This document provides a comprehensive guide on setting up and using the Marel Tracker system for tracking and identifying objects in a conveyor belt system.

Table of contents

Package Dependencies

To ensure the successful execution of the Marel Tracker, it is essential to install the following required packages using the provided commands:

```
pip3 install opencv-python
pip3 install numpy
```

These packages, namely OpenCV-Python and NumPy, provide crucial functionalities that the project relies on. They can be installed in either the terminal for the local environment of the computer or in a virtual environment.

Projects

In this package, there are two main projects:

1. SortTracker
2. SortTrackerSimulator

Keep in mind that the SortTrackerSimulator is only accessable for Marel employees and SortTracker only runs on computers that are inside Marel's supervision and are on the Marel network.

Sort Tracker

The Tracker is located under the **SortTracker** project. It houses four different python scripts. The main python script is `tracker.py`. When running the Tracker script, there are arguments that can be included to modify the workings of the tracker.

Argument	Usage

Argument	Usage
-e	Activates a debugger mode that saves raw images and images from the modeling in a folder for debugging purposes (bool)
-f	Changes the frame rate of the video stream and the saved video (int)
-t	Affects the duration of which the Data stream is listened to. If not used, the dataset will be run until shut down (int)

So the code can be run like such:

```
python3 tracker.py [options][arguments]
```

For example (duration is 10 seconds, frame rate is 7 fps and debug mode is on)

```
python3 tracker.py -t 10 -e -f 7
```

After the code has been run, it actively searches for a data stream, information about which can be found in Usage

Sort Tracker Simulator

The **SortTrackerSimulator** is a Unity project that, when run, creates a WebSocket where the images can be retrieved on `localhost:8080`. The tracker code collects this data.

In the **SortTrackerSimulator**, there are three example projects that will help get you started on creating your own scenario

- Singular Linear conveyor
- Two conveyors in an L-shaped setup
- Three conveyors with a curved transfer conveyor

Usage

There are three ways to use the Marel Tracker:

1. Data stream via Json
2. Data stream via Unity
3. Data stream via Camera

Data stream via Json

To run the Marel Tracker using a Json data stream, you have to use the following flags:

Argument	Usage
-d	dataset path (str)

Argument	Usage
-i	image folder path (str)

The previously mentioned arguments are still usable in this case (-f is not usable)

Data stream via Unity

To set up the Marel Tracker using a data stream via Unity, follow these steps:

1. Open the **SortTrackerSimulator**
2. Select the desired conveyor setup from the available examples.
3. Run the Unity project. This will create a WebSocket where the images can be retrieved on `localhost:8080`.
4. Run the `tracker.py` script with the desired arguments.

During operation, the Marel Tracker will process the images received from the Unity project and display tracking information on the screen. All captured images will be stored in a directory named after the time and date the script was run.

Data Stream via Camera

To set up the Marel Tracker using a data stream via Camera, follow these steps:

1. Make sure you have a camera that can stream onto `localhost:8080`. You can use a webcam or a specialized camera compatible with the Marel Tracker system.
2. Position the camera in a way that it has a clear view of the area you want to track.
3. Ensure that your camera is properly set up and streaming to `localhost:8080`.
4. Run the `tracker.py` script with the desired arguments. When prompted, input the amount of time to run in seconds (type in -1 for an endless data stream).

During operation, the Marel Tracker will process the images received from the camera and display tracking information on the screen. All captured images will be stored in a directory named after the time and date the script was run.

In both cases (Unity and Camera setups), ensure you have the necessary dependencies installed and that your hardware is compatible with the Marel Tracker system for optimal performance.

Troubleshooting

If you encounter any issues while using the Marel Tracker, consider the following troubleshooting steps:

1. Verify that the required package dependencies (OpenCV-Python and NumPy) are correctly installed.
2. Double-check the camera or Unity project setup to ensure the data stream is accessible at `localhost:8080`.
3. Ensure you are on Marel network to ensure the image recognition is accessable
4. Make sure the flags and arguments used with the `tracker.py` script are correct and compatible with your setup.
5. If using a camera, ensure that it is compatible with the Marel Tracker system and properly positioned to capture the area of interest.

If issues persist after attempting these troubleshooting steps, consult the project documentation and seek assistance from the Marel Tracker community or support team.