

📄 Football Analytics with OpenCV and Python

Overview

A comprehensive football analytics system built from scratch using **OpenCV**, **YOLOv8**, and **Python** for processing complete match videos. This system can detect and analyze key football events including passes, goals, saves, and ball juggling while identifying players through jersey number recognition from uploaded video files.

📄 Features

- 📄 **Object Detection:** Advanced player, ball, and referee detection using YOLOv8
- 📄 **Player Identification:** Jersey number recognition using OCR technology
- 📄 **Event Detection:** Automatic detection of passes, goals, saves, and juggling
- 📄 **Video File Processing:** Designed for 30-60 minute match video analysis
- 📄 **Comprehensive Reports:** Detailed JSON and CSV analysis reports
- 📄 **Event Timeline:** Minute-by-minute event breakdown
- 📄 **Match Statistics:** Complete player and team performance metrics

📄 Table of Contents

- [Installation](#)
- [Quick Start](#)
- [Camera Setup](#)
- [Usage](#)
- [Project Structure](#)
- [Implementation Timeline](#)
- [Configuration](#)
- [Troubleshooting](#)
- [Contributing](#)

📄 Installation

Prerequisites

- **Python 3.8+**
- **4GB+ RAM** (8GB+ recommended)
- **GPU with CUDA support** (optional but recommended)

- **GoPro cameras** or similar for video capture

Step 1: Clone/Download Project

```
# Create project directory
mkdir football_analytics_project
cd football_analytics_project
```

Step 2: Set up Virtual Environment

```
# Create virtual environment
python -m venv football_env

# Activate environment
# Windows:
football_env\Scripts\activate
# Mac/Linux:
source football_env/bin/activate
```

Step 3: Install Dependencies

```
# Upgrade pip
python -m pip install --upgrade pip

# Install required packages
pip install opencv-python>=4.8.0
pip install ultralytics>=8.0.0
pip install numpy>=1.21.0
pip install pandas>=1.3.0
pip install matplotlib>=3.4.0
pip install pytesseract>=0.3.8
pip install scikit-learn>=1.0.0
pip install scipy>=1.7.0
pip install pillow>=8.3.0
pip install tqdm>=4.62.0
pip install seaborn>=0.11.0
```

Step 4: Install Tesseract OCR

Windows: Download from [GitHub releases](#)

Mac: brew install tesseract

Linux: sudo apt-get install tesseract-ocr

Step 5: Verify Installation

```
python -c "import cv2, ultralytics, numpy; print('✔ Installation successful!')"
```

📌 Quick Start

1. Set up Camera Positions

```
# Generate optimal camera positioning
python camera_positioning.py
```

This creates a visual diagram showing where to place your cameras for optimal coverage.

2. Analyze Your Match Video

```
from football_video_analyzer import FootballVideoAnalyzer

# Initialize analyzer
analyzer = FootballVideoAnalyzer()

# Analyze your 30-60 minute match video
results = analyzer.analyze_full_video(
    video_path='your_match.mp4',
    output_dir='analysis_results'
)

# Results are automatically saved as JSON and CSV files
```

3. Command Line Usage

```
python football_video_analyzer.py
# (Update the video_file path in the script to your video)
```

📌 Camera Setup

Recommended Setup for Video Recording

Camera Position	Location	Height	Distance	Purpose
Main Tactical	Center line sideline	10-12 ft	20-25 ft	Overall game analysis
Goal Camera A	Behind goal A	8-10 ft	10-15 ft	Goal detection
Goal Camera B	Behind goal B	8-10 ft	10-15 ft	Goal detection
Overview	Corner elevated	12-15 ft	15-20 ft	Formation analysis

GoPro Settings for Best Results

- **Resolution:** 1080p @ 60fps (recommended for processing speed)
- **FOV:** Wide or Linear (Linear preferred for less distortion)
- **Stabilization:** HyperSmooth ON
- **Protune:** ON for better post-processing
- **Recording:** Continuous recording for full match

▢ Usage

Basic Video Analysis

```
from football_video_analyzer import FootballVideoAnalyzer

# Create analyzer instance
analyzer = FootballVideoAnalyzer()

# Analyze complete match video
report = analyzer.analyze_full_video(
    video_path="match_2024_01_15.mp4",
    output_dir="match_analysis"
)

# The system will generate:
# 1. Detailed JSON report with all events and statistics
# 2. CSV summary of event counts
# 3. Timeline CSV with minute-by-minute events
# 4. Console summary of key statistics
```

Analysis Results

After processing, you'll get:

```
▢ Analysis Results Saved:
▢ Detailed Report: match_2024_01_15_analysis_20240815_143022.json
▢ Summary CSV: match_2024_01_15_summary_20240815_143022.csv
▢ Timeline CSV: match_2024_01_15_timeline_20240815_143022.csv

▢ FOOTBALL MATCH ANALYSIS COMPLETE
=====
▢ Video Duration: 45.2 minutes
▢ Players Detected: 18
▢ Total Events: 156
▢ Events/Minute: 3.45

▢ EVENT BREAKDOWN:
▢ Passes: 142
▢ Goals: 3
```

▮ Saves: 8
▮ Juggling: 3

Batch Processing Multiple Videos

```
import os
from football_video_analyzer import analyze_match_video

# Process all videos in a folder
video_folder = "match_videos"
for video_file in os.listdir(video_folder):
    if video_file.endswith(('.mp4', '.avi', '.mov')):
        video_path = os.path.join(video_folder, video_file)
        print(f"Processing: {video_file}")
        analyze_match_video(video_path, f"analysis_{video_file[:-4]}")
```

▮ Project Structure

```
football_analytics_project/
├── ▮ analysis_results/           # Generated analysis reports
├── ▮ match_videos/              # Your input video files
├── ▮ config/
│   ├── settings.py             # Configuration settings
│   └── model_config.json       # Model parameters
├── ▮ data/
│   ├── ▮ models/               # YOLO and custom models
│   └── ▮ datasets/              # Training data (optional)
├── ▮ src/                       # Source code modules
├── ▮ notebooks/                 # Jupyter notebooks for analysis
├── football_video_analyzer.py  # Main video analyzer
├── camera_positioning.py       # Camera setup tool
├── requirements.txt            # Dependencies
├── test_setup.py               # Setup verification
└── README.md                   # This file
```

▮ Implementation Timeline

Week	Phase	Tasks	Deliverable
1	Setup	Environment, cameras, test footage	Working setup
2	Detection	Player/ball detection, OCR	Basic detection
3	Tracking	Multi-object tracking	Player tracking
4	Events	Pass, goal, save, juggling detection	Event detection
5	Integration	Video processing pipeline	Complete system

Week	Phase	Tasks	Deliverable
6-7	Optimization	Performance tuning, testing	Optimized system
8	Production	Reports, documentation, deployment	Production ready

Total Duration: 8-12 weeks
Effort: ~20-30 hours per week

⚙️ **Configuration**

Model Configuration

```
# In football_video_analyzer.py, you can adjust:
confidence_threshold = 0.5      # Detection confidence
track_history_length = 30       # Frames to keep in tracking history
```

Video Processing Settings

```
# Skip frames for faster processing (optional)
# In analyze_full_video method, uncomment:
# cap.set(cv2.CAP_PROP_POS_FRAMES, frame_count + 2) # Skip 2 frames
```

Event Detection Tuning

```
# Adjust detection parameters in respective methods:
movement_distance > 50      # Pass detection threshold
distance < 100              # Player proximity threshold
direction_changes >= 2      # Juggling detection sensitivity
```

📊 **Performance Benchmarks**

Component	Target Accuracy	Processing Speed	Memory Usage
Player Detection	90%+	15-30 FPS	4GB RAM
Ball Detection	85%+	15-30 FPS	2GB RAM
Pass Detection	75%+	Real-time	1GB RAM
Goal Detection	95%+	Real-time	512MB RAM
Jersey Recognition	70%+	10 FPS	2GB RAM

Expected Processing Times

- **30-minute video:** 10-20 minutes processing time
- **60-minute video:** 20-40 minutes processing time
- **Processing speed:** 1.5-3x video duration depending on hardware

▮ Troubleshooting

Common Issues

Video Loading Problems

```
# Convert video format if needed
ffmpeg -i input.mov output.mp4

# Check supported formats
python -c "import cv2; print(cv2.getBuildInformation())"
```

Low Detection Accuracy

```
# Try larger YOLO model
analyzer = FootballVideoAnalyzer(model_path="yolov8m.pt") # or yolov8l.pt

# Adjust confidence threshold
analyzer.confidence_threshold = 0.3 # Lower = more detections
```

Slow Processing

```
# Process every 3rd frame for 3x speed improvement
# Uncomment frame skipping in analyze_full_video method
```

Memory Issues

- Process shorter video segments
- Use smaller YOLO model ([yolov8n.pt](#))
- Close other applications during processing

Performance Optimization

For Faster Processing:

- Use GPU acceleration if available
- Process at lower resolution
- Skip frames (process every 2nd or 3rd frame)
- Use [yolov8n.pt](#) model for speed

For Better Accuracy:

- Use [yolov8l.pt](#) or [yolov8x.pt](#) models
- Process all frames
- Higher video resolution
- Manual calibration of goal areas

▮ Contributing

1. Fork the repository
2. Create feature branch (`git checkout -b feature/new-feature`)
3. Commit changes (`git commit -am 'Add new feature'`)
4. Push branch (`git push origin feature/new-feature`)
5. Create Pull Request

Development Setup

```
# Install development dependencies
pip install -r requirements-dev.txt





# Run tests
python -m pytest tests/

# Code formatting
black src/
```

▮ Future Enhancements

- **Multi-camera synchronization** for 3D analysis
- **Advanced player tracking** with pose estimation
- **Tactical analysis** with formation detection
- **Heat maps and pass networks** visualization
- **Real-time processing** optimization
- **Mobile app** for quick analysis
- **Cloud processing** for large video files

▮ Project Status

-  **Video Processing Pipeline:** Completed
-  **Event Detection:** Completed
-  **Report Generation:** Completed
-  **Advanced Analytics:** In Progress

- 📺 **Multi-camera Support:** Planned
- 🕒 **Real-time Processing:** Planned

★ **Star this repository if it helped you build your football analytics system!**

Built with ❤️ for football coaches and analysts