



TITLE: Programing a robot - line following

LEARNING SCENARIO

School:	Duration (minutes):	90
Teacher:	Students' age:	13-14

Essential Question:

How to make a robot ready for line following

Topics:

- Programing robot for line following using a visible light camera, an infrared thermal imager and other detection instruments

Aims:

- To learn how to program a robot for line following

Outcomes:

- Knowing how to code a robot for line following

Work forms:

- work in pairs, group work

Methods:

- presentation, talk, discussion, interactive exercise

ARTICULATION

Course of action (duration in minutes)

INTRODUCTION

We have learned how to program robots to be able to track ad follow objects in the previous lesson. Make sure your students understand everything from the last class and are ready for the next step. Let's now learn how to track and follow the line.



MAIN PART

Line tracking refers to the process of object moving along a designated route. A fully functional line tracking robot uses a mobile robot as a carrier, a visible light camera, an infrared thermal imager and other detection instruments as a load system, a multi-field information fusion of machine vision, electromagnetic field, GPS and GIS as a navigation system for autonomous movement and tracking of the robot, and an embedded computer as a software and hardware development platform for the control system.

Line tracking sensors

Type	Infrared Line Tracking Sensor	Visual Sensor
Comparison Items		
Cost	Low	High
Range of Vision	The sensor has a small range of view; it needs to be close to the ground .	The range of vision is wide, and the movement state can be adjusted in advance according to the line changes.
Environment Adaptation	When the usage environment is changed, the sensitivity of the sensor needs to be adjusted, and the adjustment process is complicated.	When the use environment is changed, only the lines need to be relearned, and the operation is simple.
Map Adaptation	Generally, only suitable for simple maps with clear background lines, black and white lines or solid lines	Suitable for maps with clear background lines, multi-colour lines, solid lines, dotted lines and other complex conditions.

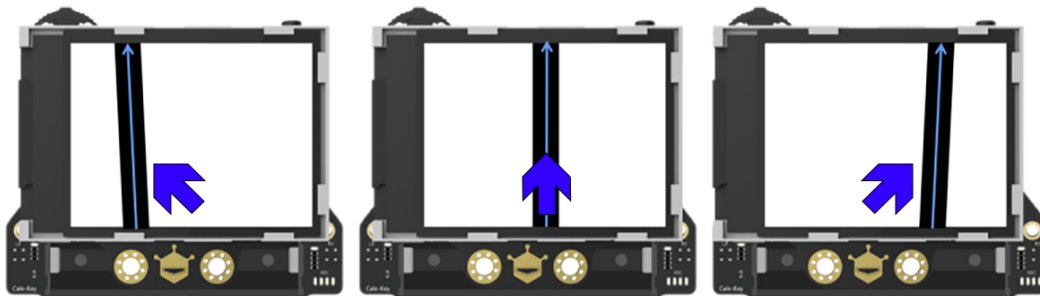
HuskyLens line tracking algorithm

HuskyLens line tracking function is based on Pixy, an open-source project of Carnegie Mellon University. Pixy's algorithm can recognize the colour of pictures. Its basic idea is to use the colour space to remove the background no users are interested in and extract the foreground (such as lines).



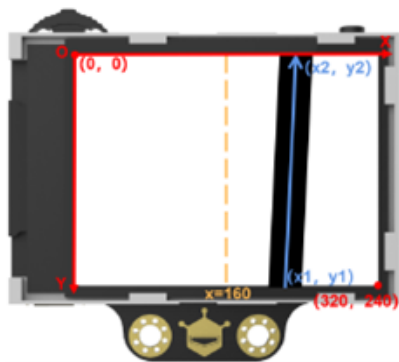
How can ARTIEbot follow the black line on the tracking map (which has black lines on white ground)? In fact, we only need to know the relative position of ARTIEbot to the black line. Basically, we have following three situations:

1. When ARTIEbot is on the right side of the black line, ARTIEbot should turn left
2. When ARTIEbot is in centre, aligned with the black line, ARTIEbot should go straight
3. When ARTIEbot is on the left side of the black line, ARTIEbot should turn right



Implementation

The resolution of HuskyLens screen is 320×240. The O point in the upper left corner of the screen is the origin of the screen's coordinates (0, 0), the horizontal right direction is the positive direction of the X axis, and the vertical down direction is the positive direction of the Y axis, so the coordinates in the lower right corner of the screen are (320, 240). The dotted orange line in the above picture is the central axis of the screen, and the x value of this line is 160. The black line in the screen below is the map line "seen" by HuskyLens camera. The blue arrow is the line direction calculated by HuskyLens. The starting point coordinates of the blue arrow are (x1, y1) and the ending point coordinates are (x2, y2).



To make it simple - we only need to know the starting point (x1) of the blue arrow relative to the central axis (x=160) to implement line tracking.

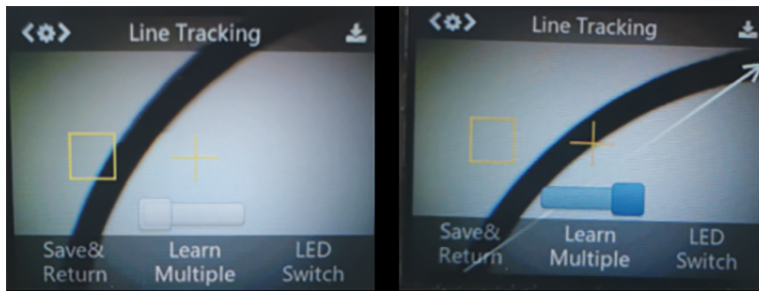
This function can track lines of specified colours and make path prediction. The default setting is to track lines of only one colour and this project will use **one colour line tracking**.

Camera Settings

Step 1: Dial the function button to the left or right until the word "Line Tracking" is displayed at the top of the screen.

Step 2: Long press the function button to enter the parameter setting of the line tracking function.

Step 3: Dial the function button right or left until "Learn Multiple" is selected, then short press the function button, and dial it to the left to turn off the "Learn Multiple" switch, that is, the square icon on the progress bar is turned to the left. Then short press the function button to complete this parameter.

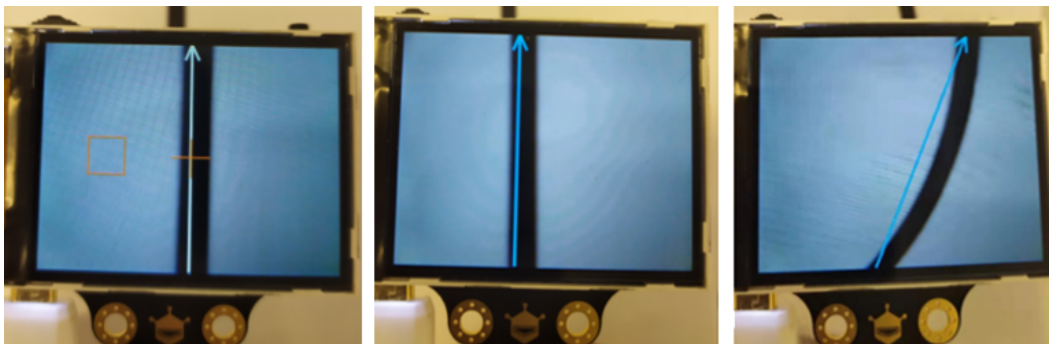


Step 4: You can also turn on the LED by setting "LED Switch". This is very useful in the dark environment. Referring to the above method, turn on the "LED Switch".

Step 5: Dial the function button to the left until "Save & Return" is selected, and short press the function button to save the parameters and it will return automatically.

Learning and Tracking

Line Learning: Point the "+" symbol at the line, then point the orange box at the background area. It is recommended that no other lines are on the screen. Try to keep HuskyLens parallel to the target line; HuskyLens will automatically detect the line and a white arrow will appear. Then short press the "learning button" and the white arrow turns into a blue arrow.



- When learning the line, we need to adjust the position of HuskyLens to be parallel to the line.
- HuskyLens can learn line in any colour that have an obvious colour contrast to background, but this line must be monochrome to keep line tracking process stable.
- HuskyLens can learn and track multiple lines with different line colours, but any of these lines must be monochromatic and in visible contrast against the background. In this example we will use black line (black insulation tape on white background such is paper or white MDF).
- The visibility of the line depends much on the ambient light. When following the line, please try to keep the ambient light as stable as possible and use HuskyLens LED if necessary.

Open your Mind+ and load extensions.



Open your Mind+ and load extensions. **When using the micro:Maqueen plus robot, make sure that you select the right version (V1 or V2).**

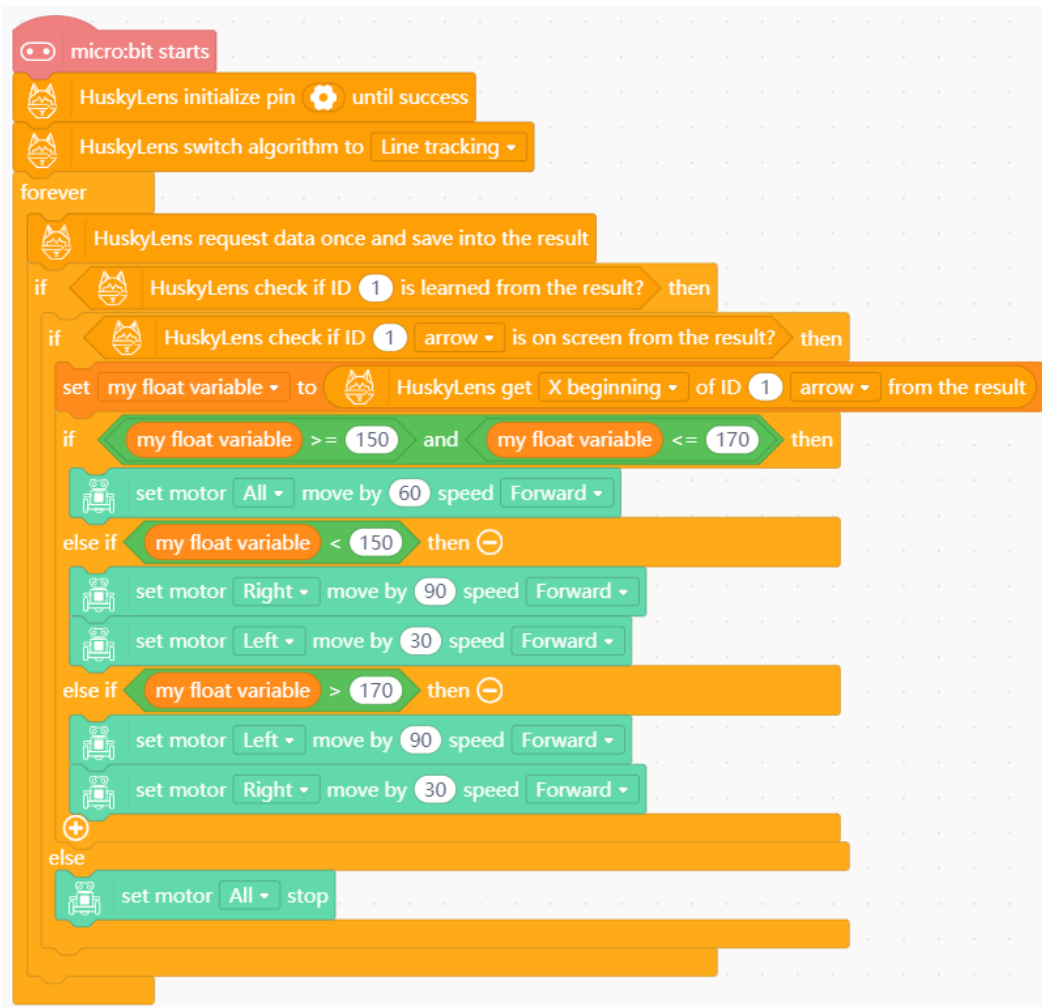
Rename **my float variable** to **x**. Right click on variable -> Rename numeric variable.

Algorithm

- Read x2 (or x endpoint) from HuskyLens Line tracking function - it is the ending point of the blue arrow
- If the black line is on the left side of the screen ($x_1 < 150$) - ARTIEbot should turn left.
- If the black line is on the right side of the screen ($x_1 > 170$) - ARTIEbot should turn right.
- If the black line is in the middle of the screen ($150 \leq x_1 \leq 170$) - control ARTIEbot should go straight.

Option 1 - Line following with Maqueen Plus

Use this code:



```

micro:bit starts
  HuskyLens initialize pin until success
  HuskyLens switch algorithm to Line tracking
  forever
    HuskyLens request data once and save into the result
    if HuskyLens check if ID 1 is learned from the result? then
      if HuskyLens check if ID 1 arrow is on screen from the result? then
        set my float variable to HuskyLens get X beginning of ID 1 arrow from the result
        if my float variable >= 150 and my float variable <= 170 then
          set motor All move by 60 speed Forward
        else if my float variable < 150 then
          set motor Right move by 90 speed Forward
          set motor Left move by 30 speed Forward
        else if my float variable > 170 then
          set motor Left move by 90 speed Forward
          set motor Right move by 30 speed Forward
        else
          set motor All stop
  
```

Option 2 - Line following with Arduino (ArtieBot)

First define blocks Drive and Stop as described in lesson Programming the robot. It is a good idea to put the often-used scripts in backpack and easily move them between projects.

```

define Drive SpeedA SpeedB Time
  digital pin 8 output LOW
  digital pin 9 output LOW
  if SpeedA > 0 then
    digital pin 12 output HIGH
  else
    digital pin 12 output LOW
  if SpeedB > 0 then
    digital pin 13 output HIGH
  else
    digital pin 13 output LOW
  pwm pin 3 output abs of SpeedA
  pwm pin 11 output abs of SpeedB
  wait Time seconds

define Stop
  pwm pin 3 output 0
  pwm pin 11 output 0
    
```

```

Uno starts
  HuskyLens initialize pin until success
  HuskyLens switch algorithm to Line tracking
  forever
    HuskyLens request data once and save into the result
    if HuskyLens check if ID 1 is learned from the result? then
      if HuskyLens check if ID 1 arrow is on screen from the result? then
        set x to HuskyLens get X endpoint of ID 1 arrow from the result
        if x >= 150 and x <= 170 then
          Drive 90 90 0.25
        else if x < 150 then
          Drive 40 90 0.25
        else if x > 170 then
          Drive 90 40 0.25
        else
          Stop
      else
        Stop
    else
      Stop
  end forever

define Stop
  pwm pin 3 output 0
  pwm pin 11 output 0

define Drive SpeedA SpeedB Time
  digital pin 8 output LOW
  digital pin 9 output LOW
  if SpeedA > 0 then
    digital pin 12 output HIGH
  else
    digital pin 12 output LOW
  if SpeedB > 0 then
    digital pin 13 output HIGH
  else
    digital pin 13 output LOW
  pwm pin 3 output abs of SpeedA
  pwm pin 11 output abs of SpeedB
  wait Time seconds
    
```



Both options - Test your algorithm

Prepare the white surface (paper or MDF) and use the insulation tape to make a line (see example below or make something similar).



Upload the program to your robot, place the robot somewhere on the line and see how it moves. Does it follow the line?

Is the line being lost? Is there a way to deal with it?

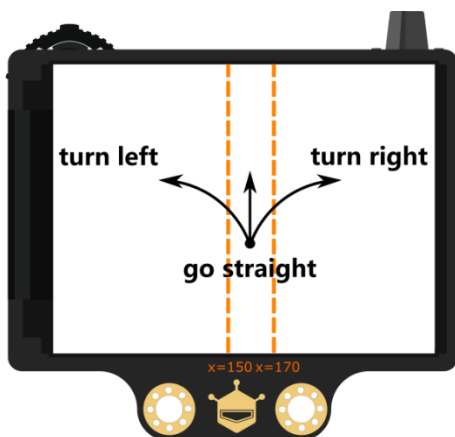
The following hints will help you:

- Try changing the angle of camera
- Try to get **X beginning** of ID 1 arrow instead of X endpoint

Add fail-safe code using following hint:

- If a line is present and an arrow is detected, proceed to the arrow handling code and save the content of **x** variable to a new variable called **lastx**
- If line is lost and arrow is not detected, use **lastx** instead of **x** to reach the line again.

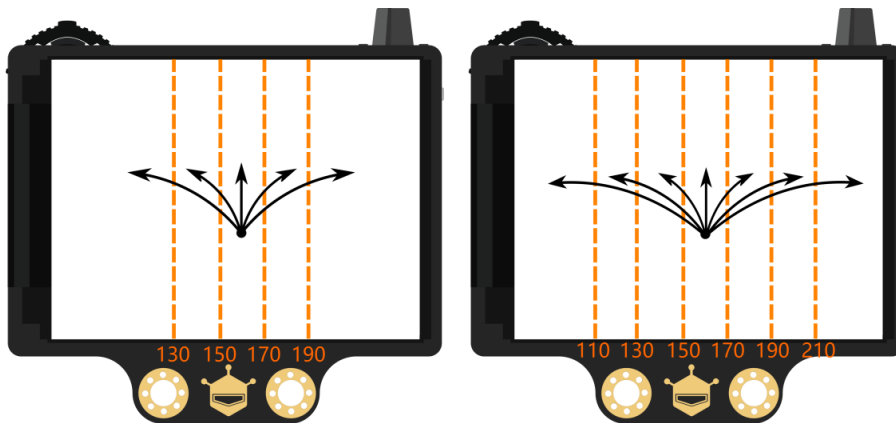
And finally the most important - apply the corrections to make the movement smooth by adjusting motors speed and time of driving. This algorithm analyses the position of **x**; it has 3 cases as on the picture below.



Can you optimize this algorithm for 5 cases or even 7 to make the movement smoother?
See pictures below to get the idea how to do this.



ARTIE: Artificial Intelligence in Education - challenges and opportunities of the new era:
development of a new curriculum, guide for educators and online course for students
Project co-funded by European Union under Erasmus+ Programme, 2020-1-HR01-KA201-077800



Knowledge Recap

1. Understand the main principles of line tracking
2. Learn to apply HuskyLens line tracking function
3. Apply and optimize the line following algorithm

CONCLUSION

As the name suggests, the line follower robot is an automated vehicle that follows a visual line embedded on the surface. This visual line is a path on which the line follower robot runs. Generally, it uses a black line on a white surface, or you can adjust it as a white line on a black surface.

Usually, beginners and students would get their first robotic experience with this type of robot.

In industries, giant line follower robots are used for assisting the automated production process. They are also used in military applications, human assistance purposes, delivery services, etc.

Now we understand the main principles of line tracking, and know how to apply HuskyLens line tracking function.

We also know how to apply and optimize the line following algorithm.

Discuss with your students the differences and similarities between object tracking and following and line tracking and following.

Do the K.W.L. (**K**now, **W**ant, **L**earned) chart with your students.

What I K now	What I W ant to Know	What I L earned





Methods

presentation
interactive exercise / simulation on the computer

Work forms

work in pairs
group work

Material:

- <http://mindplus.cc/download-en.html>

Literature

-

PERSONAL OBSERVATIONS, COMMENTS AND NOTES