🚀 YOLO People Counter

A **real-time people counting system** powered by **Ultralytics YOLO v11** that accurately tracks people entering and exiting defined regions with advanced movement direction analysis and comprehensive logging capabilities.

✨ Features

🎯 Core Functionality

- **Real-time People Detection**: Leverages YOLO v11 for accurate person detection
- **Multi-Object Tracking**: Persistent tracking across video frames
- **Bi-directional Counting**: Separate counts for entering and exiting
- **Region-based Analysis**: Customizable polygonal counting zones
- **Movement Direction Detection**: Intelligent algorithm determines entry/exit intent

🚀 Performance Optimizations

- **Frame Skipping**: Process every nth frame for faster performance
- **GPU Acceleration**: CUDA support for high-speed processing
- **Smart Caching**: Reuses detection results on skipped frames
- **Memory Management**: Optimized track history with automatic cleanup

📊 Visualization & Logging

- **Live Counter Display**: Real-time statistics overlay
- **CSV Data Export**: Timestamped counting logs with FPS metrics
- **Track Visualization**: Visual tracking trails for each person
- **Interactive Regions**: Drag-and-drop region adjustment
- **Clean Output Mode**: Optional region line hiding

🎛️ Advanced Configuration

- **Multiple Detection Classes**: Filter specific object types
- **Customizable Thresholds**: Adjustable sensitivity parameters
- **Visual Customization**: Configurable colors, thickness, and styling

- **Output Control**: Flexible save options and directory management

🛠️ Installation

Prerequisites

- **Python 3.8+**
- **CUDA-compatible GPU** (recommended for optimal performance)
- **Video files** for input

Quick Install

```
# Clone the repository
git clone https://github.com/SahiL911999/People-count-entering-exiting-yolo.git
cd People-count-entering-exiting-yolo

python -m venv myenv
cd myenv/Scripts
activate
cd ../..

# Install dependencies
pip install -r requirements.txt

# Download YOLO weights (automatic on first run)
# The script will automatically download yolo11l.pt if not present.
```

🚀 Quick Start

Basic Usage

```
cd Code
# Run with video file
python Main.py --source path/to/your/video.mp4 --view-img

# Save output video
python Main.py --source video.mp4 --save-img
```

Advanced Examples

```
# High-speed processing with frame skipping
python Main.py --source video.mp4 --skip-frames 10 --view-img

# CPU-only mode
python Main.py --source video.mp4 --device cpu --view-img

# Custom model and clean display
python Main.py --source video.mp4 --weights yolo11x.pt --no-regions-lines

# Filter only people (class 0 - Persons only)
python Main.py --source video.mp4 --classes 0 --view-img
```

⚙️ Configuration Options

Parameter	Type	Default	Description
`--source`	str	**Required**	Video file path or camera index (0 for webcam)
`--weights`	str	`yolo11l.pt`	YOLO model weights file
`--device`	str	`cuda`	Processing device:`cuda`, `cpu`, or specific GPU ID
`--view-img`	flag	`False`	Display real-time video processing
`--save-img`	flag	`True`	Save processed video to output directory
`--skip-frames`	int	`2`	Process every nth frame (higher = faster, lower accuracy)
`--classes`	list	`None`	Filter specific object classes (e.g.,`--classes 0` for people only)
`--line-thickness`	int	`2`	Bounding box line thickness
`--track-thickness`	int	`2`	Object tracking trail thickness
`--region-thickness`	int	`4`	Counting region boundary thickness

Parameter	Type	Default	Description
`--no-regions-lines`	flag	`False`	Hide region boundaries for clean output
`--exist-ok`	flag	`False`	Overwrite existing output directory

📋 Output Structure

```
ultralytics_rc_output/
├── exp/
│   ├── your_video.avi          # Processed video output
│   └── 2025-09-08_04-30-15_your_video.csv  # Timestamped counting data
```

CSV Data Format

```
time,entering,exiting,total,fps
04:30:15,5,2,7,28.3
04:30:25,8,3,11,29.1
04:30:35,12,5,17,28.7
```

🎮 Interactive Features

Region Adjustment

- **Click and Drag**: Move counting regions by clicking inside them
- **Real-time Preview**: See region changes immediately
- **Persistent Tracking**: Counts continue during region adjustments

Keyboard Controls

- `Q`: Quit application
- **Mouse**: Interactive region manipulation

🎯 Customizing Counting Regions

Edit the `counting_regions` list in the code to define your custom areas:

```
counting_regions = [
    {
        "name": "Store Entrance",
        "polygon": Polygon([(x1, y1), (x2, y2), (x3, y3), (x4, y4)]),
        "enter_count": 0,
        "exit_count": 0,
        "region_color": (255, 42, 4),    # Blue-Green-Red
        "text_color": (255, 255, 255),   # White text
    }
]
```

Finding Coordinates

1. Run with `--view-img` flag
2. Note pixel coordinates from your video
3. Create polygon points in clockwise order
4. Test and adjust as needed

📊 Performance Tips

For Maximum Speed

```
python Main.py --source video.mp4 --skip-frames 10 --device cuda --no-regions-lines
```

For Maximum Accuracy

```
python Main.py --source video.mp4 --skip-frames 1 --weights yolo11x.pt --classes 0
```

For Real-time Processing

```
python Main.py --source 0 --skip-frames 3 --view-img
```

🔧 Troubleshooting

Common Issues

GPU Not Detected

```
# Check CUDA installation
python -c "import torch; print(torch.cuda.is_available())"

# Force CPU mode
python Main.py --source video.mp4 --device cpu
```

Model Download Failed

```
# Manually download YOLO weights
wget https://github.com/ultralytics/assets/releases/download/v0.0.0/yolo11l.pt
```

Memory Issues

```
# Reduce model size or increase frame skipping
python Main.py --source video.mp4 --weights yolo11s.pt --skip-frames 5
```

Inaccurate Counting

- Adjust polygon region to better match entry/exit paths
- Lower `skip_frames` value for better tracking
- Ensure good lighting and camera angle
- Use higher resolution model (`yolo11x.pt`)

📈 Performance Benchmarks

Model	Device	FPS (1080p)	Accuracy	Memory Usage
YOLO11s	RTX 3080	~45 FPS	Good	2GB
YOLO11l	RTX 3080	~30 FPS	Excellent	4GB
YOLO11x	RTX 3080	~20 FPS	Best	6GB
YOLO11l	CPU	~8 FPS	Excellent	2GB

🤝 Contributing

We welcome contributions! Please follow these steps:

1. **Fork** the repository
2. **Create** a feature branch (`git checkout -b feature/amazing-feature`)
3. **Commit** your changes (`git commit -m 'Add amazing feature'`)
4. **Push** to the branch (`git push origin feature/amazing-feature`)
5. **Open** a Pull Request

📄 License

This project is licensed under the **AGPL-3.0 License** - see the Ultralytics License for details.

- ✅ **Free for research and non-commercial use**
- ⚠️ **Commercial use requires Enterprise License**
- 📋 **Open-source derivative works must be published under AGPL-3.0**

🙏 Acknowledgments

- **Ultralytics** for the amazing YOLO framework
- **OpenCV** community for computer vision tools
- **Shapely** for geometric operations
- **Contributors** who help improve this project

📞 Support

- 📧 **Issues**: Use GitHub Issues for bug reports
- 💬 **Discussions**: Join GitHub Discussions for questions
- 📖 **Documentation**: Check this README and code comments
- 🌐 **Ultralytics Docs**: Official YOLO Documentation

Made with ❤️ using Ultralytics YOLO, Demonstration by Sahil Ranmbail - AI Engineer

⭐ Star this repo • 🐛 Report Bug • 💡 Request Feature