Complete Coverage Navigation for Autonomous Clay Roller in Salt-Farming Application

Norawit Nangsue – ST110

Institute of Field Robotics, King Mongkut's University of Technology Thonburi
Bangkok, THAILAND

King Mongkut's University of Technology Thonburi

FIBO

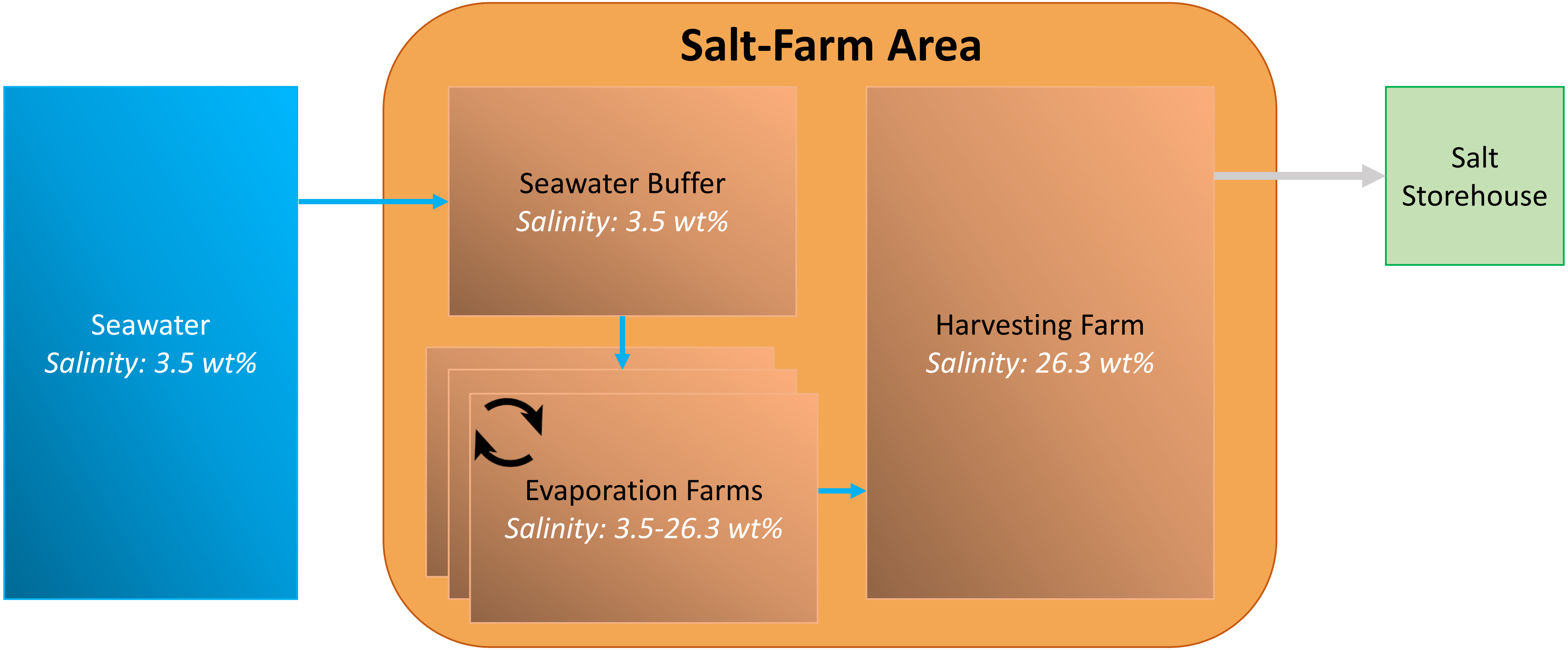
Download Slides

This presentation is a part of 2021 6th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS 2021)

Background

- Basic of salt-farming process in Thailand
- Why do we need to smoothen our farm?
- A tool to ease on roughness problem: Clay Roller

Basic of salt-farming process in Thailand



Salinity (/səˈlɪnɪti/) is the saltiness or amount of salt dissolved in a body of water, called saline water
source: https://en.wikipedia.org/wiki/Salinity

Why do we need to smoothen our farm?

Currently, salt-gathering are performed by a farmer stepping their feet to the farm field which causes an ***order of roughness*** in the farm field.

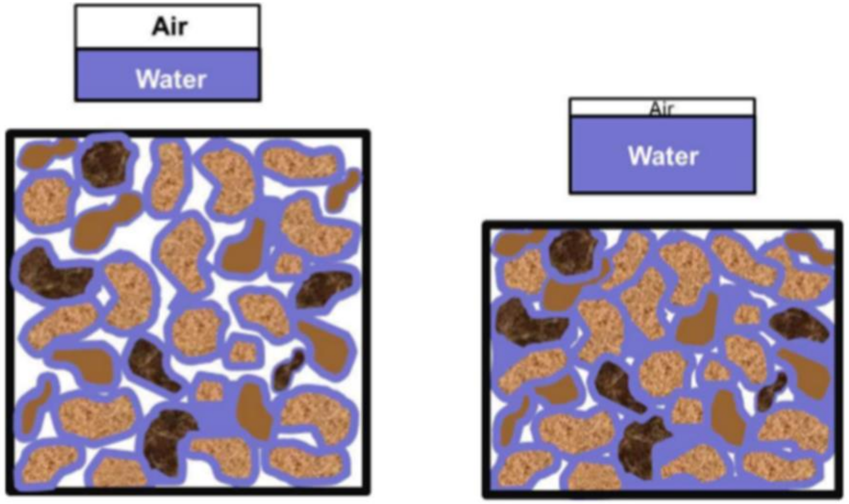
Neglecting this problem leads to…

- ***slowness*** of the evaporation rate
- ***inferior salt quality*** for the next agricultural cycle

Clay Roller

Nowadays, we used ***Clay Roller*** for rolling over and over on the clay farm field. (*and farmers are still driving it.)*

This makes the clay ***tighter*** and ***smoother*** which is more ideal for salt-farming.



Complete Coverage Navigation for Autonomous Clay Roller in Salt-Farming Application

Norawit Nangsue – ST110

Institute of Field Robotics, King Mongkut's University of Technology Thonburi
Bangkok, THAILAND

ACIRS

King Mongkut's University of Technology Thonburi

FIBO

Download Slides

This presentation is a part of 2021 6th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS 2021)

“Complete Coverage Navigation-type” Mobile Robot

Deterministic Navigation

- Certain converging time
- Predictable behavior

Random Navigation

-Uncertain complete coverage converging time.

-Widely used in differential drive kinematics.

-Unpredictable behavior

Scope of this work

Path Planner aka. global_planner

- Plan the overall complete coverage path that suits with *“**Bicycle Kinematics**”*

Path Tracker aka. local_planner

- Control the clay roller to follow the generated path.

Not in this scope of work

Low Level Control *(for controlling steering angle and driving speed)*

- Hardware-Level PID controller has been done

Localization

- System is assumed to use RTK which provided 5 cm. of accuracy.

Rear-Steering Kinematics Fundamental

Most literatures describes about ***front-steering*** bicycle model

Continuous Kinematics of Rear Steering Bicycle Model

According to rolling constraint to the front wheel and the sliding constraint from both wheels, we obtains

$$\begin{bmatrix} \sin\left(\frac{\pi}{2}\right) & -\cos\left(\frac{\pi}{2}\right) & 0 \\ \cos\left(\frac{\pi}{2}\right) & \sin\left(\frac{\pi}{2}\right) & 0 \\ \cos\left(\frac{\pi}{2}+\delta\right) & \sin\left(\frac{\pi}{2}+\delta\right) & L\sin\left(-\frac{\pi}{2}+\delta\right) \end{bmatrix} \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} r\dot{\varphi} \\ 0 \\ 0 \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta} \end{bmatrix} = r\dot{\varphi} \begin{bmatrix} 1 \\ 0 \\ -\frac{\tan\delta}{L} \end{bmatrix} \quad (2)$$

$$\begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta} \end{bmatrix} = r\dot{\varphi} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -\frac{\tan\delta}{L} \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta} \end{bmatrix} = r\dot{\varphi} \begin{bmatrix} \cos\theta \\ \sin\theta \\ -\frac{\tan\delta}{L} \end{bmatrix} \quad (4)$$

*The frame position is at the middle of front wheel axle.

[1] I. R. N. a. D. S. Roland Siegwart, Introduction to Autonomous Mobile Robots, London: The MIT Press, 2011.

Discrete Kinematics of Rear Steering Bicycle Model

$$\begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta} \end{bmatrix} = r\dot{\varphi} \begin{bmatrix} \cos\theta \\ \sin\theta \\ -\dfrac{\tan\delta}{L} \end{bmatrix} \qquad (4)$$

$\dot{x} = \Delta x / \Delta t$

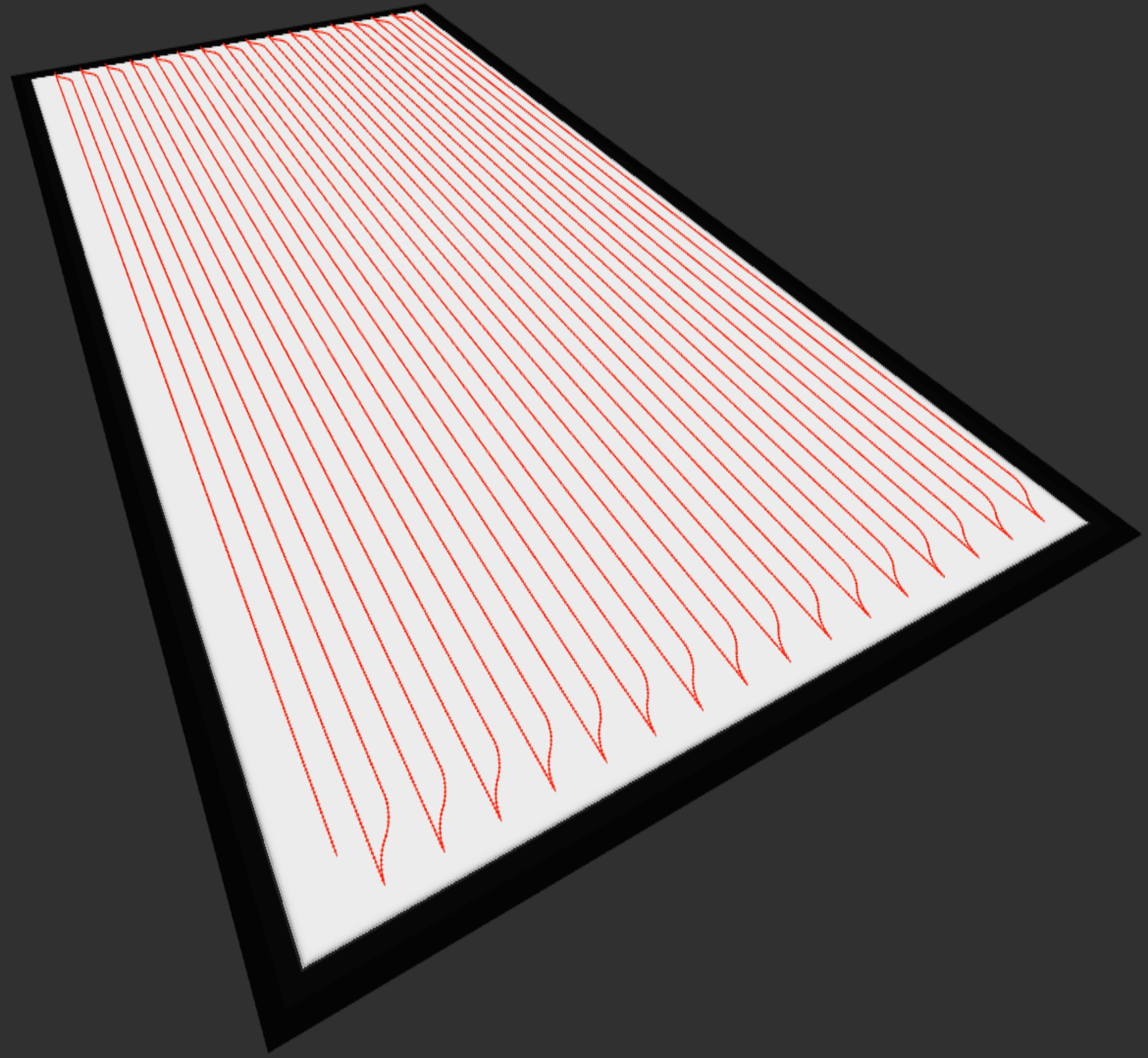
$$\begin{bmatrix} \Delta x_I \\ \Delta y_I \\ \Delta \theta_I \end{bmatrix} = r\Delta\phi \begin{bmatrix} \cos\theta \\ \sin\theta \\ -\dfrac{\tan\delta}{L} \end{bmatrix} \qquad (5)$$

$$\begin{bmatrix} x_I(n+1) \\ y_I(n+1) \\ \theta_I(n+1) \end{bmatrix} = \begin{bmatrix} x_I(n) \\ y_I(n) \\ \theta_I(n) \end{bmatrix} + r\Delta\phi \begin{bmatrix} \cos\theta \\ \sin\theta \\ -\dfrac{\tan\delta}{L} \end{bmatrix} \qquad (6)$$

This form will be used in ***Path Planner***

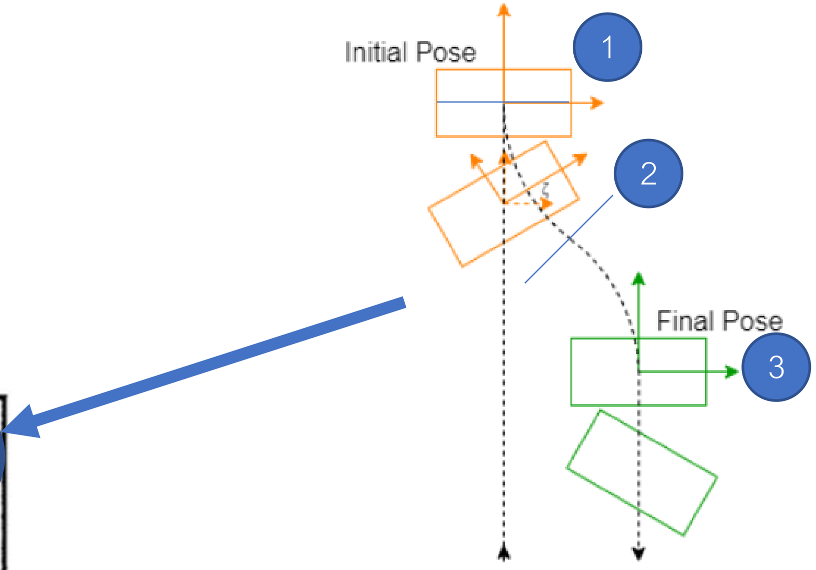
Complete Coverage
Path Planner

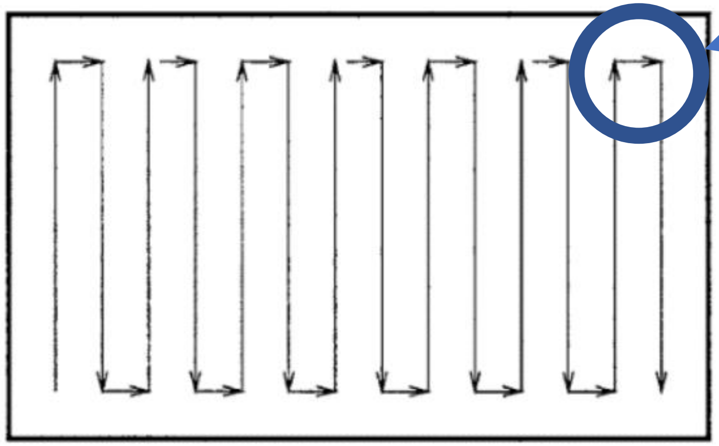
- Path Modification
- Costmap and Thresholds
- Overlapping Angle and Turning Angle
- Algorithm Flowchart



Path Modification from Back-and-forth Motion

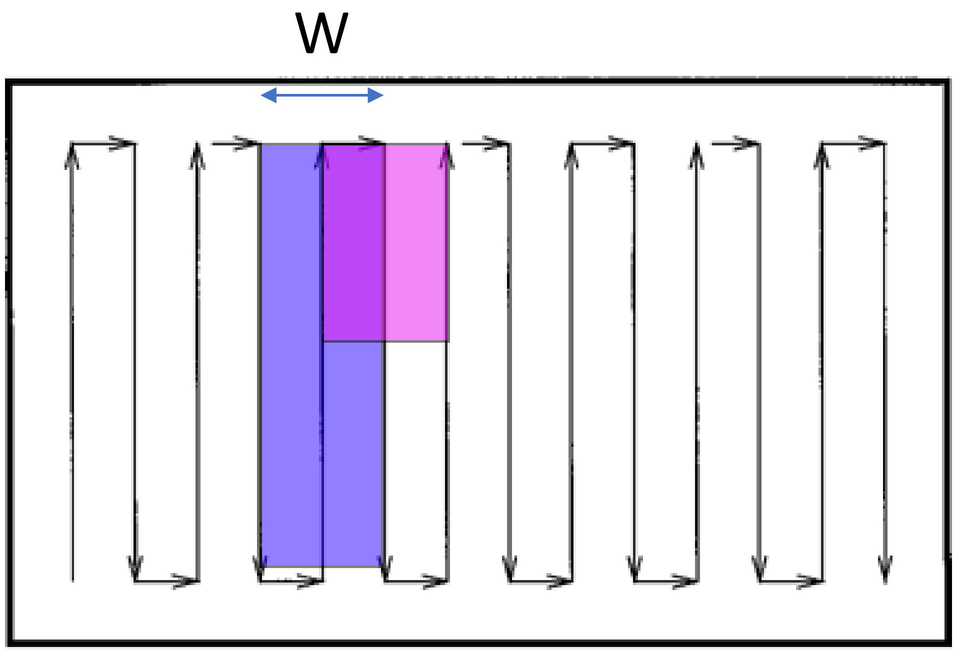
Normal ***back-and-forth**** path doesn't suit with ***bicycle kinematic***, path modification is, therefore, applied to ***every corner***.





* P. P. Choset H., "Coverage Path Planning: The Boustrophedon Cellular Decomposition," in *Field and Service Robotics*, London, 1998

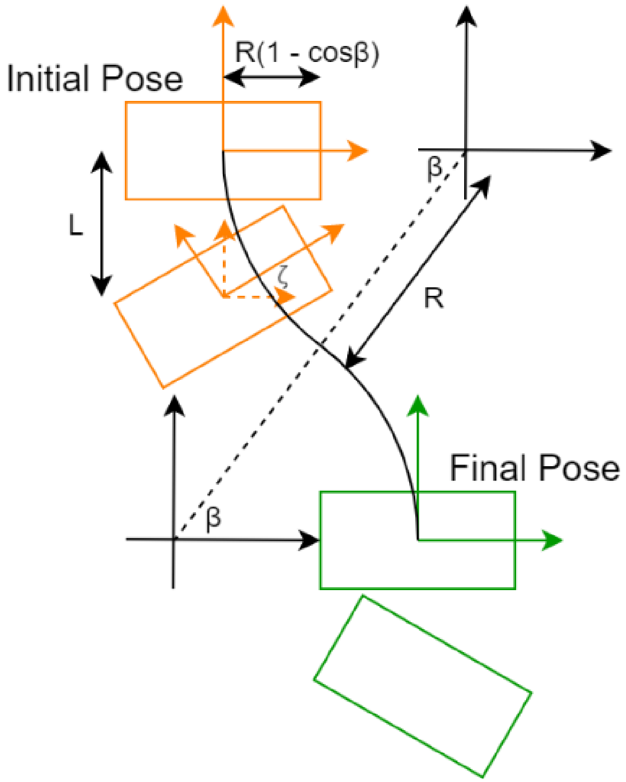
Overlapping Number



In this example, $\Delta y^R = \frac{W}{2}$, therefore, $\eta = 2$

$$\eta = \frac{W}{\Delta y^R} \tag{7}$$

Turning Angle



$$\eta = \frac{W}{\Delta y^R} \qquad (7)$$

According to the geometry,

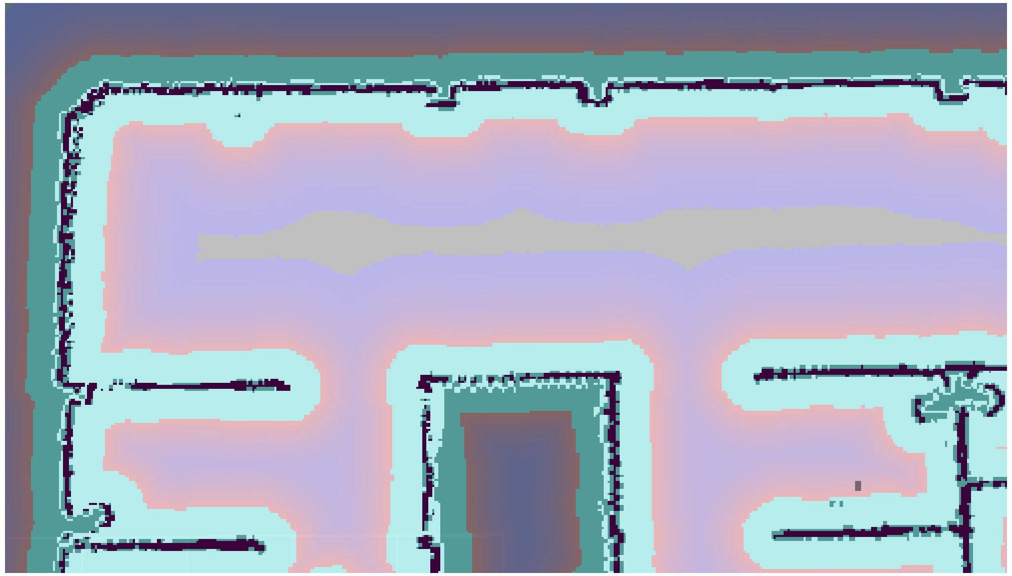
$$R = L \cot \delta_{max} \qquad (8)$$

$$\Delta y^R = 2R(1 - \cos \beta). \qquad (9)$$

From (7), (8), and (9) we could conclude that

$$\beta = \cos^{-1}\left(1 - \frac{W}{2\eta L \cot \zeta_{max}}\right) \qquad (10)$$

Inflation Costmap and Thresholds

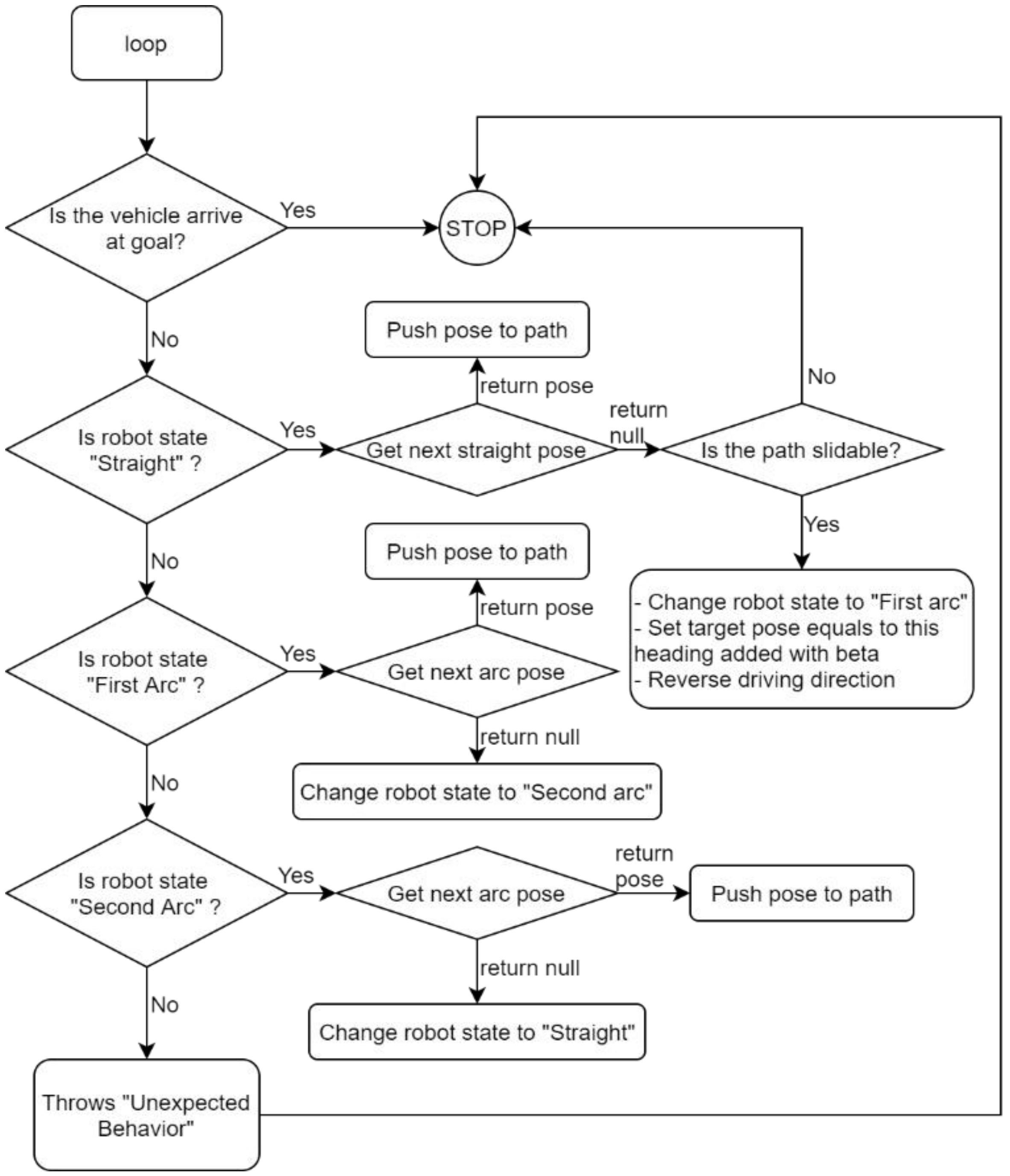


It represents the cost (difficulty) of traversing different areas of the map

In the program procedure, the operation could be depended on the costmap value of that position

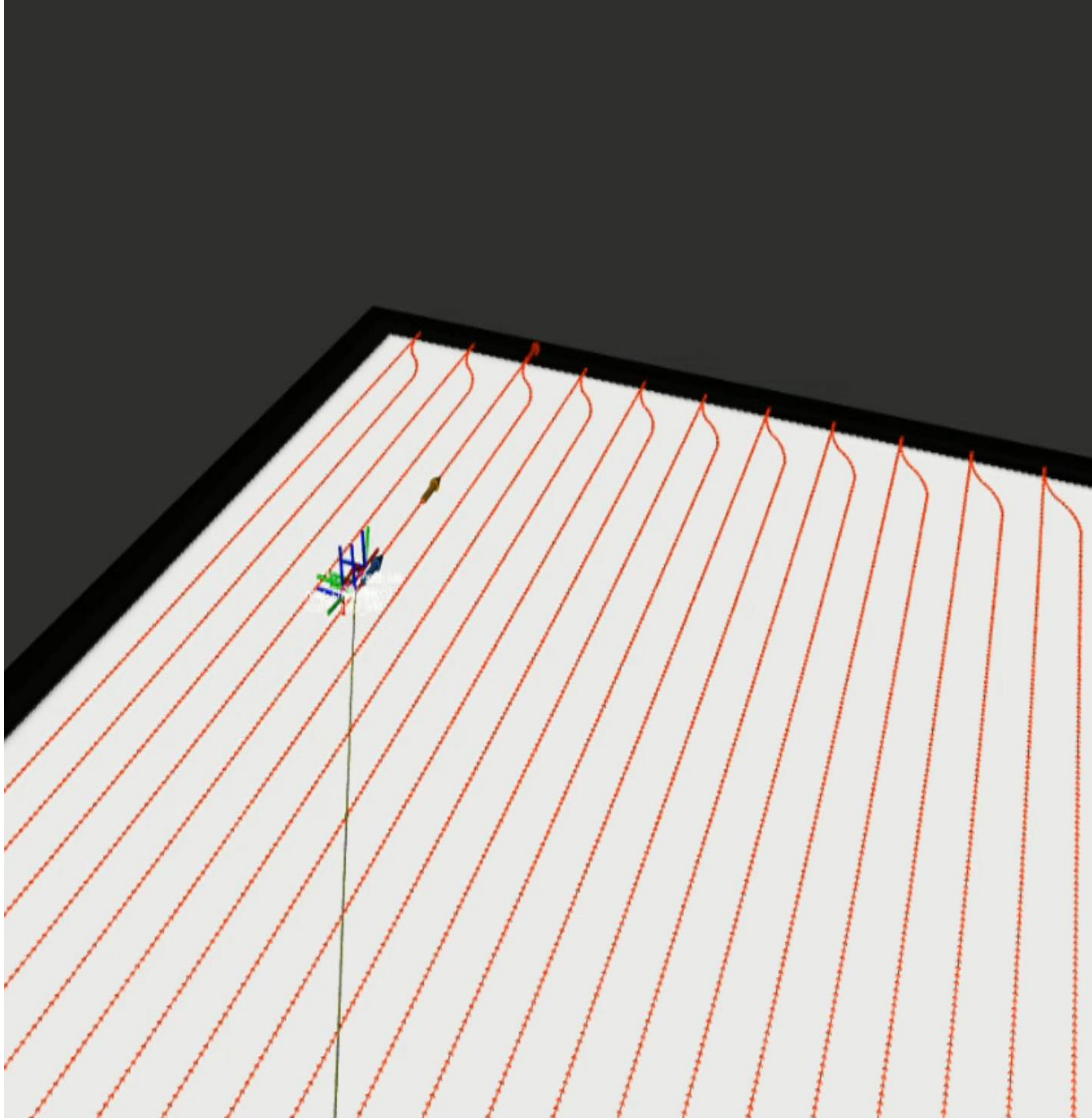
source: https://answers.ros.org/question/309308/what-is-a-costmap/

(Simple) Algorithm Flowchart



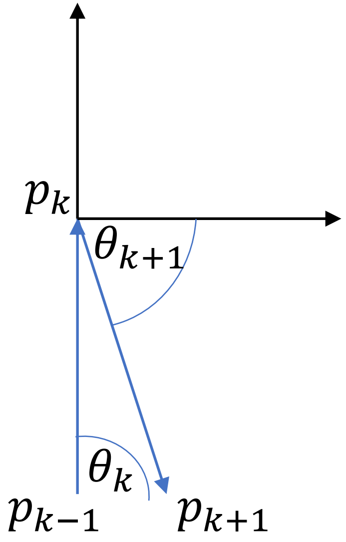
Complete Coverage
Path Tracker

- Turning Edge Detector
- Steering Control with Pure Pursuit
- Driveline Control with Trapezoidal Velocity Profile
- Supervisor



Turning Edge Detector

A detector is used to detect edges of the generated path to inform other modules.

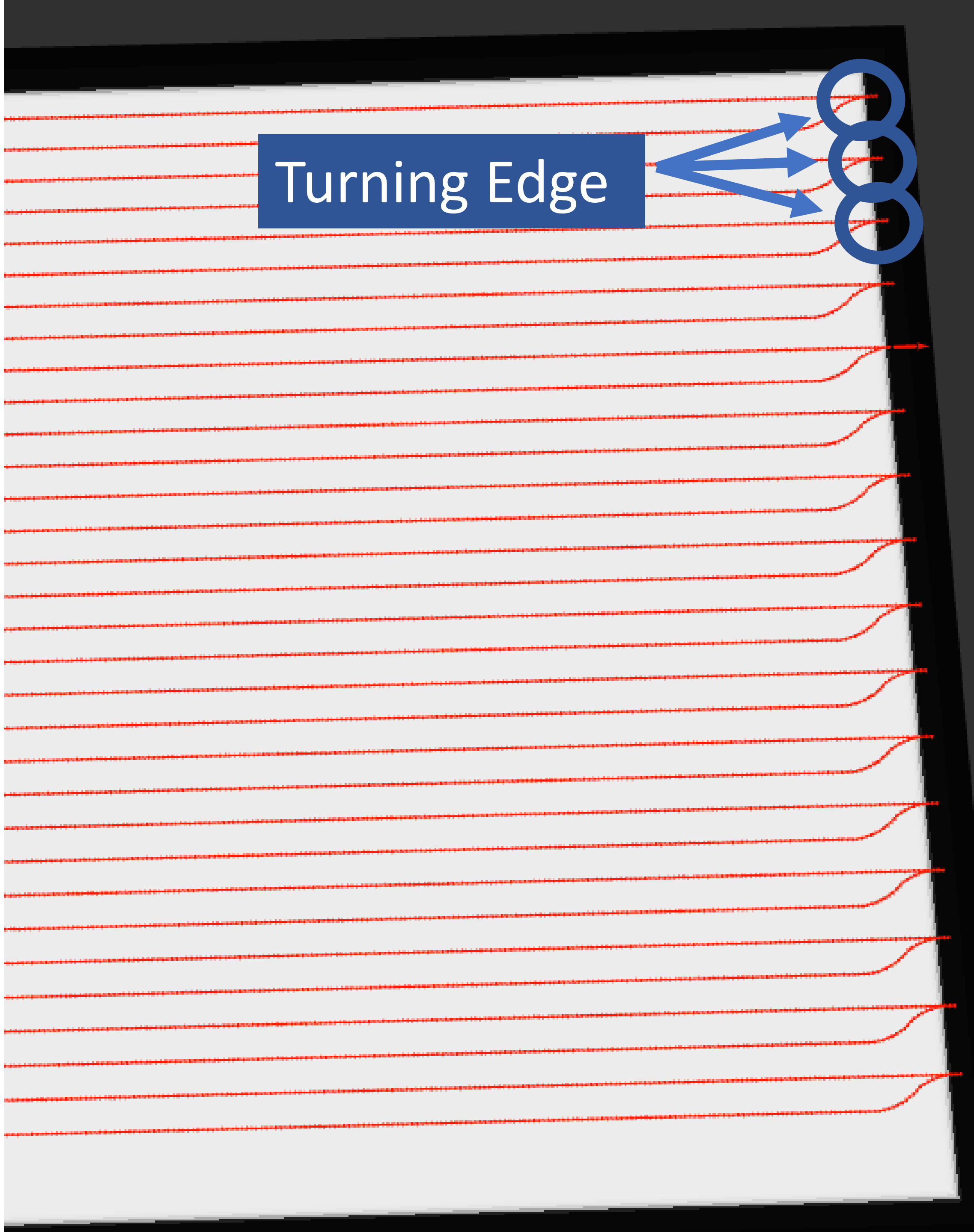


$$p_{edge} = \begin{bmatrix} x_{edge} \\ y_{edge} \end{bmatrix}$$

$$\Delta p_k = p_k - p_{k-1} = \begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix}$$

$$p_{edge} = \left\{ p_k \ such\ that\ \left| \operatorname{atan}\left(\frac{\Delta y_{k+1}}{\Delta x_{k+1}}\right) - \operatorname{atan}\left(\frac{\Delta y_k}{\Delta x_k}\right) \right| \geq \frac{\pi}{2} \right\}$$

$$p_{edge} = \left\{ p_k \ such\ that\ |\theta_{\mathrm{k+1}} - \theta_k| \geq \frac{\pi}{2} \right\}$$



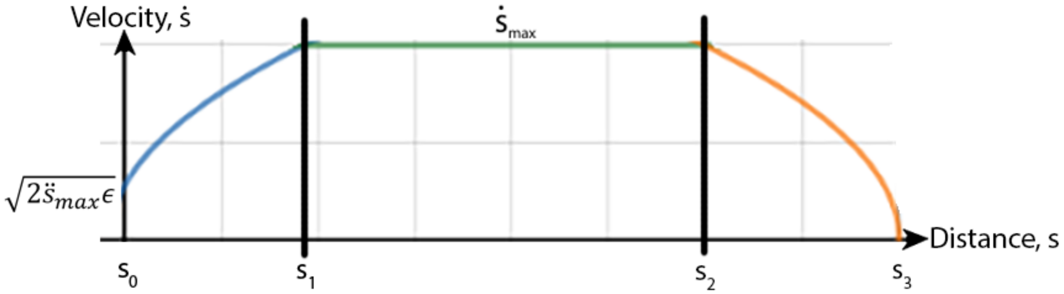
Driveline Control with Trapezoidal Velocity Profile

This controller is based on a simple equation of motion:

$$\dot{s}^2(t) = \dot{s}^2(0) + 2\ddot{s}_{max}s(t) \longrightarrow \dot{s}(s) = \sqrt{\dot{s}^2(0) + 2\ddot{s}_{max}s}$$

This control law is feasible due to the fact that we've a control authority over $\dot{s}(s)$.

$$\dot{s}(s) = \begin{cases} \sqrt{2\ddot{s}_{max}(s - s_0 + \epsilon)} & ; \; s_0 \le s < s_1 \\ \dot{s}_{max} & ; \; s_1 \le s \le s_2 \\ \sqrt{2\ddot{s}_{max}(s_3 - s)} & ; \; s_2 < s \le s_3 \end{cases}$$



where s_0 is the start position of a lane.

s_3 is the end position of a lane (turning edge)

$$s_1 = s_0 + \frac{\dot{s}_{max}^2}{2\ddot{s}_{max}} - \epsilon \text{ (the position when it stops accelerating)}$$

$$s_2 = s_3 - \frac{\dot{s}_{max}^2}{2\ddot{s}_{max}} \text{ (the position when it starts decelerating.)}$$

Steering Control with Pure Pursuit

1) Obtain target point robot frame, p_T^R from

$$p_T^R = T_W^R p_T^W$$

where $$T_W^R = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & -x_R^W \cos(\varphi) - y_R^W \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) & x_R^W \sin(\varphi) - y_R^W \cos(\varphi) \\ 0 & 0 & 1 \end{bmatrix}$$

2) Obtain the path curvature, γ, from

$$\gamma = \frac{2y_T^R}{l^2},$$

3) Obtain the target steering angle, δ, from

$$\delta = \tan^{-1}(-\gamma L).$$

Nomenclature

p_T^W is the target point in world frame

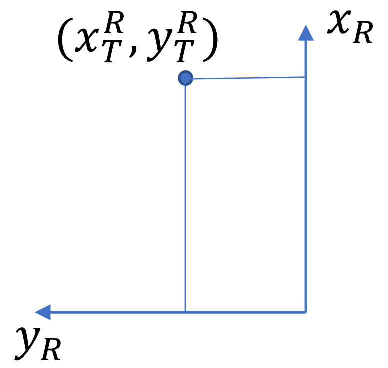
p_T^R is the target point in robot(clay roller) frame

T_R^W is the transformation from robot frame to world frame

γ is the path curvature

l is the lookahead distance

δ is the target steering angle

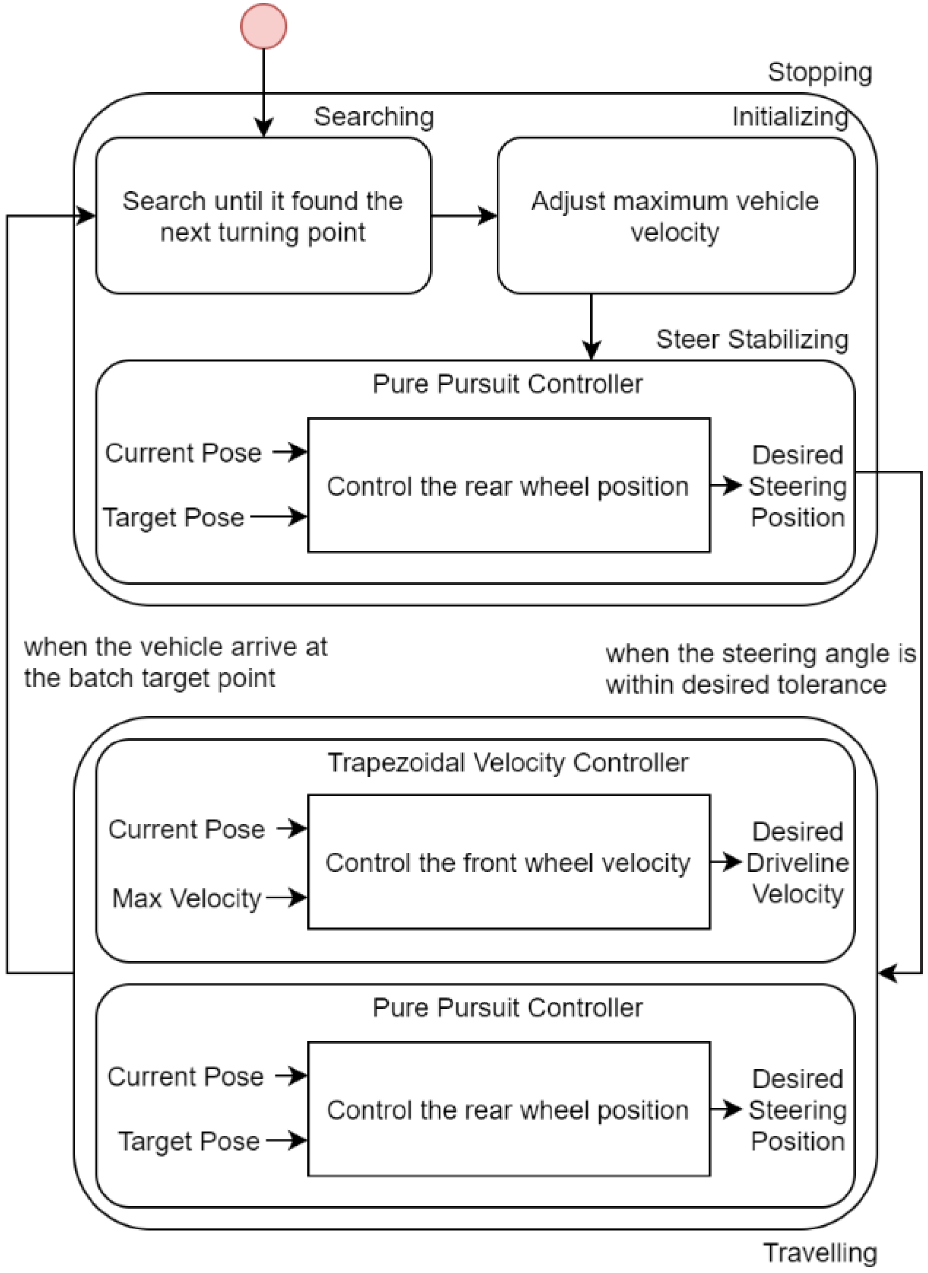


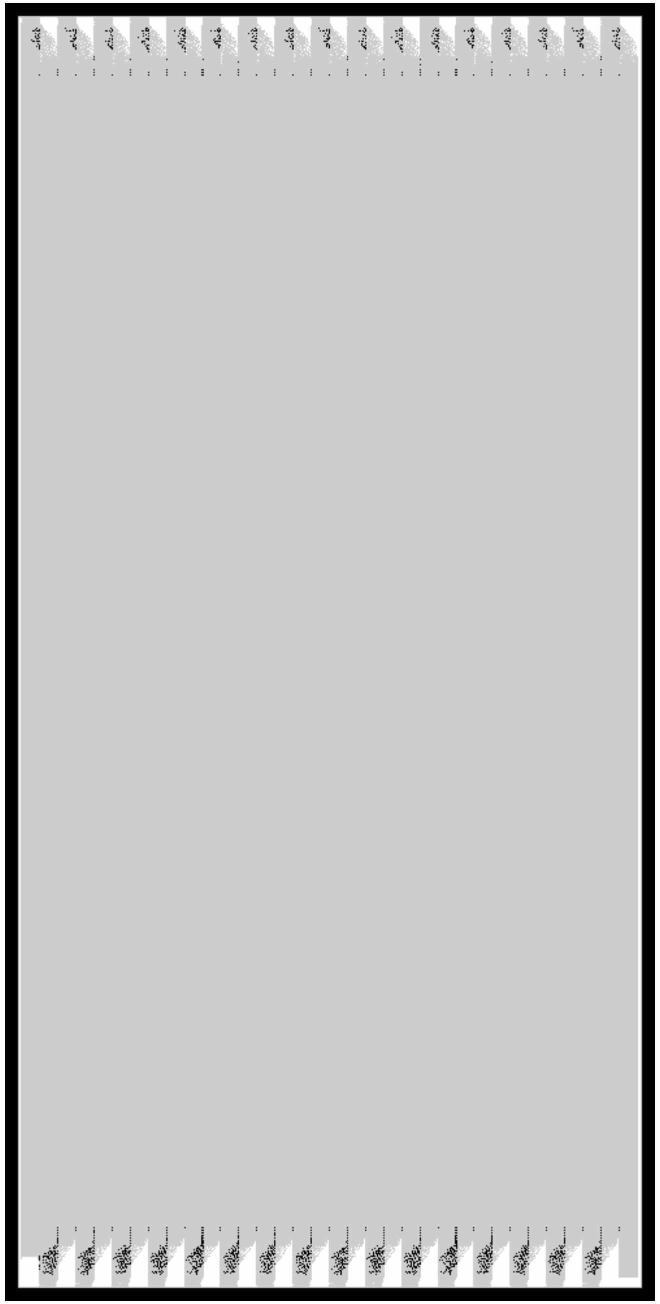
source: R. C. Conlter, "Implementation of the Pure Pursuit Path Tracking Algorithm," Carnegie Mellon, Pennsylvania, 1992.

Supervisor

A state machine with 4 states

1) Searching
2) Initializing
3) Steer Stabilizing (Optional)
4) Travelling





Predicted coverage from the planner

Results

Path Planner

Area Coverage: 97.3%

Path Tracker

Area Coverage: 95.8%

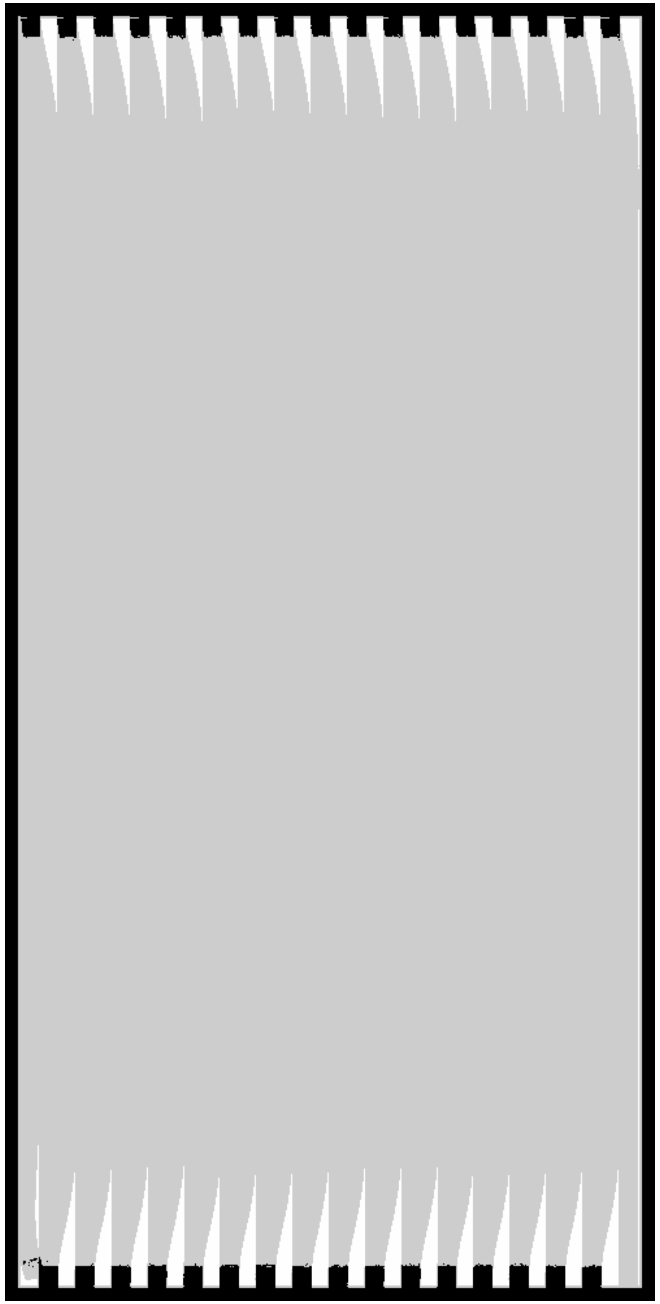
Time Consumed: 34 min.

Table I: Planner Configurations

Parameter Name	Value
Step Size	0.2 meters
Overlapping Number	1
Maximum Steering Angle	0.5 radians
Sliding Stop Threshold	50
Straight Pausing Threshold	2
Stopping Bound	1.5 meters
Initial Position	(-46.5, 23.8)

Table II: Tracker Configurations

Parameter Name	Value
Minimum Velocity	0.1 meters/second
Maximum Velocity	1.0 meters/second
Maximum Acceleration	0.1 meters/second^2
Lookahead distance	5.0 meters
Distance Tolerance	0.2 meters



Actual coverage from the tracker

Conclusion

- The foundation of Complete Coverage Navigation could be formed from existing simple algorithms.
- This **Path Planner** missed 2.7% of the farm-field area.
- This **Path Tracker** missed 1.6% of the planned area while its travelling is within 34 minutes.

Thank you for your attention

Q&A Session

For further inquiries, contact me at

norawitn@hotmail.com

Download
Slides