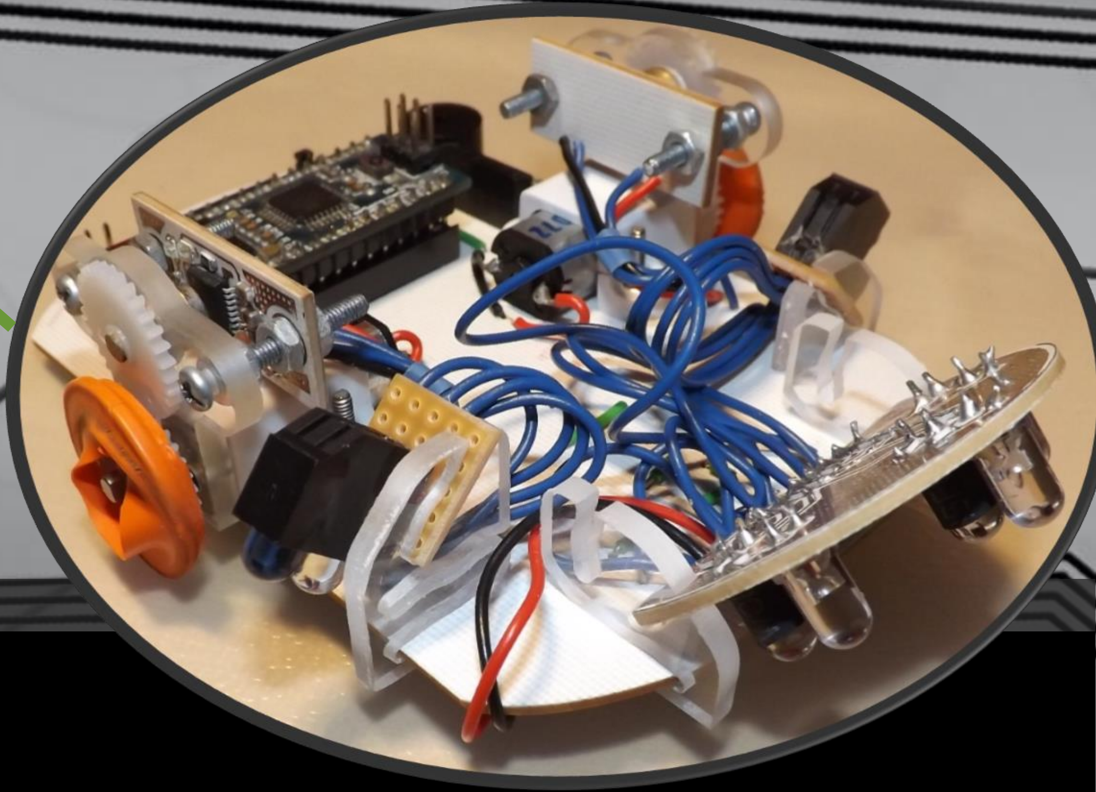


Why not make it go faster?

Stephen Pithouse

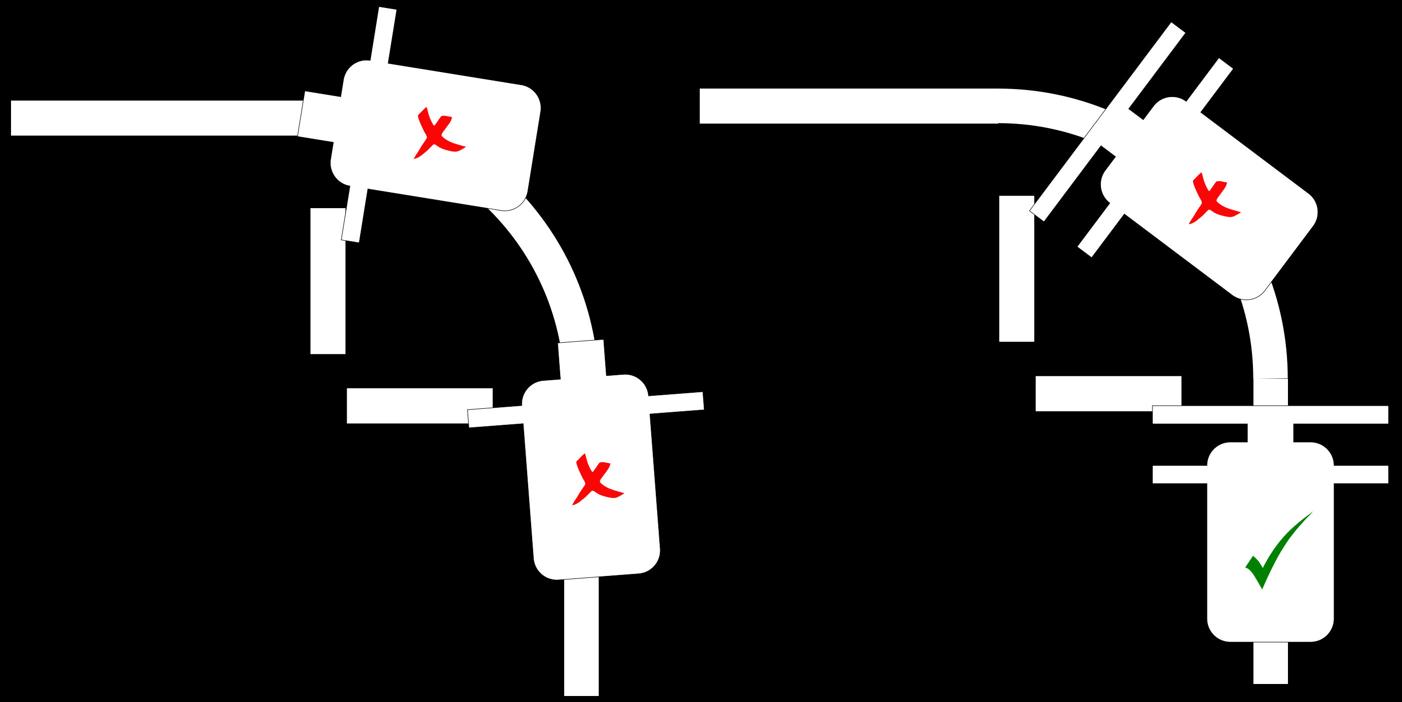
Disclaimer – I may have overlooked aspects which render this theory useless!!

Why not just turn the speed up?

- It is possible:
  - The speed can be increased 50% more along the straights
- Without distinguishing between straights and turns, the same control algorithm becomes erratic at corners
- Therefore, need to determine where the corners are, by mapping the track, either:
  - Using Corner Markers
  - Using Encoders

Using Corner Markers

- Need to detect markers at point where robot is just starting a new curvature section
    - Sensor placement is crucial

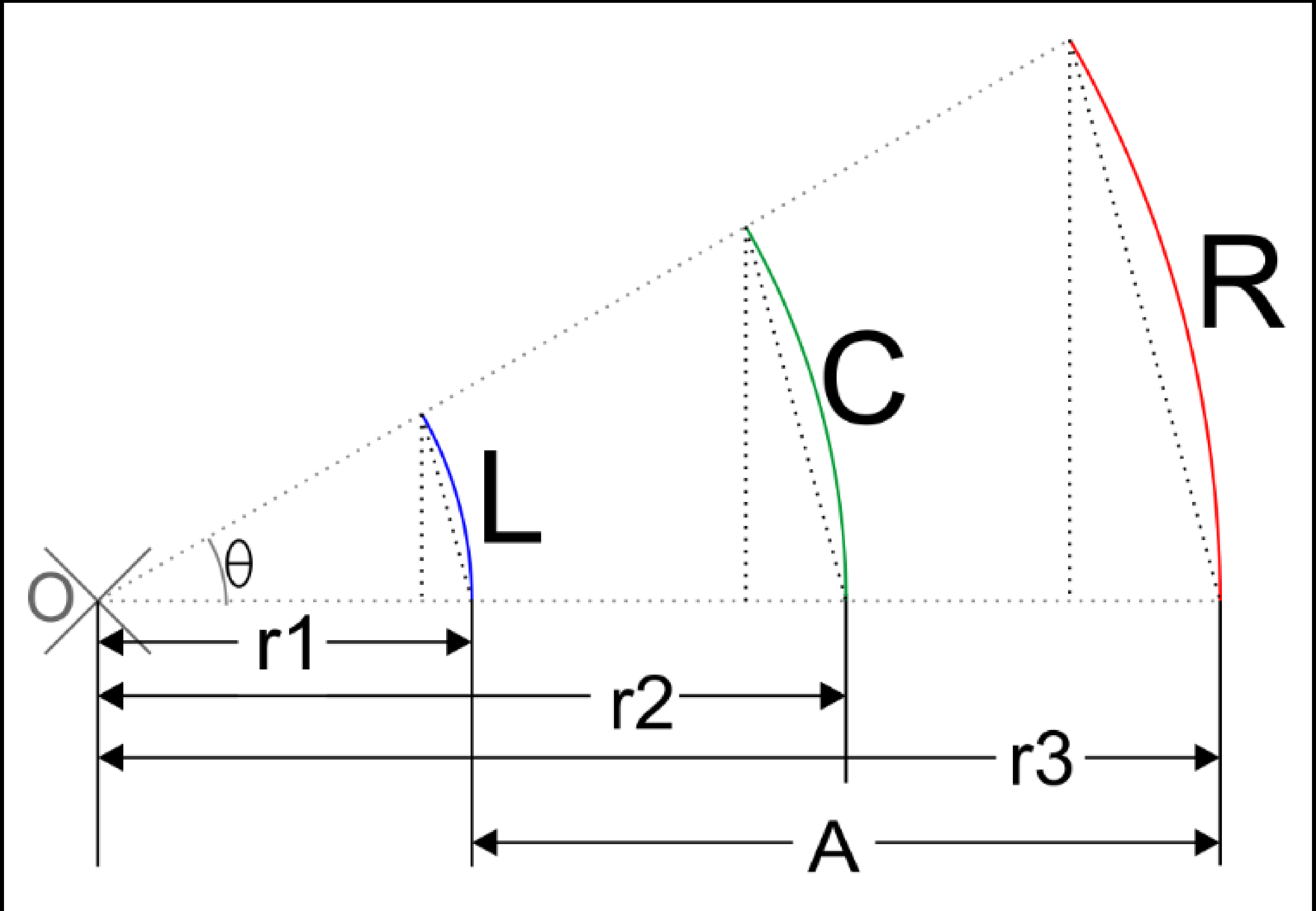


- Tried many different sensor placements, but none proved satisfactory, when coming out of curves, the corner marker was detected too early

Using Encoders

- Forget corner markers altogether, and map track using wheel encoders
  - Need to find a way of mapping the track using only the encoder count data from each wheel
- Save the Δ encoder counts at regular intervals (of time or distance)
  - Use these 2 numbers to map the path taken by the robot

Using Wheel counts to map track



The Maths…

Aims:

Need to find:

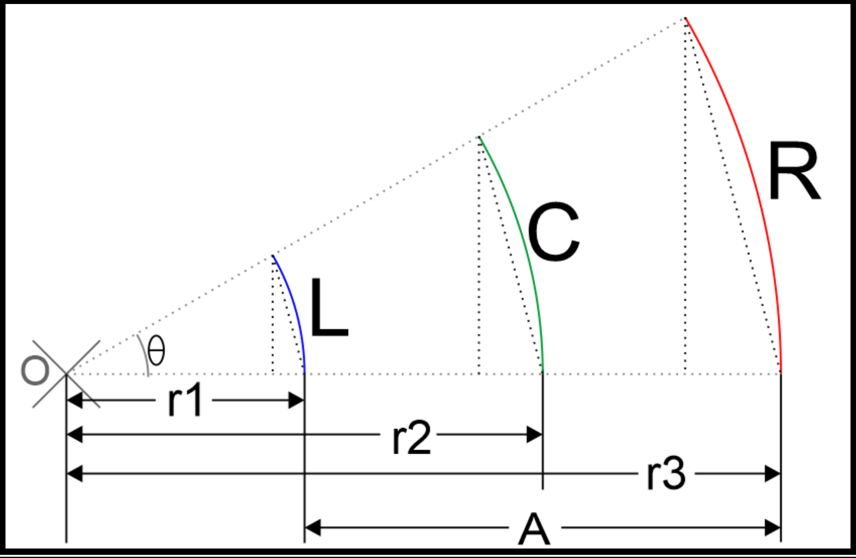
- Radius of track section (r2)

In order to plot on-screen:

- Horizontal, and vertical displacement
- Change in heading angle

Assumptions:

- Travelled along uniform arc-shaped path
- No wheel slip
- Left turn (L<R)
- Constant distance between wheel contact points



Variables:

L – Left encoder count (since last measurement)

R – Right encoder count (since last measurement)

A – Distance between wheels

C – length of track

θ – Angle of curvature (also relates to new heading)

O – Origin of arc

r1, r3 – radii of wheel arcs

r2 – radius of track section

The Maths…

1) Arc Length Formula

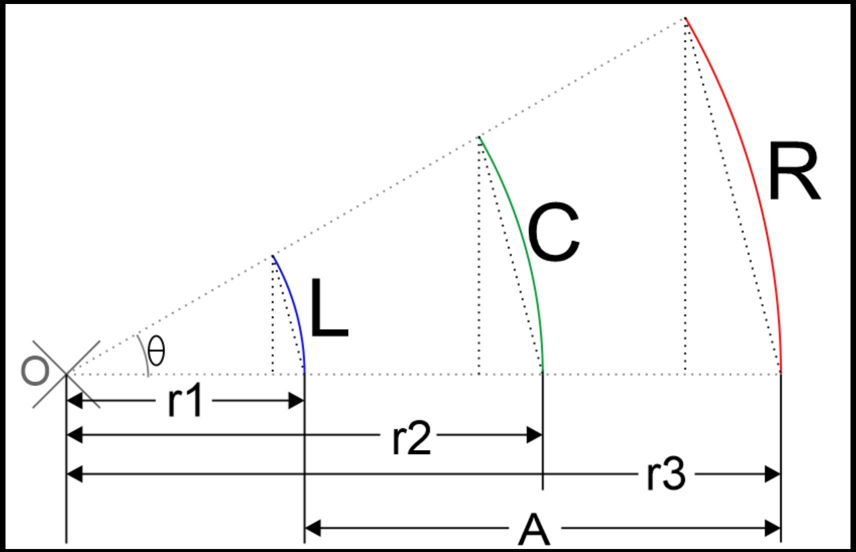
Arc Length: $l = r\theta$, hence $\theta = \frac{l}{r}$

Therefore:

$$\theta = \frac{L}{r_1} = \frac{R}{r_3}$$

$$r_3 L = r_1 R$$

One equation with 2 unknowns, need to find another equation



2) Get in terms of r1

Since: $r_3 = r_1 + A\ldots$

$$(r_1 + A)L = r_1 R$$

$$r_1 L + AL = r_1 R$$

$$AL = r_1 R - r_1 L$$

$$AL = r_1 (R - L)$$

$$\frac{AL}{R - L} = r_1$$

The Maths…

3) Finding the radius of the centreline

$$\frac{AL}{R-L} = r_1$$

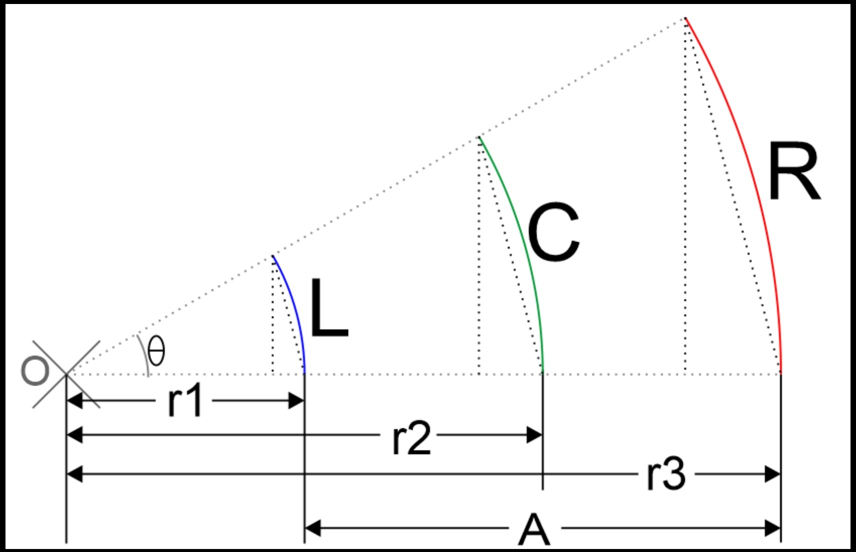
$$r_2 = r_1 + \frac{A}{2}$$

$$r_2 = \frac{AL}{R-L} + \frac{A}{2}$$

$$r_2 = \frac{2AL}{2(R-L)} + \frac{A(R-L)}{2(R-L)} = \frac{2AL + AR - AL}{2(R-L)} = \frac{A(L+R)}{2(R-L)}$$

$$r_2 = \frac{A(L+R)}{2(R-L)} = \frac{L+R}{2} \times \frac{A}{R-L} \text{ for } (L < R)$$

This shows that the radius of the centre line is just the average of the left and right encoder values multiplied by the ratio of distance between wheels to the difference in encoder readings.



Now that we have the formula, it can be written in the more general case (for a left or right turn):

$$r_2 = \frac{L+R}{2} \times \frac{A}{|R-L|}$$

Ensure that L ≠ R, else the radius becomes infinite (straight line)

– Maybe add a catch before the calculation to check if this is the case…

The Maths…

4) To find the angle of curvature (Heading Angle):

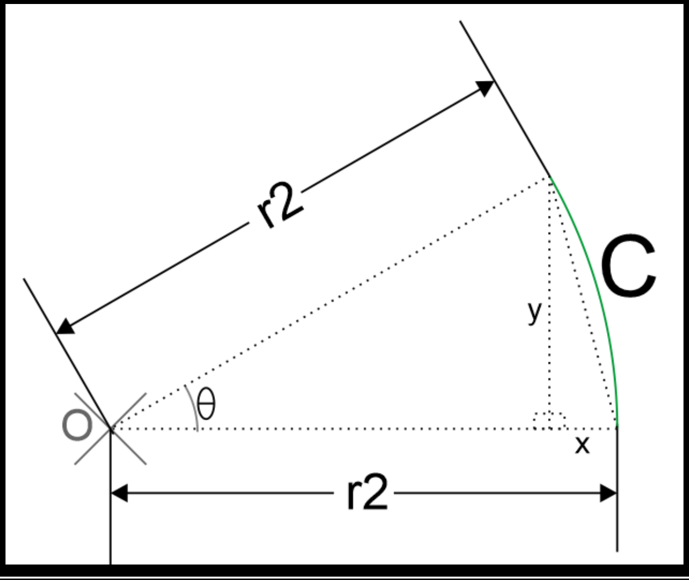
From Step 1 (Arc Length Formula):

$$\theta = \frac{L}{r_1}$$

Substituting r1 (from end of step 2):

$$\theta = L\frac{1}{r_1} = \cancel{L}\frac{R-L}{A\cancel{L}} = \frac{R-L}{A}$$

Applies in general case (L or R turn)



5) Side note, the length of the track, can be found, it is trivial:

C = $r_2\theta$ = $\frac{R+L}{2}$, hence track length: $\sum_{n=1}^{N} C_n$

Can therefore calculate average speed – very useful to determine if the mapping approach is working!

- - Note that we haven't had to use trig. at all - -

6) Displacement (starting from facing upwards):

$$y = r_2 \sin\theta$$

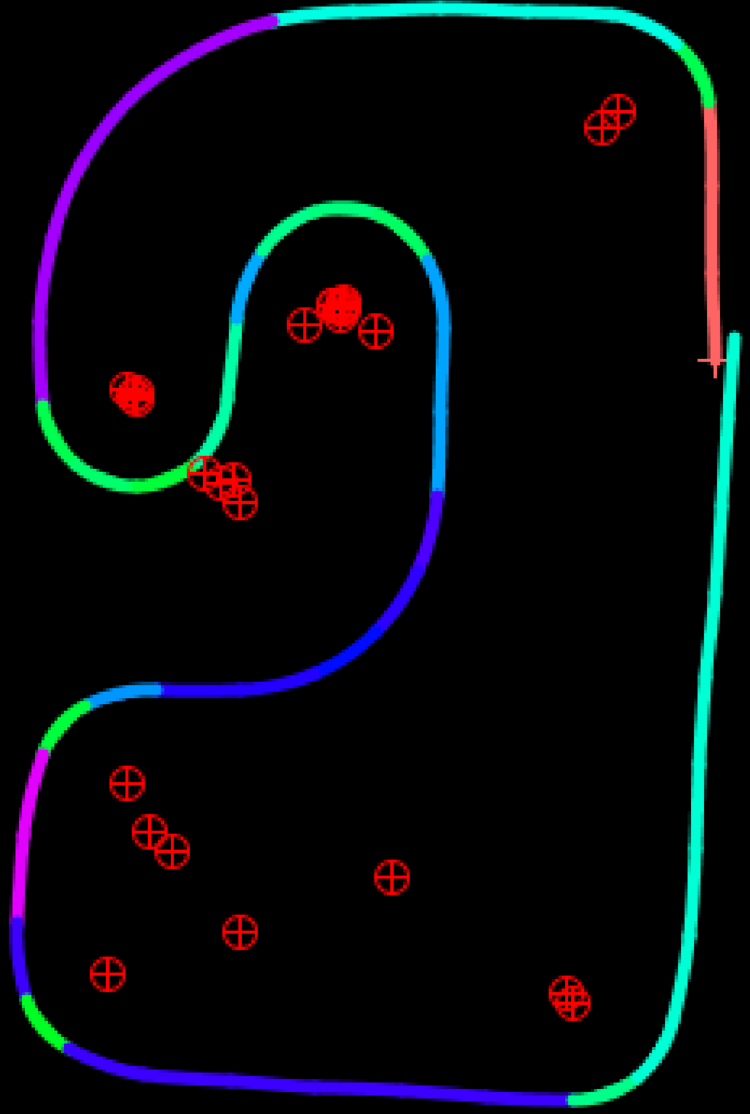
$$x = -r_2 + r_2\cos\theta = r_2(\cos\theta - 1)$$

So… What can these formulae actually do to help??

Tell Line follower to send encoder data (over Bluetooth Serial connection) every ##ms, then reset the values and start counting again

L , R	317, 368	354, 274	352, 343
349, 354	324, 356	355, 269	353, 349
354, 346	189, 372	355, 258	339, 359
332, 360	193, 369	351, 264	199, 369
189, 371	184, 370	352, 331	212, 367
215, 367	204, 368	241, 368	330, 359
345, 360	338, 345	183, 367	348, 363
348, 358	354, 228	305, 358	351, 352
351, 356	351, 190	347, 363	355, 346
311, 357	353, 188	280, 366	353, 354
300, 366	355, 184	180, 370	352, 355
300, 368	349, 226	279, 365	354, 353
300, 369	347, 349	347, 366	335, 334
303, 370	354, 335	353, 358	

Convert values from raw encoder readings to mm, then plot results



So… What does the track look like??

Reasons for error:

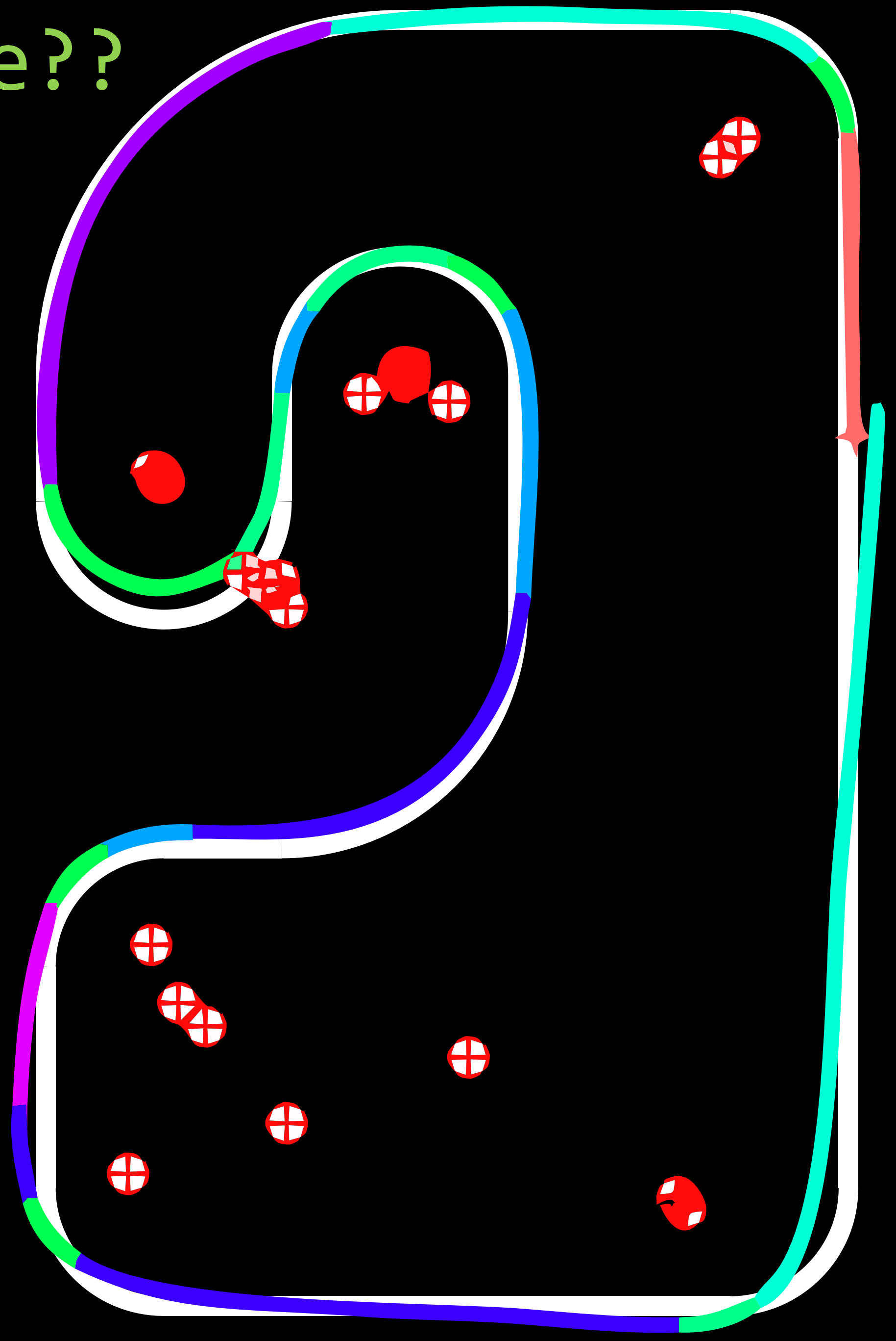
1) Distance measured between wheels may not be accurate
Has much more significance when it is a tight turn

2) Encoder pulses to mm conversion may be out

3) Could have wheel slip (although was travelling at relatively slow 'search' speed)

4) Interval between readings was too long

5) Wheels are different sizes



So… What does the track look like??

Loose colour coding

- Very crude colour coding, based upon radius of each curve

For example each colour could have its own speed profile and control loop parameters

This could be improved by decreasing the interval between encoder data readings

Possible future developments

- Fully investigate sources of error and try and eliminate these
- Move the formulae into the Line Follower and save to map inside memory
- Actually implement active speed profiles in the robot
- Perhaps ‘snap’ curves to known radii (150, 300, 450mm) and angles (90, 180, 270)

Unanswered problems

- Thin point of contact between wheel and surface
  - Good for accuracy
  - But is it bad for wheel grip at higher speeds?
- How often to ‘check encoder data’
  - Frequently – Could pick up unwanted robot oscillation (putting radii into otherwise straight sections)
  - Less often – Smoother output, but may merge features (eg. merge part of straight with sharp corner)

Thank You for Listening...

Any questions

or

suggestions?