MiniMetroBot

Table of Contents

Description

MiniMetroBot is an object-detection bot designed for the game Mini Metro. Utilizing YOLOv8 for object detection and OpenCV for image processing, the bot is currently capable of identifying the game's main objects. Before initiating detection, it also performs basic automated mouse clicks and movements.

Project Scope and Targets

The primary objective of this project is to detect game objects, including their positions, and to incorporate the ability to automate user inputs. Steps to train an additional model for fully automated gameplay are beyond the current project's scope.

Targets Within Scope

- [x] Object detection within the game
- [x] Basic automated user input
- [x] Compatibility with screen resolutions available on a 2021 MacBook Pro M1 Max (best at 1312x848)

Targets Outside Scope (For Future Projects)

- [ ] Fully automated gaming performed by the final bot

Project Structure Guide

Our project is organized into two primary segments to ensure clarity and efficiency. The first segment focuses on the operational bot that identifies objects in the Mini Metro game, and the second segment provides the necessary resources for training the object detection model.

Bot Operation

- ***Core***: Located within the ***minimetrobot*** package, this folder is the heart of the bot. It contains the primary code responsible for detecting game objects.
- ***Models***: This folder also resides within the ***minimetrobot*** package and holds the pre-trained YOLOv8 model used for object detection.
- `__main__.py`: This is the main Python file that serves as the launching point for the bot. It calls various functions and scripts to perform the bot's operations.

Model Training

- ***yolo_tuning***: This folder is a complete environment for model training. It is also situated in the ***minimetrobot*** package and is central to the model's development and fine-tuning.
- ***yolo_model_tuning.py***: This script is designed for customizing and training the YOLOv8 model. It offers a structured way to perform model tuning.
- **Subfolders**: These contain the essential training data, including raw images, annotated images, and other relevant files.

Supporting Evidence

- ***evidences***: This folder provides tangible proof of the bot's functionality through screen recordings. This is particularly useful for verifying the system's operation on different platforms.

By understanding the layout and purpose of these folders, one can more easily navigate the project, modify existing functionalities, or even add new features.

Project Structure

```
MiniMetroBot/                       # Root directory
│
├── minimetrobot/                   # Source code
│   ├── core/                       # Core modules with the main bot functionality
│   │   ├── __init__.py
│   │   └── ScreenDetector.py       # Class for detecting objects (docstring available)
│   ├── evidences/                  # Folder contains evidences
│   │   ├── mock_activity_start.mov # Screenvideo activity mock
│   │   └── detection_evidence.mov  # Screenvideo from bot image detection
│   ├── models/                     # Model files
│   │   ├── __init__.py
│   │   └── best.pt                 # Best model from training exercise
│   ├── yolo_tuning/                # Main folder for model training (not in whl file)
│   │   ├── images_annotated/       # Annotated training data for YOLOv8 training
│   │   ├── images_pre_selected/    # Updated version of the raw data images
│   │   ├── images_raw/             # Raw images
│   │   ├── model_source/
│   │   │   └── yolov8n.pt          # Base YOLOv8 model used for tuning
│   │   ├── model_tuned
│   │   └── yolo_model_tuning.py    # Script for model tuning (docstring available)
│   ├── __init__.py
│   └── __main__.py                 # Main program (docstring available)
│
├── docs/                           # Documentation
│   ├── README.md                   # Project description as markdown
│   └── README.pdf                  # Markdown converted to pdf
│
├── LICENSE                         # License information
└── pyproject.toml                  # Poetry configuration file
```

Dependencies

The project has the following dependencies:

- `python`: 3.10.9
- `comet-ml`: 3.33.6
- `ipython`: 8.14.0
- `numpy`: 1.24.2
- `opencv-contrib-python`: 4.7.0.72
- `opencv-python`: 4.7.0.72
- `pillow`: 10.0.0
- `pyyaml`: 6.0.1
- `screeninfo`: 0.8.1
- `torch`: 2.0.0

- `torchvision`: 0.15.1
- `ultralytics`: 8.0.162
- `pyautogui`: 0.9.54

To install the system and its dependencies follow the installation instructions below.

Installation of the Final Bot

Option 1: Using Git and Poetry

1. **Clone the repository:**

   ```
   git clone https://github.com/TomGFFM/PraxisProjektSemester4_30313.git
   ```

2. **Navigate to the project directory:**

   ```
   cd PraxisProjektSemester4_30313/MiniMetroBot
   ```

3. **Install dependencies using Poetry:**

   ```
   poetry install
   ```

Option 2: Using Wheel File

1. **Download the latest wheel file, `minimetrobot-0.1.6-py3-none-any.whl`, from the repository.**
2. **Install the package using `pip`:**

   ```
   pip install path/to/minimetrobot-0.1.6-py3-none-any.whl
   ```

Usage

Usage Instructions

To utilize the MiniMetroBot, a few prerequisites must be met as outlined in the above setup guide. Additionally, you'll need to have the game Mini Metro installed on your system.

For optimal object detection, it's recommended to run the game on your main display at a resolution of 1312x848. For the best results, the game should also be in fullscreen mode.

To execute the bot, open a terminal window on a secondary screen. Once launched, the bot will display a new window on this secondary screen, showing a real-time feed of the game with object detection overlays, including bounding boxes and class labels.

By adhering to these instructions, you can experience the most effective performance of the MiniMetroBot.

Option 1: Using the Poetry Entry Point

(notice different naming convention here. Just: "metrobot")
If you have Poetry installed, you can run MiniMetroBot directly from the environment where the package is installed using the following command:

```
poetry run metrobot
```

Option 2: Running the Python Script Manually

Activate the virtual environment and execute the following command:

```
python -m minimetrobot
```

Either of these options will start the MiniMetroBot.

YOLOv8 Model Training Information

Model Overview

The `yolo_tuning` directory houses a sub-directory named `model_tuned`, which features multiple folders containing trained YOLOv8 object detection models. The final model currently integrated into the bot resides in the folder named "round_optimized_with_Adamax_True_bs_16_20230826_1342."

Evaluation Metrics

You can assess the pertinent training metrics through the public CometML report designated for this project. To explore the detailed report, click the following link: MiniMetroBot CometML Metrics Report.

How to Train the Model

The YOLOv8 model can be trained using the `yolo_model_tuning.py` script. The foundational model for training is stored in the `model_source` folder and serves as the baseline YOLOv8 model for training. For additional usage guidelines and information, consult the included docstrings.

Contributing

Contributions, issues, and feature requests are welcome! Feel free to check issues page.

License

This project is BSD licensed.

Show your support

Give a ⭐ if this project helped you!