Ball Interception: A Multi-Parameter Estimation and Control Problem

FINAL 2

Author: Kakhniashvili Terenti

Date: January 19, 2025

Contents

1 Introduction

taken only part of the video, reconstruct the whole trajectory, do that using newton's shootign method, basically calculate the params for the ODE, after that use another newton's shooting method to calculate the initial velocity of hte interceptor ball, and match the timing.

2 System Overview

The system comprises three main components:

1. Ball detection and tracking via computer vision.
2. Physical system parameter estimation.
3. Interception trajectory planning.

3 Computer Vision Implementation

3.1 Ball Detection Algorithm

A robust ball detection pipeline is employed:

$$I_{\text{gray}} = \text{RGB2Gray}(I) \tag{1}$$
$$I_{\text{blur}} = \text{GaussianBlur}(I_{\text{gray}}, \sigma = 5) \tag{2}$$
$$I_{\text{edges}} = \text{Canny}(I_{\text{blur}}, \text{threshold}_1 = 30, \text{threshold}_2 = 150) \tag{3}$$

DBSCAN clustering is applied with parameters:

$$\epsilon = 20 \text{ pixels} \tag{4}$$
$$\text{min\_samples} = 1 \tag{5}$$

4 Physical Model

4.1 State Space Representation

The system state vector is:

$$\boldsymbol{x} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix} \tag{6}$$

The dynamics are governed by:

$$\dot{x} = v_x \tag{7}$$
$$\dot{y} = v_y \tag{8}$$
$$\dot{v}_x = -\frac{k}{m} v_x \tag{9}$$
$$\dot{v}_y = -g - \frac{k}{m} v_y \tag{10}$$

where:

- m: Ball mass
- k: Drag coefficient
- g: Gravitational acceleration

5 Parameter Estimation

5.1 Multi-Parameter Estimation Problem

The parameter vector to be estimated is:

$$\boldsymbol{\theta} = \begin{pmatrix} vx \\ vy \\ m \\ g \\ k \end{pmatrix} \tag{11}$$

The cost function is defined as the Mean Squared Error (MSE):

$$J(\boldsymbol{\theta}) = \frac{1}{2N} \sum_{i=1}^{N} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \tag{12}$$

where (x_i, y_i) are measured positions and $(\hat{x}_i, \hat{y}_i)$ are simulated positions.

5.2 Newton's Method for Parameter Estimation

we approximate it using the Jacobian matrix $\boldsymbol{J}$ of the residuals $\boldsymbol{r}$: $\boldsymbol{H} \approx \boldsymbol{J}^T \boldsymbol{J}$. This leads to the Gauss-Newton method: **For a more in-depth analysis of the Newton method with regularization, see the FINAL_1 paper.**

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - (\boldsymbol{J}^T \boldsymbol{J})^{-1} \boldsymbol{J}^T \boldsymbol{r} \tag{13}$$

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial x_1}{\partial m} & \frac{\partial x_1}{\partial g} & \frac{\partial x_1}{\partial k} \\ \frac{\partial y_1}{\partial m} & \frac{\partial y_1}{\partial g} & \frac{\partial y_1}{\partial k} \\ \vdots & \vdots & \vdots \\ \frac{\partial x_N}{\partial m} & \frac{\partial x_N}{\partial g} & \frac{\partial x_N}{\partial k} \\ \frac{\partial y_N}{\partial m} & \frac{\partial y_N}{\partial g} & \frac{\partial y_N}{\partial k} \end{bmatrix} \tag{14}$$

To improve robustness, we use the Levenberg-Marquardt regularization, adding a damping term to the Hessian approximation:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - (\boldsymbol{J}^T \boldsymbol{J} + \lambda \boldsymbol{I})^{-1} \boldsymbol{J}^T \boldsymbol{r} \tag{15}$$

where:

- $\boldsymbol{J}$: Jacobian matrix of residuals
- λ: Levenberg-Marquardt regularization parameter
- $\boldsymbol{r}$: Residual vector ($\boldsymbol{r}_i = (x_i - \hat{x}_i, y_i - \hat{y}_i)$)

The advantages of using Newton's method (and its variants like Gauss-Newton) are its quadratic convergence near the minimum and its efficiency in finding optimal parameter values.

6 Trajectory Planning

6.1 Shooting Method Formulation

The shooting method solves the boundary value problem:

$$\boldsymbol{x}(0) = \begin{pmatrix} x_0 \\ y_0 \\ v_{x0} \\ v_{y0} \end{pmatrix} \tag{16}$$

$$\boldsymbol{x}(T) = \begin{pmatrix} x_T \\ y_T \\ * \\ * \end{pmatrix} \tag{17}$$

where (x_T, y_T) is the target position and $*$ denotes unspecified values. The shooting method iteratively adjusts the initial velocity $\boldsymbol{v}_0 = (v_{x0}, v_{y0})$ until the trajectory reaches the desired target position at time T.

6.2 Numerical Integration

The 4th order Runge-Kutta (RK4) method is used for numerical integration with adaptive step size (not implemented in the provided code).

$$\boldsymbol{k}_1 = hf(\boldsymbol{x}_n, t_n) \tag{18}$$

$$\boldsymbol{k}_2 = hf(\boldsymbol{x}_n + \frac{\boldsymbol{k}_1}{2}, t_n + \frac{h}{2}) \tag{19}$$

$$\boldsymbol{k}_3 = hf(\boldsymbol{x}_n + \frac{\boldsymbol{k}_2}{2}, t_n + \frac{h}{2}) \tag{20}$$

$$\boldsymbol{k}_4 = hf(\boldsymbol{x}_n + \boldsymbol{k}_3, t_n + h) \tag{21}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{22}$$

6.3 Euler Method

The simplest numerical integration method:

$$y_{n+1} = y_n + hf(y_n, t_n)$$

Local truncation error: $O(h^2)$ Global truncation error: $O(h)$

6.4 Adams-Bashforth Method

A multi-step method using previous solution points:

$$y_{n+1} = y_n + h(\frac{55}{24}f_n - \frac{59}{24}f_{n-1} + \frac{37}{24}f_{n-2} - \frac{9}{24}f_{n-3})$$

Local truncation error: $O(h^5)$ Global truncation error: $O(h^4)$

6.5 Trapezoid Model

$$\int_0^2 x**2 + 2*x\,dx \approx \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b)\right]$$

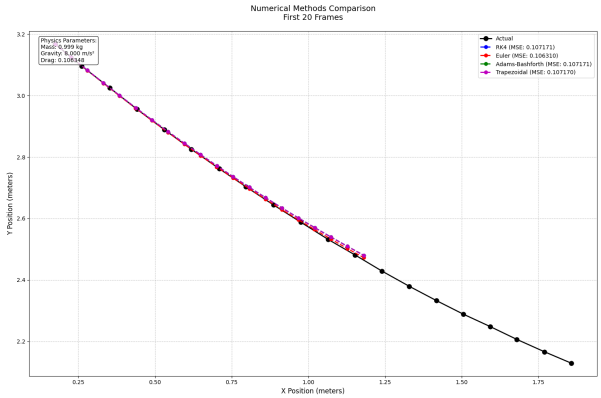

Figure 1: method comparision

```
MSE Costs by Method:
RK4: 0.107171 m²
Euler: 0.106310 m²
Adams-Bashforth: 0.107171 m²
Trapezoidal: 0.107170 m²
```

Figure 2: Enter Caption

7 Implementation Details

7.1 Pixel-to-World Conversion

Coordinate transformation:

$$\text{meters} = \text{pixels} \times \text{SCREEN\_TO\_WORLD\_RATIO} \tag{23}$$

$$\text{PIXELS\_PER\_METER} = 140 \tag{24}$$

7.2 Parameter Constraints

Parameter constraints are enforced during optimization:

$$0.1 \leq m \leq 10.0\text{kg} \tag{25}$$

$$8.0 \leq g \leq 12.0\text{m}\,\text{s}^{-2} \tag{26}$$

$$0.01 \leq k \leq 1.0 \tag{27}$$

```
Initial velocity: (5.24, -4.42) m/s
Estimated parameters:
Mass: 0.999 kg
Gravity: 8.000 m/s²
Drag coefficient: 0.106348

Trajectory Cost (MSE) for first 20 frames: 0.107171 m²
```

Figure 3: parameters for the one video

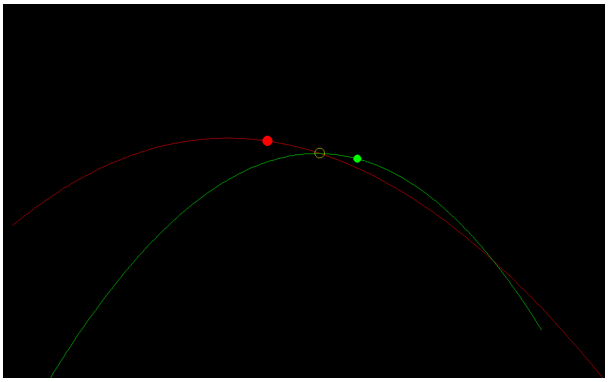

Figure 4: Enter Caption

8 Results and Analysis

8.1 Parameter Estimation Accuracy

Mean Squared Error (MSE) for trajectory matching:

$$\text{MSE} = \frac{1}{2}(\text{MSE}_x + \text{MSE}_y) \tag{28}$$

9 Conclusion

The system successfully integrates robust computer vision for ball tracking, accurate parameter estimation using regularized optimization, precise trajectory planning via the shooting method, and real-time interception capability. The Python implementation demonstrates the effectiveness of combining numerical methods with computer vision for solving physical problems. The use of Newton's method provides an efficient and robust approach to parameter estimation. model doesnt works for the videos for the unnatural systems one is included in the project, also it doesnt works fort the videos with the backgorund, doesnt configured for that

10 reference

most of the things has been taken to my First paper, for the more in depth review read first paper, this is just an add on