Robotic pick and place using computer vision for object localisation and sensory real-time collision avoidance

Zak Graham Rackham
Faculty of technology
University of portsmouth
Portsmouth, UK
up899802@myport.ac.uk

***Abstract*—This paper explores the methods used in pick and place operations for object manipulation and sorting within a virtual environment (V-Rep) using computer vision for object localisation and proximity sensor measurements for object localisation and real-time collision avoidance.**

I. Introduction

A. The problem

The problem this paper will explore is object manipulation and sorting for warehouse fulfilment. Boxes are retrieved from a conveyor belt and sorted by color to the correct bay. Boxes are carried by the robots' arm or dropped to a platform attached to the robot for re-orientation and placement. The robot also needs to avoid human workers within the scene.

B. Motivation

The motivation of this paper is to further decrease item fulfilment time. Amazon uses a mobile robot to move shelving, each of these robots is equipped with a collision sensor. Amazon also utilize a robotic arm called "Robo Stow" that can move pallets and transfer them to smaller robots. After adopting the use of robotics amazon was able to cut operational costs in fulfilment centers by 20% [1]. However, robotics is not utilized for loading items into delivery vehicles.

C. State of the art

I consider the state of the art to better evaluate current research to identify how the field has developed and in witch direction it's heading and to implement appropriate methods based on their findings.

1) Software frameworks: Software framework and architecture is key in implementing various technologies for robotic applications. The robotic operating system abbreviated as "ROS" is a Linux based middle-ware that uses peer to peer communication comprised as a network of executable programs denoted as nodes that communicate with each over at runtime. Nodes are registered to a ROS master node that handles routing of information through publishing and subscribing messages via topics [3]. ROS can be implemented as a distributed system running on multiple computers or microcontrollers. This creates a dynamic and flexible environment for robotic applications. Various robotic simulation environments exist the one used in this paper is Coppelia robotics v-rep version 3.6.2.(rev0), v-rep is also capable of being inter-operable with ROS via the CoppeliaSim API framework CoppeliaSim can act as a ROS node that other nodes can communicate with via ROS services (publishers & subscribers) allowing for more comprehensive real world testing, another example of such an environment is Gazebo. These software environments provide a virtual gateway to developing and testing robotic systems and algorithms without the cost of real world hardware.

2) Hardware: Sensor technology is vital for robotics and they allow a computer to understand and interpret its environment. Proximity sensors by definition are used to detect nearness in space, there are a wide range of sensors featuring radically different operating principles [4].Ultrasonic sensors utilize ultrasonic pulses of sound to detect the presence of an object, or with additional processing, the distance to the object.They use both a transmitter and receiver and the principles of echolocation to function, by calculating the time to receive a return pulse knowing the speed of sound (dry air at 20 degrees = 343m/s) distance to an object can be calculated [4]. Photoelectric Proximity Sensors are used to detect the presence of an object but distance to the object cannot be calculated [4]. Laser and Rangefinder Sensors use the same principle of function as ultrasonic sensors, except they use light and electromagnetic beams respectively. Inductive Sensors are the most archaic yet still viable sensory technology that use magnetic induction to detect variations in a magnetic field when a metal object passes through a generated field [4].

Robotic movement is achieved using kinematic functions and actuators [8]. Various actuators exist, the most common being DC motors, stepper motors and servos. Motor drivers are used to interface with a computer to apply pulse width modulation as a principle of speed control in real-world robotics. Encoders are used to return the speed of a motor. Odometry combined with inertia measurements are necessary for accurate operation in the real-world.

D. Implementation proposal

Computer vision will be implemented for object identification and localisation using blob detection. For color detection the RGB coloration model will be used to return the RGB value of each object after pickup [2]. Within vrep a predefined function is used to retrieve an image from the vision sensor and retrieve the RGB value for a given position within the image. Both forward and inverse kinematics will be implemented to drive the robots arm joints to manipulate objects and to drive the omni-directional base to a target location within the world cartesian coordinate system. A proximity sensor will be used to detect potential collisions and will be used to achieve real-time collision avoidance by making adjustments to the robots absolute velocity to repel the robot from collideable objects within the scene.

II. Details of the approach

A. Description & contribution

1) Design: In order to address the given problem in warehouse fulfilment a robot needs to be designed to meet the motivation of this paper. Industrial robotics are offered by many companies, the robot used in this paper will be kuka's youBot. This robot consists of a omni-directional mobile platform implemented using a 4 mecanum wheel configuration this allows the youBot to achieve holonomic movement about a 2D plane [5]. A 5 degrees of freedom robotic arm is attached to the platform that can operate independently to the base. The Kuka youBot remains relevant and active in academic literature so many approaches have been developed for its operation. Using the v-rep simulation platform a virtual version of the kuka youBot is implemented. For the scenario this paper explores the use of sensory data needs to be applied, 4 ultrasonic proximity sensors are affixed on each face of youBot's plastic covering denoted as Pn.

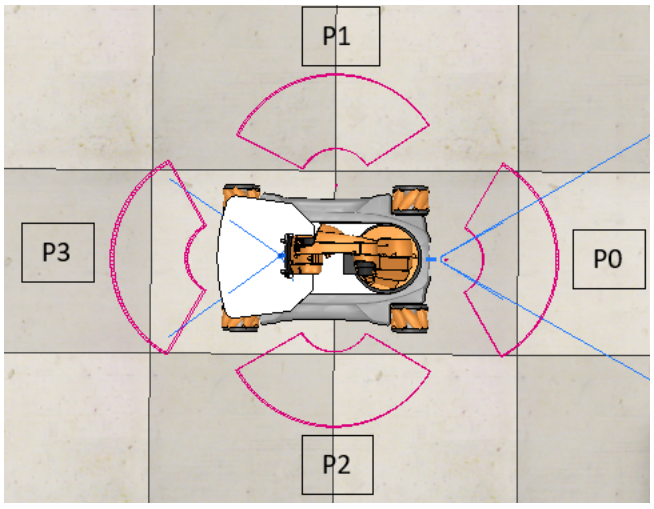

Fig. 1. Proximity sensor configuration

Cameras are also implemented, one is affixed to the front of the youBot for object localisation so its reference frame remains constant for faster data processing. To gather the required information from the sensor a blob recognition algorithm will be implemented to return the absolute location of the box within the world coordinates. A second camera is affixed to the end-effector (gripper) of youBot's arm this will be used to return the RGB value of the currently manipulated object.

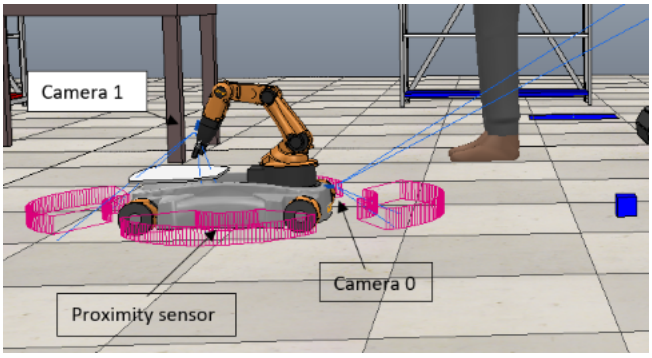

Fig. 2. Sensory configuration.

2) Movement: In order to move a robotic manipulator its kinematic structure needs to be defined. Within v-rep the scene hierarchy is used to build a kinematic structure formed as a tree of joints and links. Using the simulators' API a kinematic group can be used to build an equivalent kinematic model via the functions provided by the kinematics plugin.

Within the scenario detailed forward kinematics are used to reach a pre-defined position that will be used as a reference position for object manipulation. Forward kinematics is used in the first instance of movement where the arm is initialised to its reference position, due to the lower time complexity of the algorithm that controls its movement, setting joint angles is less processing heavy than inverse kinematics where joint angles need to be calculated based on the position of the end-effector.

First the robot's omni-directional base is translated to a new target position, each wheel joint is given a set velocity to drive the robot to the target position using forward kinematics. The arm is then actuated to its reference position using forward kinematics. Concurrently whist the arm is being actuated, the front camera is active and a reading is taken from its image frame blob detection is applied and a measurement is taken of blob1 relative to the camera frame, this is then transformed to the world cartesian coordinate system where the location of the cube can be identified. YouBot's arm can then be manipulated to the target location using inverse kinematics and the object can be picked.

3) Computer vision: For object localisation blob detection has been implemented to return the x,y location of the center of the front face of the box, however its depth cannot be perceived without the use of a collocated second camera [6]. Since the depth of each box is constant it's Y coordinate can be adjusted by half the depth of the box to correspond the new point with the center of the box. Once the object has been picked camera

1 returns the RGB value for the box being manipulated and the box is sorted accordingly to the correct drop point, youBot's base is then driven to a new target location where the box is to be dropped.

```
sortByColor=function(dropHeight)
    RGBValue=sim.getVisionSensorImage(vS0,0,0,0.1,0.1,0)
    print("RED-",RGBValue[1],"BLUE-",RGBValue[2],"GREEN- ",RGBValue[3])

    if (RGBValue[1] < 1)and(RGBValue[2] < 1)and(RGBValue[3] == 1)then
        dropToPlace(d0,0,dropHeight,pickup2,false)
    end
    if (RGBValue[1] < 1)and(RGBValue[2] == 1)and(RGBValue[3] < 1)then
        dropToPlace(d1,0,dropHeight,pickup2,false)
    end
    if (RGBValue[1] == 1)and(RGBValue[2] < 1)and(RGBValue[3] < 1)then
        dropToPlace(d2,0,dropHeight,pickup2,false)
    end
end
```

Fig. 3. Sorting algorithm using RGB values

4) Navigation: Navigation is achieved by setting a target position for the base to drive towards, the robot is driven towards the target position using forward kinematics via the shortest possible path. Obstructions to navigation are beyond the scope of this function and are handled separately since the robot is not aware of its path it cannot detect an intersection with a collideable object at this phase. Nodes can be set to intermediate points between the origin of the base and its target position, the same approach is used where the robot finds the shortest path to each node, using this method any navigation path can be achieved so the robot with a few adjustments can better adapt to a changing environment. Figure 3 shows node placement, sequence 1,2 is used when driven to node 3, node sequence 2,1 is used when returning to the pickup location.

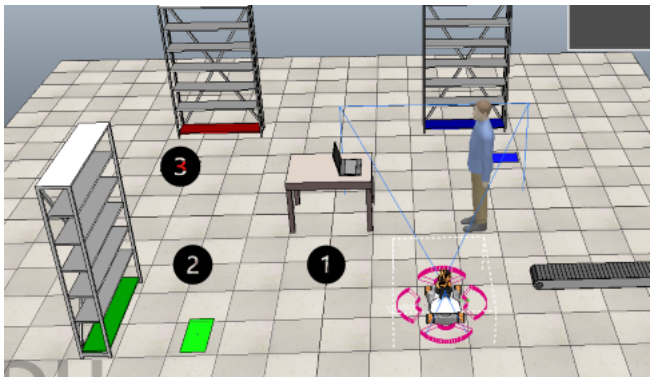

Fig. 4. Node based navigation planning

5) Collision avoidance: Collision avoidance is achieved using 4 independent ultrasonic sensors figure 5 shows the lua implementation used. Sensor P3 is reserved for stopping the robot, if a user needs the robot to stop they can approach from behind and the robot will stay in position until the user moves away. If an object is detected by P0 then the robot is driven to the right until the object is no longer detected youBot then resumes its previous trajectory. In the same way, P1 and P2 drive the robot away from a detected object right and left respectively.

```
if frontCollisionDetected == 1 then
    result,distance,detectedPoint=sim.readProximitySensor(sensor)
    forwBackVel=0
    leftRightVel=leftRightVel-leftRightVel+2
end
if leftCollisionDetected == 1 then
    result,distance,detectedPoint=sim.readProximitySensor(sensor1)
    forwBackVel=0
    leftRightVel=leftRightVel-leftRightVel+2
end
if rightCollisionDetected == 1 then
    result,distance,detectedPoint=sim.readProximitySensor(sensor2)
    forwBackVel=0
    leftRightVel=leftRightVel-leftRightVel-2
end
if rearCollisionDetected == 1 then
    result,distance,detectedPoint=sim.readProximitySensor(sensor3)
    forwBackVel=0
    leftRightVel=0
end
```

Fig. 5. An algorithm for collision avoidance

B. Justification

Each solution implemented is appropriate and works well within in this scenario. Movement has been achieved with the use of the appropriate kinematic structures and approaches where needed. Computer vision has been used to localise an object using blob detection, this was the most appropriate method in this scenario. As boxes fall from the conveyor their position is not absolute therefore the robot cannot return to a pre-defined position as pickup may fail, localising the object at each pickup is needed to achieve and accurate pick. Collision avoidance has been implemented to avoid a human worker within the scene so the pick and place task can still be completed with obstructions, due to the implementation of ultrasonic sensors any object can be avoided. The real world semantics of how this approach will work outside the virtual environment have also been considered.

III. Results Analysis

In order to test and evaluate the reliability of this approach the scenario simulation was run multiple times Table 1 below shows a scenario error test, checkpoints were set at the pick and placing of each box.

All repeats of the simulation completed as expected except for repeat 2, in this instance youBot failed to pick the second box, from my observations and quantitative analysis of the pickup point returned by computer vision is that it can be seen that the x,y position has been calculated correctly, instance without orientation change (position X = 2.15163230896, Position Y = -1.7301875352859). An instance with orientation change (position X = 2.1329510211945, position Y = -1.730813741684). x has a difference of 0.019 this is a variation in drop position and not a failure of computer vision. The failure in this instance was caused by a change in the orientation of the box since the blob detection algorithm does not consider orientation the box slips through the gripper pushed backwards rather than being picked correctly. Boxes are then sorted by color, since no color is detected due to the box not being picked the simulation moves to the next pickup. As the previous pickup has failed this effects subsequent drops so all subsequent pickups also fail.

Scanario error test											
	c0	c1	c2	c3	c4	c5	c6	c7	c8	Completed	Time
r0	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Yes	414.90625s
r1	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Yes	380.1875s
r2	Sucsess	Sucsess	Failure	Failure	Failure	Failure	Failure	Failure	Failure	Partial	256.875s
r3	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Yes	359.59375s
r4	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Yes	388.15625s
r5	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Yes	383.6875s
r6	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Sucsess	Yes	404.9375s

Fig. 6. A table showing simulation results

Collision avoidance works as intended, robotic manipulation is adjusted so the robot can still continue to operate and achieve its task correctly whilst collideable objects are within the scene. Color detection also works as intended each box is identified correctly on each pickup with no errors observed.

IV. Discussion and Conclusions

Collision avoidance works within the planned scenario, outside this however there will be instances where this approach would fail. Shown in figure 7 an object has been detected on the right of the robot and also on the left in this instance the robot changes velocity to avoid the object on the right and the left simultaneously this creates an infinite loop where the robot moves left to avoid an object and again right to avoid the object creating a left right motion and causing excessive jerk.

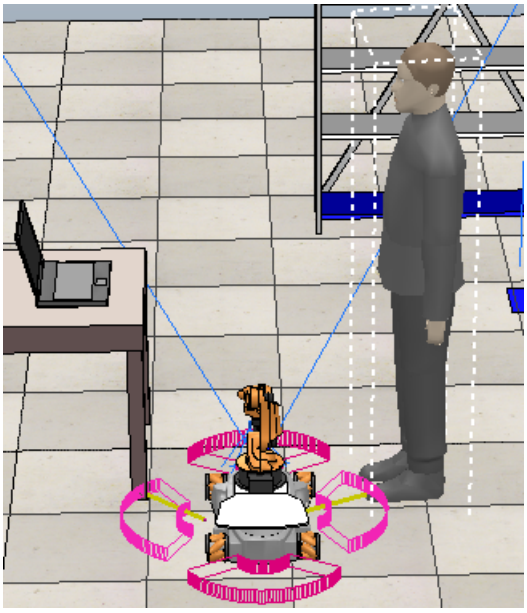

Fig. 7. Instance of failure of collision avoidance

This failure however is not fatal, once the person on the right moves away from the robot the left obstacle will be avoided and the robot will again be driven towards its target. Figure 8 shows youBot's velocity, consider the time interval between 1050 seconds to 1070 seconds this is when the robot is stuck in it's infinite loop, after the time of 1070 the human has moved and the robot can return to its previous trajectory.

The method of collision avoidance presented in this paper could be further improved by creating a more comprehensive

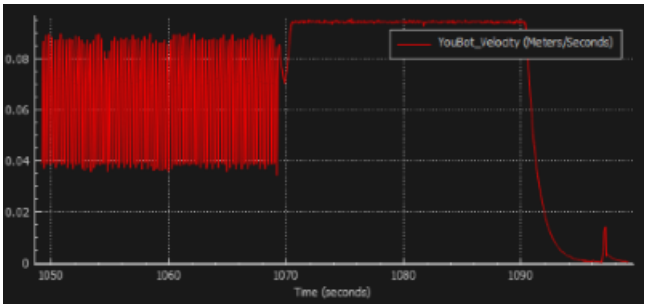

Fig. 8. Graphical representation of absolute velocity

decision tree where scenarios such as the infinite left right loop presented can be detected where a more appropriate decision can be made. Alternatively it may be appropriate given the scenario to implement another method of object avoidance by using a better method of navigation. The method presented in [7] uses a location sensor system and environment mapping to plan the robots path that can better adjust to a dynamic environment.

The method of computer vision used in this paper proved to return a fatal result where the entire simulation was effected and of consequence then failed. It may be more appropriate not to have the conveyor belt drop boxes and instead have the robot pick boxes directly from the conveyor this would decrease the potential for orientation change and would create a more reliable solution. In order to improve the current scenario in the real world a Kinect sensor can be implemented. Since the Kinect incorporates a depth sensor [8] orientation of the box can be calculated by comparing the depth value at adjacent sides of the box, if an edge is facing the depth sensor, depth values will be larger and a change in orientation can be perceived .

References

[1] Robert Bogue Growth in e commerce boosts innovation in the warehouse robot market " Industrial Robot: An International Journal 43/6 (2016) 583 587.

[2] S. Chakole and N. Ukani, "Low Cost Vision System for Pick and Place application using camera and ABB Industrial Robot," 2020 11th International Conference on Computing, Communication and Networking Technol ogies (ICCCNT), Kharagpur, India, 2020, pp. 1 6, doi:10.1109/ICCCNT49239.2020.9225522. K. Elissa, "An Overview of Decision Theory," unpublished. (Unpublished manu- script)

[3] R. Mishra and A. Javed, "ROS based service robot platform," 2018 4th International Conference on Control, Automation and Robotics (ICCAR), 2018, pp. 55-59, doi: 10.1109/ICCAR.2018.8384644.

[4] Jeff Smoot, 18/12/21, Comparing Proximity Sensor Technologies, https://www.cuidevices.com/blog/comparing-proximity-sensor-technologies

[5] R. Bischoff, U. Huggenberger and E. Prassler, "KUKA youBot - a mobile manipulator for research and education," 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 1-4, doi: 10.1109/ICRA.2011.5980575.

[6] L. Wang and H. Ju, "A Robust Blob Detection and Delineation Method," 2008 International Workshop on Education Technology and Training 2008 International Workshop on Geoscience and Remote Sensing, 2008, pp. 827-830, doi: 10.1109/ETTandGRS.2008.294.

[7] Jae-Han Park, Seung-Ho Baeg and Moon-Hong Baeg, "An intelligent navigation method for service robots in the smart environment," 2007 International Conference on Control, Automation and Systems, 2007, pp. 494-499, doi: 10.1109/ICCAS.2007.4406960.

[8] A. Basiri, M. A. Oskoei, A. Basiri and A. M. Shahri, "Improving Robot Navigation and Obstacle Avoidance using Kinect 2.0," 2017 5th RSI International Conference on Robotics and Mechatronics (ICRoM), 2017, pp. 486-489, doi: 10.1109/ICRoM.2017.8466145.

[9] E. Maulana, M. A. Muslim and V. Hendrayawan, "Inverse kinematic implementation of four-wheels mecanum drive mobile robot using stepper motors," 2015 International Seminar on Intelligent Technology and Its Applications (ISITIA), 2015, pp. 51-56, doi: 10.1109/ISITIA.2015.7219952.