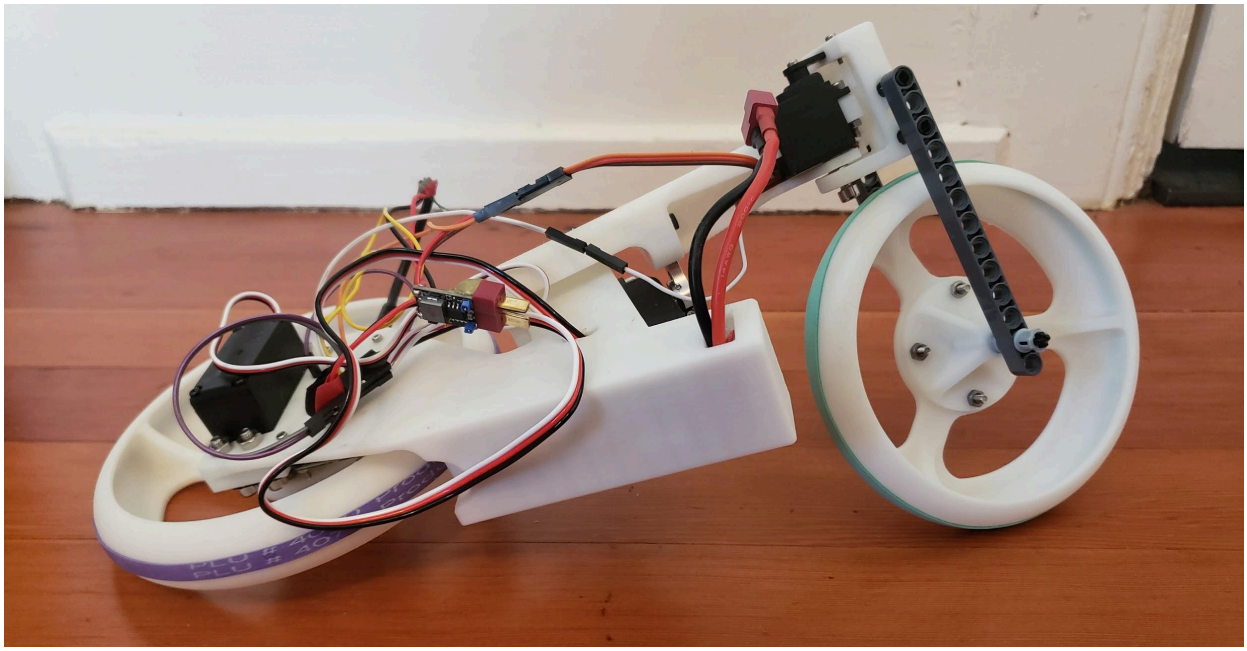
Robocycle

Arthur Lovekin
08-16-2024



Introduction

Let me paint you a picture of what I want the future of transportation to look like:
I'm at school and want to get to the soccer field two miles away. I open up an uber-like app and hail an autonomous bicycle, which promptly arrives at my location in only a few minutes. Once I get on, I am fully in control. Autonomous collaborative balancing is not something I need in my life. But as soon as I arrive at the field, I hop off and the bike goes off to the next rider. No clutter on the sidewalk, time and money well-spent.

Why a bicycle? Well for one thing their lightweight frame and only two contact points with the ground make bicycles the most efficient ground vehicle that can successfully navigate a city. They are able to move as quickly as a car in an urban setting, but their lower profile makes them much more nimble and less dangerous in the case of a collision. Humans are also able to ride them safely and intuitively, which opens new opportunities unavailable to other small robot delivery vehicles. In short, autonomous bicycles fit in the perfect niche of efficiency, safety, speed, and human-friendliness that make them highly desirable in a community.
Use Cases:

- Very cool personal bicycle
- Convenient, clutter-free bike-share programs (with all the advantages of autonomous vehicles)
- Faster, more efficient robot delivery services (better than Starship or Kiwibots)
- City-scale inspections/mapping/security

- For academics in robotics, the fact that this vehicle is non-trivial (harder than a cartpole) but still analytically approachable (not as bad as a humanoid) make it an excellent platform to test control algorithms both in hardware and in simulation.

Now you may be thinking, I like this idea, but wouldn't the bicycle just fall over and get stuck? That is certainly the primary limitation of a normal bicycle, but not so with this design! By putting a joint roughly coaxial with the where the bottom-tube would normally be, this bike is able to passively rest in a half-upright position (you kind of have to see it to believe it). With a good controller, it can swivel into the upright position, and once it is moving balance is easily obtained through steering, as a human would do. Notably using the joints this way is entirely novel as far as I'm aware. Existing autonomous bikes rely on flywheels (heavy and clunky), and/or do not include a mechanism to stand up from a fallen down position.

With all this in mind, my goals for this project are to design a bike that can (1) automatically right itself from any position on the ground, and (2) drive autonomously (without person onboard), so that riders can summon it and it can do deliveries. The primary challenges are to design hardware that is physically capable of righting itself, creating a balancing controller, and finally building a perception, planning, and control stack that can effectively navigate autonomously. As of August 8, 2024 I've completed a physics (pybullet) simulation and a small physical model that prove the concept, but have yet to build the larger system.

Previous work

Overall, it seems like that bicycles are an understudied area, probably because the financial reward to effort ratio is very low. Furthermore, no one has tackled the problem that bikes in their fallen position are unable to get up. That being said, here are some very helpful resources I've found that cover the bicycle dynamics and other autonomous bicycle attempts.

Bicycle Dynamics: There seems to be a singular great survey paper of bicycle dynamics from TU Delft Bicycle Dynamics (tudelft.nl) (Meijaard et.al. 2007) Linearized dynamics equations for the balance and steer of a bicycle: a benchmark and review

Two-wheeled balancing system:
Ascento [2005.11435] Ascento: A Two-Wheeled Jumping Robot (arxiv.org) This is Ascento Pro (youtube.com)

Revolutionary design from Google ;) : Introducing the self-driving bicycle in the Netherlands (youtube.com)

Autonomous Bicycle using Flywheel: Self Driving Bicycle That Can Run Without The Need For Humans - YouTube

Yamaha Motoroid (same joint configuration, but training wheels keep it from lying down…)
ヤマハ発動機「MOTOROiD2（モトロイド ツー）」のパフォーマンス - YouTube
Self Balancing Autonomous Yamaha Motobot Motorcycle (youtube.com)

Autonomous bike with two wheels (not as cool/practical as my geometry but same large-scale goal)
Overview ‹ The MIT Autonomous Bicycle Project — MIT Media Lab

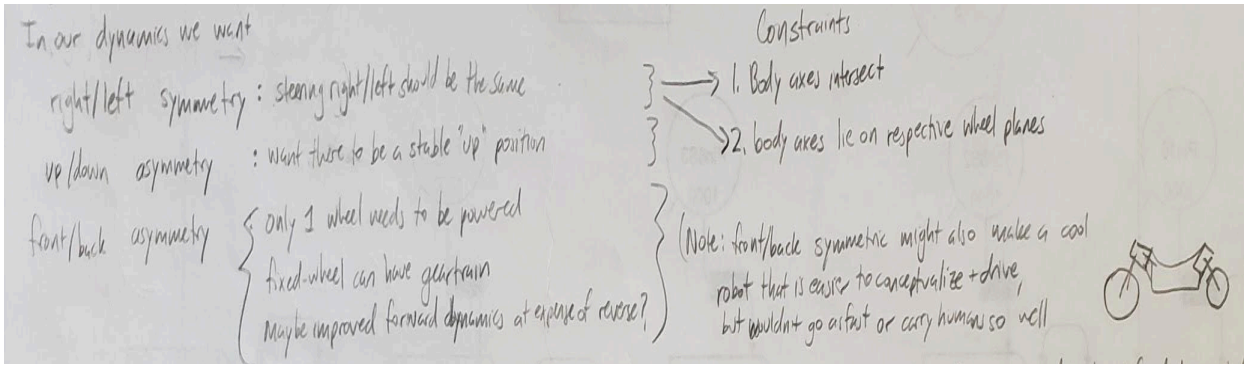
Underactuated Robotics (Rus Tedrake MIT) has a lot of useful information on how I'd actually control this thing.

Dynamics of a Rolling Disk The rolling disk | Rotations (berkeley.edu) is helpful for thinking about wheel contact

Geometric derivation

The goal of this derivation is to arrive at an analytical model of the robocycle system that can then be used by a controller to balance the bike through all stages of movement: stand-up and riding. I'll approach this by first writing the kinematic equations, which describe the geometric structure (eg. link lengths and angles), and then writing the dynamics equations. For this system, the largest challenge arises from dealing with the point-contact between the wheels and the ground. In previous work (Meijaard et.al. 2007) they use a set of coordinates that assumes the contact points are fixed. This is not an option for the robocycle as it stands up. However, all is not lost!

As described in Chapter 18 of the MIT Underactuated Robotics textbook We begin by calculating the kinematic equations. At the most abstract level, this system consists of four angular joints – body, steering, and two wheel axes – and four links – front wheel, rear wheel, body link, and steering link. We can impose some constraints so that these joints and links are related in the way we want. Note that a variety of geometries with this joint configuration are possible, including a normal bicycle, and also a vehicle where both wheels are steerable (this may be interesting to pursue in the future!).



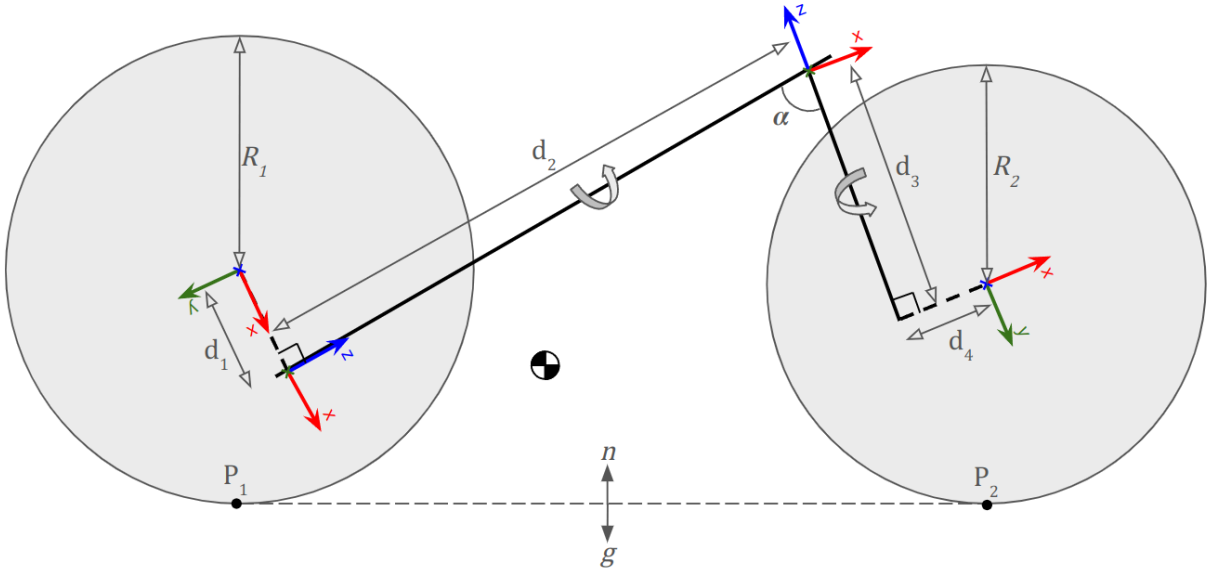
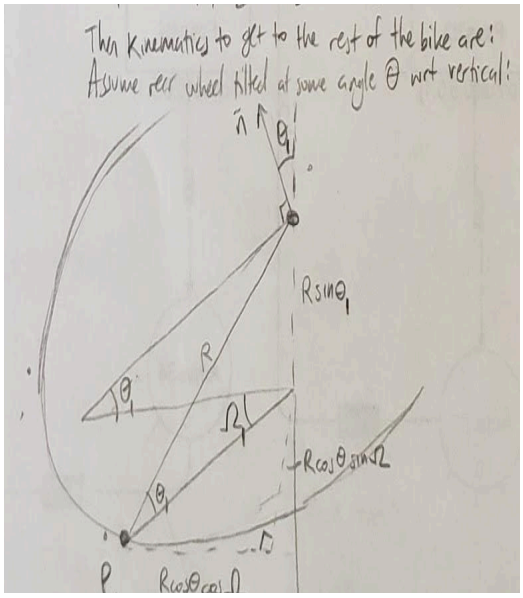


Figure 1: the purely geometric features include the link lengths (d_1,d_2,d_3,d_4), the wheel radii (R_1,R_2), and the angle between the joint axes α. Each link also has an associated coordinate system, where I've put the z-axes along joint axes and x-axes along the common-normal between the wheel frames and body frames. This choice is somewhat arbitrary, but follows conventions similar to the Denavit–Hartenberg rules. Dynamic parameters include the ground-plane normal vector *n*, the gravity vector *g*, the center of mass, and the contact points (P_1, P_2). Given a particular joint configuration and orientation of the robocycle with respect to the ground plane, and assuming that the ground touches at least one wheel, P_1 and P_2 can be found by taking the *z* vector for the particular wheel, and finding $Rz \times (z \times n)$.
Once these contact points are determined, the kinematic chain of the bicycle from wheel center to wheel center can be calculated using trigonometry, assuming the link lengths and angles are known. I haven't written these out for the whole bike yet, but here's one wheel:



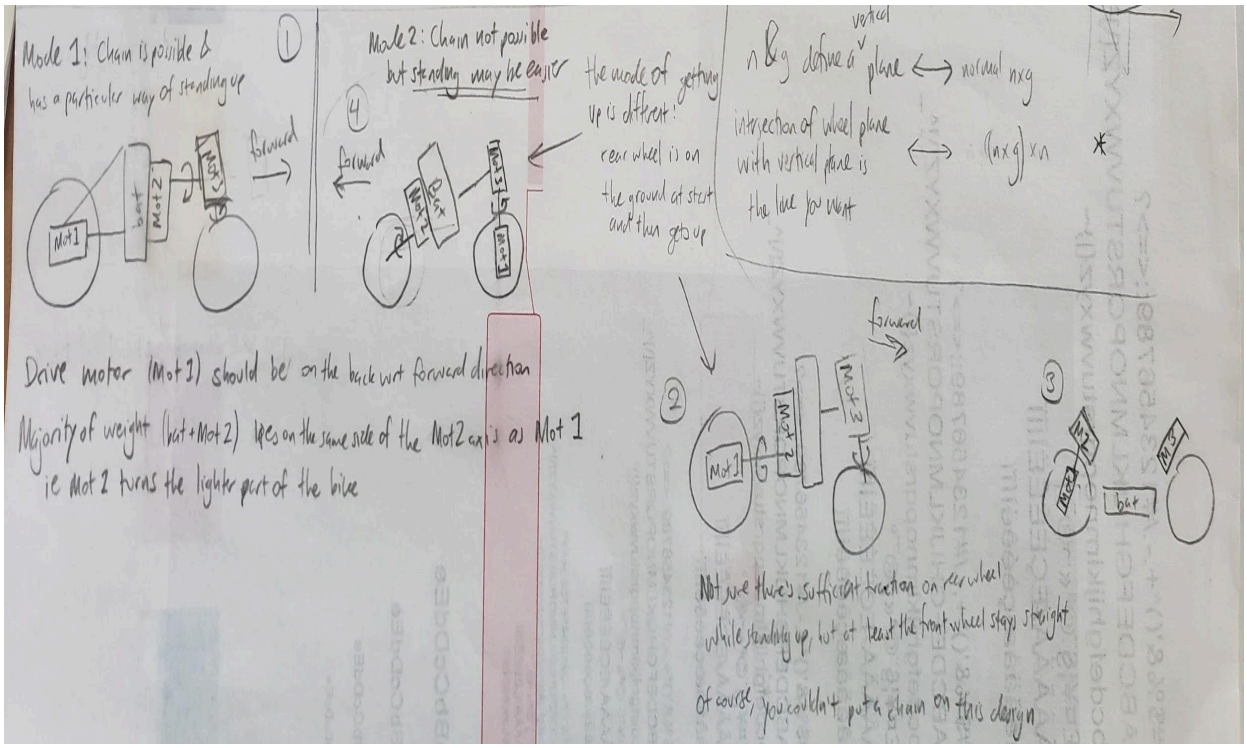
With these kinematic calculations satisfied, the next step is to find the dynamics. Specifically, I need the Center of Mass (COM) and moments of inertia of each link, and then need to do conservation of momentum and energy or equivalently the method of Lagrange and assemble it into one large Manipulator Equation. This is still a work in progress.

Continuing our abstractions, we find that the equations of motion of a general robotic manipulator (without kinematic loops) take the form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \tau_g(\mathbf{q}) + \mathbf{B}\mathbf{u}, \tag{3}$$

where $\mathbf{q}$ is the joint position vector, $\mathbf{M}$ is the inertia matrix, $\mathbf{C}$ captures Coriolis forces, and τ_g is the gravity vector. The matrix $\mathbf{B}$ maps control inputs $\mathbf{u}$ into generalized forces. Note that we pair $\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}}$ together on the left side because these terms together represent the force of inertia; the contribution from kinetic energy to the Lagrangian:

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\mathbf{q}}}^T - \frac{\partial T}{\partial \mathbf{q}}^T = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}.$$



Alternative motor/battery and joint-configuration layouts.

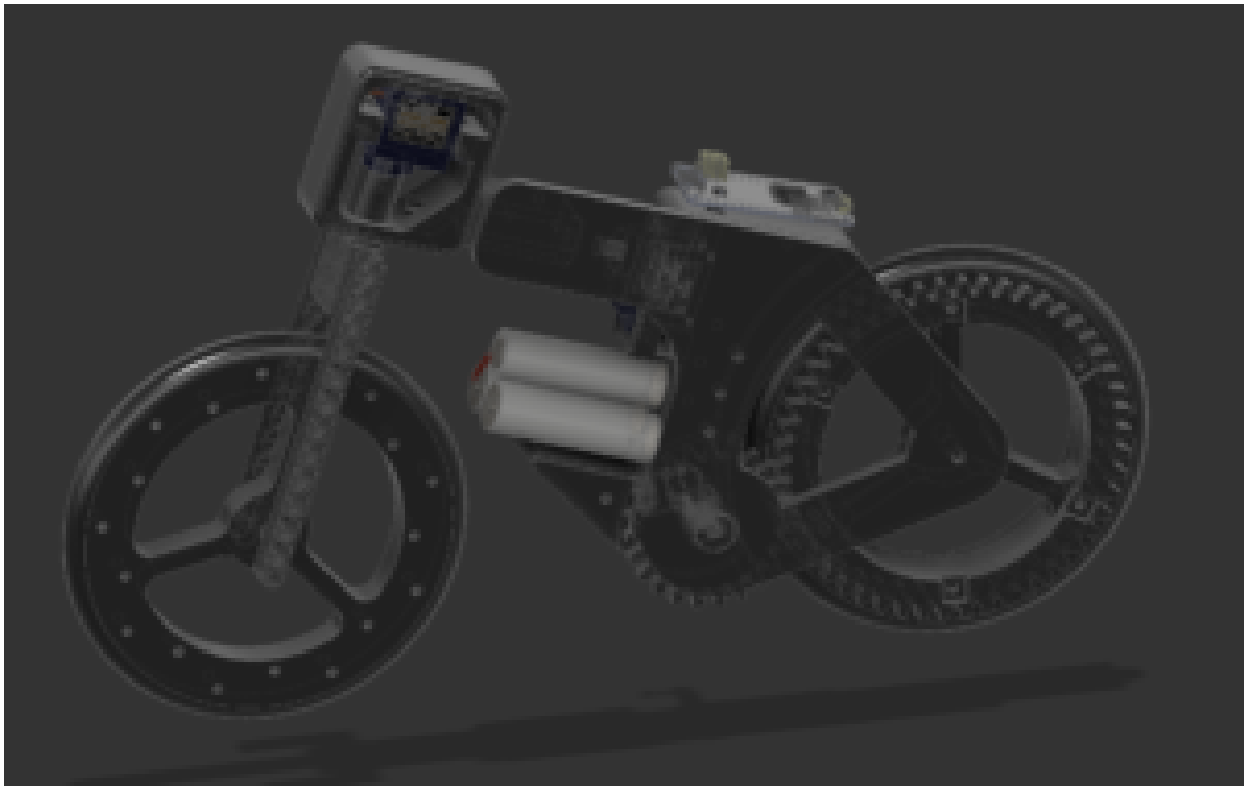
Geometric insights

1. The mass distribution is critical! (this was the primary problem for V3). You need to make sure that the center of mass is above the “support” – the line between the two points where the wheels touch the ground – throughout the entire standing trajectory. In practice, this means that the steering servo needs to be small because its weight is

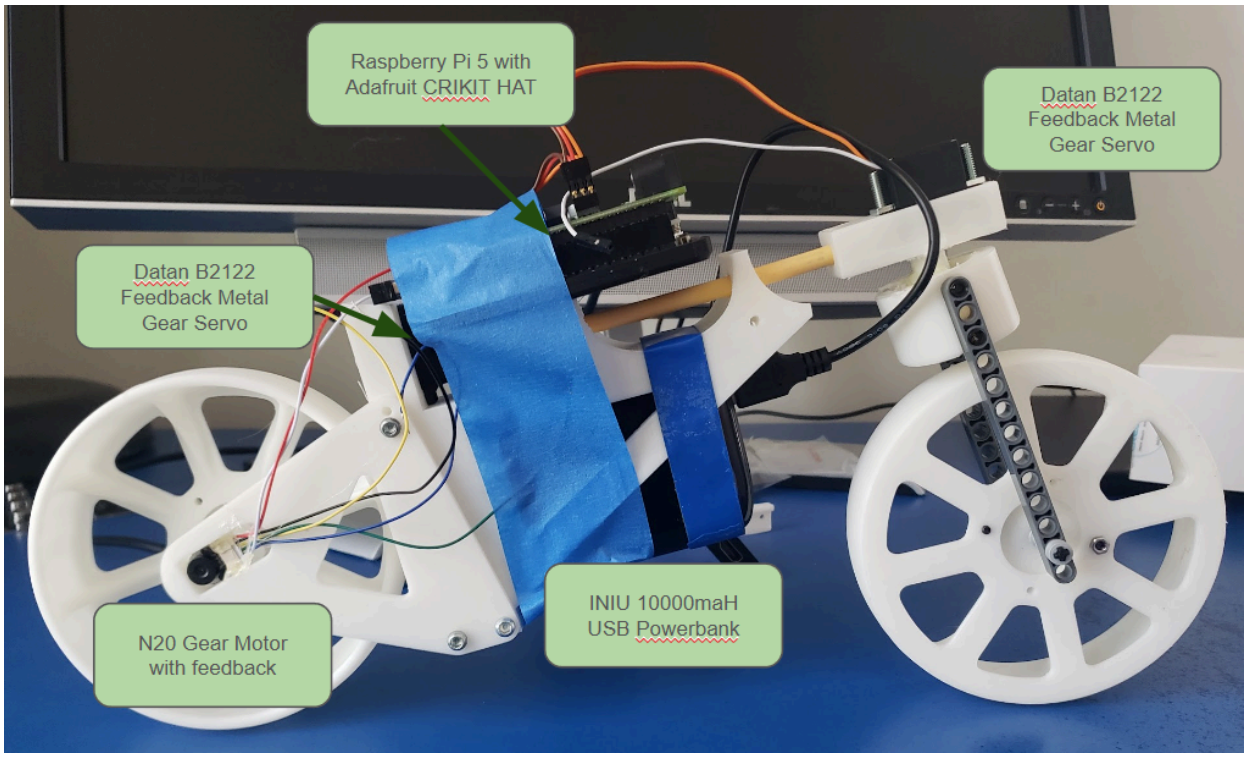
always outside the support, and the rest of the components should be placed low to bring the COM to a place that is comfortably in the low middle of the bike.

2. Larger wheels increase how much the rolling contact point can move, which means a larger support. It also makes the relative size of the motors/batteries smaller so it is easier to get the COM where you want. Too large though and the servos won't be strong enough and the rear wheel will spin too fast. Larger wheels also give a longer moment arm and more inertia (see below).
3. A larger system with more inertia and longer moments will have slower dynamics and be easier to control. This is analogous to how it is easier to balance a broom on your finger than a pencil – the pencil just moves too fast to control well. This indicates that controlling a full-size bike would be significantly easier than the small model, which is practically impossible
4. The torque on the servo joint (especially the body servo) is large. A good design in Design V4, which had plastic that could screw the servo horn and also wrapped around to the bottom where it was attached with a screw. The hole was larger than the screw and no bearing was needed for this model. Note that screw connections were better than friction-fit connections to the servo horn (although the friction-fits in V3 worked).

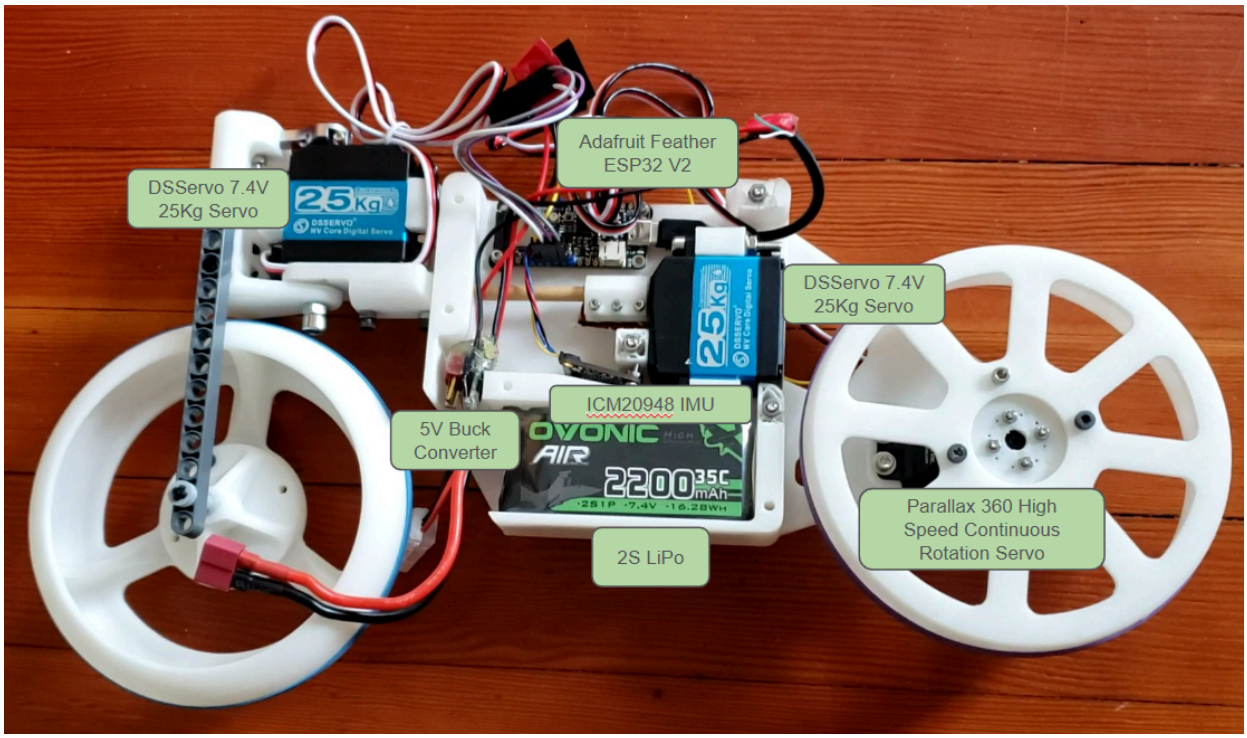
Prototypes and learnings



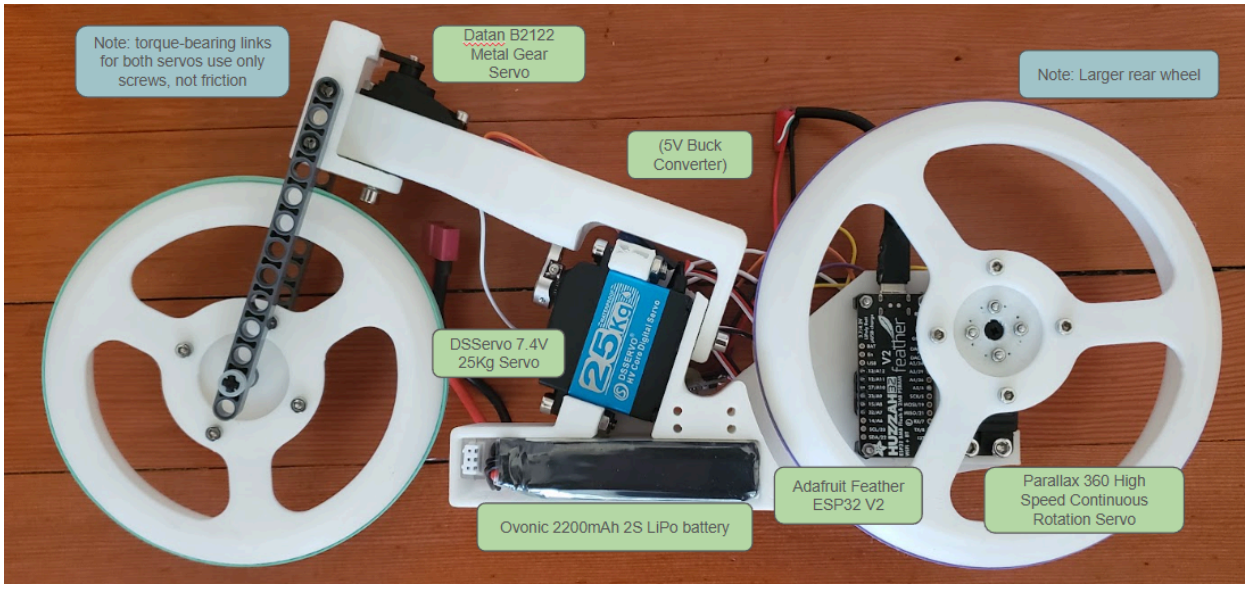
V1 The CAD was finished but I never made it because I didn't like the gears, the front mount was ridiculous, it wasn't clear how I'd connect to the microcontroller, and the AA batteries weren't mounted well and might have not had enough power.



V2 The wheel shape allowed it to passively (though delicately) balance in the upright position. Bluetooth between the Pi and a Nintendo Switch controller. Primary issues: Raspberry Pi 5 was too big physically and required too much power (would brown out after a while). The small servo on the body joint wasn't strong enough to stand the robot up (couldn't tell if it was because of the poor joint or because the servo wasn't strong enough). Joints were hot-glued and flimsy.



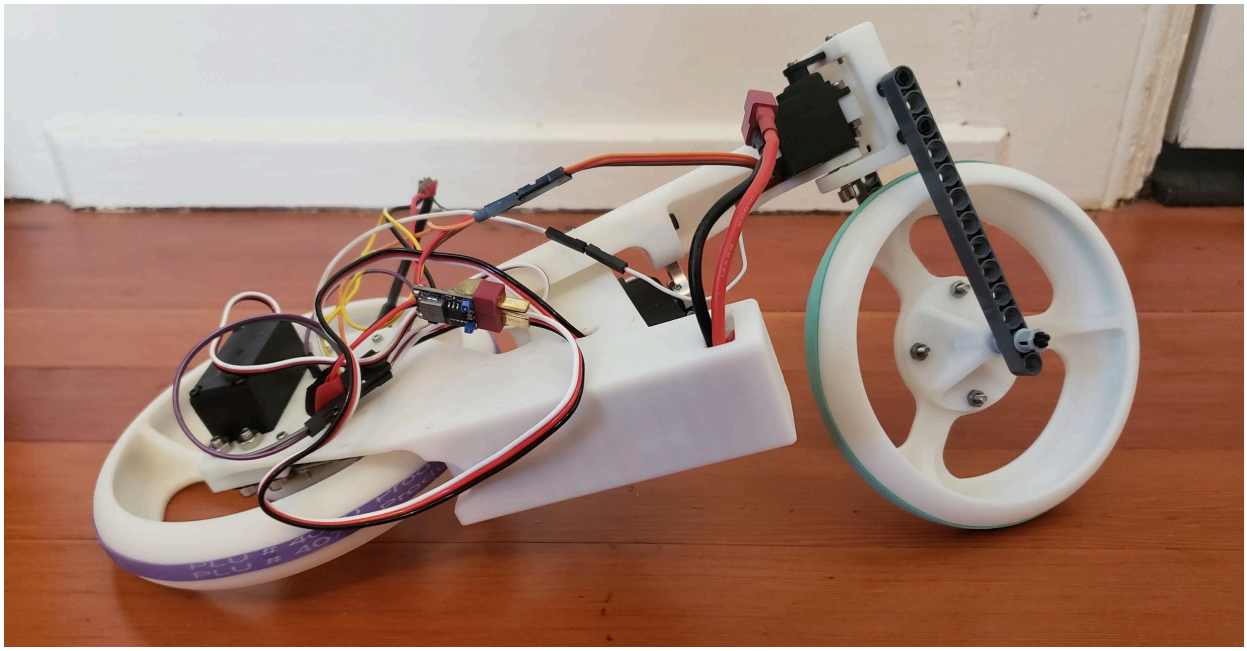
V3 Smaller Controller (Adafruit Feather ESP32) could be mounted without tape, and communicated with my computer over BLE, which in turn received inputs from a Logitech F710 controller. I upgraded to a 2S LiPo battery and the servos were very strong and the circuit worked well. The primary problem was that the COM was way too high so the robot was unable to stand up.



V4

Brought the COM to the middle of the bike by changing to a smaller steering servo and generally pushing the body servo, rear servo, and battery lower. The front wheel is the same

size as V3 so that I could keep a reasonable (73 degree) angle between the ground and the steering axis, but the rear wheel is larger in order to get a lower COM and a larger support area.



V4 was the first prototype that fulfilled my expectations of a “working” robocycle: it demonstrates that the standing motion is passively stable and all of the electronics work. I was unable to get it to actually balance in its fully upright position because the dynamics were too fast for me to control the body/steering servos accurately enough over remote control.

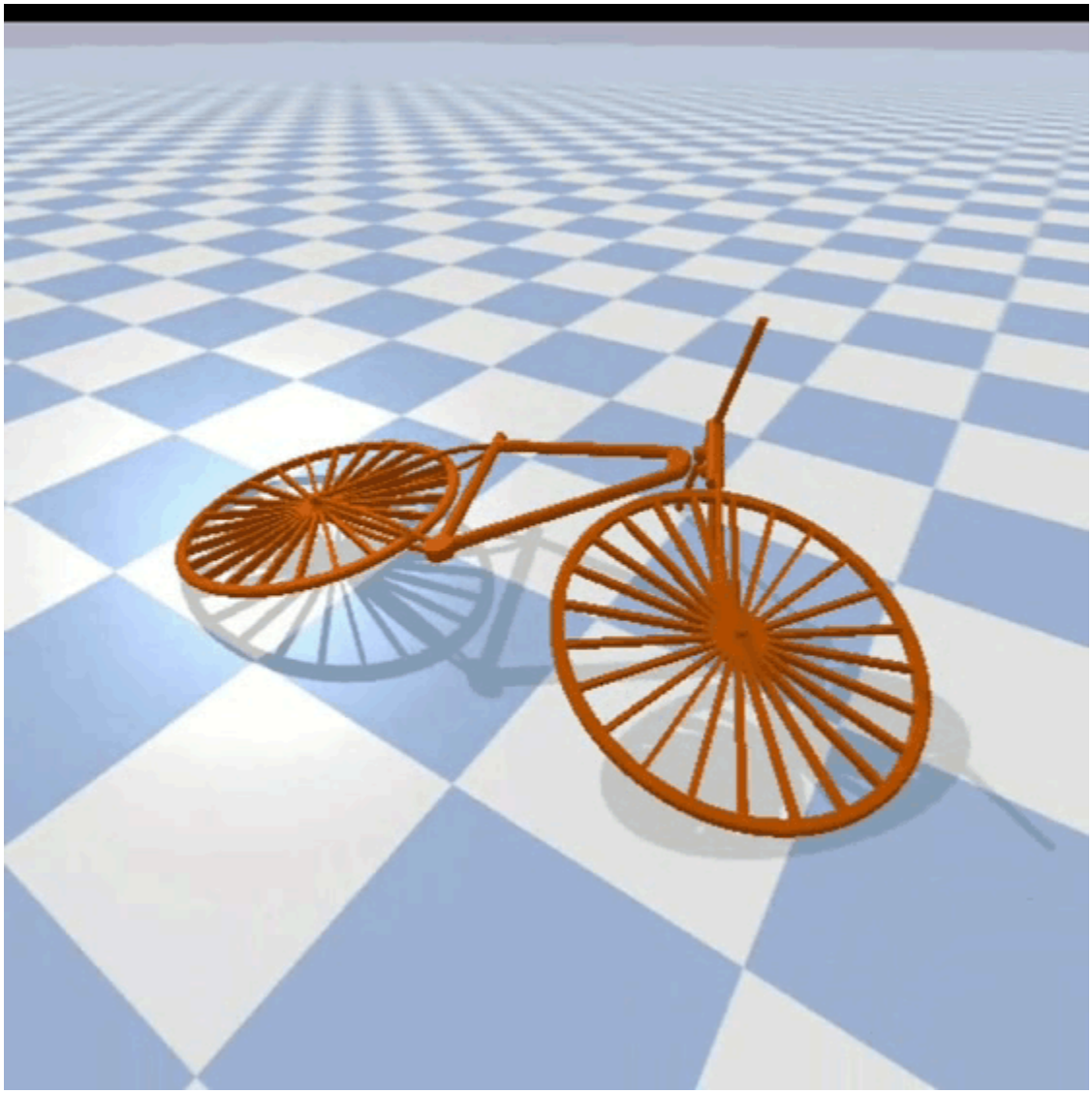
Next steps

The first thing I need to do is develop and analytical model of the system by writing out the kinematics and the dynamics (see above). It would be nice to visualize my model as a “stick figure” in 3D, but matplotlib is suboptimal for making animations in 3D space. From there I should be able to design a simple controller and prove its effectiveness in simulation. Note: A different route would be to simply simulate the bike in Gazebo or Pybullet and then train an RL agent to balance. However, the analytical model seems doable and more principled.

Once I have a controller working in sim, I should actually make a full-scale version of this thing! I may need to source hydraulic actuators for the body servo, and the steering servo will also be expensive (whether I use electronic or hydraulic). Then I need to buy a bike that I can cut up. It’s not clear whether it’s better to buy a normal bike and strap on batteries/motors, or whether I can get an ebike that does some of the work for me (in general the companies don’t make it easy to access the motor controllers or add sensors etc to the battery circuit).

If I wanted to make another mini model, here are some parts that could make it cheaper and smaller/cleaner:
Adafruit QtPy ESP32

BNO055 IMU for QtPy
Use a cheaper continuous rotation servo on the rear
Battery could be smaller than the current 2200mAh (I never got even close to running out of battery)



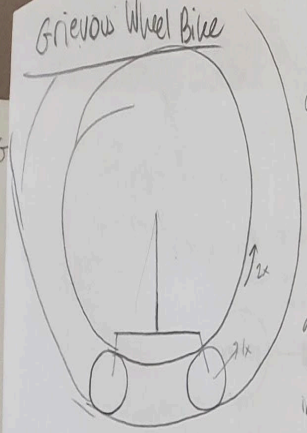
Pybullet simulation of “standing up from the ground” (controlled manually using arrow keys)

Side Quest: General Grievous Wheel bike

In order to study the dynamics of rolling contact, a more simple system (at least, 2D) is a hamster wheel, rocking chair, or General Grievous’ wheel bike. An interesting question to me is that a rocking chair has a motion like a pendulum, so how can the rolling contact equations be converted into a form that looks like a pendulum? Furthermore, what is the relationship between the path traced by the COM and the path traced by the center of curvature, or the path of the

instantaneous contact point as the chair rocks? Solving these may give insights that are very useful to the robocycle, and it would be cool to make a wheel-bike along the way.

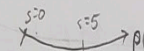
Grievous Wheel Bike



Nice thing about CC & COM trajectories (in body frame) is we can calculate them analytically beforehand instead of using a moving coordinate system that moves in a way you haven't predicted

As joints move, the COM will move along some trajectory in the body frame
As object rolls, the COM will move along some trajectory in the world frame
As the object rolls, the Center of Curvature ↑ will move along some trajectory in the world frame
of the point currently in contact with the ground

In the body frame, If we parameterize the curve of possible contact pts



then we'll get a body-frame trajectory of the center of curvature (for a circle this is just a point)

Joints move → COM moves → CC, COM, P not collinear → fall over → support point moves (or gravity vector)

Model moving contact as a prismatic joint + pendulum pin joint

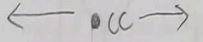
→ Due to inertia you overshoot → At some point inertia dies & you swing back to static eq

States = rate of change of inertia (ball rolling up hill: As hill gets steeper the ball slows down faster

Statics tells you stability in the same way velocity=0 tells you stability of dynamics

How is the parametrization of point-contact movement incorporated into the controller?

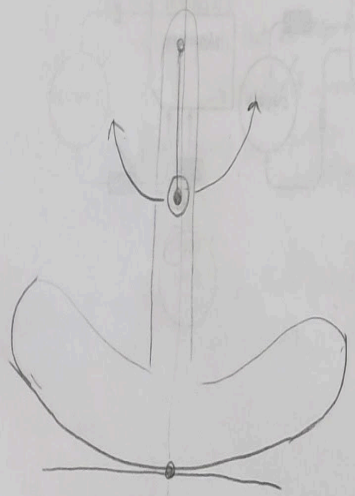
This paper makes vert coords at point of contact

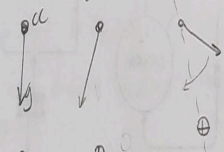


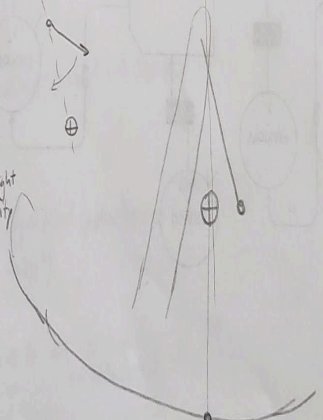
In same way, if you freeze the movement, that tells you something about where the trajectory will go ⊕ CC

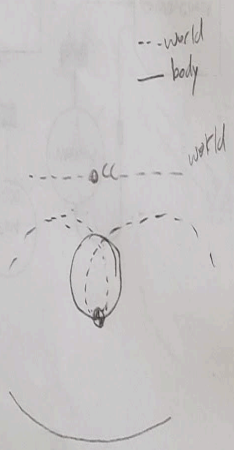
⇒ looking at vertical line in state space = different values of $\dot{x}$, and in particular where $\dot{x}=0$ can be useful to predict a stable fixed-point

Frame of CC gravity vector swings back and forth like a pendulum









What is the relationship between a rocker and a pendulum?

Practice: Acrobot on a cart
→ Grievous wheel bike
→ Robocycle

General problem: Location/direction of force is a function of global orientation → Fit this into optimization framework