Team progress

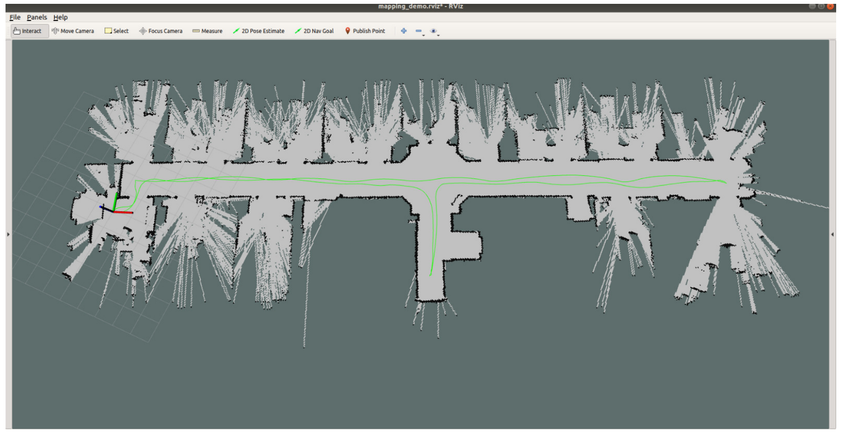
week 4

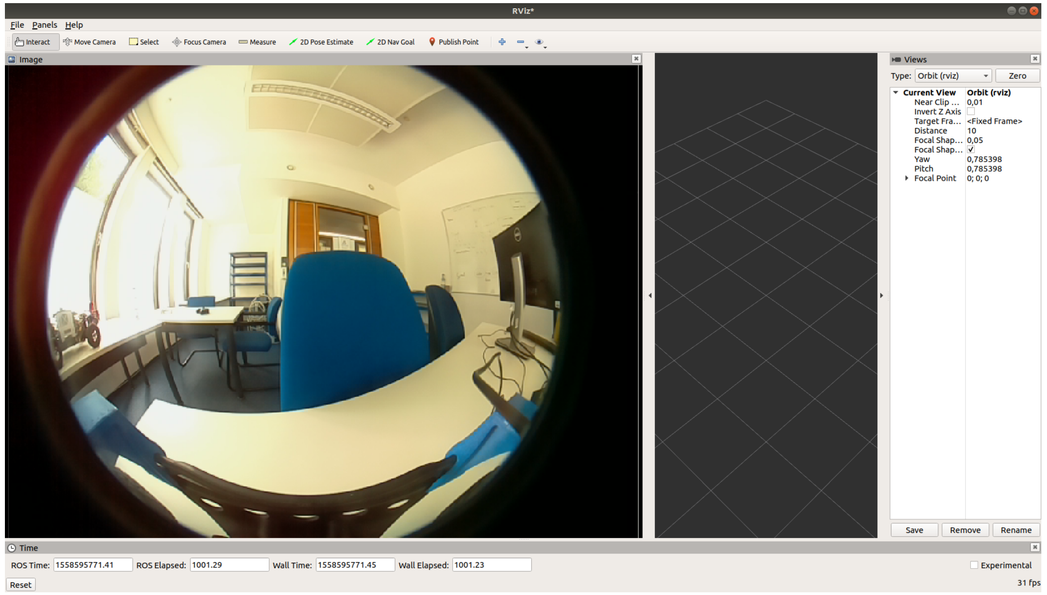
Overview

- ROS
- Path following
- TCP server
- AF3 model

ROS

- Issues with Linux distributions and ROS versions
- ROS melodic on redrovercamera Pi
- Installation of Ubuntu Mate 18.04
- rplidar_ros package for LIDAR
- video_stream_opencv package for Camera

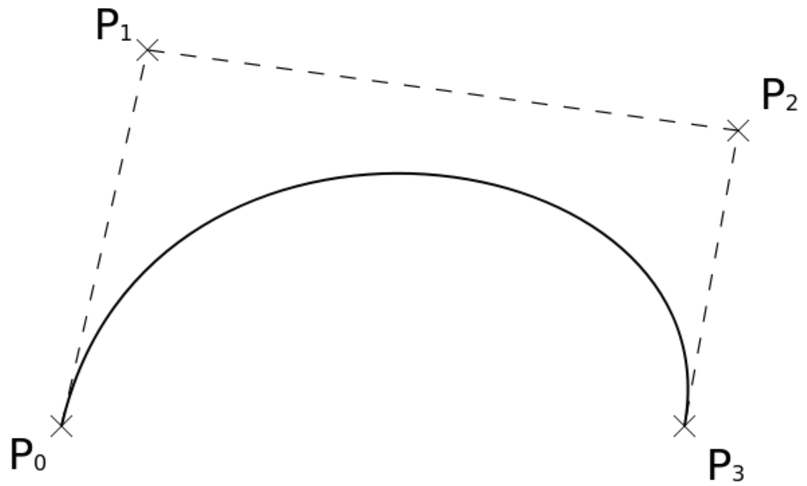




Bézier curve

- Definition
- Sampling
- Implementation in C++
- Wrapped as ROS node

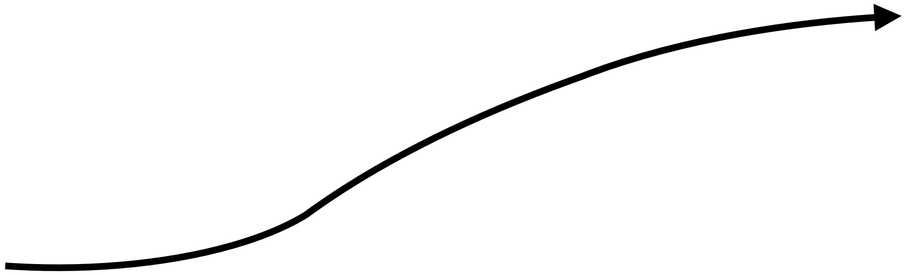
```
class CubicBezierCurve {
private:
    Vec2f start;
    Vec2f waypoint1;
    Vec2f waypoint2;
    Vec2f end;
    double len;
public:
    CubicBezierCurve(Vec2f p1, Vec2f p2, Vec2f p3, Vec2f p4);
    Vec2f positionAt(double tau);
    Vec2f derivativeAt(double tau);
    Vec2f tangentAt(double tau);
    double arclength2tau(double arc, double tauBegin = 0.0, double tauStep = DEFAULT_STEP_WIDTH);
    double length();
};
```

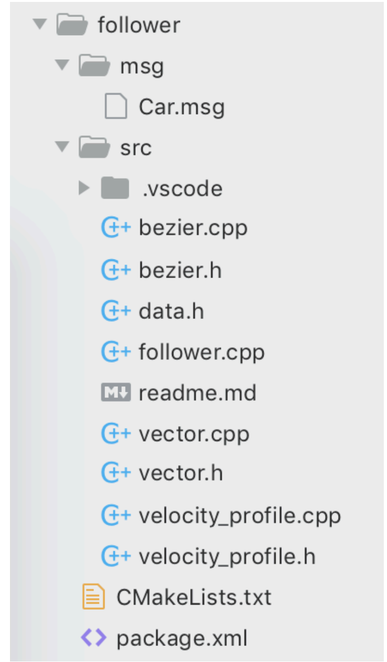


Path following

- Follow Bézier curve
- Implementation in C++
- Wrapped as ROS node

```
while(ros::ok()) {
    profile.step(delta_t);
    double delta_distance = profile.getDistance() - distance;
    distance = profile.getDistance();
    tau = curve.arclength2tau(delta_distance, tau, 0.00001);

    Vec2f rearTangent = curve.tangentAt(tau);
    double frontTau = curve.arclength2tau(WHEEL_BASE, tau, 0.0002);
    Vec2f frontTangent = curve.tangentAt(frontTau);

    double angle = rearTangent.angle(frontTangent) * 2.5;

    velocityMsg.data = (float) profile.getVelocity();
    steeringAngleMsg.data = (float) angle;

    velocityPublisher.publish(velocityMsg);
    steeringAnglePublisher.publish(steeringAngleMsg);

    ros::spinOnce();
    loop_rate.sleep();
}
```

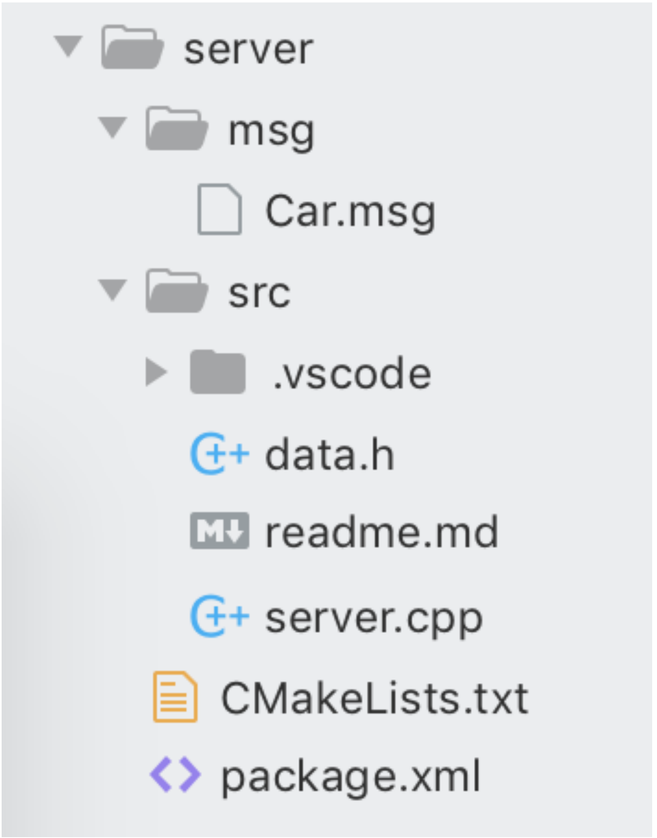




TCP server

ROS → AF3

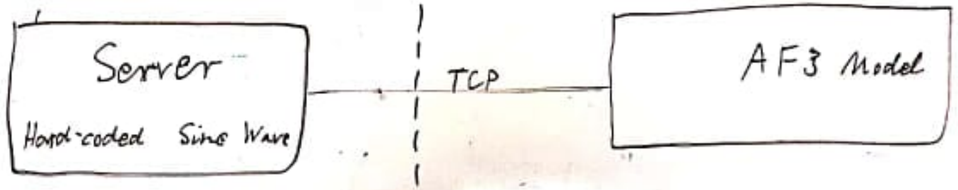
- Communication with AF3 model
- Implementation in C++
- Wrapping as ROS node



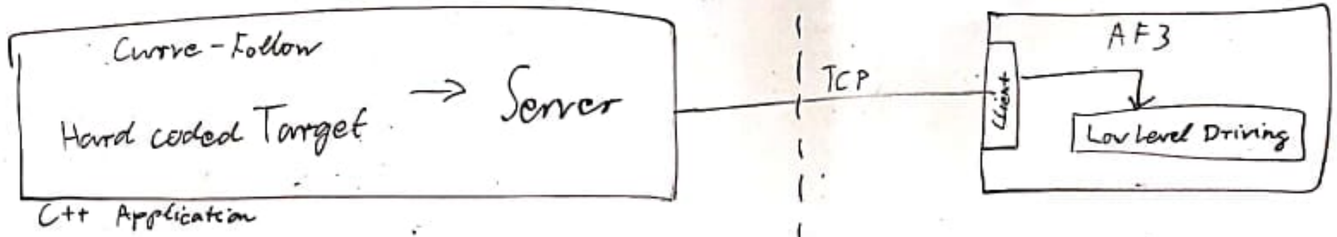
AF3 model

- TCP client
- Control of low level driving function
  - velocity
  - steering angle

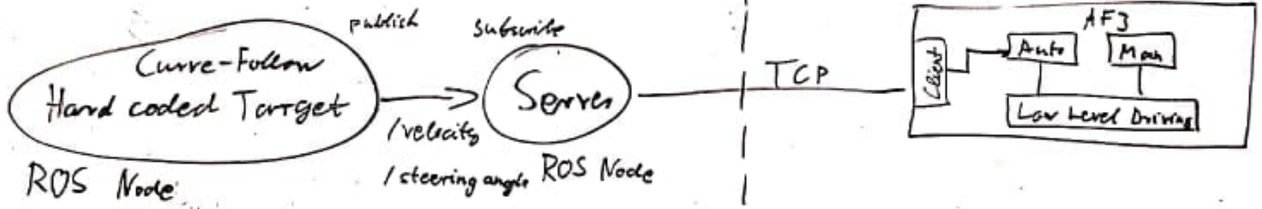
Now:



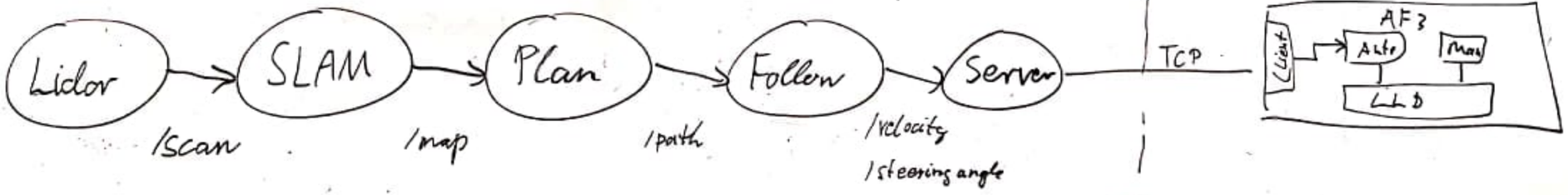
Next:



Tomorrow:



Goal:



Demo