2000004855

Emrehan Gökçay

Programming 3 Project Design Document:

Implementation Details:

Programming Language:

The code uses Python programming language and uses these libraries below:

Ultralytics YOLO: The Ultralytics YOLO library is used for real-time object detection. It provides pre-trained YOLO models that can be easily integrated into the application. The YOLOv8 model is instantiated in the ObjectTracker class.

OpenCV: OpenCV is employed for camera input and image processing. The Camera class initializes a video capture object using OpenCV, and frames are processed using OpenCV functions.

Data Description:

Input Data:

Video frames obtained from the camera using OpenCV.

Pre-trained YOLO model weights file (best.pt) used for object tracking.

Output Data:

Detected object bounding boxes and associated information.

Algorithms and Libraries:

Object Tracking Algorithm:

This Python code is designed for real-time object tracking using the YOLO (You Only Look Once) model from Ultralytics. Here's a step-by-step breakdown:

* Camera Class (Camera):

The Camera class is responsible for handling video capture. It initializes a video capture object using OpenCV (cv2.VideoCapture) with a default source index of 0 (the primary camera). It also sets the frames per second (FPS) to 120 using cv2.CAP\_PROP\_FPS. The get\_frame method retrieves a frame from the camera.

* ObjectTracker Class (ObjectTracker):

The ObjectTracker class is designed to perform object tracking using the Ultralytics YOLO model. During initialization, it takes the path to the YOLO model file (best.pt). The track method is used to perform object tracking on an input image (img). It returns the results of the tracking plane objects mi dair only, which include bounding box coordinates, class labels, and confidence scores.

* draw\_boxes Function:

The draw\_boxes function is responsible for extracting bounding box coordinates from the results of object tracking and printing them. It operates on each detected object in the results list and prints the corresponding bounding box coordinates.

* main Function:

The main function serves as the central execution point. It creates instances of the Camera and ObjectTracker classes. Within a continuous loop:

It captures a frame from the camera using the get\_frame method.

Performs object tracking on the frame using the track method of the ObjectTracker.

Calls the draw\_boxes function to print bounding box coordinates for each detected object.

Checks for a key press ('q' key) using cv2.waitKey, and if detected, breaks out of the loop, terminating the program.

* Script Execution:

The script concludes by checking if it's the main module (if \_\_name\_\_ == "\_\_main\_\_":) and, if so, executing the main function.

In summary, the code orchestrates a real-time object tracking system by capturing frames from a camera, utilizing YOLO for object detection, and printing bounding box coordinates. The program can be interrupted by pressing the 'q' key.

Image Processing:

OpenCV is employed for image processing tasks, such as reading frames from the camera, drawing bounding boxes around detected objects, and displaying the processed frames.

Hardware Acceleration:

The code assumes that the hardware supports GPU acceleration since YOLO models can benefit significantly from GPU processing.