Target Interception Report

Animal Locomotion and Bioinspired Robots

Author: Aaruni Arora
Supervisor: Prof Huai Ti Lin

Date: May 23, 2025

1 Introduction

Bioinspired robots draw on the principles of biological systems to develop autonomous agents capable of intelligent and adaptive behaviour. In particular, predator–prey dynamics help study sensorimotor integration, the process by which sensory inputs are transformed into motor commands. This has applications in autonomous driving or missile guidance systems, and maritime navigation, for example. Thus, this report presents a simulated pursuit task where in a robot (agent) attempts to intercept a moving prey (target) with different pursuit strategies under varying levels of delay and noise, mimicking real-life sensory and actuation constraints.

2 Methods

The following strategies use the Proportional-Integral-Derivative (PID) controller, a feedback mechanism that regulates the heading of an agent toward a moving target based on visual cues.

2.1 Proportional-Integral-Derivative (PID) Controller

The Proportional-Integral-Derivative (PID) controller gives a robust way to adjust the robot's heading based on the error between its current direction and the desired value, which in our case is the position of the prey. The heading rate $\dot{\theta_h}$ is given by

$$\dot{\theta_h} = K_p(\theta_r - \theta_h) + K_I \int (\theta_r - \theta_h)dt + K_d \frac{d\theta_r}{dt}, \tag{1}$$

where θ_h is the current heading angle, θ_r the reference angle to the target, and K_P, K_I and K_D the proportional, integral, and derivative gains, respectively.

2.2 Simple Pursuit

Simple or Pure Pursuit [1] is a reactive strategy where the robot continuously steers towards the target's current position, mimicking it's path. It minimises the angular difference e to bring the agent's heading directly in line with the target's direction to intercept it. The control law is thus given by

$$\dot{\theta_h} = K_p(\theta_r - \theta_h) = K_p e \quad \text{where} \quad e = \theta_r - \theta_h. \tag{2}$$

Though this is an easy to implement proportional controller, the algorithm often results in tail-chasing behaviour, causing suboptimal interception paths against agile targets.

2.3 Constant Bearing

Building on Simple Pursuit, a robot with a Constant Bearing (CB) [2] strategy maintains a fixed angle between its own heading and the line-of-sight (LOS) to the prey, effectively trying to intercept the prey. The control law is given by

$$\dot{\theta_h} = K_p(\theta_r - \theta_h) + K_I \int (\theta_r - \theta_h)dt. \tag{3}$$

This method requires more fine-tuned control but can intercept moving targets with constant velocity more efficiently than simple pursuit. The flexibility of angle-setting makes this algorithm useful in maritime navigation or missile guidance.

2.4 Parallel Navigation

In Parallel navigation [3], the robot maintains a constant absolute LOS angle to a global frame reference, i.e, the agent and the target move in parallel paths with an adjusted heading such that the lines between them converge at the interception point. Thus, $\phi = \theta_r^{global} - \theta_h^{global} = constant$, where ϕ is the absolute target angle.

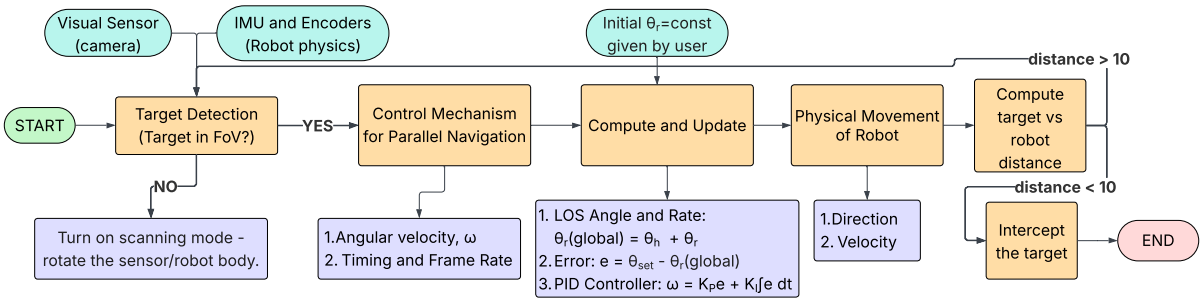


Figure 1: Flowchart of the Parallel Navigation control system that operates without requiring any hardware modifications.

2.5 Proportional Navigation

Here, the robot turns at a rate proportional to the rate of change of the LOS angle $\dot{\lambda}$ to the target. This makes this a D-controller of the PID framework. Thus, with the navigation constant $N = K_D$ [4], the control law is given by

$$\dot{\theta_h} = N\dot{\theta_r} = K_D\dot{\theta_r}. \tag{4}$$

This method, a sub-part of parallel navigation [5], is useful for fast or evasive targets, as we use the relative rate of change instead of the angular difference to intercept the target.

2.6 Motion Camouflage

Motion Camouflage is equivalent to PN, in that it has a constant global retrospective angle $\phi = constant$, but with an additional distant anchor. This is dependent on K_P.

2.7 Noise and Sensory Delay

To simulate real-world uncertainty, Gaussian noise is added to robot perception ($\sigma = 2$ px for position, 2° for angle) and a sensorimotor delay is tested from 0 to 50 frames.

2.8 Simulation Setup

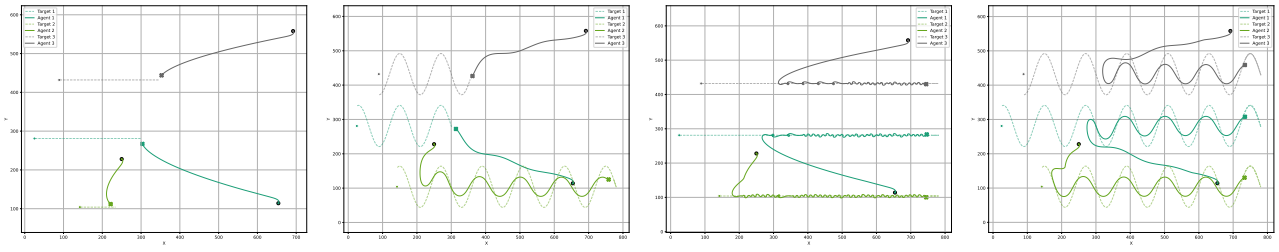
All simulations were performed in a 2D environment using Python (Pygame), and the space was restricted to 800 by 600 pixels. The robot and prey were randomly positioned at the start of each trial, with a point target prey's speed and trajectory varying trial-to-trial. The simulation explores four strategies: (i) Simple pursuit (SP); (ii) Constant-Bearing (CB); (iii) Parallel and Proportional Navigation (PN); and (iv) Motion Camouflage (MC). The prey follows one of two motion patterns: linear (x-y velocities are constants), sinusoidal ($y = 60sin(2\pi f)$, for $f = 0.008Hz$). Each method is evaluated against the mean time taken to intercept/capture the target over 10 trials. There is a set maximum duration of 60s, beyond which the robot is assumed to have failed the task.

3 Results and Discussion

3.1 Trajectories

Fig 2 plots the path of 3 trajectories taken by the robot to intercept the target moving with both linear and sinusoidal motion for a simple pursuit strategy. We can draw the following conclusions from this:

1. In Fig 2a, all agents successfully intercept the target with relatively direct and smooth paths. The linear motion makes the interception problem geometrically simpler, which serves as our baseline.



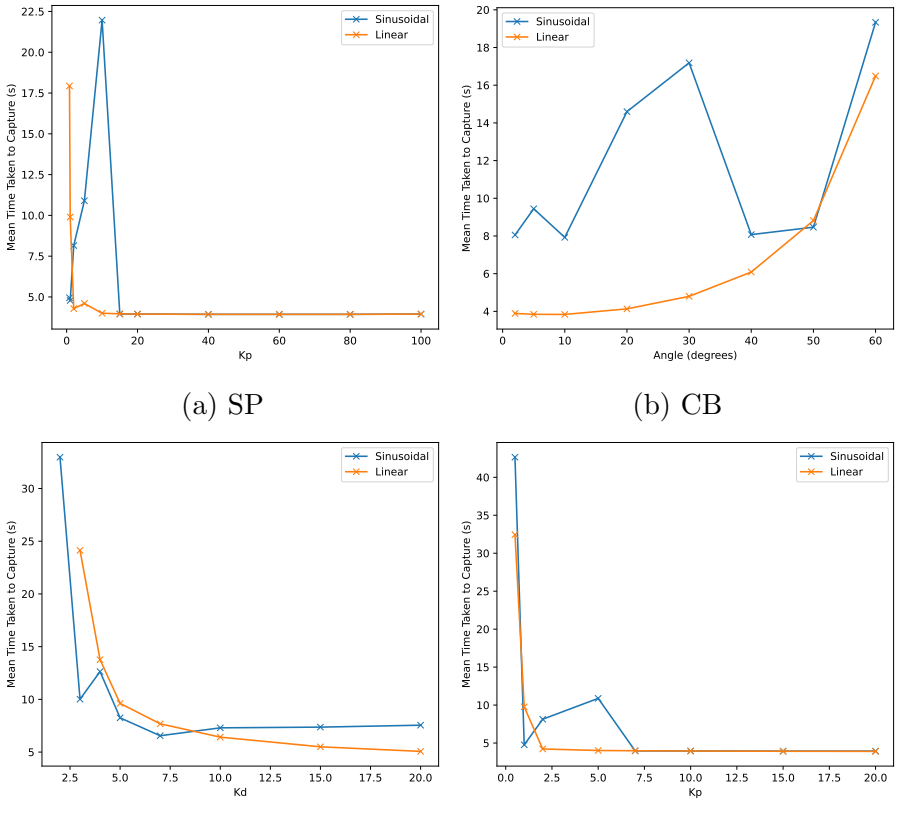
(a) Linear (delay=0s) (b) Sine (delay=0s) (c) Linear (delay=50s) (d) Sine (delay=50s)

Figure 2: Simple Pursuit Robot trajectories for linear and sinusoidal-based motion of the target, with and without sensory delay (delay=50).

2. In Fig 2b, paths are more curved and complex, requiring higher control. Thus, the agents have to adapt. This can result in inefficient paths.

3. Fig 2c and 2d, which include sensorimotor delay, show the phase lag in the control loop. The delay misaligns θ_r updates with the actual target movement, leading to oscillations or divergence. The agent only catches the prey by chance when the target bounces off a side or fails to complete the simulation in 60s.

3.2 Parameter Tuning



(a) SP (b) CB

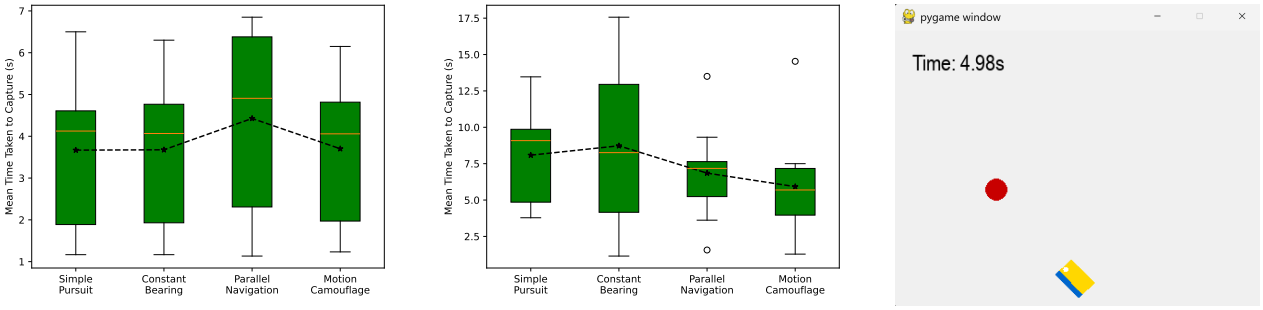
(c) PN (d) MC

Figure 3: Parameter tuning curves for pursuit strategies based on the lowest mean time taken to capture a linear vs sinusoidal target. (a) Simple Pursuit tuned for K_P with K_P optimal at 15. (b) Constant Bearing tuned for bearing angle, with optimal $\theta = 2$ and $\theta = 40$ for linear and sine motion, respectively. (c) Parallel Navigation tuned for K_D, with optimal $K_D = 10$. (d) Motion Camouflage tuned for K_P with optimal $K_P = 7$.

Fig 3 shows the tuning for all parameters of different pursuit strategies. We can draw the following conclusions:

- Simple Pursuit: A very low K_P produces high capture times for both sinusoidal and linear trajectories, and then decreases exponentially after that. This is because a low K_P leads to slow heading direction updates. To avoid overfitting/overshooting and maintain generalisation, $K_P = 15$ was chosen as the optimal value.
- Constant Bearing: Linear and Sinusoidal targets increase in time-capture with increasing constant bearing angle. However, the sinusoidal curve does tend to fall back as a small offset helps perform lateral movement without overturning. Too large a bearing angle resulted in the robot oscillating about a point.
- Parallel Navigation: Decreasing K_D values increases performance but a higher K_D can result in overcompensation, and thus delayed captures.
- Motion Camouflage: Here as well, decreasing K_P increases efficiency. A low gain means the algorithm fails to camouflage but a high gain maintains stealth. Thus, a $K_P = 7$ was chosen for optimality.

3.3 Summary



(a) Linear Summary (b) Sine Summary (c) PyGame Environment

Figure 4: Comparison of all strategies based on their best parameters taken from Fig 3. MC performs best with the least mean time taken to capture across both sine and linear trajectories.

Table 1: Summary of guidance strategy performance under different target trajectories.

Strategy	**Parameter Tuned**	**Best Param (Lin/Sine)**	**Mean Time (s) (Linear/Sine)**
Simple Pursuit (SP)	K_P	10 / 10	$\approx$ 4.2/9.8
Constant Bearing (CB)	Angle (deg)	2 / 40	$\approx$ 4.1/9.1
Parallel Navigation (PN)	K_D	10 / 10	$\approx$ 3.9/6.8
Motion Camouflage (MC)	K_P	7 / 7	$\approx$ 3.8/5.9

Fig 4 and Table 1 lists the comparisons between all strategies, with MC being the most robust algorithm out of all, provided proper parameter tuning. Rest have their specific advantages for various applications where we need to balance complexity and optimality. Although this is the case seen in some practical applications, the simulations here could improve to give better results. For example, more trials could be averaged over to give better estimates. The interception noise could be studied, and real robot deployment can help with more accurate testing.

References

[1] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie Mellon University, Tech. Rep., 1992.

[2] T. I. Fossen and E. Breivik, "Guidance laws for planar motion control," in *Proceedings of the 47th IEEE Conference on Decision and Control*, 2008.

[3] H. Myint and H. M. Tun, "Parallel navigation for 3-d autonomous vehicles," *Kybernetika*, vol. 59, no. 2, pp. 592–605, 2023.

[4] "Introduction to pid controller design," https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID, n.d., control Tutorials for MATLAB and Simulink (CTMS), University of Michigan.

[5] H. Ghaderi, M. Yadegar, N. Meskin, and M. Noorizadeh, "A novel proportional navigation based method for robotic interception planning with final velocity control," *IEEE Access*, vol. 9, pp. 106 428–106 440, 2021.

Thank you to Dr Lin and the GTAs for all their lectures, help and advice. The code can be found on GitHub.