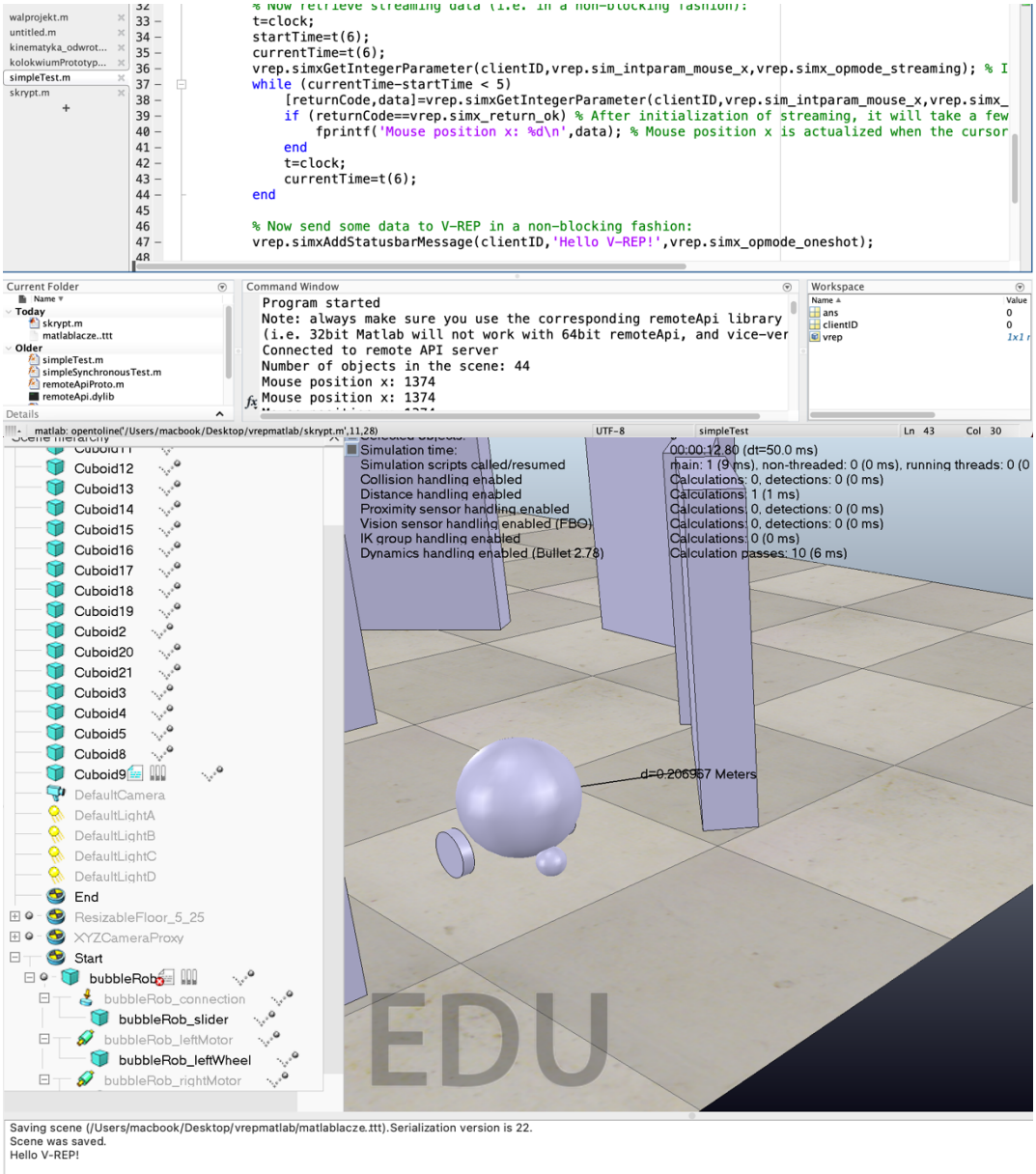
Sprawozdanie
Kamil Warchoł

Celem tych zajęć jest przetestowanie możliwości połączenia Vrepa z Matlabem oraz pisania oraz wykonywania skryptów napisanych w Matlabie do sterowania Vrepem.

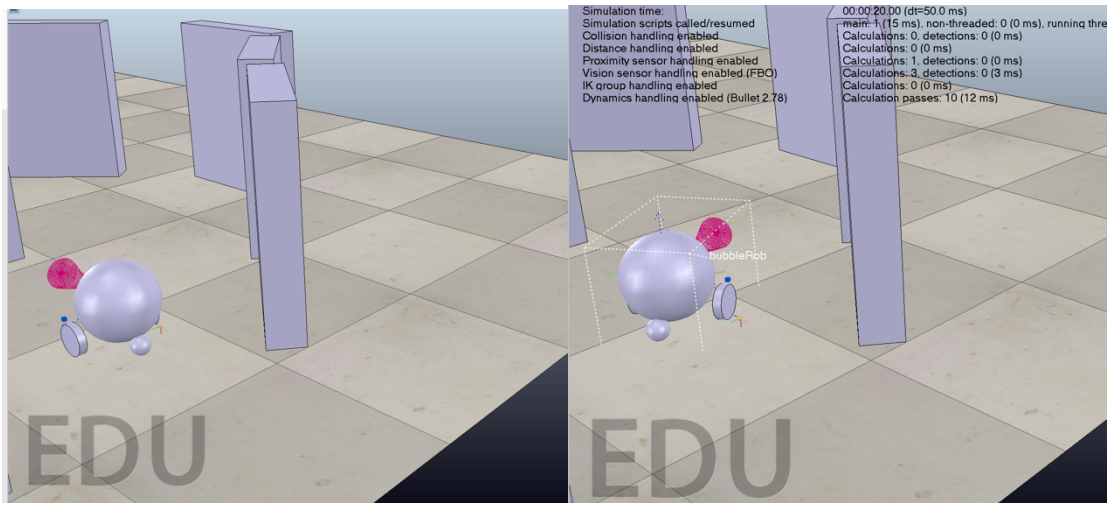
Połączenie

Po wykonaniu instrukcji łączenie Vrepa z Matlabem przebiegło pomyślnie



Silniki – wysyłanie danych do Vrepa

Przetestowane zostało uruchamianie silników oraz ich wyłączenie. Na poniższych screenach widać, że test przebiegł pomyślnie.



Pozycja robota przed oraz po uruchomieniu skryptu w Matlabie

Użyty w tym celu skrypt:

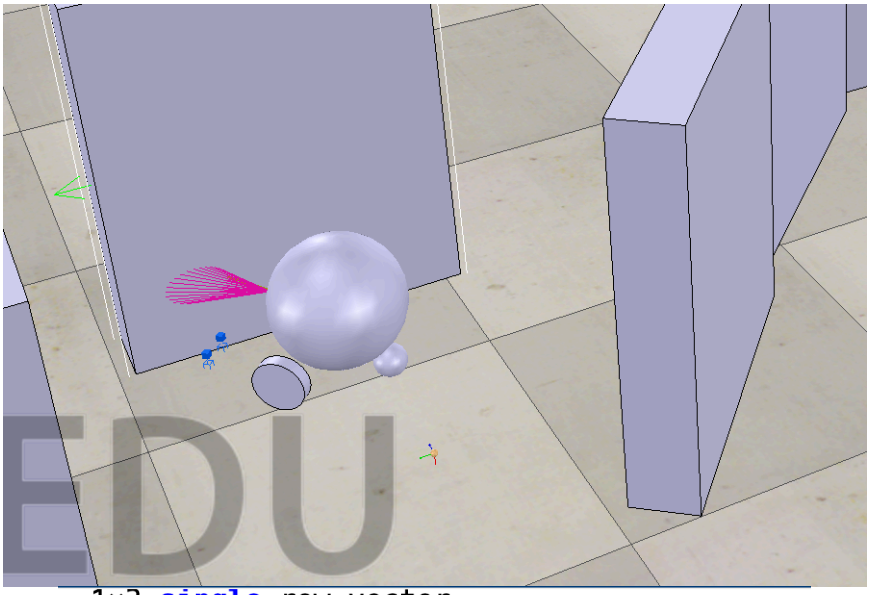
```
vrep=remApi('remoteApi');
vrep.simxFinish(-1); % just in case, close all opened connections
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);

[returnCode,
left_Motor]=vrep.simxGetObjectHandle(clientID,'bubbleRob_leftMotor'
,vrep.simx_opmode_blocking)
[returnCode,
right_Motor]=vrep.simxGetObjectHandle(clientID,'bubbleRob_rightMotor'
,vrep.simx_opmode_blocking)

[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0.5,vrep.s
imx_opmode_oneshot)

[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0,vrep.sim
x_opmode_blocking);

% Before closing the connection to V-REP, make sure that the last command
sent out had time to arrive. You can guarantee this with (for example):
vrep.simxGetPingTime(clientID);
% Now close the connection to V-REP:
vrep.simxFinish(clientID);
vrep.delete(); % call the destructor!
```

Detekcja przeszkód

Korzystając z czujnika odległości, w którego posiadaniu jest bubbleRob napisany został zgodnie z instrukcją skrypt pozwalający na zbieranie z niego danych.

```
vrep=remApi('remoteApi');
vrep.simxFinish(-1); % just in case, close all opened connections
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);

[returnCode,
left_Motor]=vrep.simxGetObjectHandle(clientID,'bubbleRob_leftMotor'
,vrep.simx_opmode_blocking)
[returnCode,
right_Motor]=vrep.simxGetObjectHandle(clientID,'bubbleRob_rightMotor'
,vrep.simx_opmode_blocking)
[returnCode,
front_sensor]=vrep.simxGetObjectHandle(clientID,'bubbleRob_sensingNose'
,vrep.simx_opmode_blocking)

[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0.5,vrep.s
imx_opmode_oneshot)
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,0.5,vrep.
simx_opmode_oneshot)

[returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySensor(
clientID,front_sensor,vrep.simx_opmode_streaming)
for i=1 : 50
[returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySensor(
clientID,front_sensor,vrep.simx_opmode_buffer)
    pause(0.1)
end

[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0,vrep.sim
x_opmode_blocking)

% Before closing the connection to V-REP, make sure that the last command
sent out had time to arrive. You can guarantee this with (for example):
vrep.simxGetPingTime(clientID);
% Now close the connection to V-REP:
vrep.simxFinish(clientID);
vrep.delete(); % call the destructor!
```

Po przestawieniu ściany zaplanowana zostałą kolizja, aby zobaczyć czy i jak zmieniać się będą dane uzyskiwane z czujnika.

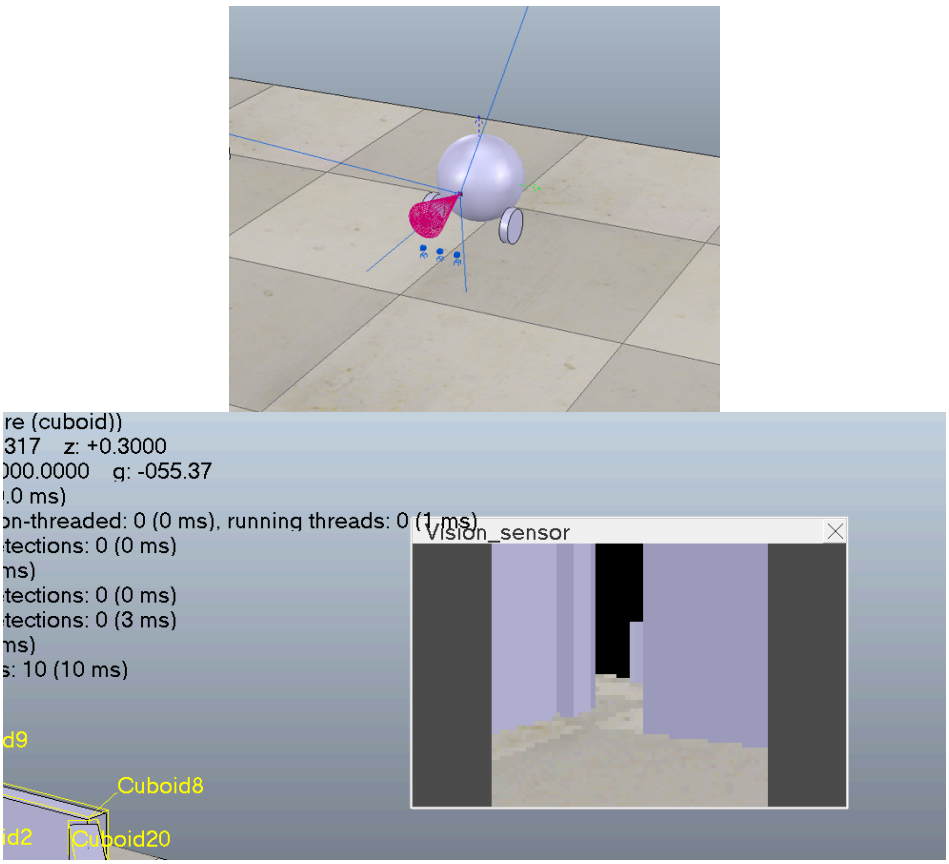


```
  1×3 single row vector

    0.0004   -0.0000    0.1589

returnCode =

     0

detectionState =

  uint8

    1

detectedPoint =

  1×3 single row vector

    0.0003   -0.0000    0.1560
```

Odbiór danych z czujnika przebiega prawidłowo

Kamera

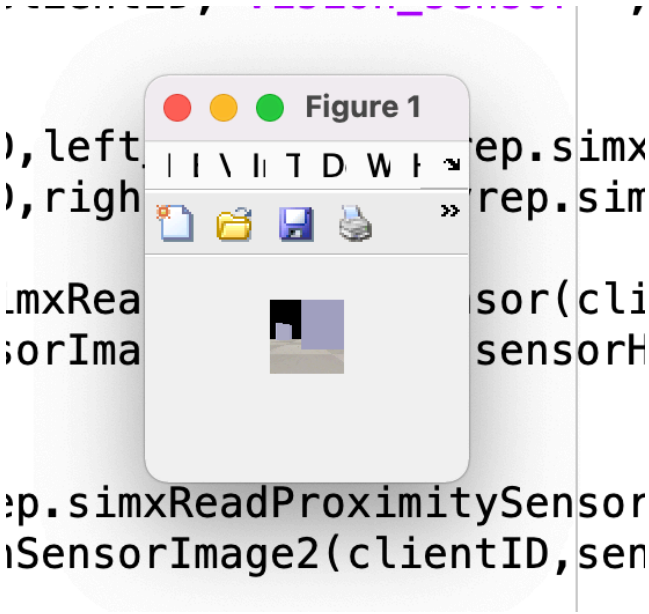
Z sukcesem udało się umieścić na bubbleRobie kamerę oraz ją skonfigurować.



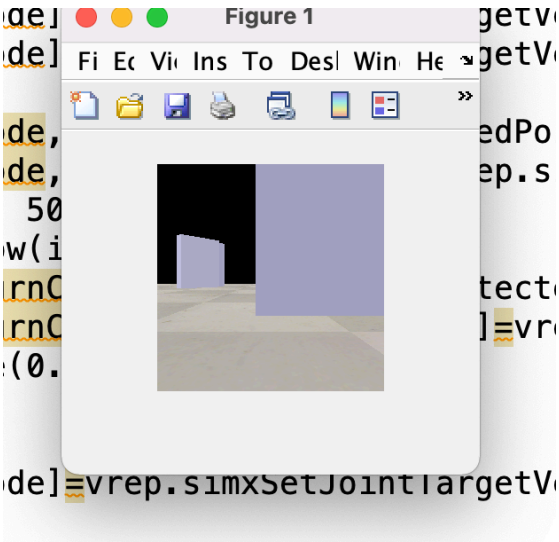
Skrypt do wyświetlania obrazu z kamery w Matlabie wygląda następująco:

```
vrep=remApi('remoteApi');
vrep.simxFinish(-1); % just in case, close all opened connections
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);

[returnCode,
left_Motor]=vrep.simxGetObjectHandle(clientID,'bubbleRob_leftMotor'
,vrep.simx_opmode_blocking)
[returnCode,
right_Motor]=vrep.simxGetObjectHandle(clientID,'bubbleRob_rightMotor'
,vrep.simx_opmode_blocking)
[returnCode,
front_sensor]=vrep.simxGetObjectHandle(clientID,'bubbleRob_sensingNose'
,vrep.simx_opmode_blocking)
[returnCode,
sensorHandle]=vrep.simxGetObjectHandle(clientID,'Vision_sensor'
,vrep.simx_opmode_blocking)


[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0.5,vr
ep.simx_opmode_oneshot)
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,0.5,v
rep.simx_opmode_oneshot)

%[returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySe
nsor(clientID,front_sensor,vrep.simx_opmode_streaming)
[returnCode,
resolution,image]=vrep.simxGetVisionSensorImage2(clientID,sensorHandle,
1,vrep.simx_opmode_streaming);
for i=1 : 50
    imshow(image);
   %
[returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySen
sor(clientID,front_sensor,vrep.simx_opmode_buffer)
    [returnCode,
resolution,image]=vrep.simxGetVisionSensorImage2(clientID,sensorHandle,
1,vrep.simx_opmode_buffer);
    pause(0.1);
end

[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0,vrep
.simx_opmode_blocking);
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,0,vre
p.simx_opmode_blocking);

% Before closing the connection to V-REP, make sure that the last
command sent out had time to arrive. You can guarantee this with (for
example):
vrep.simxGetPingTime(clientID);
% Now close the connection to V-REP:
vrep.simxFinish(clientID);
vrep.delete(); % call the destructor!
```

Z użyciem rozdzielczości 64x64 uzyskany obraz był bardzo mały – kształty przeszkód ledwo widoczne



Rozdzielczość została zwiększona do 128x128, ten rezultat był już dużo lepszy



Następnie rozdzielczość została zwiększona do 256x256 – ta wielkość obrazu była już bardzo zadowalająca, lecz w przypadku obrazu kolorowego płynność uległa dużemu pogorszeniu. Zastosowano skale szarości, co pozwoliło naprawić płynność i uzyskać optymalnie najbardziej satysfakcjonujący obraz.