How to Understand a Swerve Drive

Team 3863
Gabe Millikan
May 2019

1 Using Interactive Visualization Software **3**
1.1 Installing Python 3 3
1.2 Installing PIP 4
1.3 Installing OpenCV 4
1.4 Using the Software 5
1.4.1 Setting it Up 5
1.4.2 What's What 5
1.4.3 How to Change Values 6
2 Vector Basics **6**
2.1 What is a Vector? 6
2.2 How to Represent a Vector 7
2.2.1 Cartesian Notation 7
2.2.2 Polar Notation 8
2.3 Convert Between Representations 8
2.3.1 Polar to Cartesian 8
2.3.2 Cartesian to Polar 8
2.4 Vector Math 9
2.4.1 Addition 9
3 Swerve theory **10**
3.1 Using Vectors With Modules 10
3.2 Translation 10
3.3 Rotation **11**
3.3.1 Basics 11
3.3.2 Regular Polygon 12
3.3.3 Rectangles 13
3.3.4 How to Calculate Wheel Direction 14
3.4 Concurrent Rotation and Translation 15
3.4.1 The Idea 15
3.4.2 How To 15
3.4.3 Visualize 15
3.5 Special Cases 15
3.5.1 Non-Center Pivot Point 15
3.5.2 Non Equidistant Wheels. 15
References **16**

1 Using Interactive Visualization Software

This section will explain how to use the software provided at
https://github.com/GabeMillikan/Swerve

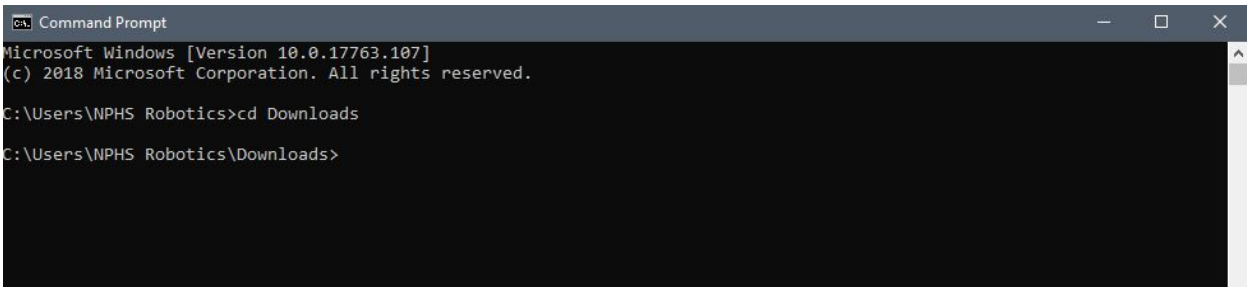
1.1 Installing Python 3

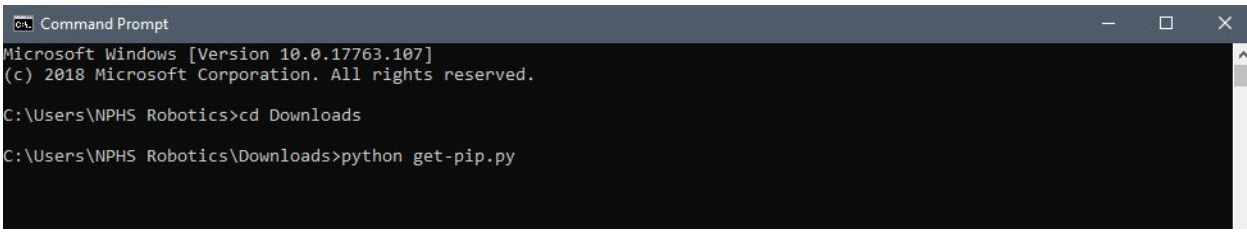
If you already have Python 3 installed, you may skip this section.

1. Go to https://www.python.org/downloads/
2. Download the latest of version 3, as of May 17, 2019 that is Python 3.7.3
3. Run the installer
4. Add Python to PATH, then install
   a.

1.2 Installing PIP

If you already have pip installed, then you can skip this section.

1. Save this file somewhere on your computer https://bootstrap.pypa.io/get-pip.py
2. Change directories to where you downloaded it
   a. 

3. Run this command: python get-pip.py
   a. 

4. Allow that to finish and you should see “Successfully installed pip-xx.x.x”

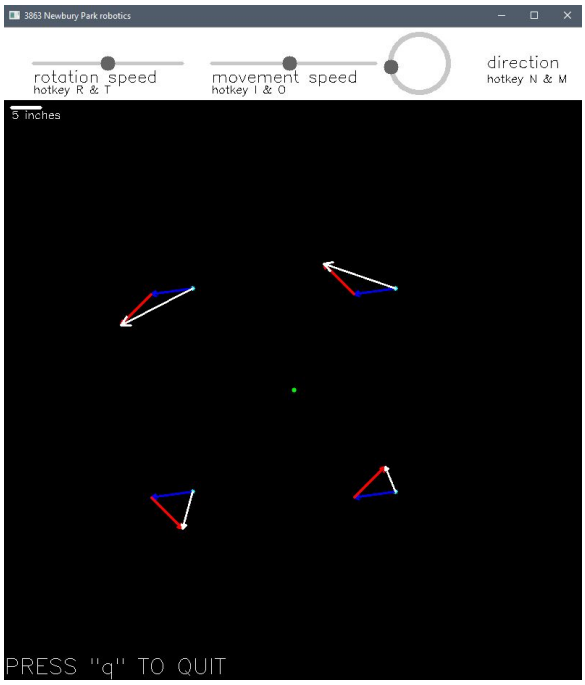
1.3 Installing OpenCV

In a command window, type “pip install opencv-python”

1.4 Using the Software

1.4.1 Setting it Up

Download the .py file from https://github.com/GabeMillikan/Swerve.
To run the program, you can double click the file, but if that fails, open a command prompt, change directory to where you downloaded the file, and use the command "python swerveDriveVisual.py"
At this point, you should be seeing something like this:



1.4.2 What's What

In the default setup, the red arrow represents each wheel's rotation vector, more on that later. And the blue arrow is each wheel's translation vector, more on that later. The white vector is the final combination of both for each wheel, more on that later.
At the top, the left slider represents the overall rotation speed of the robot, where left is the slowest rotation, and right is the fastest rotation.
The middle slider is the movement speed, this represents the translation velocity of the robot.
The right "joystick" represents the direction of translation of the robot.

1.4.3 How to Change Values

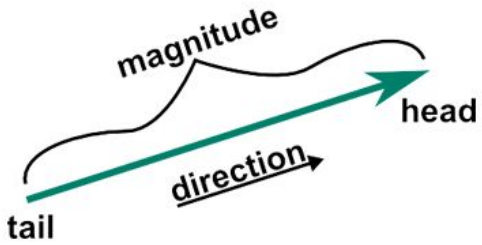
To change the rate of rotation, the left slider, press “R” and “T” on the keyboard.
To change the robot velocity, the middle slider, press “I” and “O” on the keyboard.
To change the translation direction, the right dial, press “M” and “N” on the keyboard.

At the top of swerveDriveVisual.py, you will find some values and explanations, feel free to adjust these to your liking.

2 Vector Basics

2.1 What is a Vector?

A vector is a line that has direction and length, called magnitude. This is a basic vector[1]:



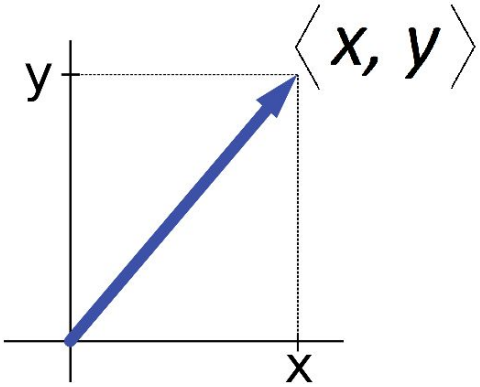
Here is what a vector looks like in my software:

2.2 How to Represent a Vector

There are two ways of representing a vector that are required for making a swerve drive.

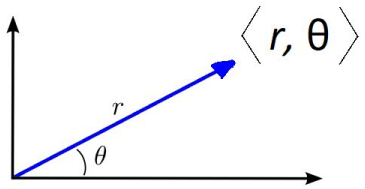
2.2.1 Cartesian Notation

Also called “Rectangular Notation”, this breaks up the vector into its X and Y components.



2.2.2 Polar Notation

Polar notation represents a vector as its “r” (magnitude)(length) and “θ” (theta) (angle).



2.3 Convert Between Representations

2.3.1 Polar to Cartesian

To convert from Polar (r, θ) to Cartesian (x, y):

$$X = r * cos(\theta)$$

$$Y = r * sin(\theta)$$

2.3.2 Cartesian to Polar

To convert from Cartesian (x, y) to Polar (r, θ):

$$r = \sqrt{x^2 + y^2}$$

$$\theta = atan2(y/x)$$

Notice here that *atan2* is the *arctangent* of *y/x*, then corrected to the corresponding quadrant

2.4 Vector Math

How to perform basic operations on vectors.

2.4.1 Addition

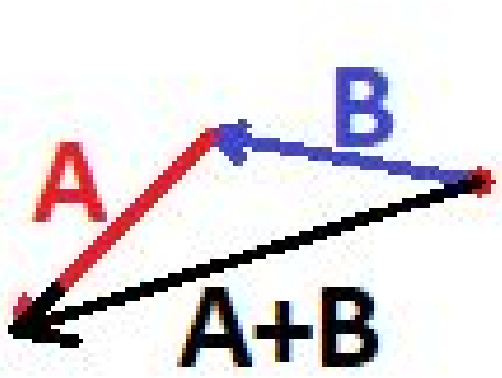
To add vectors, it is a simple as adding their individual x and y components. If the vectors are in polar, first convert them to cartesian.

$A = (1, 2)$

$B = (3, 4)$

$V = A + B$

$V = (1 + 3,\ 2 + 4) = (4,\ 6)$



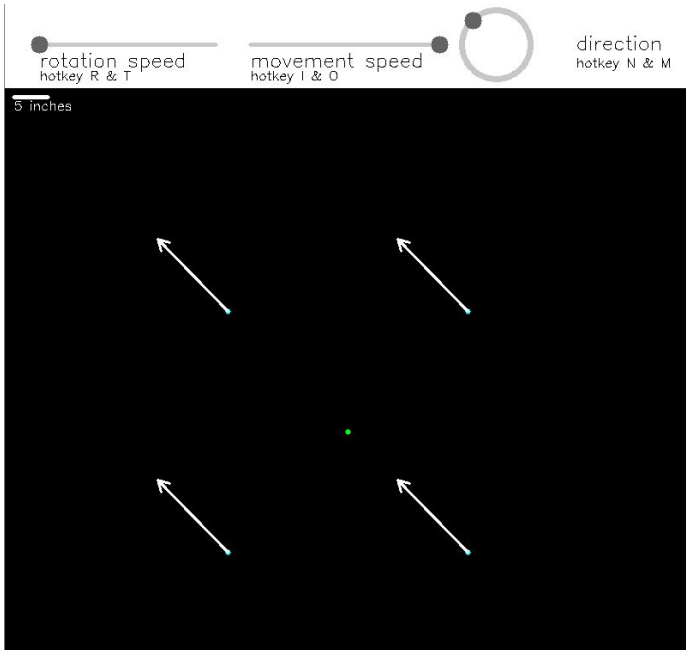
3 Swerve theory

3.1 Using Vectors With Modules

Each swerve module has two values, its rotation, and velocity.
This can be represented as a vector in polar form, where magnitude is velocity and angle is rotation. For example, a module with a velocity of 10 m/s at an angle of 45 degrees would be (10, 45) in vector form.

3.2 Translation

To translate a robot without rotation, all of the wheels must point in the same direction, and have the same velocity. To visualize this, set the “rotation speed” to zero (on the left) and set the “movement speed” to its maximum(on the right).

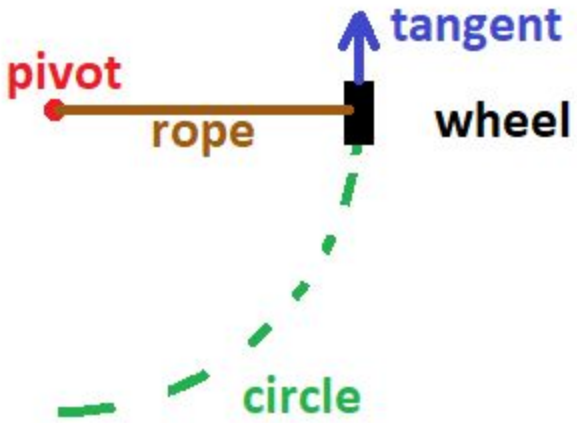


From here, you can change the direction of translation (dial on the right) to see how the vectors change.
You can also change the movement speed to see its effect on the vectors.
Remember, the direction of the vectors is the direction of movement, and the length of them is the robot’s speed.

3.3 Rotation

3.3.1 Basics

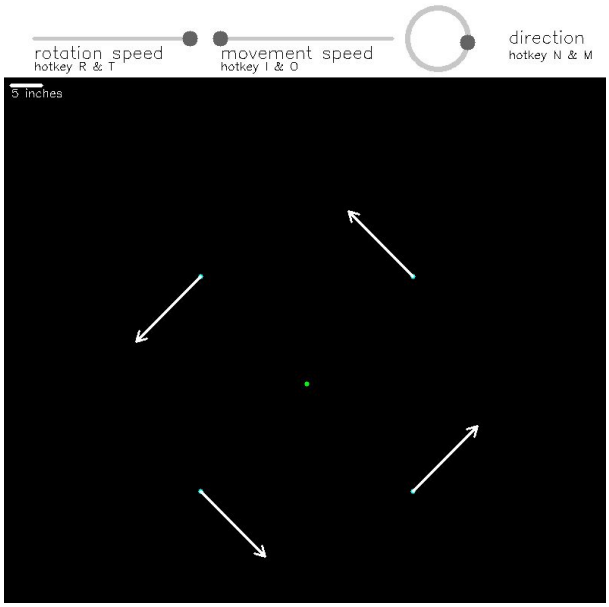
To rotate a robot without movement, all the wheels must trace a circle with equal center points. Imagine a wheel attached to a string attached to a post. When activated, the wheel will spin around and make a circle. That wheel is “tangent” to that circle.



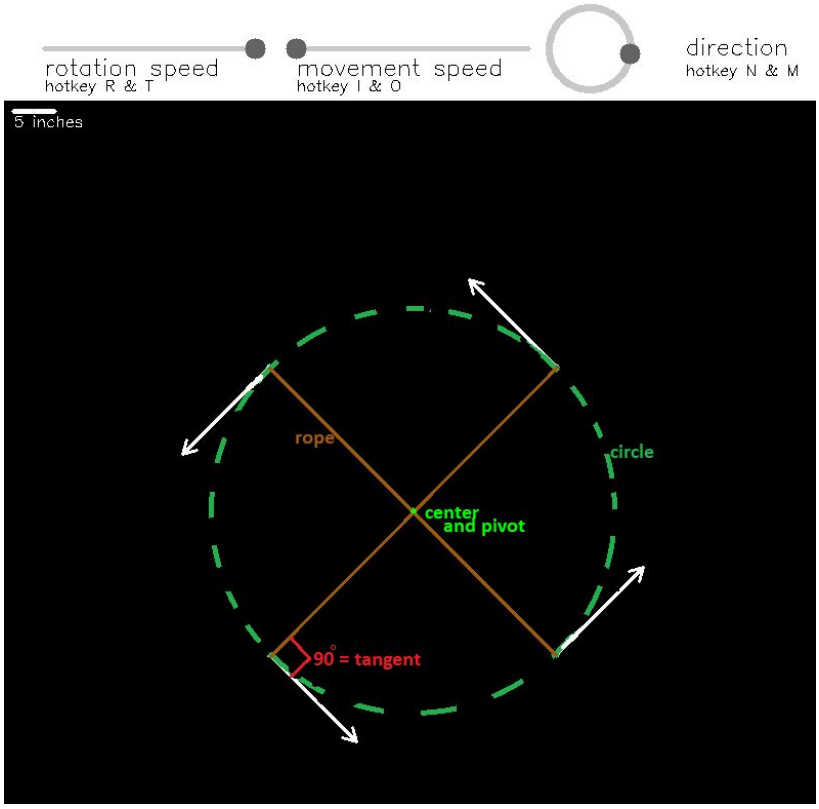
Now imagine that the wheel was facing away from the pivot point, that wheel is no longer tangent, and will not make a circle or do anything productive. That is not what we want, so keep in mind that we always want the wheels to be **tangent** to the pivot point.

3.3.2 Regular Polygon

Understanding rotation of a robot is best done visually. To see this, set “movement speed” to 0 (on the left) and set rotation speed to its maximum (on the right).



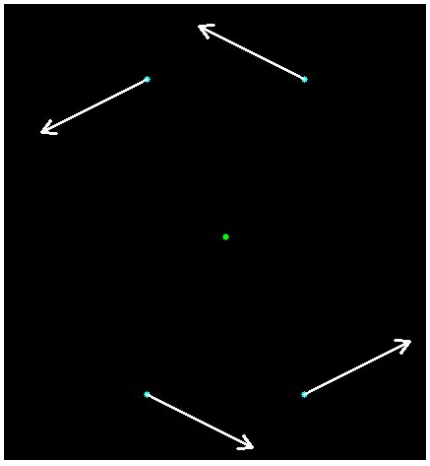
Looking at it, it makes sense, right? All the wheels will form a circle, and each wheel is tangent to the center.



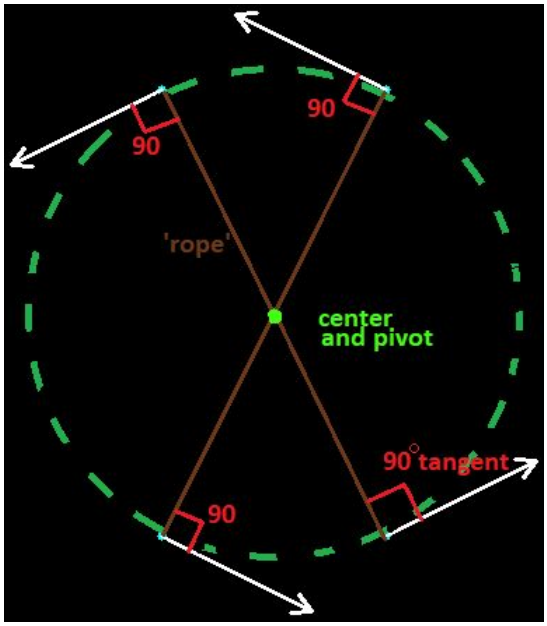
Also, the movement direction has no effect on the vectors. This makes sense because as long as the movement speed is 0, the movement should have no effect on the final angles and velocities of the wheels, $0 * x = 0$.

3.3.3 Rectangles

For rectangles, change the `RobotSize` variable in the program. For this example, I will change it to RobotSize = (20,40). If your robot size does not fit on the screen, change the `ViewScale` variable until it does.
Now, set rotation speed to its max, and movement speed to 0, like section 3.3.2.
The robot looks like this:



It no longer looks as simple as the square robot, but even here, all the wheels are tangent to the rotation pivot.

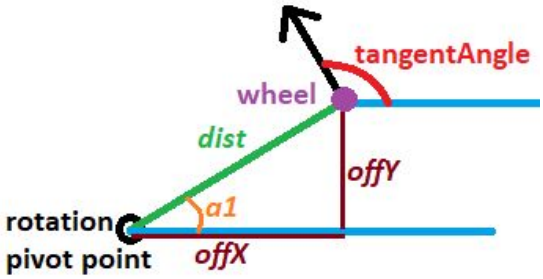


3.3.4 How to Calculate Wheel Direction

As explained in section 3.3.1, the direction of the wheel without movement is equal to the tangent angle to the pivot point.
In most cases, the pivot point is the center of the robot, but there is no real reason why it has to be. More on that in section 3.5.
Here is the formula to find the tangent angle(in degrees) from the pivot point:

$$tangentAngle = atan2(offY/offX) + 90$$

Where:



"a1" in the figure is the angle from the point of rotation. This is the first part of the equation, $atan2(offY/offX)$. The final direction of the wheel is perpendicular to this angle, which explains the second half of the equation $"+90"$.
$atan2$ is inverse tangent of the legs of the triangle, and the "2" means quadrant correction. In java, you use $Math.atan2(y, x) + 3.14159/2;$ Notice that it is 3.14159/2, and not 90; this is because Math functions use radians, not degrees, but they are equivalent.

3.4 Concurrent Rotation and Translation

This section will explain how to apply your knowledge from section 3.2 and 3.3.

3.4.1 The Idea

Once you understand sections 3.2 and 3.3, this is actually surprisingly simple. From section 3.2, you will have a vector for each wheel that represents the translation of the robot. From section 3.3 you will have one for each wheel representing the rotation. To both at the same time, you just have to add each vector together. For example, you can add the front right's translation vector with its rotation vector, and you will have your final vector to apply.

3.4.2 How To

It is not very easy to add polar vectors like the ones we have, so first convert each vector into rectangular form using section 2.3.1, then add them with section 2.4.1, then convert it back to polar using section 2.3.2. Now for each wheel you will have a polar vector, where you can then apply it to each respective wheel.

3.4.3 Visualize

To visualize this, set the rotation speed and movement speed so they are both non-zero. By default, the red arrow is rotation, the blue is translation, and white is final.

3.5 Special Cases

Weird settings, i'm not sure if any of this section is accurate, so take it with a grain of salt.

3.5.1 Non-Center Pivot Point

I would like to say that I have not tried this, and don't really know how it works, but the speed of each wheel will change because each is not the same distance from the center, so the circle that each wheel traces will not have the same circumference. This ties in closely with section 3.5.2.

3.5.2 Non Equidistant Wheels.

When rotating, each wheel traces a circle. If a wheel is not the same distance from the center, the wheel will trace a different circle. Since the circle has a different circumference, the wheel will have to adjust speeds to keep up with or slow down to the other wheels. Wheels closer to the pivot point will have a lower velocity because they travel less per rotation. The opposite is true for wheels further away from the center.

References

[1] image from https://www.sawaal.com/