

**EG2605: UNDERGRADUATE RESEARCH OPPORTUNITIES
PROGRAMME (UROP)**

PROJECT REPORT



Mobile Robot Localization: Auto-docking

Ho Kae Boon

*Department of Mechanical Engineering, College of Design and Engineering
National University of Singapore*

AY 2021/22 Semester 2 - AY2022/23 Semester 1

ABSTRACT

Localisation is a key ability of an autonomous mobile robot to identify its location before undergoing further actions. In this project, visual localisation using markers will be implemented to determine the relative position between the robot and the ARTag for auto-docking usage. This project aims to design an auto-docking algorithm of the recharging operation for a commercial mobile robot. A fiducial marker system is used to support the tracking of the robot's position and orientation. A proportional-integral control loop mechanism is implemented to receive feedback from the position and orientation of the robot and adjust the velocity. A Gazebo Simulation for the robot is set up to test the performance of the docking algorithm and tuning of the controller parameters. Finally, the algorithm is run on the physical robot for performance measure evaluation.

Keywords: *Auto-docking, Fiducial Marker, PI Controller*

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Professor Marcelo H Ang, my supervisor, for providing me with this opportunity to be involved in industrial commercial robot development, as well as for giving valuable guidance, patient feedback, and support to me throughout of the projects which without it, this paper could not have been possible. I would also like to thank the CEO of Solustar Pte Ltd, Louis Loo Kek Guan for providing the platform for me to carry out the projects. Lastly, I would also like to thank the engineers of Solustar Pte Ltd, including Jasper Tan Zhen Xuan, Ronald Wee Zhi Yang, Adriel Ho Jin Liang, Franky Laurentis and Mohammad Soleimani Amiri for lending their kind assistance throughout the projects.

TABLE OF CONTENTS

ABSTRACT	2
ACKNOWLEDGEMENTS	2
TABLE OF CONTENTS	2

1. INTRODUCTION	4
2. LITERATURE REVIEWS	4
2.1 Light source	4
2.2 Infrared Sensor	5
2.3 Mechanical Supporting Structure	5
2.4 Laser Sensor	5
2.5 Camera Visual Input	6
3. METHODOLOGY	6
3.1 ARTag Marker	6
3.2 Transformation Geometry	7
3.3 PI-control system	9
3.4 Algorithm flowchart	10
3.5 Gazebo Simulation and coefficient tuning	12
3.6 Experiment design	13
3.7 Charging station prototype	14
4. PRESENTATION OF FINDINGS AND DATA	15
4.1 Experiment 1: Test Run 10 times at $(d_{AC}, \theta) = (1.5 \text{ meters}, 0^\circ)$:	15
4.2 Experiment 2: Test Run from different d_{AC} distance and inclined angle θ :	15
4.3 Experiment 3: Special conditions	17
5. DATA ANALYSIS AND DISCUSSION	18
6. CONCLUSION	19
REFERENCES	20
APPENDICES	22
Appendix A: Auto docking script “rotate_align_entry”	22
Appendix B: ROS rqt_graph of Gazebo simulation	25

1. INTRODUCTION

Autonomous mobile robots are becoming more and more prevalent in this current age, due to their flexibility to carry out a variety of tasks. However, when there is a problem of battery energy depletion issue, a human user must manually recharge the robot. With the implementation of an auto-docking algorithm, the robot may be able to recharge itself anytime when needed by auto-docking to the charging station, thus enabling a fully autonomous robot where no human intervention is needed at any stage. This serves as a motivation for the objective of this research project, which is to design an auto-docking algorithm for the recharging operation of a commercial mobile robot, known as the Solubots Disinfectant Robot (SDR) provided by Solustar Pte Ltd. The robot hardware sensors comprise one IP65 Camera on the front top, one 180° 50m 2D Lidar on the front bottom and two Intel® RealSense™ depth camera D435 at the front and back of the robot respectively. The robot implements a differential steering mechanism, running on two driving rubber wheels on the two sides, and two omnidirectional passive wheels on the front and back. The docking process will start independently after the robot completed its navigation to the ‘opposite homing position’, which is located to be 1.5 meters away, directly facing with no inclined angle with the docking station. The charging station is planned to have two electrodes acting as the electric contact point with the back side of the robot.

2. LITERATURE REVIEWS

A literature review is conducted to gather information on different existing docking solutions.

2.1 Light source

Cassinis et. al. implemented a docking method using two light sources located at two different heights on the charging station (Cassinis et. al., 2005). Using visual sensors to check the alignment of the two light sources, it can be determined if the robot has already reached the

docking position. This method can produce robust and reliable results. However, this operation could be easily affected by the environment, for example, light disturbances.

2.2 Infrared Sensor

iRobot is an autonomous vacuum cleaning robot that uses two infrared sensors to auto-dock (Cohen et. al., 2008). Two infrared emitters are attached to the front of the charging station, while two infrared receivers are attached to the top of the robot. By detecting the alignment of the infrared beams emitted, the robot could adjust trajectory accordingly and achieve success docking with the charging station. However, this method did not produce the actual relative position of the robot with the station, thus reducing the docking efficiency.

2.3 Mechanical Supporting Structure

Some robot designs rely on rigid structures to guide the robot towards charging points. For example, Wu et. al. designed hook-shaped grabbing arms attached to the charging station to catch and guide the robot into the charging points (Wu et. al., 2009). Another robot by Song et. al., has a semicircle charging station to guide the incoming robot which also has a semicircle shell (Song et. al., 2001). This method results in the self-adjustment of the incoming robot position when the robot comes into contact with the charging station, allowing a position error of up to 7cm and an angle error of up to 60 degrees. This method could be implemented in addition to other docking methods to increase the docking success rate.

2.4 Laser Sensor

Many mobile robots use laser sensors for auto-docking. For example, Zhang et. al. proposes using the data from the lidar sensor attached to the robot (Zhang et. al., 2021). The data will undergo a segment fitting procedure to recognise the docking station feature. Then, the relative position could be calculated by partial filter positioning. A multi-state stabilization motion controller is designed to obtain the angular and linear velocity of the robot. This method has good accuracy, with a position error of less than 2cm and an angle error of less than 3°. However, the drawbacks are that lidar sensors are typically expensive.

2.5 Camera Visual Input

Many mobile robots use camera visual input for auto-docking. For example, Kartoun et. al. implemented an object recognition algorithm on the robot to recognise the docking station features (Kartoun et. al., 2006). This robot could adjust trajectory by comparing the camera input and trained image of the docking station. Meanwhile, fiducial markers are also commonly used for docking operations using camera visual input. Some popular fiducial markers choice includes AprilTag, ArUcoTag, ARToolKit and others. This project will utilize ARTag.

3. METHODOLOGY

3.1 ARTag Marker

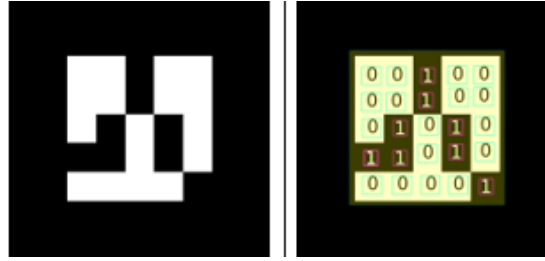


Figure 1: The left is an ARTag with id 8 generated from the ROS `ar_track_alvar` package. The right is the breakdown of the tag into a square outline and 25 interior binary values.

A visual-based localization method using an ARTag marker and visual input perception from the D435 depth camera at the back of the SDR robot is implemented. The ROS open-source `ar_track_alvar` package is used for the generation and tracking of the marker (Saito, 2016). The tags generated are bi-tonal planar patterns with a square outline, whereas the interior region is made out of a discrete grid filled with white or black cells (Fiala, 2015). The marker detection algorithm would first rectify the visual input based on the distortion parameters provided by camera info, then it locates the quadrilaterals outlines of the tags, and identify the unique id by sampling the interior into 25 binary numbers, with the black cell being 1, and the white cell being 0. With these values, the algorithm could estimate the marker position and unique id, returning a transformation geometry message.

3.2 Transformation Geometry

By setting the camera frame as the child frame, and ARTag frame as the parent frame, a 6 DOF relative position specifying the transformation from the ARTag to the camera frame could be obtained. The relative position is broken down into a translation vector and rotation quaternion. The rotation quaternion can be converted to the Euler angle.

$$\begin{aligned} trans &= \text{translation vector} = [x, y, z] \\ rotational\ quaternion &= [q_x \quad q_y \quad q_z \quad q_w] \\ rotational\ euler\ angle = [roll, pitch, yaw] &= \begin{bmatrix} \tan^{-1} \frac{2(q_w q_x + q_y q_z)}{1 - 2(q_x^2 + q_y^2)} \\ \sin^{-1}(2(q_w q_y - q_z q_x)) \\ \tan^{-1} \frac{2(q_w q_z + q_x q_y)}{1 - 2(q_y^2 + q_z^2)} \end{bmatrix} \end{aligned}$$

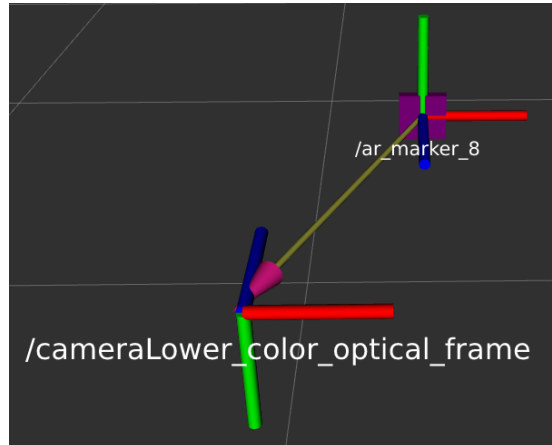


Figure 2: Transformation visualisation in RVIZ

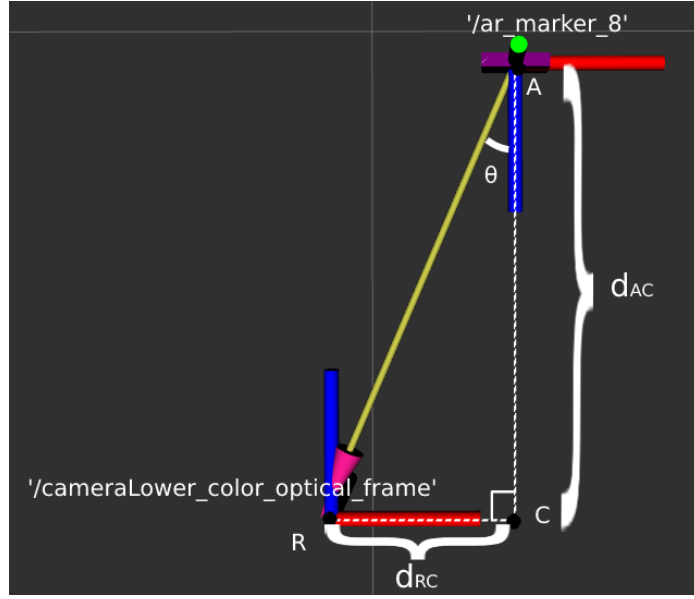


Figure 3: Top view of the coordinate frames and variables

Figure 2 shows the transformation of the coordinate frame from ARTag to the camera frame. In Figure 3, let point A be the origin of the coordinate frame of ARTag. Let point R be the origin of the coordinate frame of the robot camera. Let point C be the intersection point of two straight lines passing through A and C respectively in which the two lines are perpendicular on the same plane. Let θ be the inclined angle between AC and AR.

When the camera is directly facing the ARTag before the docking starts, their coordinate frame differs by a roll of 180° . As translation comes after rotation under ROS application (Foote, 2016), distance AR and AC can then be directly obtained from the translational vector.

$$d_{RC} = trans[0]$$

$$d_{AC} = trans[2]$$

3.3 PI-control system

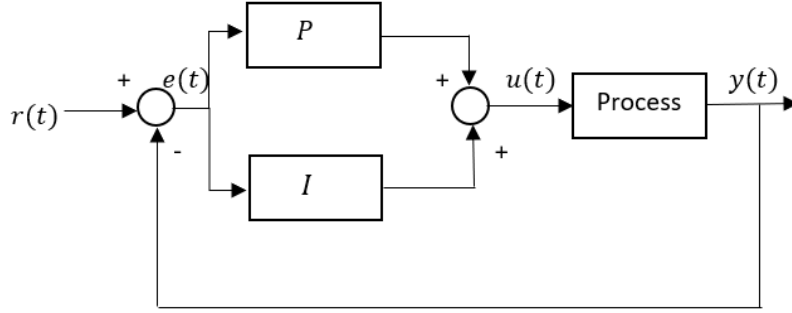


Figure 4: PI controller block diagram

Two PI-control systems are implemented for the auto-docking operation. In comparison to a pure proportional controller, the addition of an Integral term could eliminate steady-state error and accelerate the process towards setpoint. In a PI controller, an error value $e(t)$ will be calculated as the difference between the measured process variable, $PV = y(t)$ and the desired setpoint, $SP = r(t)$. A correction based on proportional (P), and integral (I) terms is applied throughout the operation to adjust the control variable, $u(t)$. The relationship between each variable is shown in Figure 4.

$$P = K_p e(t)$$

$$I = K_I \int e(\tau) d\tau$$

The first PI-control systems would take in d_{AC} value as feedback and adjust the linear velocity of the robot accordingly. It is implemented as follows:

$$y_1(t) = d_{AC}$$

$$r_1(t) = k$$

$$e_1(t) = y_1(t) - r_1(t) = d_{AC} - d$$

$$u(t) = v$$

$$P_1 = K_{p1} e_1(t)$$

$$I_1 = K_{I1} \int e_1(\tau) d\tau$$

where d is the final perpendicular distance between the camera and the ARTag, and v is the linear forward velocity of the robot.

The second PI-control system would take in d_{RC} value as feedback and adjust the angular velocity of the robot accordingly. It is implemented as follows:

$$\begin{aligned}
y_2(t) &= d_{RC} \\
r_2(t) &= 0 \\
e_2(t) &= y_2(t) - r_2(t) = d_{RC} \\
u(t) &= \omega \\
P_2 &= K_{p2}e_2(t) \\
I_2 &= K_{I2} \int e_2(\tau) d\tau
\end{aligned}$$

where ω is the angular velocity of the robot.

$K_{p1}, K_{p2}, K_{I1}, K_{I2}$ values would be tuned and obtained when running auto-docking in Gazebo Simulation via the trial-and-error method.

3.4 Algorithm flowchart

Before the start of PI-controller entry, if the ARTag is not in the camera view, the robot would spin clockwise until ARTag is detected. If the ARTag is not within $\pm 15\text{cm}$ from the middle of the camera view, the robot would spin either clockwise or anticlockwise depending on the ARTag position, to realign the ARTag to the middle of camera view. After that, the robot will proceed with the PI-control entry until it has reached the desirable distance from the ARTag.

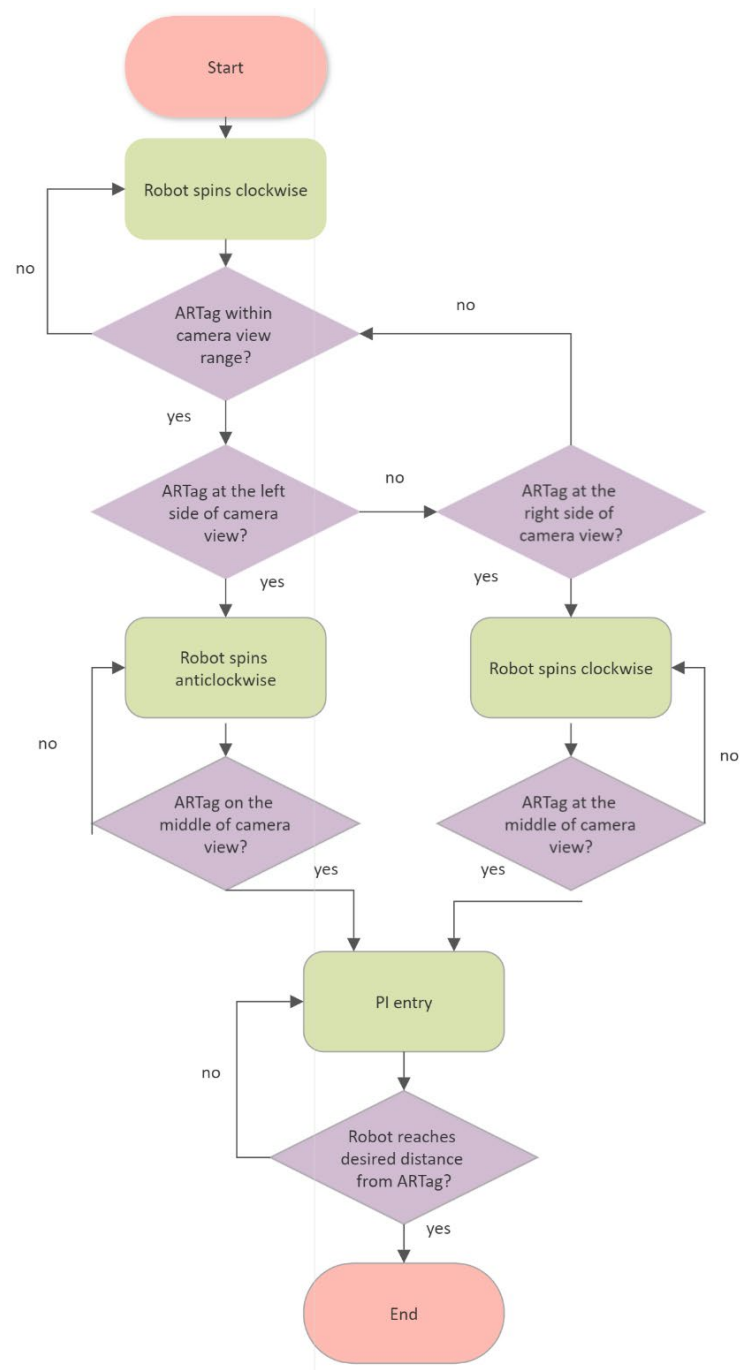


Figure 5: Block diagram for auto-docking algorithm

3.5 Gazebo Simulation and coefficient tuning

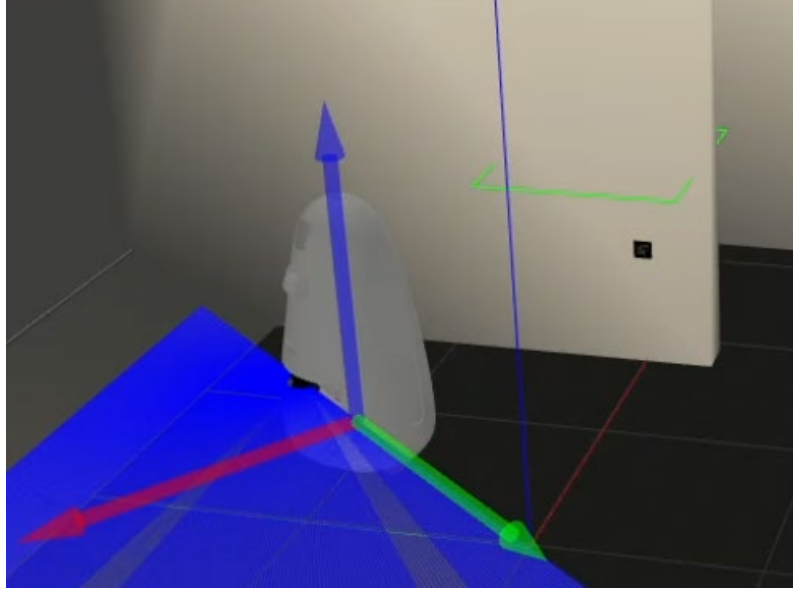
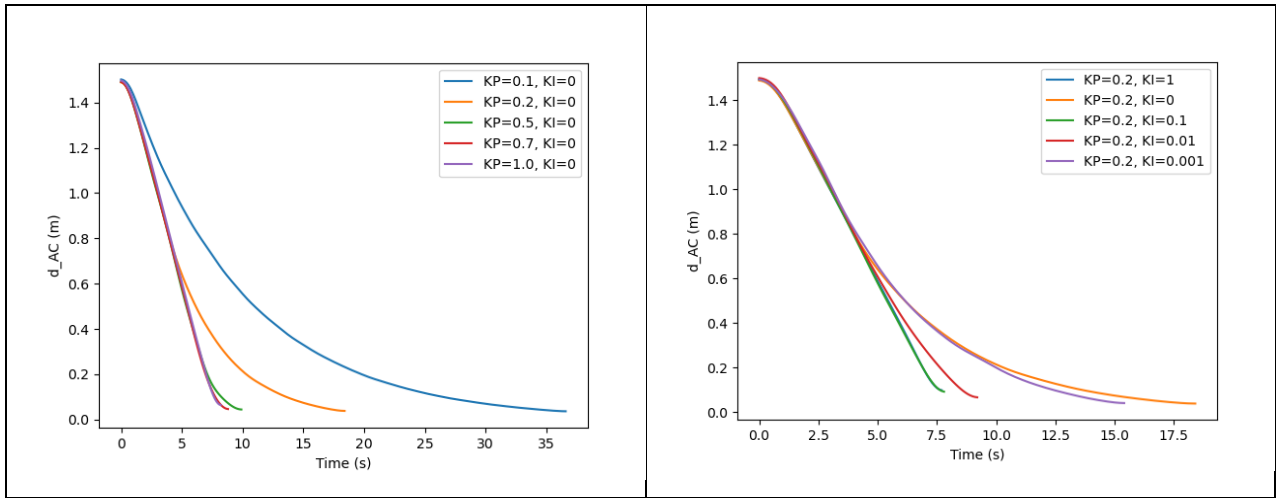


Figure 6: Gazebo simulation of SDR auto-docking operation

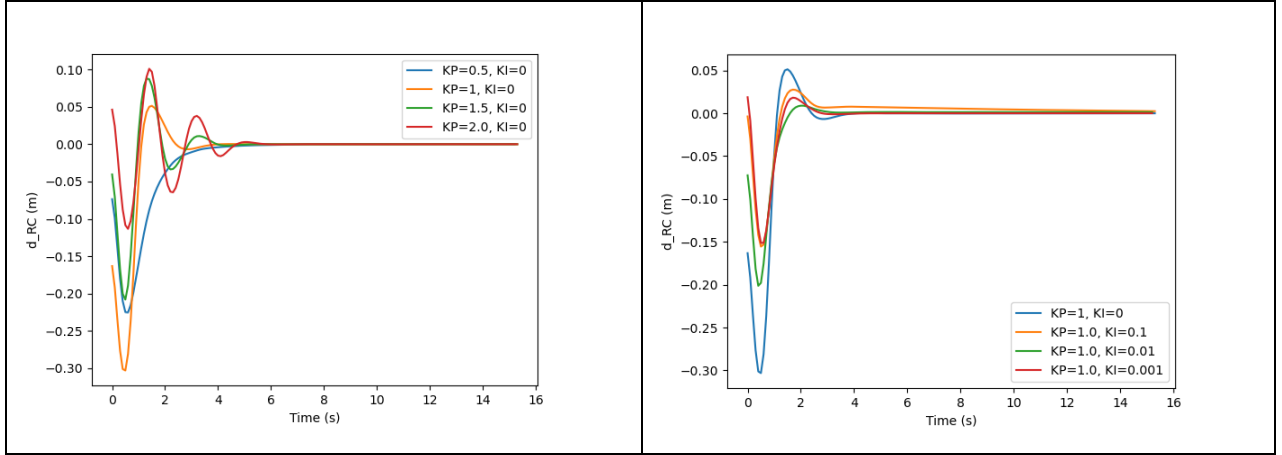
After importing the custom-made SDR and ARTag 3D models into the Gazebo Software, a simulation test run from the intended starting point of $(d_{AC}, \theta) = (1.5 \text{ meter}, 0^\circ)$ can be carried out virtually.



Graph 1(left): Graph of d_{AC} against time, varying K_P

Graph 2(right): Graph of d_{AC} against time, with $K_P = 0.2$ and varying K_I

Setting K_{I1} to zero, and varying K_{P1} values. $K_{P1} = 0.2$ is chosen as it allows reasonable docking time without sharp breaks near the ARTag. Next, setting $K_{P1} = 0.2$ and varying K_{I1} values. $K_{I1} = 0.001$ is chosen as the most ideal value.



Graph 3(left): Graph of d_{RC} against time, varying K_{P2}

Graph 4(right): Graph of d_{RC} against time, with $K_{P2} = 1.0$ and varying K_{I2}

The robot will now start facing left. Setting K_{I2} to zero, and varying K_{P2} values, $K_{P2} = 1.0$ is chosen as it has a quick rise time and lower percentage overshoot. Next, setting $K_{P2} = 1.0$ and varying K_{I2} values. $K_{I2} = 0.01$ is chosen as it has the lowest percentage overshoot.

3.6 Experiment design

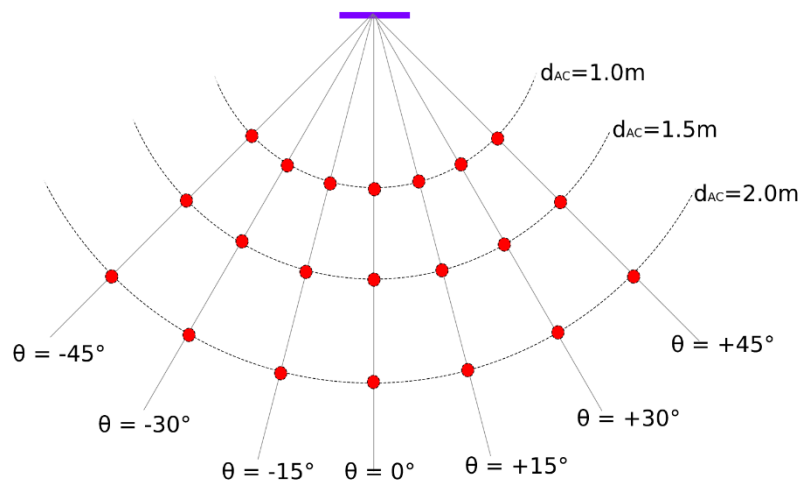


Figure 7: Top view of experiment 2 starting positions: purple line represents ARTag, red dots represent the various starting positions to test

To examine the performance measures, three experiments would be carried out. The first experiment is to dock 10 times from the ideal position of $(d_{AC}, \theta) = (1.5 \text{ meter}, 0^\circ)$. The second experiment is to test the electric contact of docking operation from the different starting points (d_{AC}, θ) as shown in Figure 7. In each position, the robot will face 4 different directions (facing the tag, facing left, facing right, facing back) when it starts. The third experiment is to test run the auto-docking under various special conditions, including far distances and dark indoor environments.

3.7 Charging station prototype

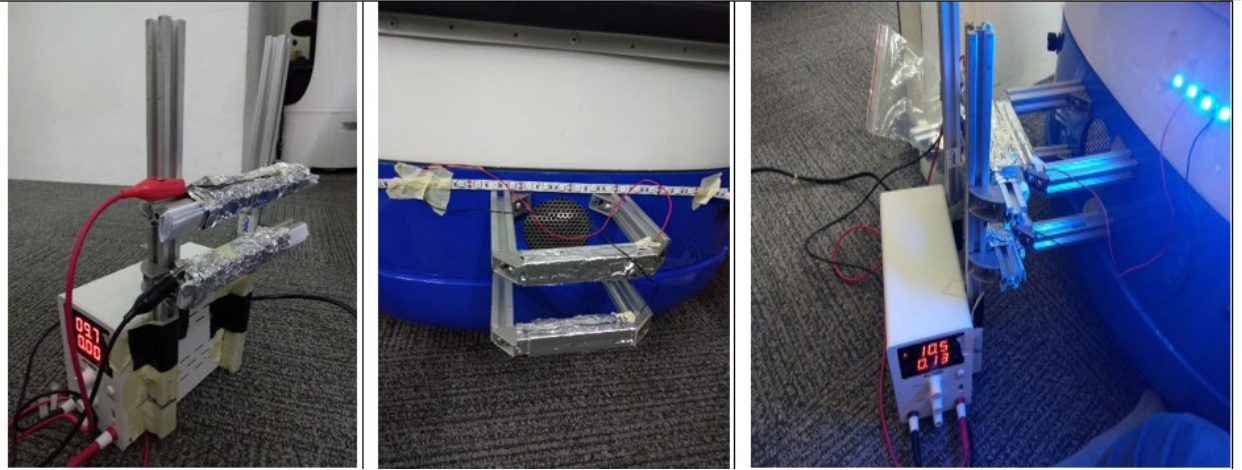


Figure 8: From left to right: DC power supply charging station, aluminium-coated bars on the SDR, electric contact between the charging station and SDR

To simulate the charging operation, a DC power supply is used as the charging station, and a LED light strip attached to the SDR is used as an indicator for electric contact. Aluminium foil is wrapped around the two bars attached to both the SDR and the charging station, simulating the anode and the cathode conducting bars.

4. PRESENTATION OF FINDINGS AND DATA

4.1 Experiment 1: Test Run 10 times at $(d_{AC}, \theta) = (1.5 \text{ meters}, 0^\circ)$:

Test Run	1	2	3	4	5	6	7	8	9	10
Electric Contact	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Time (s)	17.77	16.82	16.61	17.14	16.91	16.40	16.22	16.63	17.08	16.79
Average Time (s)	16.84									

Table 1 : Results of the 10 test runs at starting position of (1.5 meters, 0 °)

4.2 Experiment 2: Test Run from different d_{AC} distance and inclined angle , θ :

	Angle, θ (°)	-45	-30	-15	0	+15	+30	+45
Starting position at 1.0 meter away from ARTag	Facing tag	✗	✓	✓	✓	✓	✓	✗
	Facing left	✗	✓	✓	✓	✓	✓	✗
	Facing right	✗	✓	✓	✓	✓	✓	✗
	Facing back	✗	✓	✓	✓	✓	✓	✗
Starting position at 1.5 meter away from ARTag	Facing tag	✗	✓	✓	✓	✓	✓	✗
	Facing left	✗	✓	✓	✓	✓	✓	✗
	Facing right	✗	✓	✓	✓	✓	✓	✗
	Facing back	✗	✓	✓	✓	✓	✓	✗
Starting position at 2.0 meter away from ARTag	Facing tag	✗	✓	✓	✓	✓	✓	✗
	Facing left	✗	✓	✓	✓	✓	✓	✗
	Facing right	✗	✓	✓	✓	✓	✓	✗
	Facing back	✗	✓	✓	✓	✓	✓	✗

(✓: LED lights up; ✗: LED does not light up)s

Table 2: Electric contact test from different starting position

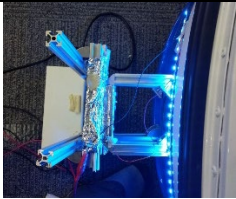

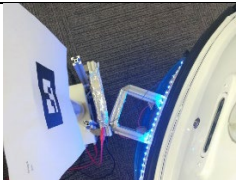
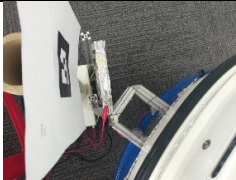
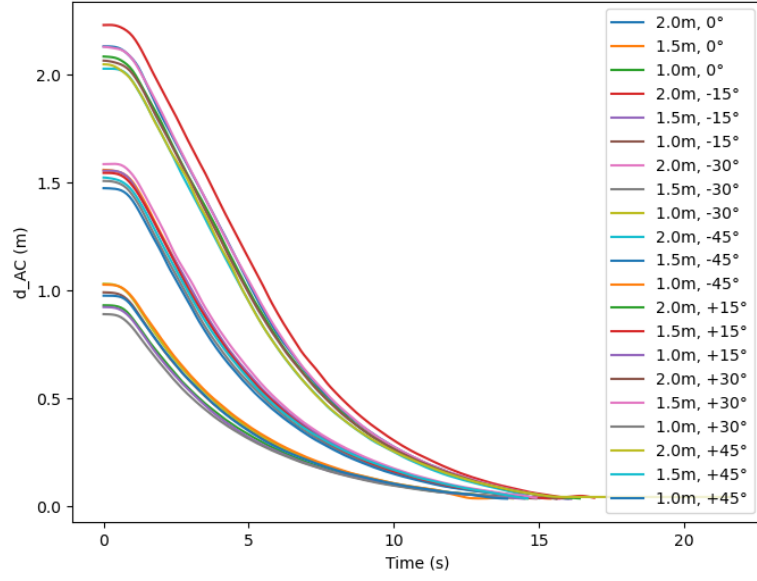
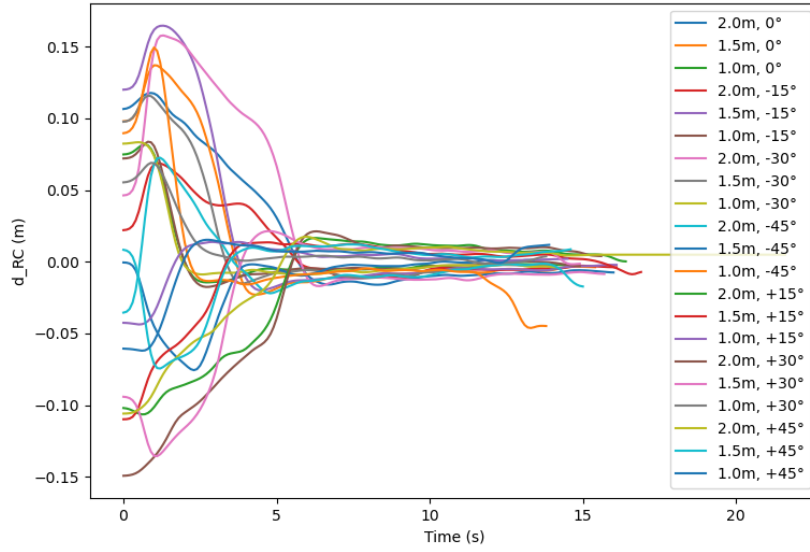
Inclined angle, θ	Image of Electric contact	Descriptions of docking
0°		The robot could maintain its direction towards the ARTag and drive towards the charging station. It would slowly come to a stop when electric contacts are made. The bars on the SDR could align nicely with the bars on the charging station and electric contact could be made, as shown in the experiment video captured (Ho, 2022, 0:07).
$\pm 15^\circ$		The robot could drive towards the charging station at an angle. It would gradually slow down and come to a stop when the bars on the SDR touch the bars on the charging station at a small angle. Electric contact could be made, as shown in the experiment video captured (Ho, 2022, 0:30).
$\pm 30^\circ$		The robot could drive towards the charging station at an angle. It would gradually slow down and stop when the bars on the SDR touch the edge of the bars on the charging station at a greater angle. Electric contact could be made, as shown in the experiment video captured (Ho, 2022, 0:54).
$\pm 45^\circ$		The robot could drive towards the charging station at a greater angle. It would still gradually slow down but the bars on the SDR would collide with the side of the bars on the charging station, thus electric contact could not be made, as shown in the experiment video captured (Ho, 2022, 1:15).

Table 3: Description of docking from different angles, θ



jn

Graph 5: d_{AC} against time



Graph 6: d_{RC} against time

4.3 Experiment 3: Special conditions

In addition, when the camera is directly facing the ARTag, the maximum distance away from the ARTag where the auto-docking is still functional is measured to be $d_{AC} = 4.9$ meters. When

running the auto-docking operation under a dark environment, the camera is not consistent with the detection of the ARTag, and the docking is unsuccessful. When the camera visual is obstructed by passers-by, the robot will deviate from the trajectory and the docking is unsuccessful.

5. DATA ANALYSIS AND DISCUSSION

From Table 1, the robot successfully contacted the charging station in all ten trial runs. This shows its consistency when the robot starts from the ideal position, which is 1.5 meters away from the ARTag. From Table 2, the robot is not able to perform electric contact with the charging station when its inclined angle is more than $\pm 30^\circ$. Below this angle threshold, the robot can dock consistently at different d_{AC} distances. The different facing direction does not affect the results as the initial spinning has mitigated it. From Graph 5, the d_{AC} gradually decreases to zero, ensuring soft contact with the charging station without sudden collisions. From Graph 6, the settling time for the d_{RC} converges to near zero is around 5 seconds. From experiment 3, the marker detection only works when distance between it and the robot is within 4.9 meters. The marker detection does not work well when there is not enough light source.

The experiment has its limitations. It does not cover all possible special scenarios that could affect the docking operations, such as outdoor conditions, the environment with flickering lights, different floor conditions, low batteries, and others. The d_{AC} and d_{RC} values are obtained from the marker detection using ARTag, thus any error in the camera visual input and marker detections might lead to inaccurate data. On top of that, the charging station used in this project has design flaws, including the lack of a damper to receive the impact from the contact with the robot, as well as a lack of tolerance for enabling contact from wider angles.

The experiments have revealed some pros and cons of this auto-docking solution. Pros include its simplicity, lightweight, docking consistency at the intended starting position, as well as the mitigation of facing directions. Cons include the limitation on the starting inclined angle due to the auto-docking operation being a straight-line path to the ARTag. Relying on visual input for

marker detection has its downside. The operation only works if there is no more than one ARTag with the specific id, complex background patterns showing resemblances to the tag pattern could also cause disturbances to the docking operation. A low environmental intensity and obstruction of the camera by passer-by may also cause failure.

6. CONCLUSION

In this paper, an auto-docking solution has been proposed using ARTag fiducial marker to estimate relative position and PI-controller to adjust velocity. This method is effective and stable provided that the robot's starting position is within $\pm 30^\circ$ of the inclined angle to the ARTag and around 1.5 meters from the ARTag. External disturbances would cause failure to the operation, including obstruction of camera visual input, the presence of extra ARTag as disturbance as well as low light intensity.

Recommended future works include full integration of the auto-docking scripts into the current SDR navigation scripts to achieve full autonomy, designing a robust charging station capable of withstanding impact and tolerance, establishing the electric connection between the SDR battery and the aluminium bars, and designing an algorithm allowing the auto-repositioning of the robot if it is out of range of the allowable starting position for auto-docking.

REFERENCES

Cassinis, Riccardo & Tampalini, Fabio & Bartolini, Paolo & Fedrigotti, Roberto. (2005).

Docking and charging system for autonomous mobile robots.

D. Cohen, D. Ozick, C. Vu, J. Lynch, and P. Mass, "Autonomous robot auto-docking and energy management systems and methods," Feb. 19 2008. US Patent 7,332,890

Fiala, M. (2015). ARTag, a Fiducial Marker System Using Digital Techniques. *2005 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05).

<https://doi.org/10.1109/cvpr.2005.74>

Foote, T. (Ed.). (2016, March 28). *tf/Overview/Transformations - ROS wiki*. ROS.org.

<http://wiki.ros.org/tf/Overview/Transformations>

Ho, K. (2022, November 17). *SDR Auto docking Test* [Video]. YouTube.

<https://www.youtube.com/watch?v=4Hauzwl-VmU&feature=youtu.be>

Kartoun, U., Stern, H., Edan, Y., Feied, C., Handler, J., Smith, M., & Gillam, M. (2006).

Vision-Based Autonomous Robot Self-Docking and Recharging. *2006 World*

Automation Congress. <https://doi.org/10.1109/wac.2006.375987>

Saito, I. (Ed.). (2016, July 19). *ar_track_alvar - ROS Wiki*. ROS.org.

http://wiki.ros.org/ar_track_alvar

Song, G., Wang, H., Zhang, J., & Meng, T. (2011). Automatic docking system for recharging

home surveillance robots. *IEEE Transactions on Consumer Electronics*, 57(2), 428–435.

<https://doi.org/10.1109/tce.2011.5955176>

- Yi-Cheng Wu, Ming-Chang Teng, & Yi-Jeng Tsai. (2009). Robot docking station for automatic battery exchanging and charging. *2008 IEEE International Conference on Robotics and Biomimetics*. <https://doi.org/10.1109/robio.2009.4913144>
- Zhang, X., Li, X., & Zhang, X. (2021). Automatic Docking and Charging of Mobile Robot Based on Laser Measurement. *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*.
<https://doi.org/10.1109/iaeac50856.2021.9390995>

APPENDICES

Appendix A: Auto docking script “rotate_align_entry”

```
#!/usr/bin/env python
import rospy
import tf
import geometry_msgs.msg
from std_msgs.msg import Int32
from std_msgs.msg import Float64, String
from tf.transformations import quaternion_from_euler, euler_from_quaternion
import time
import math

#cmd_vel unit: m/s, rad/s

if __name__ == '__main__':
    rospy.init_node('rotate_and_dock')

    #Initialising variables
    listener = tf.TransformListener()
    rate = rospy.Rate(10.0)
    cmd_pub = rospy.Publisher('/sdr/navigation/cmd_vel', geometry_msgs.msg.Twist,
queue_size=1)
    cmd = geometry_msgs.msg.Twist()
    x_pub = rospy.Publisher('/x_value', Float64, queue_size=10)
    y_pub = rospy.Publisher('/y_value', Float64, queue_size=10)
    status_pub = rospy.Publisher('/status', String, queue_size=10)
    operation = 'rotate'
    linear_vel = 0.0
    prev_linear_vel = 0.0
    angular_vel = 0.0
    prev_angular_vel = 0.0
    prev_x_dist=0.0
    prev_y_dist = 0.0
    x_error = 0.0
    prev_x_error = 0.0
    x_error_integral = 0.0
    y_error = 0.0
```

```

prev_y_error = 0.0
y_error_integral = 0.0
itr_entry=0
itr_align=0

while not rospy.is_shutdown():

    if(operation == "rotate"):
        trans = []
        print("operation:" + operation)
        try:
            (trans,rot) =
listener.lookupTransform('/cameraLower_color_optical_frame',
'/ar_marker_8',rospy.Time(0))
        except (tf.LookupException, tf.ConnectivityException,
tf.ExtrapolationException):
            print("artag_not_detected")
            trans = [1000000000]

        print("trans[0]:" + str(trans[0]))

        if(trans[0]<0):
            cmd.angular.z = 0.2
        elif(trans[0]>0):
            cmd.angular.z = -0.2

        if(-0.15<trans[0] and trans[0]<0.15):
            cmd.angular.z = 0
            cmd_pub.publish(cmd)
            for i in range(2): #sleep for 0.2 seconds
                rate.sleep()
            operation = 'entry'
            cmd_pub.publish(cmd)

    elif(operation == "entry"):
        print("operation:" + operation)
        itr_entry+=1
        try:
            (trans,rot) =
listener.lookupTransform('/cameraLower_color_optical_frame',

```

```

'/ar_marker_8',rospy.Time(0))
    except (tf.LookupException, tf.ConnectivityException,
tf.ExtrapolationException):
        continue

    x_dist = -(trans[2] - 0.20)
    y_dist = -(trans[0])

    print("                                " + "\n" + "\n")
    print("iteration: " + str(itr_entry))
    print("x-distance:" + str(trans[2] - 0.20))
    print("y-distance:" + str(trans[0]))

    ##PI initialisation:
    K_P_x = 0.2
    K_P_y = 1.0

    K_I_x = 0.001
    K_I_y = 0.01

    x_error = x_dist - 0
    y_error = y_dist - 0

    print("x_error:" + str(x_error))
    print("y_error:" + str(y_error))

    x_error_integral += x_error*0.1
    y_error_integral += y_error*0.1

    print("x_error_integral: " + str(x_error_integral))
    print("y_error_integral: " + str(y_error_integral))

    #PI-algo
    linear_vel = K_P_x*x_error + K_I_x*x_error_integral
    angular_vel = K_P_y*y_error + K_I_y*y_error_integral

    cmd.linear.x = linear_vel
    cmd.angular.z = angular_vel
    cmd_pub.publish(cmd)

```



```

print("cmd: " + str(cmd))

prev_x_dist = x_dist
prev_y_dist = y_dist
prev_x_error = x_error
prev_y_error = y_error

#for collecting data
x = Float64()
y = Float64()
x = x_dist
y = y_dist
x_pub.publish(x)
y_pub.publish(y)

#end of docking
if(abs(x_error)<0.035):
    print("auto-docking operation complete")
    break

rate.sleep()

```

Appendix B: ROS rqt_graph of Gazebo simulation

