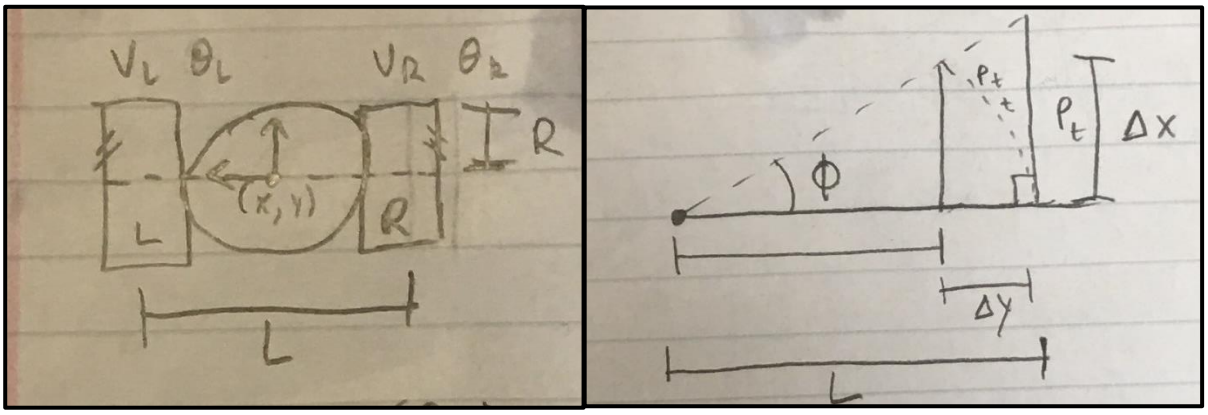
Caleb Taylor

ECE 370

Odometry Report



The analytical method works by constantly updating the value phi based on the baseline, wheel radius, and gear ratio. Delta x and y are found using the baseline and phi values. With each interrupt on either left or right, the global phi value is increased or decreased by phi (depending on which side). Delta prime x and y are found and added to the global values of x and y.

The matrix method works by finding phi and using it within a matrix for both the right and left wheels. These two matrices are translated using the identity matrix and the baseline. With each interrupt, the global matrix is updated by multiplying it with the left or right transformed matrix. The value of delta x, delta y, and z angle are found in the global matrix.

Github Link: https://github.com/ctaylo43/odometry

Analytical Method

```
// Pin Definitions //
#define IRPIN_L 11
#define IRPIN_R 10

// Constants //
#define L 80  // baseline in mm
#define RADIUS  20 // radius of the wheels
#define GEAR_RATIO  75.81 // Motor gear ratio



// Global Variables //
float global_phi = 0, phi = (RADIUS * deg2rad(90 / GEAR_RATIO));
float delta_x = (L / 2) * sin(phi);
float global_x = 0, d_prime_x;
float delta_y = (L / 2) * (1 - cos(phi));
float global_y = 0, d_prime_y;



void setup() {
  Serial.begin(9600);
  pinMode(IRPIN_L, INPUT);
  pinMode(IRPIN_R, INPUT);
  attachInterrupt(digitalPinToInterrupt(IRPIN_L), ISR_L, RISING);
  attachInterrupt(digitalPinToInterrupt(IRPIN_R), ISR_R, RISING);
}

void loop()
{
  float delta_x = (L / 2) * sin(phi);
  float delta_y = (L / 2) * (1 - cos(phi));
}
```

```
void ISR_L()
{
  g_phi += phi;
  d_prime_x = (delta_x * cos(phi)) + (delta_y * sin(phi));
  d_prime_y = (delta_x * sin(phi)) + (delta_y * cos(phi));
  global_x += d_prime_x;
  global_y += d_prime_y;
  Serial(global_x);
  Serial(global_y);
}

void ISR_R()
{
  g_phi -= phi;
  d_prime_x = (delta_x * cos(phi)) + (delta_y * sin(phi));
  d_prime_y = (delta_x * sin(phi)) + (delta_y * cos(phi));
  global_x += d_prime_x;
  global_y += d_prime_y;
  Serial(global_x);
  Serial(global_y);
}
```

Matrix Method

```
#include <BasicLinearAlgebra.h>

// Pin Definitions //
#define IRPIN_L 11
#define IRPIN_R 10

// Constants //
#define L 80  // baseline in mm
#define RADIUS  20 // radius of the wheels
#define GEAR_RATIO  75.81 // Motor gear ratio



// Global Variables //
float phi = (RADIUS * deg2rad(90 / GEAR_RATIO));
double delta_x = 0, delta_y = 0, z = 0;

// Matrices //
Matrix<4,4> right_wheel = {
   cos(phi),  -sin(phi),  0,  0,
   sin(phi),  cos(phi), 0,  L / 2,
   0, 0,  1,  0,
   0, 0,  0,  1};

Matrix<4,4> right_translate = {
  1,  0,  0,  0,
  0,  1,  0,  -L,
  0,  0,  1,  0,
  0,  0,  0,  1};

Matrix<4,4> left_wheel = {
   cos(-phi),  -sin(-phi),  0,  0,
   sin(-phi),  cos(-phi), 0,  -L / 2,
```

```
   0, 0,  1,  0,
   0, 0,  0,  1};

Matrix<4,4> left_translate = {
  1,  0,  0,  0,
  0,  1,  0,  L,
  0,  0,  1,  0,
  0,  0,  0,  1};

Matrix<4,4> global_matrix = {
  1,  0,  0,  0,
  0,  1,  0,  0,
  0,  0,  1,  0,
  0,  0,  0,  1};

Matrix<4,4> right_transform = right_wheel * right_translate;
Matrix<4,4> left_transform = left_wheel * left_translate;

void setup() {
  Serial.begin(9600);
  pinMode(IRPIN_L, INPUT);
  pinMode(IRPIN_R, INPUT);
  attachInterrupt(digitalPinToInterrupt(IRPIN_L), ISR_L, RISING);
  attachInterrupt(digitalPinToInterrupt(IRPIN_R), ISR_R, RISING);
}

void loop()
{
  delta_x = global_matrix(0,4);
  delta_y = global_matrix(1,4);
  z = acos(global_matrix(0,0));
  Serial(delta_x);
  Serial(delta_y);
```

```
  Serial(z);
}

void ISR_L()
{
  global_matrix *= left_transform;
}

void ISR_R()
{
  global_matrix *= right_transform;
}
```