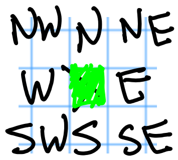
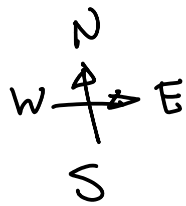
Driving Directions: "8- Connected Neighbors"





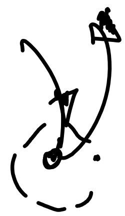


Fig.4

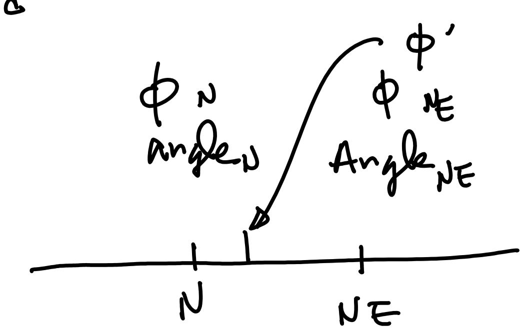
Find the Direction of Driving at Each step that in the end will minize the Objective Function in Eqn(9).

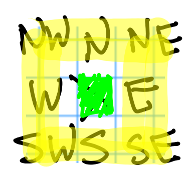
$$\text{Policy } \pi(\alpha_{k+1}, S_{k+1} \mid S_k) \to r_{k+1}$$

Action (8 Directions)

Reward

Action	Reward
N	$r_N = ?$
NW	$r_{NW} = ?$
W	$r_W = ?$
SW	⋮
S	
⋮	





Determine Reward Function Based ON Moving Direction of Shortest Path.

List of Possible moving Directions

1. From Fig.4. Only 5 possible Directions

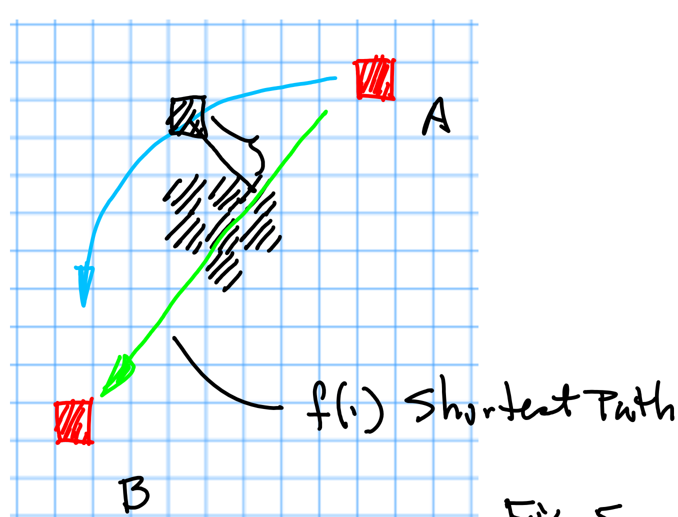


Fig.5

Example 1. 1. place 8-Direction Template (N, S, ...) on top of Point A. Use shortest path, Green Line, as a reference to Define A set of Reward function Based on the direction Matching Reward (DMR) = Policy

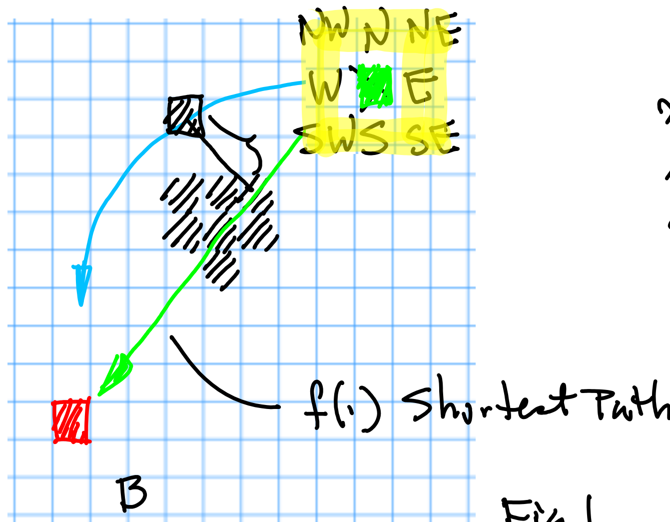


Fig. 6

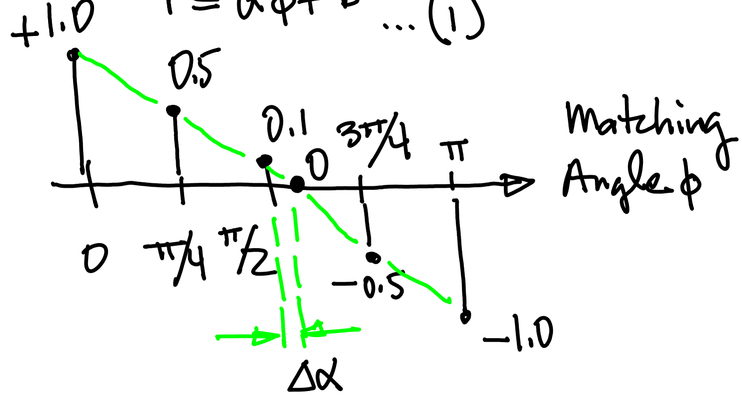
π_{DMR} : $\Delta = 0.5$

x-SW +1.0 Best Matching x-SW overlap
x-W +0.6 Next Best x-W Angle < π/2
x-S +0.6 " " x-S " < π/2
x-SE +0.1 x-SE " < π/2
x-E -0.1 Opposite x-E " > π/2

x-NW -0.1 Angle > π/2
x-N -0.6 Angle > 3π/4
x-NE -1.0 Angle ≃ π

Algorithm: Best Matching Direction. Highest "+" Reward
Worst matching Direction Smallest "-" Reward

$r = a\phi + b$ ...(1)



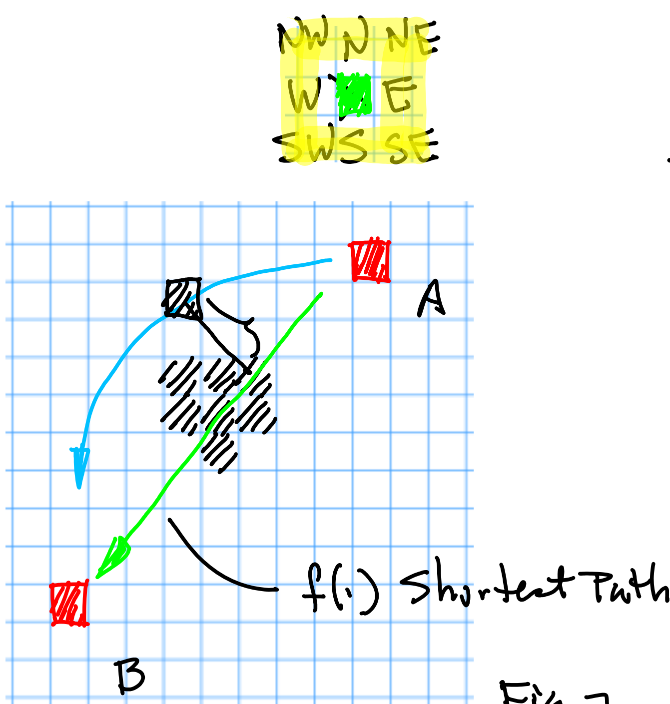


Fig. 7

Program Implementation:

1° Implement Reward Function (1).

Note: Angle φ is formed Betwee Blue line and green Line.

Compute Reward.

March 12 (Sat), 22

Step 1. Setup the System As in Fig 1., Fig.2 is the Abstract Form for Fig.1, So use Fig.2.

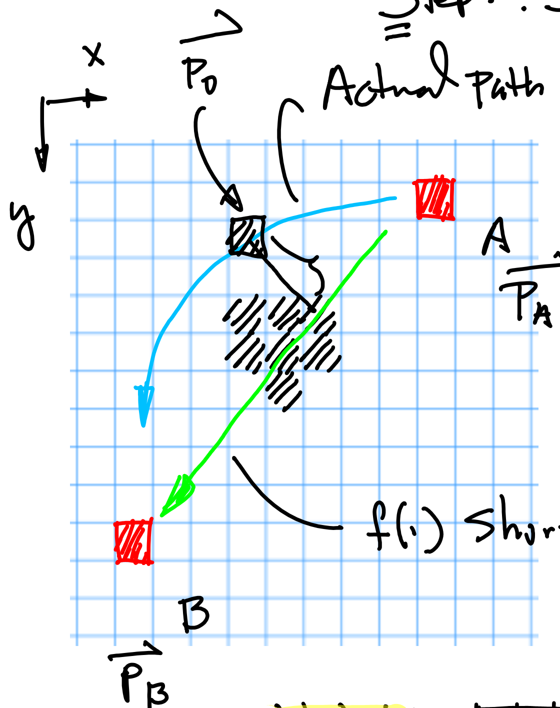


Fig 1.

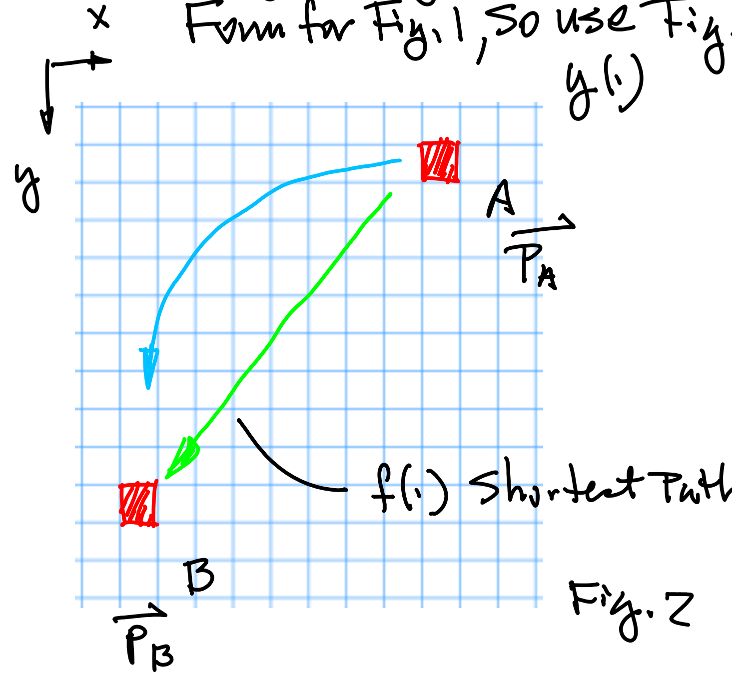


Fig. 2

Step 2.

Fig 2b. 8-Connected Neighbour.

Define Python module for the Angle Computation for 8-Connected Neighbours in Fig. 2b.

Step 3. Compute Angle Between green and Blue at location 1.

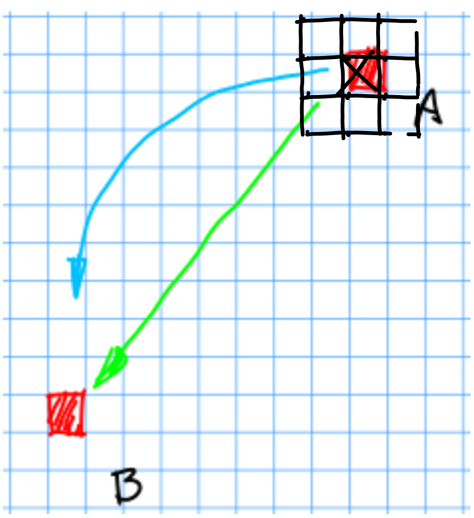


Fig.3

Compute Angle Between green and Blue at location 2.

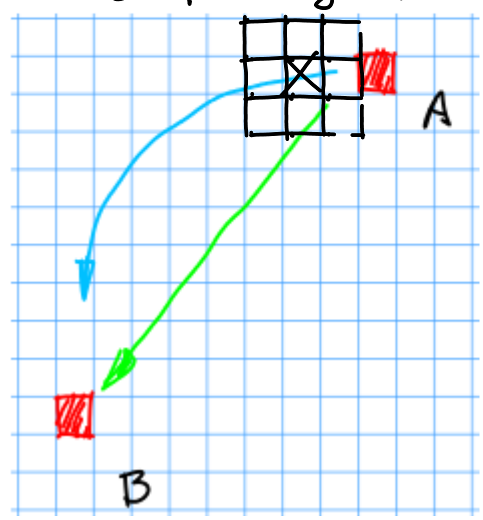


Fig.4

March12(Sat)22

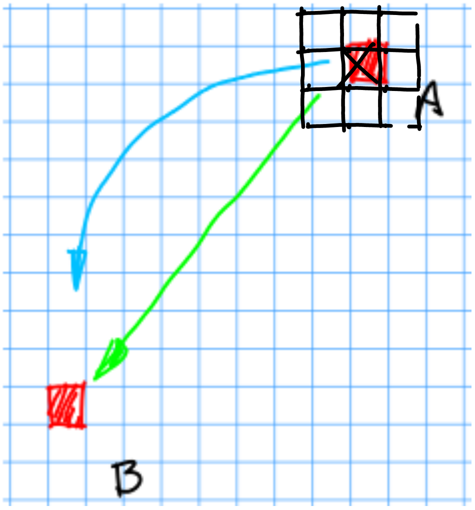
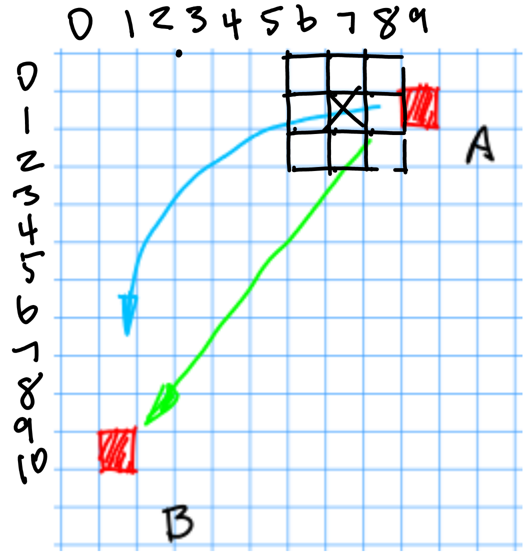


Fig.3



48.36°

Fig.4

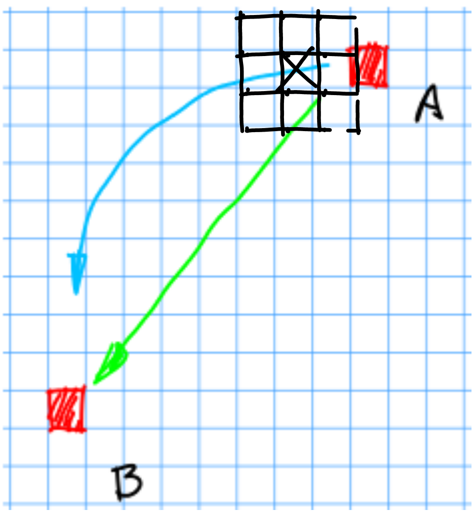


Fig5

Compute Angle Between green and Blue at location3.

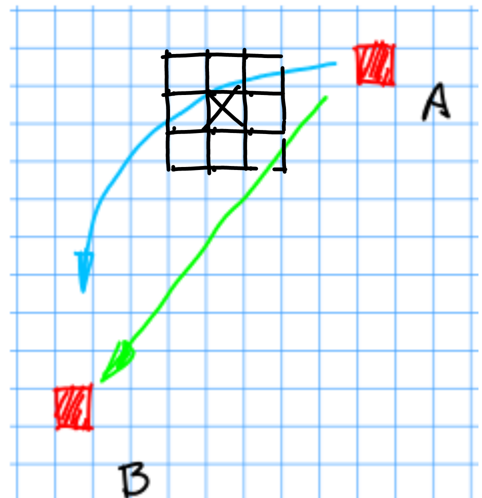


Fig.6

Compute Angle Between green and Blue at location4.

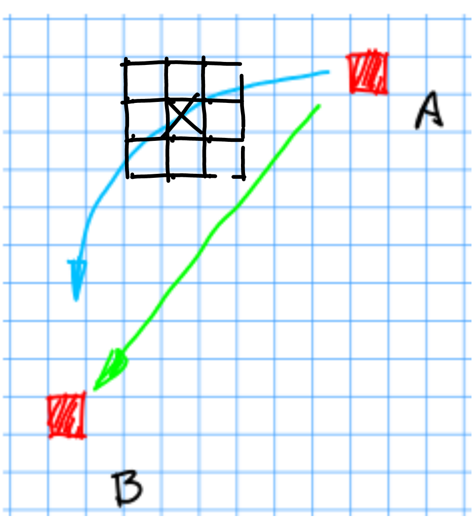


Fig7

Compute Angle Between green and Blue at location5.

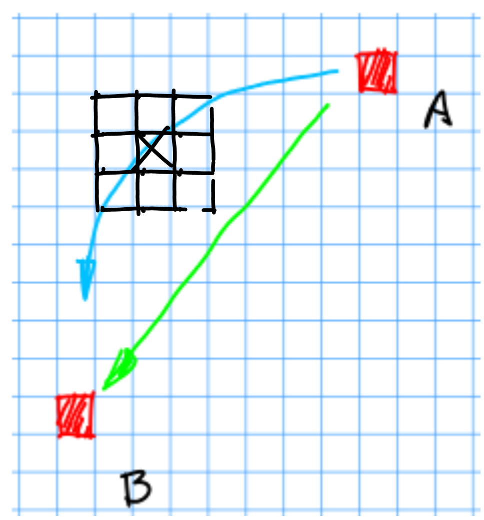


Fig. 8

Compute Angle Between green and Blue at location6.

March 12 (Sat), 22

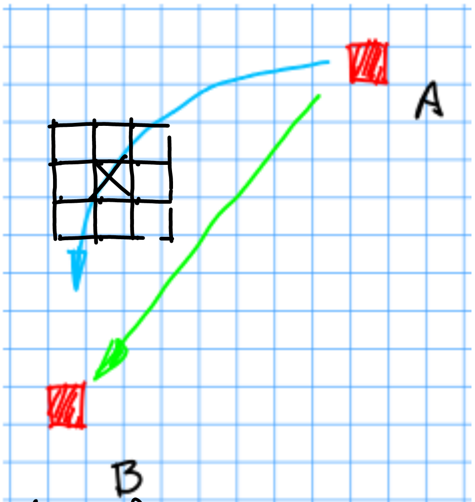


Fig 9

Compute Angle Between green and Blue at location 7.

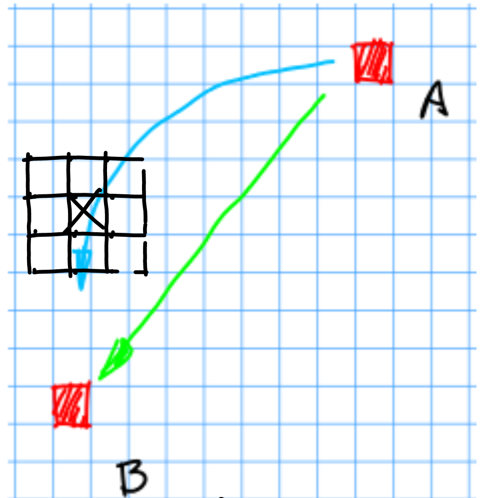


Fig. 10

Compute Angle Between green and Blue at location 8.

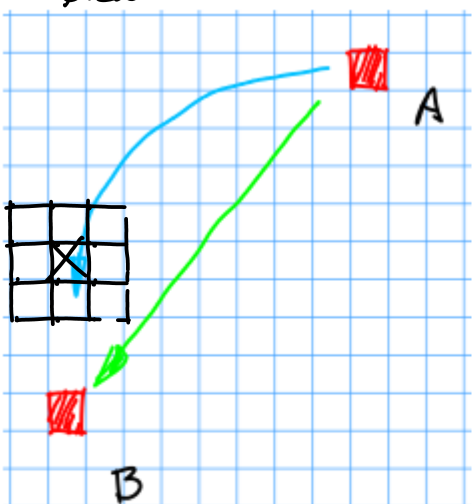


Fig 11

Compute Angle Between green and Blue at location 9.

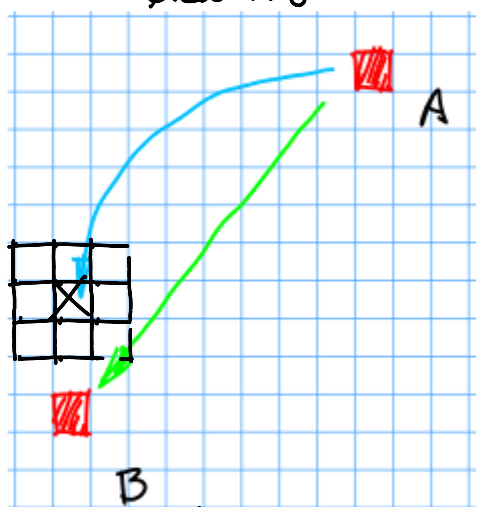


Fig. 12

Compute Angle Between green and Blue at location 10.

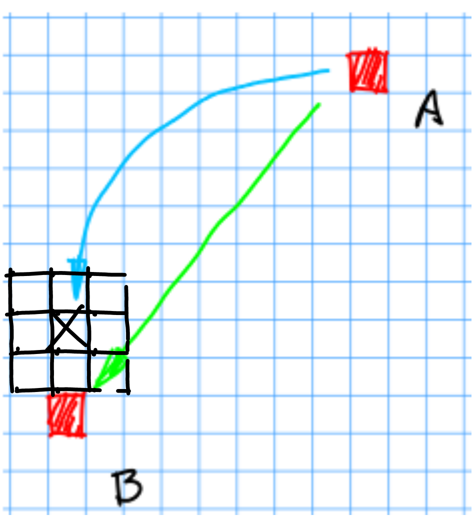


Fig 13

Compute Angle at location 11.

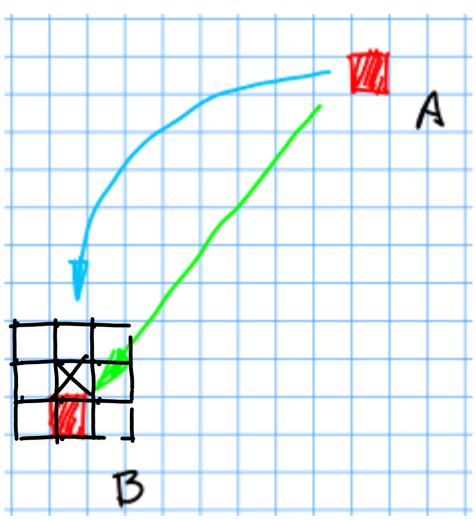


Fig. 14

Compute Angle at location 12.

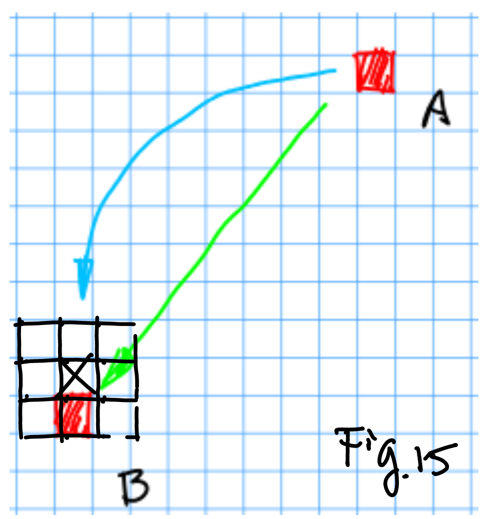


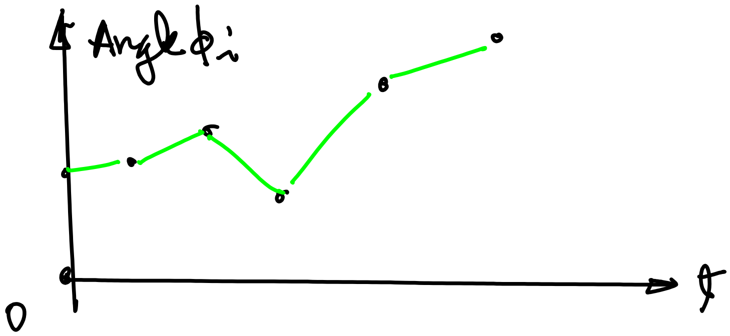
Fig. 15

Compute Angle at location 13.

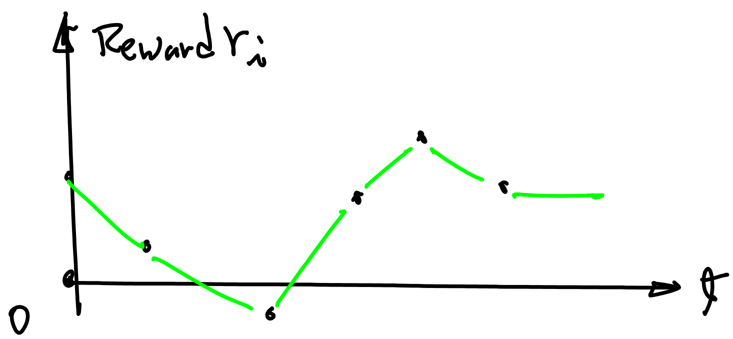
March 12 (Sat), 22

Step 4

Plot All the Angles $\phi_1, \phi_2, \cdots, \phi_i \cdots$ in the plot below, plot All Reward Function values $r_1, r_2, \cdots, r_i \cdots$ in the plot below. Then find Sum of all Rewards.



$$R_\Sigma = \sum_{i=1}^{N-1} r_i \quad \cdots (1)$$



March 13th (Sun) with B.P.

(End)

$\vec{x} \times \vec{y} = \vec{z}$

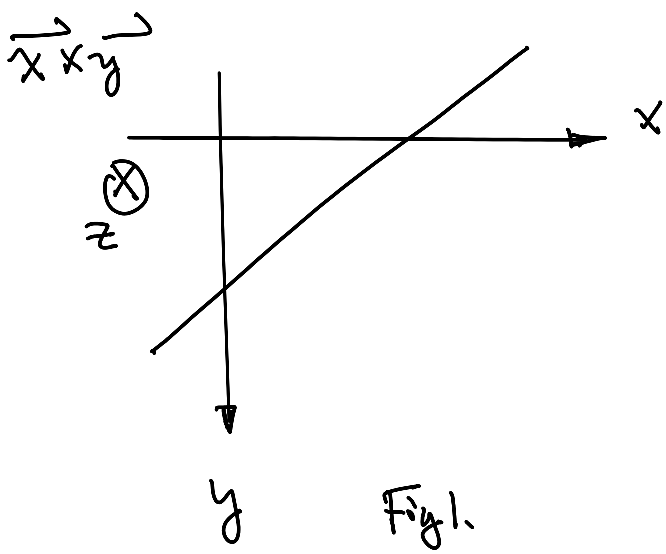
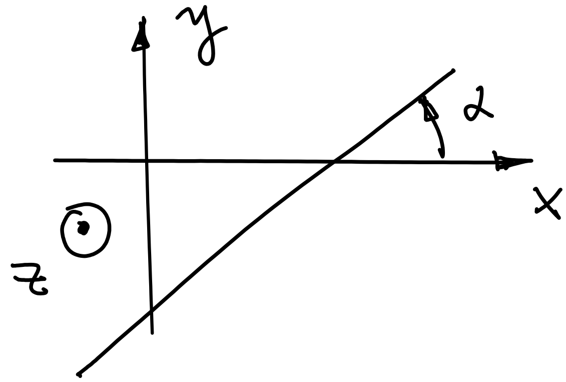


Fig 1.

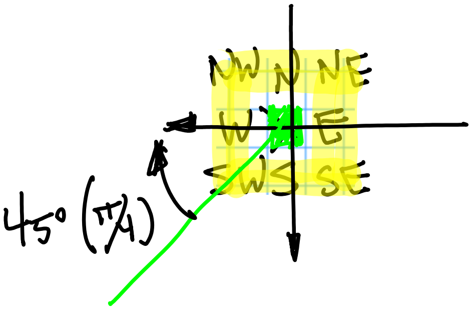


Guideline

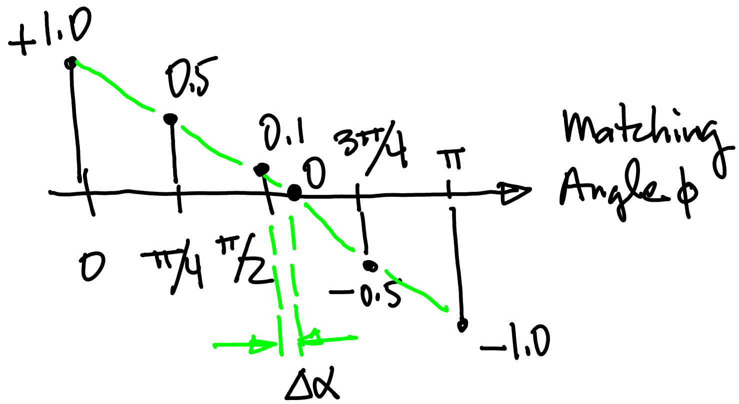
Fig. 2

No "World coordinate System" yet

7.



W.r.t Green Line Direction



π_{DMR} : $\Delta = 0.5$

α-SW	+1.0	Best Matching	α-SW	Overlap	
α-W	+0.6	Next Best	α-W	Angle	$< \pi/2$
α-S	+0.6	" "	α-S	"	$< \pi/2$
α-SE	+0.1		α-SE	"	$< \pi/2$
α-E	−0.1	Opposite	α-E	"	$> \pi/2$

α-NW	−0.1	Angle $> \pi/2$
α-N	−0.6	Angle $> 3\pi/4$
α-NE	−1.0	Angle $\simeq \pi$

March 14 (Monday), with YY, BP.

robotics-open_abb / aiv200 / 190g-deep-reinforcement-learning / **190g-3-6DoF-Action-State-Reward-SS-2021-03-17.pdf**

6 DoF Robot Unity

[How to train your Robot Arm?. Training a 6 axis robot arm using Unity… | by Raju K | XRPractices | Medium]

[rkandas/RobotArmMLAgentUnity: Training 6 axis robot arm Inverse kinematics using Unity ML Agents (github.com)]

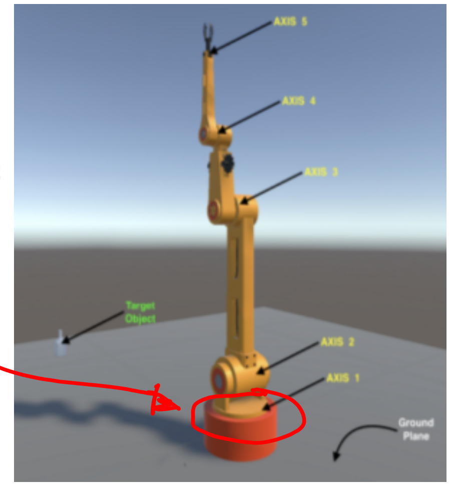
1. Actions: An array of actions – each action in the array represents the degree of rotation.
   We have 5 types of actions in total: 1 Rotate and 4 Bends.

   1.1. Axis 1: is the bottom-most axis and can rotate 0 to 360 degrees [Rotate]

```
armAxes[0].transform.localRotation =
Quaternion.AngleAxis(angles[0] * 180f, armAxes[0].GetComponent<Axis>().rotationAxis);
```

1.1. Axis 1: is the bottom-most axis and can rotate 0 to 360 degrees [Rotate]

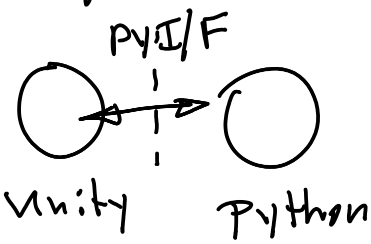
```
armAxes[0].transform.localRotation =
Quaternion.AngleAxis(angles[0] * 180f, armAxes[0].GetComponent<Axis>().rotationAxis);
```



C#
a. Physical model (Dimension)
b. Physics of the model. { Rotation direction, Angle }
c. Graphical model

Step 1. a–c : Move OpenCV model to Unity
Step 2. C# ML Interface

from CTI ONE model, And the implementation code is from CTI One Team, especially from Mr. Yusuke Yakuwa.



OpenCV { 1. Angle, 2. Index, 3. Reward (I) }

March16 (Wed)

1. Verification of YY's Implementation
2. Provide Hand Calculation.
① from start position to the end position.

	Position	Angle	Distance	Reward
$\vec{P}_A$	(9,1)	ϕ_i	~~d_i~~ L_i	$r(\phi_i, L_i)$
$\vec{P}_i$	(4,2)			
$\vec{P}_B$	(1,10)			

② Record Heuristic Motion, Path.txt

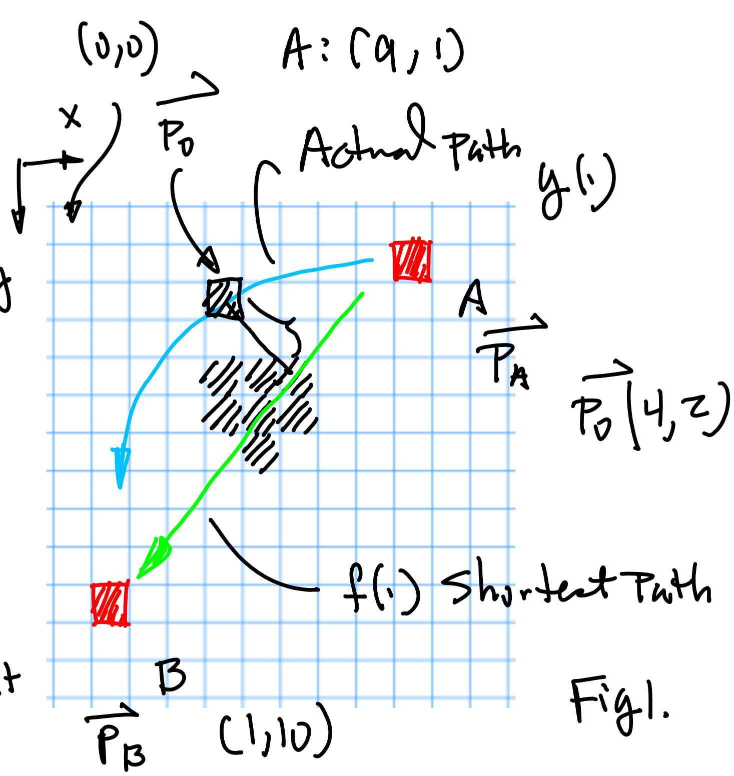


Fig1.

(3) Run GTI One Version 0.1 Code,
Make Comparison for Verification.

March 17 (Thursday) Hand Calculation

Step 1. Given initial condition
$A=(9,1)$, $B=(1,0)$
x-y Coordinate System Setup
as shown in the figure below.

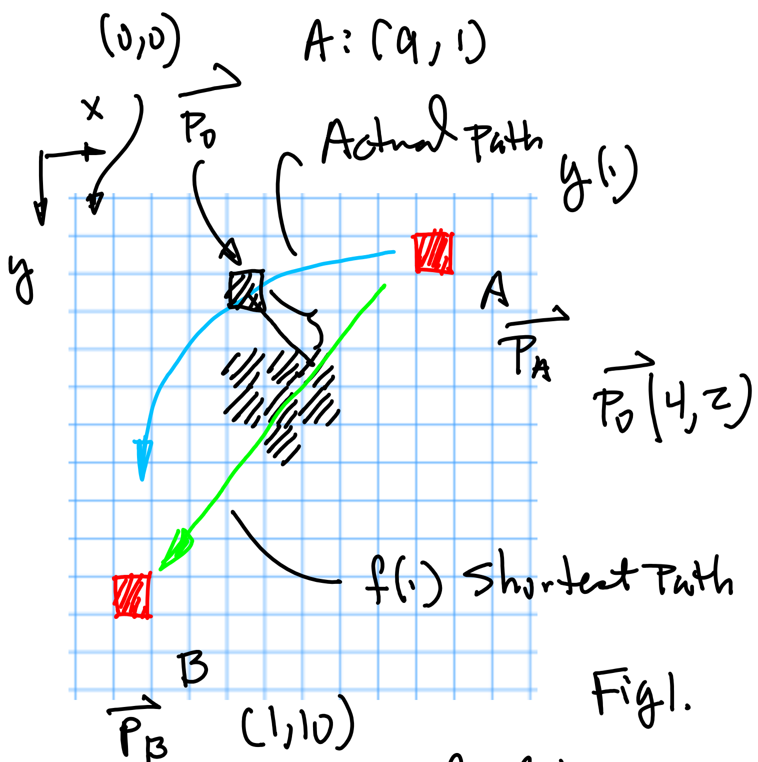


Fig 1.

Step 2. The Angle calculation
Formula:

$\vec{P_A}-\vec{P_i}=(x_a-x_i, y_a-y_i)$ ...(1) (Blue Line)

$\vec{P_A}-\vec{P_B}=(x_a-x_b, y_a-y_b)$ ...(2) (Green Line)

$(\vec{P_A}-\vec{P_i})\cdot(\vec{P_A}-\vec{P_B})=\|\vec{P_A}-\vec{P_i}\|\,\|\vec{P_A}-\vec{P_B}\|\cos\alpha$ ... (3)

Therefore, for Eqn (3), we have Eqn (4) below,

$$\cos\alpha=\frac{(\vec{P_A}-\vec{P_i})\cdot(\vec{P_A}-\vec{P_B})}{\|\vec{P_A}-\vec{P_i}\|\,\|\vec{P_A}-\vec{P_B}\|}=\frac{(x_a-x_i, y_a-y_i)\cdot(x_a-x_b, y_a-y_b)}{\sqrt{(x_a-x_i)^2+(y_a-y_i)^2}\sqrt{(x_a-x_b)^2+(y_a-y_b)^2}}=$$

$$\frac{(x_a-x_i)(x_a-x_b)+(y_a-y_i)(y_a-y_b)}{\sqrt{(x_a-x_i)^2+(y_a-y_i)^2}\sqrt{(x_a-x_b)^2+(y_a-y_b)^2}} \quad \text{... (4)}$$

Now, Calculation. Denote Robot Position as $\vec{P_i}(x_i, y_i)$,

for Postion $i = 1$, $\vec{P_1}(7,1)$ illustrated in Fig.4

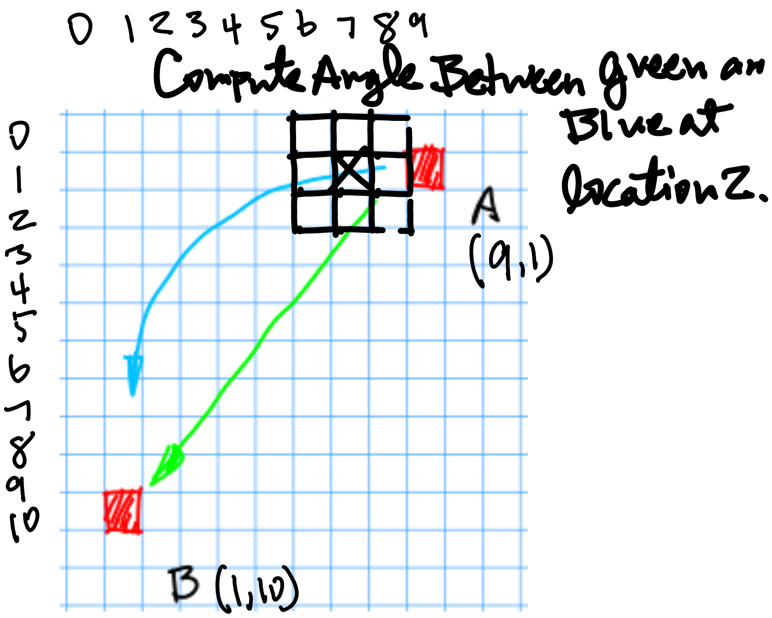


Fig.4

From Eqn (4), we have

$$\cos\alpha = \frac{(x_a - x_i)(x_a - x_b) + (y_a - y_i)(y_a - y_b)}{\sqrt{(x_a - x_i)^2 + (y_a - y_i)^2}\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}}$$