Safety Vest & Helmet Detection System Documentation

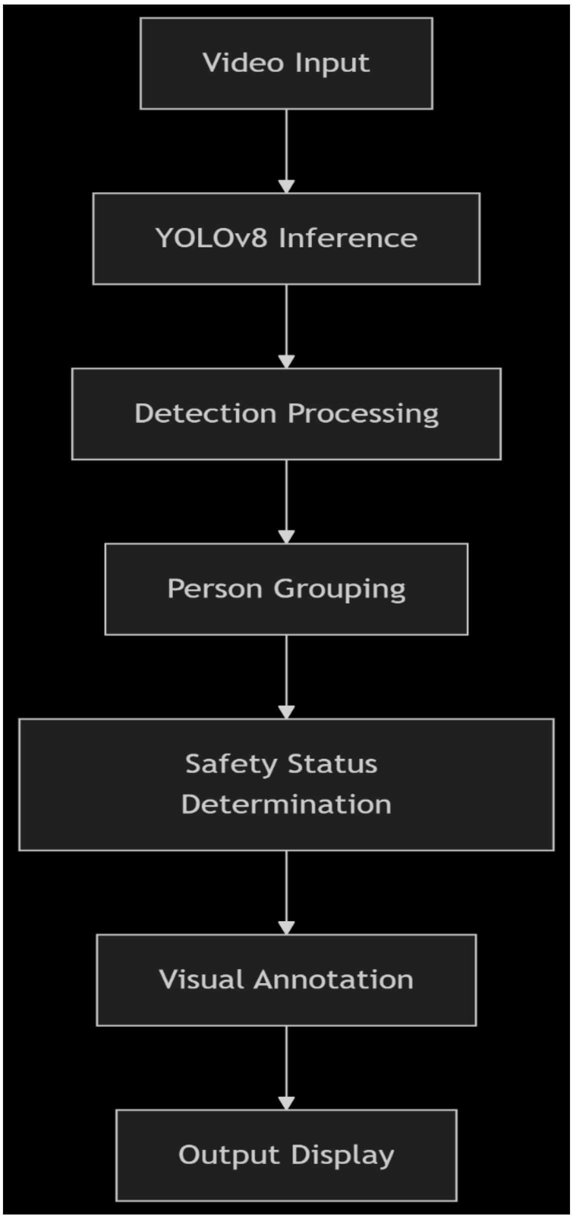
Overview

This system uses computer vision and YOLOv8 object detection to monitor safety compliance in industrial environments. It detects whether personnel are wearing safety vests, helmets, both, or neither, providing real-time visual feedback with color-coded bounding boxes and status labels.

Key Features

- Real-time detection using webcam or video input
- Four safety status classifications with color coding
- Person-centric detection grouping
- Visual feedback with bounding boxes and status labels
- Configurable confidence thresholds

System Architecture



Technical Specifications

Dependencies

Library	Version	Purpose
OpenCV	4.x	Video processing and display
Ultralytics	8.x	YOLOv8 object detection
NumPy	1.2x+	Numerical operations
Python	3.8+	Runtime environment

Detection Classes

Class ID	Label	Description
0	No Vest	Person without safety vest
1	Helmet	Safety helmet
2	Vest	Safety vest

Safety Status Codes

Status	Color	RGB Value	Description
Safe	Green	(0, 255, 0)	Both vest and helmet detected
Vest Only	Yellow	(0, 255, 255)	Only safety vest detected
Helmet Only	Cyan	(255, 255, 0)	Only helmet detected
Unsafe	Red	(0, 0, 255)	No safety equipment detected

Implementation Details

Core Functions

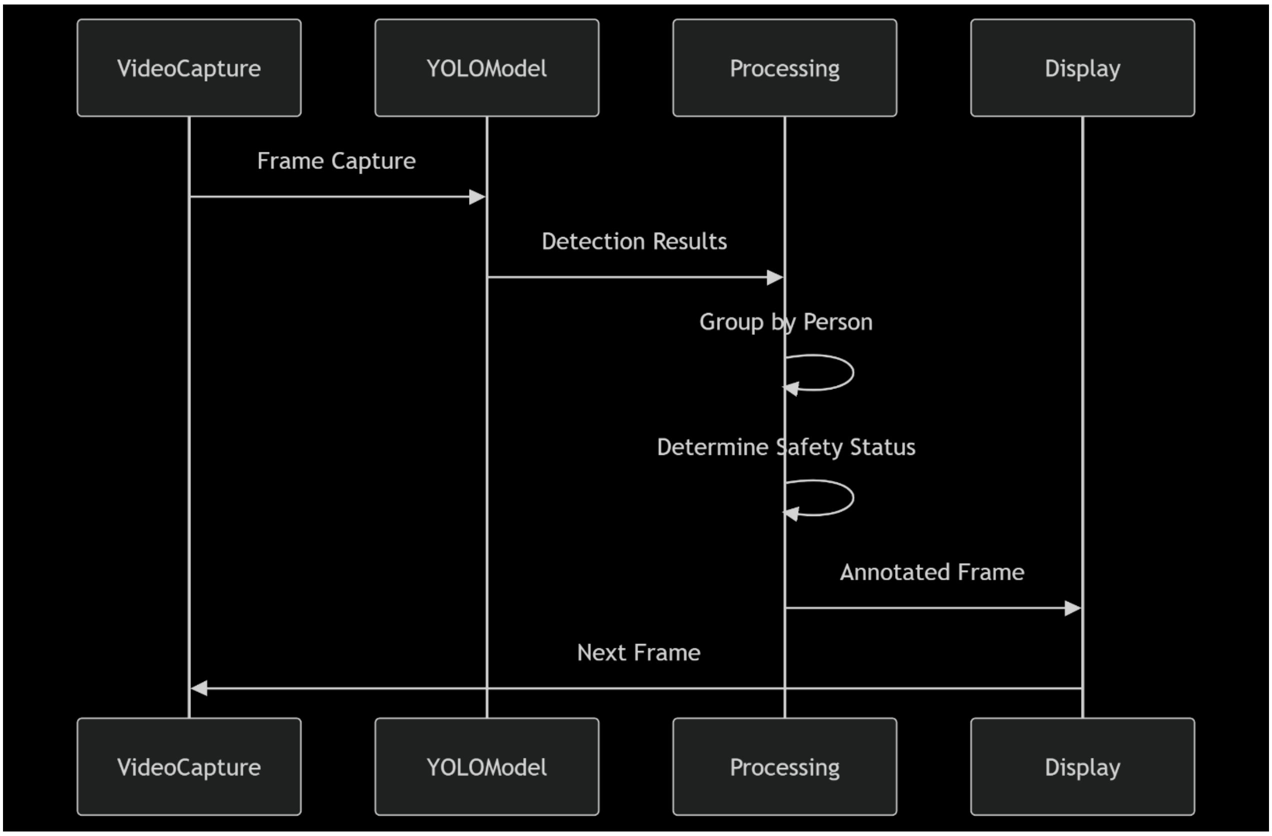
1. get_person_status(detections)

- Groups detections by person using bounding box centers
- Tracks safety equipment status per person
- Handles overlapping detections with distance threshold (100px)
- Returns dictionary with person status data

2. draw_safety_status(frame, person_status)

- Visualizes safety status using color-coded bounding boxes
- Displays safety status labels with background
- Uses largest bounding box per person for visualization
- Returns annotated frame

Main Workflow



Setup Instructions

1. **Install dependencies:**

bash

```
pip install opencv-python numpy ultralytics
```

2. **Prepare model:**
   - Place trained YOLOv8 model at: D:\Internship\AI GURU Internship\Final\Q1\runs\detect\vest_helmet_final\weights\best.pt
   - Or update MODEL_PATH variable to your model location
3. **Run the application:**

bash

```
python safety_detection.py
```

Configuration Options

Parameter	**Location**	**Description**	**Default Value**
Confidence Threshold	model.predict()	Minimum detection confidence	0.5
Device	model.predict()	Inference device	"cpu"
Max Distance	get_person_status()	Person grouping threshold	100 pixels
Frame Width	cap.set()	Camera input width	640
Frame Height	cap.set()	Camera input height	480

Performance Considerations

1. **Hardware Acceleration:**
   - To enable GPU inference, change device parameter to "cuda"

python

```
results = model.predict(frame, conf=0.5, device="cuda")
```

2. **Resolution Adjustment:**
   - Lower frame dimensions for faster processing:

Python

```
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)
```

3. **Model Optimization:**
   - Use YOLOv8n (nano) version for edge devices
   - Quantize model for faster inference

Potential Enhancements

1. **Tracking Integration:**
   - Implement object tracking (ByteTrack, DeepSORT) for consistent IDs
   - Add safety violation duration thresholds
2. **Alerting System:**
   - Audio alerts for unsafe conditions
   - SMS/email notifications for supervisors
3. **Deployment Options:**
   - Docker containerization
   - REST API for integration with other systems
   - Edge device deployment (NVIDIA Jetson, Raspberry Pi)
4. **Additional Features:**
   - Safety zone geofencing
   - Multiple camera support
   - Violation logging and reporting

Troubleshooting

Common Issues:

1. **Model not found:**
    - Verify MODEL_PATH exists
    - Use absolute path for reliability
2. **Poor detection accuracy:**
    - Increase model confidence threshold
    - Improve lighting conditions
    - Retrain model with more diverse data
3. **Low frame rate:**
    - Reduce input resolution
    - Switch to GPU inference
    - Use smaller YOLO model variant

Sample Output

The system displays real-time video with:

- Color-coded bounding boxes based on safety status
- Descriptive safety status labels
- On-screen instructions ("Press 'q' to quit")

Conclusion

This safety compliance monitoring system provides an effective solution for real-time detection of safety equipment usage. By leveraging YOLOv8's state-of-the-art object detection capabilities, it offers accurate monitoring that can be deployed in various industrial settings to enhance workplace safety protocols.