GOETHE UNIVERSITÄT FRANKFURT AM MAIN

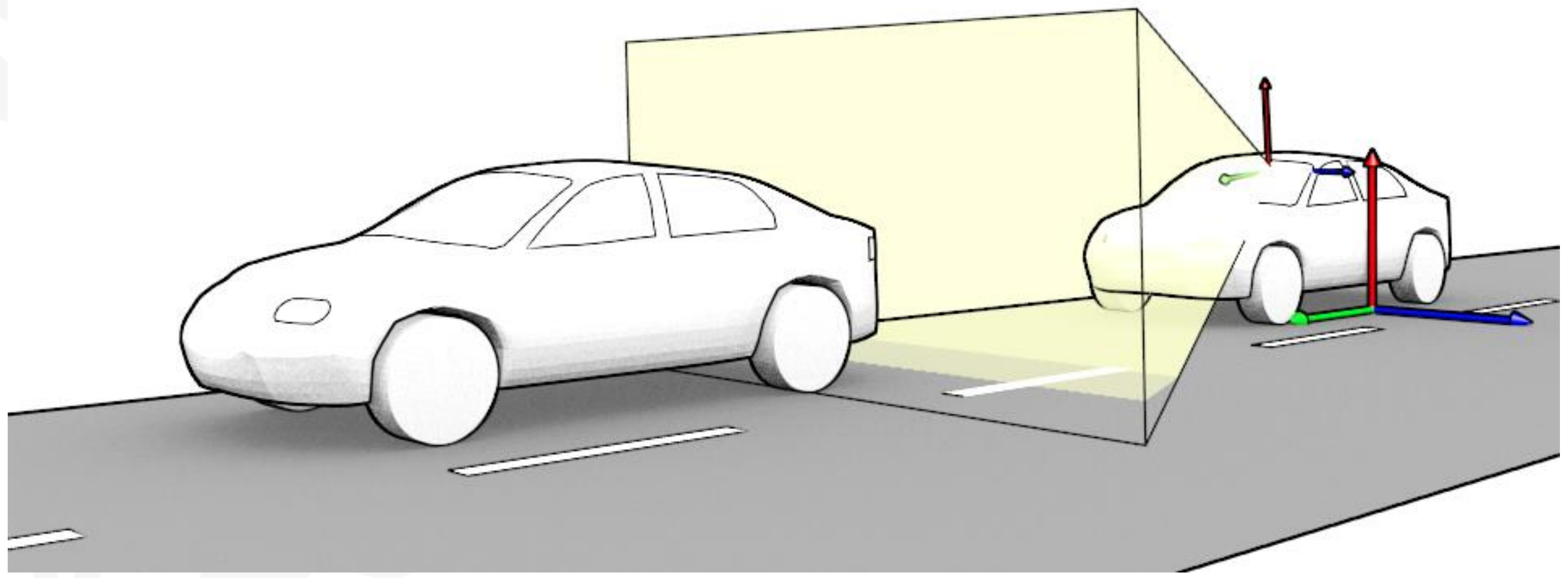
Tobias Weis

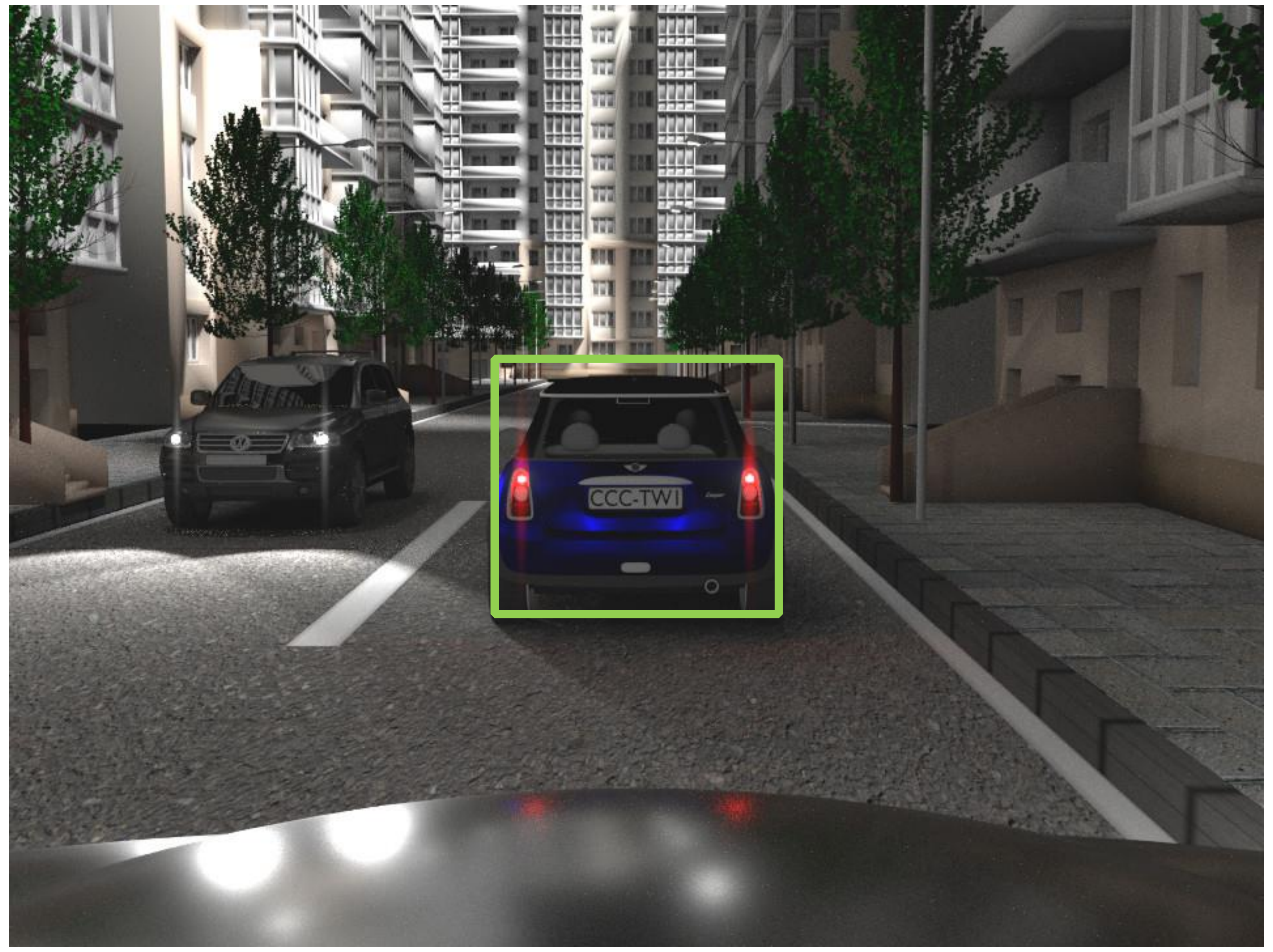
ML Praktikum 17/18

Polynomial regression and overfitting

Problem set

Suppose we have a detector that is able to detect cars in images from a forward-facing camera.

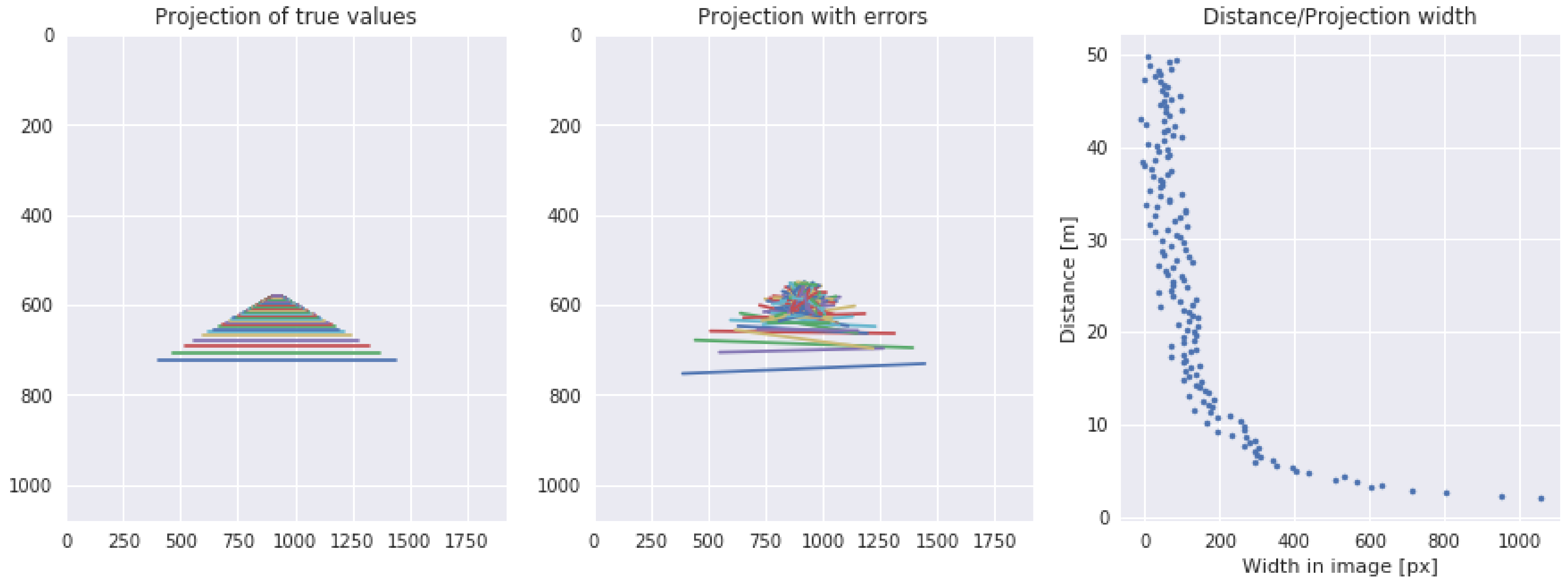




We do not have the means to directly measure the distance of the detected cars.

Problem set

Are we able to infer the distance of a car that has been detected in an image given annotated data?



Assumptions for simplification

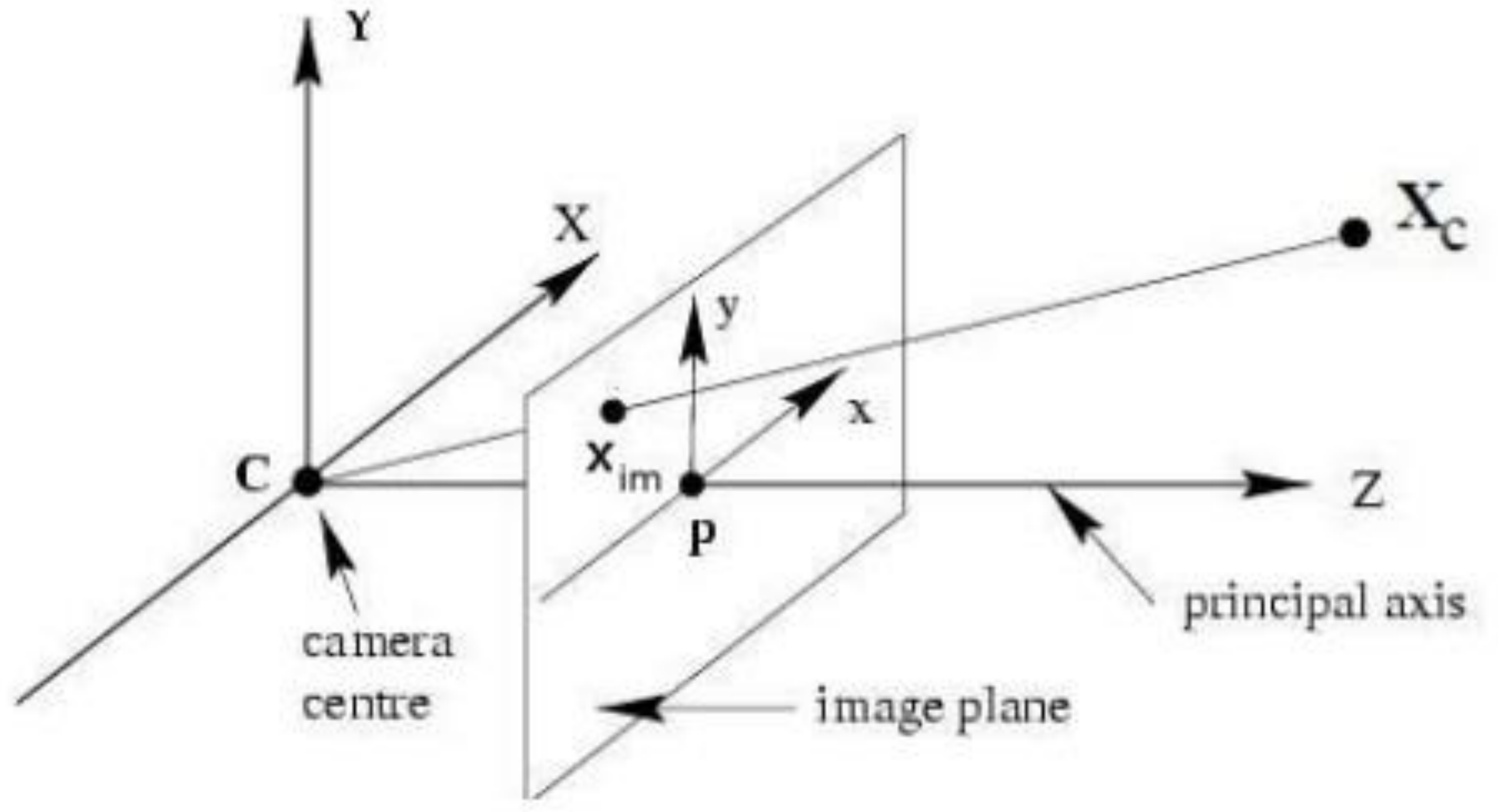
- The position and angle of our camera, as well as extrinsic and intrinsic parameters does not change
- Other cars are on the same groundplane as we are
- Other cars are not rotated
- Other cars all have the same width

Problem set

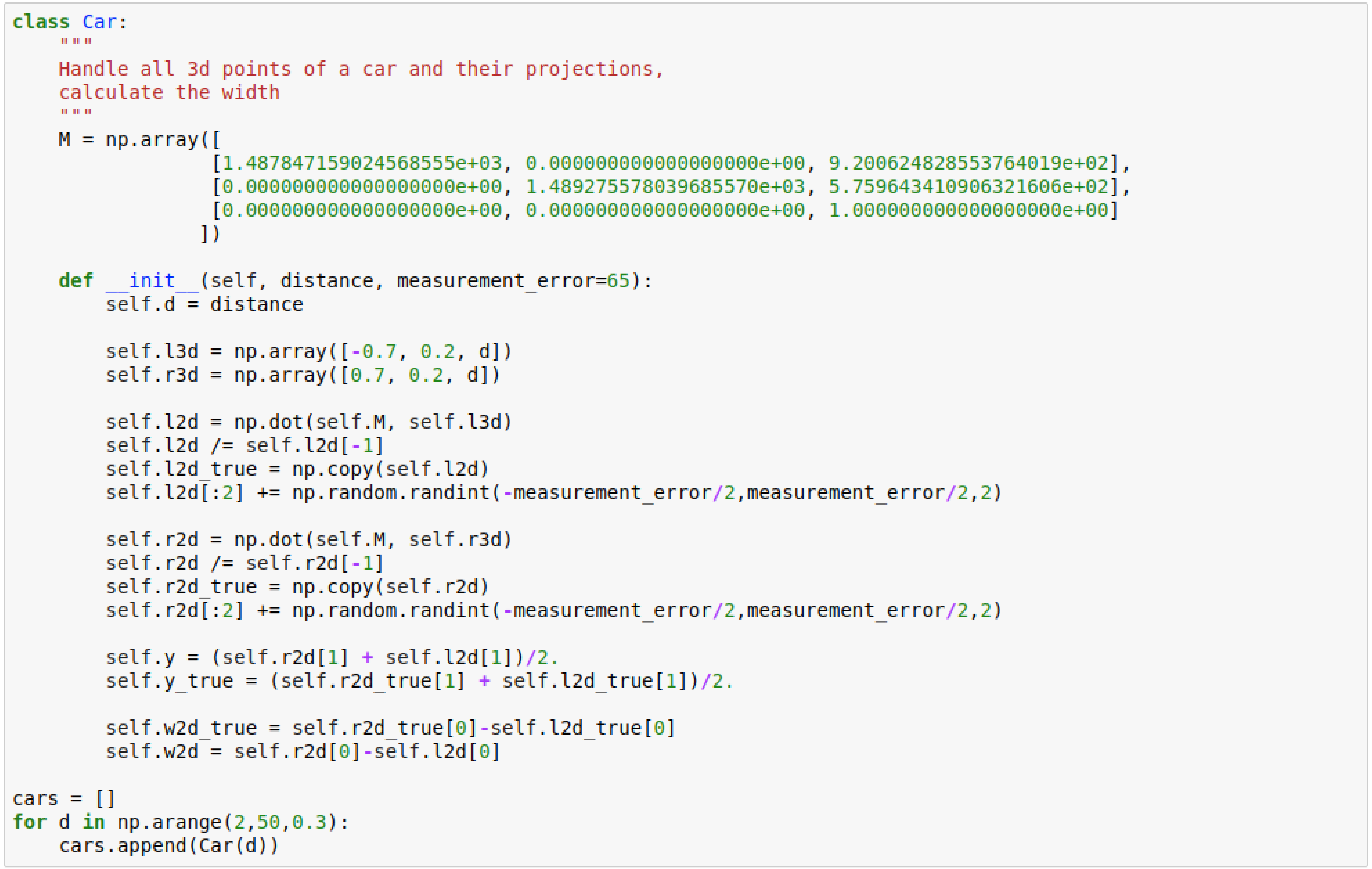
Two ways of generating data:

- Manual annotation by a human
- Simulation
  - Specify 3D-points of a car
  - Project them using camera parameters
  - Save known distances and projections
  - Add measurement error

$$s\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}f_x & 0 & u_0\\0 & f_y & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

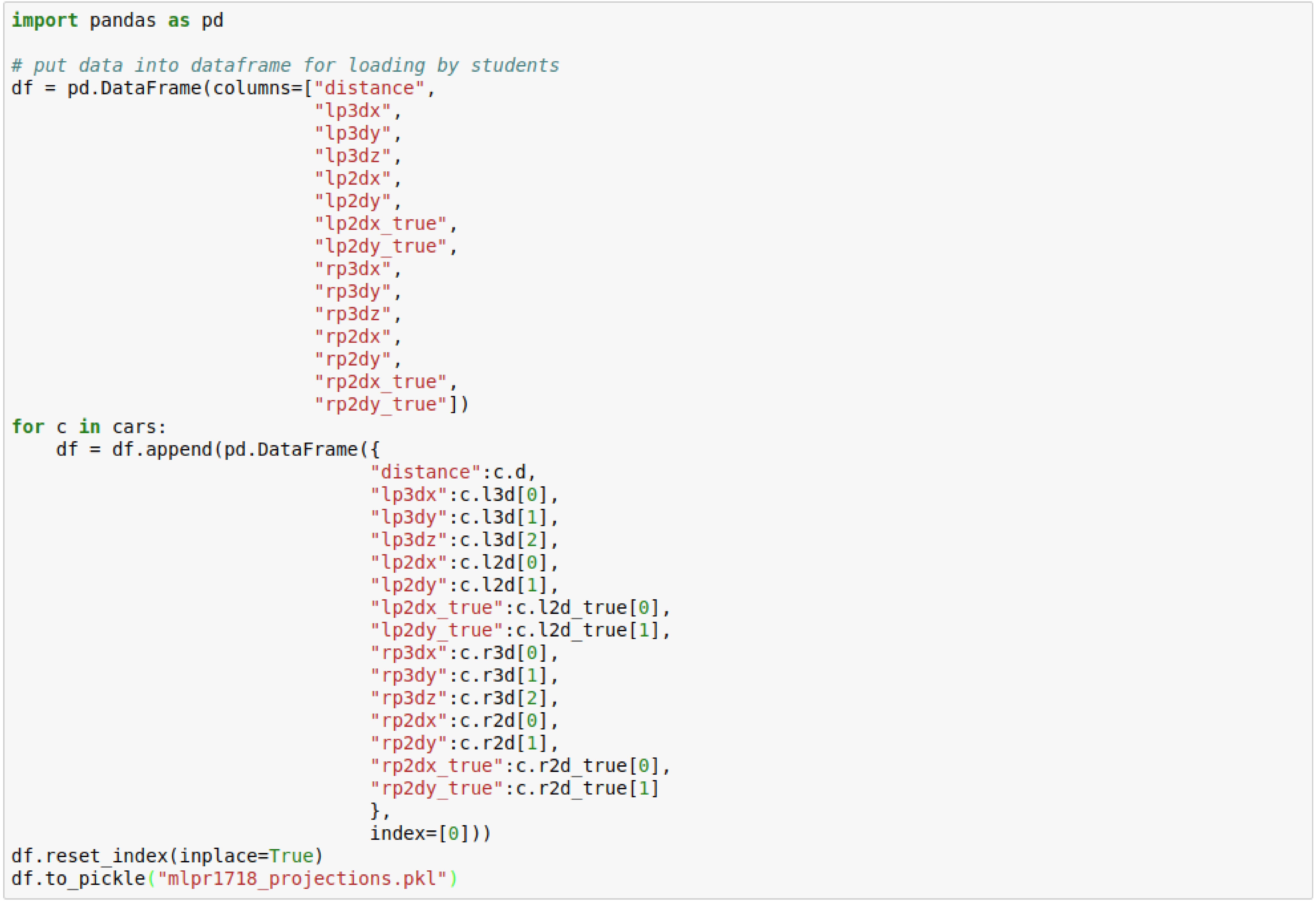


Problem set – data simulation



```
class Car:
    """
    Handle all 3d points of a car and their projections,
    calculate the width
    """
    M = np.array([
                 [1.487847159024568555e+03, 0.000000000000000000e+00, 9.200624828553764019e+02],
                 [0.000000000000000000e+00, 1.489275578039685570e+03, 5.759643410906321606e+02],
                 [0.000000000000000000e+00, 0.000000000000000000e+00, 1.000000000000000000e+00]
               ])

    def __init__(self, distance, measurement_error=65):
        self.d = distance

        self.l3d = np.array([-0.7, 0.2, d])
        self.r3d = np.array([0.7, 0.2, d])

        self.l2d = np.dot(self.M, self.l3d)
        self.l2d /= self.l2d[-1]
        self.l2d_true = np.copy(self.l2d)
        self.l2d[:2] += np.random.randint(-measurement_error/2,measurement_error/2,2)

        self.r2d = np.dot(self.M, self.r3d)
        self.r2d /= self.r2d[-1]
        self.r2d_true = np.copy(self.r2d)
        self.r2d[:2] += np.random.randint(-measurement_error/2,measurement_error/2,2)

        self.y = (self.r2d[1] + self.l2d[1])/2.
        self.y_true = (self.r2d_true[1] + self.l2d_true[1])/2.

        self.w2d_true = self.r2d_true[0]-self.l2d_true[0]
        self.w2d = self.r2d[0]-self.l2d[0]

cars = []
for d in np.arange(2,50,0.3):
    cars.append(Car(d))
```

Problem set – data simulation



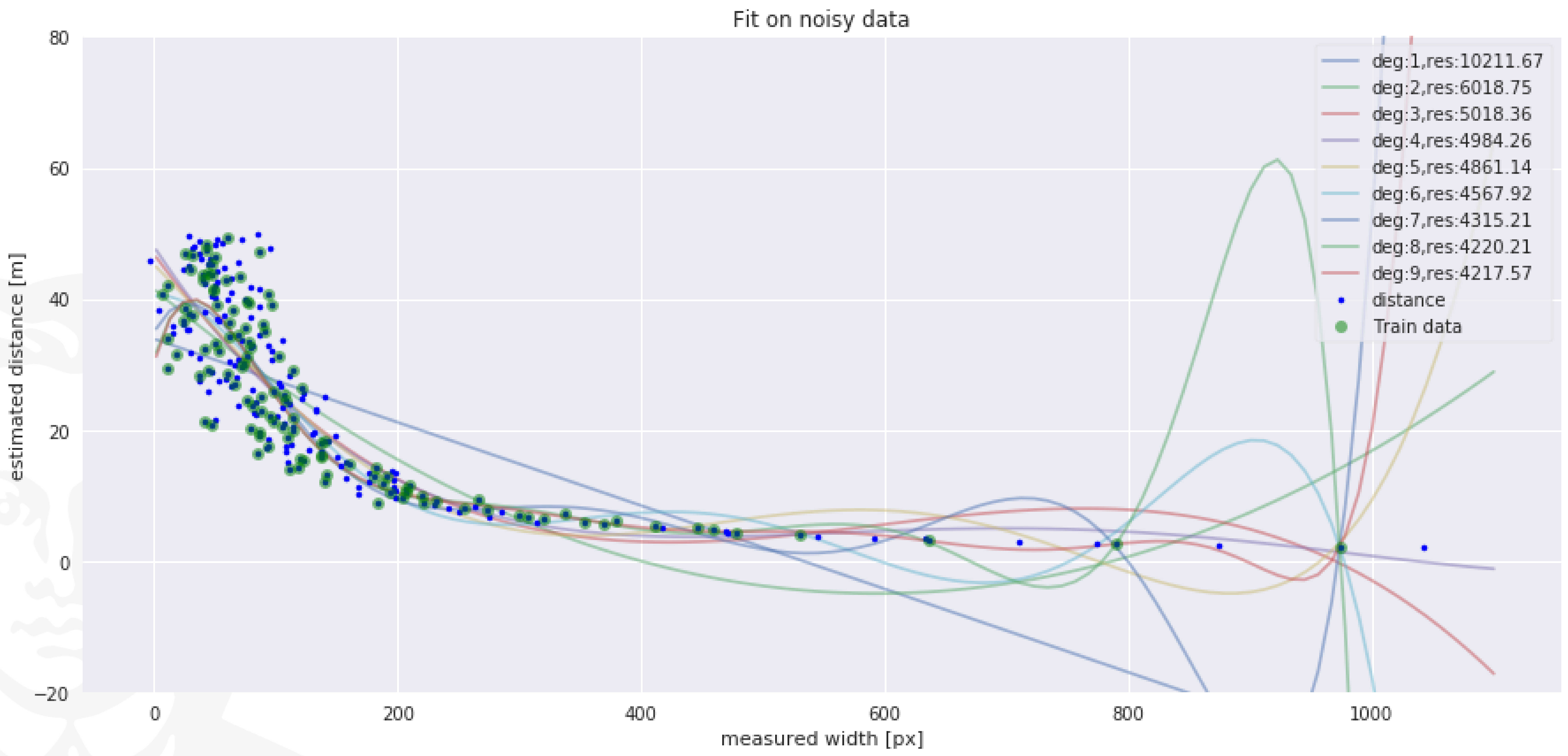
```python
import pandas as pd

# put data into dataframe for loading by students
df = pd.DataFrame(columns=["distance",
                           "lp3dx",
                           "lp3dy",
                           "lp3dz",
                           "lp2dx",
                           "lp2dy",
                           "lp2dx_true",
                           "lp2dy_true",
                           "rp3dx",
                           "rp3dy",
                           "rp3dz",
                           "rp2dx",
                           "rp2dy",
                           "rp2dx_true",
                           "rp2dy_true"])
for c in cars:
    df = df.append(pd.DataFrame({
                                "distance":c.d,
                                "lp3dx":c.l3d[0],
                                "lp3dy":c.l3d[1],
                                "lp3dz":c.l3d[2],
                                "lp2dx":c.l2d[0],
                                "lp2dy":c.l2d[1],
                                "lp2dx_true":c.l2d_true[0],
                                "lp2dy_true":c.l2d_true[1],
                                "rp3dx":c.r3d[0],
                                "rp3dy":c.r3d[1],
                                "rp3dz":c.r3d[2],
                                "rp2dx":c.r2d[0],
                                "rp2dy":c.r2d[1],
                                "rp2dx_true":c.r2d_true[0],
                                "rp2dy_true":c.r2d_true[1]
                                },
                                index=[0]))
df.reset_index(inplace=True)
df.to_pickle("mlpr1718_projections.pkl")
```

Your part – practice time

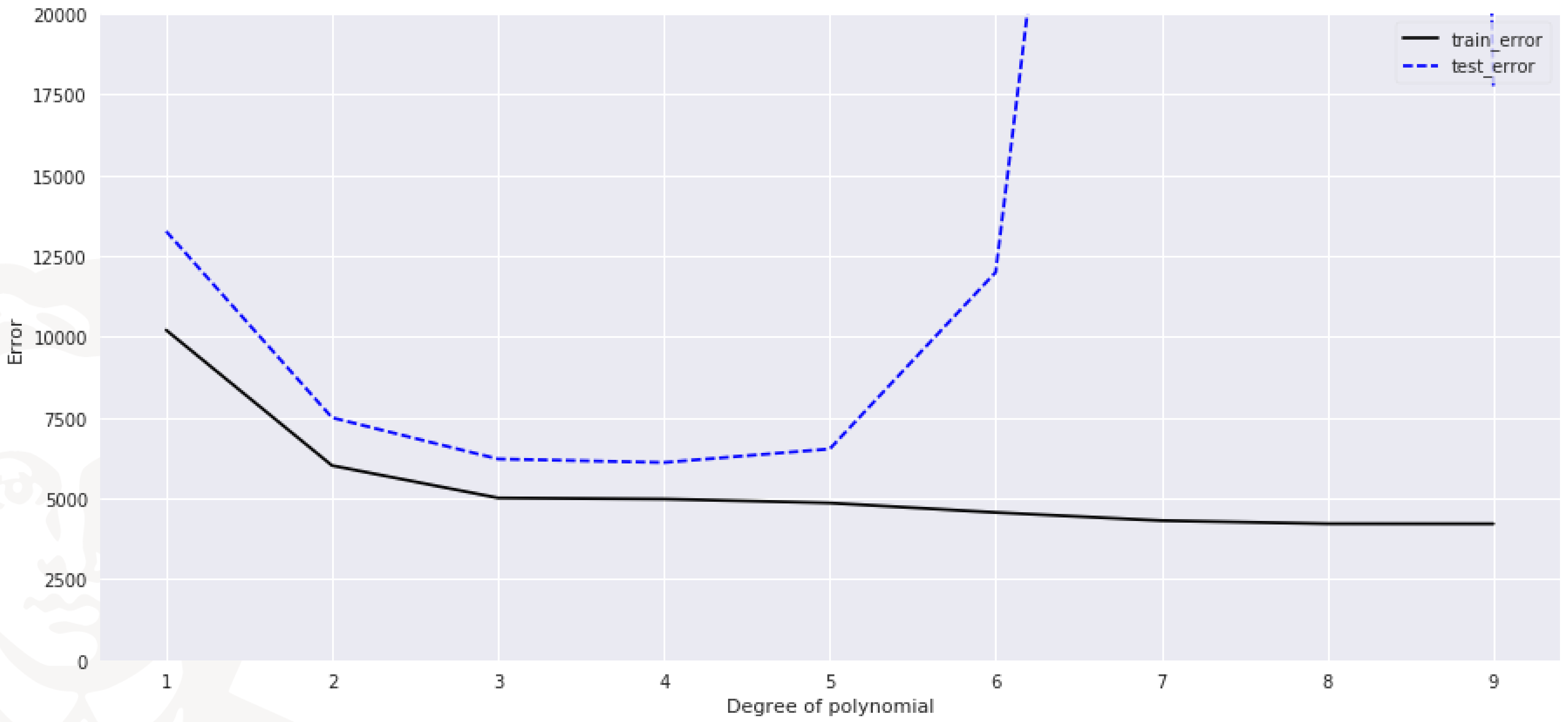
Now that you have seen how the data has been generated, you will have to do the rest:

- Read the saved file from above („mlpr1718_projections.pkl“) into a dataframe
- Create a new column in the dataframe that contains the width of the 2D projection (just use rp2dx – lp2dx)
- Create a plot that contains the following three subplots:
  - Projections in 2D of the true values
  - Projections in 2D of the „annotated“ values with errors
  - A plot displaying the width of each car against it‘s distance
- Split the data into train- and test-set (use 50/50)
- Use the function np.polynomial.polynomial.polyfit() to fit polynomials of degrees 1-10
- Calculate the squared error for train and test for each polynomial
- Plot the datapoints and overlay each polynomial in the plot
- Display the train- and test-errors with respect to the degree of the polynomial
- Automatically select the fit with the least test error and display the polynomial on top of the data

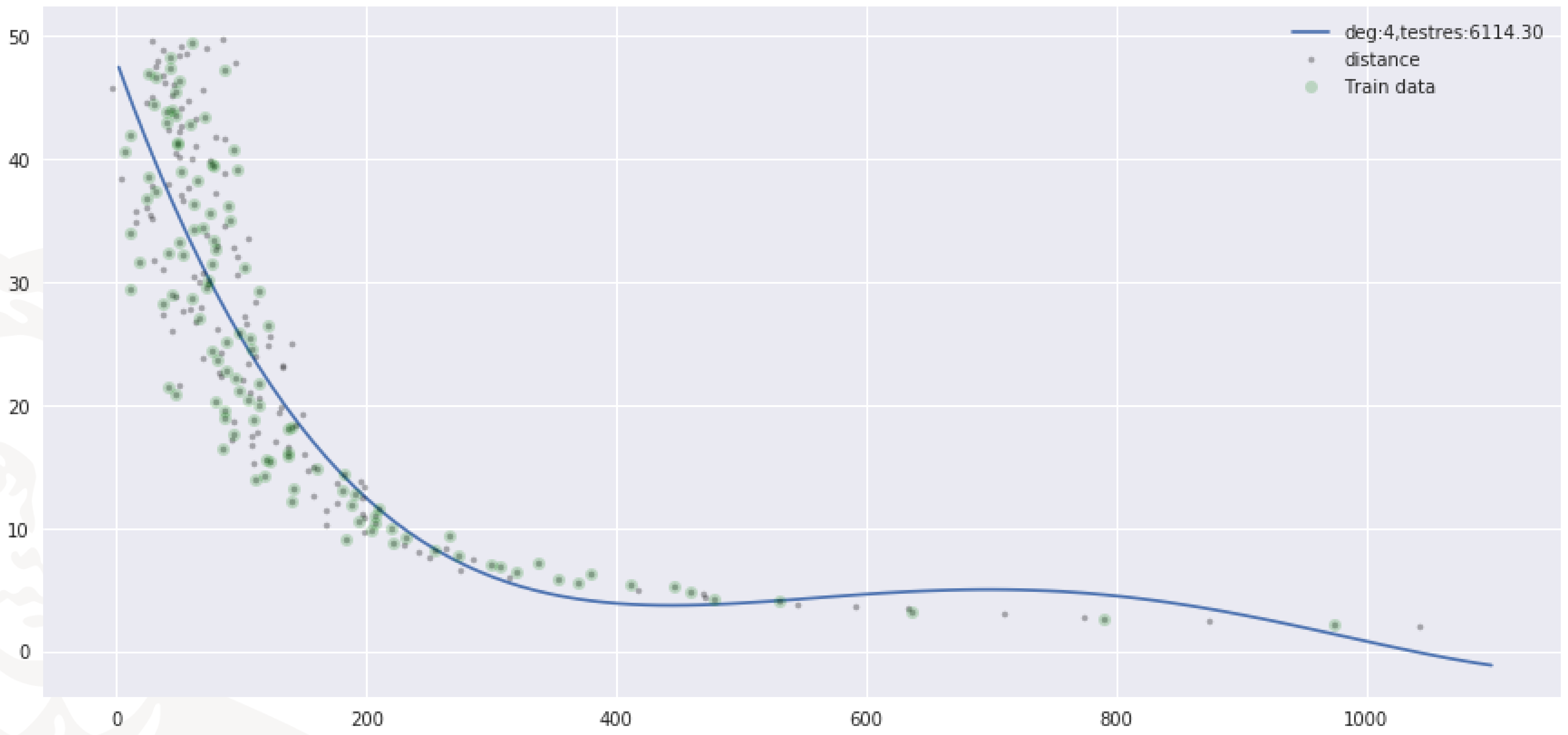
Examples – All polygonfits



Examples – Train/Test error for different degrees



Examples – Fit with least test error


deg:4,testres:6114.30
distance
Train data