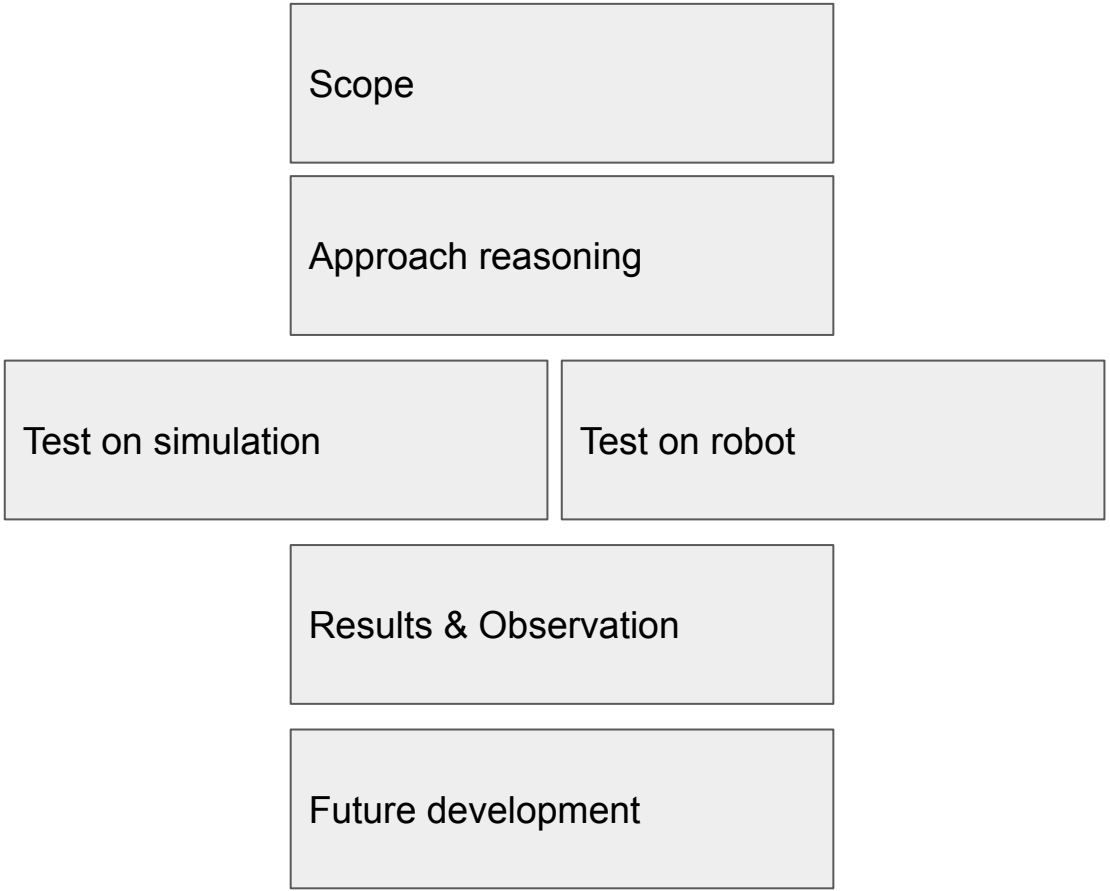
Auto-docking of SDR Solustar

Ho Kae Boon

Content


Scope
Approach reasoning
Test on simulation
Test on robot
Results & Observation
Future development

Scope

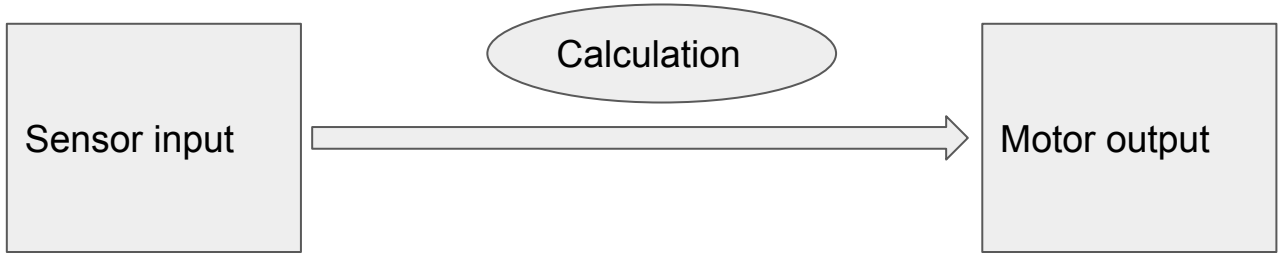
- Explain the task of autodock.py - ✓
- Focus today: PID_entry (state 7) - run independent

During this state

1. Robot is 1.2 meter from the tag
2. Robot is directly facing the tag

Approach

TF position
PI-control



Front realsense depth camera
+ ARTag

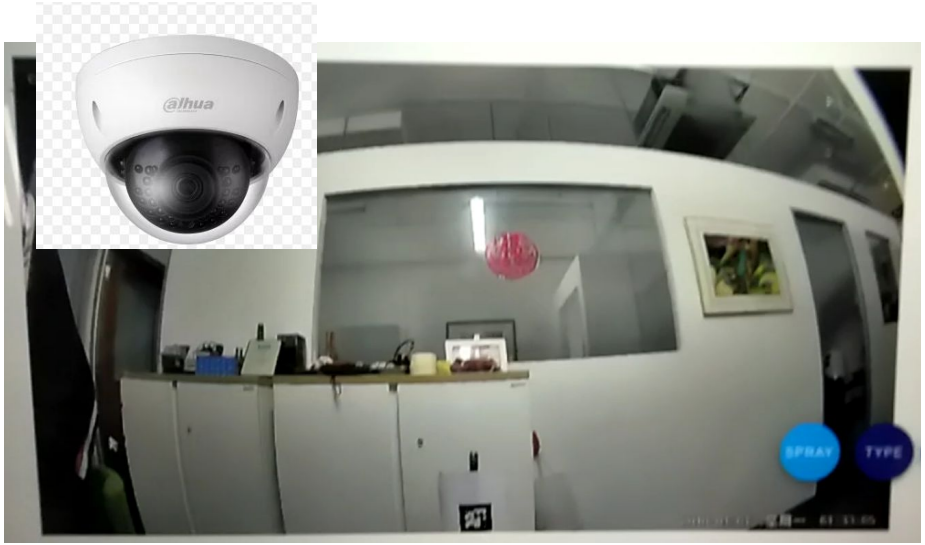
Adjust wheel linear & angular velocity

Sensor input - why visual input

- Cheap
- Easy to set up
- Accurate
- Widely used - open source tool

Sensor input - why front realsense camera

- Clear image
- Not distorted



Ip cam

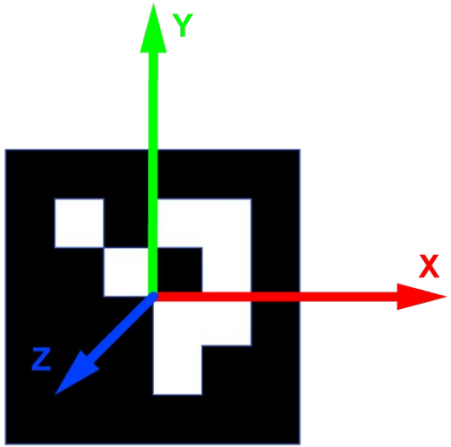
Realsense cam

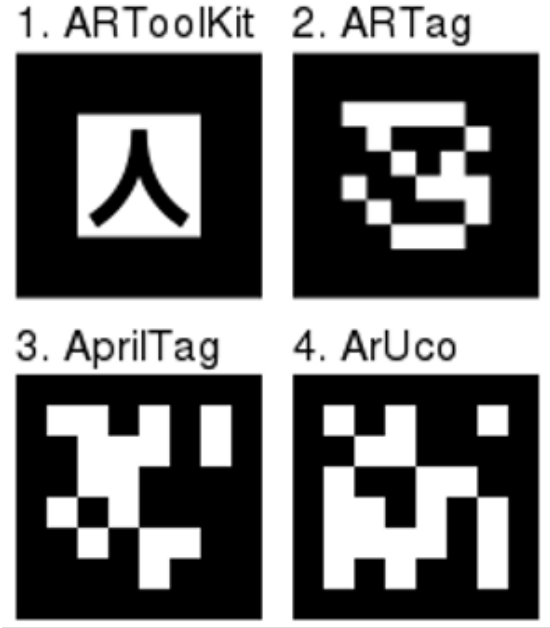
Sensor input - Why ARTag

- Widely used & Popular
- Easy to set up
- Lightweight
- Fairly Accurate

We use 9cm

With id of 255





Sensor input - Approach

5.2 API for individual markers

5.2.1 Published Topics

`visualization_marker` (visualization_msgs/Marker)

This is an rviz message that when subscribed to (as a Marker in rviz), will display a colored square block at the location of each identified AR tag, and will also overlay these blocks in a camera image. Currently, it is set to display a unique color for markers 0-5 and a uniform color for all others.

`ar_pose_marker` (ar_track_alvar/AlvarMarkers)

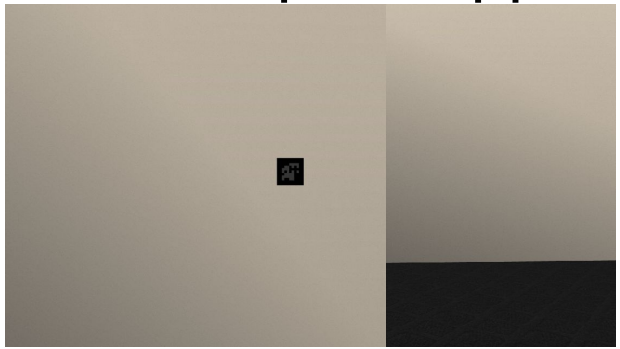
This is a list of the poses of all the observed AR tags, with respect to the output frame

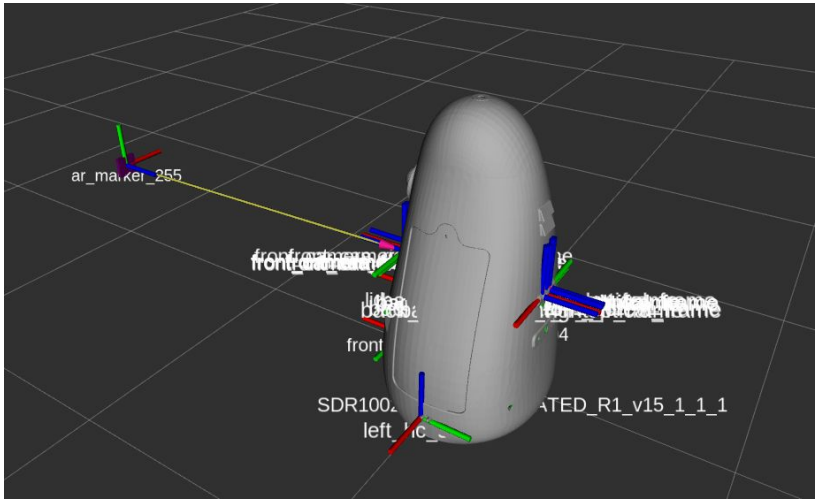
5.2.2 Provided tf Transforms

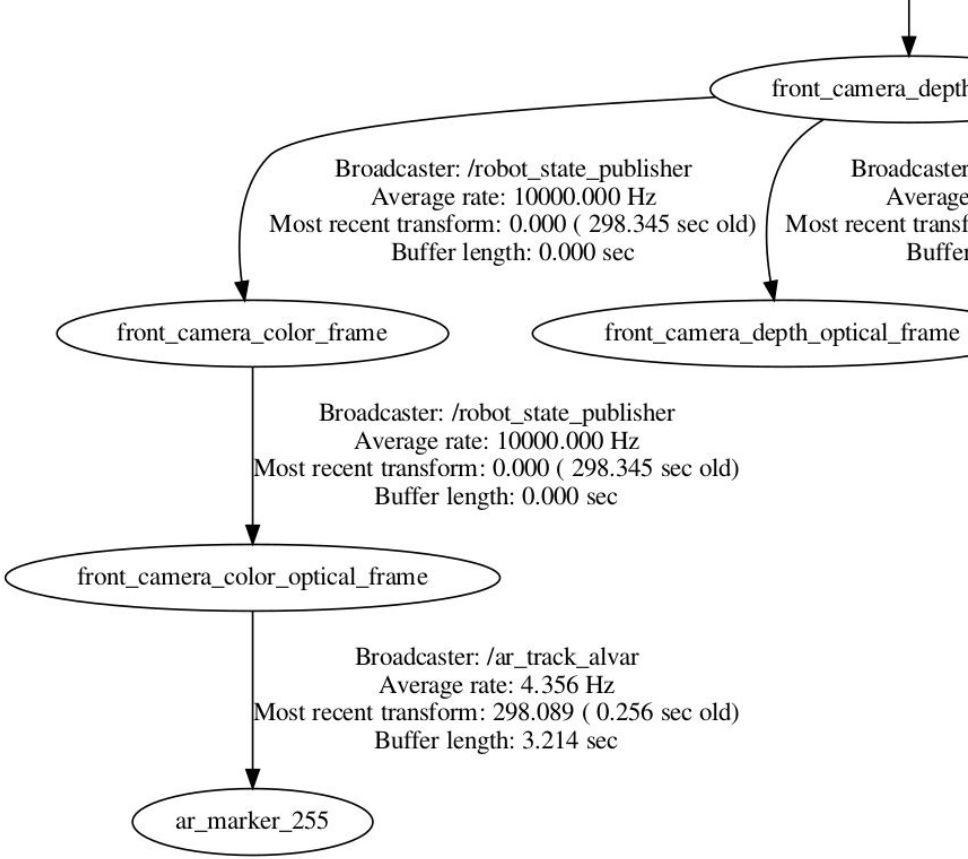
`Camera frame (from Camera info topic param) → AR tag frame`

Provides a transform from the camera frame to each AR tag frame, named ar_marker_x, where x is the ID number of the tag.

Sensor input - Approach







(from simulation)

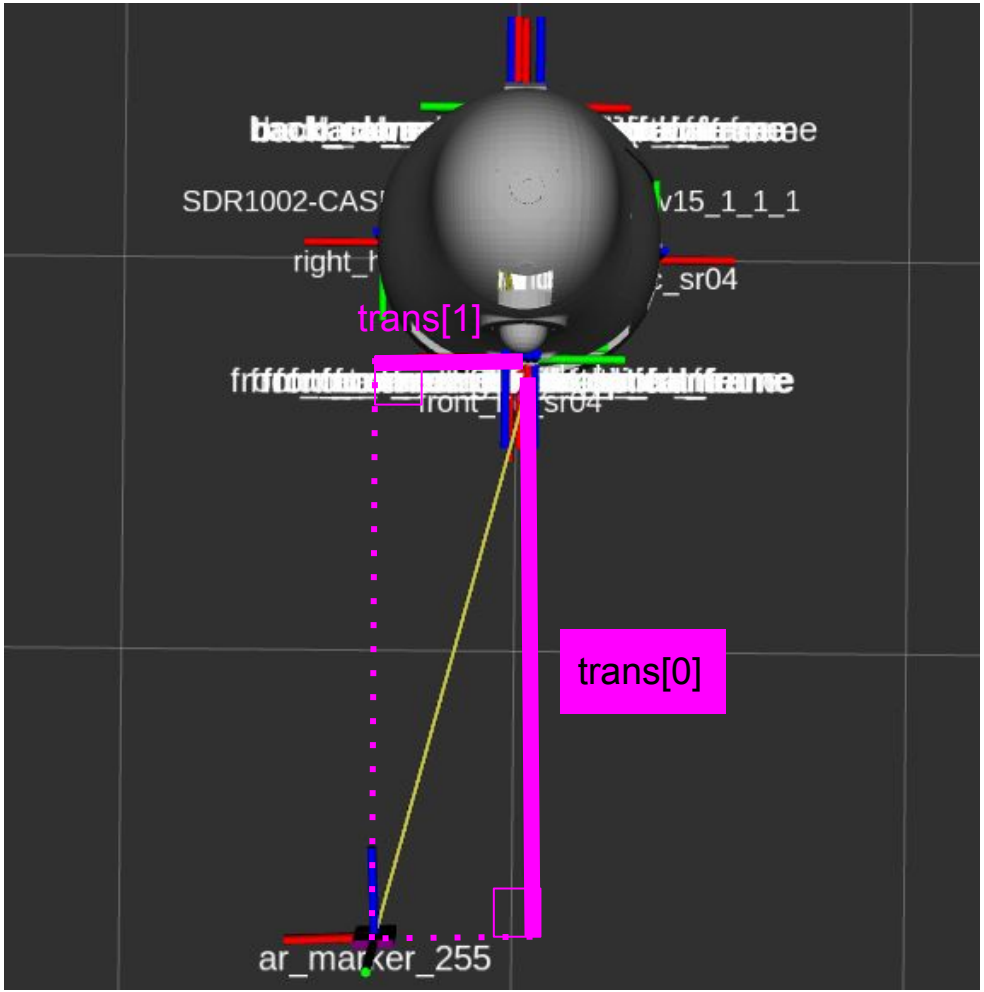
Calculation

Topic: subscribe to the tf topic between tf

```
try:
    (trans,rot) = listener.lookupTransform('/front_camera_color_frame', '/ar_marker_255',rospy.Time(0))
    #if use ('/front_camera_color_frame', '/ar_marker_255', ..), then the linear is same as published in the ar_pose_marker, which we want
    #if use ('/ar_marker_255', '/front_camera_color_frame', ..) then the angular is same as published in the ar_pose_marker
except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
    continue
```

(trans, rot) will give the transformation from front_camera_color_frame to ar_marker_255

trans : x, y, z
rot: quaternion values



trans[0] = x-dist
trans[1] = y-dist

Motor output

cmd_vel topic

-> linear

x(front-back), y(left-right), z(up-down)

-> angular

x(roll), y(pitch), z(yaw)

Calculation

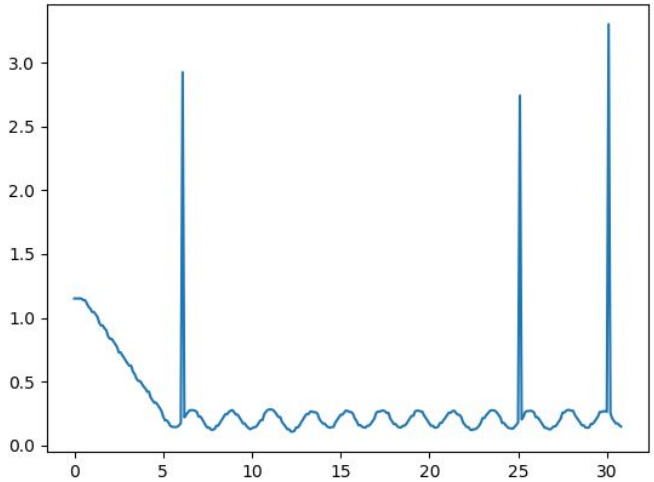
trans[0] = x-dist ——---> to adjust linear x
trans[1] = y-dist ——---> to adjust angular z

Using proportional control

Larger x-dist, then larger linear x velocity
Larger y-dist, then larger angular z velocity

P-control results

x-dist



Time

K_p = 10

Observation:
Robot will never dock

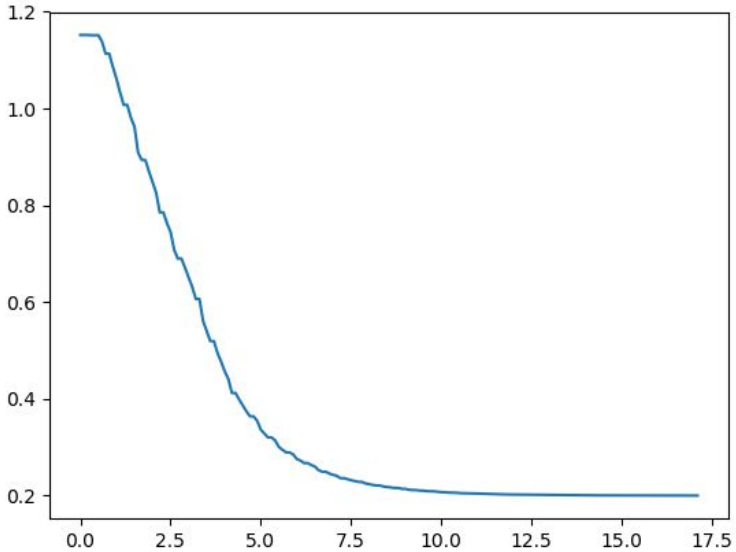
Problem:
Keep oscillating
(the peak are error in data-collection)

To improve:
Tuning is required

Note: Testing is done on simulation

P-control results - after some tuning

x-dist



time

Kp = 1

Observation:
As x-value decrease,
Linear x velocity decrease

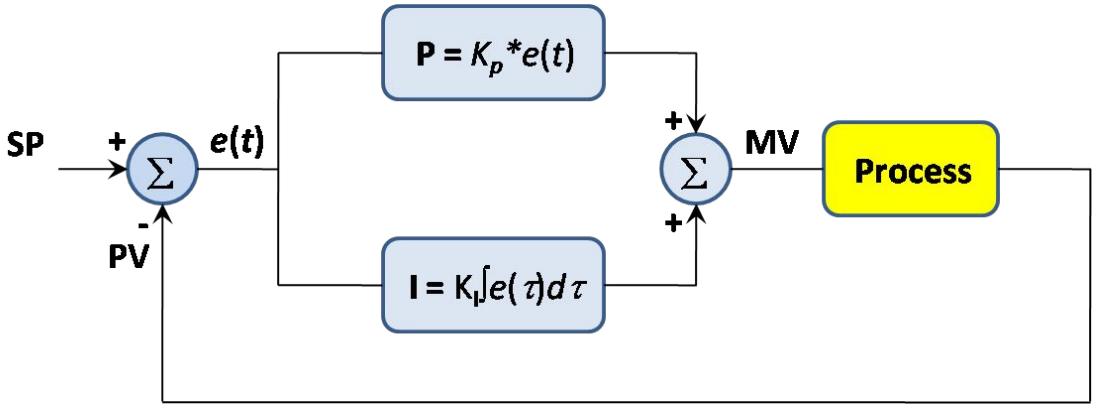
Problem:
Takes longer time to dock

To improve:
Use PI-control

Note: Testing is done on simulation

PI-control

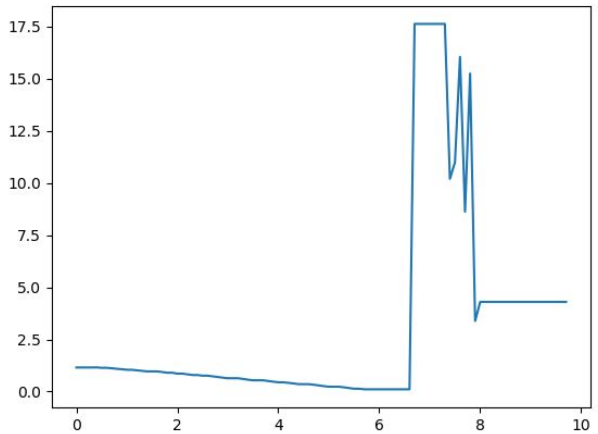
- Addition of integral term - proportional to accumulation of errors



When the robot moves slower as it got closer to the tag, the I-term will still provides velocity as the accumulation of error is still there, prompting the robot to continue going.

PI-control results

x-dist



time

K_p = 1, K_i = 10

Observation:
Robot hit the artag too fast

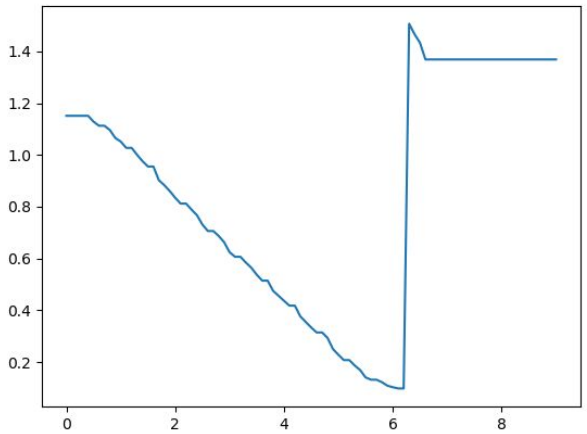
Problem:
Robot still goes too fast when nearing the tag.
Overshoot occurs

To improve:
tuning

Note: Testing is done on simulation

PI-control results

x-dist



time

K_p = 1, K_i = 0.005

Observation:
Robot slows down when nearing artag, but still hit the artag

Problem:
Robot still goes too fast when nearing the tag.
Overshoot occurs

To improve:
Add in integral wind-up

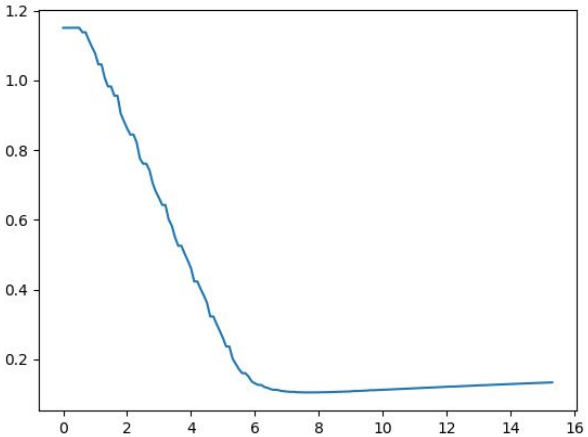
Note: Testing is done on simulation

Integral wind up

To restrict the integral term within certain value, so that the velocity will not overshoot

PI-control with wind-up results

x-dist



time

K_p = 1, K_i = 0.005, wind-up = 0.0025

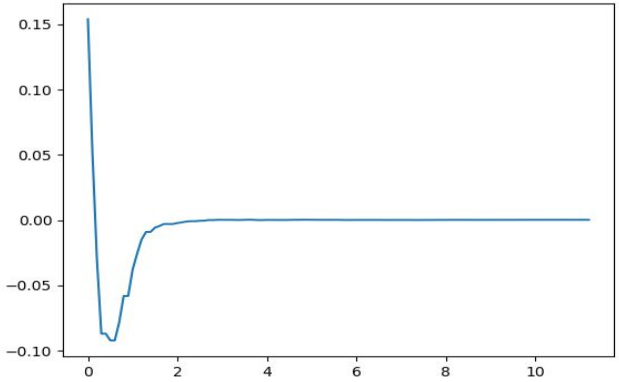
Observation:
Robot able to reach artag quickly, success

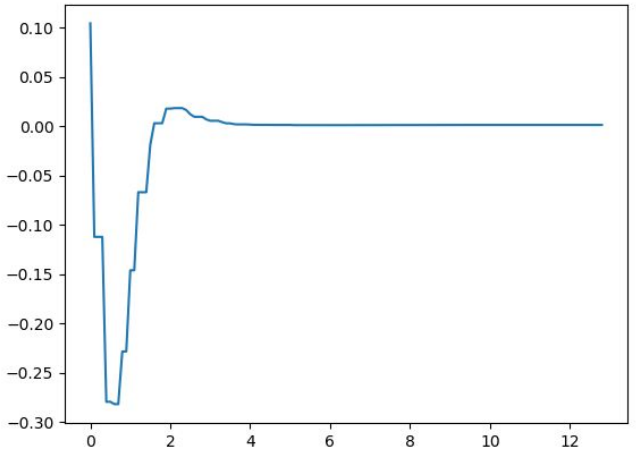
Small issue:
It will slowly backs up from artag after reaching 0.2meter

To improve:
Use battery charging status to end the operation.

Note: Testing is done on simulation

Same approach is used for y-distance





The initial drop: Robot rotating
Then it maintain at 0 for a long time.

Code review

```
#for docking--------------
linear_vel = 0.0
prev_linear_vel = 0.0
angular_vel = 0.0
prev_angular_vel = 0.0
x_dist = 0.0
prev_x_dist=0.0
y_dist = 0.0
prev_y_dist = 0.0

x_error_accu = 0.0
x_error = 0.0
y_error_accu = 0.0
y_error = 0.0

x_pub = rospy.Publisher('/x_value', Float64, queue_size=10)
y_pub = rospy.Publisher('/y_value', Float64, queue_size=10)
```

```
##PI initialisation:
    P_x = 1
    P_y = 1
    I_x = 0.005
    I_y = 0.005

    x_error = x_dist - prev_x_dist
    y_error = y_dist - prev_y_dist
```

1) initialisation

```
x_dist = trans[0] - 0.25
y_dist = trans[1]
```

2) So that robot will dock 0.25 meter infront of the tag

Code review

```
##PI control
    x_error_accu += (x_error*0.1)
    y_error_accu += (y_error*0.1)

    linear_vel = P_x*x_error + min(max(I_x*x_error_accu,-0.0025),0.0025) + prev_linear_vel
    angular_vel = P_y*y_error + min(max(I_y*y_error_accu,-0.0025),0.0025) + prev_angular_vel
```

3) PI control and wind up

Code review

```
################## for moving the robot based on the linea_vel or the angular_vel
    cmd = geometry_msgs.msg.Twist()
    cmd.linear.x = linear_vel
    cmd.angular.z = angular_vel
    cmd_pub.publish(cmd)

    prev_x_dist = x_dist
    prev_y_dist = y_dist
    prev_linear_vel = linear_vel
    prev_angular_vel = angular_vel

################################(below code is for my own documentation purpose, can remove)

    x = Float64()
    y = Float64()
    x = x_dist + 0.2
    y = y_dist
    x_pub.publish(x)
    y_pub.publish(y)
```

4) publish cmd_vel & publish x, y distance for data collection

Code review

1) Rotate operation

-0.3 < y-value < 0.3

2) /status

“Rotating”

“Docking”

How to tune the PI

Variable to tune:

P_x

P_y

I_x

I_y

Windup value (in the lines below)

```
            linear_vel = P_x*x_error + min(max(I_x*x_error_accu,-0.0025),0.0025) + prev_linear_vel
            angular_vel = P_y*y_error + min(max(I_y*y_error_accu,-0.0025),0.0025) + prev_angular_vel
```

Test on simulation - prerequisite

ROS package: realsense SDK 2.0, realsense ros package, realsense gazebo plugin package, artag_alvar package, and my files

Gazebo: add in willowgarage_with_artag (my custom made world file) inside .gazebo/worlds, add in artag123123 (my custom made model file) inside .gazebo/model

Launch file (inside SDR_simulation):
movingparts.launch: will launch the gazebo with artag
pr2_indiv_no_kinect_copy: will launch the artag detection
rotate_and_dock: will launch the docking process

Test on simulation (how to edit the gazebo environment)

To add artag into gazebo
-Generate artag image with rosrun ar_track_alvar createMarker

-Use blender to create sdf model with the texture of the generated artag (problem with solidworks: no texture)

-add the model with the image file into .gazebo/model

-include the model into the world file (if you create a new world, then need edit your launch file too_!)

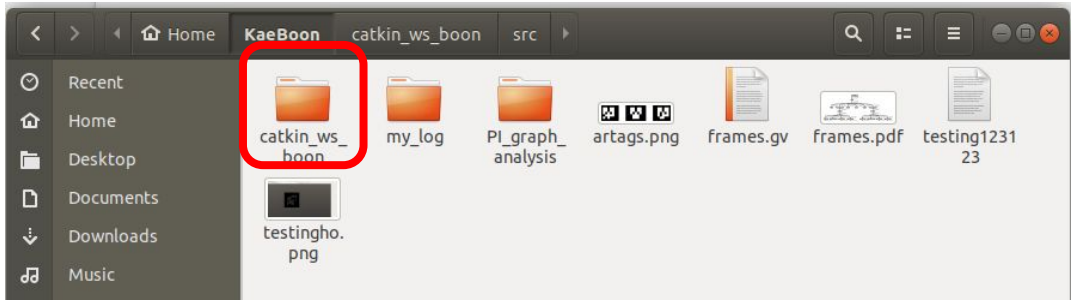
```
<include>
  <uri>model://artag123123</uri>
  <name>artag_01</name>
  <pose>1.747017 0 0.612284 -1.570796 0 1.570796</pose>
</include>
```

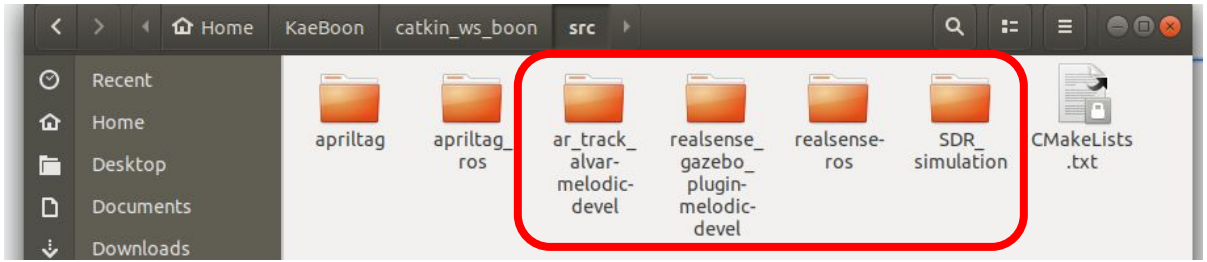
Test on simulation - my current file system

```
sdr-01@sdr01-NUC8i7BEH:~/KaeBoon/catkin_ws_boon/src/SDR_simulation/launch$
```

SDR-01 NUC

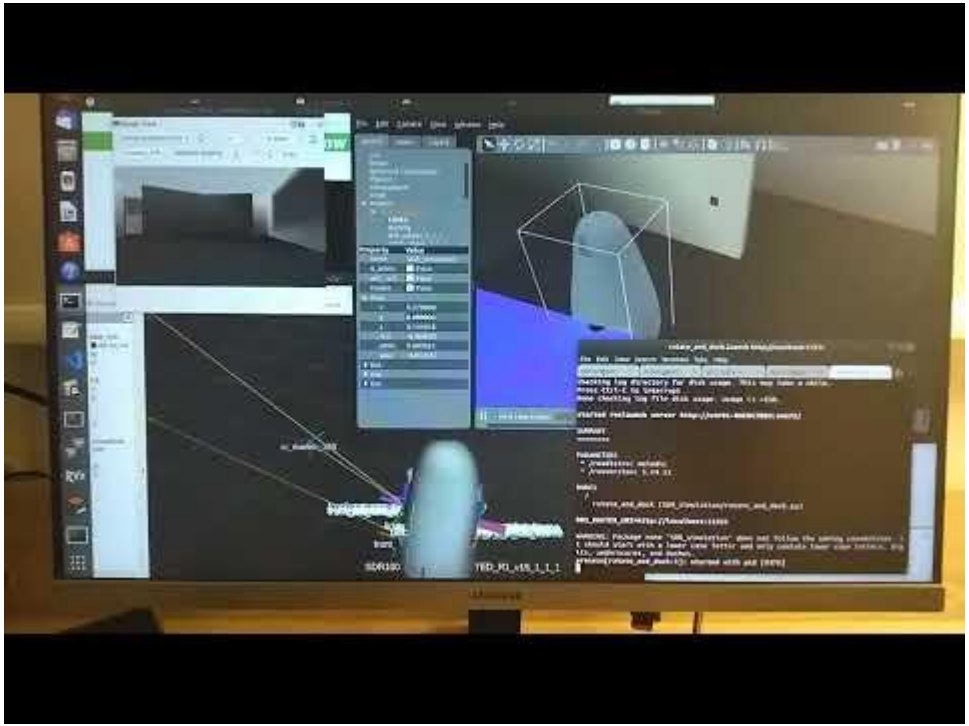
FYI





Testing on Gazebo results

See video



Go t=47s

Conclusion from Gazebo testing

Works perfectly, as long as

1) Starts at around 1.2 meter from artag and directly facing it.

Testing on robot

Problems and solutions

Problem 1 - unable to locate node

Unable to run the program

Solution:

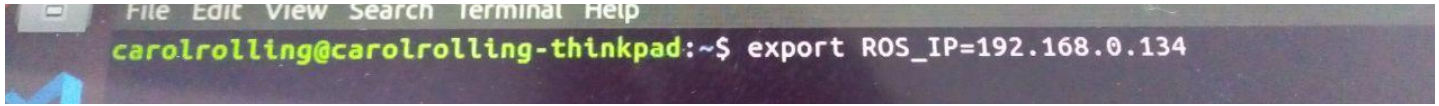
- Need to install the files in carol’s laptop instead.
- Need to catkin_build and source devel.
- Install ar_tag alvar directly from open source instead.

Problem 2 - topics not published correctly

The SDR does not seem to receive the topics that I published

Solutions:

Need to run export ros ip first before launching the files inside carol computer



Problem 3 - artag tf-related warning keep appearing

Not able to detect artag, keep receiving warning

Solution:

Change the tf frame name of the front_color_camera_frame inside both the pr2_no_kinect_copy launch file as well as the rotate_and_dock.py file

Problem 4 - the rviz marker pose is still weird

It does not accurately show the position of artag

Reason: realsense camera is inversed

Solutions:

Change the y-dist to negative in the code. Mask tape the realsense properly.

X-dist is not affected so its ok.

Tune the distance by adjusting image size parameter and error tolerance in the file pr2_no_kinect_copy.launch

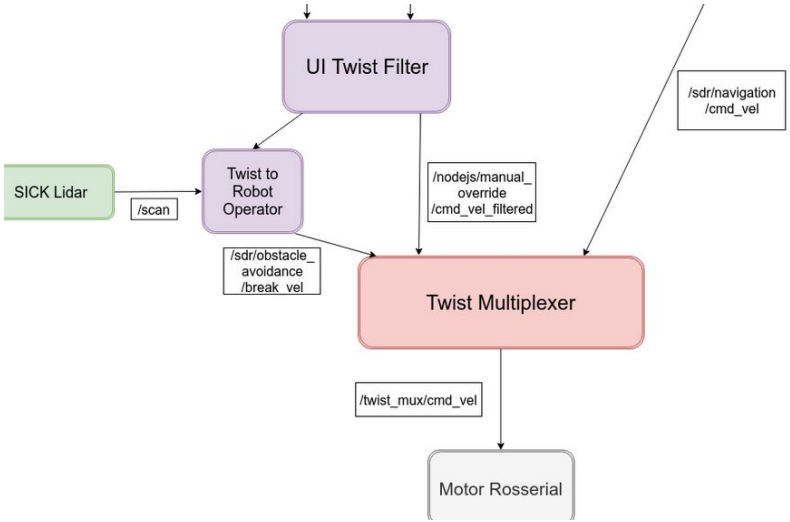
Problem 5 - the cmd_vel published seem not working

Due to usage of multiplexer

Temporary Solution***:

Change configuration

sdr_control/config/twist_mux_topics.yaml



** this might affect normal operation from manual control

```yaml
topics:
-
  name    : keyboard
  topic   : key_vel
  timeout : 0.5
  priority: 90
-
  name    : obstacle_avoidance
  topic   : /sdr/obstacle_avoidance/break_vel
  timeout : 0.5
  priority: 100
-
  name    : autonomous_waypointing
  topic   : /sdr/navigation/cmd_vel
  timeout : 0.5
  priority: 110
-
  name    : spot_turn_spray
  topic   : /sdr/spot_turn_spray/cmd_vel
  timeout : 0.5
  priority: 120
-
  name    : manual_override
  topic   : /nodejs/manual_override/cmd_vel_filtered
  timeout : 0.5
  priority: 130
```

Results & Observation

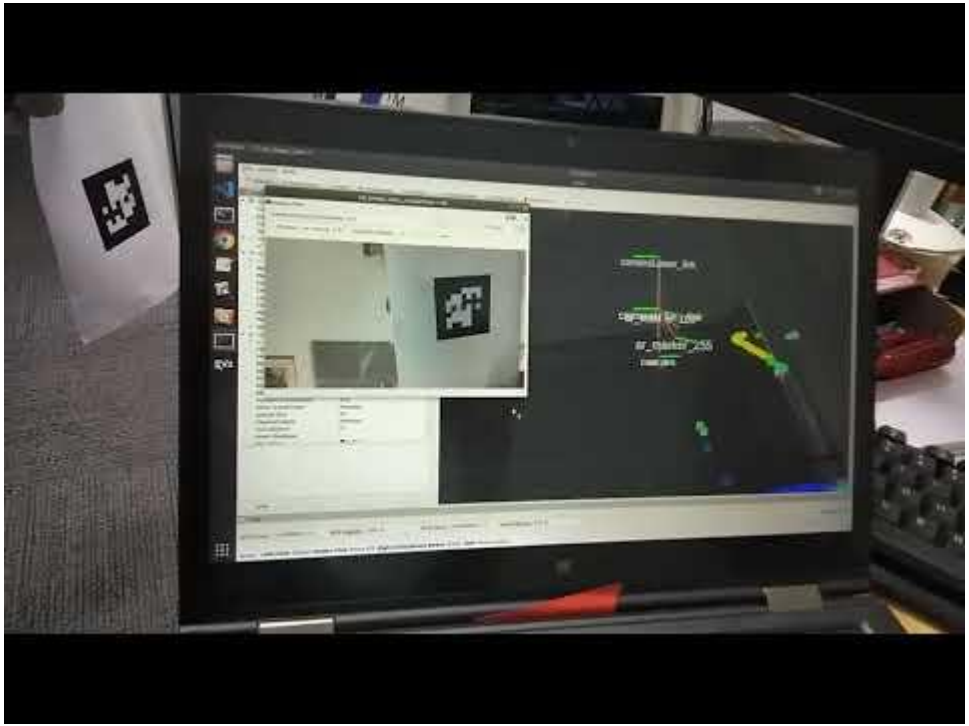
Program able to run, able to read and publish all topics successfully

When y-dist is already in range, robot goes into docking. Docking not accurate

When y-dist not in range, robot keep on rotating without going into docking phase.

Main issue 1 - Image published is too laggy

Show video



Main issue 2 - robot is slow to respond to cmd_vel

Show video

Go t = 1m11s

Conclusion

The Autodocking algo can work accurately in Gazebo simulation.

It works with low accuracy on the actual robot.

Suggestion & handover

Ensure stablility of SDR

Integrate my research findings into the autodock.py state machine

Try other method of docking (extra sensors?)

Work on the charging station and battery management system

All documents will be uploaded to the github by this week.


The End