A Proposal: Robotics Tracking Gimbal with Machine Learning

Yenpang Huang

March 2024

1 Introduction

With the growing demand for autonomous systems, there arises an imperative need for robust tracking mechanisms capable of accurately monitoring moving targets. This necessity spans across diverse domains including surveillance, cinematography, and assistive robotics, where the seamless tracking of human movements holds immense value. In this proposal, the integration of a robotics gimbal system onto a drone, meticulously engineered to specialize in tracking walking movements is presented.

2 Objective

The primary objective of this proposal is to develop a robotic gimbal system capable of accurately tracking person movements in real-time. This system aims to seamlessly follow a walking subject, continuously adjusting its orientation and position to maintain tracking throughout the motion.

The core concept of the system involves analyzing image data captured by the vision system frame by frame. Based on this data, the system determines the optimal pan and tilt angles required by the robotics gimbal system to center the target for tracking.

To enhance the gimbal system's accuracy, safety, and self-calibration capabilities, a machine learning approach will be integrated into the system. The chosen method is reinforcement learning (RL). This method will enable the system to learn and adapt over time, resulting in improved stability and performance. Additionally, implementing self-calibration features will significantly reduce the risk of malfunctions. Furthermore, incorporating a learning-capable controller will enhance the drone's versatility, allowing it to adapt to various tracking scenarios and environmental conditions.

By combining traditional control techniques with machine learning methods, the proposed system aims to achieve optimal tracking performance while ensuring robustness and adaptability in real-world applications.

3 Proposed Method

Assumptions for simulation simplicity are essential for the proposed system:

- Person movement speed remains constant and predictable for the robotic system's responsiveness.
- Image data from the vision system is presented as sequences of frames over time.
- The images provided are in grayscale for efficient image processing.
- The drone maintains a consistent distance from the target throughout tracking.

Figure 1 illustrates a conceptual overview of the proposed system. It comprises two main subsystems. Initially, the desired angles are analyzed within the gimbal controller stage. Subsequently, the error between these angles and their current values is computed. This error signal is then fed into a PID controller, which adjusts the system based on proportional, integral, and derivative terms. Ultimately, the calculated output serves as a command to the motor control, facilitating precise gimbal adjustments for effective tracking. Further details on the pseudo code can be found in Appendix 5.

Integrating machine learning into the system introduces complexities, particularly with reinforcement learning. The proposed approach involves adapting the PID controller into an RL-PID controller. By treating the observed errors as states and the system's outputs as rewards, an agent with a predefined policy can be trained to learn the optimal control strategy. Through learning, the agent updates the control gains, leading to refined tuning over time.

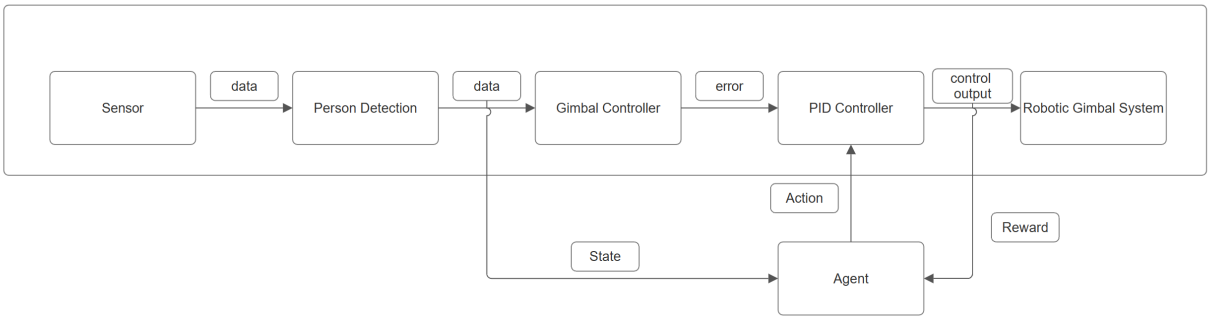

Figure 1: A rough scheme of how the system works.

4 Expectation

The anticipated simulation results should demonstrate the progressive convergence of the robotic gimbal system towards accurate tracking positions, showcasing improved performance over successive iterations. However, it is expected that certain issues may arise regarding the gimbal's responsiveness.

One concern lies in the computational complexity of both the forward and backward pass algorithms, which may consume significant time and memory resources. Given the real-time nature of tracking tasks, it is imperative for the system to provide instant responses to maintain continuous tracking without interruptions.

Moreover, the integration of machine learning techniques introduces additional considerations, including potential weight issues for the drone. The increased power consumption and hardware requirements associated with machine learning algorithms may necessitate careful optimization to mitigate any adverse effects on the system's overall performance and efficiency.

5 Future Work

While the simulation outlined in this proposal acknowledges its limitations for practical application in real-world scenarios, there exists potential for adaptation through reinforcement learning systems within autonomous frameworks. Beyond tracking human movement, critical components such as path planning and trajectory optimization play pivotal roles in project success.

It is essential to recognize that while the method proposed here may require considerable effort, the availability of Python libraries can significantly streamline the implementation process. Moreover, exploring alternative machine learning methods, such as fitness functions for PID tuning, could prove promising in enhancing system performance.

Furthermore, there is ample room for experimentation with the training process to maximize efficiency and effectiveness. Future efforts should focus on a comprehensive exploration of these aspects to refine the proposed system's capabilities.

Despite its challenges, the integration of control and machine learning remains an intriguing and promising topic. With continued development, the combination of both technologies holds the potential to unlock new capabilities for drones in various applications. As such, future research endeavors should aim to leverage these advancements to further enhance the capabilities and adaptability of drone systems.

References

Appendix

```
class Gimbalcontrol:
# Orientation of the gimbal
def set_orientation(self, pan_angle, tilt_angle):
    command = ... # Generate a command with the specified pan and tilt angles for motor
        control

class PIDcontrol:
    def __init__(self, Kp, Ki, Kd):
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd
        self.prev_error = 0
        self.integral = 0

    def PID(self, error):

        # P = self.Kp*error
        # self.integral += error
        # I = self.Ki*self.integral
        # D = self.Kd*(error - self.prev_error)
        # output = P+I+D
        # self.prev_error = error

        # Unlike the general PID controller, a RL-PID is required to accommodate the
            reinforcement learning

        return output

gimbal = Gimbalcontrol()
Kp = 0.5 # Initial value
Ki = 0.5 # Initial value
Kd = 0.5 # Initial value
pid = PIDcontrol(Kp, Ki, Kd)
real_time_image = ... # Real time image from the Person Detection where the target is
    centerlized (Images in grayscale)
current_pan = ... # Pan angle update from the vision system for each image
current_tilt = ... # Tilt angle update from the vision system for each image
image_stored = ...

# Compare data_stored[-2] and data[-1] and determine the adjustment of the gimbal system
     based on the data differences.
while True:
    image_stored.append(real_time_image)
    if len(image_stored) >= 2: # At least two images for making adjustment
        # Find the difference between data_stored[-2] and data_stored[-1] with threshold
        # Find the desired pan and tilt angles
        desired_pan = ...
        desired_tilt = ...
        #
        error_pan = desired_pan - current_pan
        error_tilt = desired_tilt - current_tilt
        control_pan = pid.PID(error_pan)
        control_tilt = pid.PID(error_tilt)
        gimbal.set_orientation(control_pan, control_tilt)
```