Final Report

August 2, 2017

1 Search Algorithm

1.1 Room Segment

The general function of the base board is moving to your appointed destination and return. To realize it, we cut the room into many small squares with coordinates. Then, we can tell the coordinates of the destination. In addition, since the coordinate system is absolute, Metbin also should know the coordinates of its initial place to calculate its relative position about the destination.

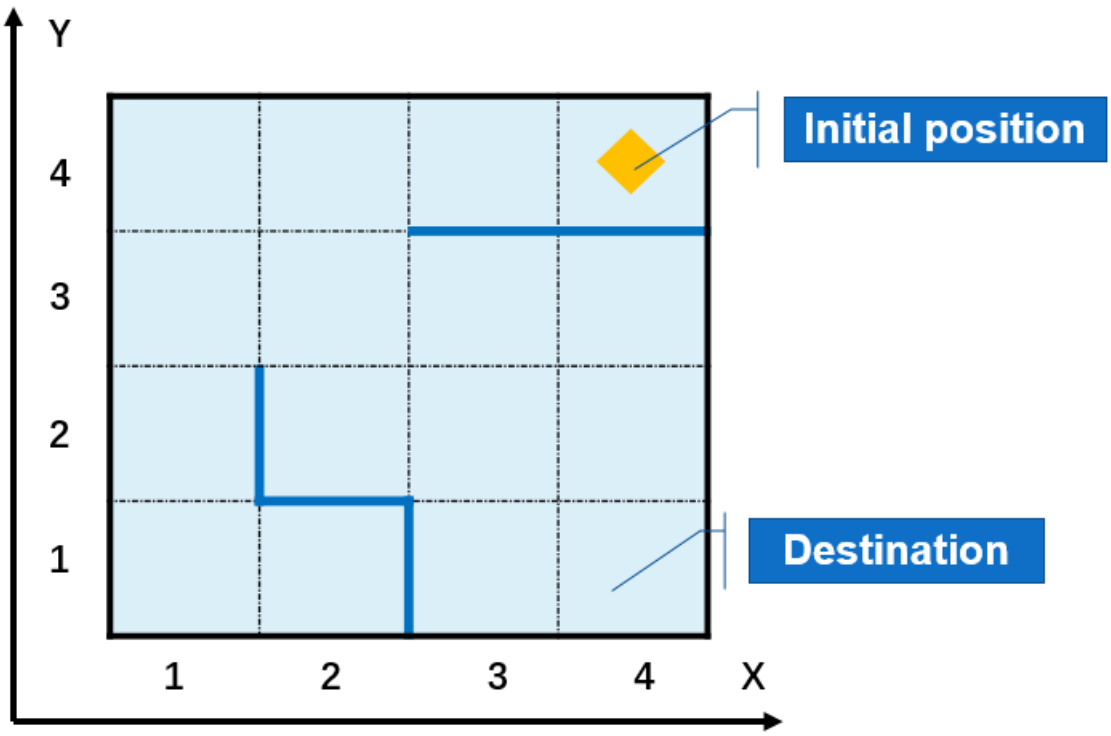

Figure 1: Room segment with coordinates

In the figure 1, the room is cut into 4×4 squares with coordinates, and there are x and y axis. So we can know the initial position of Metbin is (4,4) and the destination's coordinates are (4,1).

1.2 Direction of Metbin

Since we use a search algorithm, Metbin must know its direction relative to x and y axis, so it can know towards which square it's moving. Since the room is cut into squares, so we introduce a direction variable **d** of four values to represent robot's current direction.

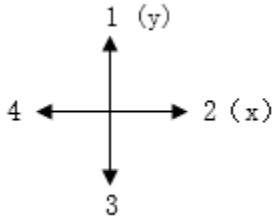

Figure 2: Direction variable

The figure shows the direction in the room. Since its also absolute, it should be input to the Metbin in advance. We set the direction parallel to y axis is 1, and it will become plus one after Metbin turn right 90 degrees, and it will minus 1 after it turn left. It is worth to mention that **d** only has 4 values (1∼4), so when its reaches 5, it will minus 4. Similarly, when (d) is smaller than 1, it will plus 4.

In the figrue 1, it's clear that Metbin can only move against the direction of x axis, so its initial **d** is 4.

1.3 Square Information and its Variables

To realize the search function, we introduce ten variables to describe each square.

variable name	meaning	value range
E	The order of the square first reached by Metbin	1∼16
X[**E**]	The x coordinate of square **E**	1∼4
Y[**E**]	The y coordinate of square **E**	1∼4
D[**E**]	Metbin's **d** when it first reaches square **E**	1∼4
L[**E**]	The left obstacle situation when Metbin first reaches square **E**	-1∼16
F[**E**]	The front obstacle situation when Metbin first reaches square **E**	-1∼16
R[**E**]	The right obstacle situation when Metbin first reaches square **E**	-1∼16
l[**E**]	The left priority when Metbin reach square **E**	0∼3
f[**E**]	The front priority when Metbin reach square **E**	0∼3
r[**E**]	The right priority when Metbin reach square **E**	0∼3

Table 1: Variables describing a square

1.3.1 Square Order E and Direction D[E]

As mentioned in table 1, **E** represents the order of the square first reached by Metbin. For example, in Figure 1, Metbin's initial position's **E** is 1 and the square with coordinates (3,4) is 2. Every square only has one corresponding **E**. If Metbin return a square which already has **E**, it won't change. Since there are 16 squares in the field, $E_{max} = 16$.

Additionally, when Metbin first reaches the square, its **d** will be recorded as square **E**'s **D**[**E**]. It also won't change later.

1.3.2 Obstacle situation L[E], F[E], and R[E]

As mentioned the before, the three variable describe the obstacle situation in three sides when Metbin is in the square with direction **D**[**E**]. If there is an obstacle on right side, then **R**[**E**] will be 0. If there is no obstacle then, if there is a reached square **E'**, then **R**[**E**]=**E'**, else]**R**[**E**]=-1. Since there are 16 squares in the field, the obstacle variable can't be bigger than 16.

If every square's obstacle situation is bigger than -1, than Metbin has already reached every squares.

1.3.3 Priority Variable l[E], f[E], and r[E]

The three variable in lower case describe the order of directions to turn when Metbin reach **E** with **D**[**E**]. These variable value is smaller than 3 because there is only 3 directions Metbin can advance. Metbin will choose the biggest variable and go its corresponding direction.

For example , if Metbin reaches square **5** and **l**[**5**]=0, **f**[**E**]=1, and **r**[**5**]=3, then it will turn right keep going. Then **r**[**5**] will become 0 after Metbin go back.

When Metbin reaches a new square, the three variables will generated according to Metbin's relative location to the destination and obstacle situation variables. If there is an obstacle on the left side (**L**[**E**]=0) then **l**[**E**] will also be 0 because there can't go. If there is no wall, then if the obstacle is in front of Metbin, then **f**[**E**] will be 3. Then if the obstacle is on the right, **r**[**E**] will be 2 and **l**[**E**]=1. If the obstacle is on the back of Metbin, then **f**[**E**] will be 1 and other two variables will plus one correspondingly. Otherwise, Metbin will go at the order of left, front, then right.

1.3.4 An Example without return from Figure 1

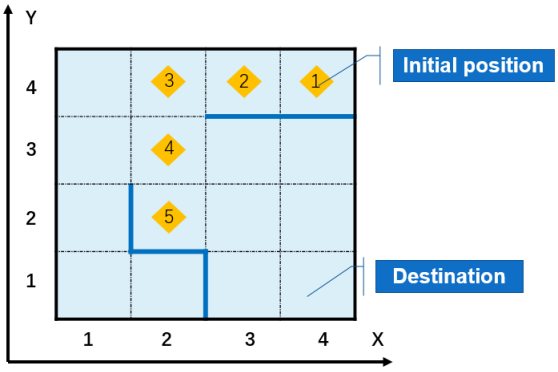

Figure 3: Direction variable

The first 5 squares' **E** is already shown on the picture and the following table shows the information recorder when Metbin first reaches square 5.

E	**X[E]**	**Y[E]**	**D[E]**	**L[E]**	**F[E]**	**R[E]**	**l[E]**	**f[E]**	**r[E]**
1	4	4	4	0	2	0	0	2	0
2	3	4	4	0	3	0	0	1	0
3	2	4	4	4	-1	0	3	2	0
4	2	3	3	-1	5	-1	2	3	1
5	2	2	3	-1	0	0	2	0	0

Table 2: Information variables of the first 5 squares.

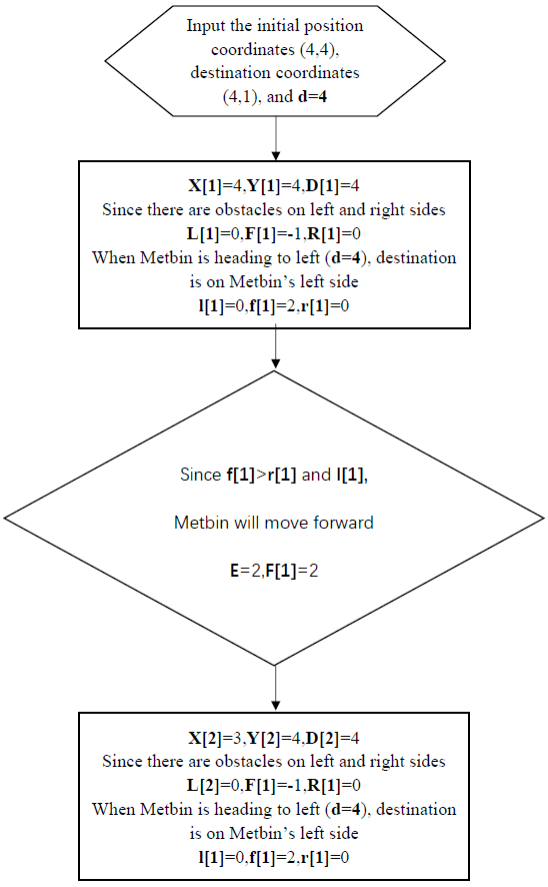

Figure 4: Flow chart between square 1 and square 2

If there are only few obstacles, Metbin can go to the destination according to the above flow chart.

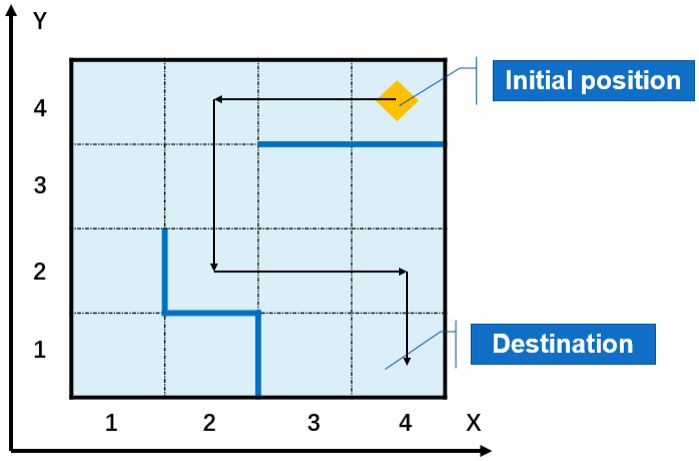

Figure 5: Metbin's route to destination

1.4 Realization of Return Function

However, if there are many obstacles in the room, Metbin may reach a square **E** which has three walls around it (**L**[**E**]=0,**F**[**E**]=0, and **R**[**E**]=0). Then Metbin must return to another square **E'** who must have one obstacle situation variable equals to **E**. If there are many **E'** has such variable, then Metbin will go to the square with smallest **E'**.

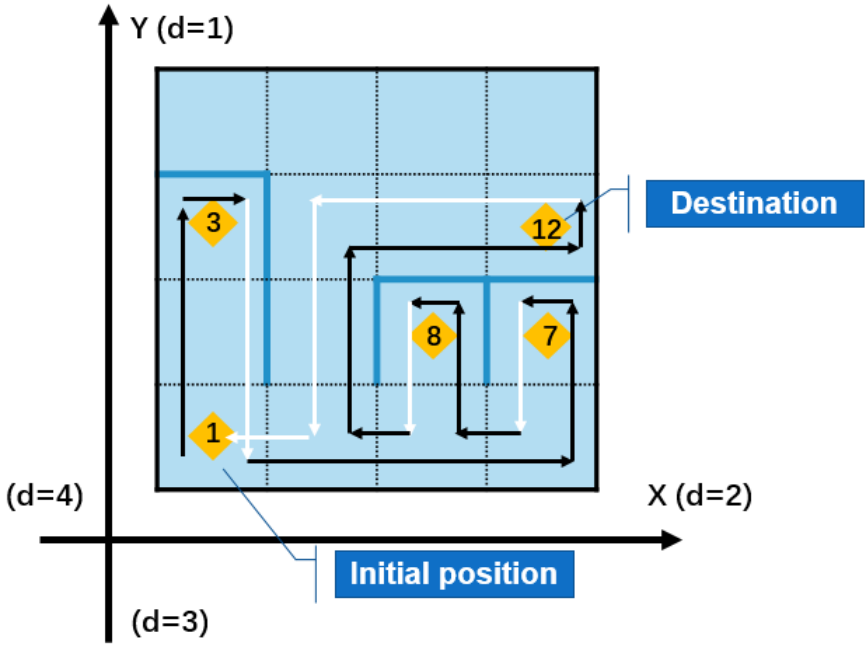

Figure 6: Metbin's whole route in example 2

In the Figure 6, Metbin's initial square's coordinates are **X**[**1**]=1, **Y**[**1**]=1, and **D**[**1**]=1. The destination's coordinates are **X**[**E**]=4 and **Y**[**E**]=3. We draw the route of Metbin according to our algorithm, the black lines represent Metbin's route to a new square and the white lines represent its route of returning. Some square's **E** are already shown on the figure. This will be recorded when Metbin's moving. In this field, square **3**, **7**, and **8** has three obstacles around it, so Metbin will return after reached these squares. In addition, since the destination's coordinates has already been input to Metbin and it keeps recording its own coordinates. When these coordinates are same, Metbin knows that it has reached the destination and return.

2 Computer Simulation

Since the algorithm is hard to realize, we decide to simulate it on computer first. In this procedure, we draw a field and Metbin's route. Then we input the obstacle situation variable according to Metbin's route. This will be detected by infra-red sensors in real field. We only input 0 or -1 as every square's obstacle situation variable's value. (-1 means there is no obstacle and 0 represents there is an obstacle.)

for example in figure 6, we can input the following list

E	**L**[**E**]	**F**[**E**]	**R**[**E**]
1	0	-1	-1
2	0	-1	0
3	0	0	0
4	-1	-1	0
5	-1	-1	0
6	-1	0	0
7	0	0	0
8	0	0	0
9	0	-1	0
10	0	-1	-1
11	-1	-1	0
12	-1	0	0

Table 3: Input of Obstacle Situation

Then the Metbin can generate every square's information variable according table 3

E	**X[E]**	**Y[E]**	**D[E]**	**L[E]**	**F[E]**	**R[E]**	**l[E]**	**f[E]**	**r[E]**
1	1	1	1	0	2	4	0	3	2
2	1	2	1	0	3	0	0	3	0
3	1	3	1	0	0	0	0	0	0
4	2	1	2	9	5	0	2	3	0
5	3	1	2	8	6	0	2	3	0
6	4	1	2	7	0	0	3	0	0
7	4	2	1	0	0	0	0	0	0
8	3	2	1	0	0	0	0	0	0
9	2	2	1	0	10	0	0	3	0
10	2	3	1	0	-1	11	0	2	3
11	3	3	2	-1	12	0	2	3	0
12	4	3	2	-1	0	0	0	0	0

Table 4: Information variable

To test whether Metbin will move according to the correct route, we introduce a array **A[i]** of four values from $1 \sim 3$.

A[i]	1	2	3
	move forward to next square	turn left 90 degrees	turn right 90 degrees

Table 5: Array **A**

So Metbin should generate **A[i]**={11133111333111212213133212213133211311221121131} if the Metbin can generate the correct **A[i]** according to our input obstacle information, we just need to let Metbin read the array and make the correct motion

3 Realization of A[i]

Since the environment is of great randomness, we must use multiple sensors to ensure Metbin is in the correct position in the field. We decide to use posture sensors, encoders and many infra-red sensors.

3.1 Posture Sensors

The posture sensors have many functions such as detect the magnetic field and its angle. It can calculate its deflection angle by angular acceleration integral and this is the only function required. so I can set it on computer.

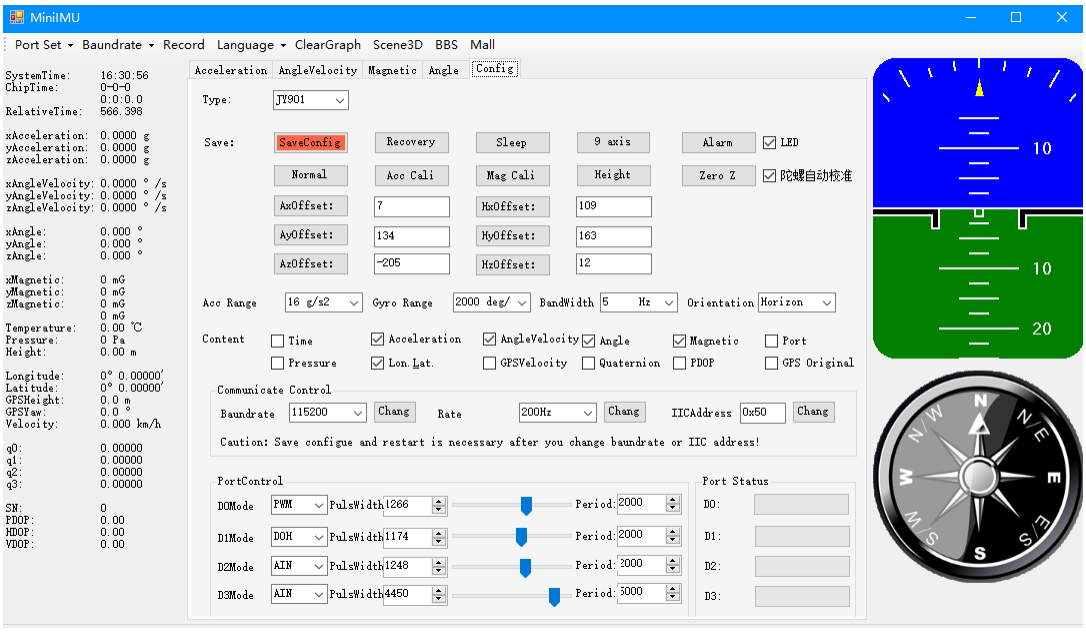

Figure 7: Posture Sensors' setting surface on computer

The posture sensors can return a value **z** describing the angle deflected. It's value range is $0 \sim 360$. If the Metbin needs to turning right 90 degrees, it will only turn until $\mathbf{z\text{-}z_{initial}} \approx 90^o$. Similarly to **d** in Search Algorithm, $\triangle_{\mathbf{z}}$ may be small than 0 sometime, so it need to be dealt with.

3.2 Encoders

We use encoders to recorded how far does Metbin move on the field. To eliminate accumulative, it will be set to 0 every time Metbin meet an obstacle. The encoders can return 1 and 0 when it receive a change in Motor's voltage. So we can record the number of pulse to calculate how far does Metbin move. After many experiments, we think when Metbin will move 55cm when 1500 pulse are returned.

3.3 Infra-red Sensors

Infra-red sensors can return its distance from obstacle.

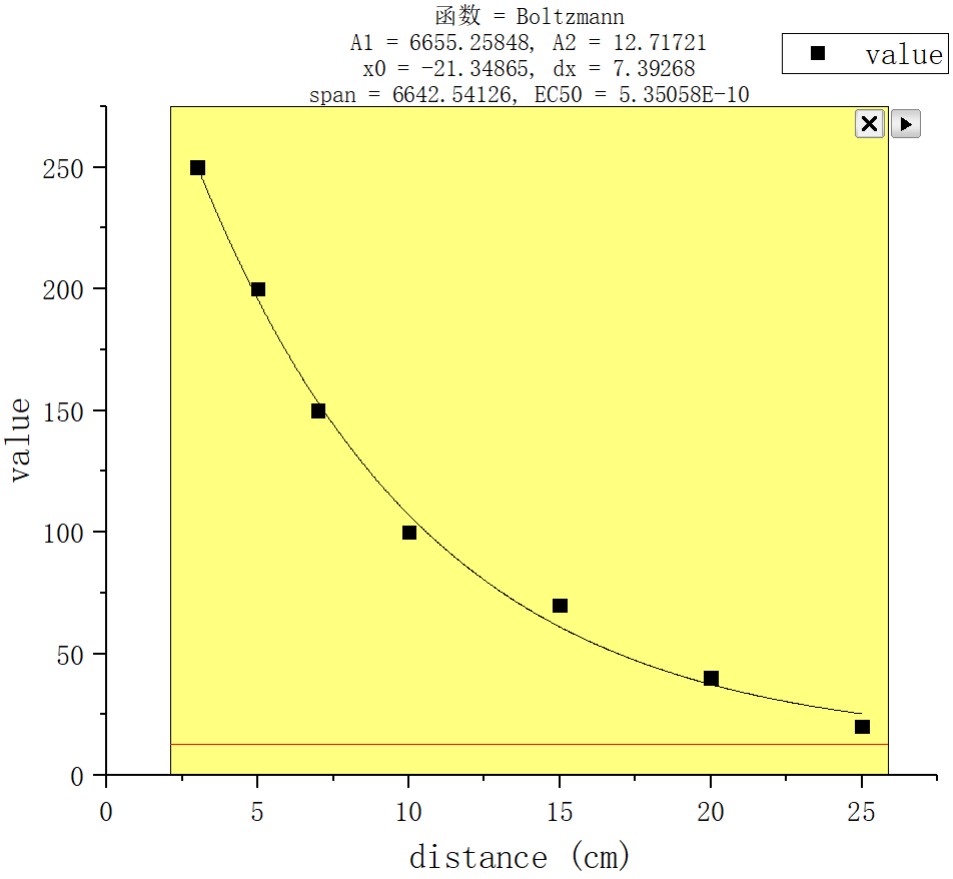

Figure 8: Relation between the distance and return value.

When infra-red sensors' return value is bigger than 100, we assume there is an obstacle before it. We install five infra-red sensors on Metbin, one is installed at the head and two are installed at left and right sides respectively. In addition ,infra-red sensors can be used to detect whether Metbin is parallel to the wall. Since there are two infra-red sensors on one side. If $\frac{v_1-v_2}{v_1+v_2} < 0.01$ then Metbin is parallel to the wall. The posture's accumulative error will be eliminate at the time.