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

eYRC 2020-21: Nirikshak Bot (NB)

Task 1A

Explore OpenCV

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

A. Task 1A Folder Layout

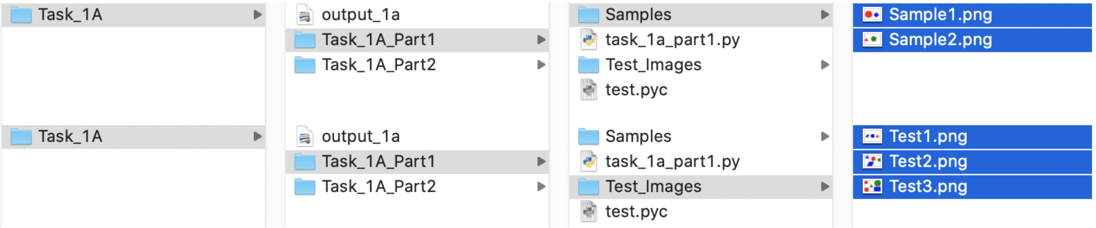
Download the following zip file containing the files for Task 1A. Right-click on the hyperlink and select **Save Link As...** option to download.

- **task_1a_explore_opencv.zip**

Following is the **file/folder layout** for **Task 1A**:

- Inside ***Task_1A*** folder you will find **two subfolders** and **one pdf file** as shown below:
  - **Task_1A_Part1**
  - **Task_1A_Part2**
  - **output_1a.pdf**
- Inside ***Task_1A*** folder you will find **files & folders** as shown below:

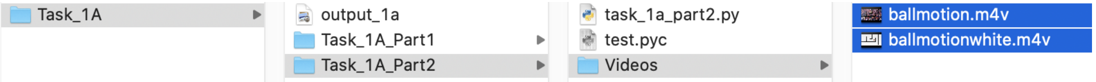
- **Task_1A_Part1**
  - *task_1a_part1.py*
  - *test.pyc*
  - *Samples*
    - *Sample1.png*
    - *Sample2.png*
  - *Test_Images*
    - *Test1.png*
    - *Test2.png*
    - *Test3.png*
- **Task_1A_Part2**
  - *task_1a_part2.py*
  - *test.pyc*
  - *Videos*
    - *ballmotion.m4v*
    - *ballmotionwhite.m4v*



```
Task_1A      output_1a       Samples            Sample1.png
             Task_1A_Part1   task_1a_part1.py   Sample2.png
             Task_1A_Part2   Test_Images
                             test.pyc

Task_1A      output_1a       Samples            Test1.png
             Task_1A_Part1   task_1a_part1.py   Test2.png
             Task_1A_Part2   Test_Images        Test3.png
                             test.pyc
```

Figure 1: Task_1A_Part1 folder layout



```
Task_1A      output_1a       task_1a_part2.py   ballmotion.m4v
             Task_1A_Part1   test.pyc           ballmotionwhite.m4v
             Task_1A_Part2   Videos
```

Figure 2: Task_1A_Part2 folder layout

B. Problem Statement - Part1

- All the images in the ***Task_1A_Part1*** folder have **different shapes** and have **only one of the three colors *(Red, Green, Blue)***.
- Task is to **find out the following details** for each image in the manner as shown below:
  1. **Detect all the non-white** shapes in the images.

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

2. **Store** the details of the these detected shapes **in a dictionary** in the same order as mentioned below in form of **key:value pair**, where ***key*** is a **string** in single quotation marks and ***value*** is an **array** of details.
   - **{ 'Shape': ['color', Area, cX, cY] }**
   - *Example* => **{ 'Circle': ['red', 1011.0, 350, 420] }**
3. It is **mandatory** to make sure that each of these **details are of definite data type** as listed below:

   - **Key:**

     - **'Shape'** => *String* in single quotation marks, with ***only first letter capital***, can take any one of these values: ***Circle/ Triangle/ Trapezium/ Rhombus/ Square/ Quadrilateral/ Parallelogram/ Pentagon/ Hexagon***

   - **Value:**

     - **'color'** => *String* in single quotation marks, with ***all letters in small caps*** can take any one of these values: ***red/ blue/ green***
     - **Area** => *Float* value up to ***one decimal point*** (area of the detected shape)
     - **cX** => *Int* value (***centroid coordinate*** of shape on ***horizontal X-axis*** direction)
     - **cY** => *Int* value (***centroid coordinate*** of shape on ***vertical Y-axis*** direction)

Note:

- Failing to follow this data type convention will lead to **deduction of marks.**

- Shapes could be any of these: *Circle/ Triangle/ Trapezium/ Rhombus/ Square/ Quadrilateral/ Parallelogram/ Pentagon/ Hexagon*. **No other shape except these**.

- For *cX* and *cY* consider the **top-left point of the image as origin (0, 0) reference.**

- If there are multiple shapes in the image then they should be stored in **decreasing order of their respective area** as shown below:

  - Example =>

    { 'Circle' : ['red', 1011.0, 350, 420], 'Square' : ['green', 706.0, 469, 786], 'Triangle' : ['blue', 555.0, 350, 50] }

Task Instructions for Part1

For this part of the task we have provided a **"snippet"** of outline code in `task_1a_part1.py` file:

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

- Teams are **not allowed** to import any other library/module, other than the ones already imported in the `task_1a_part1.py` .
- Teams are **not allowed** to edit the **main()** function.
- Teams have to modify the `scan_image()` function to take **image file path** as input and returr a **dictionary** with ***details of non-white shapes*** detected in the image as explained above.
- Make sure you run `task_1a_part1.py` using the *Conda environment created in Task 0*.
- When you run the `task_1a_part1.py` , as a default, the **main()** function will feed the file path (**Sample1.png** file which is present in the **Samples** folder as shown above.
- It will print the output of the `scan_image()` function i.e. the **dictionary** returned. You can verify the accuracy of your code by referring to the **output_1a.pdf** which lists the expected output for **Sample1.png**.
- It will then ask you if you want to run the same code for the rest of the images in the **Samples** folder. If you are satisfied with your output for **Sample1.png**, then you can press ***y*** and press ***Enter*** to proceed.
- Again verify the output accuracy by referring the **output_1a.pdf**.
- **Permissible error** limit for:
    - **Area** of detected Shapes should be within **± 1%** range.
    - **cX** and **cY** i.e. Shape's centroid coordinates should be within **± 5** pixels.

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

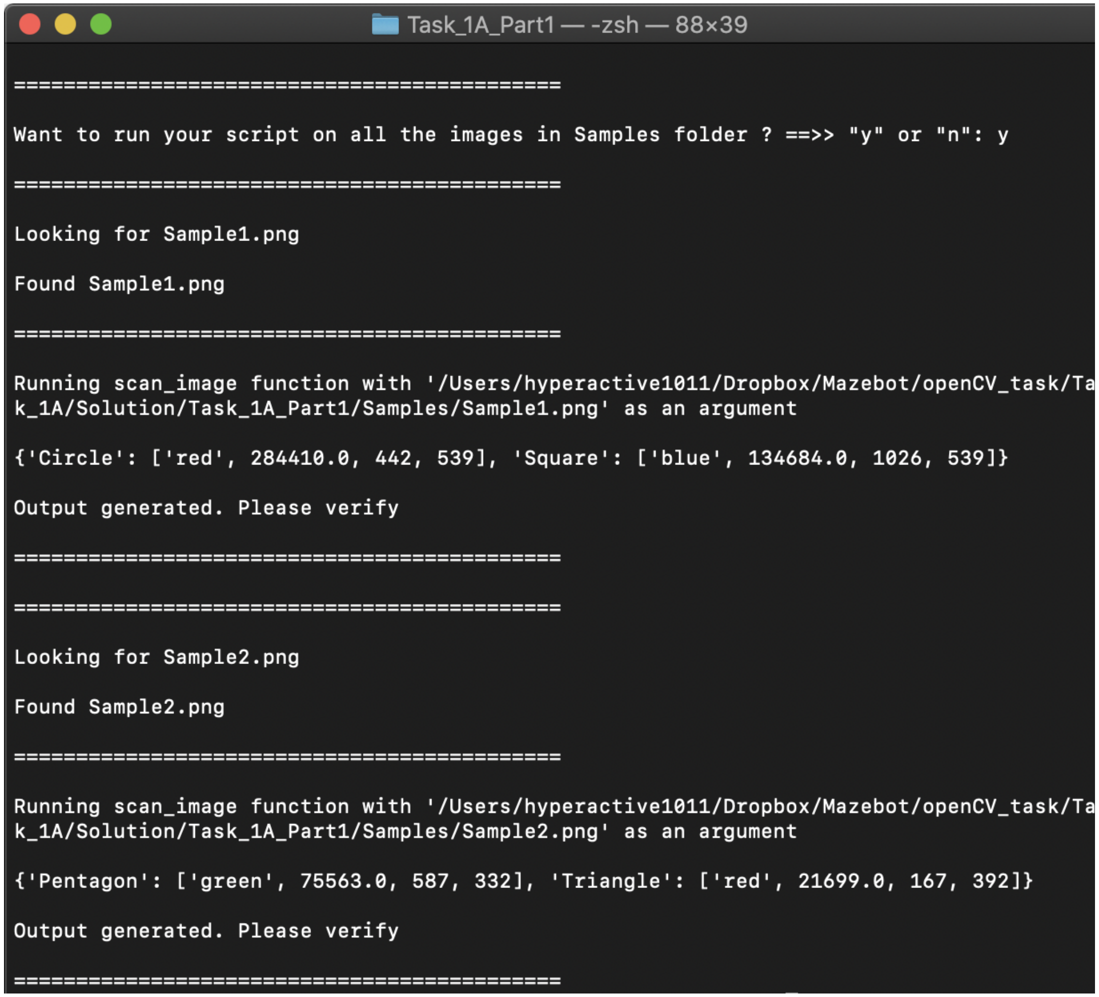
Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog



```
============================================

Want to run your script on all the images in Samples folder ? ==>> "y" or "n": y

============================================

Looking for Sample1.png

Found Sample1.png

============================================

Running scan_image function with '/Users/hyperactive1011/Dropbox/Mazebot/openCV_task/Ta
k_1A/Solution/Task_1A_Part1/Samples/Sample1.png' as an argument

{'Circle': ['red', 284410.0, 442, 539], 'Square': ['blue', 134684.0, 1026, 539]}

Output generated. Please verify

============================================

============================================

Looking for Sample2.png

Found Sample2.png

============================================

Running scan_image function with '/Users/hyperactive1011/Dropbox/Mazebot/openCV_task/Ta
k_1A/Solution/Task_1A_Part1/Samples/Sample2.png' as an argument

{'Pentagon': ['green', 75563.0, 587, 332], 'Triangle': ['red', 21699.0, 167, 392]}

Output generated. Please verify

============================================
```

Figure 3: Overall Output of Task 1A - Part 1

- Once done with the **Task 1A - Part 1**, run **test.pyc** provided in the same folder / directory wit command: `python test.pyc` .
- It will run your modified code `task_1a_part1.py` on **Test1.png, Test2.png and Test3.png** which are located in the **Test_Images** folder as shown above.
- It will show the output of your program on Terminal / Anaconda Prompt and also generate `task_1a_part1_output.txt` in the same folder / directory.

WARNINGS !!

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

- **Do not edit the name** of any files / folders.
- **Do not move/delete** any file / folder.
- **Do not edit** contents of `task_1a_part1_output.txt` file.
- **Do not edit** the **main()** function.
- **Do not edit** the name of the **scan_image()** function.
- **Input and output arguments** of the **scan_image** function are as specified. **Do not change them**.
- Before submission make sure to **remove all the statements responsible for displaying any image.**
- While completing the function in `task_1a_part1.py`, you might require to print some information on Terminal / Anaconda Prompt for the debugging process, but make sure to **comment / remove all of them** before submitting the script to us.

C. Problem Statement - Part2

- For ***Task_1A_Part2***, you are provided with **two video files** namely **ballmotion.m4v** and **ballmotionwhite.m4v**. In both these videos, there is a **red circle moving** in frame inside a maze.
- Task is to **find the coordinates** of the ***red circle*** for particular frame(s):
  1. Store the details of coordinates of the *red circle* in a dictionary in the same order as mentioned below i.e. in the form of **key:value pair**.
     - **Frame Number: [cX, cY]**
     - *Example* => **{ 44: [350, 420] }**
  2. It is mandatory to make sure that each of these details are of definite data type as listed below:
     - **Key:**
       - **Frame Number** => *Int* value
     - **Value:**
       - **cX** => *Int* value (***centroid coordinate*** of red circle on ***horizontal X-axis*** directio
       - **cY** => *Int* value (***centroid coordinate*** of red circle on ***vertical Y-axis*** direction)

Note:

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

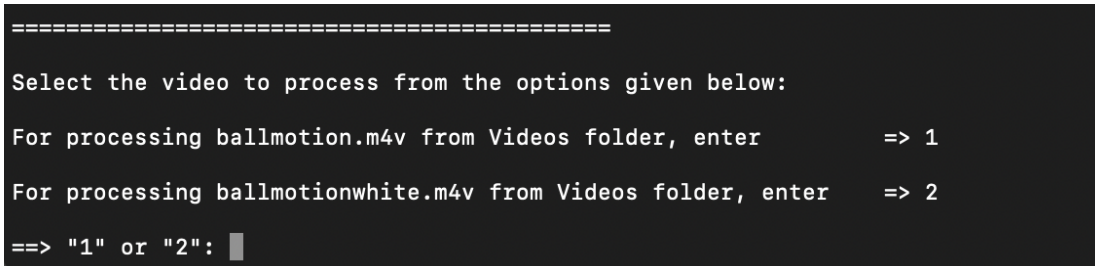
- Failing to follow this data type convention will lead to **deduction of marks**.
- For *cX* and *cY* consider the **top left point of the frame as origin (0, 0) reference.**
- If there are multiple frame numbers for processing, then they should be stored in **increasing order of their frame number** as shown below:
  - Example =>

    { 44:[350, 420], 69:[469, 786], 99:[350, 50] }

Task Instructions for Part 2

For this part of the task we have provided a **“snippet”** of outline code in `task_1a_part2.py` file:

- Teams are **not allowed** to import any other library/module, other than the ones already imported in the `task_1a_part2.py`.
- Teams are **not allowed** to edit the **main()** function.
- Teams modify the `process_video()` function to take a **video file path** and a **list of frame numbers** as input and return a **dictionary** with ***details of coordinates of the red circle in those frames*** as explained above.
- Make sure you run `task_1a_part2.py` using the *Conda environment created in Task 0*.
- When you run the `task_1a_part2.py`, as a default, the **main()** function will ask you to select ***one of the two available videos*** as shown in Figure 4 below. Choose an appropriate option.



```
============================================

Select the video to process from the options given below:

For processing ballmotion.m4v from Videos folder, enter         => 1

For processing ballmotionwhite.m4v from Videos folder, enter    => 2

==> "1" or "2":
```

Figure 4: Option to select a video in task_1a_part2.py

- Then it will ask for a ***list of frame number(s)*** you want to process for finding the coordinates c the red circle as shown below:

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

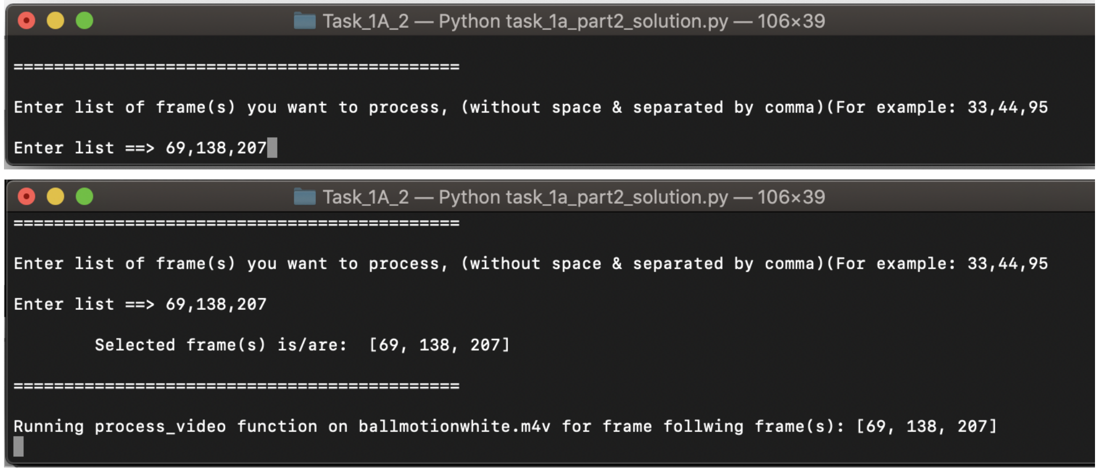
Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog


Task_1A_2 — Python task_1a_part2_solution.py — 106×39
Task_1A_2 — Python task_1a_part2_solution.py — 106×39

```
============================================

Enter list of frame(s) you want to process, (without space & separated by comma)(For example: 33,44,95

Enter list ==> 69,138,207
```

```
============================================

Enter list of frame(s) you want to process, (without space & separated by comma)(For example: 33,44,95

Enter list ==> 69,138,207

        Selected frame(s) is/are:  [69, 138, 207]

============================================

Running process_video function on ballmotionwhite.m4v for frame follwing frame(s): [69, 138, 207]
```

Figure 5: Entering the frame list in task_1a_part2.py

Note: To provide the frame list **only use commas** to separate the values as shown in Figure 5.

- Now the **main()** function will feed the appropriate file path of the selected video and the fram list to the `process_video` function.
- It will print the output of the function i.e. the **dictionary** returned. You can verify the accuracy of your code by referring to the **output_1a.pdf** which lists the expected output for *some frame* from both the videos.
- **Permissible error** limit for:
  - **cX** and **cY** i.e. red circle's centroid coordinates should be within **± 5** pixels.

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

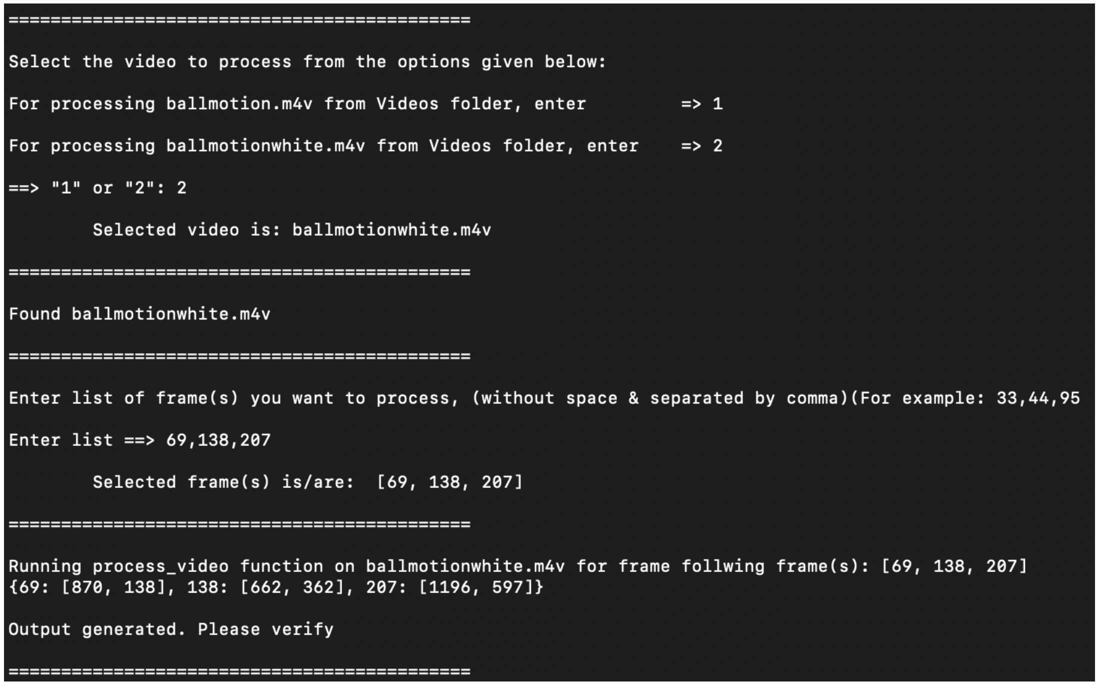
Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog



```
============================================

Select the video to process from the options given below:

For processing ballmotion.m4v from Videos folder, enter         => 1

For processing ballmotionwhite.m4v from Videos folder, enter    => 2

==> "1" or "2": 2

        Selected video is: ballmotionwhite.m4v

============================================

Found ballmotionwhite.m4v

============================================

Enter list of frame(s) you want to process, (without space & separated by comma)(For example: 33,44,95

Enter list ==> 69,138,207

        Selected frame(s) is/are:  [69, 138, 207]

============================================

Running process_video function on ballmotionwhite.m4v for frame follwing frame(s): [69, 138, 207]
{69: [870, 138], 138: [662, 362], 207: [1196, 597]}

Output generated. Please verify

============================================
```

Figure 6: Overall Output of Task 1A - Part 2

- Once done with the **Task 1A - Part 2**, run **test.pyc** provided in the same folder / directory witl command: `python test.pyc` .
- It will run your modified code `task_1a_part2.py` on the video file **ballmotion.m4v** for a ***frame_list* = [55, 110, 165, 220, 275, 330, 385]**.
- It will show the output of your program on Terminal / Anaconda Prompt and also generate `task_1a_part2_output.txt` in the same folder.

WARNINGS !!

- **Do not edit the name** of any files / folders.
- **Do not move/delete** any file / folder.
- **Do not edit** contents of `task_1a_part2_output.txt` file.
- **Do not edit** the **main()** function.
- **Do not edit** the name of the **process_video()** function.

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

- **Input and output arguments** of the process_video function are as specified. **Do not change them.**
- Before submission make sure to **remove all the statements responsible for displaying any image.**
- While completing the function in `task_1a_part2.py`, you might require to print some information on Terminal / Anaconda Prompt for the debugging process, but make sure to **comment / remove all of them** before submitting the script to us.

D. Submission Instructions

For **Task_1A 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_1A`.
   - For example: if your team ID is **9999** then you need to create a folder named `NB_9999_Task_1A`.
2. Now copy and paste following files into this folder:
   - `task_1a_part1.py` (with modified **scan_image()** function)
   - `task_1a_part1_output.txt` (generated after running `test.pyc` in ***Task_1A_Part1*** folder
   - `task_1a_part2.py` (with modified **process_video()** function)
   - `task_1a_part2_output.txt` (generated after running `test.pyc` in ***Task_1A_Part2*** folder
   - Now your folder must look as depicted in Figure 7.

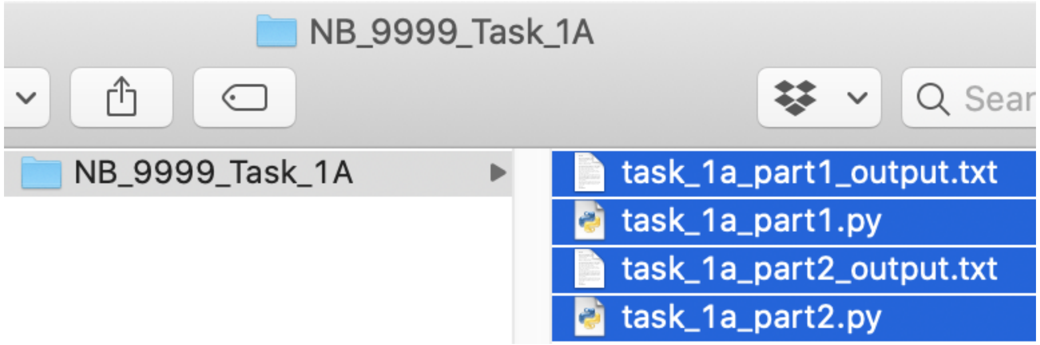
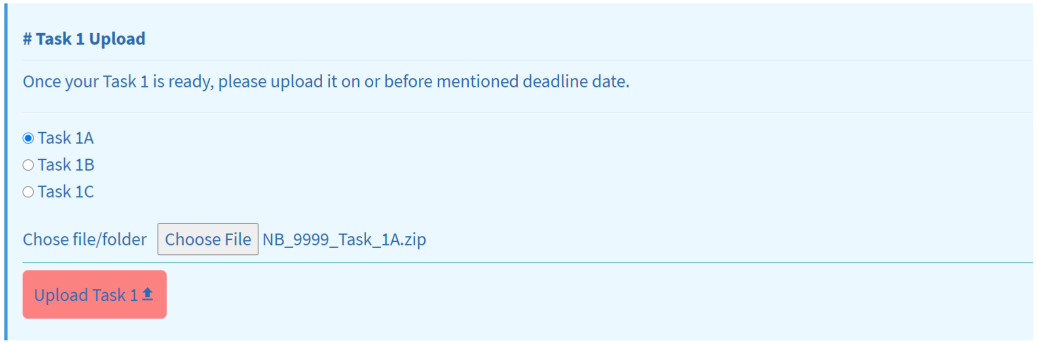


Figure 7: Submission folder structure of NB_9999_Task_1A

3. Compress this folder into a `NB_9999_Task_1A.zip` file.

4. Now go to the eYRC Portal and follow the instructions to upload this **.zip** file for **Task_1A** as shown in Figure 8.



# Task 1 Upload

Once your Task 1 is ready, please upload it on or before mentioned deadline date.

● Task 1A
○ Task 1B
○ Task 1C

Chose file/folder [Choose File] NB_9999_Task_1A.zip

Figure 8: Submission of **NB_9999_Task_1A.zip** file on eYRC portal

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

ALL THE BEST !!
