Models & Transforms

Alexander Koldy

Forest Hills Robotics League

alexanderkoldy.ak@gmail.com

September 18, 2024

Introduction

In this lecture, we will use geometry, linear algebra and physics to discuss *FIRST Tech Challenge* (FTC) robotics-related mathematics. The topics this lecture will cover are:

- Coordinate frames
- Robot state representations
- Transforms
- Drivetrain kinematics

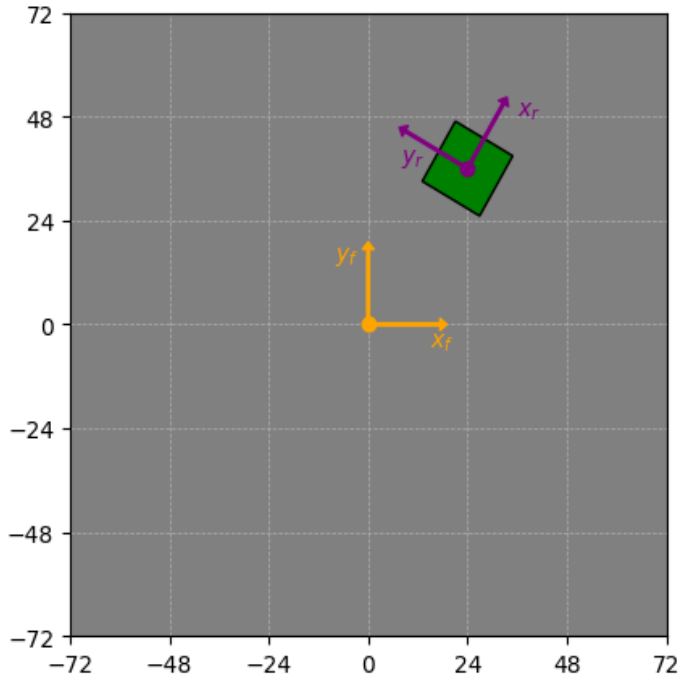
Note: Due to the high popularity of mecanum drivetrains in FTC, we will be omitting tank, swerve and other types of drives for now

Coordinate frames

In modern FTC, we have three main coordinate frames to worry about:

- **Field** (or "global") frame - orange
- **Robot** (or "body") frame - purple
- **Camera** frame

For this lecture, we will worry about the field frame and robot frame in $\mathbb{R}^2$. Incorporating the camera frame (assuming the camera is not tilted), simply builds upon the concepts discussed later.



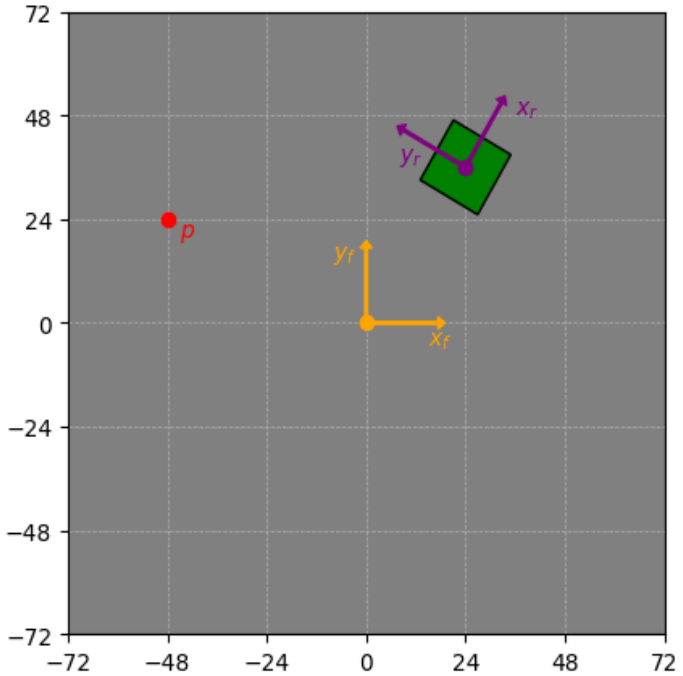
Note: technically, there is a z axis pointing upwards in each frame. We call these frames "right handed" coordinate frames.

Transforms - relativity

Let's introduce some point p on the field. What is the position of p *relative* to each frame?

Well, for the field frame, the position of p is obvious! The origin of our field frame is at $(0, 0)$ so the position of p relative to frame f is

$$^{f}p = \begin{bmatrix} -48 & 24 \end{bmatrix}^{\top}$$

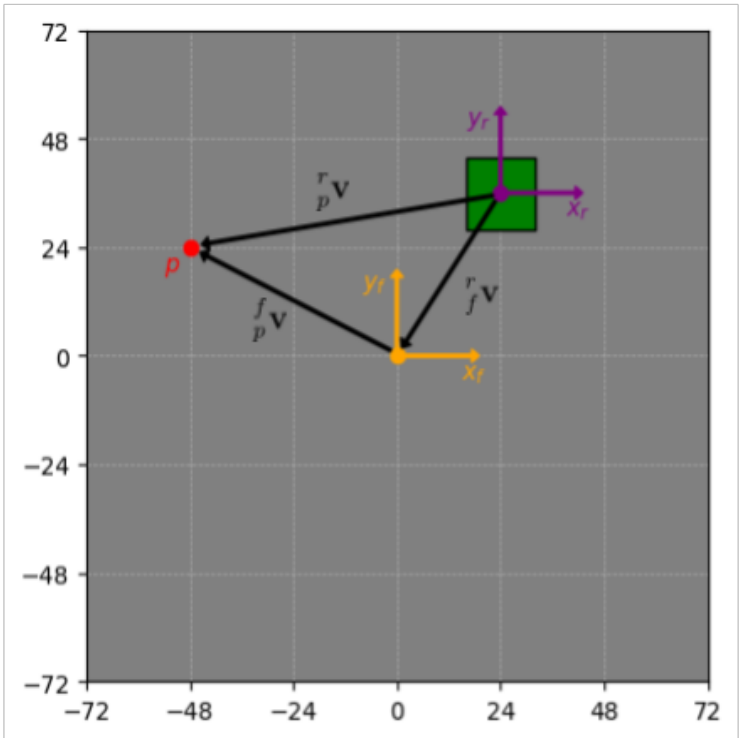


However, the position of p relative to the robot is a bit trickier. Since the robot is moving, and turning, its frame also moves and turns. Thus, the origin and rotation of frame r are not constant!

Let's try to work through finding ^{r}p or point p relative to frame r.

Transforms - translations

Let's simplify the problem and align the rotation of the robot frame to the field frame. We construct vectors ${}^{r}_{f}\mathbf{v}$, which is the negative of the vector defining the position of the robot relative to the field frame and ${}^{f}_{p}\mathbf{v}$, which is the position of p relative to the field frame. From these, we can easily figure out ${}^{r}_{p}\mathbf{v}$, which is the position of point p relative to the robot.

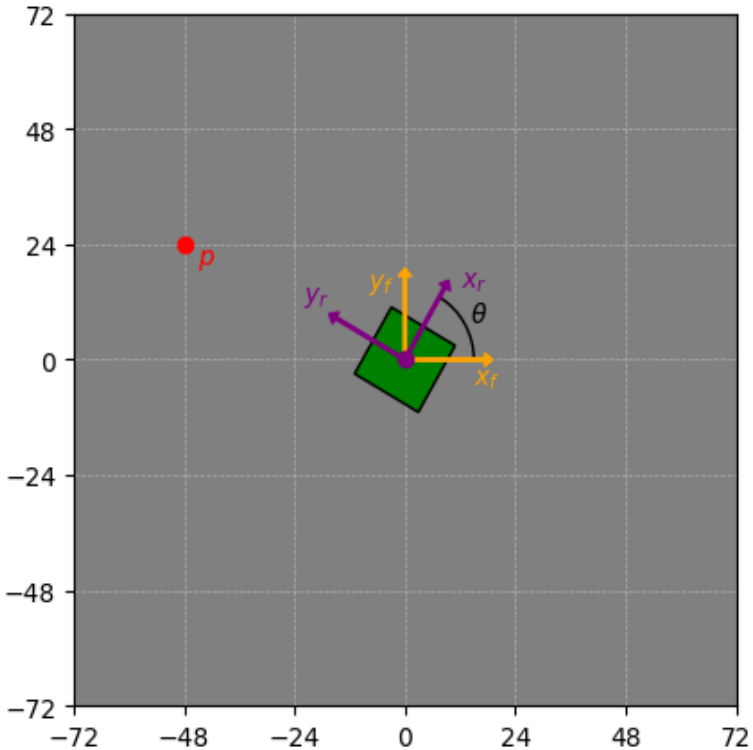


Graphically, using tip-to-tail, we see

$$ {}^{r}_{p}\mathbf{v} = ({}^{r}_{f}\mathbf{v}) + ({}^{f}_{p}\mathbf{v}) = \begin{bmatrix} -24 \\ -36 \end{bmatrix} + \begin{bmatrix} -48 \\ 24 \end{bmatrix} = \begin{bmatrix} -72 \\ -12 \end{bmatrix} $$

Transforms - rotations

Let's try another simplified example, where the origins of each frame are aligned, but the robot frame is rotated θ counter-clockwise relative to the field frame. We can find the position of point p in the r frame by applying a **rotation matrix**. The derivation of this matrix is simple, and requires some trigonometry, but it is outside the scope of this lecture. Check out this video for a derivation.



The rotation matrix that defines the rotation from robot frame to the field frame is defined by

$$\mathbf{R}_{r \to f} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Transforms - rotations

The opposite (the rotation from the field frame to the robot frame) can be found by inverting this matrix. Since this matrix is **orthogonal**, the inverse is equal to it's transpose:

$$\mathbf{R}_{f\to r} = (\mathbf{R}_{r\to f})^{-1} = (\mathbf{R}_{r\to f})^{\top}$$

We can therefore find the vector representing p relative to the robot frame with:

$${}^{r}_{p}\mathbf{v} = (\mathbf{R}_{r\to f})^{-1}\left({}^{f}_{p}\mathbf{v}\right)$$

Note: in some literature, you may see the rotation from frame r to frame f written as ${}^{r}\mathbf{R}_{f}$ or $\mathbf{R}^{r}_{f}$

In the case of our robot being rotated 60° our representation or p relative to frame r is

$${}^{r}_{p}\mathbf{v} = \mathbf{R}_{r\to f} = \begin{bmatrix} \cos(60) & \sin(60) \\ -\sin(60) & \cos(60) \end{bmatrix} \begin{bmatrix} -48 \\ 24 \end{bmatrix} = \begin{bmatrix} -3.21 \\ 53.57 \end{bmatrix}$$

Transforms

We can actually represent a rotation and a translation in two dimensional space with one matrix! Let's define the transform from the robot frame to the field frame as

$$\mathbf{T}_{r\to f} = \begin{bmatrix} \mathbf{R}_{f\to r} & \mathbf{t}_{f\to r} \\ \mathbf{0} & 1 \end{bmatrix}$$

where $\mathbf{0}$ is a row vector of zeros of size 2 and $\mathbf{t}$ represents position of the robot frame relative to the field frame. We can also use a transformation matrix to represent a transform from the field frame to the robot frame, however we need to be careful and consider the rotation of the robot frame in our translation. This "inverse" transform looks as follows:

$$\mathbf{T}_{f\to r} = \begin{bmatrix} (\mathbf{R}_{f\to r})^{-1} & -(\mathbf{R}_{f\to r})^{-1}\,\mathbf{t}_{f\to r} \\ \mathbf{0} & 1 \end{bmatrix}$$

Using this with our original example on slide 4, we get the position of point p relative to frame r is

$$\begin{bmatrix} {}^{r}_{p}\mathbf{v} \\ 1 \end{bmatrix} = \mathbf{T}_{f\to r} \begin{bmatrix} {}^{f}_{p}\mathbf{v} \\ 1 \end{bmatrix} = \begin{bmatrix} -12 \\ 72 \\ 1 \end{bmatrix}$$

Note: we append a 1 *to the vectors to ensure consistent shaping*

State variables

In FTC, we typically like to keep track of our robot's position (in the field frame) and heading. From now on, let's denote our **pose** to be

$$\mathbf{q} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Moreover, we may also want to keep track of the rate of change of these variables, i.e., the velocities. Oftentimes in robotics we keep track of our positions in the global (field) frame and our velocities in the body (robot) frame. We will call the total representation of our velocities the **twist**

$$\boldsymbol{\xi} = \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

Our robot's **state** is simply the pose and the twist combined into one vector, i.e.,

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \boldsymbol{\xi} \end{bmatrix} = \begin{bmatrix} x & y & \theta & v_x & v_y & \omega \end{bmatrix}^\top$$

Note: in code, it may be a good idea to create some classes for these for easy organization

Drivetrain kinematics

Let's discuss the relationship between our "inputs" to the robot and "output" response. We will define our inputs as

$$\mathbf{u} = \begin{bmatrix} \omega_{lf} \\ \omega_{lb} \\ \omega_{rb} \\ \omega_{rf} \end{bmatrix}$$

Where each ω is a wheel speed corresponding to the left-front, left-back, right-back, right-front wheels respectively. Our output will be our twist:

$$\boldsymbol{\xi} = \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

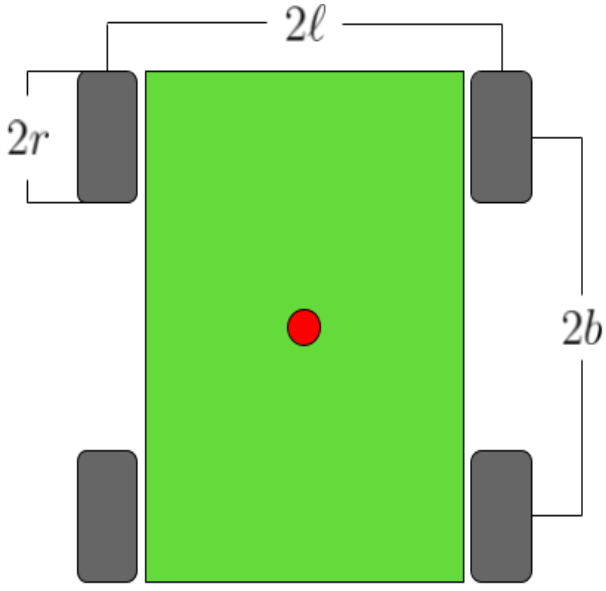
Let's also define our track width (distance between opposing wheels) as 2ℓ and our wheelbase (distance between adjacent wheels) as $2b$. Our wheel radius is r.

Note: for a more detailed derivation of drivetrain kinematics see this paper

Drivetrain kinematics

Our **forward kinematics** define our robot's velocities in the body frame (twist) based off of its wheel speeds

$$\underbrace{\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}}_{\boldsymbol{\xi}} = \frac{r}{4} \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -\frac{1}{\ell+b} & -\frac{1}{\ell+b} & \frac{1}{\ell+b} & \frac{1}{\ell+b} \end{bmatrix}}_{\mathbf{H}_{fk}} \underbrace{\begin{bmatrix} \omega_{lf} \\ \omega_{lb} \\ \omega_{rb} \\ \omega_{rf} \end{bmatrix}}_{\mathbf{u}}$$



Our **inverse kinematics** yield the opposite result:

$$\underbrace{\begin{bmatrix} \omega_{lf} \\ \omega_{lb} \\ \omega_{rb} \\ \omega_{rf} \end{bmatrix}}_{\mathbf{u}} = \frac{1}{r} \underbrace{\begin{bmatrix} 1 & -1 & -(\ell+b) \\ 1 & 1 & -(\ell+b) \\ 1 & -1 & (\ell+b) \\ 1 & 1 & (\ell+b) \end{bmatrix}}_{\mathbf{H}_{ik}} \underbrace{\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}}_{\boldsymbol{\xi}}$$

Note: we can also multiply both sides by Δt to yield the forward and inverse kinematics based off of relative position changes in the robot and wheels