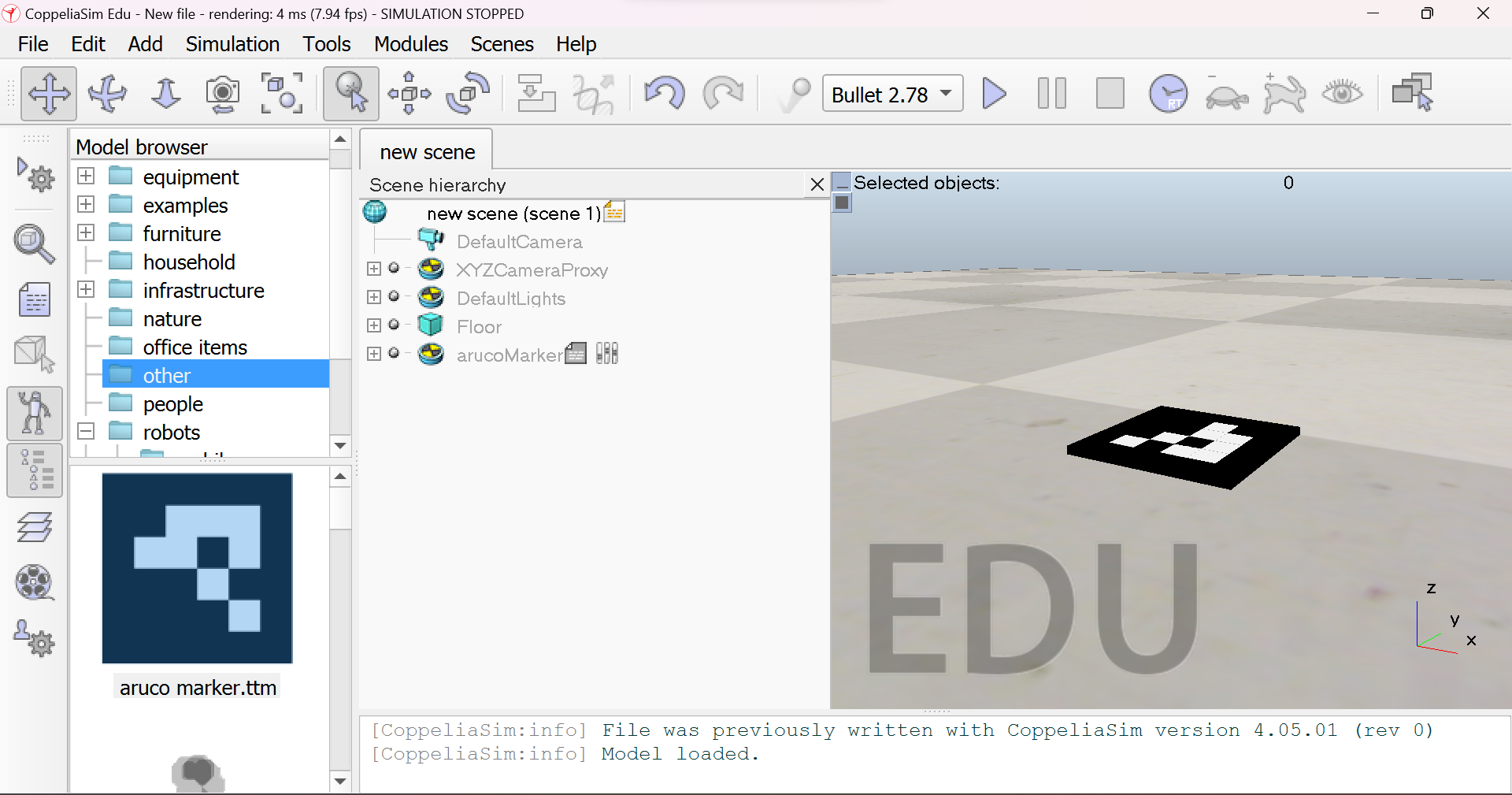
Task 3

Importing and tracking Aruco marker in VREP

Task 3A: Importing aruco marker in VREP

In the latest version of vrep, Aruco marker is present as a model. Hence it is extremely easy to import.

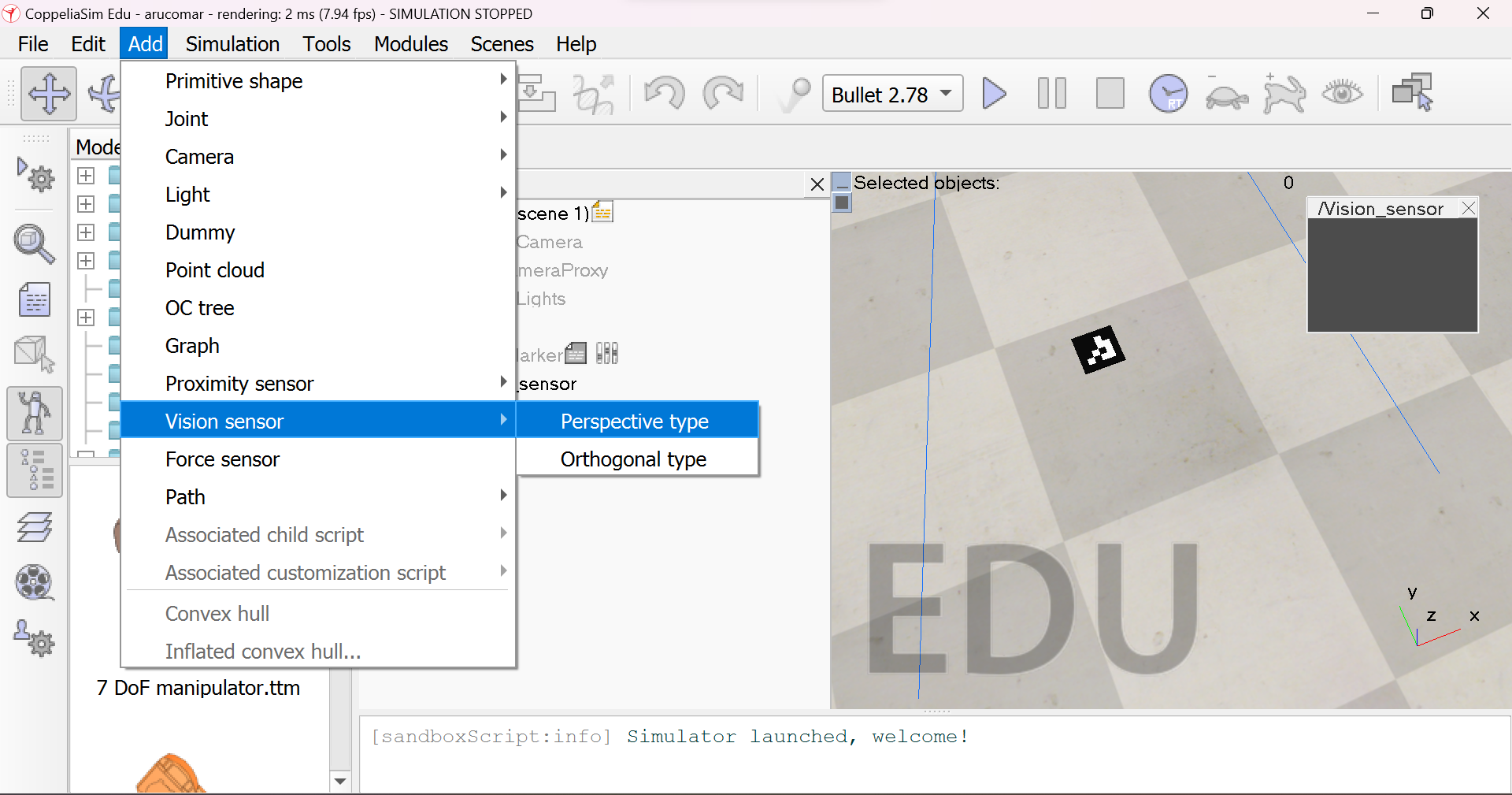
Click on the ‘other’ section in the model browser and choose Aruco marker.



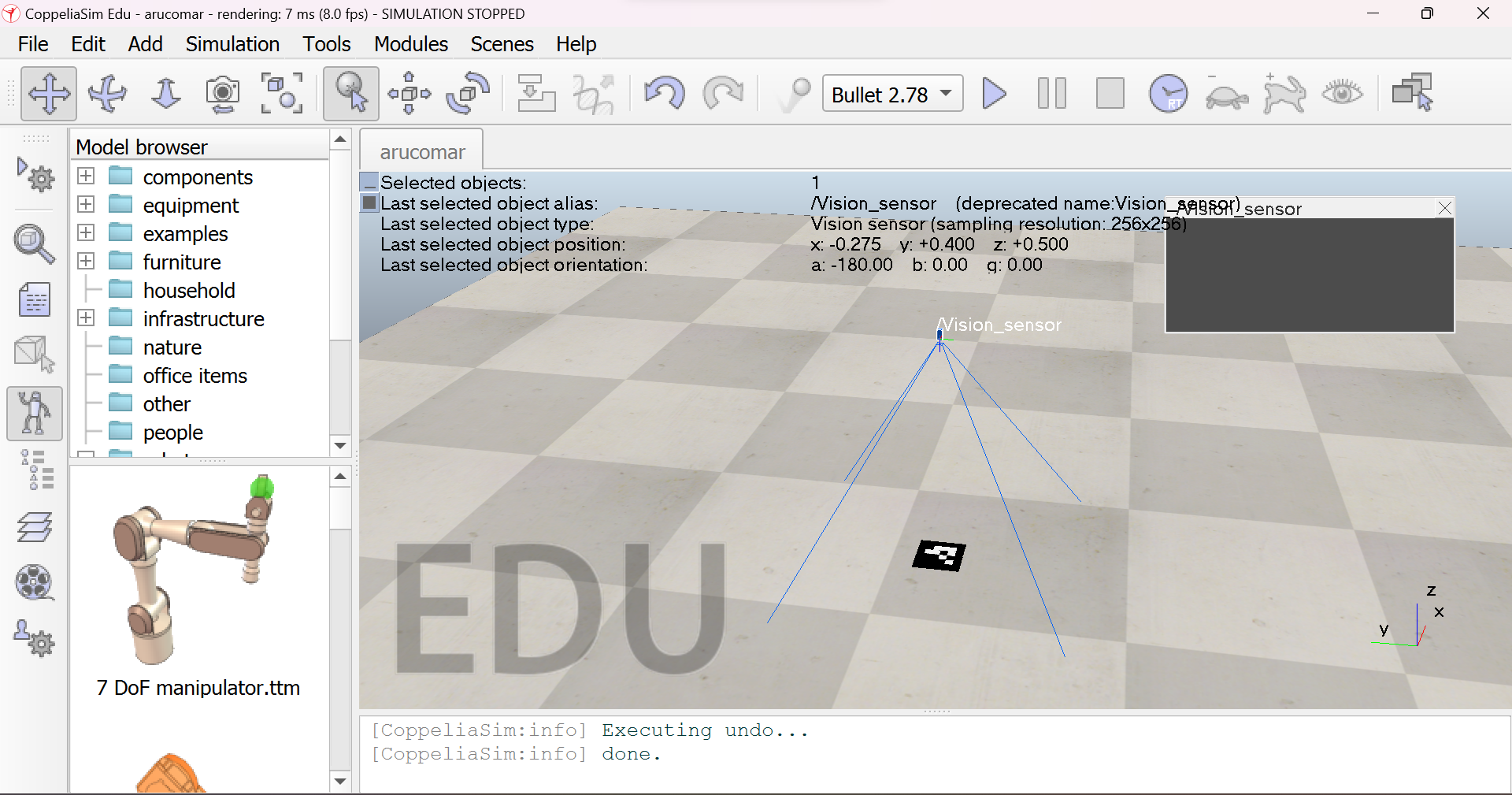
Task 3B

Adding a vision sensor to capture the image and send to matlab

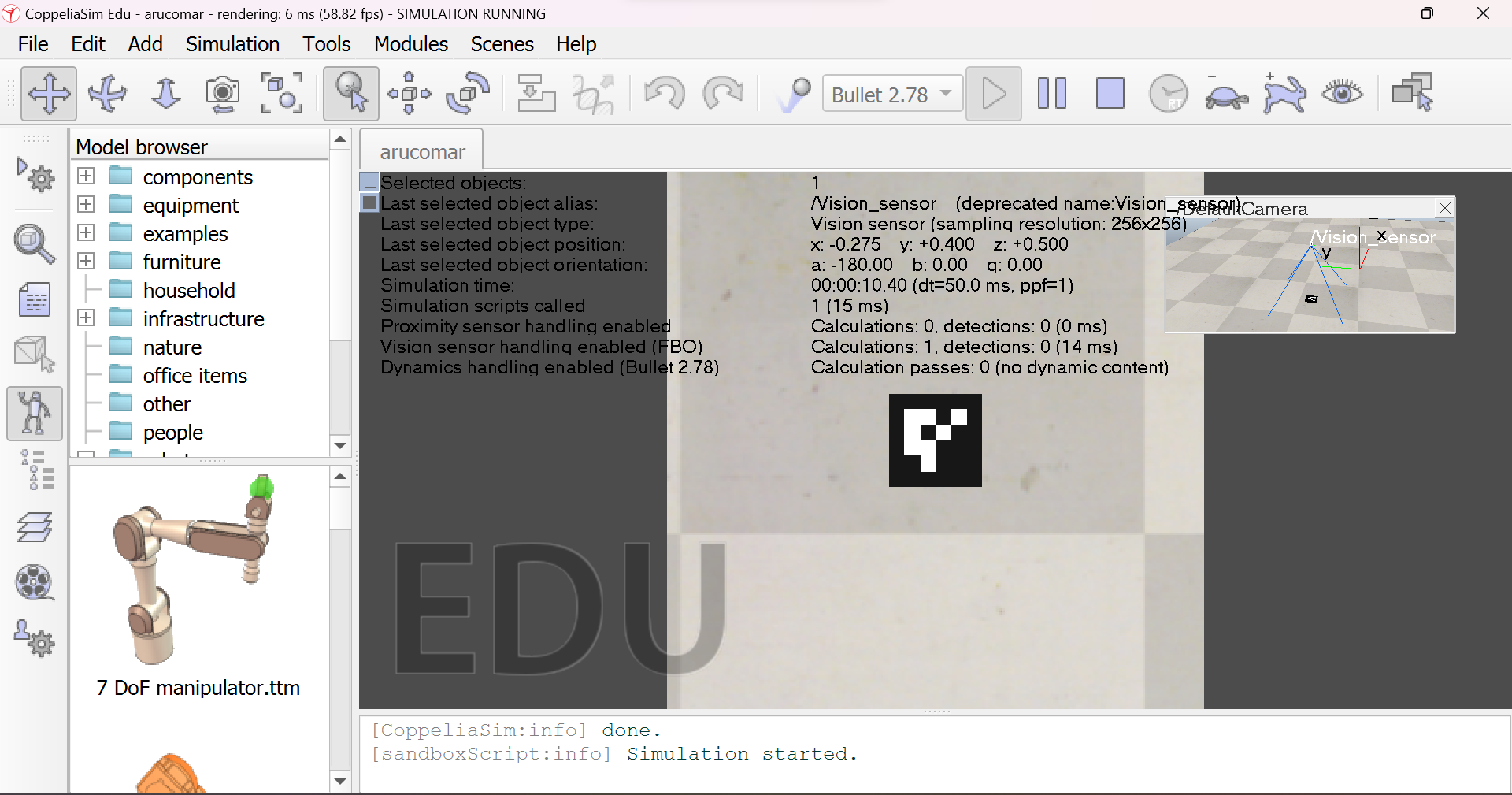
1. Add the perspective view of the vision sensor



1. The coordinates of the vision sensor can be set as follows to view the aruco marker.



1. Image as seen by vision sensor



1. Writing code in matlab to obtain the image from vision sensor

printf('Program started\n')

client = RemoteAPIClient();

sim = client.getObject('sim');

camera=sim.getObject('/Vision\_sensor');

sim.startSimulation();

for i=1:35

[imagevrep,resolution]=sim.getVisionSensorImg(camera,1);

resolution=cell2mat(resolution);

imagevrep=reshape(imagevrep,resolution(1),resolution(2));

imagevrep=im2gray(imagevrep);

imshow(imagevrep)

sim.setObjectPosition(camera,camera,{0 0 0.02});

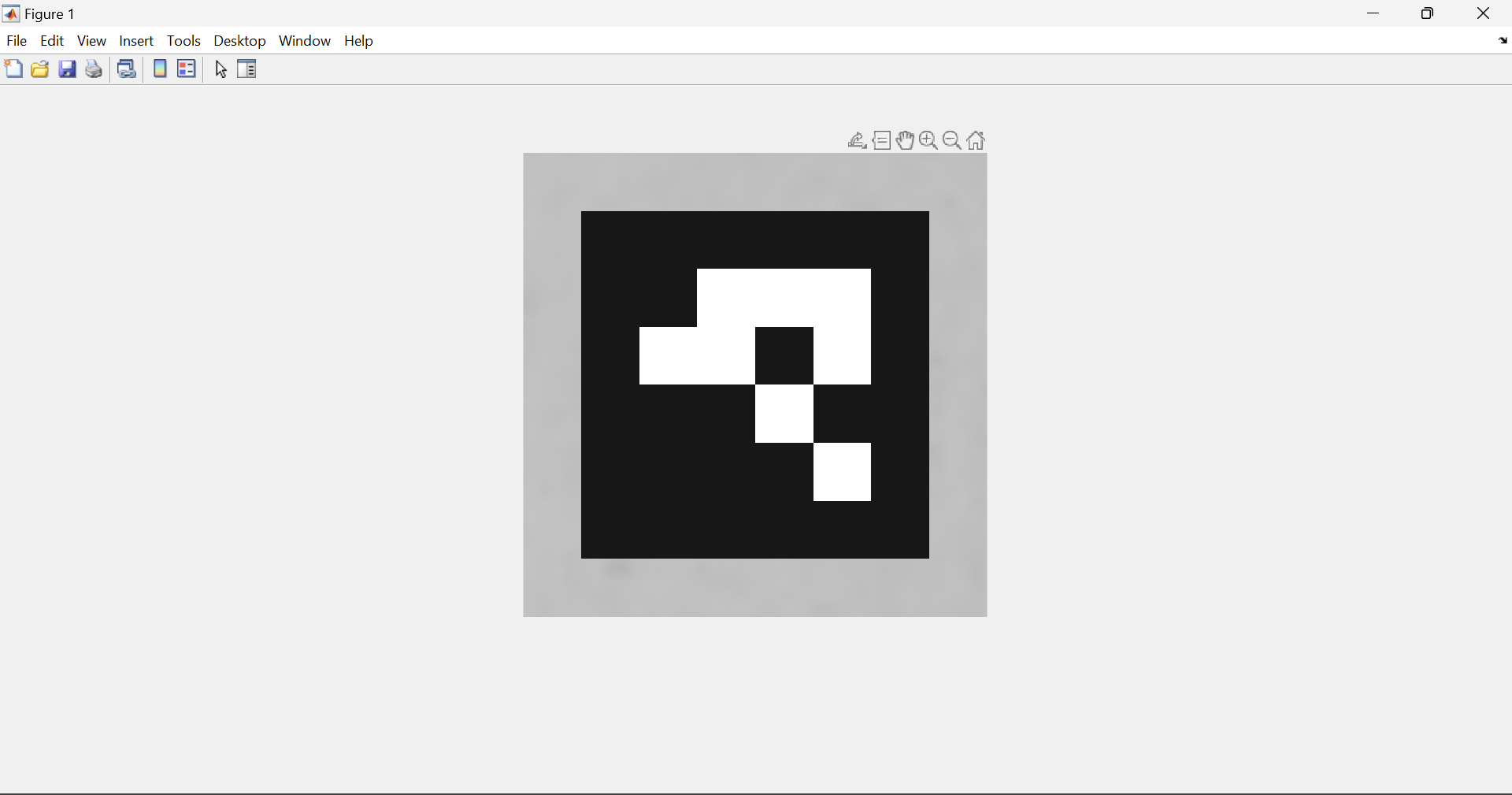
pause(0.1);

end

sim.stopSimulation();

The above program receives the image from the vision sensor and increases the camera’s position by 0.02 every iteration.

Here we use the function sim.getVisionSensorImg() to obtain the image and its resolution with arguments as the handle of the vision sensor and an option implying a RGBA output or greyscale image.Output image in matlab is as follows.



Task 3C: Tracking the aruco marker from matlab

OpenCv library provides functions that help in tracking the aruco marker. Hence, we integrate the OpenCv software with MATLAB to use these functions.

1. Installation of required software:
   * 1. The Computer Vision, Image processing, and OpenCV interface toolboxes in MATLAB were installed.
     2. Since there are no direct functions in the OpenCV interface, we need to use the mexOpenCV provision in the interface to use the functions. With this, we can directly use programs written in C++ in MATLAB.
     3. The other required softwares were installed as followed in the link:

<https://in.mathworks.com/matlabcentral/answers/372403-how-to-use-aruco-marker-with-the-opencv-support-package-of-matlab-and-the-computer-vision-system-too>

1. Next,we write the C++ program that takes in an image as input and returns another image with the aruco id identified as output. Note: We need to know the dictionary of the aruco marker in order to track it. In this case it is 4x4 dictionary with max id as 1000.
2. Code:
3. #include "opencvmex.hpp"
4. #include <opencv2/aruco.hpp>
5. #include <vector>
6. #include <iostream>
7. using namespace cv;
8. using namespace std;
9. //////////////////////////////////////////////////////////////////////////////
10. // Check inputs
11. //////////////////////////////////////////////////////////////////////////////
12. void checkInputs(int nrhs, const mxArray \*prhs[])
13. {
14. if (nrhs != 2)
15. {
16. mexErrMsgTxt("Incorrect number of inputs. Function expects 2 scalar inputs.");
17. }
18. }
19. ///////////////////////////////////////////////////////////////////////////
20. // Main entry point to a MEX function
21. ///////////////////////////////////////////////////////////////////////////
22. void mexFunction(int nlhs, mxArray \*plhs[], int nrhs, const mxArray \*prhs[])
23. {
25. cv::Mat img;
26. ocvMxArrayToImage\_uint8(prhs[0], img);
27. int dictionaryId = (int)mxGetScalar(prhs[1]);
29. std::vector<int> markerIds;
30. std::vector<std::vector<cv::Point2f>> markerCorners, rejectedCandidates;
31. aruco::DetectorParameters detectorParams = aruco::DetectorParameters();
32. aruco::Dictionary dictionary = aruco::getPredefinedDictionary(aruco::PREDEFINED\_DICTIONARY\_NAME(dictionaryId));
33. aruco::detectMarkers(img, dictionary,markerCorners, markerIds,detectorParams,rejectedCandidates);
34. int sizearr[2];
35. if(markerCorners.size()==1)
36. {
37. cv::Point2f center(0, 0);
38. for (size\_t j = 0; j < 4; j++) {
39. center += markerCorners[0][j];
40. }
41. center\*=0.25;
42. cv::Point centerInt(cvRound(center.x), cvRound(center.y));
43. sizearr[0]=centerInt.x;
44. sizearr[1]=centerInt.y;
45. }
46. plhs[0]=mxCreateNumericMatrix(1,2,mxINT32\_CLASS,mxREAL);
47. int \*outputarray=(int\*)mxGetData(plhs[0]);
48. outputarray[0]=sizearr[0];
49. outputarray[1]=sizearr[1];
50. }

Given any input image, this function returns the center of the aruco marker in the image if present else it return 0,0.

1. Importing this file onto MATLAB using the mexOpenCV interface, we can track the aruco marker
2. Once the center of the marker is obtained, a target position can be sent to the drone according to the difference between the center of the image and the marker.
3. The drone then centers along the marker and then lands on it

Final MATLAB Code:

fprintf('Program started\n')

client = RemoteAPIClient();

sim = client.getObject('sim');

%Handles

targetObj=sim.getObject('/target');

d=sim.getObject('/Quadcopter/base');

heli=sim.getObject('/Quadcopter');

camera=sim.getObject('/Quadcopter/Vision\_sensor');

sim.setObjectOrientation(targetObj,sim.handle\_world,{0 0 0});

sim.setObjectPosition(heli,sim.handle\_world,{0 0 1});

sim.setObjectPosition(targetObj,sim.handle\_world,{0 0 1});

kp=2;

kd=10;

laste=0;

vparam=-2;

prevy=0;

prevRollE=0;

prevx=0;

prevPitchE=0;

prevEuler=0;

tarO=[0 0 1];

pos=sim.getObjectPosition(d,sim.handle\_world);

targetPos=1;

client.setStepping(true);

sim.startSimulation();

r=0.5;

while sim.getSimulationTime()==0 || sim.getSimulationState()==sim.simulation\_stopped

pause(0.01);

client.step();

end

while true

t=sim.getSimulationTime();

[imagevrep,resolution]=sim.getVisionSensorImg(camera,1);

resolution=cell2mat(resolution);

imagevrep=reshape(imagevrep,resolution(1),resolution(2),[]);

center=getaruco3(imagevrep,3)

[ny,nx]=size(imagevrep);

C= round([nx ny]/2);

dif1=double(C(1)-center(1));

dif2=double(C(2)-center(2));

sim.setObjectPosition(targetObj,targetObj,{-0.0001\*dif1 0.0001\*dif2 0});

%Altitude Control

if dif1==dif2 && dif2==0 && targetPos>=0.13

targetPos=targetPos-0.03;

end

p=sim.getObjectPosition(d,sim.handle\_world);

pos=p(3);

pos=cell2mat(pos);

vel=sim.getVelocity(heli);

l=cell2mat(vel(1,3));

e=(targetPos-pos);

angvel=7.02+kp\*e+kd\*(e-laste)+vparam\*l;

laste=e;

object=sim.getObjectPosition(d,sim.handle\_world);

tar=[0.003\*dif1 -0.003\*dif2 0];

%tar=[cell2mat(object(1))-tarO(1) cell2mat(object(1))-tarO(2) cell2mat(object(1))-tarO(3)];

%tar=sim.getObjectPosition(d,targetObj)

m=sim.getObjectOrientation(d,sim.handle\_world);

%roll

roll=0.03\*(tar(2)+6.5\*(tar(2)-prevy));

prevy=tar(2);

rollE=roll-cell2mat(m(1));

rollCof=2\*rollE+7\*(rollE-prevRollE)+0.1\*cell2mat(vel(2));

prevRollE=rollE;

cell2mat(m(1));

%pitch

pitch=-0.03\*(tar(1)+6.5\*(tar(1)-prevx));

prevx=tar(1);

pitchE=pitch-cell2mat(m(2));

pitchCof=2\*pitchE+7\*(pitchE-prevPitchE)-0.1\*cell2mat(vel(1));

prevPitchE=pitchE;

%yaw

rotCorr=cell2mat(m(3))\*0.1+2\*(cell2mat(m(3))-prevEuler);

prevEuler=cell2mat(m(3));

angvel1=angvel-pitchCof+rollCof+rotCorr;

angvel2=angvel+pitchCof+rollCof-rotCorr;

angvel3=angvel+pitchCof-rollCof+rotCorr;

angvel4=angvel-pitchCof-rollCof-rotCorr;

sim.callScriptFunction('setVel@/Quadcopter',sim.scripttype\_childscript,angvel1,angvel2,angvel3,angvel4);

client.step(); % triggers next simulation step

end

sim.stopSimulation();