdki8PDEwMEg1V0E00WA=
R14iMyd5e; \$y1UbilPaD: Ny5UYzI4LC J1NKsmFoR THJ7WD80t J5pdXWBsZi WXUuKSJEV ejFRaic3UD: UWxALkdec eGUuWXtRXzE0S3l9dHRcZg= 6 XDEGWYZO V IRCLINNXI XZZN 336FD fCwkRTIKZ_I NFhSIWpjVJ fmlqP2d1bjl dGwsOVVQł MWtmdWw, OHF-+ejdwYi bUZoKZNHVTEDIX:maHSGfg= 7 N3VWRUFC YVQINVBW; Tyhae1rzM RThOSUJ6dz PDwYYmdzP | ZN1T1wyZ¹ aUdjXHRFTJ dTYnSZV6Jz QTZCdkhjTj_L TWI4X310bTEOCWBn1TlbOg== 8 Zyc2J31Gbz TjB3XXs+OL Si06J1c0JTE LXRgPlxvQz, YFpcckSUaj RidPS3xiMT eml4QD9LZ Ml03QTJWz J2xbVURiJzt a3lfbF9gfjZGJEpnJmJE SOtrWlojXT/ LkEqaXhXOza0RuXkg2U SVI6TzFzMD UnY8cmpzN MWMtNHV ZEBucXA6Kj RXArfkhbOT NUxES09Qe dmooSF5TbDEzUSQ8bHAnZg== 30 S3h+c2NGU eFpGSG4iW diU7IWIOSE N1tiQyFwYz PW1RdVF5L SFYvWGApC IORNLIEULzf d0JpVUcoZz ZkhxOkRmX a3BVNyFJPTE0MDYwa1hZKg=tack th output +

Figure 13: Generation of task_1b_output.csv file

Submission Instructions

For **Task_1B submission** you have to upload a **.zip** file. To create the appropriate file please follow instructions given below:

- 1. Create a new folder named NB_<Team-ID>_Task_1B.
 - For example: if your team ID is 9999 then you need to create a folder named
 NB_9999_Task_1B.
- 2. Now copy and paste following files into this folder:
 - task_1b.py (with modified applyPerspectiveTransform() and detectMaze() functions
 - task_1b_output.csv (generated after running test_task_1b.py in task_1b_detect_and_encode_maze folder)
- 3. Compress this folder into a NB_9999_Task_1B.zip file.
- 4. Now go to the eYRC Portal and follow the instructions to upload this .zip file for Task_1B as shown in Figure 14.

Welcome to NB theme!

Rulebook

Task 0

Task 1

CoppeliaSim Tutorials

1A - Explore OpenCV

1B - Detect and Encode Maze

1C - Design Ball Balance Platform

Task 2
Task 3
Task 4
Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

Welcome to NB theme! Rulebook Task 0 Task 1 CoppeliaSim Tutorials 1A - Explore OpenCV 1B - Detect and Encode Maze 1C - Design Ball Balance Platform Task 2 Task 3 Task 4 Task 5 Practice Task Instructions for Task 6 Task 6 Scene Details **Coding Standard** Git and GitHub Live Session 1 - 24th October 2020 Live Session 2 - 21st November 2020 Live Session 3 - 12th December 2020 Live Session 4 - 10th January 2021

Changelog

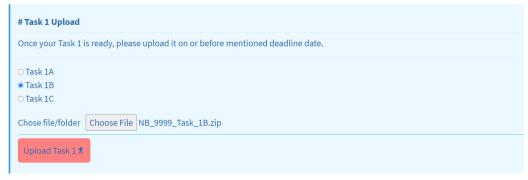


Figure 14: Submission of **NB_9999_Task_1B.zip** file on eYRC portal

5. Congrats, you have successfully completed Task 1B!

ALL THE BEST!!