# Drone Tracking Web Application

A full-stack real-time drone detection system capable of tracking multiple drone targets with 99% accuracy, utilizing YOLO, DeepSORT, React.js, and a FastAPI backend.

## Repo link: <https://github.com/Nitish-Biswas/drone_live_detection_and_tracking>

**Features**

* **Real-time Video Feed**: Live camera stream with drone detection overlay
* **YOLO + DeepSORT Integration**: Advanced object detection and tracking
* **Real-time Notifications**: WebSocket-based instant alerts for new detections
* **Interactive Dashboard**: Modern React UI with Material-UI components
* **Detection Database**: SQLite storage for all detection records
* **Interactive Map**: Leaflet map showing drone detection locations
* **Statistics Tracking**: Daily detection counts and analytics

## System Requirements

* Python 3.8+
* Node.js 16+
* Camera/Webcam
* YOLO model file (best.pt)

## ⚡ Important Notes

#### CUDA GPU Setup:

If you want to use CUDA GPU for acceleration, please run:

```bash

# Activate your backend virtual environment first:

# On Windows:

venv\Scripts\activate

# On macOS/Linux:

source venv/bin/ activate

# Then run:

pip uninstall -y torch torchvision torchaudio

pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/ whl/cu128

```

Or, get the appropriate version for your system from the official [PyTorch website](https://pytorch.org/).

#### Changing Camera Feed Source:

* To change the camera/video feed source, edit **line 304** in backend/ tracker.py and change the parameter in cv2.VideoCapture(#source number) accordingly.

## Download Pre-trained Model & Dataset

You can use the developer's custom-trained YOLOv8l model and the dataset:

#### Custom YOLOv8l Model:

[Download yolov8l\_ep10](https://www.kaggle.com/models/nitishbiswas1/yolov8l_ep10)

#### Training Dataset:

[Drone Detection YOLO Dataset](https://www.kaggle.com/datasets/nitishbiswas1/drone-detection-yolo-dataset)

## Quick Start

### Clone and Setup Backend

```bash

# Navigate to backend directory

cd backend

# Create virtual environment

python -m venv venv

# Activate virtual environment On Windows:

venv\Scripts\activate

# On macOS/Linux:

source venv/bin/activate

# Install dependencies

pip install -r requirements.txt ```

### Add Your YOLO Model

Place your trained YOLO model file (best.pt) in the backend directory, or update the MODEL\_PATH in main.py:

python MODEL\_PATH = "path/to/your/model.pt"

### Start Backend Server

```bash

# Start FastAPI server

uvicorn main:app --reload --host 0.0.0.0 --port 8000 ``` The backend will be available at: [http://localhost:8000](http://localhost:8000/)

### Setup Frontend

```bash

# Navigate to frontend directory

cd ../frontend

# Install dependencies

npm install

# Start development server

npm run dev ```

The frontend will be available at: [http://localhost:3000](http://localhost:3000/)

## Usage

* 1. **Start the Application**: Open [http://localhost:3000](http://localhost:3000/) in your browser
  2. **Start Camera**: Click the "Start Camera" button to begin detection
  3. **View Live Feed**: Watch the real-time video with detection overlays
  4. **Monitor Detections**: See new drone alerts and view detection statistics
  5. **Check Map**: View detection locations on the interactive map
  6. **Review Data**: Browse today's detections in the data table

## API Endpoints

### Camera Control

* + - POST /camera/start - Start camera tracking
    - POST /camera/stop - Stop camera tracking
    - GET /camera/status - Get camera status

### Detections

* + - GET /detections/today - Get today's detections
    - GET /detections/ - Get all detections (with pagination)
    - GET /detections/date/{date} - Get detections for specific date
    - DELETE /detections/{id} - Delete detection

### Real-time

* + - GET /video - Video stream endpoint
    - WebSocket /ws - Real-time updates

### System

* + - GET /health - Health check
    - GET / - API documentation

## Project Structure

drone-tracking/

├── backend/

│ ├── main.py # FastAPI application

│ ├── tracker.py # DroneTracker class

│ ├── models.py # Database models

│ ├── database.py # Database configuration

│ ├── requirements.txt # Python dependencies

│ └── static/

│ └── index.html # Backend test page

└── frontend/

├── src/

│ ├── components/ # React components

│ ├── hooks/ # Custom hooks

│ ├── services/ # API services

│ ├── utils/ # Utilities

│ ├── App.jsx # Main app component

│ └── main.jsx # React entry point

├── package.json # Node dependencies

└── vite.config.js # Vite configurationconfiguration

## Configuration

### Backend Configuration

Edit backend/main.py to configure: - Model path: MODEL\_PATH = "your- model.pt" - Confidence threshold: confidence\_threshold=0.5 - Database URL: Set DATABASE\_URL environment variable

### Frontend Configuration

Edit frontend/src/utils/constants.js to configure: - API base URL - WebSocket URL - Map settings - Notification settings

## Troubleshooting

### Common Issues

**Camera not working:** - Check camera permissions - Verify camera is not in use by another application - Try different camera index in tracker.py

**Model not found:** - Ensure best.pt file exists in backend directory - Check file permissions - Verify model format is compatible

**Connection issues:** - Check if backend is running on port 8000 - Verify frontend proxy configuration in vite.config.js - Check firewall settings

**WebSocket connection failed:** - Ensure both frontend and backend are running - Check browser console for connection errors - Verify WebSocket URL in constants

### Performance Tips

1. **Reduce video resolution** in tracker.py for better performance
2. **Adjust confidence threshold** to reduce false positives
3. **Limit frame rate** for lower CPU usage
4. **Use GPU acceleration** if available with CUDA

## Development

### Adding New Features

1. **Backend**: Add new endpoints in main.py
2. **Frontend**: Create new components in src/components/
3. **Database**: Update models in models.py
4. **Real-time**: Extend WebSocket handlers

### Testing

```bash

# Backend tests

cd backend python -m pytest

# Frontend tests

cd frontend npm test ```

### Building for Production

```bash

# Build frontend

cd frontend npm run build

# Deploy backend

cd backend pip install gunicorn gunicorn main:app --workers 4 --worker- class uvicorn.workers.UvicornWorker ```

## License

This project is licensed under the MIT License.

## Contributing

1. Fork the repository
2. Create feature branch
3. Commit changes
4. Push to branch
5. Create Pull Request

## Support

For issues and questions: - Check the troubleshooting section - Review API documentation at <http://localhost:8000/docs> - Create an issue on GitHub

#### You can also contact the developer:

* + **Name:** Nitish Biswas
  + **Email:** [nitishbiswas066@gmail.com](mailto:nitishbiswas066@gmail.com)
  + **Number:** 8979053318

**Note**: Make sure to replace best.pt with your actual YOLO model file trained for drone detection.