Final In-Course Project Report

For Subjects:

MECH642(Machine Vision)- MV

MECH623(Hydraulics and Electro-hydraulic)- MR

By

Student Name: Faizaan Mustafa(BH16500026)

SY. 2019-2020

Trimester- 3

Table of Contents

1. ***Title of the Project:*** **Machine Vision & Hydraulics Combinational Project**
2. ***Objective*:** To use image processing for sorting objecting with the use of hydraulic cylinders.
3. ***Tools/Software Required:*** MATLAB, CoppeliaSim.
4. ***Summary:***

This project involved building a functioning conveyer belt with the inclusion of hydraulic cylinders for sorting colored cubes. A camera was used to detect the cubes and to capture the images of the same cubes as they moved on the conveyor belt.

Two different software platforms were used for this project including the MATLAB software and the Coppelia Simulation software. Both platforms communicate with each other simultaneously as the whole process begins. Coppelia Sim includes the sensors, actuators, and the complete model required for the sorting application while MATLAB is used to detect the inputs from the sensors and to send commands to the actuators in the Coppelia software. The actuators in this application are the cylinders that are used to sort the cubes. This project fuses the applications of hydraulics and machine vision into one to simulate an industrial application of sorting different objects.

5. ***Abstract:***

The purpose of this project is to build a prototype that makes use of hydraulics and machine vision for the industrial application of sorting colored objects.

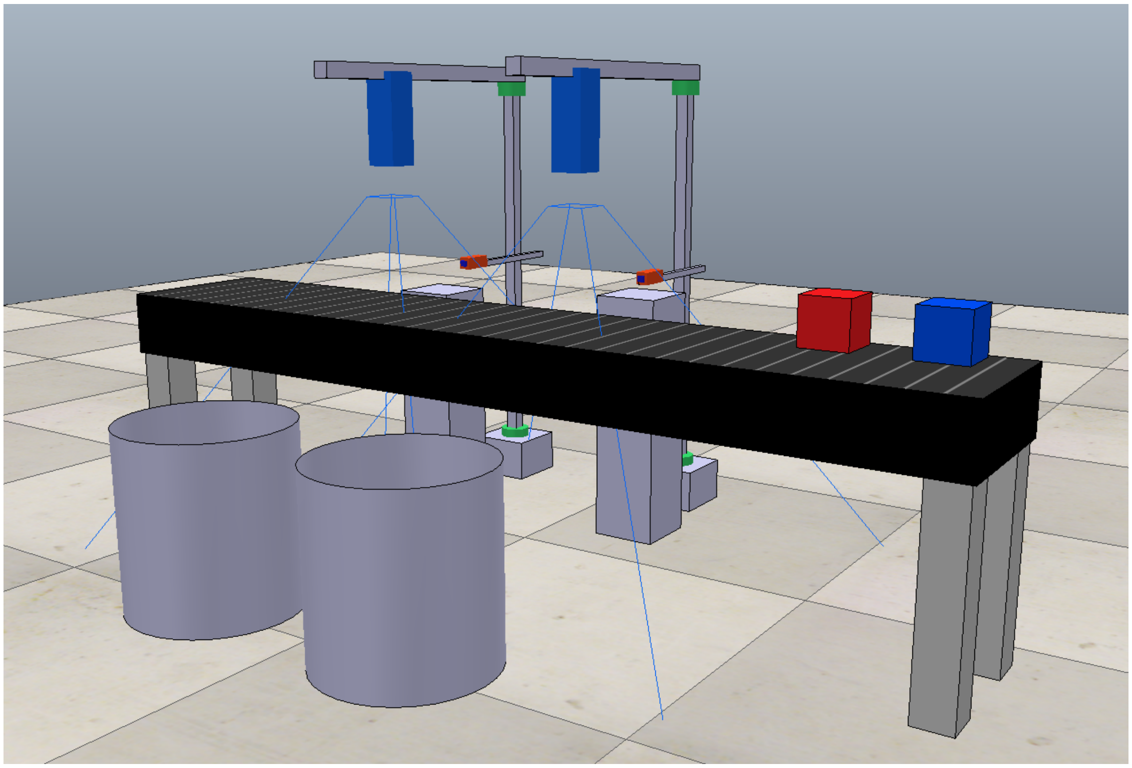
6. ***Introduction:***

Hydraulics and Electro Hydraulics are widely used in the industry for many applications where power is needed. Machine vision is also used for image processing in many fields including medicine, and self- driving transportation. This project involves building an industrial prototype that involves Machine Vision and Hydraulics with the use of the MATLAB and the Coppelia Simulation software.

7. ***Design and Analysis:***

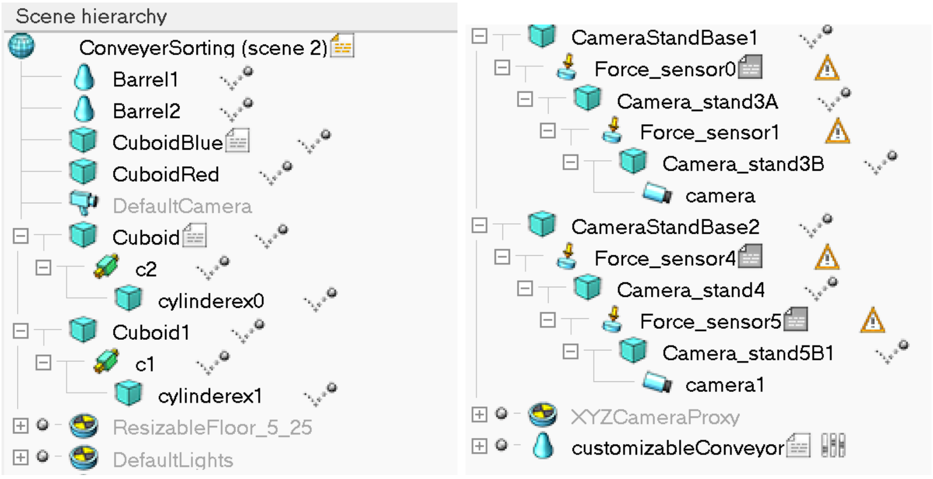
The application involves sorting two colored cubes including a red cube and a blue cube that move on a conveyer belt before being detected by two cameras.

Initially, the complete scene is constructed in the Coppelia simulation environment as shown on the next page.



I.e CoppeliaSim Scene

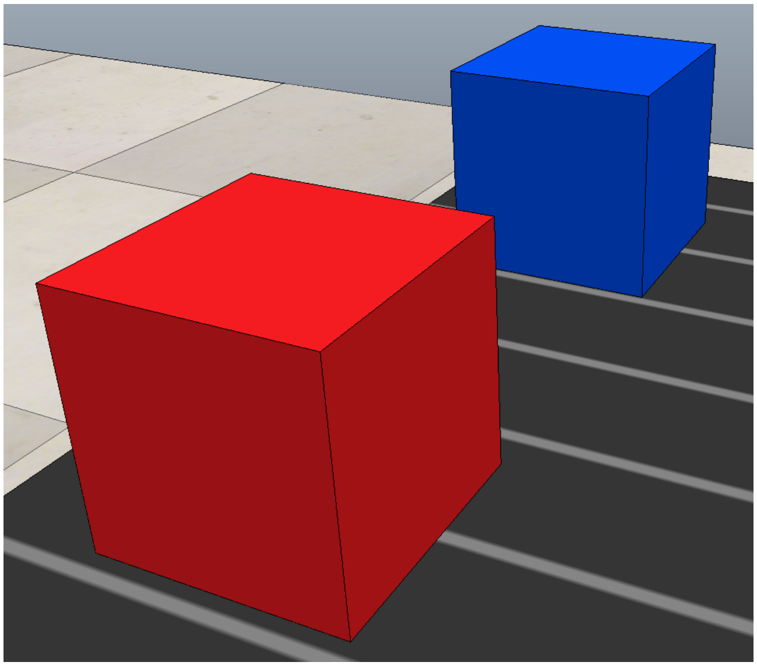
The scene hierarchy for the scene is the following:



```
Scene hierarchy
ConveyerSorting (scene 2)
    Barrel1
    Barrel2
    CuboidBlue
    CuboidRed
    DefaultCamera
⊟   Cuboid
    ⊟   c2
            cylinderex0
⊟   Cuboid1
    ⊟   c1
            cylinderex1
⊞   ResizableFloor_5_25
⊞   DefaultLights
```

```
⊟ CameraStandBase1
    ⊟ Force_sensor0
        ⊟ Camera_stand3A
            ⊟ Force_sensor1
                ⊟ Camera_stand3B
                      camera
⊟ CameraStandBase2
    ⊟ Force_sensor4
        ⊟ Camera_stand4
            ⊟ Force_sensor5
                ⊟ Camera_stand5B1
                      camera1
⊞ XYZCameraProxy
⊞ customizableConveyor
```

The components used include a conveyer belt, two colored cubes, two barrels, and two camera stands with two cameras attached.

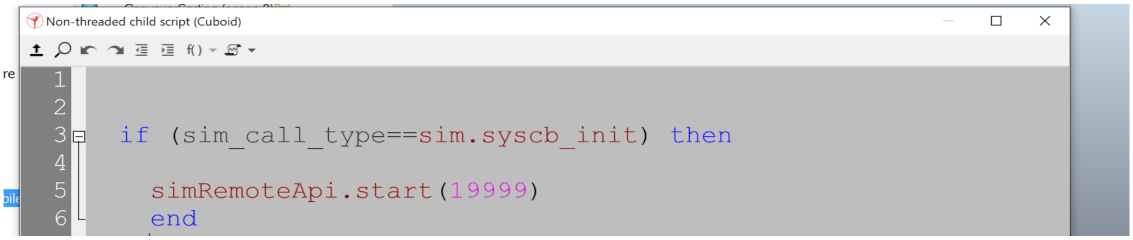


I,e The cubes measure at 0.07* 0.07* 0.07 meters.

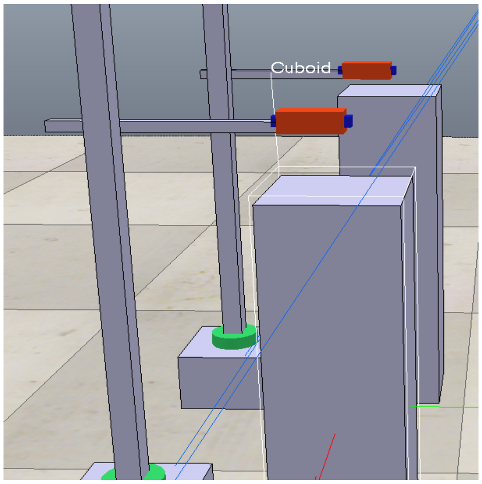
All these components are built in the environment and are presented in the scene hierarchy. Each camera is used to detect a specific color by capturing and processing the images taken to then actuate the cylinder for pushing the colored cube to the assigned barrel based on the cube's color. The red cube should be pushed into the first barrel and the blue cube should be pushed in the second barrel. The sorting is performed in the Coppelia simulation software environment scene while MATLAB is used for processing the images and for actuating the cylinders.

To establish a connection between MATLAB and Coppelia Simulation, the following code is entered in the child script of the 'cuboid' object present in the Coppelia Sim. software(1):

“



```
if (sim_call_type==sim.syscb_init) then
  simRemoteApi.start(19999)
  end
```



I.e object ‘cuboid’

From the MATLAB side, the following code must be entered in the MATLAB script(2):

“sim=remApi('remoteApi');

sim.simxFinish(-1);

clientID=sim.simxStart('127.0.0.1',19999,true,true,5000,5);”

The codes mentioned above initiate a remote API connection between MATLAB and CoppeliaSim.

The files for the MATLAB and CoppeliaSim software are named as ‘ConveyerS.m’ and ‘ConveyerSorting’ respectively for both software and are then placed in the same folder known as ‘MATLAB-VREP’.

The following code was then entered in the MATLAB with the guidance of the remote API functions present in the CoppeliaSim website(3) for programming the camera and the cylinder actuators:

MATLAB Code:

```
sim=remApi('remoteApi');
sim.simxFinish(-1);
 clientID=sim.simxStart('127.0.0.1',19999,true,true,5000,5);

 if (clientID>-1).   % It is checked if the connection is established between MATLAB and CoppeliaSim.
        disp('Connected');

% Initially, handles are created for the cylinder actuators and the two cameras.

        [returnCode,cylinder1 ]= sim.simxGetObjectHandle( clientID,'c1',sim.simx_opmode_blocking) %cyl1Handle
        [returnCode,cylinder2 ]= sim.simxGetObjectHandle( clientID,'c2',sim.simx_opmode_blocking) %cyl2Handle
```

```
                [returnCode,cameraM]= sim.simxGetObjectHandle(
clientID,'camera',sim.simx_opmode_blocking)  %camera1Handle
                  [returnCode,resolution, image1]=sim.simxGetVisionSensorImage2(
clientID, cameraM,0,sim.simx_opmode_streaming);

                    [returnCode,cameraM1]= sim.simxGetObjectHandle(
clientID,'camera1',sim.simx_opmode_blocking) %camera2Handle
                  [returnCode,resolution, image2]=sim.simxGetVisionSensorImage2(
clientID, cameraM1,0,sim.simx_opmode_streaming);

                for i=0:100     % a for loop with 100 iterations is used for this process
                  [returnCode,resolution, image1]=sim.simxGetVisionSensorImage2(
clientID, cameraM,0,sim.simx_opmode_buffer);
                  [returnCode,resolution, image2]=sim.simxGetVisionSensorImage2(
clientID, cameraM1,0,sim.simx_opmode_buffer);

                  %color matrices are for separating the red from the images.
                  redImg1= image1(:,:,1);
                  bluImg2= image2(:,:,1);

                  %display camera images
```

```
subplot(1,2,1);

imshow(image1);


subplot(1,2,2);

imshow(image2);
```

% For programming the color detection segment, a for loop is used to check all the elements in image 1 and 2 as shown below. If statements are used to compare the red values in both images with the thresholds specified below to detect the cubes.

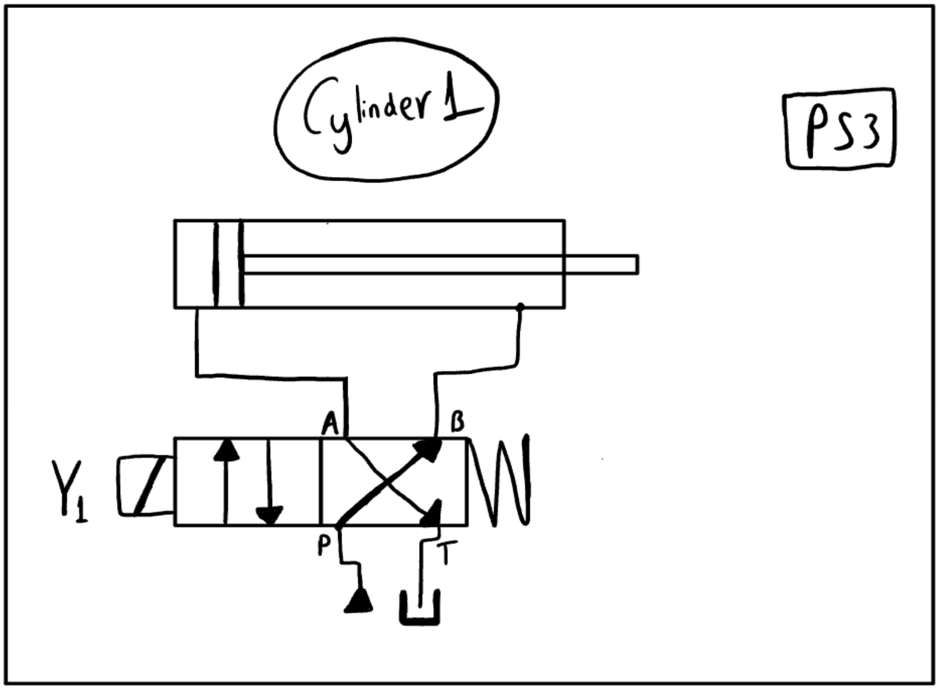
```
 %check for the red cube.

for k= 1:numel(redImg1)

   if redImg1(k)>220  % the threshold, in this case is 220.

         disp('it is red')

  pause(1.48);

[returnCode]=sim.simxSetJointTargetVelocity( clientID,cylinder1,1.25
, sim.simx_opmode_blocking)        %extend cylinder 1 for 0.2 seconds

pause(0.2)


[returnCode]=sim.simxSetJointTargetVelocity( clientID,cylinder1,-1.25
, sim.simx_opmode_blocking)

pause(0.6)    %retract cylinder 1 for 0.6 seconds
```

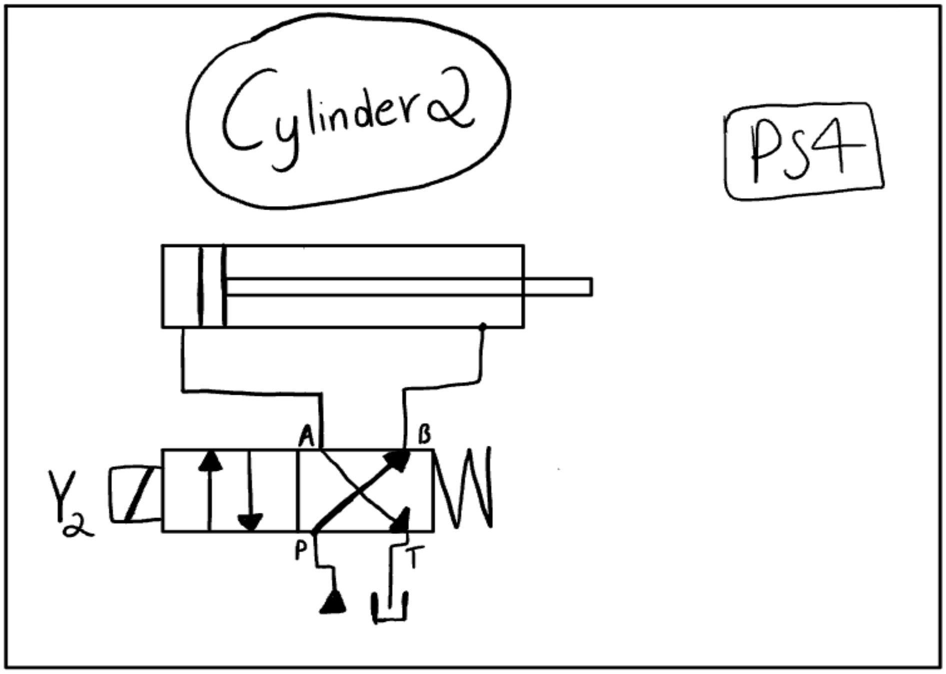
```
            break;

             end

           end


           %check for blue cube

              for j= 1:numel(bluImg2)

             if  bluImg2(j)>20   % the threshold in this case is 20.

                 disp('it is blue')

            pause(1.48);


           [returnCode]=sim.simxSetJointTargetVelocity( clientID,cylinder2,1.25

, sim.simx_opmode_blocking)     %extend cylinder 2 for 0.2 seconds

           pause(0.2)

           [returnCode]=sim.simxSetJointTargetVelocity( clientID,cylinder2,-1.25

, sim.simx_opmode_blocking)     %retract cylinder 2 for 0.6 seconds

           pause(0.6)

           break;

             end


             end

           end

            sim.simxFinish(-1);
```

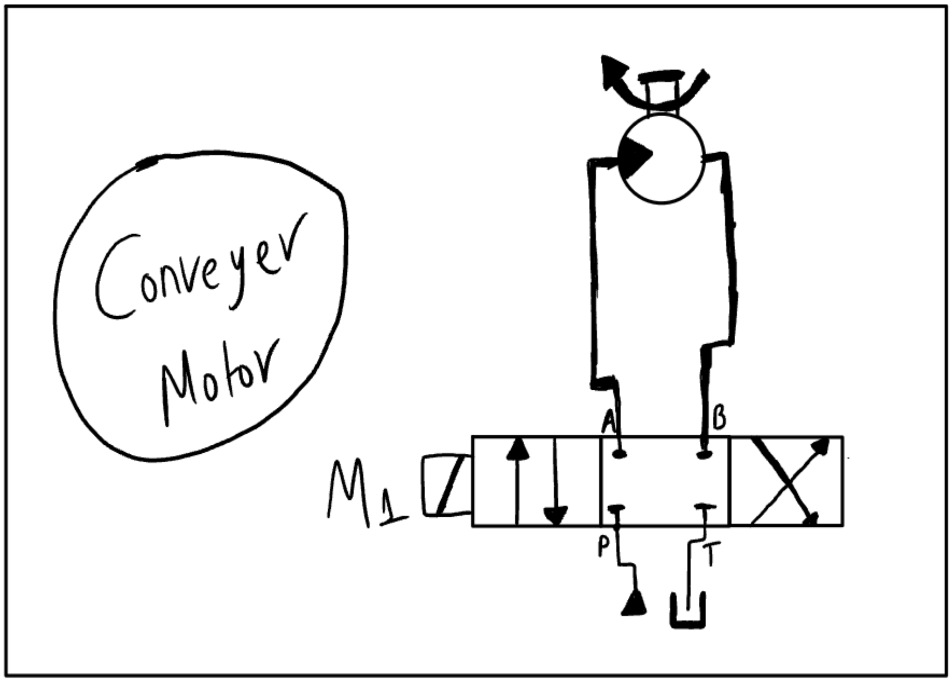
```
  end


   sim.delete();
 % Program Ends.
```

The hydraulic aspect of this project involves the use of cylinders that are actuated when the objects are detected. An equivalent hydraulic circuit for this application has been presented below but with the use of proximity sensors instead of a camera as cameras have not been used in this Hydraulics and Electro- hydraulics course for sensing objects. The cylinders were also motorized as a hydraulic source was not available on the simulation software when the environment was being constructed. The circuit has been presented on the next few pages.

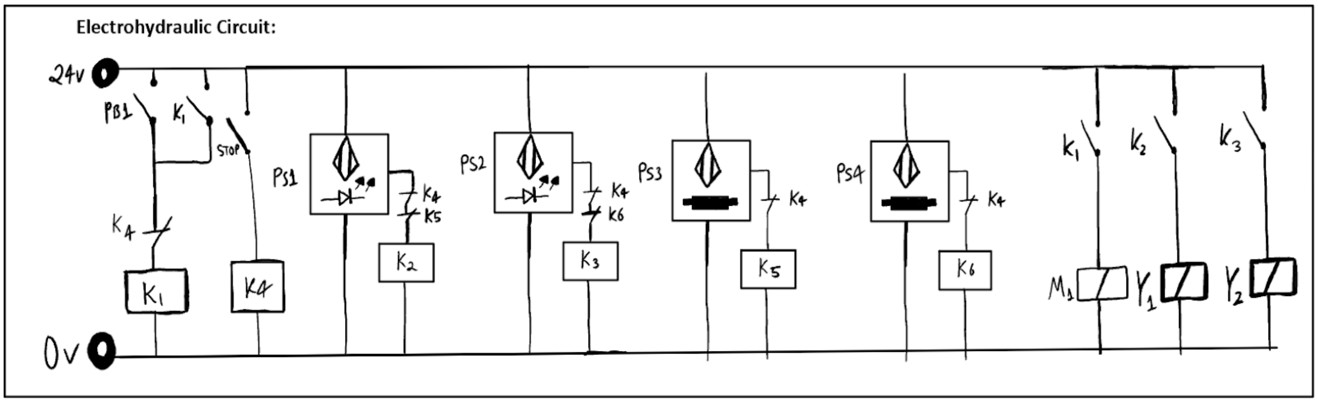
Hydraulic Circuit:







Electro-hydraulic Circuit:



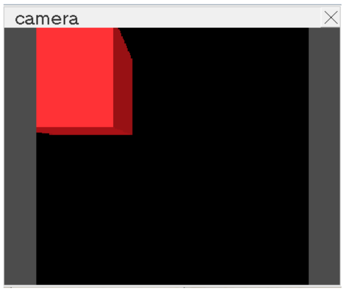
In the circuits shown above ‘PB1’ is the start button or the debug button. Once this button is pressed, ‘K1’ is energized which leads to the energization of solenoid ‘M1’. Once solenoid ‘M1’ is actuated, the conveyer motor starts. ‘PS1’ and ‘PS2’ are optical proximity sensors used for sensing the red cube and the blue cube respectively. Once the red cube is detected by sensor ‘PS1’, K2 is energized, and thus, solenoid Y1 is energized which leads to the extension of cylinder 1 in the circuit. Once cylinder C1 is fully extended, the inductive proximity sensor 'PS3' detects it and energizes K5 which causes cylinder 1 retract as the closed contact of K5 is broken.

When the blue cube is detected by sensor 'PS2', K3 is energized and thus, solenoid Y2 is energized which leads to the extension of cylinder 2 in the circuit. Once cylinder C2 is fully extended, the inductive proximity sensor 'PS4' detects it and energizes K6 which causes cylinder 2 retract as the closed contact of K2 is broken.

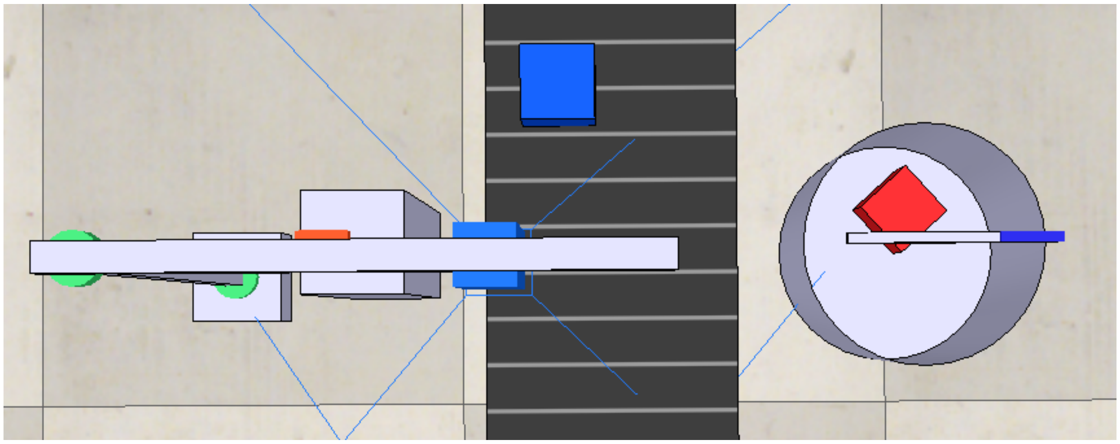
A stop button is placed to stop the process at any time. When this button is pressed, K4 is energized.

8. ***Results & Discussion:***

To start the process, the CoppeliaSim had to be debugged first and then the MATLAB program. Once both programs were running, the conveyer belt started mocking with the two colored cubes. When the first camera detected the red cube, the cylinder extended and pushed that red cube in barrel one.

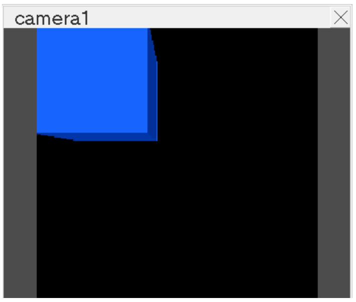


I.e Red Cube Detected

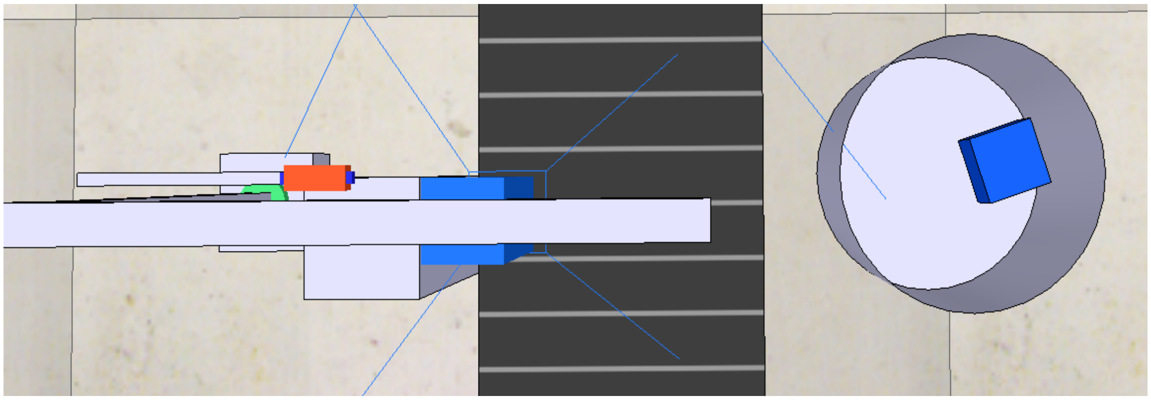


I.e Red Cube Sorted

The blue cube came next and was ignored by the camera 1. As soon as the blue cube was detected by the second camera, the second cylinder extended and pushed that blue cube into the second barrel.

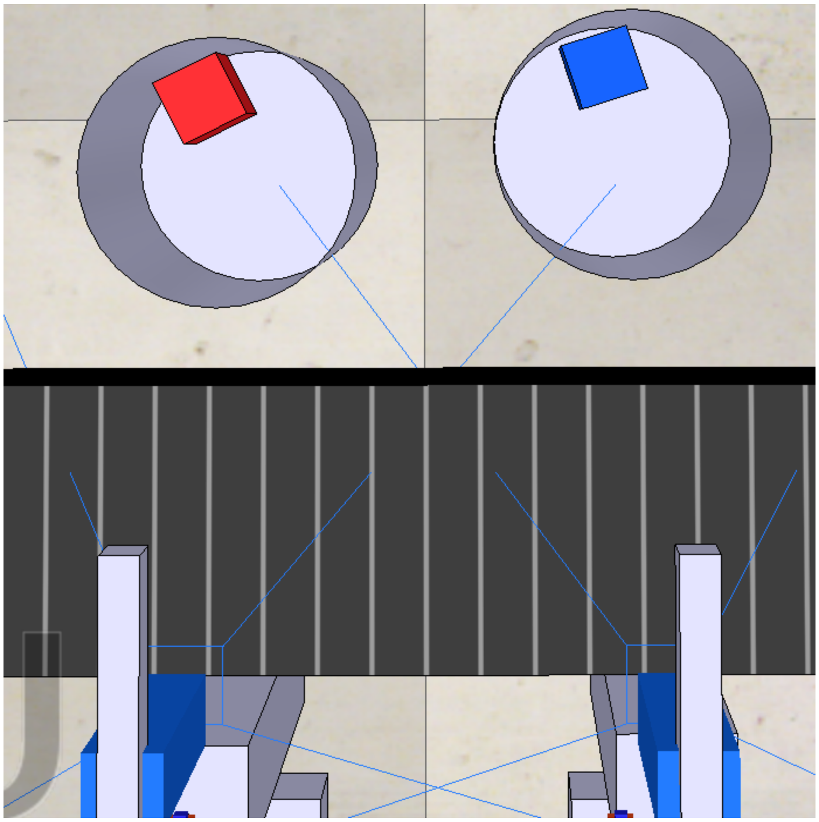


I.e Blue Cube Detected



I.e Blue Cube Sorted

Thus, the sorting process was successfully completed with both cubes sorted to their respective barrels.



I.e Both Cubes Sorted.

9. ***Conclusion:***

In this project, a functioning sorting system was constructed and simulated using MATLAB and CoppeliaSim. Images were acquired from Coppelia Sim and were transferred to MATLAB for image processing. Once the cubes were detected, their respective cylinders were actuated from MATLAB to sort those cubes in their respective barrels. In conclusion, this project was successfully able to combine an application of Hydraulics and an application of Machine Vision to deliver a successful Sorting system.

10. __References:__

(1) Home. (n.d.). Retrieved August 16, 2020, from https://forum.coppeliarobotics.com/viewtopic.php?t=2987

(2) Enabling the remote API - client side. (n.d.). Retrieved August 16, 2020, from https://www.coppeliarobotics.com/helpFiles/en/remoteApiClientSide.htm

(3) Remote API functions (Matlab). (n.d.). Retrieved August 16, 2020, from https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	920	Date	August 18,2020
Characters	7733	Exclude Url	

0% Plagiarism	100% Unique	0 Plagiarized Sentences	48 Unique Sentences

Content Checked For Plagiarism

Table of Contents 1. Title of the Project ....Page 2 2. Objective................................. ....Page 2 3. Tools & Software Required... Page 2 4. Summary ....Page 2 5. Abstracts ....Page 3 6. Introduction ....Page 3 7. Design and Analysis ....Page 3 8. Results and Discussion..Page 13 9. Conclusion..Page 15 10. References ....Page 16 Page 2 1. Title of the Project: Machine Vision & Hydraulics Combinational Project 2. Objective: To use image processing for sorting objecting with the use of hydraulic cylinders. 3. Tools/Software Required: MATLAB, CoppeliaSim. 4. Summary: This project involved building a functioning conveyer belt with the inclusion of hydraulic cylinders for sorting colored cubes. A camera was used to detect the cubes and to capture the images of the same cubes as they moved on the conveyor belt. Two different software platforms were used for this project including the MATLAB software and the Coppelia Simulation software. Both platforms communicate with each other simultaneously as the whole process begins. Coppelia Sim includes the sensors, actuators, and the complete model required for the sorting application while MATLAB is used to detect the inputs from the sensors and to send commands to the actuators in the Coppelia software. The actuators in this application are the cylinders that are used to sort the cubes. This project fuses the applications of hydraulics and machine vision into one to simulate an industrial application of sorting different objects. Page 3 5. Abstract: The purpose of this project is to build a prototype that makes use of hydraulics and machine vision for the industrial application of sorting colored objects. 6. Introduction: Hydraulics and Electro Hydraulics are widely used in the industry for many applications where power is needed. Machine vision is also used for image processing in many fields including medicine, and self- driving transportation. This project involves building an industrial prototype that involves Machine Vision and Hydraulics with the use of the MATLAB and the Coppelia Simulation software. 7. Design and Analysis: The application involves sorting two colored cubes including a red cube and a blue cube that move on a conveyer belt before being detected by two cameras. Initially, the complete scene is constructed in the Coppelia simulation environment as shown on the next page. Page 4 I.e CoppeliaSim Scene The scene hierarchy for the scene is the following: The components used include a conveyer belt, two colored cubes, two barrels, and two camera stands with two cameras attached. All these components are built in the environment and are presented in the scene hierarchy. Each camera Page 5 is used to detect a specific color by capturing and processing the images taken to then actuate the cylinder for pushing the colored cube to the assigned barrel based on the cube's color. The red cube should be pushed into the first barrel and the blue cube should be pushed in the second barrel. The sorting is performed in the Coppelia simulation software environment scene while MATLAB is used for processing the images and for actuating the cylinders. To establish a connection between MATLAB and Coppelia Simulation, the following code is entered in the child script of the 'cuboid' object present in the Coppelia Sim. software(1): " I.e object 'cuboid' From the MATLAB side, the following code must be entered in the MATLAB script(2): Page 6 "sim=remApi('remoteApi'); sim.simxFinish(-1); clientID=sim.simxStart('127.0.0.1',19999,true,true,5000,5);" The codes mentioned above initiate a remote API connection between MATLAB and CoppeliaSim. The files for the MATLAB and CoppeliaSim software are named as 'ConveyerS.m' and 'ConveyerSorting' respectively for both software and are then placed in the same folder known as 'MATLAB-VREP'. The following code was then entered in the MATLAB with the guidance of the remote API functions present in the CoppeliaSim website(3) for programming the camera and the cylinder actuators: MATLAB Code: sim=remApi('remoteApi'); sim.simxFinish(-1); clientID=sim.simxStart('127.0.0.1',19999,true,true,5000,5); if (clientID>-1). % It is checked if the connection is established between MATLAB and CoppeliaSim. disp('Connected'); Page 7 % Initially, handles are created for the cylinder actuators and the two cameras. [returnCode,cylinder1]= sim.simxGetObjectHandle(

created for the cylinder actuators and the two cameras. [returnCode,cylinder1]= sim.simxGetObjectHandle(clientID,'c1',sim.simx_opmode_blocking) %cyl1Handle [returnCode,cylinder2]= sim.simxGetObjectHandle(clientID,'c2',sim.simx_opmode_blocking) %cyl2Handle [returnCode,cameraM]= sim.simxGetObjectHandle(clientID,'camera',sim.simx_opmode_blocking) Êmera1Handle [returnCode,resolution, image1]=sim.simxGetVisionSensorImage2(clientID, cameraM,0,sim.simx_opmode_streaming); [returnCode,cameraM1]= sim.simxGetObjectHandle(clientID,'camera1',sim.simx_opmode_blocking) Êmera2Handle [returnCode,resolution, image2]=sim.simxGetVisionSensorImage2(clientID, cameraM1,0,sim.simx_opmode_streaming); for i=0:100 % a for loop with 100 iterations is used for this process [returnCode,resolution, image1]=sim.simxGetVisionSensorImage2(clientID, cameraM,0,sim.simx_opmode_buffer); Page 8 [returnCode,resolution, image2]=sim.simxGetVisionSensorImage2(clientID, cameraM1,0,sim.simx_opmode_buffer); %color matrices are for separating the red from the images. redImg1= image1(:,:,1); bluImg2= image2(:,:,1); %display camera images subplot(1,2,1); imshow(image1); subplot(1,2,2); imshow(image2); % For programming the color detection segment, a for loop is used to check all the elements in image 1 and 2 as shown below. If statements are used to compare the red values in both images with the thresholds specified below to detect the cubes. %check for the red cube. for k= 1:numel(redImg1) if redImg1(k)>220 % the threshold, in this case is 220. Page 9 disp('it is red') pause(1.48); [returnCode]=sim.simxSetJointTargetVelocity(clientID,cylinder1,1 , sim.simx_opmode_blocking) %extend cylinder 1 for 0.2 seconds pause(0.2) [returnCode]=sim.simxSetJointTargetVelocity(clientID,cylinder1,-1 , sim.simx_opmode_blocking) pause(0.6) %retract cylinder 1 for 0.6 seconds break; end end %check for blue cube for j= 1:numel(bluImg2) if bluImg2(j)>20 % the threshold in this case is 20. disp('it is blue') pause(1.48); Page 10 [returnCode]=sim.simxSetJointTargetVelocity(clientID,cylinder2,1 , sim.simx_opmode_blocking) %extend cylinder 2 for 0.2 seconds pause(0.2) [returnCode]=sim.simxSetJointTargetVelocity(clientID,cylinder2,-1 , sim.simx_opmode_blocking) %retract cylinder 2 for 0.6 seconds pause(0.6) break; end end end sim.simxFinish(-1); end sim.delete(); % Program Ends 10. References: Page 17 (1) Home. (n.d.). Retrieved August 16, 2020, from https://forum.coppeliarobotics.com/viewtopic.php?t=2987 (2) Enabling the remote API - client side. (n.d.). Retrieved August 16, 2020, from https://www.coppeliarobotics.com/helpFiles/en/remoteApiClientSide.htm (3) Remote API functions (Matlab). (n.d.). Retrieved August 16, 2020, from https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatla b.htm

Sources	Similarity

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	541	Date	August 18,2020
Characters	3462	Exclude Url	

0% Plagiarism	100% Unique	0 Plagiarized Sentences	29 Unique Sentences

Content Checked For Plagiarism

The hydraulic aspect of this project involves the use of cylinders that are actuated when the objects are detected. An equivalent hydraulic circuit for this application has been presented below but with the use of Page 11 proximity sensors instead of a camera as cameras have not been used in this Hydraulics and Electro- hydraulics course for sensing objects. The cylinders were also motorized as a hydraulic source was not available on the simulation software when the environment was being constructed. The circuit has been presented on the next few pages. Hydraulic Circuit: Page 12 Electro-hydraulic Circuit: Page 13 In the circuits shown above 'PB1' is the start button or the debug button. Once this button is pressed, 'K1' is energized which leads to the energization of solenoid 'M1'. Once solenoid 'M1' is actuated, the conveyer motor starts. 'PS1' and 'PS2' are optical proximity sensors used for sensing the red cube and the blue cube respectively. Once the red cube is detected by sensor 'PS1', K2 is energized, and thus, solenoid Y1 is energized which leads to the extension of cylinder 1 in the circuit. Once cylinder C1 is fully extended, the inductive proximity sensor 'PS3' detects it and energizes K5 which causes cylinder 1 retract as the closed contact of K5 is broken. When the blue cube is detected by sensor 'PS2', K3 is energized and thus, solenoid Y2 is energized which leads to the extension of cylinder 2 in the circuit. Once cylinder C2 is fully extended, the inductive proximity sensor 'PS4' detects it and energizes K6 which causes cylinder 2 retract as the closed contact of K2 is broken. A stop button is placed to stop the process at any time. When this button is pressed, K4 is energized. Page 14 8. Results & Discussion: To start the process, the CoppeliaSim had to be debugged first and then the MATLAB program. Once both programs were running, the conveyer belt started mocking with the two colored cubes. When the first camera detected the red cube, the cylinder extended and pushed that red cube in barrel one. I.e Red Cube Detected I.e Red Cube Sorted The blue cube came next and was ignored by the camera 1. As soon as the blue cube was detected by the second camera, the second cylinder extended and pushed that blue cube into the second barrel. Page 15 I.e Blue Cube Detected I.e Blue Cube Sorted Thus, the sorting process was successfully completed with both cubes sorted to their respective barrels. Page 16 I.e Both Cubes Sorted. 9. Conclusion: In this project, a functioning sorting system was constructed and simulated using MATLAB and CoppeliaSim. Images were acquired from Coppelia Sim and were transferred to MATLAB for image processing. Once the cubes were detected, their respective cylinders were actuated from MATLAB to sort those cubes in their respective barrels. In conclusion, this project was successfully able to combine an application of Hydraulics and an application of Machine Vision to deliver a successful Sorting system. 10. References: Page 17 (1) Home. (n.d.). Retrieved August 16, 2020, from https://forum.coppeliarobotics.com/viewtopic.php?t=2987 (2) Enabling the remote API - client side. (n.d.). Retrieved August 16, 2020, from https://www.coppeliarobotics.com/helpFiles/en/remoteApiClientSide.htm (3) Remote API functions (Matlab). (n.d.). Retrieved August 16, 2020, from https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatla b.htm

Sources	Similarity