# ME 5751 - Module 1: Proportional Control for Point Tracking

Johanna Martinez-Felix
California Polytechnic University
Mechanical Engineering Dep.
Pomona, CA
johannafelix@cpp.edu

***Abstract*— In this module a python based differential drive robot is controlled using mobile kinematic control. Proportional Control for Point Tracking is achieved by using the mobile kinematics transformation equations to transform from cartesian to polar coordinates for a drive robot described in the inertial frame. Mobile kinematics transforms variables from x, y theta to rho, alpha and beta. After this transformation is accomplished, we can find linear and angular velocities using random K-rho, K-alpha and K-beta that fall withing the descriptive requirement properties for local stabilization, and rho, beta found earlier in the mobile kinematics transformation. Angular and linear velocities are then used to find the spinning speed of the left and right wheel, phi left and phi right. Phi left/right is found using forward kinematics formulas for differential drive robots.**

***Keywords— proportional control, point tracking, differential-drive, kinematics , polar coordinates, angular velocity, linear velocity, wheel speed phi***

## I. INTRODUCTION (*HEADING 1*)

Proportional Control is a type of linear Feedback system. It's used when its necessary to have tighter tolerance and quick response for process variables. Predominantly used to stabilize a process, decreasing the steady state error.

There are a few types of proportional controls. For the proportional-only control a short fall is its inability to completely remove residual error, a desired setpoint (SP) minus measured process variable (PV), in certain processes. This prompted the second point tracking method into existence, it is the proportional integrated control system or PI controller. The PI controller behaves the same as proportional controller, however, the integral component is used to remove the error found and separated by the proportional component. PI removes the error by integrating it over time taken to produce a component for output. A third is the proportional integral derivative control system or PID, they regulate pressure, level, temperature, among other things. Unlike Proportional- only control PID's regularly calculates the residual error (SP- PV) and corrects the error.

Other point tracking methods include the on-off control, usually found in domestic thermostats, it is not as complex as proportional control. In fact, it the simplest feed back control option. Depending on the position of the controlled variable relative to a set point, on-off control drives the variable from open to closed.

In the industry proportional control has been proven to be effective in the application of Inner Loop Cascade Control and Surge tank Level Control. In Cascade applications Proportional control counteracts upstream disturbances. In Surge tank level Control the purpose of proportional control is to mitigate effects of upstream disturbances to downstream processes.

## II. THEORY OF PROPORTIONAL CONTROL

Proportional feedback control is defined by the multiplication of proportional gain times residual error plus controller error with zero as output.

$$P_{Out} = K_p * e(t) + P_0 \quad (1)$$

Where the residual error as described in the introduction is a desired set point minus measured process variable.

$$e(t) = (SP - PV) \quad (2)$$

In the case of motion control for a differential-drive robot, depicted in Figure (1), we can use proportional control equation (1) to describe and solve for velocity.

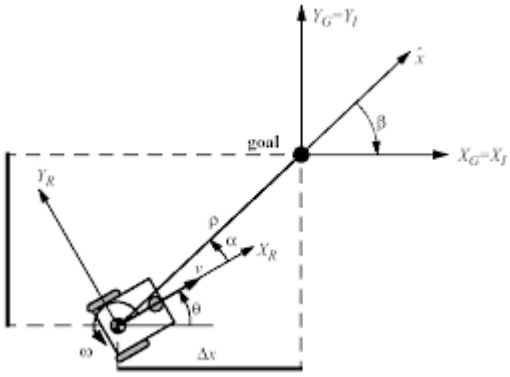


**Figure 1** Differential drive robot kinematics and frames.

The control block diagram describing proportional feedback control for a differential drive robot is figure 2

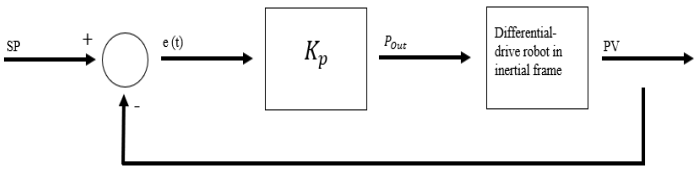


**Figure 2** Proportional feedback control block diagram.

However, the terms are still in need of modification to relate velocity to proportional control. If we let the output, $P_{Out}$ be the velocity we are looking for, $\dot{x}$, and residual error, $e_t$, be $\rho$, the distance between the current location and the distance to the goal location, then

$$\dot{x} = K_p * \rho. \quad (3)$$

In the cartesian coordinate form (3) is expanded to

$$\begin{bmatrix} \dot{x}_R(t) \\ \omega(t) \end{bmatrix} = K_p \begin{bmatrix} x_R \\ y_R \\ \theta \end{bmatrix}. \quad (4)$$

Using coordinate transformation equations (5), (6) and (7)

$$\rho = \sqrt{\Delta x^2 + \Delta y^2} \quad (5)$$
$$\alpha = -\theta + atan2(\Delta y, \Delta x) \quad (6)$$
$$\beta = -\theta - \alpha \quad (7)$$

*Note: $atan2(y, x)$ is only for $-\pi$ to $\pi$

we can represent velocity in polar coordinate as well (8)

$$\begin{bmatrix} \dot{x}_R(t) \\ \omega(t) \end{bmatrix} = K_p \begin{bmatrix} \rho \\ \alpha \\ \beta \end{bmatrix} \quad (8)$$

where linear control law $\omega(t)$, rotation velocity is

$$\omega(t) = \alpha K_\alpha + \beta K_\beta \quad (9)$$

when we assume that $K_p$ is

$$K_p = \begin{bmatrix} k_\rho & 0 & 0 \\ 0 & k_\alpha & k_\beta \end{bmatrix}. \quad (10)$$

When referring to differential drive robot proportional gain $K_p$ refers to local stability. Local stability is typically an issue, however, if the following the conditions are applied the robot control system sees better stabilization:

$$k_p > 0\ ;\ k_\beta < 0\ ;\ k_\alpha - k_\rho > 0 \quad (11)$$

or the strong stability conditions

$$k_p > 0\ ;\ k_\beta < 0\ ;\ k_\alpha + \frac{5}{3}k_\beta - \frac{2}{\pi}k_\rho > 0 \quad (12)$$

for heightened stability of positional control.

In forward kinematics for a differential drive robot, it is also needed to account for the spinning speed of each wheel to achieve full proportional control. This spinning speed can be derived from the rotation velocity.

$$\omega_1 = \frac{r\dot{\varphi}_1}{2l} \text{ and } \omega_2 = \frac{-r\dot{\varphi}_2}{2l}, \quad (13)(14)$$

Where $\omega_1$ and $\omega_2$ are rotation velocities for the right and left wheel respectively. Variable r is the diameter of the wheels, and $l$ is half the distance between the wheels.

This gives the kinematic model for the differential drive robot.

Equations (15) and (16) are the relationship between wheel spinning speed and angular and linear velocity.

$$\dot{\theta}(t) = \omega_1(t) + \omega_2(t) = \frac{\dot{\varphi}_1 r}{2L} - \frac{\dot{\varphi}_2 r}{2L}$$
$$\dot{x_R}(t) = \frac{1}{2}[v_1(t) + v_2(t)] = \frac{\dot{\varphi}_1 r}{2} + \frac{\dot{\varphi}_2 r}{2} \quad (15)(16)$$

## III. Module Experimentation

In this experiment section, three different K values to navigate to a pose were selected. In cartesian coordinates, the pose the robot will be navigating is x = 25, y = 200, theta = .4.

The three different K options are

1. k_rho = 3, k_alpha = 8, k_beta = -1.5
2. k_rho = 8, k_alpha = 20, k_beta = -5
3. k_rho = 1, k_alpha = 3, k_beta = -.2

All three K values follow the local strong stability conditions (12). Proof of stability conditions for k_rho and k_beta are obvious and for k_alpha we can calculate using the third requirement in (12). The third requirement using the K option 1 is 3.59 which is grater than zero. For K option 2 the third requirement in (12) condition results in 6.57, grater than zero. Lastly the third K option values results in 2.03 for the last requirement in condition (12). Thus all conditional requirements for local strong conditions are met.

Resulting Robot Paths:

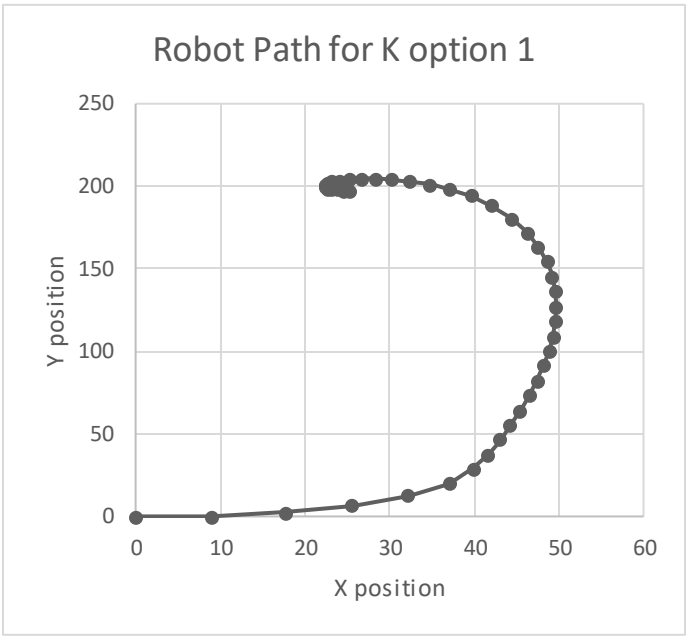


**Figure 3** Resulting Robot Path for K option 1 in centimeters.

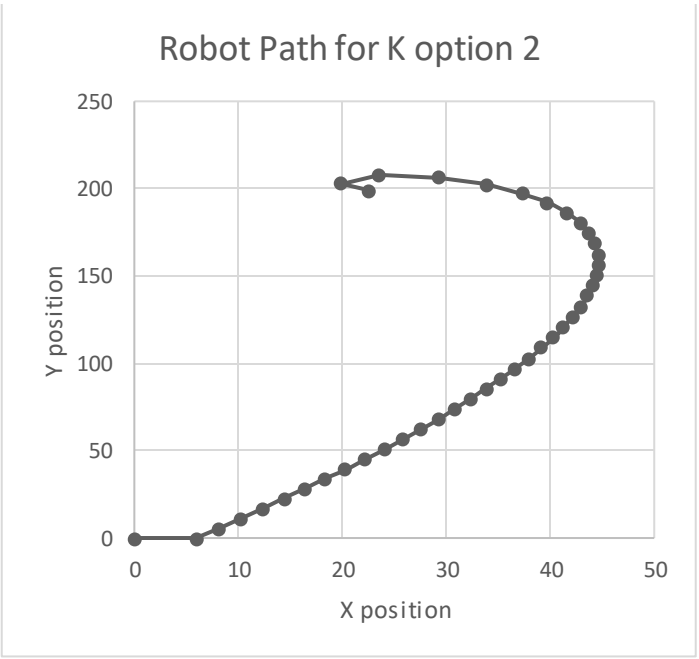


**Figure 4** Resulting Robot Path for K option 2 in centimeters.

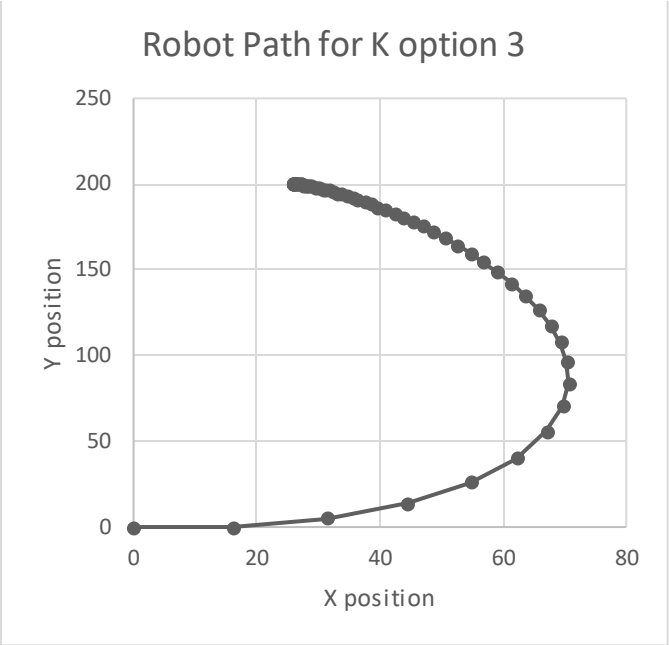


**Figure 5** Resulting Robot Path for K option 3 in centimeters.

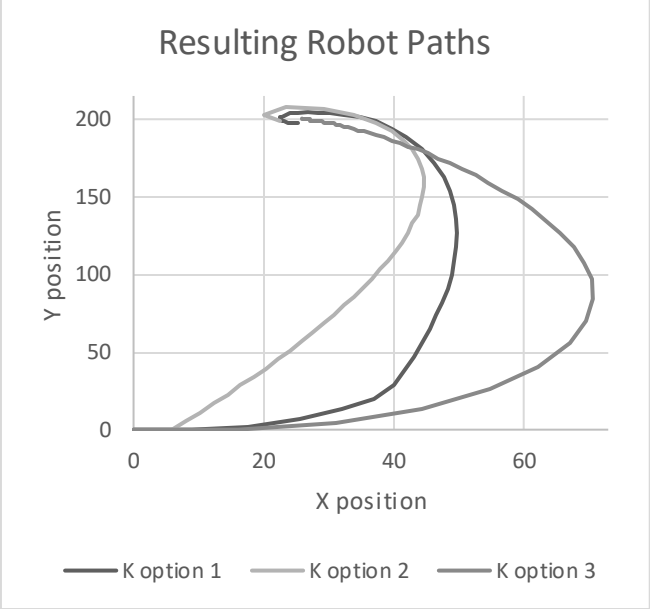


**Figure 6** Resulting Robot Path for K options 1, 2 and 3 in centimeters

Figures 3 through 6 are plots of the path the robot took to get to the selected pose. The Y position and X position on the global frame are in centimeters. Different K values resulted in different robot baths with similar movement strategies.

Robot Angular Velocities:

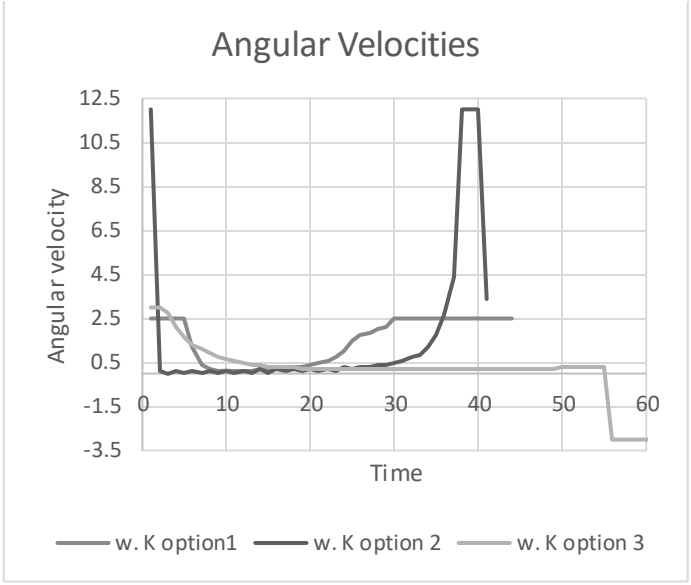


**Figure 7** Robot Angular Velocities for K options 1, 2 and 3

Figure 7 depicts the angular velocity $\omega(t)$ in radians per second versus time(s). Different K values has resulted in different angular velocities.

Linear Velocities:

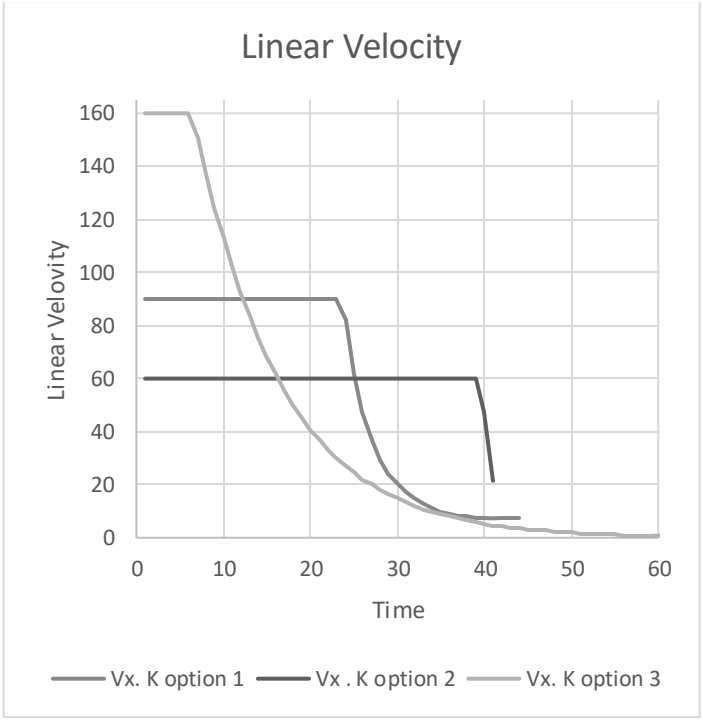


**Figure 8** Robot Linear Velocities for K options 1, 2 and 3.

Figure 8 depicts the linear velocity $\dot{x} = v$ in centimeters per second versus time(s). Different K values has resulted in different linear velocities.

*A. K Analysis: The K values for option one had a set max and min linear velocity of -90 to 90 m/s and -2.5 to 2.5 angular velocity for better control of the robot in the python-based GUI. When K values were changed to option 2 the max and min values set in the code had to change to -60 to 60 for linear velocity and -12 to 12 for angular velocity because the previous values no longer tolerated linear and angular velocity values within the range of velocities outputted by the new K values. The same with the option 3 of K values, the new range for linear velocity was -160 to 160 and -3 to 3 for angular velocity.*

## IV. Succesful Navigation of Three Way Ponts

The three way points selected are show in figure 9.

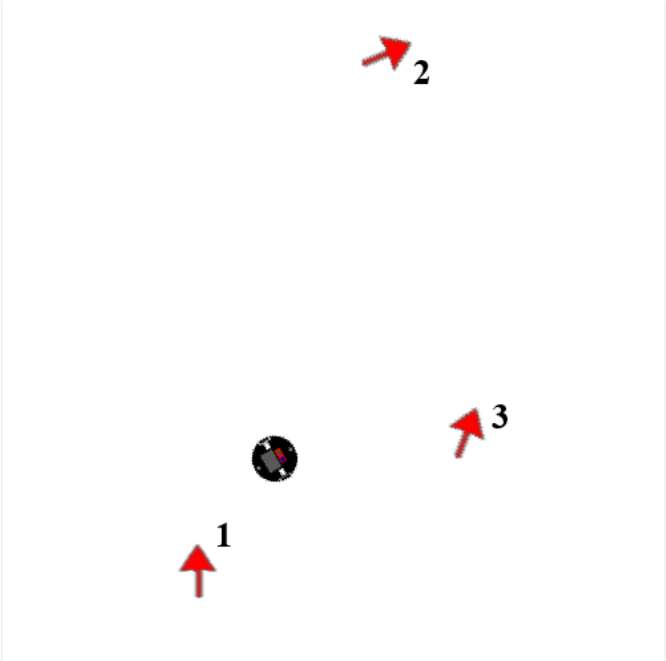


**Figure 9** The three way pint selected shown on the python based GUI.

The cartesian coordinates are:

1. x = -100, y = 200, theta = 1.6,
2. x = 25, y = 200, theta = .4, and
3. x = 95, y= -95, theta = 1.2.

### *A. Code applications of equations.*

Three way point navigation was accomplished by converting the cartesian coordinates to polar coordinates using (5), (6) and (7) to get rho, alpha and beta. Shown in figure 10. Note the if statement. As previously mentioned atan2(y,x) only spans between -pi to pi.

```
chngX = d_posX - c_posX
chngY = d_posY - c_posY

rho = math.sqrt(math.pow(chngX,2) + math.pow(chngY,2))  #rho
alpha = math.atan2(chngY, chngX ) - c_theta  #alpha
if(alpha>math.pi):
    alpha=alpha-2*math.pi
if(alpha<-math.pi):
    alpha=alpha+2*math.pi
beta = - d_theta + alpha +c_theta # beta
if(beta>math.pi):
    beta=beta-2*math.pi
if(beta<-math.pi):
    beta=beta+2*math.pi
```

**Figure 10** Shows code implementation of (5), (6) and (7).

Solving for rho, alpha and beta allows to use linear control laws to get linear (3) and angular velocities (9). Shown in figure 11.

```
c_v = k_rho*rho
c_w = k_alpha*alpha + k_beta*beta
```

**Figure 11** Implementation of (3) and (9).

However, for linear velocity random Kp values that conform to local stability conditions found in (11) and (12) need to be chosen.

```
k_rho = 3  # rho
k_alpha = 8 # alpha
k_beta = -1.5 # beta
assert k_alpha+5/3*k_beta-2/math.pi*k_rho>0
```

**Figure 12** Implementation of local stability conditions for selected K values. In the previous section on the lab the K values were experimented with by changing them.

Using the equations that hold relationship to angular and linear velocity (15) and (16) and solutions already gathered we can solve for wheel speed for the right and left wheel, $\varphi_l$ *and* $\varphi_r$.

```
r=3
L=12
trix = np.array([[r/2, r/2], [r/(2*L),-r/(2*L)]])
invtrix = np.linalg.inv(trix)
[phi_r, phi_l] = np.matmul(invtrix, [c_v, c_w])
```

**Figure 13** Solving for right an left wheel speed. Variable r and l were given in the assignment description.

Figures 14 through 16 depict the robot at the desired location with the command window reading that the way point goal has been reached by the robot successfully.

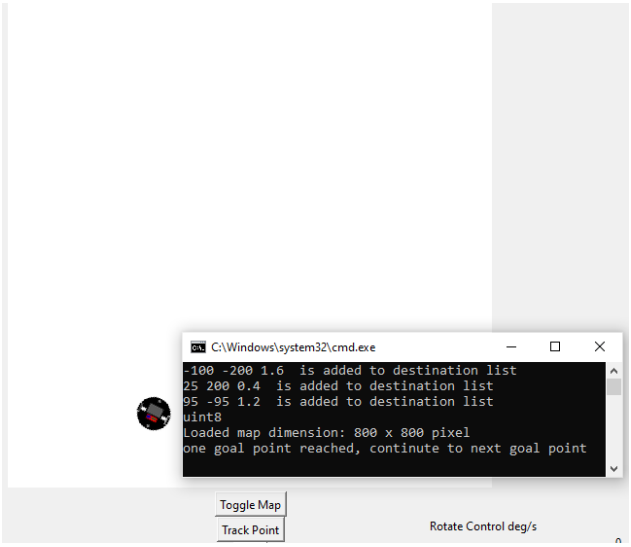


**Figure 14** Command window response after hitting the first way point.

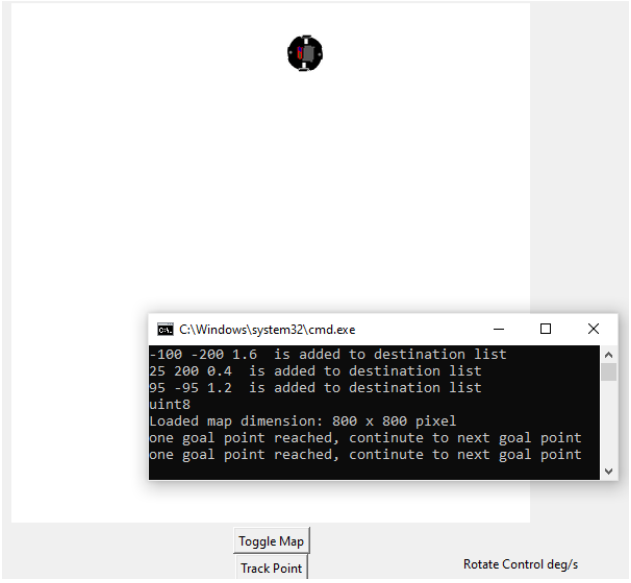


**Figure 15** Command window response after hitting the second way point.

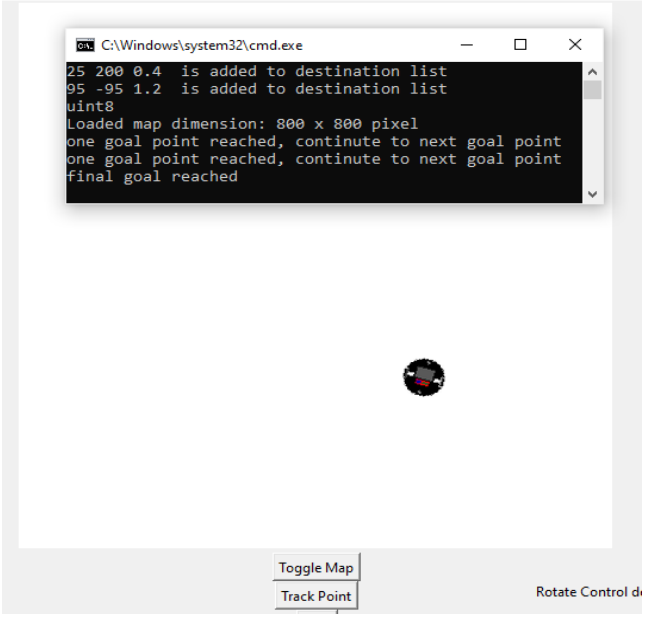


**Figure 16** Command window response after hitting the third and last way point.

## V. WOULD THE ROBOT BE ABLE TO GO BACKWARDS?

Yes, the robot should be able to go backwards if the wheel speed is negative. That is if we apply scenarios where, $\varphi_l$ *and* $\varphi_r$ are negative, $-\varphi_l$ *and* $-\varphi_r$. This means that the wheels are moving in reverse. This would potentially result in much shorter paths to goal way points from the current coordinates.

## VI. CONCLUSION

Different K values result in different robot paths, different angular and linear velocities. Proportional Control for a differential drive robot can be achieved through geometry found in the kinematic model and through control laws. In addition scenarios where alpha and beta are outside of the -pi to pi range must be accounted for.

## VII. ACKNOWLEDGEMENTS

Thank you Dr.Chang for providing help in solving the cartesian to polar coordinate equation issue in regards to alpha and beta outside of -pi to pi range during office hours. Sorry you missed your bus.

## REFERENCES

[1] R. Siegwart, I. Nourbakhsh, D. Scaramuzza, Introduction to Autonomous Mobile Robots, 2nd ed., The MIT Press : Cambridge, 2011. pp 57 – 98.

[2] Y. Chang, Robotics Motion Planning , Lecture Note Set #4-5 , Lecture Note Set #2. CPP: ME5751, 2022.

[3] Bequette, B. Wayne. Process Control: Modeling, Design, and Simulation. Upper Saddle River, New Jersey: Prentice Hall PTR. 2003, pp. 165–168