Task: Designing an Autonomous Spatial Tracking and Guidance System

The design of the tracking system will include computer vision-based target detection, coordinate mapping, and the mathematical derivation of real-time movement vectors for a virtual robotic platform.

Objective of the Task

The objective of this task is to help students gain a practical understanding of:

- Computer vision fundamentals and coordinate space manipulation.
- Real-time image processing and feature tracking algorithms.
- Translating visual tracking metrics into actionable digital control logic for robotics.

Instructions for Performing the Task

Select and implement the following real-time visual tracking method:

- **ArUco Marker Detection** (DICT_5X5_50 or DICT_6X6_250)

Develop a Python script to establish a real-time tracking interface using a standard laptop webcam.

The tracking script and user interface must include:

- A fixed target crosshair rendered at the exact center of the video frame.
- A dynamic error vector (a line) drawn from the frame center to the calculated centroid of the tracked object.
- On-screen text commands (MOVE LEFT, MOVE RIGHT, MOVE UP, MOVE DOWN, APPROACH) that update dynamically based on the target's relative coordinate offset.
- A clear spatial threshold (e.g., ±10 pixels) that changes the interface state to show "LOCK ENGAGED" when the target is centered.

Recommended development software and frameworks:

- **IDE:** VS Code, PyCharm, or any standard text editor.
- **Libraries:** OpenCV (opencv-python / opencv-contrib-python), NumPy.

> **Note:** *The use of deep learning object detection frameworks (e.g., YOLO, MediaPipe) is discouraged to keep the focus on core geometric computer vision logic.*

Target Platform / Environment:

- A standard webcam stream (640 × 480 resolution recommended).
- A digital target displayed on a smartphone screen or a simple hand-drawn sheet of paper (no specialized hardware or physical props required).

Requirements for Submission

- A short screen-recorded video (maximum 30–45 seconds) demonstrating the live webcam feed, showing the dynamic arrow vector and text instructions updating smoothly as the target moves.
- A link to a public GitHub repository containing a clean, well-commented execution file (e.g., main.py).
- A concise README.md file detailing dependencies, installation commands, and a brief description of the tracking logic used.

General Guidelines

- The use of open-source software and Python libraries is mandatory.
- The script must handle basic edge cases gracefully, such as not crashing when the target moves entirely out of the camera's field of view.

Technical Guide: Setup Instructions for Tracking Methods

Method 1: ArUco Marker Detection Setup

To track an ArUco pattern configuration, targets can be quickly provisioned digitally:

- Navigate to an online configuration terminal (e.g., chev.me/aruco/).
- Set the Dictionary configuration field precisely to 5x5 (50 markers total).
- Export marker ID 0 or any integer within the range 0-49 as an image asset or screenshot.
- Open the asset on a mobile display with panel illumination maximized (80-100% brightness) to counter webcam overexposure.