My approach to the problem:

- Tracking the ball's position in each frame of the video.
- Calculating the ball's velocity.
- Using a numerical optimization method to estimate the ball's mass, drag coefficient, and rolling coefficient, based on the observed acceleration values.
- Testing the method with a "good" video and a "bad" video with poor tracking.

These are explanations of the functions:

track_ball(video_path):

Tracks the ball's position in a video using background subtraction, writes the video to file.

Only argument is video_path (str: Path to the video file).

We return a tuple containing: list - time points (s), numpy.ndarray - x coordinates of the ball's position (m), numpy.ndarray - y coordinates of the ball's position (m), frame rate of the video, frame size (tuple),

numpy.ndarray - radii of the ball for each frame in pixels.

euler_step(y, t, dt, drag_coeff, rolling_coeff, mass, g):

This method will take an Euler step for a given state using drag and gravity.

Arguments are:

- y (numpy.ndarray): the current position and velocity of the ball.
- t (float): the current time.
- dt (float): the time step.
- drag_coeff (float): The drag coefficient of the ball.
- rolling_coeff (float): Rolling coefficient of the ball on the ground.
- mass (float): the mass of the ball.
- g (float): acceleration due to gravity.

And in the end we return the next position and velocity of the ball.

objective_function(params, time_points, initial_state, g, observed_positions_x,observed_positions_y, observed_vx, observed_vy):

This is the objective function to minimize, which will compare the observed positions, and the positions calculated from ODE.

Arguments are:

- params (numpy.ndarray): The initial guess for the mass and drag coefficient of the ball.
- time_points (list): time points (s).
- initial_state (numpy.ndarray): The initial position and velocity of the ball.
- g (float): acceleration due to gravity.
- observed_positions_x (numpy.ndarray): x coordinates of the ball's position (m).
- observed_positions_y (numpy.ndarray): y coordinates of the ball's position (m).
- observed_vx (numpy.ndarray): The observed velocity in the x direction.
- observed_vy (numpy.ndarray): The observed velocity in the y direction.

And we return the sum of square differences between the observed and calculated positions.

estimate_parameters(time_points, x_positions, y_positions, initial_state, g, vx, vy):

This method will estimate parameters (mass, and drag coefficient) of a ball, using numerical methods and the observed position.

Arguments are:

- time_points (list): time points (s).
- x_positions (numpy.ndarray): x coordinates of the ball's position (m).
- y_positions (numpy.ndarray): y coordinates of the ball's position (m).
- initial_state (numpy.ndarray): The initial position and velocity of the ball.
- g (float): acceleration due to gravity.
- vx (numpy.ndarray): The observed velocity in the x direction.
- vy (numpy.ndarray): The observed velocity in the y direction.

And we return a tuple containing the estimated mass and drag coefficient of the ball.

convert_pixels_to_meters(pixels, real_length_meters, pixels_length):

This method will convert pixel units to meters.

Arguments are:

- pixels (numpy.ndarray): pixel lengths.
- real_length_meters (float): real length in meters.
- pixels_length (numpy.ndarray): length of the real object in the image in pixels.

And we return the length in meters.

calculate_velocities(time_points, x_positions, y_positions, radii, ball_radius_meters):

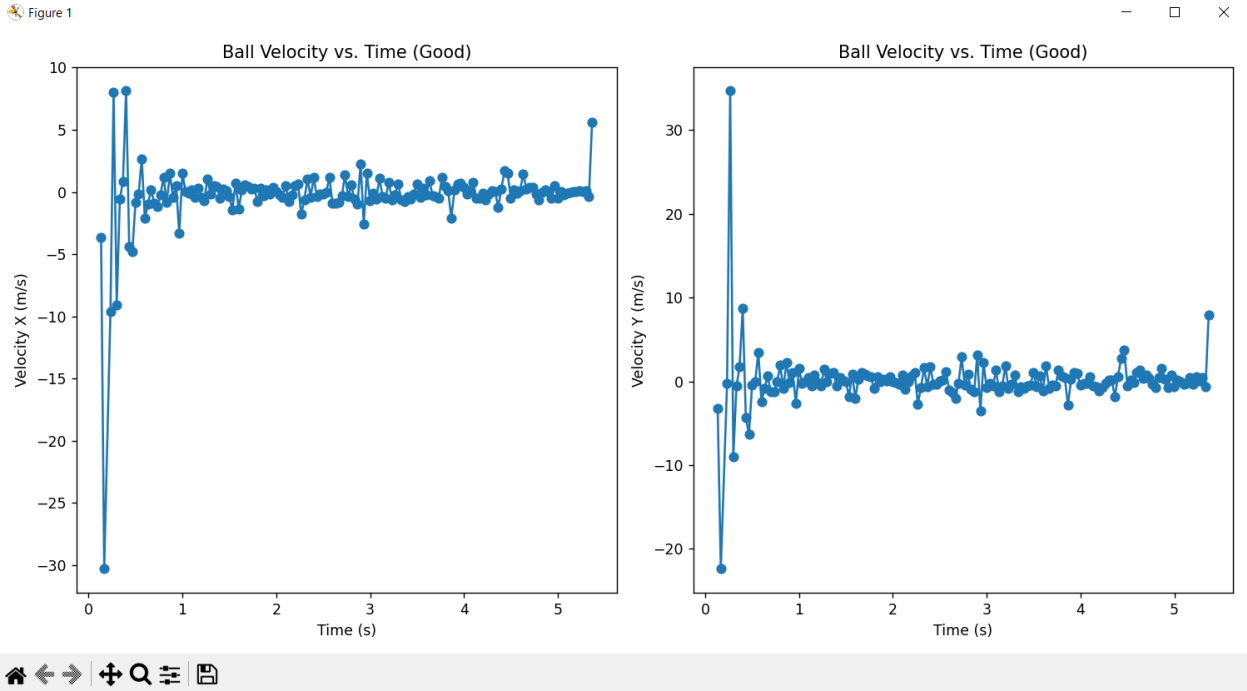
Calculates the ball's velocities.

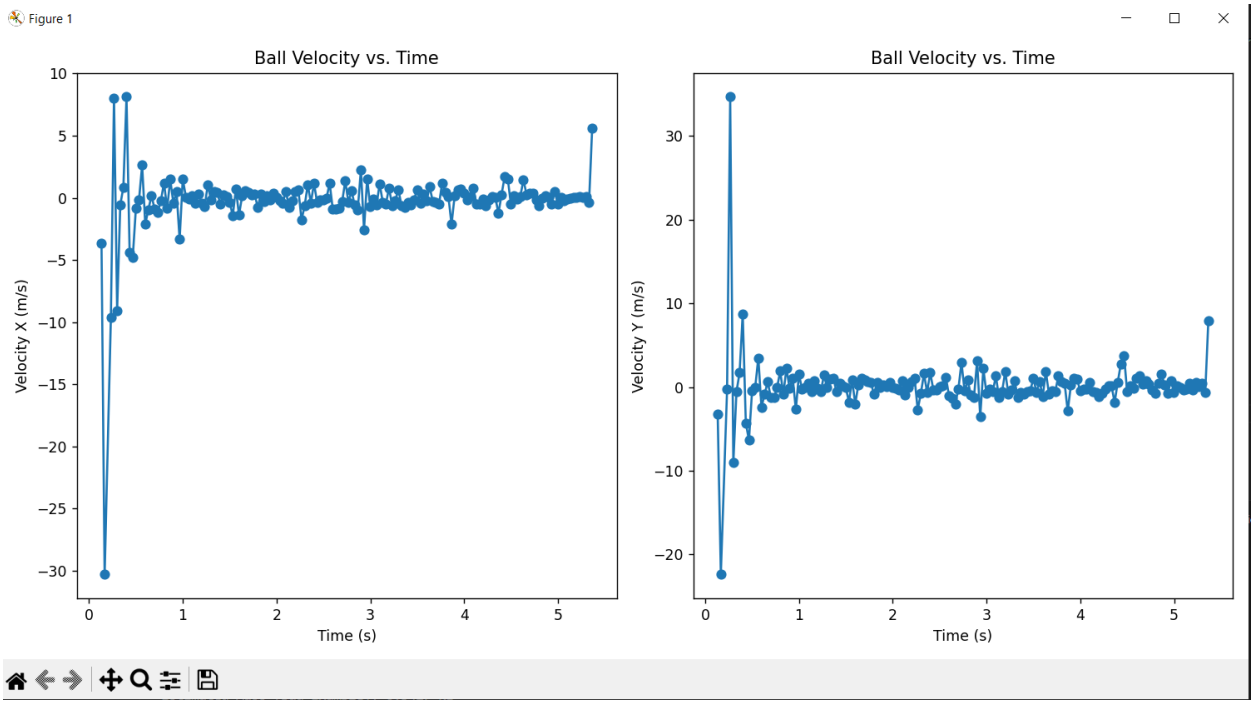
Arguments are:

- time_points (list): time points (s).
- x_positions (numpy.ndarray): x coordinates of the ball's position (m).
- y_positions (numpy.ndarray): y coordinates of the ball's position (m).
- radii (numpy.ndarray): radii of the ball in pixels
- ball_radius_meters (float): real ball radius in meters

And we return x velocity and y velocity.

These are the results:





Estimated Mass: 0.4695 kg

Estimated Drag Coefficient: 0.3942

Estimated Rolling Coefficient: 4.0755

Estimated Mass (Bad Example): 0.3427 kg

Estimated Drag Coefficient (Bad Example): 0.3999

Estimated Rolling Coefficient (Bad Example): 2.8763

From the graphs we can't really see the difference but in reality, if you watch the videos in good video ball tracking is great and in the second video it is terrible. Also, we have 0.47 and 0.4 masses but in the first video we have a football and in the second video we have a golf ball, for football the result is pretty accurate, same cannot be said for the golf ball. These bad results are not only because of bad detection in the second video. We are also looking at the ball at a bad angle and it seems like it is not moving at all.