nnn-n-obstacle-avoidance-review-w100-v4-hl-yy-2025-1-14.odp (Under Construction)

Harry Li, Ph.D.
CTI One Corporation
Santa Clara, CA 95051
Phone: (650) 463-9892
Version 2, 2024-4-13

CTI One Corporation is located in the Silicon Valley in Santa Clara, California. We design, develop and manufacture AI enabled smart rollators/walkers and exoskeleton devices for senior living operators, physically challenged as well as aging population for a better independent living. We design Artificial Intelligence, Computer Vision, and Robotics to modernize next generation autonomous mobility devices. Our smart rollators/walkers are battery powered, can be driven with a remote controller, and can respond and recognize the user and drive up to the user.

CTI One Corporation
This document is created by:
Harry Li, Ph.D.
Yusuke Yakuwa

Jan 14, 2025

Reference (1/10, 2025)

1. ROI intersection: 101-1-ROI-Intersection-v3-YY-HL-2021-4-20.odp

Background of the Obstacle Avoidance Algorithm 1/2 (1/2, 2025)

Step 1. YOLOv8 to identify the destination object ROI (user), register it as an original direction OD

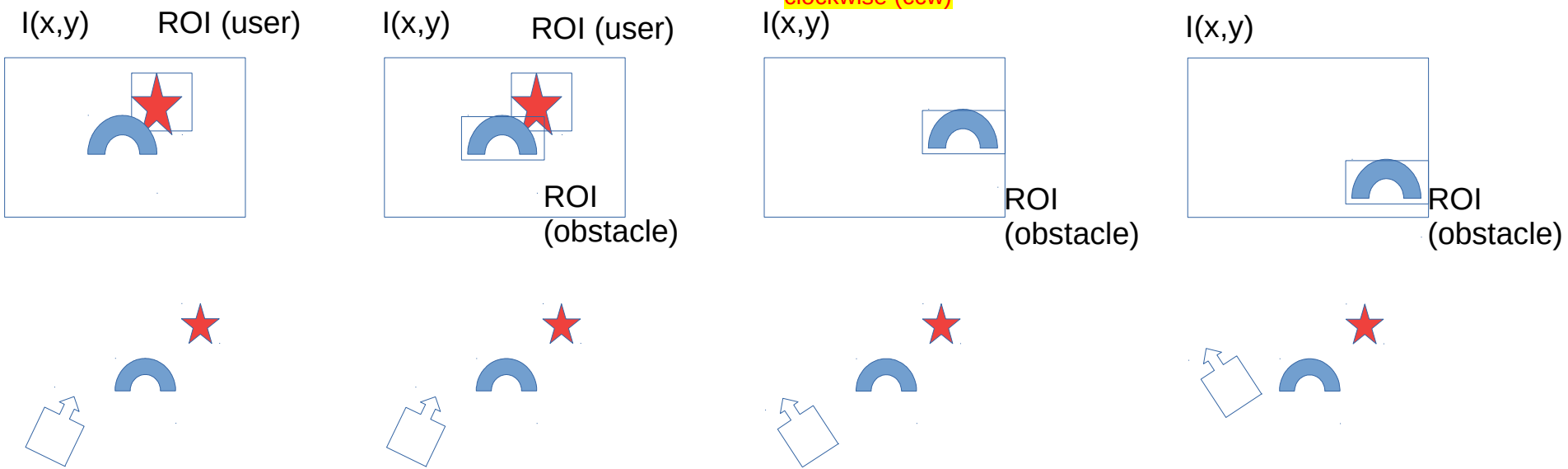
Step 2. YOLOv8 to identify an obstacle object ROI (a table for example)

Step 3. Drive W100 to a proper angle a1 away from OD, and estimate the adequate clearance passing space for W100

Left or right??? counter clockwise (ccw)

Step 4. Drive W100 to estimated distance along angle a_i.

Use multiple cameras???



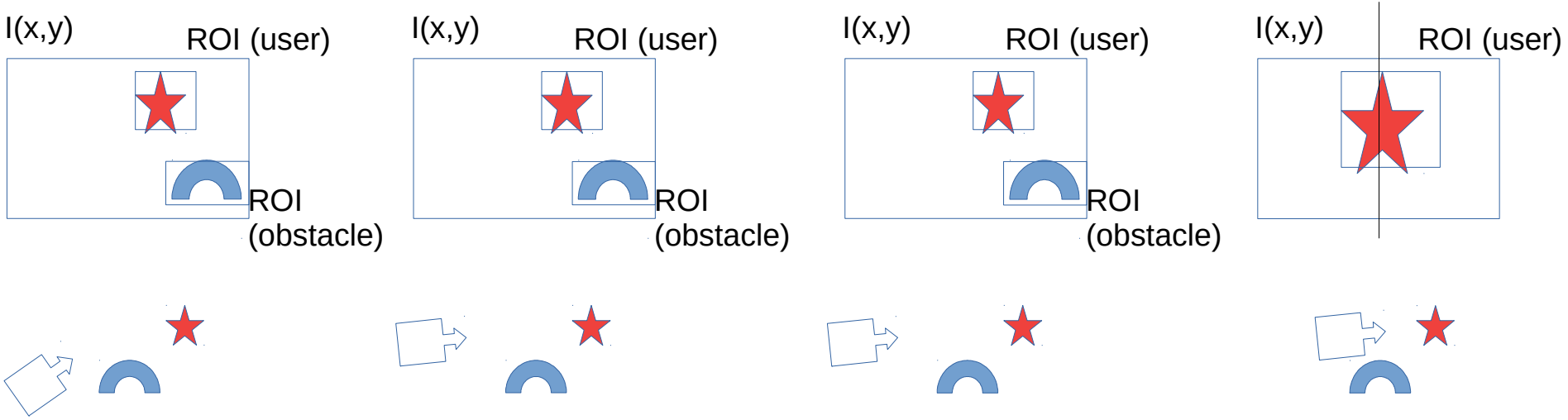
Background of the Obstacle Avoidance Algorithm 2/2 (1/2, 2025)

Step 5. Rotate (Drive) W100 till YOLOv8 observes destination object from the camera; the rotation angle of W100 can be estimated based on a_i.

Step 6. Estimate and adjust rotation if needed to an adequate clearance based on YOLOv8 ROI input. If an adequate clearance can not be achieved, then go back to Step 3.

Step 7. Identify user based on YOLOv8 from the current W100 location.

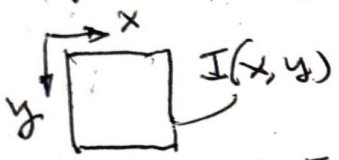
Step 8. Compute the user coordinate (x,y), then drive up W100 till reach it.



Collision Path Prediction Algorithm 1/6 (1/5-1/2, 2025)

Collision Path Prediction 1/6
2025-1-5 Sun

1. Image I(x,y) Definition



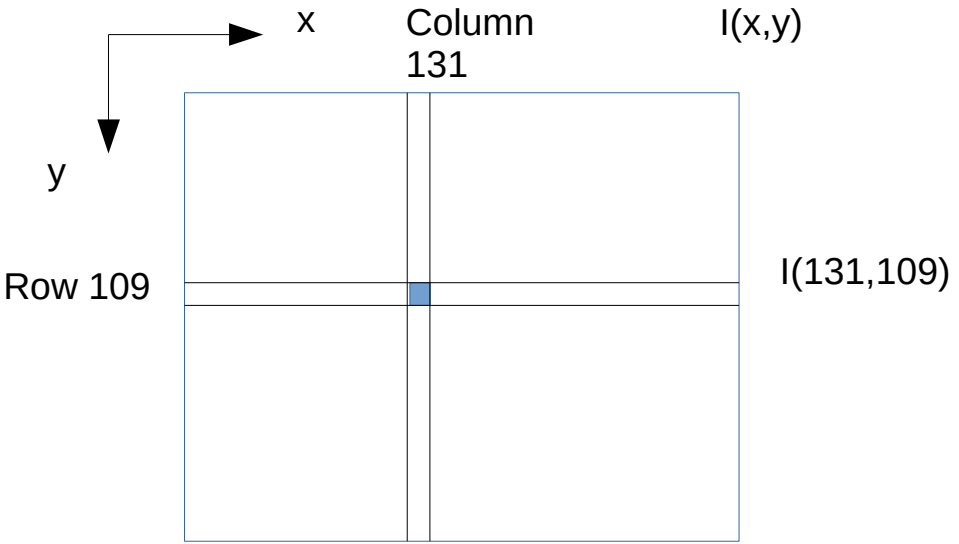
MxN Fig.1

I(x,y): Image.

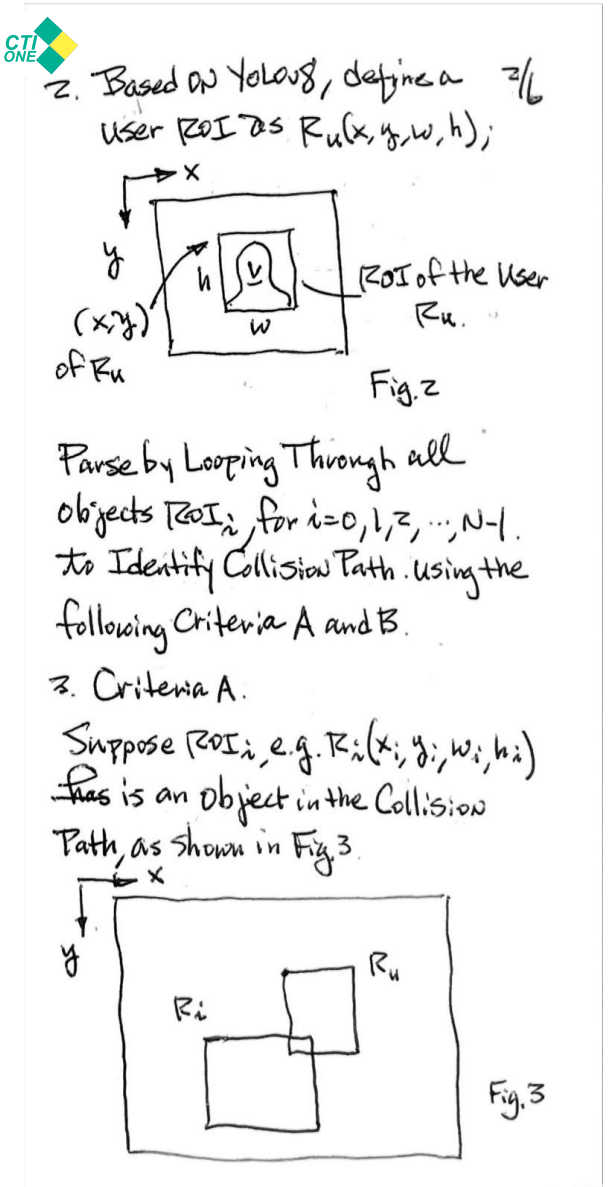
MxN: Resolution of I(x,y)
M for Number of Pixels Per Row; where I(x,y)
x: a pixel location (Col.)
~~a row~~;

N: Number of Pixels Per Col.; where I(x,y)
y: a pixel location (Row)

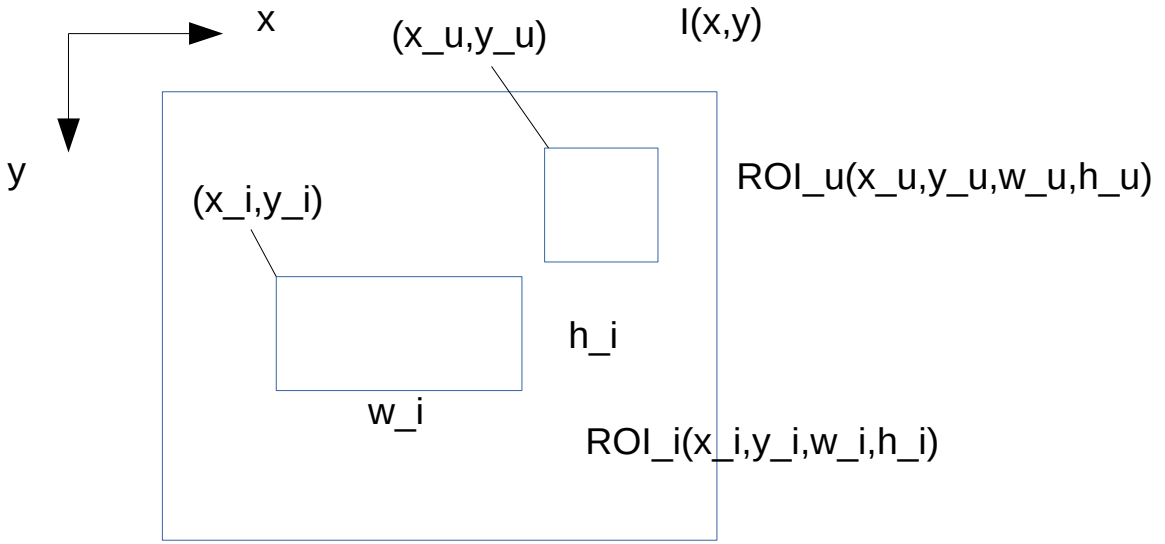
Example of image I(x,y) coordinate system and a pixel location (x,y)



Collision Path Prediction Algorithm 2/6 (1/5-1/2, 2025)



Example of ROI notation, ROI_i(x_i,y_i,w_i,h_i) for i=0,1,…,N-1



1. ROI_i(x_i,y_i,w_i,h_i) can be written as ROI_i as a simplified notation; and
2. ROI_i can be written as R_i as simplified notation, so ROI_i(x_i,y_i,w_i,h_i), R_i(x_i,y_i,w_i,h_i) and ROI_i, R_i are all referred to as the same ROI but with different details in notation. We will switch among these notations for the same ROI when needed.

Collision Path Prediction Algorithm 3/6 (1/5-1/2, 2025)

Compute IOU Index, IOU: Intersection of Union as illustrated in the Shaded Area in Fig. 4.

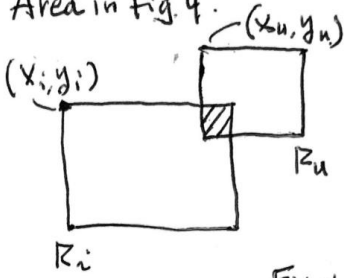


Fig. 4.

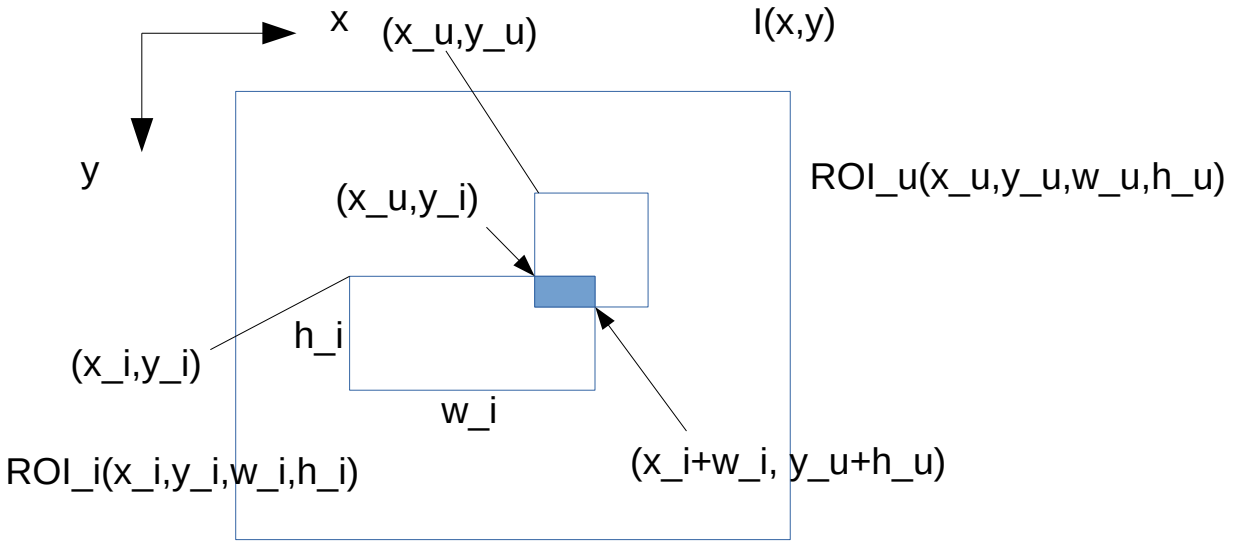
$$IOU = \frac{\text{Area of Intersection of } R_i \text{ and } R_u}{\text{Area of } R_u + \text{Area of } R_i} \quad \ldots (1)$$

$$= \frac{A(R_u) \cap A(R_i)}{A(R_u) \cup A(R_i)} \quad \ldots (2)$$

Where

$$\text{Area Intersection of } R_i \text{ and } R_u = A(R_u) \cap A(R_i) \quad \ldots (3)$$

Example: The coordinates of two corner points in the following figure, the upper left corner and the lower bottom corner:



Example: The intersection area in blue color. This area can be computed by width times height. So

The width:

$$W = x\_i + w\_i - x\_u \qquad \text{(a1)}$$

The height:

$$H = y\_u + h\_u - y\_i \qquad \text{(a2)}$$

The intersection are:

$$W \times H = (x\_i + w\_i - x\_u) \times (y\_u + h\_u - y\_i) \qquad \ldots \text{(a3)}$$

$= W_{IOU} \times R_{IOU}$ ...(4) 4/6

where

$W_{IOU} = X_i + w_i - x_u$ ...(5)

H_IOU = y_u+h_u – y_i (6)

as in Fig.5

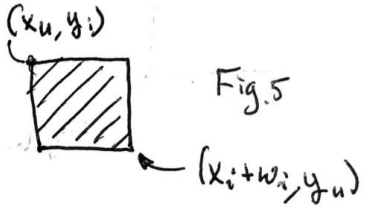


Fig.5

And

$A(R_u) = w_u \times h_u$ ...(7)

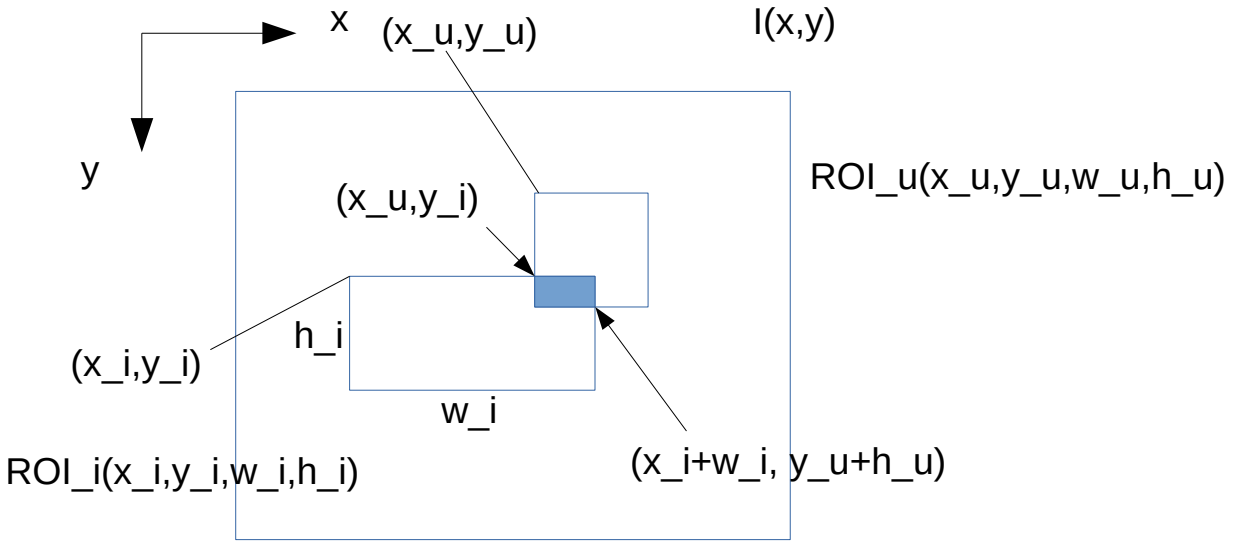
$A(R_i) = w_i \times h_i$ ...(8)

Hence,

$$IOU = \frac{W_{IOU} \times h_{IOU}}{w_u \times h_u + w_i \times h_i} = \frac{(X_i + w_i - x_u) \times (y_u + h_u - y_i)}{w_u \times h_u + w_i \times h_i}$$ ...(9)

Collision Path Prediction Algorithm 4/6 (1/8-1/2, 2025)

Example: The coordinates of two corner points in the following figure, the upper left corner and the lower bottom corner:



Example: The intersection area in blue color. This area can be computed by width times height. So

The width:

W_IOU = x_i+w_i – x_u (a1)

The height:

H_IOU = y_u+h_u – y_i (a2)

The intersection are:

W_IOU x H_IOU = (x_i+w_i – x_u) x (y_u+h_u – y_i) … (a3)

Collision Path Prediction Algorithm 5/6 (1/10-1/2, 2025)

Example: The centroid (center point) of the ROI.

Equation (6)

Note: Eqn (6) ≥ 0

Eqn (5) >= 0

We do not include the case when Eqn(6)<0, e.g. not consider

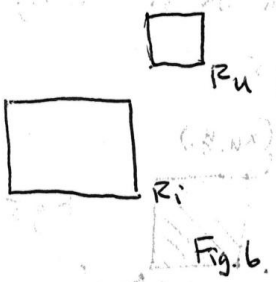


Fig. 6.

4. Criteria B,

R_i must be in Front of R_u. e.g.

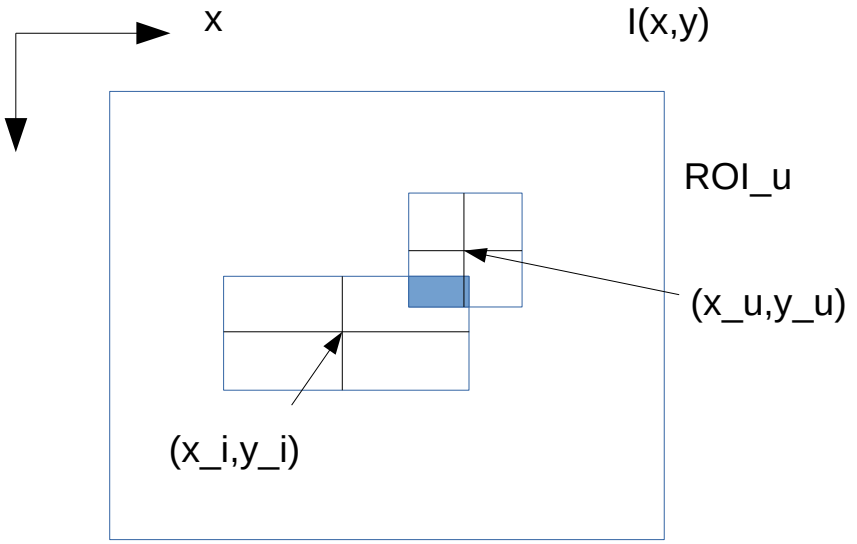
$$\overline{y_i} \geq \overline{y_u} \quad \cdots (10)$$

where

$$\overline{y_i} = \frac{y_i + (y_i + h_i)}{2} \quad \cdots (11)$$

YY 1/10/2025: Use cv2.rotatedRectangleIntersection() for Criteria A.

See 101-1-ROI-Intersection-v3-YY-HL-2021-4-20.odp



Collision Path Prediction Algorithm 6/6 (1/5-1/2, 2025)

6/6

$$\bar{y}_u = \frac{y_u + (y_u + h_u)}{2} \quad \ldots(12)$$

The Rational is the front object has Lower Centroid $(\bar{x}_i, \bar{y}_i)$ ROI.

5. Collision Path.

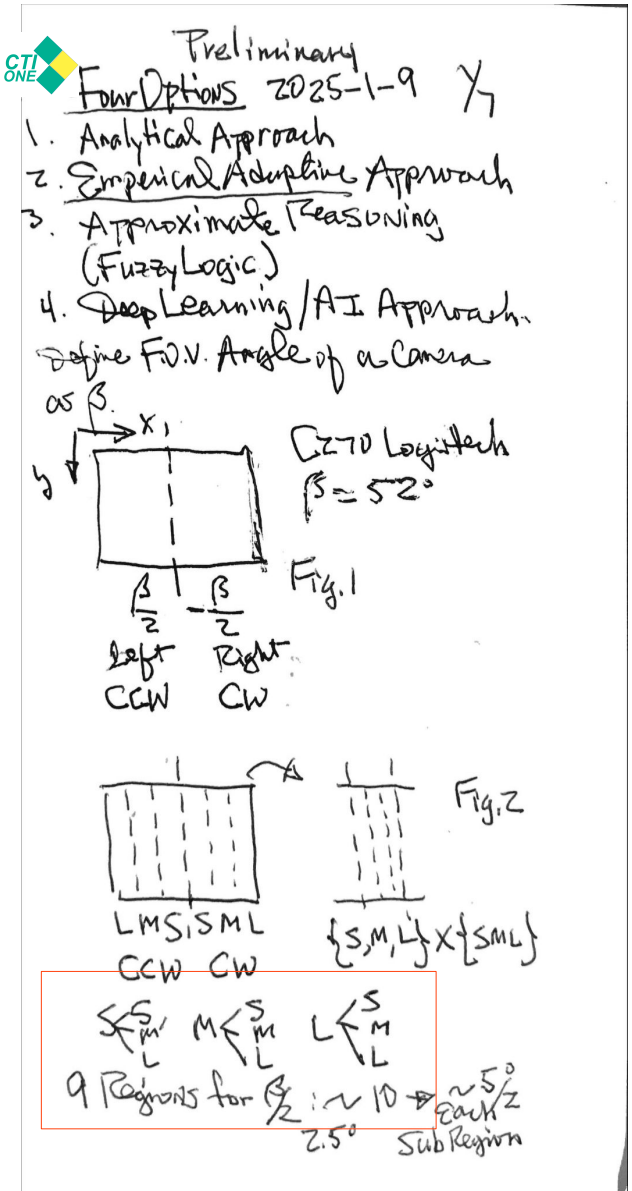
Collision Path is predicted when Both A and B hold good.

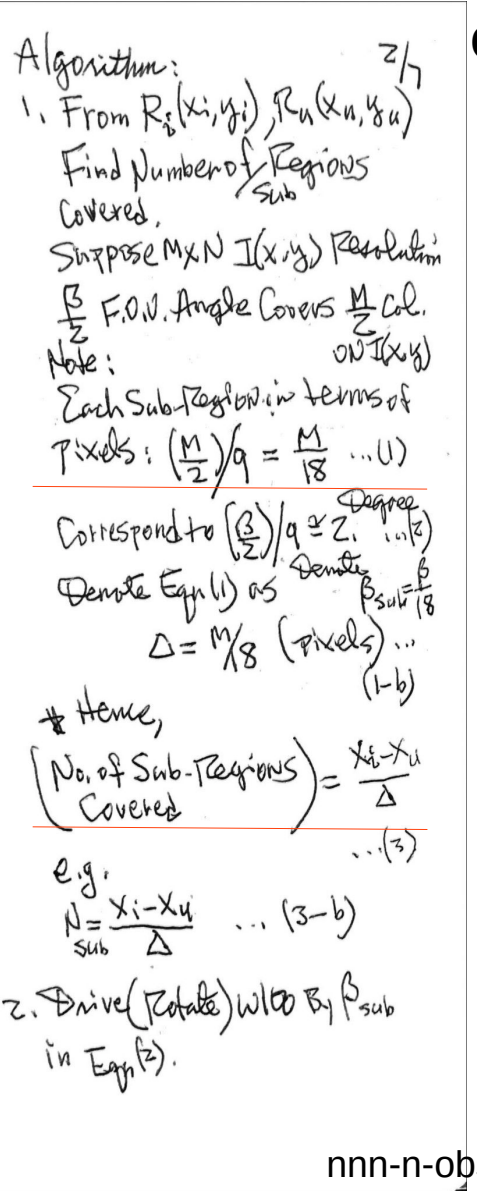
(END).

ToDo: Fig.6 Scenario.

The collision path is detected if both A and B holds good.

Collision Path Prediction Algorithm Part II (1/14-1/2, 2025)





HL 2025-1-14: find the number of sub-sub-regions between the object and the user

Collision Path Prediction Algorithm
Part II (1/14-1/2, 2025)

3/7

Re-calculate N_{sub} at step, this
Denoted as

$$N_{sub,1} = \frac{X_{i_1} - X_{u_1}}{\Delta} \quad \cdots(4)$$

And, Calculate Delta for adaptation

$$\delta_1 = N_{sub} - N_{sub1} \quad \cdots(5)$$

Save it.

3.

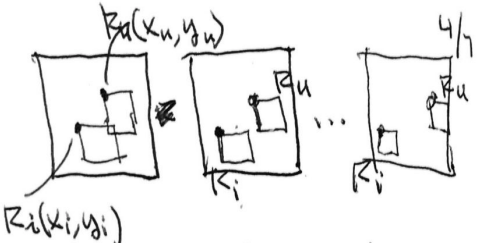
Repeat Step2. till $N_{sub} - 1$ for each Repeated Step.

$$\delta_i = N_{sub} - N_{sub\,i} \quad \cdots(5\text{-}6)$$

then

$$\sum_{i=1}^{N_{sub}-1} \delta_i = \delta_1 + \delta_2 + \cdots + \delta_{N_{sub}-1} \quad \cdots(6)$$

Note: Choose $N_{sub}-1$ for safty
Not to ~~lose~~ loss $R_u(x_u, y_u)$
on $I(x, y)$.

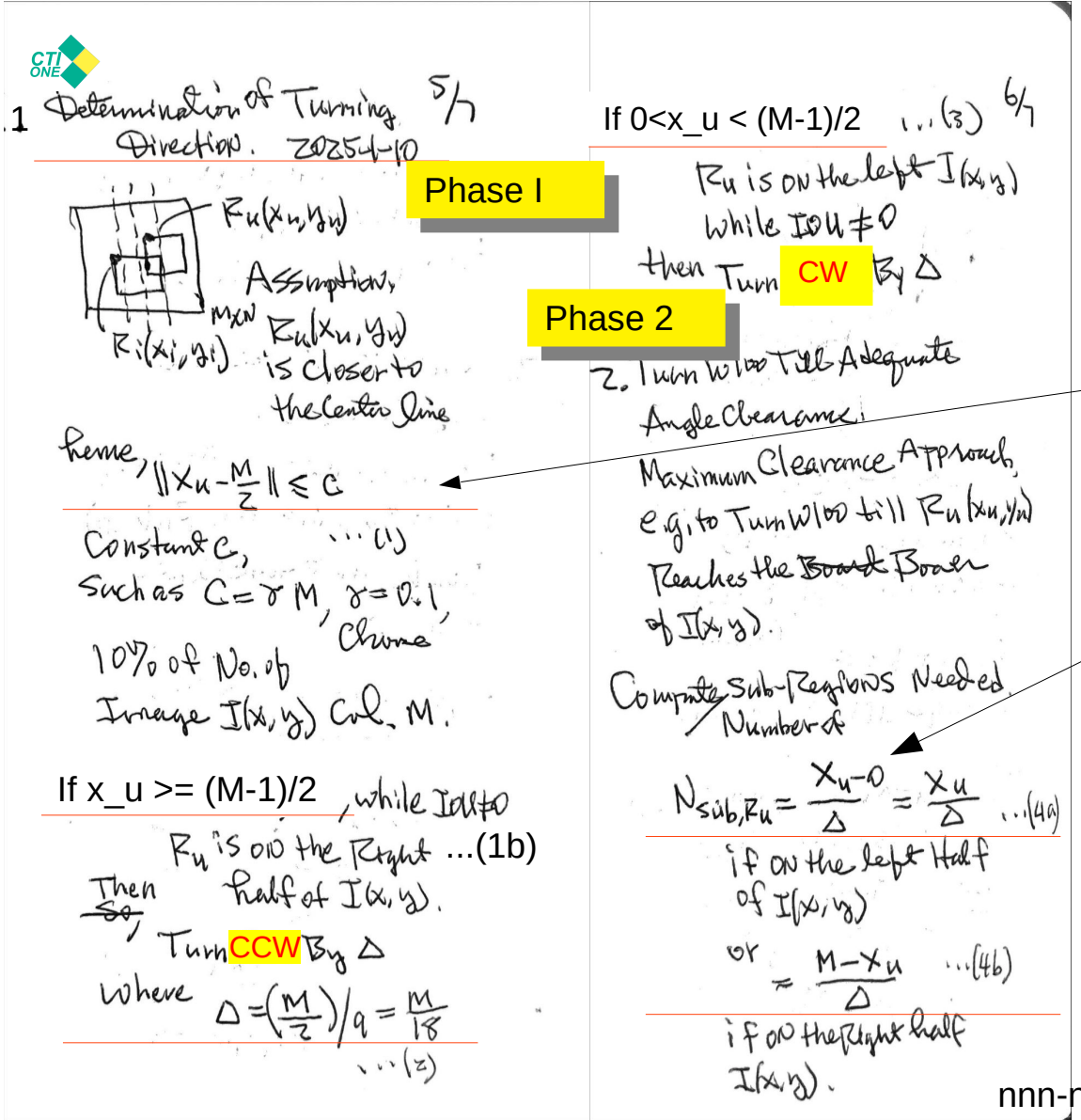


~~st~~ Time t_0 $t_1, \cdots, t_{N_{sub}-1}$

4. Compute R_i's Position Formed Trajectory End Point.

$$N_{sub,R_u(t_0,t_{N_{sub}-1})} = \frac{X_{t_{N_{sub}-1}} - X_{t_0}}{\Delta} \quad \cdots(7)$$

Collision Path Prediction Algorithm Part II (1/14-1/2, 2025)



Phase I determines the turning direction, Phase II determines the number of sub-sub-regions to cover in the turning.

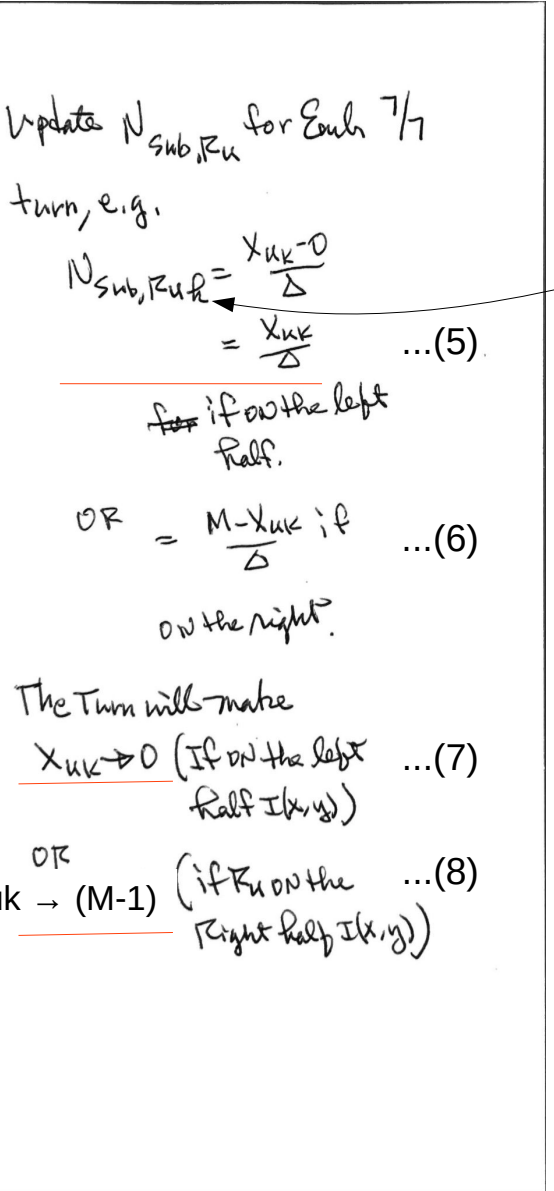
Eqn (1): ROI_u is near the center of I(x,y)

"0" is the 1st col. of I(x,y), drive W100 to make ROI_u to reach the boundary of the image at the left.

This equation gives the number sub-sub-regions from ROI_u to the 1st col of I(x,y) provided ROI_u is on the left image plane.

Eqn (4b) gives the number sub-sub-regions from ROI_u to the last col M-1 of I(x,y) provided ROI_u is on the right image plane.

Collision Path Prediction Algorithm
Part II (1/14-1/2, 2025)



Notation: R: ROI; R_u: ROI of the user; R_uk: ROI of the user at time (step) k;

Eqn (5) is computed/updated for each sub-sub-region covered due to the motor driven W100 movement.
Where k = 0, 1, 2, …

Eqn (6) is similar as eqn (5) except if R_u is on the right plane of I(x,y)

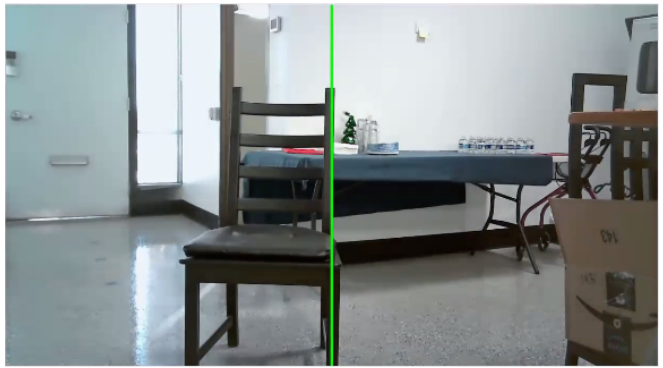
Eqn (7): R_u has reached the left most col of I(x,y)

Eqn (8): R_u has reached the right most col of I(x,y)

Testing Turning CW and CCW Direction (1/14, 2025)

100, 500, 1000 mm

Initial setup, created the environment to make the chair as ROI_i is the chair




```
print('Stop the code with Ctrl+C to stop running')
try:
    # Loop indefinitely until keyboard interrupt
    time.sleep(0.8)
    client.write_register(L_DEC_TIME, value: 1000, sla
    client.write_register(R_DEC_TIME, value: 1000, sla
    time.sleep(1)  # Allow some time for deceleration
```

0.8 for 15 degree, so 0.15 for 2.8 degree

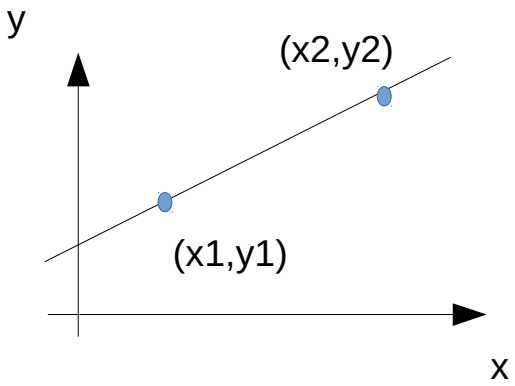
```
# ********************************************************************
if x1_u - ((width-1)/2) >= 0:
    # Turn w100 clockwise
    cv2.putText(draw, text: "CW", org: (10, 80), cv2.FONT_HERSHEY_SIMPLEX,
                fontScale: 3, color: (0, 0, 255), thickness: 3)

    n_sub = (width - x1_u) / (width / 18)
    if n_sub >= 1:
        if moving_flag is False:  # YY Move w100
            moving_thread = Thread(target=move_left_selfdriving, args=(0.15,))
            moving_thread.start()
else:
```

Testing Both CW and CCW for Turning (1/14-1/2, 2025)

cw2-testing-Screencast from 01-14-2025 02_28_34 PM (1).webm
~/Desktop/tmp/yusuke/orin/obstacle-avoidance/avoidance/turning/turning-testing$

Linear Interpolation for the Angles and Distances (1/14-1/2, 2025)



Given two points (x1,y1) and (x2,y2), the linear interpolation is illustrated as a straight line passing through both points, so we can write the following formula to describe this line

$$\frac{x - x_2}{x_1 - x_2} = \frac{y - y_2}{y_1 - y_2} \quad \ldots (1)$$

Re-arrange the above equation, eqn (1) can be written as the formula in eqn (2) .

$$y = \frac{y_1 - y_2}{x_1 - x_2} x - \frac{y_1 - y_2}{x_1 - x_2} x_2 + y_2$$

Example:
Given sin(15 degree) = 0.2588; sin(17 degree) = 0.2924
Find sin(15.93 degree), so we have

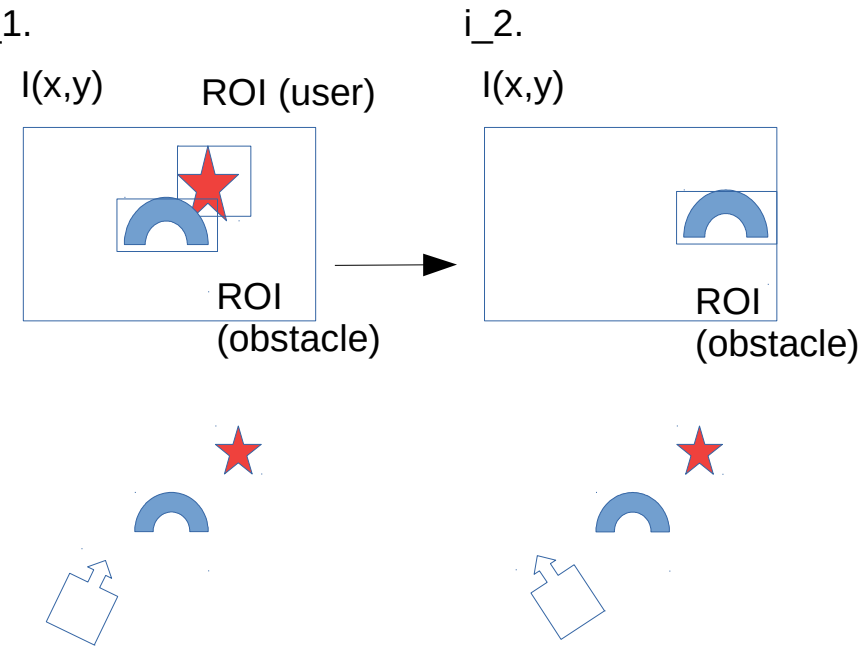
$$y = \frac{\sin(15 \text{ degree}) - \sin(17 \text{ degree})}{15 - 17} \times 15.93 - \frac{\sin(15 \text{ degree}) - \sin(17 \text{ degree})}{15 - 17} \times 17 + \sin(17 \text{ degree})$$

Ref: Harry Li's lecture notes for CMPE 240 at SJSU, pp. 59
https://github.com/hualili/CMPE240-Adv-Microprocessors/blob/master/2018F/2021F-101b-notes-cmpe240-2021-12-1.pdf

Driving W100 Forward with Arbitrary Distance, Such As 10, 100, 345 mm (1/14-1/2, 2025)

Calculation of Collision Avoidance Path Angle (1/9, 2025)

Step 3. Drive W100 to a proper angle a1 away from OD.



Calculation of Collision Avoidance Path Angle (1/9, 2025)

1. I(x, y) image with width(M) and height(N)

2. R_i1(x1, y1) is the current ROI location

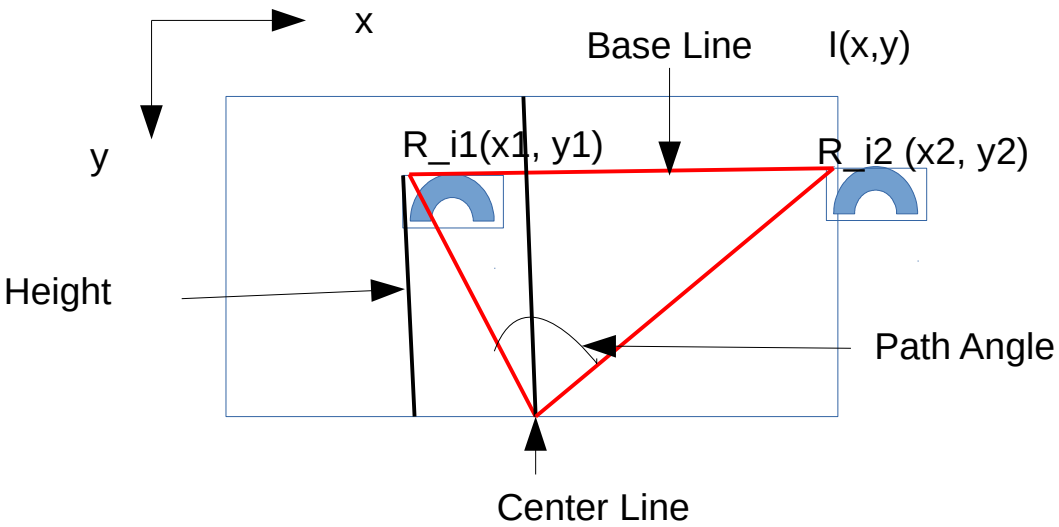
3. R_i2(x2, y2) is the result ROI location
   x2 == x and y1 == y2

4. Base Line = x - x1
   Height = y - y1

5. The angle calculation formula

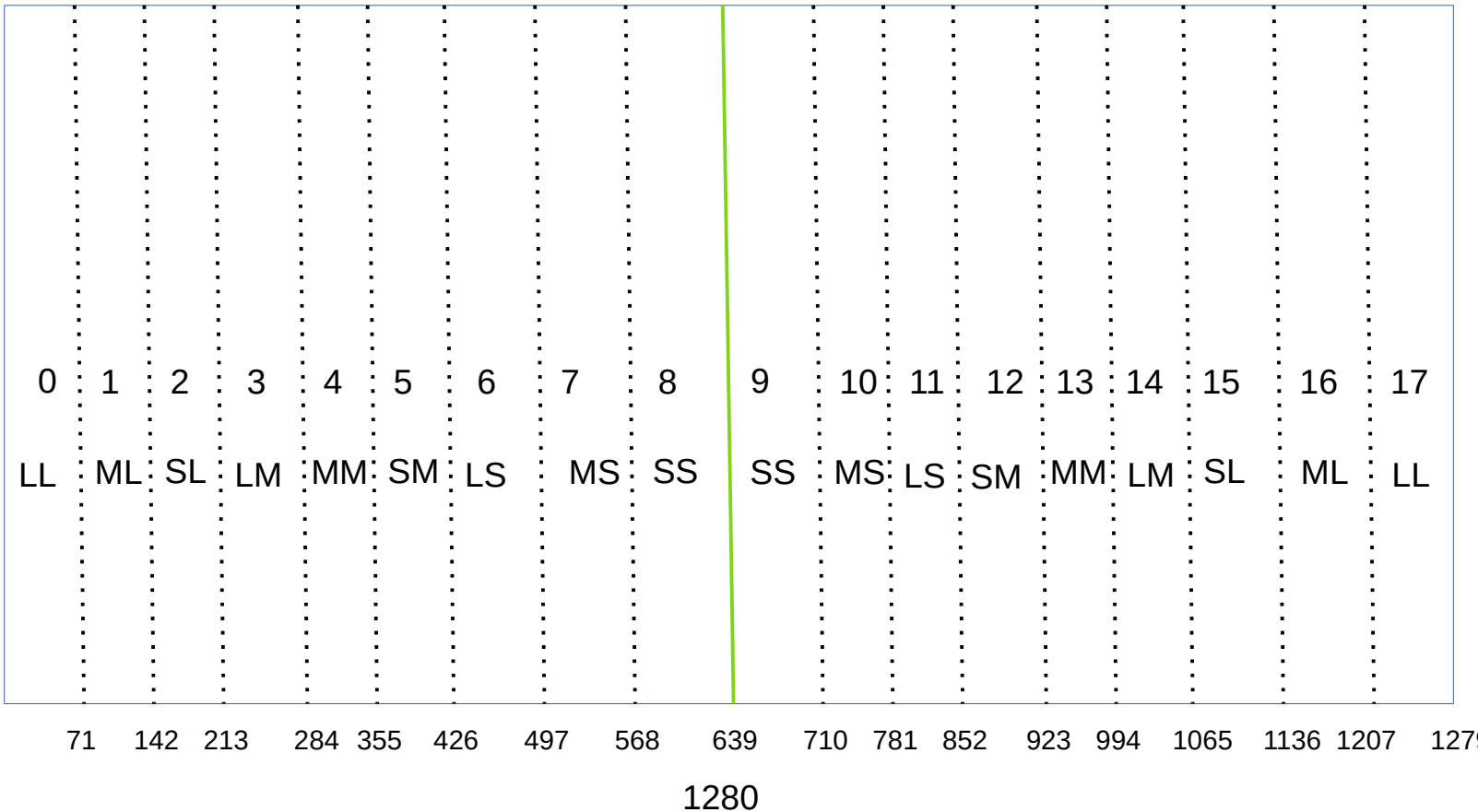
$$\text{Angle}(\theta) = \arctan\left(\frac{\text{Height}}{\text{Base}}\right)$$

$$\text{Angle in degrees} = \theta \times \frac{180}{\pi}$$



Suppose the image size is 720P (1280x720),
i_1(500, 200), i2(1280, 200)
Angle(θ) = arctan ((720-200) / (1280-500))
= arctan (520 / 780)
= arctan (0.666666667)
= 0.588
Degree = 0.588 * 180 / π
= 33.69

Sub Region (1/15, 2025)



Moving Duration (1/14, 2025)

1/14, 2025 YY the W100 turning duration is set as **0.15** seconds in roi_collision_phase3.py

0.15 seconds comes from the experiment that W100 turns 15 degrees, drive_w100_left_15_modbus_orin.py.

```
client.write_register(L_CMD_RPM_REG, int32_to_int16(LEFT_RIGHT_ASSIST_SELF_RPM), slave=MOTOR_ID)  # Using
client.write_register(R_CMD_RPM_REG, int32_to_int16(LEFT_RIGHT_SELF_RPM), slave=MOTOR_ID)  # Using single

print('Stop the code with Ctrl+C to stop running')
try:
    # Loop indefinitely until keyboard interrupt
    time.sleep(0.8)
    client.write_register(L_DEC_TIME, value: 1000, slave=MOTOR_ID)  # Set left motor deceleration time to
    client.write_register(R_DEC_TIME, value: 1000, slave=MOTOR_ID)  # Set left motor deceleration time to
```

1. 0.8 seconds provides 15 degrees
   1 degree duration = 0.8 /15 = 0.053 seconds

2. If C270 webcam provides 52 degree screen view and the screen image is divided into 18 sub subregion,
   1 sub subregion = 52 / 18 = 2.88 degree

3. 2.88 degree duration = 2.88 x 0.053 = 0.1536 seconds

End