UNIVERSITY OF PORTSMOUTH

Automated Self-Storage Management system using Industrial Robots

(December 2019)

Nikolas Markides, Member, IEEE

Abstract *— This paper overviews the concept of implementing an autonomous warehouse management solution using a virtual environment (v-rep). An automated self storage warehouse is a facility which stores a customers belongings or physical documentation in movable storage units which are stored throughout the facility. The aim of this paper is to investigate the usage of industrial robots in self-storage facilities to add a new level of efficiency to a traditional warehouse and also to visualise the application of this robotic autonomous solutions.*

Index Terms *— Robotics, Industrial Robots, Self-storage, warehouse management, youbot, vrep.*

1. INTRODUCTION

• *F.A. Author is with the National Institute of Standards and Technology*

Since the advent of robots, work has been shared between man and machine. As robots become more technologically advanced , they learn to even surpass humans achieving tasks better and faster. Industrial robots are used in the mining, manufacturing, military and public safety industries just to name a few, replacing humans in cases where the tasks are tedious, repetitive, unpleasant or even hazardous for humans. Additionally in today's market where competition is at an all time high and pressure for better quality at lower prices is rising, companies opt to reduce human labour with the more time and cost efficient robotic solutions [2]. While the initial investment of advancing their facilities can be costly, in the long run the company will have a better probability in competing in the future of their industry. Although functioning, exact replicas of humans remain technologically impossible, this is in part because robots are not yet fit to access new uncertain environments before completing some basic steps [3]. My scenario will be a self-storage warehouse company that is in need of robotic solutions to lower labour costs and improve efficiency in the warehouse. Furthermore these solutions will make this warehouse autonomous with little to no need for human employees to be present during the retrieval, storing and delivery of the customer's packages.

While many logistics and manufacturing operations still rely on manual and paper-based picking systems , autonomous mobile robots can now eliminated this old-fashioned way with improvements in sensors and artificial intelligence these machines can be deployed virtually anywhere. Companies such as IAM Robotics and GreyOrange offer robotic solutions that can add a new level of efficiency to a company with intelligent software and hardware. The Falcon Project is an acronym for Flexible Automated Logistics concepts, the main objectives of this project has been decomposed into three objectives: 1) the development of tools and methods for the design of warehouses, 2) the development of control methods to make a warehouse system perform as intended and 3) the development of robotic item-picking solutions [4] .

2. Approach

More warehouses are adopting robotics technology that ever before. The warehouse robotics market was valued at $2.28 billion in 2016 and is expect to grow at CAGR of 11.8% between 2017 and 2022, reaching $6 billion in value by 2022 [5]. In this scenario a self-storage warehouse would like to advance the process of storing and serving packages with automation in order to reduce the manual labour costs by eliminating the need for human labour drastically and also to have a less time consuming solution for the management of customers belongings.

The warehouse robotics industry includes several types of robots, serving a variety of purposes and functions, some type include, automated storage and retrieval systems, Goods-to-person technology and Automated guided Vehicles.

1) ***Automated storage and retrieval systems (AS/RS***) - automating the inventory process by retrieving goods and then returning items back to their proper storage locations.

2) ***Goods -to-person technology (G2P)*** - Includes goods-to-person picking the deliver items to picking stations. Operators fill orders as items are delivered to them.

3) ***Automated Mobile Robots*** - Autonomous mobile robots that rely on sensors to navigate more flexible routes by interpreting their environment. Furthermore when combined with RFID-tagged equipment these type of robots can conduct inventory counts at predetermined time frames.

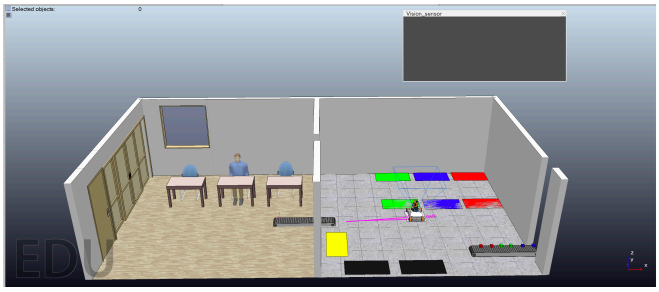
In a real life application of my scenario Automated mobile robots would fit the criteria since they are autonomous, can handle new environments and also with the help of rfid technologies many of the steps in the warehouse can be automated. A good example of an automated mobile robot the warehouse can use is the Kiva robot with the ability to carry 1000 pounds of load and also work for a couple of hours continuously before heading to a charging station for 5-minute recharge before continuing work. I will be using the virtual environment V-rep and the mobile robot Kuka youbot to visualise the solution to the problem outlined. The Kuka youbot would not be a good fit for the real life application of this scenario since it cannot carry heavy objects but it will fit to visualise this process.

— Scenario Visualisation:

In the image below we can see the simulation of the self storage warehouse.

There are two rooms in the scene, the first room on the right illustrates what happens within the storage area where the youbot will manage storage and retrieval of the boxes. The second room on the left represents a waiting room where customers request a box and wait for it to be served to them. There exist two conveyor belts, the first conveyor belt in the room on the right is where a number of customers have put their personal belongings into standard sized storage boxes, printed the label corresponding to their box and attached onto the box and lastly placed it on the conveyor belt (the coloured boxes represent different customers). Now the conveyor belt stops and waits for the robot to pickup a box and place it to a corresponding storage area (represented by the coloured pads) before moving the next box in line. After all boxes are stored. If a new box arrives on the belt, the robot will then go pickup the box and store it . If a customer

arrives and requests to receive their already stored package back, the robot will then find the corresponding box reques_ ted , pick it up and place it on the second conveyor belt which will in tern deliver it to the customer. Lastly the youbot will travel back to the white panel which is there to represent the robots charging station.



— Youbot Manipulation :

1. Get handles for gripper target and tip to handle pickup & drop. Cuboid handles which are named based on the colour of the cube and represent the customers boxes in order to pickup and drop the boxes.

```
function sysCall_threadmain()
    gripperTarget=sim.getObjectHandle('youBot_gripperPositionTarget')
    gripperTip=sim.getObjectHandle('youBot_gripperPositionTip')
    vehicleReference=sim.getObjectHandle('youBot_vehicleReference')
    vehicleTarget=sim.getObjectHandle('youBot_vehicleTargetPosition')
    redBox1=sim.getObjectHandle('Cuboid2')
    redBox2= sim.getObjectHandle('Cuboid3')
    greenBox1= sim.getObjectHandle('Cuboid4')
    greenBox2= sim.getObjectHandle('Cuboid7')
    blueBox1= sim.getObjectHandle('Cuboid5')
    blueBox2= sim.getObjectHandle('Cuboid6')
```

2. Get places 1-5 which are dummies used to get the cuboids positions and then the position to drop the objects . Place 1 is used to give the location of the conveyor belt pickup position. Place 2-4 are used to give the position where the cuboids should be dropped based on the colour of the cuboid. Place 5 will on the second conveyor belt where the robot will place a cuboid to be delivered to the customer.

```
ik1=sim.getIkGroupHandle('youBotUndamped_group')
ik2=sim.getIkGroupHandle('youBotDamped_group')
ikWithOrientation1=sim.getIkGroupHandle('youBotPoseAndOrientUndamped_group')
ikWithOrientation2=sim.getIkGroupHandle('youBotPoseAndOrientDamped_group')
gripperCommunicationTube=sim.tubeOpen(0,'youBotGripperState'..sim.getNameSuffix(nil),1)
place1=sim.getObjectHandle('place1')
place2=sim.getObjectHandle('place2')
place3=sim.getObjectHandle('place3')
place4=sim.getObjectHandle('place4')
place5=sim.getObjectHandle('place5')
```

3. Pickup objects and drop to platform

```
pickup1={0,-14.52*math.pi/180,-70.27*math.pi/180,-95.27*math.pi/180,0*math.pi/180}
pickup2={0,-14.52*math.pi/180,-70.27*math.pi/180,-95.27*math.pi/180,0*math.pi/180}
pickup3={0,-14.52*math.pi/180,-70.27*math.pi/180,-95.27*math.pi/180,90*math.pi/180}
pickup4={0,-14.52*math.pi/180,-70.27*math.pi/180,-95.27*math.pi/180,90*math.pi/180}

platformDrop1={0,54.33*math.pi/180,32.88*math.pi/180,35.76*math.pi/180,0*math.pi/180}
platformDrop2={0,40.74*math.pi/180,45.81*math.pi/180,59.24*math.pi/180,0*math.pi/180}
platformDrop3={0,28.47*math.pi/180,55.09*math.pi/180,78.32*math.pi/180,0*math.pi/180}
```

4. Open and close youbot gripper.

```
openGripper=function()
    simTubeWrite(gripperCommunicationTube,simPackInt32Table({1}))
    simWait(0.8)
end
closeGripper=function()
    simTubeWrite(gripperCommunicationTube,simPackInt32Table({0}))
    simWait(0.8)
end
```

5. Set cuboid drop height based on location, Set distance from object, IK speed, FK speed, IK acceleration, FK acceleration and Jerk

```
dist1=0.2
dropHeight1=0.155
dropHeight2=0.155
dropHeight3=0.155
dropHeight4=0.155
dropHeight5=0.655
ikSpeed={0.2,0.2,0.2,0.2}
ikAccel={0.1,0.1,0.1,0.1}
ikJerk={0.1,0.1,0.1,0.1}
fkSpeed={1,1,1,1,1}
fkAccel={0.6,0.6,0.6,0.6,0.6}
fkJerk={1,1,1,1,1}
```

6. Pickup first redbox and then drop it to place 3 (red pad). Duplicated for all boxes to pickup and then place to the correct pad. Also dropToplace function for every second same coloured box is dropped at 0.1 from the dummy to be dropped next to the last box on the pad.

```
-- redBox1 pickup:
m,angle=getBoxAdjustedMatrixAndFacingAngle(redBox1)
sim.setObjectPosition(vehicleTarget,-1,{m[4]-m[1]*dist1,m[8]-m[5]*dist1,0})
sim.setObjectOrientation(vehicleTarget,-1,{0,0,angle})

waitToReachVehicleTargetPositionAndOrientation()
sim.rmlMoveToJointPositions(armJoints,-1,nil,nil,fkSpeed,fkAccel,fkJerk,pickup1,nil)
setIkMode(true)
p=sim.getObjectPosition(gripperTarget,-1)
p[1]=m[4]
p[2]=m[8]
p[3]=m[12]
sim.rmlMoveToPosition(gripperTarget,-1,-1,nil,nil,ikSpeed,ikAccel,ikJerk,p,nil,nil)
closeGripper()
p[3]=p[3]+0.05
sim.rmlMoveToPosition(gripperTarget,-1,-1,nil,nil,ikSpeed,ikAccel,ikJerk,p,nil,nil)

-- redBox1 drop:

dropToPlace(place3,0,dropHeight1,pickup1,false)
```

7. Pickup & drop box to the customer. Robot travels to pickup a coloured box and then place on the second conveyor belt using a dummy (place 5). The box is then stopped using the conveyor belt script and waits for the customer to pickup the box.

```
  -- bluebox1 pickup2:
m,angle=getBoxAdjustedMatrixAndFacingAngle(blueBox1)
sim.setObjectPosition(vehicleTarget,-1,{m[4]-m[1]*dist1,m[8]-m[5]*dist1,0})
sim.setObjectOrientation(vehicleTarget,-1,{0,0,angle})
waitToReachVehicleTargetPositionAndOrientation()
sim.rmlMoveToJointPositions(armJoints,-1,nil,nil,fkSpeed,fkAccel,fkJerk,pickup1,nil)
setIkMode(true)
p=sim.getObjectPosition(gripperTarget,-1)
p[1]=m[4]
p[2]=m[8]
p[3]=m[12]
sim.rmlMoveToPosition(gripperTarget,-1,-1,nil,nil,ikSpeed,ikAccel,ikJerk,p,nil,nil)
closeGripper()
p[3]=p[3]+0.05
sim.rmlMoveToPosition(gripperTarget,-1,-1,nil,nil,ikSpeed,ikAccel,ikJerk,p,nil,nil)

--bluebox1 drop2:
dropToPlace(place5,0,dropHeight4,pickup1,false)


setFkMode()
sim.rmlMoveToJointPositions(armJoints,-1,nil,nil,fkSpeed,fkAccel,fkJerk,pickup3,nil)
sim.setObjectPosition(vehicleTarget,-1,{0,0,0})
sim.wait(2)
sim.setObjectOrientation(vehicleTarget,-1,{0,0,0})
waitToReachVehicleTargetPositionAndOrientation()
sim.stopSimulation()
d
```

8. Youbot travels back to the starting position and then end simulation after all tasks are done and simulation stops.

```
    sim.setObjectPosition(vehicleTarget,-1,{0,0,0})
    sim.wait(2)
    sim.setObjectOrientation(vehicleTarget,-1,{0,0,0})
    setFkMode()
    sim.rmlMoveToJointPositions(armJoints,-1,nil,nil,fkSpeed,fkAccel,fkJerk,platformIntermediateDrop,nil)
    waitToReachVehicleTargetPositionAndOrientation()
    sim.stopSimulation()
end
```

— Conveyor Belt:

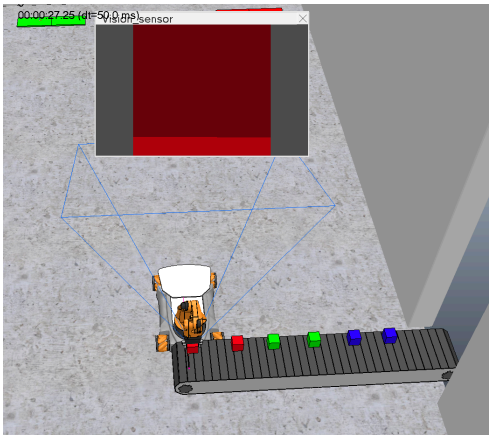
By following lab 8 to understand how the conveyor belt stops when object is detected and continues when removed.

1. Script attached to conveyor belt to have the belt stop by setting the belt velocity=0 when an object passes through the ray type proximity sensor. Then the belt continues to move after the box is picked up and stops when sensing a new object pass through the sensor. This script was given to both conveyor belts to stop the boxes before pickup.

```
function sysCall_init()

    pathHandle=sim.getObjectHandle("ConveyorBeltPath#")
    forwarder=sim.getObjectHandle("ConveyorBelt_forward#" )
    sensor=sim.getObjectHandle("conveyorbelt_sensor#1" )
    sim.setPathTargetNominalVelocity(pathHandle,0) -- for backward compatibility

end

function sysCall_cleanup()

end

function sysCall_actuation()

    beltVelocity=sim.getScriptSimulationParameter(sim.handle_self,"conveyorBeltVelocity")
    if (sim.readProximitySensor(sensor)>0) then
        beltVelocity=0
    end
    local dt=sim.getSimulationTimeStep()
    local pos=sim.getPathPosition(pathHandle)
    pos=pos+beltVelocity*dt
    sim.setPathPosition(pathHandle,pos) -- update the path's intrinsic position


    -- at each simulation pass (while not forgetting to set its initial velocity vector) :

    relativeLinearVelocity={beltVelocity,0,0}
    -- Reset the dynamic rectangle from the simulation (it will be removed and added again)
    sim.resetDynamicObject(forwarder)
    -- Compute the absolute velocity vector:
    m=sim.getObjectMatrix(forwarder,-1)
    m[4]=0 -- Make sure the translation component is discarded
    m[8]=0 -- Make sure the translation component is discarded
    m[12]=0 -- Make sure the translation component is discarded
    absoluteLinearVelocity=sim.multiplyVector(m,relativeLinearVelocity)
    -- Now set the initial velocity of the dynamic rectangle:
    sim.setObjectFloatParameter(forwarder,sim.shapefloatparam_init_velocity_x,absoluteLinearVelocity[1])
    sim.setObjectFloatParameter(forwarder,sim.shapefloatparam_init_velocity_y,absoluteLinearVelocity[2])
    sim.setObjectFloatParameter(forwarder,sim.shapefloatparam_init_velocity_z,absoluteLinearVelocity[3])
end
```

— Vision Sensor:

Following the tutorial in Lab 6 I attempted to add a vision sensor for colour detection and to organise Boxes based on their colour rather than by a predefined path. The vision sensor visualises the process of reading a customers label. The sensor has been attached to the gripper tip in order for the camera to view what the gripper is pointing at. For the colour detection I followed the BlobDetectionwithPickAndPlace example as guidance but was unable to work it out effectively.

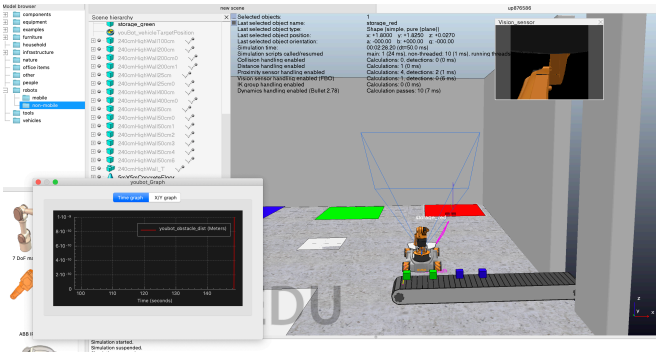


— Distance Detection:

By following Lab6 to understand how object distance between two objects is achieved.

1. In the picture below we can see the path the youbot takes and also the distance between objects and the robot.

2. Attached a graph to the youbot which calculates the time taken for the robot to follow its path and also the distance between objects (in meters) within the scene.



3. Result Analysis

— Starting with the Lab work between week 5-8 , By understanding the usage of sensors, object localisation from a given position and also the pickup and drop functions of objects using targets. After designing the scenario using this basic functionality the scene where the youbot, cuboids, conveyor belts and sensors where added along with targets and tips for the object localisation and also the ray type sensor to stop the conveyor belts before objects fall off it.

— At this point during the robotics tutorial we were introduced to the basics of moving the youbot , the code for Inverse and forward kinematics, pick and drop , open and close the gripper, set targets. The slide release for the func-

tionality has been very helpful due to the lack of resources online in manipulating the youbot.

— By experimenting with the examples provided in the vrep platform (youBotAndHanoiTower , BarrettHandPickAndPlace) and the tutorial slides, I was then able to understand how the youbot is able to move to a primitive shape, stop before the object, open & close gripper, pickup an object and place it at a desired location using dummies before repeating for all remaining tasks. After some experimentation with the examples, I changed my youbot code to reflect the functionality for my scenario.

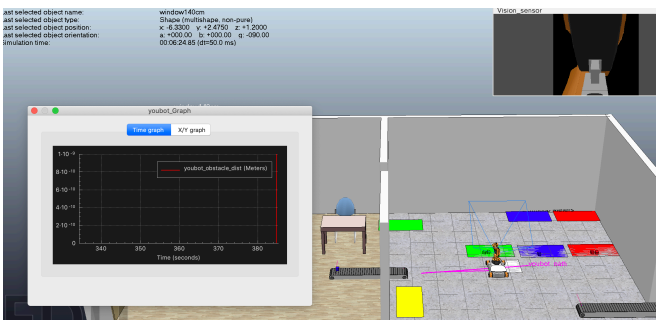
— After successfully completing the Basic functionality I attempted to complete the Extended functionality. First I added a vision sensor to implement colour detection starting with Lab6 for setting up the sensor.

— By following Lab6 I was then able to add a time graph and analyse the time taken for the youbot to complete its tasks. Furthermore the distance between objects has been implemented, in this case between the youbot and other objects in the scene.

4. Discussions and Conclusions

1) Performance:

The youbot takes approximately six minutes to store 8 boxes and then deliver one of those boxes to a customer, this time is a good time in a real life scenario but it would be beneficial to be able to reduce it by increasing the speed of the youbot and by having the youbot carry same coloured boxes on its back with the help of colour detection.



The robot completes all tasks at hand with smooth movement and no issues but if more boxes were to be stored, it could use a smarted way of navigating the warehouse with collision detection as to not disturb the objects already placed in the scene.

2) Issues encountered:

The first problem encountered was the movement of the youbot. After experimenting with the Hanoi towers example I understood that by changing the youbot script to get the coordinated of a given object and move to its location. The next issue encountered was how to pickup and move the object to a new location, the solution was to add dummies that are identified by the youbot (e.g. place1) on the place that the box should be pickup up or dropped to. Later on the next problem was to get the robot to pickup objects from the conveyor belt. To combat this issue I had to place the conveyor belt on the floor such that the robot can reach it, also I added the first dummy which was used to get the boxes on the proximity sensor location such that the target would be where the boxes would stop. Object placing issue, where the boxes would be placed on top of each other rather than next to each other. This was fixed by changing the place parameter to 0.1 rather than 0. The problems encountered up until now have all been successfully solved but some issues still remain that I was not able to find the solution to in time for submission. One of these issues is the organisation of boxes based on their colour, I was able to add the vision sensor to see other objects in the scene but I was not able to create a script to get the colour of the shape and then place it to the correct coloured pad. Furthermore I tried to implement collision detection but was unable to get it to work consistently.

3) Improvements:

To improve the scene, I would first implement colour detection such that the boxes are sorted based on their colour autonomously and not from a predefined path. Also by having colour detection I would implement an algorithm where when more than one same coloured box is detected in on the conveyor belt, the youbot would carry it on its back to reduce the execution time. Furthermore a good improvement to the existing coursework would be collision detection such that the robot would stop when detecting an object in its path and also move around it, this would be beneficial in the real life application to have the robot avoid unexpected objects in the warehouse. Lastly I would try to find a more suitable mobile robot that would have better features and to be able to carry more weight (e.g. Kuka omnirob) .

5) Conclusion :

In essence the v-rep simulation has been a good starting point in the world of robotics and has given me the opportunity to apply the outcomes learned throughout the year in a practical manner. In the future this project will be expanded to have autonomous colour detection and object collision detection such that the simulation will be more realistic to the scenario described.

5. References

- ***[1].*** t. engineering, G. family, H. sapiens, M. family, s. materials and v. effect, "US20070065259A1 - Automated self storage system - Google Patents", *Patents.google.com*, 2019. [Online]. Available: https://patents.google.com/patent/US20070065259A1/en. [Accessed: 06- Dec- 2019]

- ***[2]***. N. Pires, "Industrial Robots Programming", *Google Books*, 2019. [Online]. Available: https://books.google.co.uk/books?id=oYTg9Kn-OU4C&printsec=frontcover&dq=application+of+industrial+robots&hl=en&sa=X&ved=0ahUKEwimwcHZqqHmAhWSRBUIHZuCCecQ6AEIKDAA#v=onepage&q=application%20of%20industrial%20robots&f=false. [Accessed: 06- Dec- 2019]

- ***[3].*** J. Engleberger, "Robotics in Practice", *Google Books*, 2019. [Online]. Available: https://books.google.co.uk/books?id=r1Z-BgAAQBAJ&pg=PT76&dq=application+of+industrial+robots&hl=en&sa=X&ved=0ahUKEwimwcHZqqHmAhWSRBUIHZuCCecQ6AEINTAC#v=onepage&q=application%20of%20industrial%20robots&f=false. [Accessed: 06- Dec- 2019]

- ***[4].*** R. Hamberg and J. Verriet, "Automation in Warehouse Development", *Google Books*, 2019. [Online]. Available: https://books.google.co.uk/books?id=DYCAQ64hOsoC&printsec=frontcover&dq=warehouse+management+using+industrial+robots&hl=en&sa=X&ved=0ahUKEwilp8uuuqHmAhUEqXEKHawXD90Q6AEIKDAA#v=onepage&q=warehouse%20management%20using%20industrial%20robots&f=false. [Accessed: 06- Dec- 2019]

- ***[5].*** W. Allen, "Guide to warehouse robots: types of warehouse robots, uses & more - 6 River Systems", *6 River Systems*, 2019. [Online]. Available: https://6river.com/guide-to-warehouse-robots/. [Accessed: 06- Dec- 2019]

- [6] T. Jenkel, R. Kelly and P. shepanski, *Web.wpi.edu*, 2019. [Online]. Available: https://web.wpi.edu/Pubs/E-project/Available/E-project-031113-133138/unrestricted/Mobile_Manipulation_for_the_KUKA_youBot_Platform.pdf. [Accessed: 10- Dec- 2019].