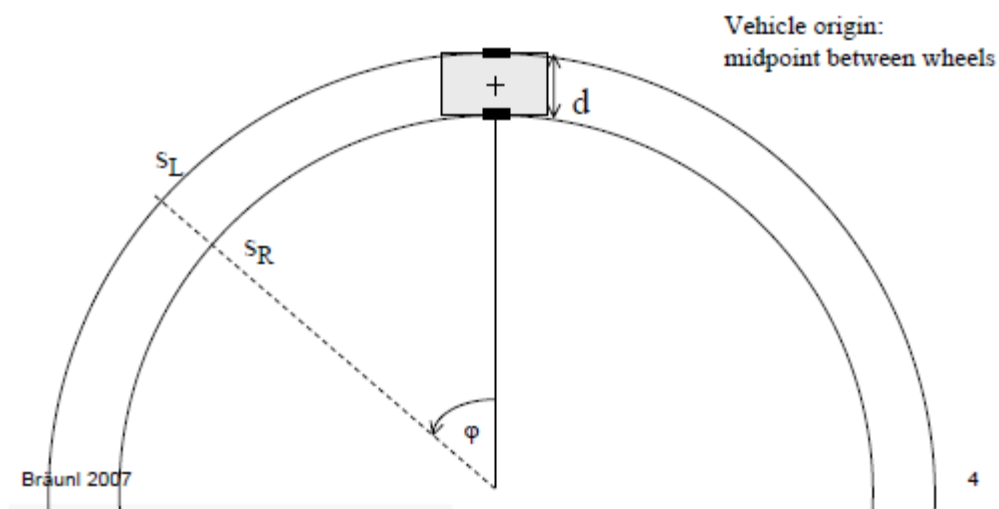


Robotics & Automation

ENGT4314

Tutorial 6

Differential Drive:



- r wheel radius
- d distance between driven wheels
- ticks_per_rev number of encoder ticks for one full wheel revolution
- ticks_L number of ticks during measurement in left encoder
- ticks_R number of ticks during measurement in right encoder

$$s_L = 2\pi \cdot r \cdot \text{ticks}_L / \text{ticks_per_rev}$$

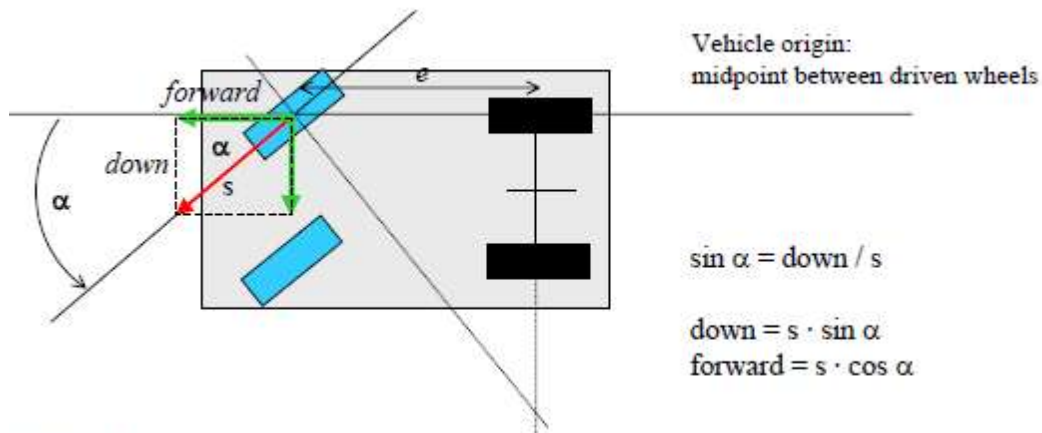
$$s_R = 2\pi \cdot r \cdot \text{ticks}_R / \text{ticks_per_rev}$$

$$\Delta s = (s_L + s_R) / 2$$

$$\Delta \phi = (s_R - s_L) / d$$

Ackerman:

Identical to single driven and steered front wheel



Bräunl 2007

8

- r wheel radius
- e distance front to back wheels
- α front wheel rotation angle, range $[-\pi/2 .. +\pi/2]$, equiv. $[-90 .. +90]$
- ticks_per_rev number of encoder ticks for one full wheel revolution
- ticks number of ticks during measurement in rear encoder

$$s_{\text{Front}} = 2\pi r \cdot \text{ticks} / \text{ticks_per_rev}$$

$$\Delta s = s_{\text{Front}}$$

$$\Delta \varphi = \sin \alpha \cdot s_{\text{Front}} / e$$

Driving Kinematics

Q Given a robot with differential drive with a wheel radius of 2cm, 1024 ticks per revolution and a distance of 8cm between its wheels

1. What is the distance one wheel has traveled in 489 ticks?
2. What is the rotation speed of the wheel if it takes 2 seconds to drive this distance?
3. If the left wheel registers 815 ticks in one second, and the other 880 ticks, what is the angle it is driving about the center of curvature?
4. If this robot is located at (200,200), what is the location of the center of curvature for this robot?

5. What will be the robots new location after “t” seconds?

6. If we wish our robot to move at 5cm / s and at an angle of 0.3rad, what wheel velocities will be required?

PGM Images,

I have updated the pgm c files, so that they are easier to compile. To recap here are the functions that we can use to interface to PGM files.

```
char *img_basename(char *filename);
```

Returns the filename from a path, eg. C:\folder\image.pgm would return image.pgm.
Not particularly useful.

```
IMAGE *img_open(char *filename);
```

Opens a pgm file and returns a pointer to the Image, to be passed to further functions. Much the same as the other file operations, we need to check to see if the returned pointer is NULL. If it is then an error occurred which would be printed to the console window.

```
IMAGE *img_creat(char *name,int nr, int nc);
```

Allocates and creates a new image in memory with the following properties:

Name – name of the image usually you can just pass the path of where you will store the image.

nr – number of rows in the image.

nc – number of columns

Note: This function does not actually save the image to disk or create a file to store the image too.

```
void img_setpixel(IMAGE *img, int r, int c, int val);
```

Sets the value of an individual pixel in an image loaded into ram,

Img – pointer to the image to be modified.

r – pixel row.

c – pixel coloumn

val – value to be stored in the pixel, will accept 0-255. 0 is black and 255 white.

```
int img_getpixel(IMAGE *img, int r, int c);
```

Similar to above, returns a pixel value.

Img – pointer to the image to be modified.

r – pixel row.

c – pixel coloumn

```
int img_write(IMAGE *img, char *filename);
```

Saves the image to disk.

Img – pointer to the image to be modified.

filename – string containing the path/filename of where you wish to store the image.

Note any pixels that are outside of the 0-255 range will be set to 0.

```
void img_free(IMAGE *img);
```

Releases the memory associated with an image, important to do after you have finished with an image to prevent memory leaks (May be checked in the lab as part of the marking key).

There are also a couple of macros available.

```
int ROWS(IMAGE *img)
```

Returns the number of rows for the passed image.

```
int COLS(IMAGE *img)
```

Returns the number of columns

```
#define NAME(img)
```

Returns the name (filename).

Problem:

To get you off to a start in lab4, lets write a short piece of code that loads in a PGM file, prints it to the screen (# for dark characters, and “ “ for white). At the same time lets create a new image that is an inverted copy of the original and save that to a new file.

Solution:

1. $\frac{489}{1024} * 2\pi * 2cm = 6.00 cm$
2. $\frac{489}{1024} * \frac{1}{2s} * 2\pi * 2 = 3 cm/s$

3. We need to calculate the distance rotated by each wheel first

$$s_L = \frac{815}{1024} * 2\pi * 2cm = 10.0 cm$$

$$s_R = \frac{880}{1024} * 2\pi * 2cm = 10.799 cm$$

$$\Delta\phi = \frac{s_R - s_L}{d} = \frac{10.799 - 10.00}{8} = 0.1 rad$$

$$\approx 5.72^\circ$$

4. So to calculate the mid-point we can use the relationship $l = r\phi$

$$l = \frac{(10.799 + 10)}{2} = 10.4$$

$$r = \frac{l}{\phi} = \frac{10.4}{0.1} = 104cm$$

Assuming the initial robot $\phi = \frac{\pi}{2}$ (robot is positioned along the y-axis (If the robot were to drive forward it would drive along the y-axis in a positive direction).

The midpoint will thus be perpendicular to the robot and on the left side.

Hence the center of curvature will be (-96,200).

5. Thus we can calculate the position of the robot at time t by using basic trig.

$$P = (x_c + r * \sin\left(\frac{\pi}{2} - 0.05t\right), y_c + r * \cos\left(\frac{\pi}{2} - 0.05t\right))$$

6. We can easily calculate the inverse kinematics via the formula provided in lectures

$$\begin{pmatrix} \theta_L \\ \theta_R \end{pmatrix} = \frac{1}{2\pi r} \begin{pmatrix} 1 & -d/2 \\ 1 & d/2 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}$$

$$\theta_L = \frac{1}{2\pi r} \left(v - \frac{d}{2} \omega \right) = \frac{1}{4\pi} \left(5 - \frac{8}{2} 0.3 \right) = 0.3 rad/s$$

$$\theta_R = \frac{1}{2\pi r} \left(v + \frac{d}{2} \omega \right) = \frac{1}{4\pi} \left(5 + \frac{8}{2} 0.3 \right) = 0.5 rad/s$$