Lesson 4

Intro to Robot Programming

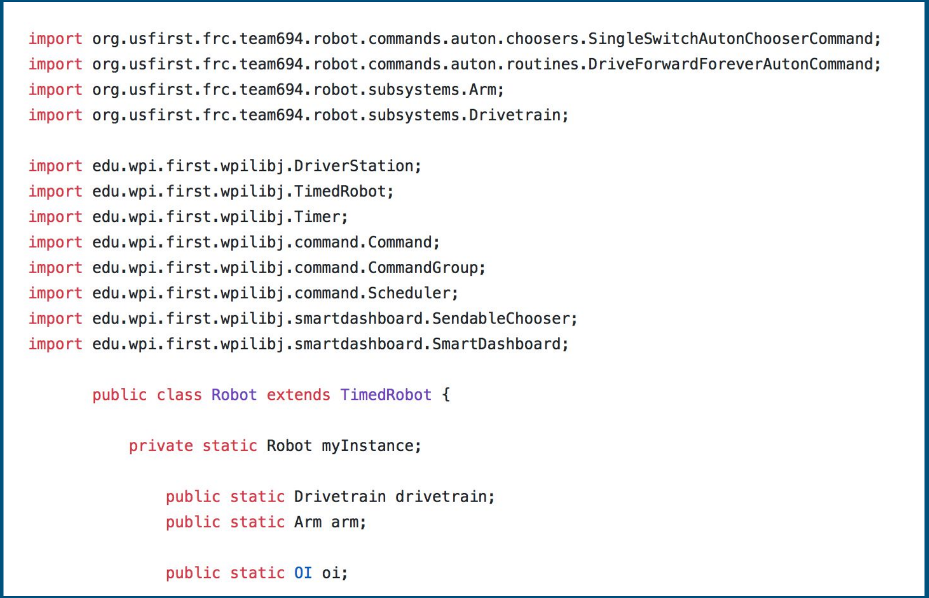
Intro to wpilib

- Wpilib is a library of code that is provided to us by FRC (because we are lazy and don't want to start from scratch)
- It contains pre-programmed classes and functions for many things like motors and sensors.
- There is documentation as to what is available and what each method does (just like Scanner does).

http://first.wpi.edu/FRC/roborio/release/docs/java/ (linked on newbie-ed-2019 repo)

What is an import? (Not History)

- Java is designed to be easily able to work on different files
- Import is a way of taking specific classes from another file instead of taking the entire file and copying and pasting it into the file
- An example of an import is the scanner class
  import java.util.Scanner;



```java
import org.usfirst.frc.team694.robot.commands.auton.choosers.SingleSwitchAutonChooserCommand;
import org.usfirst.frc.team694.robot.commands.auton.routines.DriveForwardForeverAutonCommand;
import org.usfirst.frc.team694.robot.subsystems.Arm;
import org.usfirst.frc.team694.robot.subsystems.Drivetrain;

import edu.wpi.first.wpilibj.DriverStation;
import edu.wpi.first.wpilibj.TimedRobot;
import edu.wpi.first.wpilibj.Timer;
import edu.wpi.first.wpilibj.command.Command;
import edu.wpi.first.wpilibj.command.CommandGroup;
import edu.wpi.first.wpilibj.command.Scheduler;
import edu.wpi.first.wpilibj.smartdashboard.SendableChooser;
import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;

    public class Robot extends TimedRobot {

        private static Robot myInstance;

            public static Drivetrain drivetrain;
            public static Arm arm;

            public static OI oi;
```

Intro To Command Based Programming

Subsystems: Subsystems are portions of a robot that are being controlled by the code. They are divided up so that each subsystem performs a different role on the robot

Eg. Grabber arm, drivetrain, shooter, Spinning Blade Of Doom^{TM}

Commands: These are actions that the robots can perform using the subsystems

Eg. grab, shoot, spin, self-destruct, cut Pratham’s hair

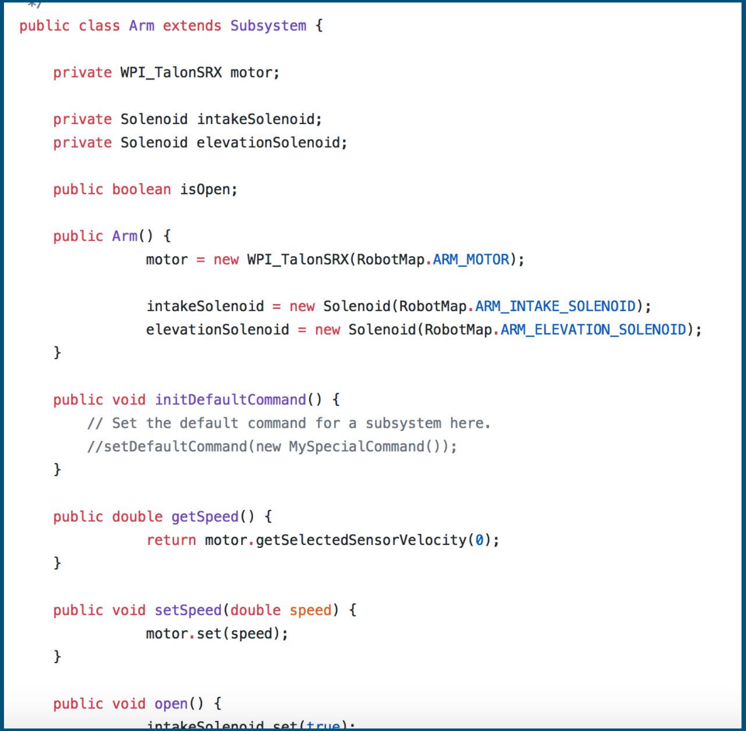
Exercise: Watch video and name several subsystems and commands on robot

Subsystems

Subsystem is also a class that we extend. Every single subsystem is its own file.

It has a couple of default methods, such as a constructor and initDefaultCommand().

An example of each being used is here.



```
 */
public class Arm extends Subsystem {

    private WPI_TalonSRX motor;

    private Solenoid intakeSolenoid;
    private Solenoid elevationSolenoid;

    public boolean isOpen;

    public Arm() {
            motor = new WPI_TalonSRX(RobotMap.ARM_MOTOR);

            intakeSolenoid = new Solenoid(RobotMap.ARM_INTAKE_SOLENOID);
            elevationSolenoid = new Solenoid(RobotMap.ARM_ELEVATION_SOLENOID);
    }

    public void initDefaultCommand() {
        // Set the default command for a subsystem here.
        //setDefaultCommand(new MySpecialCommand());
    }

    public double getSpeed() {
            return motor.getSelectedSensorVelocity(0);
    }

    public void setSpeed(double speed) {
            motor.set(speed);
    }

    public void open() {
            intakeSolenoid.set(true);
```

Commands

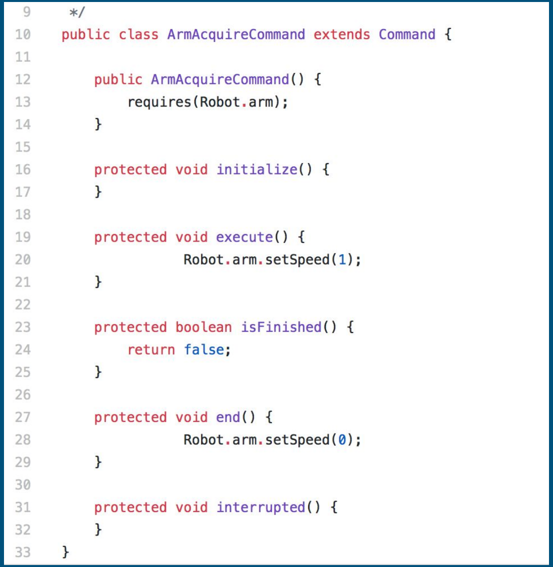
- Commands are a separate file that uses methods of a class
- Helps with organization and autons are programmed with commands

They usually have…

- A constructor
- Initialize, execute, isFinished
- End, interrupted

Commands are named as follows: SubsystemActionCommand

Here is an example. Here is an easier example.



```
 */
public class ArmAcquireCommand extends Command {

    public ArmAcquireCommand() {
        requires(Robot.arm);
    }

    protected void initialize() {
    }

    protected void execute() {
                Robot.arm.setSpeed(1);
    }

    protected boolean isFinished() {
        return false;
    }

    protected void end() {
                Robot.arm.setSpeed(0);
    }

    protected void interrupted() {
    }
}
```

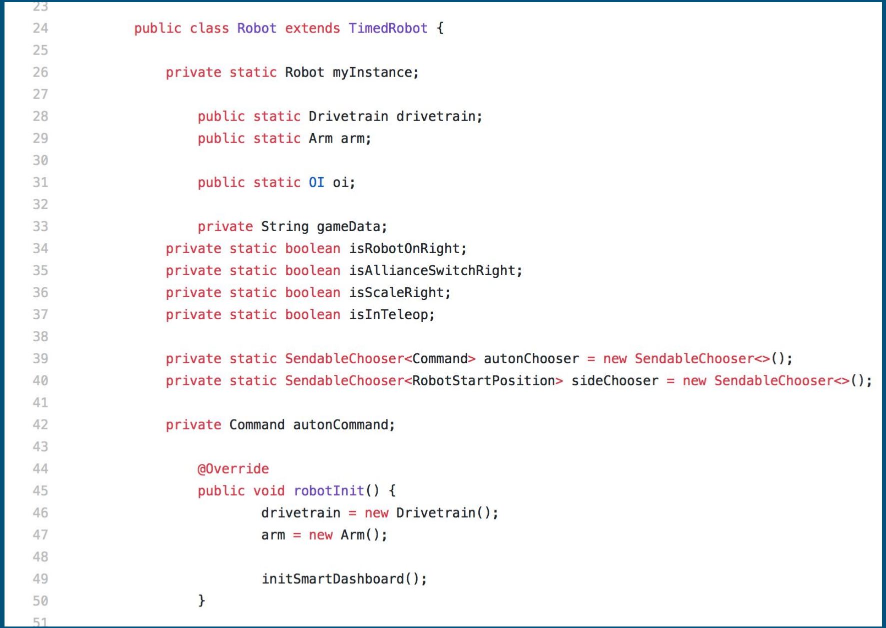
Subsystems Vs Commands

Subsystems are physical things on the robot.

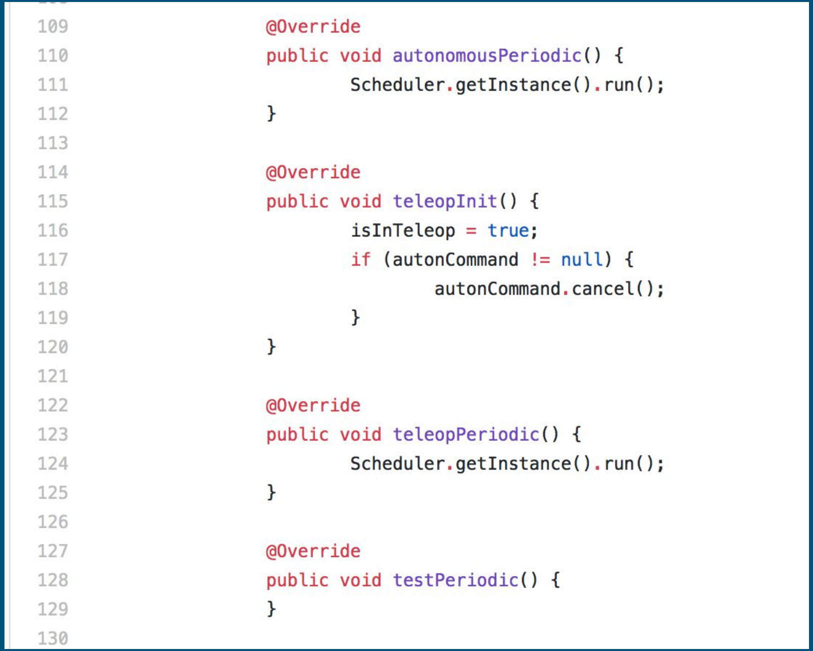
Commands are things that you tell the subsystems to do.

Eg. A grabber arm is a subsystem, but moving the grabber arm is a command

Robot.java (Link to code here)



```
        public class Robot extends TimedRobot {

            private static Robot myInstance;

                public static Drivetrain drivetrain;
                public static Arm arm;

                public static OI oi;

                private String gameData;
            private static boolean isRobotOnRight;
            private static boolean isAllianceSwitchRight;
            private static boolean isScaleRight;
            private static boolean isInTeleop;

            private static SendableChooser<Command> autonChooser = new SendableChooser<>();
            private static SendableChooser<RobotStartPosition> sideChooser = new SendableChooser<>();

            private Command autonCommand;

                @Override
                public void robotInit() {
                        drivetrain = new Drivetrain();
                        arm = new Arm();

                        initSmartDashboard();
                }
```



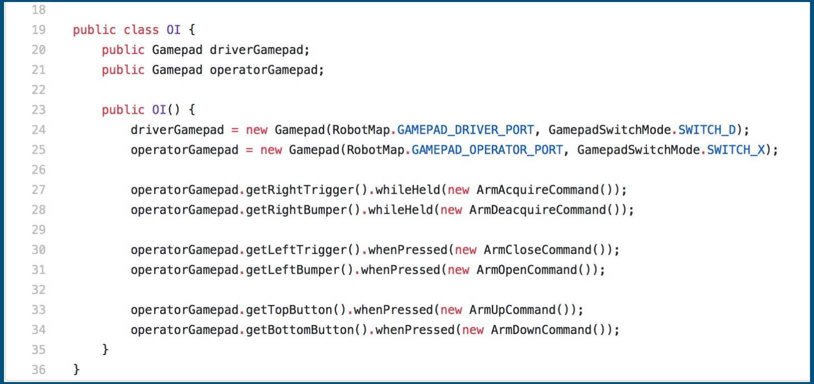
```
        @Override
        public void autonomousPeriodic() {
                Scheduler.getInstance().run();
        }

        @Override
        public void teleopInit() {
                isInTeleop = true;
                if (autonCommand != null) {
                        autonCommand.cancel();
                }
        }

        @Override
        public void teleopPeriodic() {
                Scheduler.getInstance().run();
        }

        @Override
        public void testPeriodic() {
        }
```

Fieldmap

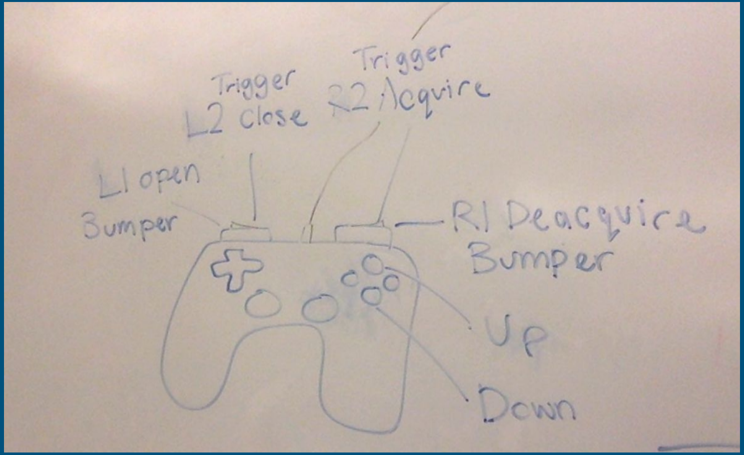
- Fieldmap is a file that contains all of the field’s measurements and helps with autons when the robot has to move a specific distance

OI

- Operator Interface (code here)
- Where we program which button on the controller does what



```java

public class OI {
    public Gamepad driverGamepad;
    public Gamepad operatorGamepad;

    public OI() {
        driverGamepad = new Gamepad(RobotMap.GAMEPAD_DRIVER_PORT, GamepadSwitchMode.SWITCH_D);
        operatorGamepad = new Gamepad(RobotMap.GAMEPAD_OPERATOR_PORT, GamepadSwitchMode.SWITCH_X);

        operatorGamepad.getRightTrigger().whileHeld(new ArmAcquireCommand());
        operatorGamepad.getRightBumper().whileHeld(new ArmDeacquireCommand());

        operatorGamepad.getLeftTrigger().whenPressed(new ArmCloseCommand());
        operatorGamepad.getLeftBumper().whenPressed(new ArmOpenCommand());

        operatorGamepad.getTopButton().whenPressed(new ArmUpCommand());
        operatorGamepad.getBottomButton().whenPressed(new ArmDownCommand());
    }
}
```

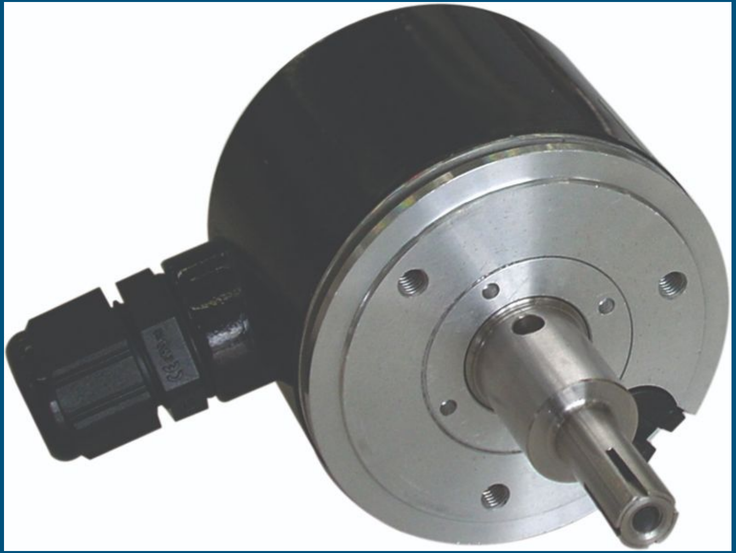


Solenoids (example code here)

- Solenoids are what allows us to control pistons with code
- Single solenoid vs Double solenoid

Encoders

- Measures how many times a wheel rotates
  - Gives distance wheel turned
  - Gives actual distance traveled
  - Gives speed of robot
  - Represented by talonsrx

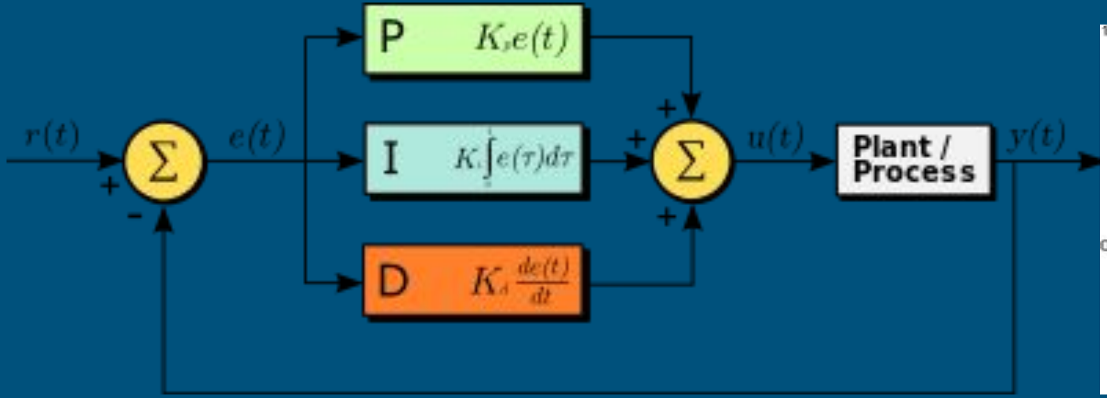


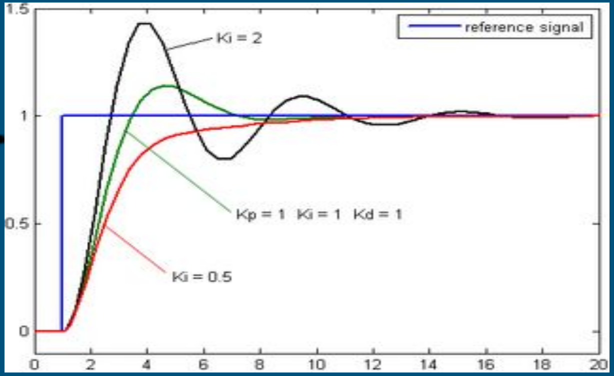
Gyro

- Detects orientation of the robot
    - Allows robot to turn

PIDF (Porportional integral derivative feed-forward)

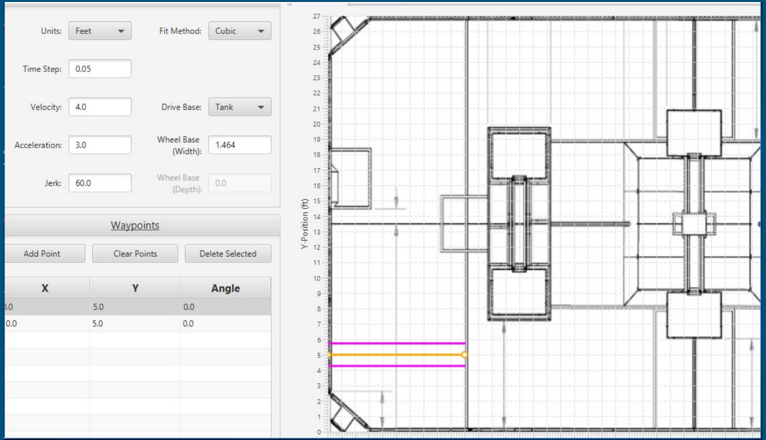
- Improves efficiency of auton using an equation using constants





Motion Profiling

- Set up trajectory points on the path to a goal for smooth autos

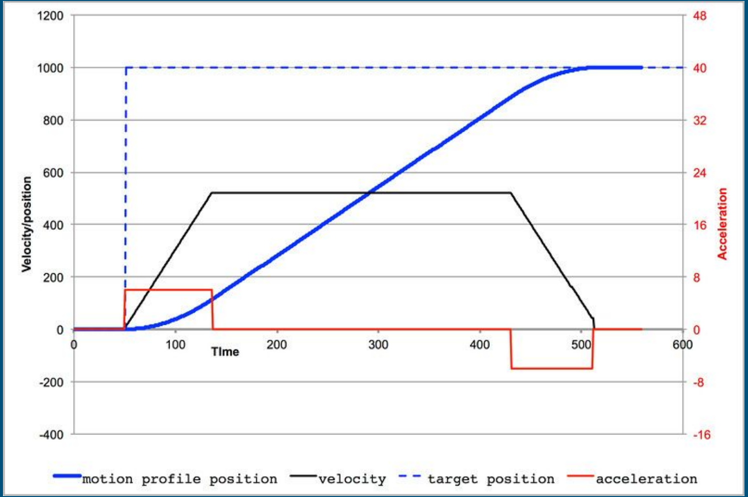


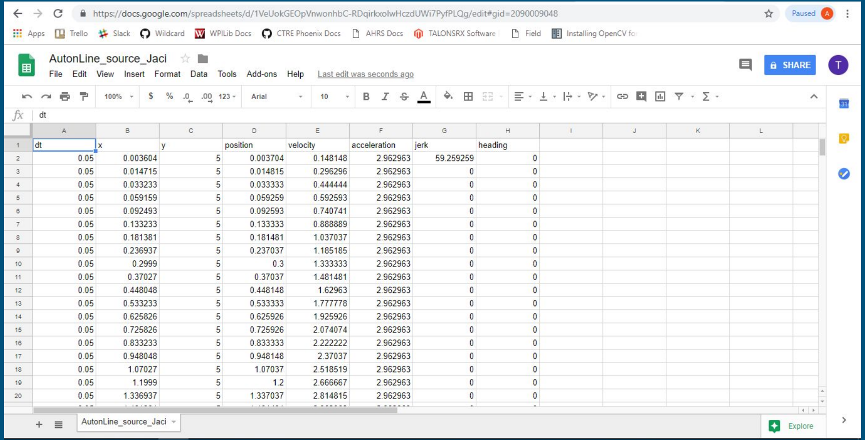
Units: Feet
Fit Method: Cubic
Time Step: 0.05
Velocity: 4.0
Drive Base: Tank
Acceleration: 3.0
Wheel Base (Width): 1.464
Jerk: 60.0
Wheel Base (Depth): 0.0

Waypoints

Add Point | Clear Points | Delete Selected

X	Y	Angle
.0	5.0	0.0
0.0	5.0	0.0





https://docs.google.com/spreadsheets/d/1VeUokGEOpVnwonhbC-RDqirkxolwHczdUWi7PyfPLQg/edit#gid=2090009048

AutonLine_source_Jaci

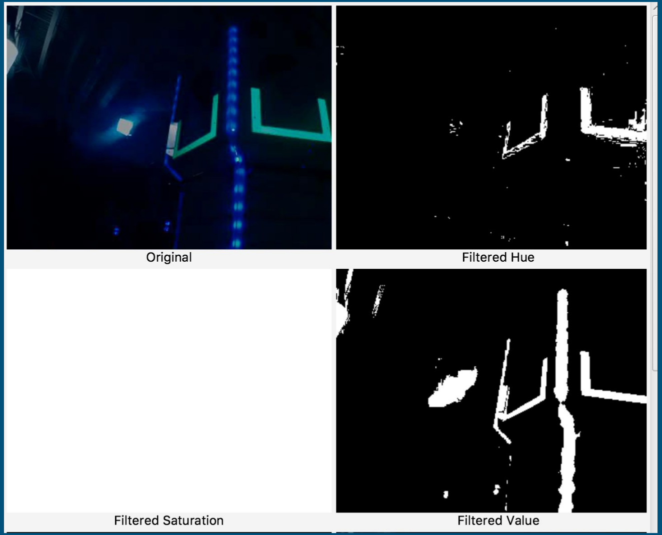
File Edit View Insert Format Data Tools Add-ons Help Last edit was seconds ago

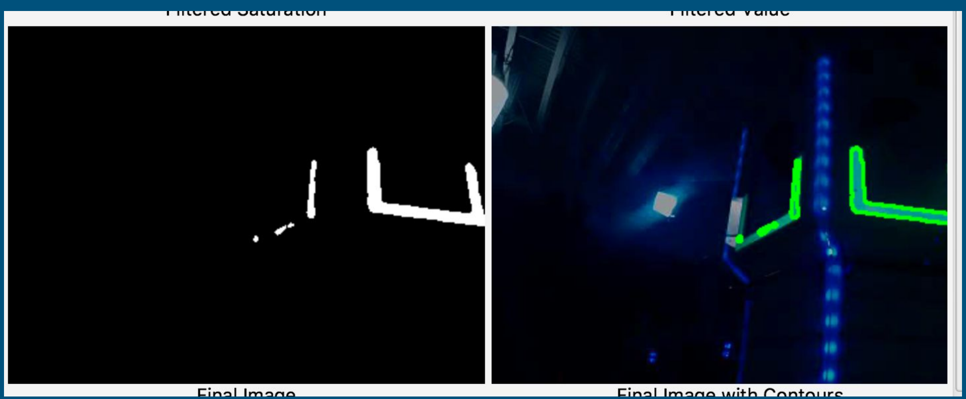
	A	B	C	D	E	F	G	H
1	dt	x	y	position	velocity	acceleration	jerk	heading
2	0.05	0.003604	5	0.003704	0.148148	2.962963	59.259259	0
3	0.05	0.014715	5	0.014815	0.296296	2.962963	0	0
4	0.05	0.033233	5	0.033333	0.444444	2.962963	0	0
5	0.05	0.059159	5	0.059259	0.592593	2.962963	0	0
6	0.05	0.092493	5	0.092593	0.740741	2.962963	0	0
7	0.05	0.133233	5	0.133333	0.888889	2.962963	0	0
8	0.05	0.181381	5	0.181481	1.037037	2.962963	0	0
9	0.05	0.236937	5	0.237037	1.185185	2.962963	0	0
10	0.05	0.2999	5	0.3	1.333333	2.962963	0	0
11	0.05	0.37027	5	0.37037	1.481481	2.962963	0	0
12	0.05	0.448048	5	0.448148	1.62963	2.962963	0	0
13	0.05	0.533233	5	0.533333	1.777778	2.962963	0	0
14	0.05	0.625826	5	0.625926	1.925926	2.962963	0	0
15	0.05	0.725826	5	0.725926	2.074074	2.962963	0	0
16	0.05	0.833233	5	0.833333	2.222222	2.962963	0	0
17	0.05	0.948048	5	0.948148	2.37037	2.962963	0	0
18	0.05	1.07027	5	1.07037	2.518519	2.962963	0	0
19	0.05	1.1999	5	1.2	2.666667	2.962963	0	0
20	0.05	1.336937	5	1.337037	2.814815	2.962963	0	0

AutonLine_source_Jaci

OpenCV

- Stands for Computer Vision
- Takes picture and finds target through filtering colors / lighting


Original
Filtered Hue
Filtered Saturation
Filtered Value


Final Image
Final Image with Contours