March 23rd (T) YY, JHL.

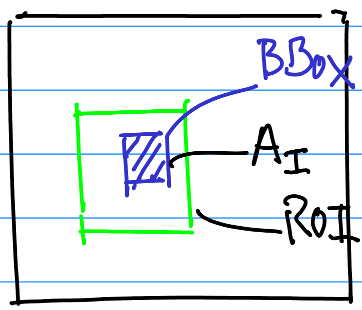
1. Demo { a Photo/Image
b Video

~ roi.py — main Program
~ Box.py
Bounding

Testing Program : rectangle_intersection.py.

Todo : Send readme.txt,

Threshold $T = 0.25$




Area (Intersection)
Blue

$A_I$ v.s. $T$

$A_I \geq T \rightarrow$ Alarm
$A_I < T$ No Action

Define 2 Thresholds { Upper Threshold $T_u$
Lower Threshold $T_L$

$A_I$

$T_u \simeq 50\%$

} $A_I$ Interested

$T_L \simeq 10\%$

} Discarded

$A_I$ is not to be Considered

Discard $A_I$ if it is Bigger than $T_u$

PART II
CTI March 24 (Wed)

1/

DRL - SAC

Soft Actor Critic Algorithm & Apps.
UC Berkeley & Google Pp. 4

Standard RL Objective to maximize Reward

$$\max \left\{ \sum_t E_{(s_t, a_t) \sim \rho_\pi} \left[ r(\vec{s}_t, \vec{a}_t) \right] \right\} \quad \cdots (1)$$

To Learn Policy

$$\pi(\vec{a}_t \mid \vec{s}_t) \quad \cdots (2)$$

to maximize Eqn (1).

Augment Eqn (1) with Entropy to maximize entropy at each visited State.

So

$$\alpha \mathcal{H}(\pi(a_t \mid s_t)) \quad \cdots (3)$$

Weight
Entropy Function
Independent Variable $\pi$ Policy

("Temperature" parameter determines the relative importance of Entropy term)

Hence Reward in Eqn(1) becomes

$$\max\left\{\sum_t E_{(s_t, a_t)\sim\rho_\pi}\left[r(\vec{s}_t, \vec{a}_t)\right] + \alpha \mathcal{H}\left(\pi(a_t|s_t)\right)\right\}$$

Soft Policy Iteration

Q-value (Reward)

$$Q: S \times A \to \mathbb{R} \quad \ldots (1)$$

Discount factor

$$T^\pi Q(\vec{s}_t, \vec{a}_t) \triangleq r(\vec{s}_t, \vec{a}_t) + \gamma E_{\vec{s}_{t+1}\sim p}\left[V(\vec{s}_{t+1})\right] \quad \ldots (1*)$$

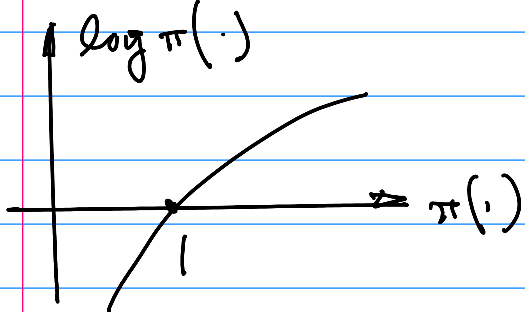
Bellman Operator for Each Action leads to Reward

Reward Function
Q-Value function Computed when $\pi(\vec{s}_{t+1}|\vec{a}_{t+1})$ is applied and reward is generated

where

$$V(\vec{s}_{t+1}) = E_{\vec{a}_t\sim\pi}\left[Q(\vec{s}_t, \vec{a}_t) - \alpha\log\pi(\vec{a}_t|\vec{s}_t)\right] \quad \ldots (2)$$

Policy introduced Q-Value (Reward)




Note: Based on (2)
$\alpha\log\pi(\vec{a}_t|\vec{s}_t)$, so $\pi(\vec{a}_t|\vec{s}_t)$ is Reward Based Function

Lemma 1. Bellman Operator T

"0" initial time t=0.

Given initial Condition $Q^0: S \times A \to \mathbb{R}$ with $|A| < \infty$

Define Bellman Operator T as

"All" total Number of finite Actions

$$Q^{k+1} = T^\pi Q^k \quad \ldots (3)$$

Part II

CTI



$$Q^{k+1} = T^{\pi} Q^{k}$$

Time Index

Q-Value (Reward) Function

Bellman Operator under Policy $\pi$

For $k=0$, we have

$$Q^1 = T^{\pi} Q^0$$

$k=1$, $Q^2 = T^{\pi} Q^1 = T^{\pi}(T^{\pi} Q^0)$

$$= (T^{\pi})^2 Q^0 \quad \ldots (4)$$

...

$k=i$ $\quad Q^{i+1} = (T^{\pi})^{i+1} Q^0 \quad \ldots (5)$

$\begin{cases} \text{update Q-Value (Reward)} \\ \text{update Policy } \pi \end{cases}$

Note: 1° "Argmin" find minimum from a collection of elements, functions, etc.

Example $\text{Argmin}\{1, -1.2, -110, 212\}$
$= -110$.

2° $\pi(\vec{a}_t | \vec{s}_t)$ Policy

$\pi(\cdot | \vec{S}_t)$ All polices for each/every $a_t$ from $\vec{a}_t$.

$\pi'(\cdot | \vec{S}_t)$ one selected from the All polices (all action, e.g. "$\cdot$" for $\vec{a}_t$)

$$\pi_{new} = \underset{\pi' \in \Pi}{\text{argmin}}\, D_{KL}\left(\pi'(\cdot | \vec{S}_t) \,\middle\|\, \frac{e^{\frac{1}{\alpha} Q^{\pi_{old}}(\vec{S}_t | \cdot)}}{Z^{\pi_{old}}(\vec{S}_t)}\right) \quad \ldots (6)$$

Note: 1° $\pi \in \Pi$

Policy from (belongs to) all collections of the Polices

Collection of All Polices

2° $D_{K.L.}$ KL: Kull Back—Leibler Divergence

3° $\pi'$ One $\pi$ realization from the collection of All Polices, A particular Policy

# Part II
# CTI



Generalize Eqn(6), we have

$$\pi_{New} = f(\pi_{old} \text{ from one of the all possible Polices} \mid \text{Conditions}) \quad \ldots (6b)$$

f to be minimization problem, so

$$\pi_{New} = \arg\min D_{KL}(\pi_{old} \text{ from One of the all polices} \mid \text{Condition}) \quad \ldots (6c)$$

Probability or likelihood

Note: Parameterized Policy as Gaussian Distribution

$$\frac{e^{\frac{1}{\alpha} Q^{\pi_{old}}(\vec{S_t}, \cdot)}}{Z^{\pi_{old}}(\vec{S_t})} \quad \ldots (7)$$

Note: In NN Softmax Activation Function we map Neuron Output in $(-\infty, +\infty)$ to Probabilistic distribution $[0,1]$ e.g.

$$f: z \in (-\infty, +\infty) \rightarrow f(z) \in [0,1]$$

where

$$f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \quad \ldots (8)$$

Lemma 2.

$$Q^{\pi^*}(\vec{S_t}, \vec{a_t}) \geq Q^{\pi}(\vec{S_t}, \vec{a_t})$$

where $\pi^*$ is from Eqn(6)

Policy $\pi \in \Pi$, under Eqn(6) converges to $\pi^*$

Note: Eqn(6) needs Better Explanation as why it is formulated as $D_{KL}$ minimization Problem.

Now, introduce parameterized Q function, and Policy $\pi$,

$Q_\theta$, $\pi_\phi$, $\theta$ (Theta), $\phi$ (phi) are

Theta phi

$\phi$: Policy parameters

$$J_Q(\theta) = E_{(\vec{s}_t,\vec{a}_t)\sim D}\left[\frac{1}{2}\left(Q_\theta(\vec{s}_t,\vec{a}_t) - \underbrace{\left(r(\vec{s}_t,\vec{a}_t) + \gamma E_{\vec{s}_{t+1}\sim p}[V_{\bar\theta}(\vec{s}_{t+1})]\right)}_{\text{Eqn (1*) Bellman Iteration}}\right)^2\right] \quad \ldots (9)$$

Objective Function

$Q(\theta)$ Q-value (Reward) Function with Parameter $\theta$ (Theta)

$$\nabla J_Q(\theta) = E_{(\vec{s}_t,\vec{a}_t)\sim D}\left\{\left[Q_\theta(\vec{s}_t,\vec{a}_t) - \left(r(\vec{s}_t,\vec{a}_t) + \gamma E_{\vec{s}_{t+1}\sim p}[V_{\bar\theta}(\vec{s}_{t+1})]\right)\right] \cdot \nabla Q_\theta(\vec{s}_t,\vec{a}_t)\right\} \quad \ldots (10)$$

$$J_\pi(\phi) = E_{\vec{s}_t\sim D}\left[E_{\vec{a}\sim\pi_\phi}\left[\alpha\log(\pi_\phi(\vec{a}_t|\vec{s}_t)) - Q_\theta(\vec{s}_t,\vec{a}_t)\right]\right] \quad \ldots (11)$$

$$J_\pi(\phi) = E_{\vec{s}_t\sim D}\left[E_{\vec{a}_t\sim\pi_\phi}[\cdot]\right]$$

For all states  For all actions $\vec{a}_t$

$\pi(\vec{a}_t, \vec{s}_t)$ Eqn(1), pp. 2

Note:

$$\frac{d}{dx}\ln(x) = \frac{1}{x}, \quad \frac{d}{dx}\log_a(x) = \frac{1}{x}\frac{1}{\ln(a)} \quad \ldots (12)$$

$$\vec{a}_t = f_\phi(\epsilon_\pi; \vec{s}_t) \quad \ldots (13)$$ Parameterized Policy? Action

Action Vector

multi-dimensional

$\vec{a}_t = (a_{1t}, a_{2t}, \ldots, a_{Nt})$

$$J_\pi(\phi) = E_{\vec{s}_t\sim D, \epsilon_t\sim N}\left[\alpha\log\pi_\phi(f_\phi(\epsilon_t;\vec{s}_t)|\vec{s}_t) - Q_\theta(\vec{s}_t, f_\phi(\epsilon_t;\vec{s}_t))\right] \quad \ldots (14)$$

Parameter for $\pi$

$\pi_\phi\left(f_\phi(\epsilon_t|\vec{s}_t)|\vec{s}_t\right)$ is from

$\pi_\phi\left(\vec{a}_{\phi t}|\vec{s}_t\right)$, or is from

$\pi_\phi(\vec{a}_t|\vec{s}_t)$ is from Eqn(2) pp.2.

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \alpha \log\left(\pi_\phi(\vec{a}_t|\vec{s}_t)\right) + \left(\nabla_{\vec{a}_t} \alpha \log\left(\pi_\phi(\vec{a}_t|\vec{s}_t)\right) - \nabla_{\vec{a}_t} Q(\vec{s}_t, \vec{a}_t)\right) \nabla_\phi f_\phi(\epsilon_t; \vec{s}_t) \quad \cdots (15)$$

Note: Jacobian

Suppose $\vec{f}: \mathbb{R}^n \to \mathbb{R}^m$

$$\vec{x} \in \mathbb{R}^n, \ \vec{f}(\vec{x}) \in \mathbb{R}^m \quad \cdots (16a)$$

Example: $f(x_1, x_2) = f(\vec{x}), \ \vec{x} \in \mathbb{R}^2$

Then $\vec{x} = (x_1, x_2)$

$$\vec{f}(\vec{x}) = \left(f_1(\vec{x}), f_2(\vec{x})\right), \quad \cdots (16b)$$

$$\vec{f}(\vec{x}) \in \mathbb{R}^2. \quad \cdots (16c)$$

Jacobian: Matrix of All its 1st order Partial Derivatives.

$$\vec{J} \triangleq \left[\frac{\partial \vec{f}}{\partial x_1}, \cdots, \frac{\partial \vec{f}}{\partial x_n}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

gradient of $f_1(\vec{x})$ $\cdots (16d)$

$\nabla f_1 =$

$$\nabla f_1(\vec{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} \\ \vdots \\ \frac{\partial f_1}{\partial x_n} \end{bmatrix} \quad \cdots (16e)$$

$$\nabla^T f_1(\vec{x}) = \left(\frac{\partial f_1}{\partial x_1} \ \cdots \ \frac{\partial f_1}{\partial x_n}\right) \quad \cdots (16f)$$

Hence, Jacobian

$$\vec{J} = \begin{bmatrix} \nabla^T f_1 \\ \vdots \\ \nabla^T f_m \end{bmatrix} \quad \cdots (16g)$$

Note: $\nabla f_1(\vec{x})$, if $f(\vec{x})$ changed to $f(\vec{x}; \phi)$

$\nabla_\phi f_1(\vec{x};\phi)$, so we are taking Partial derivative wrt $\phi$.

$\phi = (\phi_1, \phi_2, \cdots, \phi_n)$, $\vec{\phi}$ is written Simplified as $\phi$.

Automating Entropy Adjustment

$$\max_{\pi_{0:T}} E_{\rho_\pi}\left[\sum_{t=0}^{T} r(\vec{s}_t, \vec{a}_t)\right] \cdots (17)$$

$$\text{s.t. } E_{(\vec{s}_t, \vec{a}_t)\sim\rho_\pi}\left[-\log(\pi_t(\vec{a}_t|\vec{s}_t))\right] \geq \mathcal{H}, \forall t \cdots (18)$$

a. $r(\vec{s}_t, \vec{a}_t)$ : reward function

b. $\sum_{t=0}^{T} r(\vec{s}_t, \vec{a}_t)$ : Summation of reward function from $t=0$ to $t=T$;

c. $\max\left[\sum_{t=0}^{T} r(\vec{s}_t, \vec{a}_t)\right]$

maximize the reward function from $t \in [0, T]$ Period.

d. $\max E\left[\sum_{t=0}^{T} r(\vec{s}_t, \vec{a}_t)\right]$

maximize expected reward function from $[0, T]$

Now Add Notation to detail it up

$$\max_{\pi_{0:T}} E_{\rho_\pi}\left[\sum_{t=0}^{T} r(\vec{s}_t, \vec{a}_t)\right]$$

$\pi_{0:T}$ Policy $\pi$ from time 0 to T.

$\rho_\pi$ under Policy, Policy $\pi$

So, we have Eqn (17).

$$-\log(\pi_t(\vec{a}_t|\vec{s}_t)) = \log\frac{1}{\pi_t(\vec{a}_t|\vec{s}_t)}$$

PART II

CTI



Dynamic Programming, Solving for the policy backward through time

See Ref. pp. 7 "SAC" Paper

Rewrite Objective Function, Eqn (17)

$$\max_{\pi_0} \left\{ E[r(\vec{s}_0, \vec{a}_0)] + \max_{\pi_1} \left\{ E[\ldots] + \max_{\pi_T} E(r(\vec{s}_t, \vec{a}_t)) \right\} \right\} \cdots \quad (18)$$

$$\max_{\pi_0} \left\{ E[r(\vec{s}_0, \vec{a}_0)] + \max_{\pi_1} \left\{ E[r(\vec{s}_1, \vec{a}_1)] + \max_{\pi_2} \left\{ E[r(\vec{s}_2, \vec{a}_2)] \right\} \right\} \right\} \text{ for } t=2$$

$$\max_{\pi_2} \left\{ E[r(\vec{s}_2, \vec{a}_2)] + \max_{\pi_3} \left\{ E[r(\vec{s}_3, \vec{a}_3)] \right\} \right\} \text{ for } t=3$$

$$\max_{\pi_3} \left\{ E[r(\vec{s}_3, \vec{a}_3)] + \max \left\{ E[r(\vec{s}_4, \vec{a}_4)] \right\} \right\} \text{ for } t=4$$

...

Note :

1° Stochastic Gradient

$\hat{\nabla}_\theta J_Q(\theta)$ from Eqn (6) Ref. SAC.

2° Policy Gradient

$\hat{\nabla}_\phi J_\pi(\phi)$ from Eqn (10), Ref

Ref. Fujimoto, 2018

21-Dimensional Humanoid

3° Single Q function $\Rightarrow$ 2 Soft Q-function

Speed up training.

SAC ref.

Algorithm 1. (pp. 8)

Input: $\theta_1, \theta_2, \phi$ — $\theta_{1,2}$ for Reward, $\phi$ for Policy $\pi$

$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$

$\mathcal{D} \leftarrow \emptyset$ — Replay Pool

for each iteration do

for environment step do

$\vec{a}_t \sim \pi_\phi(\vec{a}_t | \vec{s}_t)$ — Sample action from Policy

$\vec{s}_{t+1} \sim p(\vec{s}_{t+1} | \vec{s}_t, \vec{a}_t)$ — Sample Transition from Enviro.

$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\vec{s}_t, \vec{a}_t, r(\vec{s}_t, \vec{a}_t), \vec{s}_{t+1})\}$ — Store transition in Replay pool

end for

for each gradient step do

$\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ — update Q-func Parameter

$\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ — update policy weights

$\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ — update temperature

$\bar{\theta}_i \leftarrow \tau \theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$ — Update network Weights

end for

end for

Output: $\theta_1, \theta_2, \phi$

Note: 4 Dual Objectives $\longrightarrow$ Dual Gradient Descent Alternating Between Lagrangian.

$$\alpha_t^* = \arg\min_{\alpha_t} E_{\vec{a}_t \sim \pi_t^*}[\cdot] \quad \dots (17)$$

5. "Off-Policy" Data? Ref.

Path planning with Reward Function
HL

1/

April 2nd, 2021

# Reward Functions Based On Difference For Autonomous Driving

- Micro-Scale: $\vec{P}_p$ to $\vec{P}_q$ within One Image $I(x,y)$
- Medium Scale: Within One Room.
- Global Scale: One Environment Multiple Rooms.

## Micro-Scale

1. Within One Image $I(x,y)$

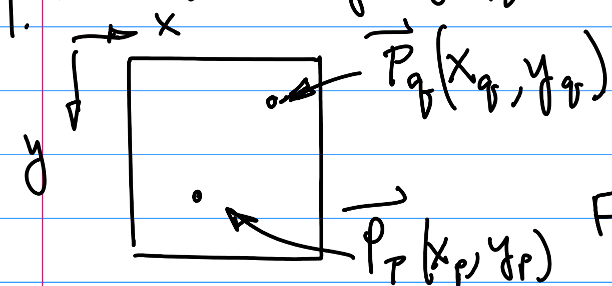



Fig.1

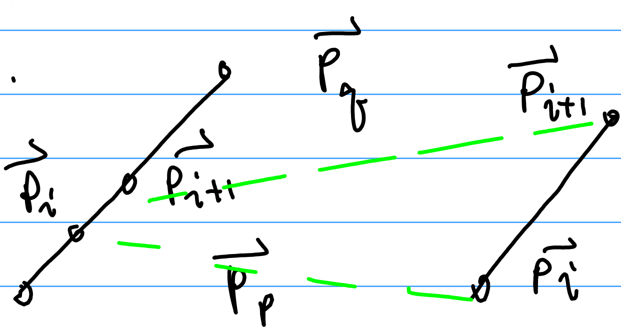
2. The Path from $\vec{P}_p$ to $\vec{P}_q$ is accomplished step by step starting from $\vec{P}_p$, then update each step $\vec{P}_i$ to $\vec{P}_{i+1}$, ..., $i=1,2,...,N$

$\vec{P}_1 = \vec{P}_p$ (starting Point)

$\vec{P}_N = \vec{P}_q$ (ending Point)

3. $\vec{P}_i$ for $i=1,2,...,N$ is Computed Based on DDA Algorithm.

Fig.2




a. $\vec{P}_i$ to $\vec{P}_{i+1}$ is one pixel equivalent distance in the world coordinate $(X_w\text{-}Y_w\text{-}Z_w)$

b. Actual AGV Travels during that step $i$ to $i+1$, e.g. time $t_i$ to $t_{i+1}$, is $\vec{P}_{ACT}$, or $\vec{P}_a(x_a, y_a)$

c. Case I Actual Displacement within $\vec{P}_{i+1}$

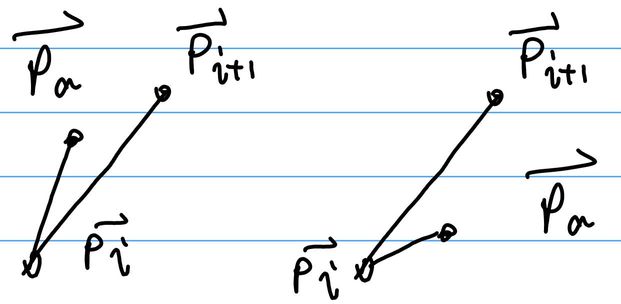



Fig 3

Case II Actual Displacement Outside $\vec{P}_{i+1}$

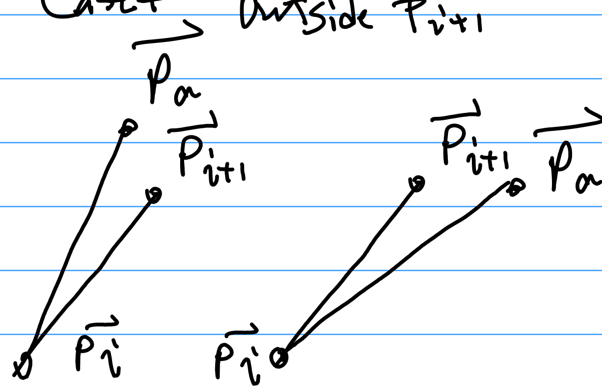



Fig 4.

Case III Opposite Directions as in Case I & II.

# Path planning with Reward Function

HL

2/

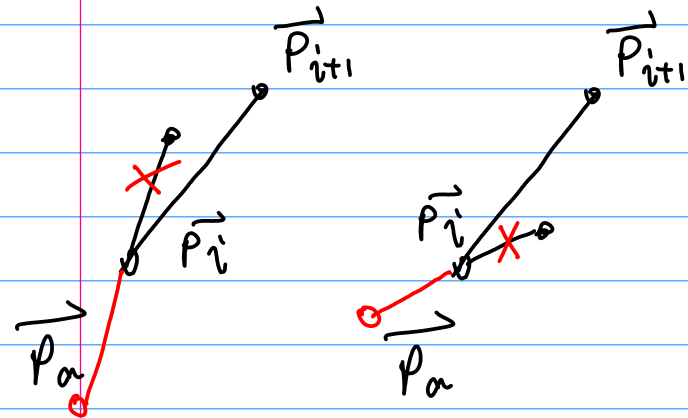


Fig 5a

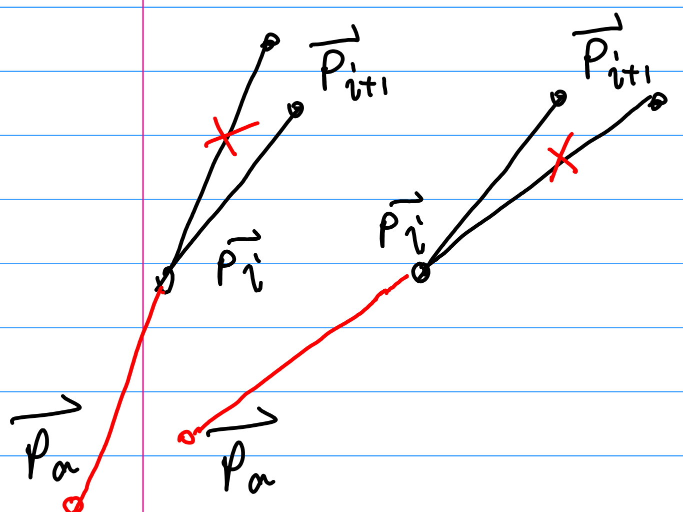


Fig 5b

4. Path Deviation $\vec{P_a}$ Calculation By Dot Product

$$(\vec{P_a}-\vec{P_i})\cdot(\vec{P_{i+1}}-\vec{P_i}) \quad \cdots (1)$$

where

Desired Direction Vector

$$\vec{d_d} \triangleq \vec{P_{i+1}} - \vec{P_i} = (x_{i+1}-x_i,\ y_{i+1}-y_i) \quad \cdots (2)$$

Actual Displacement Direction Vector

$$\vec{d_a} \triangleq \vec{P_a} - \vec{P_i} = (x_a - x_i,\ y_a - y_i) \quad \cdots (3)$$

5. Define Deviation Index Based on
   - a. Distance Difference
   - b. Directional Difference.

Note:

$$\vec{d_d}\cdot\vec{d_a} = \|\vec{d_d}\|\|\vec{d_a}\|\cos\alpha \quad \cdots (4)$$

Normalize it to make it an index

$$\frac{\vec{d_d}\cdot\vec{d_a}}{\|\vec{d_d}\|\|\vec{d_a}\|} = \cos\alpha \quad \cdots (4-1)$$

Property 1: if $\vec{d_a} = \vec{d_d}$ Displacement and distance

$$\|\vec{P_a}-\vec{P_i}\| = \|\vec{P_{i+1}}-\vec{P_i}\|$$

then, the AGV is on the right path Direction and has the exact desired displacement distance to make $\vec{P_a} = \vec{P_{i+1}}$

# Path Planning with Reward Function
## HL



~~Definition~~ Define 1. Direction Index $I_{dir}$, a scalar is defined as the angle $\alpha$ Between $(\vec{P_a}-\vec{P_i})$ and $(\vec{P_{i+1}}-\vec{P_i})$;

e.g.

$$I_{dir} = \cos\alpha = \frac{\vec{d_a}\cdot\vec{d_d}}{\|\vec{d_a}\|\|\vec{d_d}\|} \quad \ldots(5)$$

Define Distance index $I_{dis}$, as Scalar as

$$I_{dis} = \begin{cases} \dfrac{\|\vec{P_a}-\vec{P_i}\|}{\|\vec{P_{i+1}}-\vec{P_i}\|}, & \text{if } \|\vec{P_a}-\vec{P_i}\| \le \|\vec{P_{i+1}}-\vec{P_i}\| \quad \ldots(6a) \\[2ex] \dfrac{\|\vec{P_a}-\vec{P_i}\| - \|\vec{P_{i+1}}-\vec{P_i}\|}{\|\vec{P_{i+1}}-\vec{P_i}\|} & \text{if } \|\vec{P_a}-\vec{P_i}\| > \|\vec{P_{i+1}}-\vec{P_i}\| \quad \ldots(6b) \end{cases}$$

Note $I_{dis} \in [0,1]$; and $I_{dir} \in [0,1]$ $(\cos\alpha)$

Definition 2. Define Combo Index $I_\Sigma$ for AGV as

$$I_\Sigma = \beta I_{dir} + (1-\beta) I_{dis} \quad \ldots(7)$$

$$I_\Sigma = \beta I_{dir} + (1-\beta) I_{dis}$$

where $\beta \in [0,1]$

for Equal weights on direction and distance we have $\beta = \frac{1}{2}$. So

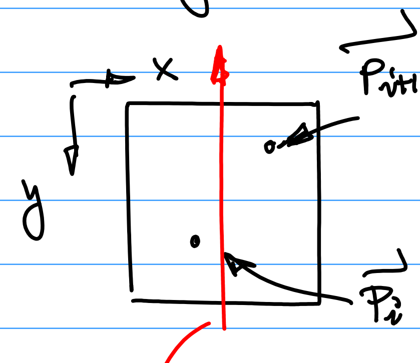
$$I_\Sigma = \frac{1}{2}(I_{dir} + I_{dis}) \quad \ldots(7a)$$

PART II. Consider Actuation.

Note 1. How to find $\vec{d_a}$? magnetometer find $\vec{d_a}$

Note 2. How to find $\vec{d_d}$

See Eqn (2), Computer Vision Technique plus $X_w$-$Y_w$-$Z_w$ mapping.



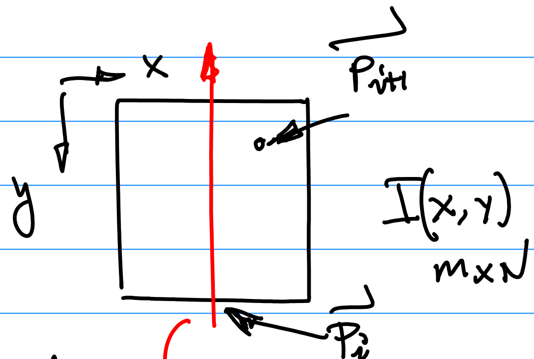

Vehicle Forward Looking Direction LSMSEN

a Physical world Calibration in $X_w$-$Y_w$-$Z_w$

LSMSEN Sensor Input

# Path planning with Reward Function
HL






1° Angle $\theta = 132.615°$ for Example (Theta), Vehicle forward direct Angle

ii° Red Line on $I(x,y)$ : Vehicle forward Looking Direction

iii. $\vec{P_i}$ Current time $i$ Position

For MxN Image, $\vec{P_i}(x,y) = I_m\left(\frac{M}{2}, N-1\right)$ ...(8)

M : Col.

N : Row

iv. mapping from $I(x,y)$ to $X_w$-$Y_w$-$Z_w$

$$f : (x,y) \in \mathbb{I}^2 \to (x_w, y_w, z_w) \in \mathbb{R}^3 \quad \text{...(9)}$$

Definition 4. Define AGV forward Looking Direction By LSMSEN Sensor Input, e.g., Equal to the Angle from LSMSEN.

Definition 5. Define AGV $\vec{P_i}$ Position as the $(x,y) = \left(\frac{M}{2}, N-1\right)$, for MxN $I(x,y)$.

Property 2. A feature Point $\vec{P}_{i+1}$ on Image $I(x,y)$'s location $(x_{i+1}, y_{i+1})$ is mapped to the $X_w$-$Y_w$-$Z_w$ By the mapping function $f$ as follows

$$f : (x,y) \in \mathbb{I}^2 \to (x_w, y_w, z_w) \in \mathbb{R}^3$$

such that

$$(x_{i+1}, y_{i+1}) \to (x_{i+1}, y_{i+1}, z_i)\Big|_{w,\ z_i = C}$$

e.g. for Each $(x_{i+1}, y_{i+1})$ we have

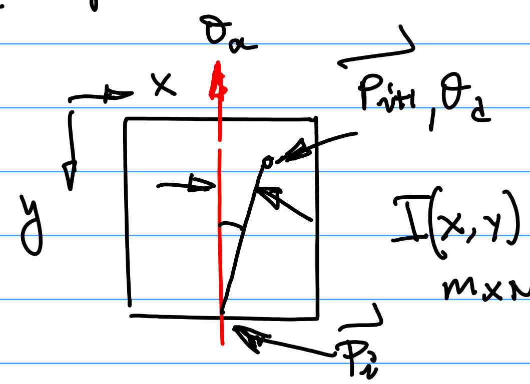
$$(x_{i+1}, y_{i+1}, C)_w$$

5. Driving objectives:

To align the Red line with $\vec{d}_{dir} = (\vec{P}_{i+1} - \vec{P}_i)$

How much to Drive?

α Angle $\theta_d - \theta_a = \Delta\theta$ to Be zero'd




distance to $\vec{P}_{i+1}$ to be zero'd.

| | map to $X_w$-$Y_w$-$Z_w$ | Result |
|---|---|---|
| $\theta_a$ | LSMSEN | — |
| $\theta_d$ | $f(\cdot)$ | $x_{i+1,w}$, $y_{i+1,w}$ |

from $x_{i,w}, y_{i,w}$ to $x_{i+1,w}, y_{i+1,w}$

- Leftwheel
  - Displacement, dis
  - Angle
- Rightwheel
  - Displacement, dis
  - Angle

6 DoF Robot Arm Reward function.

1. 3 Types Movement
   - Linear $\vec{P_i}$ to $\vec{P_{i+1}}$
   - Angular (Joint Angle)
   - Arc, $\vec{P_i}$ to $\vec{P_{i+1}}$ via Control pt. $\vec{P_c}$