Two colour sensors

Move Forward
IF both LEFT and RIGHT colour sensors detect White (>50) then
  Move Forward
IF LEFT colour sensor detects White (>50) and RIGHT colour sensor detects Black (<40) then
  Turn Right
IF LEFT detects Black and RIGHT detects White then
  Turn Left
IF LEFT and RIGHT both detect Black then
  Slow

Refined without AND:
Repeat forever
  If left sensor over white
    [and] If right sensor over white
      Drive straight
    Else [right sensor must be over black ]
      Swing right
  Elseif [left sensor is over black and] right sensor over white
    Swing left
  Else [the right sensor is over black, so] stop

Left LS Threshold = 37
Right LS Threshold = 50
Followed straight line ok, but couldn’t do curves

OR

If sensor 2 left over black (<40) line, then move motor B backwards -50 and motor c forwards 50 (rotate wheel 30 degrees)
Elseif sensor 3 right is over a black line, then motor B forward 50 and motor C backwards -50 (rotate 30)

Else, must be both white, so drive forward both motors B and C 30 speed
Worked ok, but a bit jerky. This can probably be fixed with changing the speeds and degrees.

Advanced - Proportional Line Following with two Sensors

https://www.youtube.com/watch?v=QWOftBu90o&feature=youtu.be

White reading = 68%
Black reading = 16%
Target Turn = 50  {is the value of the steer motors block turn ratio that will allow the robot to travel in a straight line}
Gain = 1  {is used to increase or decrease reaction speed of the robot relative to its position to the line}

Adjust these as well as the motor power to change how the robot handles curves.

targetValue = 40
gainValue = 2
error = LS1 - LS2

On start

gainValue = 2
Loop Forever
error = LS1 - LS2
turnRatio = error * gainValue
Steer motors A + D turnRatio speed 20

on start
set gainValue to 2

forever
set error to color sensor 1.reflected light - color sensor 2.reflected light
set turnRatio to error - gainValue
steer motors B+C turn ratio turnRatio speed 20

let turnRatio = 0
let error = 0
let gainValue = 0
gainValue = 2
forever (function () {
error = sensors.color1.light(LightIntensityMode.Reflected) - sensors.color4.light(LightIntensityMode.Reflected)
turnRatio = error - gainValue
motors.largeBC.steer(turnRatio, 20)
})

https://youtu.be/Z8KFTbo0-f8

Works OK on the straight, but can’t handle curves. Maybe play with the gain

Alternatively
I have seen:

\[
\text{error} = \text{targetValue} - \text{LS1} - \text{LS2}
\]

```javascript
let turnRatio = 0
let targetValue = 0
let error = 0
let gainValue = 0

targetValue = 40

forever(
    function () {
        error = targetValue - (sensors.color1.light(LightIntensityMode.Reflected) - sensors.color4.light(LightIntensityMode.Reflected))
        turnRatio = error * gainValue
        motors.largeBC.steer(turnRatio, 20)
    })
```

This also did not work well on curves. Maybe is you change the gain and target values?

**Proportional Integral Derivative (PID) Line Follower with 2 Colour Sensors**

If you haven’t worked through this with one colour sensor, then go here: [https://www.throughtheclassroomdoor.com/1-light-sensor-lego-ev3-line-follower-robot/](https://www.throughtheclassroomdoor.com/1-light-sensor-lego-ev3-line-follower-robot/)

From the single Colour Sensor PID control solution, we have the following:

**Algorithm for Proportional, Integral and Derivative, using 1 Colour Sensor**

On Start
- Target = 40
- \( Kp = 1 \)
- \( Ki = 1 \)
- Error = 0
- Integral = 0
Loop Forever

error = target - Reflected light value
integral = integral + error
Derivative = error - lastError

turnRatio = (error * Kp) + (integral * Ki) + (Derivative * Kd)
Steer motors turnRatio speed 75
lastError = Error

let lastError = 0
let turnRatio = 0
let Kd = 0
let derivative = 0
let Ki = 0
let integral = 0
let Kp = 0
let error = 0
let target = 0
target = 40
Kp = 1
Ki = 1
Kd = 1
error = 0
integral = 0
derivative = 0
lastError = 0
forever(function () {
  error = target - sensors.color1.light(LightIntensityMode.Reflected)
  integral = integral + error
  derivative = error - lastError
  turnRatio = error * Kp + (integral * Ki + derivative * Kd)
  steer motors turnRatio speed 75
  lastError = error
})
Step 1 - remove target variable
We don't need this variable as we are going to now compare sensor 1 against sensor 2, rather than comparing one sensor to a target value. We will then set the Error to Left sensor value - right sensor value

On Start

\[
\begin{align*}
K_p &= 1 \\
K_i &= 1 \\
Error &= 0 \\
Integral &= 0 \\
lastError &= 0 \\
Derivative &= 0 \\
K_d &= 1
\end{align*}
\]

Loop Forever

\[
\begin{align*}
Error &= \text{Left LS value} - \text{Right LS value} \\
Integral &= \text{Integral} + \text{Error} \\
Derivative &= \text{Error} - \text{lastError}
\end{align*}
\]

\[
\begin{align*}
\text{turnRatio} &= (\text{Error} \times K_p) + (\text{Integral} \times K_i) + (\text{Derivative} \times K_d) \\
\text{Steer motors turnRatio speed} &= 75 \\
lastError &= \text{Error}
\end{align*}
\]

Tuning

Step 1 - proportional
- set $K_p = 1$
- set $K_i = 0$, $K_d = 0$

This will focus on proportional and the lower this value, the smoother it should be.

Step 2 - integral
- experiment with $K_i$ until smooth

Step 3 - derivative
```javascript
let lastError = 0
let turnRatio = 0
let Kd = 0
let derivative = 0
let Ki = 0
let integral = 0
let Kp = 0
let error = 0

forever(() => {
  error = sensors.color1.light(LightIntensityMode.Reflected) - sensors.color4.light(LightIntensityMode.Reflected)
  integral = integral + error
  derivative = error - lastError
  turnRatio = error * Kp + (integral * Ki + derivative * Kd)
  motors.largeBC.steer(turnRatio, 50)
  lastError = error
})
```

Getting the right values is difficult isn't it.