# Instructions for Running the SVO Frame Extraction and Object Detection on Windows/Ubuntu

The following instructions will guide you through setting up and running the frame extraction and YOLO-based object detection script on both Windows and Ubuntu. The script involves:

1. Extracting frames from an SVO file using the ZED SDK.

2. Running object detection using a YOLO model.

3. Saving the output as a video and CSV file.

## Prerequisites

Before running the script, ensure you have the following software and libraries installed on your system:

- Python (>=3.6)

- ZED SDK (For SVO file handling)

- YOLOv8 (Depending on your model choice)

- OpenCV (for image and video processing)

- Ultralytics YOLO library (ultralytics)

- PIL (Pillow) (for image manipulation)

- NumPy and SciPy (for mathematical operations)

- PyTorch (for YOLO model inference)

- pyzed (for ZED SDK integration)

## Step-by-Step Setup and Execution Guide

### 1. Install Python and Create a Virtual Environment

Download and install Python (version 3.6 or later).

Open your command prompt (Windows) or terminal (Ubuntu) and create a virtual environment:

python -m venv yolo\_env

Activate the virtual environment:

Windows: yolo\_env\Scripts\activate

Ubuntu: source yolo\_env/bin/activate

### 2. Install Required Libraries

Install the necessary Python libraries using the following command:

pip install ultralytics opencv-python pillow torch numpy scipy

### 3. Install ZED SDK and pyzed Library

The ZED SDK is required to handle SVO file extraction. Follow these steps to install it:

1. Install ZED SDK:

- Download the ZED SDK installer from the ZED SDK Downloads page.

- Follow the instructions to install it on your system.

2. Install the pyzed Python API:

- After installing the ZED SDK, use pip to install the pyzed package: pip install pyzed

### 4. Prepare the Script and Model

Create a new Python file (e.g., svo\_yolo\_processing.py) and copy the complete script provided above into the file.

Place your YOLO model file (Fine\_tune.pt) in the same directory as your script or provide the correct path in the MODEL\_PATH variable in the script.

### 5. Prepare Your SVO File

Place your SVO file (e.g., example.svo) in the same directory as your script or provide the correct path in the SVO\_FILE\_PATH variable in the script.

### 6. Run the Script

To run the script, use the following command:

python svo\_yolo\_processing.py

The script will:

1. Extract frames from the SVO file and save them in the extracted\_frames directory.

2. Perform object detection on the extracted frames using the YOLO model.

3. Save the output video and CSV file in the specified output directories.

## Troubleshooting and Additional Tips

1. ZED SDK and Python Integration Issues:

- Ensure that the ZED SDK is properly installed and the pyzed library is configured correctly. If you encounter issues, try reinstalling the SDK and the pyzed library.

2. YOLO Model Compatibility:

- Ensure that your YOLO model file (Fine\_tune.pt) is compatible with the ultralytics library and PyTorch version you have installed. If you encounter compatibility issues, consider downloading a compatible version of the YOLO model.

3. Setting Up Environment Variables for ZED SDK (Ubuntu only):

- After installing the ZED SDK on Ubuntu, you may need to set up environment variables:

export ZED\_SDK\_ROOT='/usr/local/zed'

export PYTHONPATH=$PYTHONPATH:$ZED\_SDK\_ROOT/pyzed/

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$ZED\_SDK\_ROOT/lib/

4. Check for Frame Extraction:

- After running the script, check the extracted\_frames directory to verify if frames were successfully extracted from the SVO file.

5. Running in GPU Mode:

- Ensure that your system has a compatible NVIDIA GPU and CUDA drivers installed to leverage GPU acceleration for object detection.

## File and Directory Structure

Ensure your project directory is structured as follows:

project\_directory/

|

├── Fine\_tune.pt # Your YOLO model file

├── example.svo # Your SVO file

├── svo\_yolo\_processing.py # The main script

├── extracted\_frames/ # Directory for extracted frames (auto-created by the script)

├── output/

│ ├── VRI\_detection\_output.mp4 # Final output video

│ └── VRI\_detection\_data.csv # Final output CSV with detection data

## Sample Command to Run the Script

python svo\_yolo\_processing.py

## Example Output

Output Video: VRI\_detection\_output.mp4 – A video showing the frames with detected objects annotated.

Output CSV: VRI\_detection\_data.csv – A CSV file containing detection data for each frame, including class detections, number of VRIs, and distance between VRIs.