Joystick to Tank Drive conversion.

Introduction

When it comes to remote control robotics, or tele-operated machines as some robotics purists may call them, one of the most common drive systems is what is referred to as "Tank Drive". This is where the drive wheels are in fixed positions, and steering is controlled by speeding up or slowing the wheels on one side of the robot relative to the other. This can be found in two, four, six, even eight wheeled robots, robots with tank treads, and some of these concepts can even be applied to legged robots as well as control interface in game programming.

Likewise, one of the most common control systems is a two axis joystick, typically a computer joystick, or a joystick found on an R/C (Radio control) transmitter.

This page will not consider drive systems where the drive wheels or non-driven wheels are steerable, or other control systems such as separate sticks.

Some of this might be a bit basic for experienced programmers and designers, but I'm writing for a broad audience, and a grounding on the simple stuff will make sure we're all on the same footing when it gets complex.

The goal here is to come up with a totally simple formula that will change the X and Y values from a joystick into L and R values for the drive wheels. For that we need to nail down what we want those L and R values to be based on the X and Y values we have coming in, and we want a formula with no ifs, or exceptions, or special cases, and with smooth, linear transitions.

Basic Concepts

When considering a solution to this problem. I set out from a vague, brute-force idea, with some background knowledge that inspired my solution. I think that explaining the process by which I came to this solution, rather than simply giving out the formulas, you can better understand why this works, and thus be able to customize and optimize this solution for your own situation.

The first thing is to figure out what your desired result is. You need to know where you want to go before you can set out to go there.

Traditional, Two Stick Tank Control, as one might find in WWII and earlier tanks, worked with two gigantic clutch levers in the driver's hands. He'd push them both forward to go straight ahead, pull both backwards to back up, and push one forward and one back to rotate in place, one way or the other. You could represent this in a truth table like so.

Sticks	**Right Forward**	**Right Backward**
Left Forward	Drive Forward	Rotate Right
Left Backward	Rotate Left	Drive Backward

From experience with "Antweight" BattleBots, I can tell you that the simplest form of conversion from Joystick to Tank Drive involves no computation, and no electronic mixing. Simply physically hold the transmitter at a 45 degree angle to the left, and the top right corner becomes "Drive Forward"

Joystick	**Left**	**Right**
Up	Rotate Left	Forward
Down	Backward	Rotate Right

At least, that's how it usually works, depending on which channels you have connected to your left and right drive. Moving the stick up and down makes the left side of the robot's drivetrain go forwards and back, and moving the stick left to right makes the right side of the robot's drivetrain go forwards and back.

If you look at these tables, you can see the data is rotated one space clockwise. This is a result of how R/C joysticks are traditionally set up. On the Up/Down axis, up is a higher value, and on the Left/Right axis, right is a higher value, and in speed controllers, a high value means forward. You could make the tables match if you had Servo reversing in your R/C transmitter. But we're not looking to make a canted joystick our solution, and this doesn't work for a computer joystick anyway.

What we want to do is rotate these tables halfway.

Joystick	**Left**	**Center**	**Right**
Up		Drive Forward	
Center	Rotate Left	Stop	Rotate Right
Down		Drive Back	

You'll notice I've added a little something there, the center position. It's there in the other tables, if you imagine it. But that's not important, because this is what we've been looking for as a control input. Joysticks spring back to the center, and we want to be stopped. We want to go forward when the stick is pushed forward, back when it's pulled back, and we want to back up. We want to turn left and right when we move the stick from side to side. In short, we want each axis of the joystick to control a function. Forward and back are speed, and left and right are steering. You don't have that with a canted R/C joystick.

Let's fill in that table a bit more.

Joystick	**Left**	**Center**	**Right**
Up	Turn Left	Drive Forward	Turn Right
Center	Rotate Left	Stop	Rotate Right
Down	Backing Turn Left	Drive Back	Backing Turn Right

(One side note, the backing turns would be opposite if we had a car-steering system. And the rotate positions wouldn't change our heading at all since the drive wheels wouldn't be turning at all. This is something to keep in mind if you're using this formula to model vehicle control in a game.)

Okay, so this table depicts what is happening to the vehicle as we manipulate the joystick. But the speed controllers are actually controlling the drive wheels. So what does this mean for where the rubber meets the road? Let's have yet another table.

Joystick	**Left**	**Center**	**Right**
Up	1. Right Wheels Ahead Left Wheels Stopped	2. Right Wheels Ahead Left Wheels Ahead	3. Right Wheels Stopped Left Wheels Ahead
Center	4. Right Wheels Ahead Left Wheels Reverse	5. Right Wheels Stopped Left Wheels Stopped	6. Right Wheels Reverse Left Wheels Ahead
Down	7. Right Wheels Stopped Left Wheels Reverse	8. Right Wheels Reverse Left Wheels Reverse	9. Right Wheels Reverse Left Wheels Stopped

The numbers are for reference. Now, some of this might seem a little confusing, particularly the corner cases. Other parts, like the center column, should make perfect sense. Let's address the confusion. For example, if you want to turn left, why is it the right side the goes forward? Here's a simple exercise: Stand up. Put your right foot forward. Straighten your hips. You should now be facing more to the left than you were. You can do the same thing stepping back with your left foot. On a robot, the effect is more about which point the robot is rotating about. In the case of 4 and 6, the robot tends to rotate about its center. In 1,3,7 and 9, it tends to rotate around the stopped side. (This is especially true in a two-drive-wheel robot).

Right now, all this table shows is all or nothing drive. Full speed, stopped, or reverse. This might be valid if we were just using buttons and relays to control the robot. That's a pretty jerky way to control a robot. And in the real world, the joystick and speed controller havecontinuously variable values.

Getting more complex.

First, let's take this table, and assign values to the drive. 100 for 100% speed ahead, 0 for stopped, and -100 for 100% speed backwards.

Joystick	Left	Center	Right
Up	Right 100 Left 0	Right 100 Left 100	Right 0 Left 100
Center	Right 100 Left -100	Right 0 Left 0	Right -100 Left 100
Down	Right 0 Left -100	Right -100 Left -100	Right -100 Left 0

Next, let's expand the table a bit, adding rows and columns for halfway, and since it's halfway, let's fill the new spaces in with 50% speeds. If a new cell is between two 100's, it should be filled with a 100, if it's between a 100 and a 0, put in 50. If it's between 0 and -100, put in -50. And let's see what we get.

Joystick	Left	Half left	Center	Half Right	Right
Up	Right 100 Left 0	Right 100 Left 50	Right 100 Left 100	Right 50 Left 100	Right 0 Left 100
Half Up	Right 100 Left -50	Right 50 Left 0	Right 50 Left 50	Right 0 Left 50	Right -50 Left 100
Center	Right 100 Left -100	Right 50 Left -50	Right 0 Left 0	Right -50 Left 50	Right -100 Left 100
Half Down	Right 50 Left -100	Right 0 Left -50	Right -50 Left -50	Right -50 Left 0	Right -100 Left 50
Down	Right 0 Left -100	Right -50 Left -100	Right -100 Left -100	Right -100 Left -50	Right -100 Left 0

The new blue cells all make a nice transition with the white cells they've been added between. Looking at the center row and center column, it couldn't be a simpler transition between stopped and full forward or reverse. That should lead to a nice simple formula. It's like a straight line.

The edges of the table are a trifle trickier. Taking the Joystick Up row as an example, it goes from both wheels on full at the center, and drops off one or the other as you go from left to right. This isn't as simple to come up with a formula for. But it's still kind of linear in the sections where it's changing.

And when we get to the Red cells, well, sticking in a 50 doesn't seem right at all, even if the joystick IS halfway over. Going down the Half Left column, the Right value goes from 100, to 50, stays at 50, then goes to 0 before going to -50. That's not very linear. Imagine if we stuck in even more rows. We'd get 100, 75, 50, 50, 50, 25, 0, -25, -50. That string of 3 50's in a row... that's not smooth.

Clearly, in those red squares, even if the stick is halfway, a 75 would work a lot better. When I was first working this out on paper, I drew circles on the chart I made. That gave me a critical insight that even though I've got a grid here, what I'm looking at is more like a target. Those red squares are a different ring than the inner blue ones. And once I put in 75, it worked.

So, here's the new chart. This is the output we want going to the drive wheels for various positions of the joystick. After all this mess, this is the output we want from our formula.

Joystick	**Left**	**Half left**	**Center**	**Half Right**	**Right**
Up	Right 100 Left 0	Right 100 Left 50	Right 100 Left 100	Right 50 Left 100	Right 0 Left 100
Half Up	Right 100 Left -50	Right 75 Left 0	Right 50 Left 50	Right 0 Left 75	Right -50 Left 100
Center	Right 100 Left -100	Right 50 Left -50	Right 0 Left 0	Right -50 Left 50	Right -100 Left 100
Half Down	Right 50 Left -100	Right 0 Left -75	Right -50 Left -50	Right -75 Left 0	Right -100 Left 50
Down	Right 0 Left -100	Right -50 Left -100	Right -100 Left -100	Right -100 Left -50	Right -100 Left 0

So, that's all well and good, but how do you get a formula for this?

The trick is to look for patterns.

The other trick is to realize that we're looking for two formulas. Or rather, the same formula, but run twice for left and right. Our Formula will have two inputs, for the X and Y values of the joystick, and one output for either the left or the right wheels.

Pattern Recognition

So, to help figure this out, lets split that table into left and right outputs.

Left					
Joystick	**Left**	**Half left**	**Center**	**Half Right**	**Right**
Up	0	50	100	100	100
Half Up	-50	0	50	75	100
Center	-100	-50	0	50	100
Half Down	-100	-75	-50	0	50
Down	-100	-100	-100	-50	0

Right					
Joystick	**Left**	**Half left**	**Center**	**Half Right**	**Right**
Up	100	100	100	50	0
Half Up	100	75	50	0	-50
Center	100	50	0	-50	-100
Half Down	50	0	-50	-75	-100
Down	0	-50	-100	-100	-100

A couple of things should leap right out at you. First, they are mirror images of each other. Second, there's that nice diagonal row of 0's, and diagonal rows of 50's and -50's on either side of it. It almost makes you wish that the center positions were 75 instead of 100 so you'd have more diagonal rows.

Now, let's take a moment to check out the input side of the equation. Technically, depending on the system, a joystick might produce values from -255 to 255, or -127 to 128, or even 0 to 255. Or, sometimes they fall just short of those limits. Getting into that would be a real distraction at this point, so I'm going to use a percentage from -100% to 100%, and talk about scaling at the end.

Joystick X					
Joystick	**Left**	**Half left**	**Center**	**Half Right**	**Right**
Up	-100	-50	0	50	100
Half Up	-100	-50	0	50	100
Center	-100	-50	0	50	100
Half Down	-100	-50	0	50	100
Down	-100	-50	0	50	100

Joystick Y					
Joystick	**Left**	**Half left**	**Center**	**Half Right**	**Right**
Up	100	100	100	100	100
Half Up	50	50	50	50	50
Center	0	0	0	0	0
Half Down	-50	-50	-50	-505	-50
Down	-100	-100	-100	-100	-100

Hmmm, there's another pattern that should leap out at you. Vertical and horizontal rows of 0's, and other numbers. Man, if we could somehow average those together and get a diagonal pattern, wouldn't that be nice?

Let's try bashing these tables together various ways and see if we can find anything useful.

As a little bit of background, a long time ago I learned about how they added stereo to FM radio (It wasn't always in stereo), and they had to add it in a way that was compatible with Mono. What they did was add the left and right audio signals together for the Mono, and then broadcast, at a slightly higher frequency, a signal that was the right audio MINUS the left audio. By adding or subtracting this difference signal from the combined signal, they could derive the separate left and right audio. It's pretty impressive especially if you consider they did it in analog, and in real time, with vacuum tubes. There's something to be learned from old technology. When I was working this out, that little factoid was in the back of my head, and so the idea of combining left and right signals applied itself to this problem

So, here are two tables, one of right PLUS left, and one of right MINUS left. Let's see what patterns we can find in this.

R + L				
100	150	200	150	100
50	75	100	75	50
0	0	0	0	0
-50	-75	-100	-75	-50
-100	-150	-200	-150	-100

R - L				
100	50	0	-50	-100
150	75	0	-75	-150
200	100	0	-100	-200
150	75	0	-75	-150
100	50	0	-50	-100

Check it out, horizontal and vertical lines of 0's. That could be useful....

Let's try that FM Stereo trick and see if we can change these tables back into the originals.

(R+L - R-L) / 2 = Left Wheel Drive				
0	50	100	100	100
-50	0	50	75	100
-100	-50	0	50	100
-100	-75	-50	0	50
-100	-100	-100	-50	0

(R+L + R-L) / 2 = Right Wheel Drive				
100	100	100	50	0
100	75	50	0	-50
100	50	0	-50	-100
50	0	-50	-75	-100
0	-50	-100	-100	-100

Okay, so that's the last half of the formula. Let's see what we can do to the joystick values to make them into R+L and R-L and get the first half of the formula.

Well, first thing I see is that R-L has all the negatives on the right side of the table, but the Joystick X has them on the left. So let's invert X. That would be 0 - X. And just for repetition's sake, here's Joystick Y.

Joystick X Inverted				
100	50	0	-50	-100
100	50	0	-50	-100
100	50	0	-50	-100
100	50	0	-50	-100
100	50	0	-50	-100

Joystick Y				
100	100	100	100	100
50	50	50	50	50
0	0	0	0	0
-50	-50	-50	-50	-50
-100	-100	-100	-100	-100

Now, how to massage these into R+L? Well, look for the patterns. On R+L the first thing you see is the row of 0's. So it's just like Joystick Y. Then you see at the left and right, it's the same as Joystick Y, but as it gets closer to the center, it doubles. This is the OPPOSITE of Joystick X. And the only negatives are on the bottom half. So the sign comes from Joystick Y. We don't need any negatives from X.

So, how do we make X into the opposite, and get rid of the negatives? First, we get rid of the negatives with the Absolute Value function: ABS(x), and then, we subtract X from its maximum value. So part of the formula is (100 - ABS(X)). Let's apply that to the table and see what it looks like.

100-ABS(X)				
0	50	100	50	0
0	50	100	50	0
0	50	100	50	0
0	50	100	50	0
0	50	100	50	0

Now that looks closer. If we just added this to Y, the top row would look right, but everything else would be wrong.

100-ABS(X) +Y				
100	150	200	150	100
50	100	150	100	50
0	50	100	50	0
-50	0	50	0	-50
-100	-50	0	-50	-100

Yeah, that's totally wrong. Clearly the amount of X to add in varies with the amount of Y. When Y is 0, there's no X, when Y is 100, we want to add X, and when Y is -100, we want to subtract X. Now remember when I said that these numbers were percentages? I've been leaving the % sign off. When you want to multiply percentages, you have to express them as decimals. 100% becomes 1.00. 50% becomes 0.50 (When you multiply by 50% or .5, you get half of the original value. Right?) And when we multiply by -100% or -1, we will be subtracting instead of adding.

So, to make the amount of 100 - ABS(X) we're adding in vary with Y, let's multiply it by Y expressed as a percentage (that's Y/100).

R+L = (100-ABS(X)) * (Y/100) + Y				
100	150	200	150	100
50	75	100	75	50
0	0	0	0	0
-50	-75	-100	-75	-50
-100	-150	-200	-150	-100

It works!

Now for R-L. But here's a hint, it's the exact same formula, but you switch X and Y. That's why I first inverted the X value, so the negatives would be on the right side.

R-L = (100-ABS(Y)) * (X/100) + X				
100	50	0	-50	-100
150	75	0	-75	-150
200	100	0	-100	-200
150	75	0	-75	-150
100	50	0	-50	-100

So to get the value for the speed of the right wheels, you add these two values together, and to get the speed for the left wheels, you'd subtract the second from the first.

Final Formula

So, one thing that might be confusing is that I've been doing all these tables with 25 numbers in then, when you've only got one X and one Y, and you only need one L and one R. Do you really have to calculate all those? No. This isn't matrix algebra or anything like that. Those are just a bunch of calculations in parallel, and the position in the chart corresponds to the position of the joystick. Doing a bunch of them at the same time is the only way to make sure that the formula I derived was the right one.

But your Robot Controller doesn't know what a Joystick is. It just knows numbers. Those numbers could even come from a pair of knobs, like a steering wheel and a throttle control.

Your robot may have some numbers that I don't know. That's why I expressed the values as a percentage while working this out. You have some scaling to do, especially if those maximum joystick values have nothing to do with the maximum input your speed controllers can take. (And here's a hint, R/C transmitter sticks rarely make it to 100% of the range the signal can carry. You can have the stick all the way forward and still only be running the controllers at 80% of full speed).

Scaling and matching to your control system is beyond the scope of this article. But at the most basic level, you can substitute the maximum value you get out of a joystick for 100 wherever I've used it.

For now, my imaginary joystick goes from -100 to 100 in X and Y, and my imaginary Speed Controllers take -100 to 100.

So here's what I do in some kind of processing loop in my robot program.

1. Get X and Y from the Joystick, do whatever scaling and calibrating you need to do based on your hardware.
2. Invert X
3. Calculate R+L (Call it V): V =(100-ABS(X)) * (Y/100) + Y
4. Calculate R-L (Call it W): W= (100-ABS(Y)) * (X/100) + X
5. Calculate R: R = (V+W) /2

6. Calculate L: L= (V-W)/2
7. Do any scaling on R and L your hardware may require.
8. Send those values to your Robot.
9. Go back to 1.

So why didn't I just give you this to begin with? Because just giving people answers without understanding where they come from leaves them unable to figure out other answers for themselves. For example, suppose you wanted to use the joystick to simulate car-like behavior? Knowing this formula wouldn't help you. But knowing HOW I got this formula, you can apply the same process to come up with a different one. Here's a hint. Full left and right joystick with no forward or reverse stands still, rather than spinning in place. In fact, cars can't spin in place like a tank drive vehicle, so plan your output accordingly.

Errata

I did all this work in an Excel spreadsheet. To verify that I got the formulas correct, I did it with an even bigger table, adding columns for +/- 25% and 75% throws on the Joystick.

X								
100	75	50	25	0	-25	-50	-75	-100
100	75	50	25	0	-25	-50	-75	-100
100	75	50	25	0	-25	-50	-75	-100
100	75	50	25	0	-25	-50	-75	-100
100	75	50	25	0	-25	-50	-75	-100
100	75	50	25	0	-25	-50	-75	-100
100	75	50	25	0	-25	-50	-75	-100
100	75	50	25	0	-25	-50	-75	-100
100	75	50	25	0	-25	-50	-75	-100

Y								
100	100	100	100	100	100	100	100	100
75	75	75	75	75	75	75	75	75
50	50	50	50	50	50	50	50	50
25	25	25	25	25	25	25	25	25
0	0	0	0	0	0	0	0	0
-25	-25	-25	-25	-25	-25	-25	-25	-25
-50	-50	-50	-50	-50	-50	-50	-50	-50
-75	-75	-75	-75	-75	-75	-75	-75	-75
-100	-100	-100	-100	-100	-100	-100	-100	-100

V								
100	125	150	175	200	175	150	125	100
75	93.75	112.5	131.25	150	131.25	112.5	93.75	75
50	62.5	75	87.5	100	87.5	75	62.5	50
25	31.25	37.5	43.75	50	43.75	37.5	31.25	25
0	0	0	0	0	0	0	0	0
-25	-31.25	-37.5	-43.75	-50	-43.75	-37.5	-31.25	-25
-50	-62.5	-75	-87.5	-100	-87.5	-75	-62.5	-50
-75	-93.75	-112.5	-131.25	-150	-131.25	-112.5	-93.75	-75
-100	-125	-150	-175	-200	-175	-150	-125	-100

W								
100	75	50	25	0	-25	-50	-75	-100
125	93.75	62.5	31.25	0	-31.25	-62.5	-93.75	-125
150	112.5	75	37.5	0	-37.5	-75	-112.5	-150
175	131.25	87.5	43.75	0	-43.75	-87.5	-131.25	-175
200	150	100	50	0	-50	-100	-150	-200
175	131.25	87.5	43.75	0	-43.75	-87.5	-131.25	-175
150	112.5	75	37.5	0	-37.5	-75	-112.5	-150
125	93.75	62.5	31.25	0	-31.25	-62.5	-93.75	-125
100	75	50	25	0	-25	-50	-75	-100

Right								
100	100	100	100	100	75	50	25	0
100	93.75	87.5	81.25	75	50	25	0	-25
100	87.5	75	62.5	50	25	0	-25	-50
100	81.25	62.5	43.75	25	0	-25	-50	-75
100	75	50	25	0	-25	-50	-75	-100
75	50	25	0	-25	-43.75	-62.5	-81.25	-100
50	25	0	-25	-50	-62.5	-75	-87.5	-100
25	0	-25	-50	-75	-81.25	-87.5	-93.75	-100
0	-25	-50	-75	-100	-100	-100	-100	-100

Left								
0	25	50	75	100	100	100	100	100
-25	0	25	50	75	81.25	87.5	93.75	100
-50	-25	0	25	50	62.5	75	87.5	100
-75	-50	-25	0	25	43.75	62.5	81.25	100
-100	-75	-50	-25	0	25	50	75	100
-100	-81.25	-62.5	-43.75	-25	0	25	50	75
-100	-87.5	-75	-62.5	-50	-25	0	25	50
-100	-93.75	-87.5	-81.25	-75	-50	-25	0	25
-100	-100	-100	-100	-100	-75	-50	-25	0

As I said, you don't need to calculate all these values at once. On the other hand, for some robots, where they have little processing power, but a lot of memory, it actually might be easier to calculate these tables in advance for all possible values, and then just look them up based on X and Y. For a simple formula like this, that technique probably isn't worth it, but for other control systems, big tables with tweaked numbers can make things a lot simpler and faster.