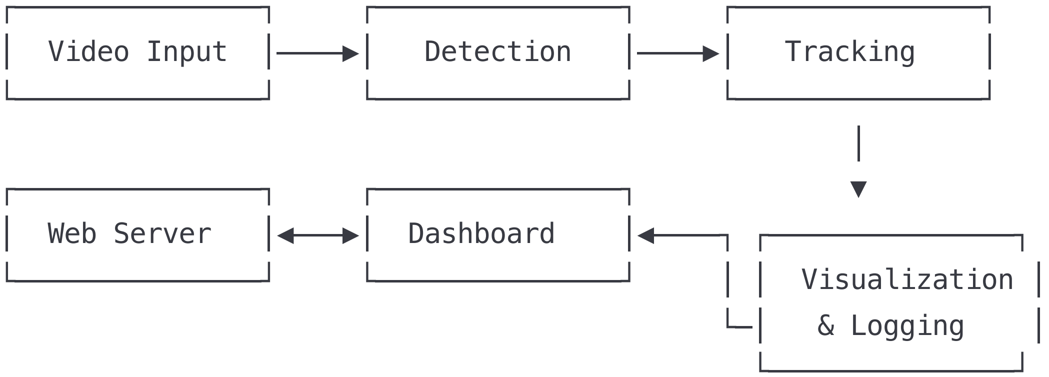
Real-Time Surveillance System

Object Detection & Tracking Application

System Overview

- **Purpose**: Real-time object detection and tracking for surveillance
- **Core Components**:
  - Video capture and processing
  - Object detection using YOLOv8
  - Object tracking with Kalman filtering
  - Web-based dashboard interface

Architecture



Technical Implementation

1. Video Processing (`video_processor.py`)

- Multi-threaded design with queue-based frame processing
- Performance metrics tracking (FPS, frame count)
- Thread management for read/process operations

```python
def start(self):
    self.stopped = False
    Thread(target=self.read_frames, daemon=True).start()
    Thread(target=self.process_frames, daemon=True).start()
    return self
```

---

2. Object Detection (`detection.py`)

- YOLOv8 model integration for real-time detection
- Automatic device selection (CPU/GPU)
- Configurable confidence threshold

```python
# Detection code snippet
results = self.model(frame, conf=self.conf_threshold, verbose=False)[0]
detections = []
for box in results.boxes:
    x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()
    confidence = box.conf[0].cpu().numpy()
    class_id = int(box.cls[0].cpu().numpy())
    class_name = results.names[class_id]

    detection = [int(x1), int(y1), int(x2), int(y2),
                float(confidence), class_id, class_name]
    detections.append(detection)
```

---

3. Object Tracking (`tracking.py`)

- Kalman filtering for motion prediction
- IOU (Intersection Over Union) based matching
- Hungarian algorithm for track assignment
- Track management (creation, updates, deletion)

python

```
# Matching algorithm
cost_matrix = np.zeros((len(self.tracks), len(detection_objects)))
for i, track in enumerate(self.tracks):
    for j, det in enumerate(detection_objects):
        cost_matrix[i, j] = 1 - self.iou(track.box, det.box)

# Hungarian algorithm for optimal assignment
row_indices, col_indices = linear_sum_assignment(cost_matrix)
```

---

4. Kalman Filtering

- State representation: `[x, y, vx, vy]`
- Prediction and update steps for smooth tracking
- Handles occlusions and missed detections

python

```
# State transition matrix
self.F = np.array([
    [1, 0, 1, 0],  # x = x + vx
    [0, 1, 0, 1],  # y = y + vy
    [0, 0, 1, 0],  # vx = vx
    [0, 0, 0, 1]   # vy = vy
])
```

---

5. Web Dashboard (`main.py` & `index.html`)

- Flask web server for streaming video
- Real-time statistics and controls
- API endpoints for system interaction:
  - `/video_feed` - MJPEG streaming
  - `/logs` - Track history
  - `/stats` - System performance
  - `/toggle_tracking` - Feature control

---

6. User Interface

- Live video feed with detection overlays
- Real-time statistics panel
  - FPS counter
  - Active tracks count
  - Runtime tracking
- Object detection counts by class
- Tracking log with timestamps
- Interactive controls (tracking toggle, snapshot)

Key Features

1. **Real-time Processing**
   - Threaded design for uninterrupted video processing
   - Optimized for performance with queue management
2. **Robust Tracking**
   - Persistent IDs for tracked objects
   - Motion prediction via Kalman filtering
   - Handles occlusions and brief disappearances
3. **Comprehensive Logging**
   - CSV-based tracking data
   - Timestamps for all detected objects
   - Position, class, and confidence scores

Technical Optimizations

- **Multi-threading** for parallel frame processing
- **Queue-based architecture** to handle varying processing times
- **Device optimization** for GPU acceleration when available
- **Efficient visualization** with color-coded object classes
- **Persistent tracking** with configurable parameters:
  - `max_age`: Maximum frames to keep track without detection
  - `min_hits`: Minimum detections to confirm a track
  - `iou_threshold`: Matching threshold

Running the Application

Command Line Options

```
python main.py --source 0 --output output --web
```

- `--source`: Video source (0 for webcam, or file path)
- `--output`: Output folder for logs and snapshots
- `--web`: Enable web dashboard (http://localhost:8080)

Future Enhancements

- Multi-camera support
- Advanced analytics (dwell time, path analysis)
- Event-based notifications
- Integration with external security systems
- Custom detection model training

Thank You

Questions?