SISTEM ROBOT OTONOM

Section 8:

FIS-based obstacle avoidance

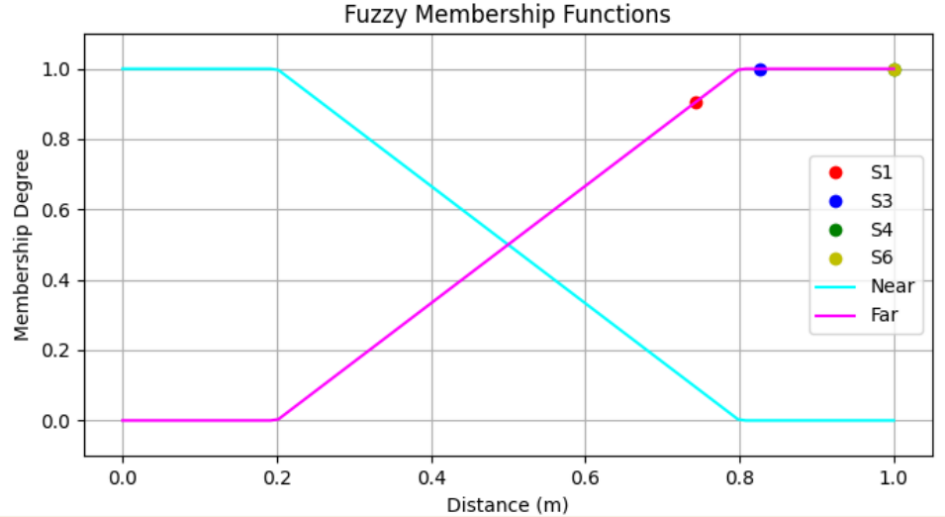
Djoko Purwanto; M. Q. Zaman

djoko@its.ac.id; muhammad.zaman@its.ac.id

MULTIPLE INPUT MULTIPLE OUTPUT

MULTIPLE INPUT MULTIPLE OUTPUT

Assignment 7 showcase: 5022221099

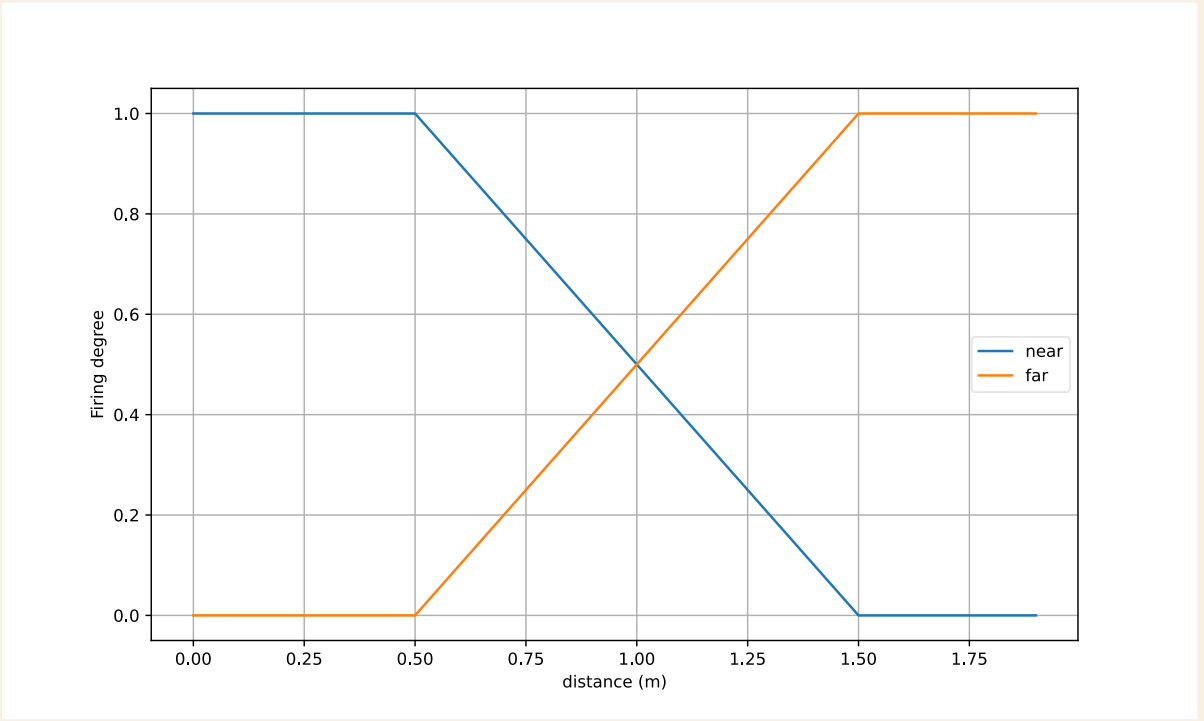


4 inputs, 2 outputs

Left_L Sensor (S1)	Left Sensor (S3)	Right Sensor (S4)	Right_L Sensor (S6)	Output vL	Output vR	Keterangan
Near	Near	Near	Near	-3	4	Semua sisi dekat, robot berbelok tajam ke kanan
Near	Near	Far	Far	4	-4	Kiri dekat, kanan jauh, robot berbelok tajam ke kiri
Near	Far	Far	Far	4	-3	Hanya sisi paling kiri dekat, robot berbelok ke kiri
Far	Far	Near	Near	-3	4	Hanya sisi kanan dekat, robot berbelok ke kanan
Far	Far	Far	Near	-3	4	Hanya sudut kanan dekat, robot berbelok ke kanan
Far	Near	Near	Far	3	3	Kiri tengah dekat dan kanan tengah dekat, robot maju perlahan
Far	Near	Near	Near	1	3	Semua sisi kecuali sudut kanan dekat, robot cenderung ke kanan
Far	Far	Near	Far	-4	3	Sisi kanan tengah dekat, robot sedikit berbelok ke kanan
Far	Far	Far	Far	5	5	Semua sensor jauh, robot bergerak lurus cepat

2 INPUTS

Input membership function for left sensor ([2]) and right sensor ([5])



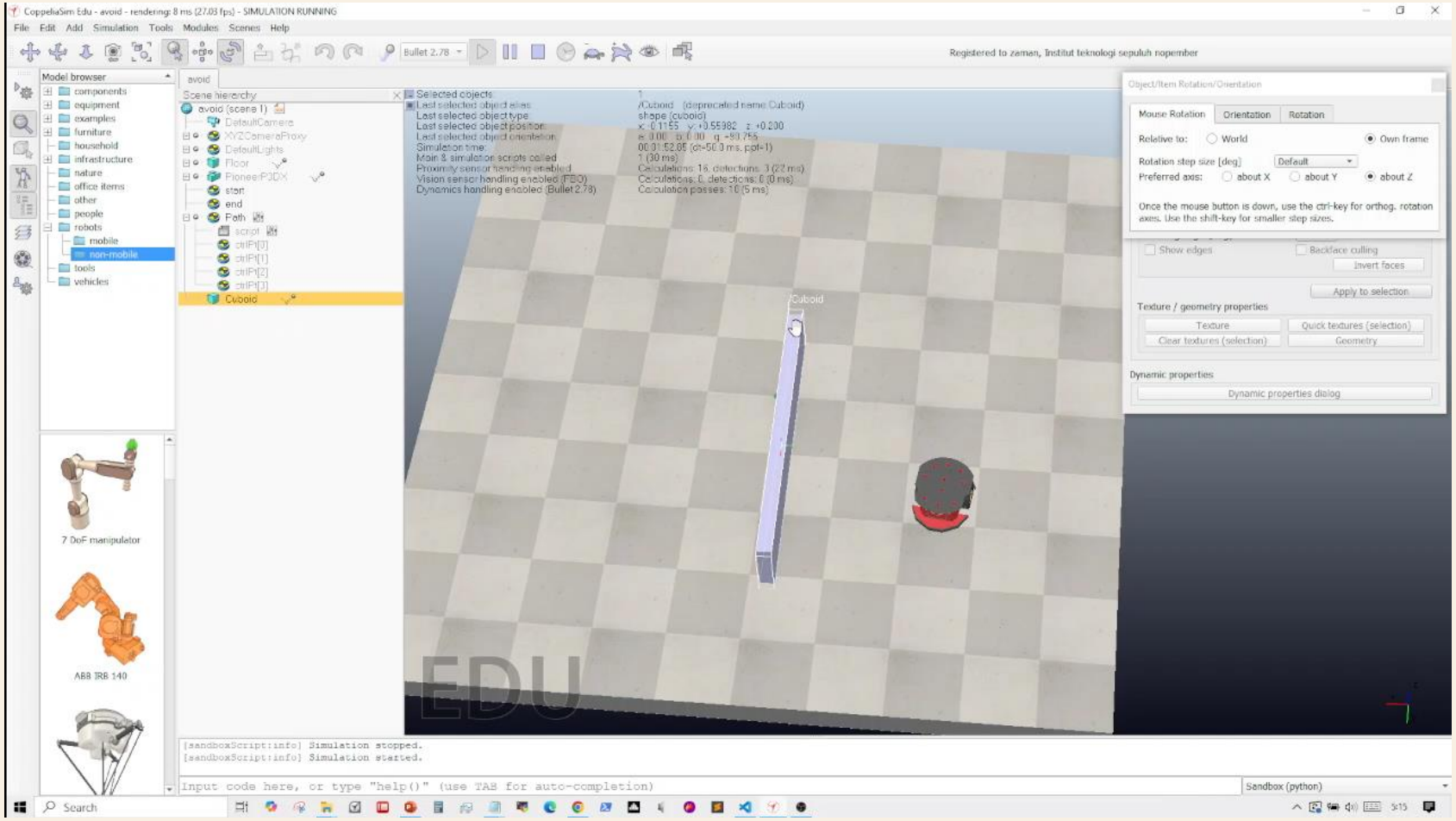
2 OUTPUTS

Input membership function for left sensor ([2]) and right sensor ([5])

Left sensor	Right sensor	Angular velocity	Linear velocity
Near	Near	0	-0.5
Near	Far	-1	0
Far	Near	1	0
Far	Far	0	0

How many singleton fuzzy outputs?

VIDEO


CoppeliaSim Edu - avoid - rendering: 8 ms (27.03 fps) - SIMULATION RUNNING
File Edit Add Simulation Tools Modules Scenes Help
Bullet 2.78
Registered to zaman, Institut teknologi sepuluh nopember
Model browser
components
equipment
examples
furniture
household
infrastructure
nature
office items
other
people
robots
mobile
non-mobile
tools
vehicles
7 DoF manipulator
ABB IRB 140
avoid
Scene hierarchy
avoid (scene 1)
DefaultCamera
XYZCameraProxy
DefaultLights
Floor
PioneerP3DX
start
end
Path
script
ctrlPt[0]
ctrlPt[1]
ctrlPt[2]
ctrlPt[3]
Cuboid
Selected objects
Last selected object alias
Last selected object type
Last selected object position
Last selected object orientation
Simulation time
Main & simulation scripts called
Proximity sensor handling enabled
Vision sensor handling enabled (FBO)
Dynamics handling enabled (Bullet 2.78)
/Cuboid (deprecated name Cuboid)
shape (cuboid)
Calculation passes: 10 (5 ms)
Cuboid
Object/Item Rotation/Orientation
Mouse Rotation
Orientation
Rotation
Relative to:
World
Own frame
Rotation step size [deg]
Default
Preferred axis:
about X
about Y
about Z
Once the mouse button is down, use the ctrl-key for ortho. rotation axes. Use the shift-key for smaller step sizes.
Show edges
Backface culling
Invert faces
Apply to selection
Texture / geometry properties
Texture
Quick textures (selection)
Clear textures (selection)
Geometry
Dynamic properties
Dynamic properties dialog
EDU
[sandboxScript:info] Simulation stopped.
[sandboxScript:info] Simulation started.
Input code here, or type "help()" (use TAB for auto-completion)
Sandbox (python)
Search

IMPLEMENTATION

```
class FIS():
    def __init__(self):
        print("FIS class Started")

    def triangleMF(self, x, a, b, c, FD0, FD2):
        if x < a:
            FD = FD0
        elif x >= a and x < b:
            FD = (x-a)/(b-a)
        elif x >= b and x < c:
            FD = (c-x)/(c-b)
        elif x >= c:
            FD = FD2
        return FD

    # input membership fuzzification
    def dis_MF(self, x):
        near = self.triangleMF(x, 0.5, 0.5, 1.5, 1, 0)
        far = self.triangleMF(x, 0.5, 1.5, 1.5, 0, 1)
        y = np.array([[near],
                      [far]],
                      dtype=float)
        return y
```

```
    def calc(self, reading1, reading2):
        # singleton membership fuzzification of output 1: angular velocity
        singleton_rotation_outputs = np.array([
            [-1],
            [0],
            [1]
            ],dtype=float)
        FD_rotation_outputs = np.ones(shape=singleton_rotation_outputs.
        shape, dtype=float)

        # singleton membership fuzzification of output 1: linear velocity
        singleton_translation_outputs = np.array([
            [-0.5],
            [0],
            [0.5]
            ],dtype=float)
        FD_translation_outputs = np.ones
        (shape=singleton_translation_outputs.shape, dtype=float)


        # read sensor [2] (left) and sensor [5] (right)
        fuzzy_x_1 = self.dis_MF(reading1)
        fuzzy_x_2 = self.dis_MF(reading2)
```

```
    # Rule table of linear velocity
    rule_table_1 = np.array(
        [
            [1, 2],
            [0, 1] # for now, only stop and move backward
            ],
        dtype=int)

    # Rule table of angular velocity
    rule_table_2 = np.array(
        [
            [0, 1],
            [1, 1]
            ],
        dtype=int)
```

```
    # r-th rule firing degree calculation
    # and defuzzification
    num1, num2 = 0, 0;
    den1, den2 = 0, 0;
    for r, rval in enumerate(fuzzy_x_2):
        for c,cval in enumerate(fuzzy_x_1):
            tab_idx_1 = rule_table_1[r][c]
            tab_idx_2 = rule_table_2[r][c]

            fd1andfd2 = min(cval,rval)

            FD_rotation_outputs[tab_idx_1] = fd1andfd2
            FD_translation_outputs[tab_idx_2] = fd1andfd2

            num1 = num1 + (fd1andfd2*singleton_rotation_outputs
            [tab_idx_1][0])
            den1 = den1 + (fd1andfd2)

            num2 = num2 + (fd1andfd2*singleton_translation_outputs
            [tab_idx_2][0])
            den2 = den2 + (fd1andfd2)

    crisp_out1 = num1/den1
    crisp_out2 = num2/den2
    return [crisp_out1, crisp_out2]
```

FIS-BASED DECISION MAKER

DECISION MAKER

Fuzzy rules example

minimum sensor reading	Tracking gain (*g*)
Near	0
Far	1

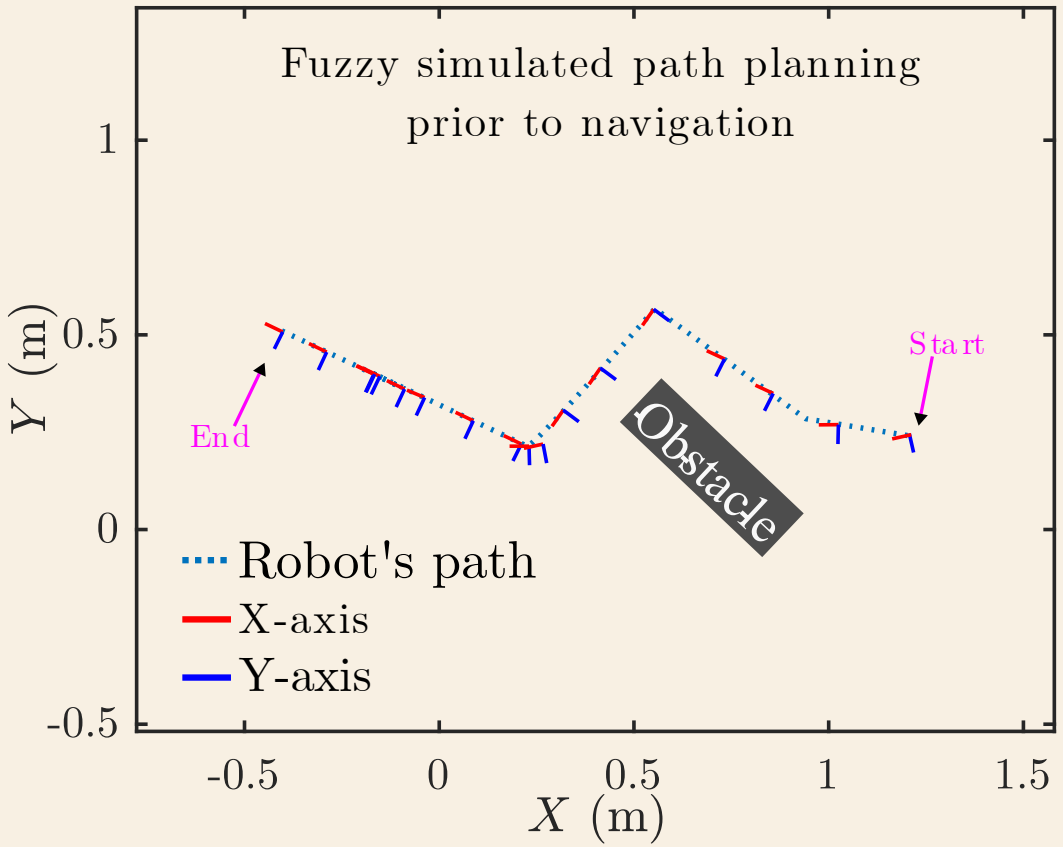
combinedLinearVelocity = trackingLinearVelocity*g + avoidanceLinearVelocity*(1-g)

combinedAngularVelocity = trackingAngularVelocity*g + avoidanceAngularVelocity*(1-g)

ASSIGNMENT

ASSIGNMENT

- Combine pose tracking and collision avoidance
  - (1 obstacle and 1 goal point)
- Record the robot's pose



In the report (PDF file):

- - Plot your fuzzification design of sensors
- - Plot your fuzzification design of distance
- - Show your fuzzy rule design in table style (tracking, avoidance, combination)
- Navigation plot

Files to be submitted:

- CoppeliaSim *.ttt file
- Python code *.py
- Report in *.pdf
- Demo video (youtube link)