|  |  |
| --- | --- |
| Task 1 | |
| task\_1a\_part1.py | s**hapes**(dictionary) **getContours(imgColor**(np.array)**)**  **mask**(np.array) **colorMask(imgHsv**(np.array)**)**  **cX,cY**(number) **calibrateCentroid(cX,cY**(number)**)**  **obj**(String) **getShape4(cnt**(list)**)**  **bool isEqual(v1**(list)**,v2** (list)**)**  **bool isPerpendicular(v1** (list)**,v2** (list)**)**  **bool isParallel(v1** (list)**,v2** (list)**)**  **Shapes({Circle:[color,cX,cY]}) = scan\_image(img\_file\_path**(np.array)**)** |
| Task\_1a\_part2.py | **Global:frame\_details={}**  **Frame\_details process\_video(vid\_file\_path, frame\_list)** |
| Task\_1b.py | **rect**(2-D list) **orderedPolyDp(corners**(2-D list)**)**  **rect**(np.array) **getBorderCoordinates(imgMorph**(np.array)**)**  **drawContours(imgCpy**(np.array)**, contours\_poly** (3D list)**, boundRect**(2D list)**, top\_right**(1D list)**, top\_left** (1D list)**,bottom\_right** (1D list)**,bottom\_left**(1D list)**)**  **imgMorph**(np.array) **threshInputImage(img**(np.array)**)**  **warped\_img**(np.array) **= applyPerspectiveTransform(input\_img**(np.array)**)**  **maze\_array**(2D list) **= detectMaze(warped\_img**(np.array)**)**  **warped\_img = writeToCsv('test\_cases/maze00.csv', maze\_array)** |
| Task 2 | |
| Task\_2a.py | **shapes getShape(client\_id,vision\_sensor\_handle)**  **client\_id init\_remote\_api\_server()**  **return\_code start\_simulation(client\_id)**  **vision\_sensor\_image, image\_resolution, return\_code get\_vision\_sensor\_image(client\_id,sensor\_handle)**  **transformed\_image transform\_vision\_sensor\_image(vision\_sensor\_image, image\_resolution)**  **return\_code stop\_simulation(client\_id)**  **exit\_remote\_api\_server(client\_id)** |
| Task\_2b.py | **return\_code send\_data(client\_id,maze\_array,table\_number)** |
| Task\_3 | |
| Task\_3.py | **Global:**  **outMax=60**  **outMin=-60**  **kp=np.array([0.02,0.02],dtype='float64')**  **ki=np.array([0.001,0.001],dtype='float64')#ki=ki\*SampleTime**  **kd=np.array([0.135,0.135],dtype='float64')#kd=kd/SampleTime**  **SampleTime=0.1**  **setAngles(client\_id,revolute\_handle,Output)**  **revolute\_handle,vision\_sensor\_handle init\_setup(rec\_client\_id,table\_number)**  **np.array(transformed,dtype='float64') coordinateTransform(xy)**  **ITerm,lastInput,lastTime,Input,lastOutput,summation,Output control\_logic(setpoint,client\_id,center\_x,center\_y,ITerm,lastInput,lastTime,Input,lastOutput,summation,Output)** |
| Task\_4 | |
| Task\_4a.py | **distance\_mat make\_step(k, distance\_mat, maze\_array)**  **path\_copy[min\_indx] getMinPath(path)**  **final\_path getPath(start\_coord,end\_coord,distance\_mat,maze\_array)**  **path = find\_path(maze\_array, start\_coord, end\_coord)**  **start, end = read\_start\_end\_coordinates("start\_end\_coordinates.json", "maze00")** |
| Task4b.py | **Shapes getBallData(client\_id,vision\_sensor\_handle)**  **maze\_array, path = calculate\_path\_from\_maze(maze\_array,start\_coord, end\_coord)**  **send\_data\_to\_draw\_path(rec\_client\_id,path, table\_number)**  **pixel\_path = convert\_path\_to\_pixels(path)**  **traverse\_path(client\_id,pixel\_path,vision\_sensor\_handle,revolute\_handle)** |
|  | **send\_color\_and\_collection\_box\_identified(ball\_color, collection\_box\_name)**  **calculateMazeArrays()**  **send\_mazeData()**  **vs\_conveyer()**  **stopStreaming(vs\_number)**  **revolute\_handle , vision\_sensor\_handle setup\_maze\_for\_ball( client\_id , table\_number , path )**  **stopSimulation(client\_id)**  **Ball\_info**(list) **getBallInfo(ball\_color)**  **read\_ball\_details(file\_name)**  **processMaze(client\_id,ball\_info,revolute\_handle,vision\_sensor\_handle)**  **main(rec\_client\_id)** |
|  | |
|  | |
|  | |