Shooting Our Shot

Shooting Our Shot - How to Create an Interpolating Tree Map

An Interpolating Tree Map (aka a Shooter Map) boiled down to its core is just linear regression. It was highly used during the Rebuilt Season for shooting into the hub and shuttling across the field.

For linear regression to work, it needs a bunch of data points. In the case of Rebuilt, the Shooter Map used the *distance to the center of the hub* as the input and it gave us the *target rps for the shooter* as the output.

Setting up the Map

To begin, you first need to manually find a bunch of good data points. We used the spots on the field where we would most often shoot from to determine our initial data points. Using these data points, we made a line graph on Google Sheets, and this helped us see where we were missing data. After this, we tested shooting with the shooter map automation and added in more data where the map failed.

The Code

The following are examples of code that was used during Rebuilt to setup our Shooter Map:

ShooterStateData.java in the *util* folder

```
package frc.robot.util;


import static edu.wpi.first.units.Units.Radians;
```

```
import edu.wpi.first.math.interpolation.Interpolator;
import edu.wpi.first.units.measure.Angle;

public class ShooterStateData {

    public static final Interpolator<ShooterStateData> interpolator = (startValue, endValue,
t) -> {
        double initialHood = startValue.hoodPos.in(Radians);
        double finalHood = endValue.hoodPos.in(Radians);

        double initialRPS = startValue.rps;
        double finalRPS = endValue.rps;

        double initialToF = startValue.timeOfFlight;
        double finalToF = endValue.timeOfFlight;

        double interpolatedHood = initialHood + t * (finalHood - initialHood);

        return new ShooterStateData(
                Radians.of(interpolatedHood),
                initialRPS + t * (finalRPS - initialRPS),
                initialToF + t * (finalToF - initialToF));
    };
    public final Angle hoodPos;
    public final double rps;
    public final double timeOfFlight;

    public ShooterStateData(Angle hoodPos, double rps, double timeOfFlight) {
        this.hoodPos = hoodPos;
        this.rps = rps;
        this.timeOfFlight = timeOfFlight;
    }
}
```

Initializing the *Shooter Map* in *ShooterConfigsBeta.java*

```
public static final InterpolatingTreeMap<Double, ShooterStateData> SHOOTER_MAP =
            new InterpolatingTreeMap<>(InverseInterpolator.forDouble(),
```

```
ShooterStateData.interpolator);
```

Logging *Distance to Hub* in *Robot Container*

```
public void updateLoggers() {
        Pose2d currentPose = drive.getState().Pose;
        Translation2d modifiedTarget =
AllianceFlipUtil.apply(centerHubOpening.toTranslation2d());
        Translation2d currentPosition = currentPose.getTranslation();
        double distance = modifiedTarget.getDistance(currentPosition);

        Logger.recordOutput("AutoAimCommands/Shooter Map/hub distance", distance);
    }
```

Putting data into the *Shooter Map* (this is also in *ShooterConfigsBeta.java*, right underneath the initialization)

```
static {
        SHOOTER_MAP.put(1.74, new ShooterStateData(HoodPositions.STOW.getPosition(), 25,
0.0));
        SHOOTER_MAP.put(2.13, new ShooterStateData(HoodPositions.STOW.getPosition(), 27,
0.0));
        SHOOTER_MAP.put(2.43, new ShooterStateData(HoodPositions.STOW.getPosition(), 30,
0.0));
        SHOOTER_MAP.put(2.78, new ShooterStateData(HoodPositions.STOW.getPosition(), 31,
0.0));
        SHOOTER_MAP.put(3.15, new ShooterStateData(HoodPositions.STOW.getPosition(), 33,
0.0));
        SHOOTER_MAP.put(3.78, new ShooterStateData(HoodPositions.STOW.getPosition(), 38.7,
0.0));
        SHOOTER_MAP.put(4.36, new ShooterStateData(HoodPositions.STOW.getPosition(), 44,
0.0));
    }
```

Using the *Shooter Map* to actually calculate the *target rps* of the Shooter (in *AutoAimCommands.java*)

```
public static Command readyAim(CommandSwerveDrivetrain drive, Shooter shooter, Translation2d
target) {

        return Commands.defer(
                () -> {
                    Pose2d currentPose = drive.getState().Pose;
                    Translation2d modifiedTarget = AllianceFlipUtil.apply(target);
                    Translation2d currentPosition = currentPose.getTranslation();
                    double distance = modifiedTarget.getDistance(currentPosition);

                    ShooterStateData state = ShooterConfigsBeta.SHOOTER_MAP.get(distance);

                    double targetRPS = state.rps;

                    Logger.recordOutput("AutoAimCommands/Shooter Map/target rps", targetRPS);

                    return shooter.shoot(targetRPS);
                },
                Set.of(shooter));
    }
```

The Shooter Map uses the **distance to the hub the input**, it does the linear regression, and then **outputs the target rps**.

Once you have set everything up, you'll be able to call the readyAim command using a controller binding and it'll work!

Revision #1
Created 2026-03-16 21:50:56 UTC by Amit Choudhary
Updated 2026-03-16 23:03:09 UTC by Amit Choudhary