API Documentation

CrocoMarine ROV

August 23, 2024

Contents

1 Introduction

This API is the program of the MATE NOAA Competition by the CrocoMarine ROV team. It is designed to track objects in video frames and is open-sourced. and NOAA given permission to use it on additional videos.

2 ObjectTracker API

==========

List all the API endpoints along with their descriptions, request methods, and parameters.

2.1 Overview

- The **ObjectTracker** class is a Python module designed to track objects in video frames.
- It provides two main modes of operation: object detection and tracking.
- The class uses YOLOV8 (You Only Look Once) for object detection and tracking.

2.2 Main Functionality

- **Object Detection Mode (mode=1)**: In this mode, the class uses YOLO to detect objects in each frame and returns the bounding boxes for each.
- **Object Tracking Mode (mode=0)**: In this mode, the class uses the tracking data from the previous frame to track objects in the current frame.
- The tracked objects are then updated with new bounding boxes and track IDs.
- The class also provides options to save the tracking data to an Excel file using ***ExcelHandler***and video output using the ***VideoSaver*** class.

2.3 Class Methods

- **_init_ :**
  Description : Initializes the **ObjectTracker** object.
  Parameters :
  - *model_path (str):* The path to the YOLOv8 model file.
  - *confidence_threshold (float):* The confidence threshold for detection (default: 0.25).
  - *classes (list):* The list of classes to detect (default: [0]).
  - *save_output (bool):* A flag to save output (default: False).
  - *save_data (bool):* A flag to save data (default: False).
  - *origional_size (bool):* A flag to use original size (default: False).
  - *mode (int):* The mode of operation (default: 0).
  - *show_output (bool):* A flag to show output (default: False).
- **run :**
  Description : Runs the object tracking process on the specified input video or image.
  Parameters :
  - *input_path (str):* The path to the input video or image.

  Returns : None

2.4 Example Usage

```
from CrocoMarine_program_2024 import ObjectTracker

# Create an ObjectTracker object
detector = ObjectTracker(
    model_path="CrocoMarine_model_2023.pt",
    confidence_threshold=0.4,
    classes=[0],
    save_output=True,
    save_data=True,
    origional_size=False,
    mode=1, # detection mode
    show_output=False
)

# Run the object tracking process
detector.run("seafloor_footage.mp4")
```

2.5 Command-Line Interface

2.5.1 Overview :

The 'ObjectTracker' class can also be used through a command-line interface using the following command:

This will run the API with the specified model file, input video, confidence threshold, classes to detect, and flags to save output and data.

2.5.2 Example :

```
python main.py --model_path "CrocoMarine_model_2023.pt" -ip "
    seafloor_footage.mp4" --mode 1  --conf 0.4 -show_output --
    save_output --save_data --origional_size
```

2.5.3 Arguments :

The API uses the 'argparse' library to parse command line arguments. The arguments are defined in the 'main.py' file, and they are used to configure the 'ObjectTracker' class.

- **-model_path (required):** The path to the model file.
- **-conf (optional, default=0.25):** The confidence threshold for detection.
- **-classes (optional, default=[0]):** The list of classes to detect.

- **–save_output (optional, default=False):** Flag to save output.
- **–save_data (optional, default=False):** Flag to save data.
- **–origional_size (optional, default=False):** Flag to use original size.
- **–show_output (optional, default=False):** Flag to show the model prediction output.
- **–mode (optional, default=0):** The mode of operation (choices: 0, 1).
- **–input_path (required):** The path to the input video or image.

2.6 Output Formats

The 'ObjectTracker' class generates output in the following formats:

- **Video(.MP4)** : A video file showing the tracked objects.
- **Data(.xlsx)** : A excel file containing information about the tracked objects.

3 ExcelHandler API Documentation

==========

3.1 Overview :

The 'ExcelHandler' class is designed to handle Excel file operations, specifically adding data to an internal data list and saving it to an Excel file. This class provides a simple and efficient way to manage Excel data.

3.2 Main Functionality :

The 'ExcelHandler' class is designed to handle the following main functionality:

- **Data Addition**
  - Description: The 'add_data' method allows you to add rows of data to the internal data list. This data can be in the form of lists, where each list represents a row in the Excel file.
- **Data Saving**
  - Description: The 'save' method saves the internal data to the specified Excel file. The data is saved in a structured format, with each row representing a list of values.

3.3 Class Methods :

The ‘ExcelHandler‘ class offers the following methods:

- **_init_**
  - Description: Initializes the ‘ExcelHandler‘ object with a file path.
  - Parameters:
    - ‘file_path‘ (str): The path to the Excel file.
  - Returns: None
- **add_data**
  - Description: Adds a row of data to the internal data list.
  - Parameters:
    - ‘row‘ (list): A list of values to be added as a row in the Excel file.
  - Returns: None
- **save**
  - Description: Saves the internal data to the Excel file.
  - Parameters: None
  - Returns: None

4 VideoSaver API Documentation

==========

4.1 Overview

The ‘VideoSaver‘ class is designed to save video frames to a file. It provides a simple and efficient way to record video from various sources, such as cameras or video processing pipelines.

4.2 Main Functionality

- The ‘VideoSaver‘ class provides a simple and efficient way to record video from various sources, such as cameras or video processing pipelines.

4.3 Class Methods

- **_init_**
  - Description: Initializes the ‘VideoSaver‘ object with the specified filename, frames per second, and FourCC code.
  - Parameters:
    - ‘filename‘ (str): The filename to save the video to.
    - ‘fps‘ (float): The frames per second to save the video at. Default is 15.0.
    - ‘fourcc‘ (str): The FourCC code to use for the video codec. Default is ’mp4v’.
  - Returns: None
- **start**
  - Description: Starts the video saver, creating a ‘VideoWriter‘ object with the specified width and height.
  - Parameters:
    - ‘width‘ (int): The width of the video frames.
    - ‘height‘ (int): The height of the video frames.
  - Returns: None
- **write**
  - Description: Writes a single frame to the video file. If the video saver has not been started, this method does nothing.
  - Parameters:
    - ‘frame‘ (numpy.ndarray): The frame to write.
  - Returns: None
- **stop**
  - Description: Stops the video saver, releasing any system resources associated with the ‘VideoWriter‘ object.
  - Parameters: None
  - Returns: None