Name: Nada Elsayed Mohamed Ahmed Elsayad **Matrícula num:** 256292
Professor Name: CONCI NICOLA **Subject Name:** Computer Vision [CV2024-25]

Human Motion Analysis Project Report

1. Introduction

Human behavior analysis in the context of video surveillance is a critical area in computer vision. Understanding the dynamics of human trajectories offers insights into patterns of movement, potential anomalies, and interactions with the environment. This analysis is especially significant in crowded or complex environments, where individual motion contributes to overall activity dynamics. The Stanford Drone Dataset, captured by UAVs surveilling different areas of the Stanford Campus, provides a rich resource for studying such trajectories. This dataset includes annotations that make it ideal for tasks like detection, tracking, and trajectory reconstruction. For this project, scenes "video0" and "video3" from the "little" set were selected for analysis. Additionally, the project sought to identify the most frequent paths of movement and reconstruct missing trajectories based on prior observations.

---

2. System Design and Features

The **Human Motion Analysis System** is a cutting-edge computer vision application designed to track and analyze human motion in video sequences. It offers three primary modes of analysis:

1. **Simple Analysis**: Basic object detection and tracking.
2. **Occlusion Tracking**: Advanced tracking with occlusion handling when the objects are hidden.
3. **Path Pattern Analysis**: Comprehensive ground truth trajectory analysis.

2.1 Key Features

- **Object Detection**: Accurately identify and localize moving objects in video sequences using advanced detection algorithms.
- **Object Tracking**: Maintain consistent object identities across frames, ensuring seamless trajectory continuity.
- **Kalman Filtering:** Provides smooth trajectory prediction and reduces noise.
- **Trajectory Analysis**: Map, analyze, and visualize movement patterns to derive insights into behavioral dynamics.
- **Path Pattern Analysis**: Examines movement patterns and identifies entry/exit points.
- **Accuracy Assessment**: Evaluate the precision of computed trajectories by comparing them against annotated ground truth data using quantitative metrics like Mean Squared Error (MSE).
- **Occlusion Handling**: Address challenges posed by temporary object occlusions through the integration of ground truth data and interpolation methods.
- **Visualization:** Offers real-time and post-processing visualization tools.

2.2 Architecture

The system is built on a modular architecture with clear separation of components as in the Figure.

```
project/
├── src/                          # Main package source code
│   ├── core/                     # Core analysis components
│   │   ├── analyzer.py           # Main orchestrator class
│   │   ├── detection.py          # Object detection module
│   │   ├── tracking.py           # Object tracking module
│   │   ├── kalman_filter.py      # Kalman filtering module
│   │   └── occlusion.py          # Occlusion handling module
│   ├── utils/                    # Utility functions
│   │   ├── metrics.py            # Accuracy metrics calculation
│   │   ├── visualization.py      # Visualization utilities
│   │   └── file_utils.py         # File I/O operations
│   ├── data/                     # Data handling
│   │   └── annotations.py        # Annotation file reading
│   ├── config/                   # Configuration
│   │   └── settings.py           # Centralized settings
│   └── __init__.py               # Package exports
├── Videos and Annotations/       # Dataset directory
├── output/                       # Output directory (created automatically)
├── run.py                        # Main run script
└── README.md                     # Basic documentation
```

The architecture is organized with a **HumanMotionAnalyzer** as the main controller coordinating the following components:

- **ObjectDetector**: Manages object detection tasks.
- **ObjectTracker**: Handles object tracking across frames.
- **KalmanFilterManager**: Applied Kalman filtering for trajectory prediction and smoothing.
- **OcclusionHandler**: Detects occlusions and re-identifies objects effectively.
- **Visualizer**: Facilitates both real-time and post-processing visualizations.
- **File Utils**: Handles file I/O operations.

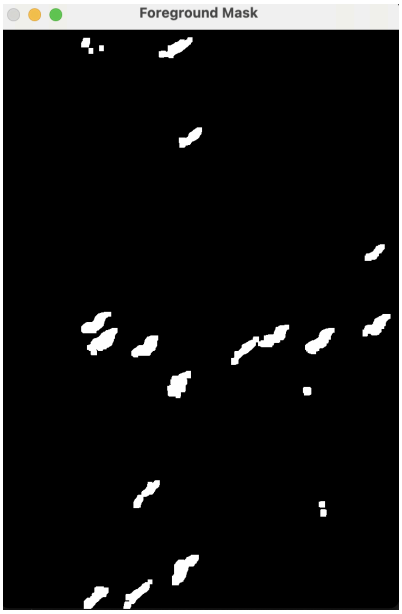
How to Run: python “run.py” **then choose an option:**

1. Simple analysis (no occlusion tracking).
2. Analysis with occlusion tracking (Bonus).
3. Trajectory and path pattern analysis based on ground truth data. Or, 4.Show help.

Output: The analysis generated a variety of output files stored in the output/ directory, showcasing the system's capabilities in object detection, tracking, and trajectory analysis. Key outputs include raw and Kalman-filtered trajectory visualizations, occlusion masks, detailed performance metrics, and path pattern analyses. Individual trajectories for tracked objects and ground truth data are also provided for comparison.

---

3. Methodology



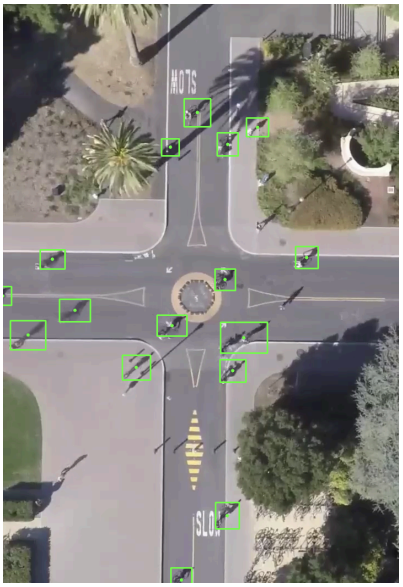
3.1 Object Detection

The object detection module is designed for efficient and accurate identification of moving objects within video frames. It leverages a combination of well-established algorithms and optimization techniques to ensure high performance and reliability.

Implementation Details

- **Class**: `ObjectDetector`
- **Location**: `project/src/core/detection.py`

Pipeline steps:



- **Background Subtraction**: MOG2 algorithm extracts moving foreground.
- **Noise Reduction**: Morphological opening and Gaussian blur remove noise.
- **Thresholding and Dilation**: Binary thresholding isolates strong detections; dilation strengthens object regions.
- **Contour Detection**: External contours identify object boundaries as in the Fig.
- **Area Filtering**: Integrates a configurable filter for minimum and maximum object sizes, ensuring detection relevance and eliminating false positives.

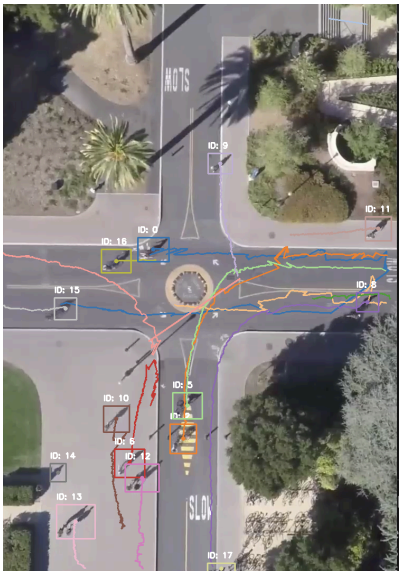
3.2 Object Tracking

The tracking module maintains persistent identities for detected objects, manages trajectories, and handles occlusions. It supports multi-object tracking and integrates with external occlusion and Kalman filtering components.

Implementation Details

- **Class**: `ObjectTracker`
- **Location**: `project/src/core/tracking.py`

Pipeline steps:



- **ID Assignment**: Unique IDs assigned and maintained across frames.
- **Distance-Based Matching**: Uses Euclidean distance for associating detections.
- **Boundary and Occlusion Handling**: Special logic for objects near frame edges or occlusion zones, reducing inaccuracies caused by partial visibility or occlusions.
- **Disappearance Handling**: Implements a configurable timeout mechanism to manage temporarily lost objects, allowing re-identification within a predefined interval.
- **Trajectory Management**: Maintains both live and full trajectory data for visualization and analysis.

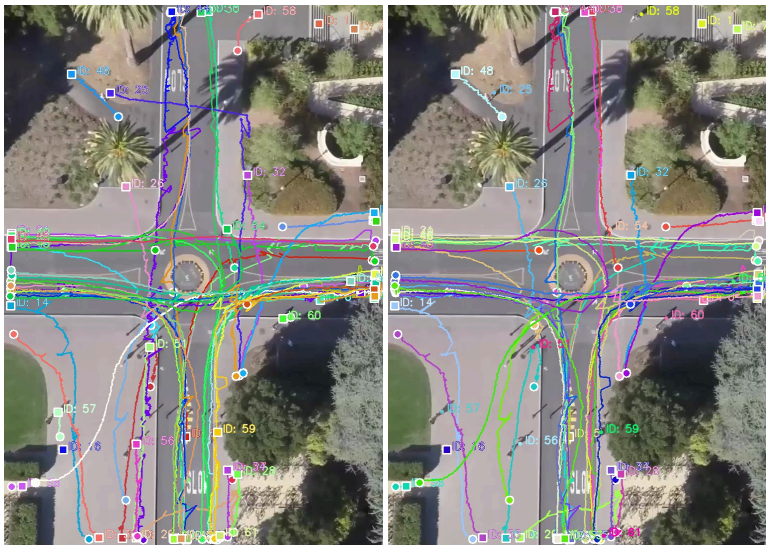
Tracking Algorithm

- Initialize or update tracked objects using detected bounding boxes.
- Handle missing detections by incrementing disappearance counters.
- Integrate occlusion handler for occluded objects.
- Match detections and tracked objects based on minimal distance.
- Manage object lifecycle including creation, update, and deletion.

3.2 Trajectory Analysis

The Trajectory Analysis processes tracked object trajectories to extract meaningful insights about movement patterns, behavior analysis, and performance evaluation against ground truth data.

Implementation Details



- **Class**: `ObjectTracker`
- **Location**: `project/src/core/tracking.py`

Pipeline steps:

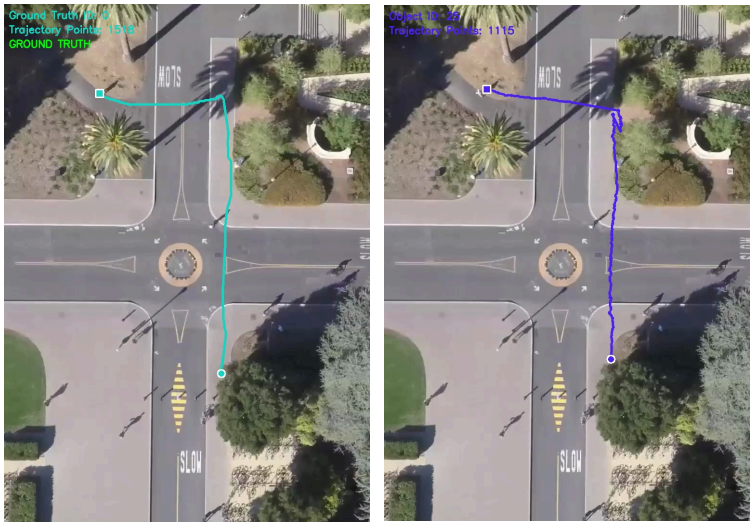
- **Data Collection:** Gather position data for each tracked object across video frames.
- **Kalman Filtering:** Apply smoothing algorithms to refine trajectory accuracy.
- **Occlusion Management:** Detect occlusion periods and pause trajectory updates during these intervals.
- **Data Storage**: Maintain two sets of trajectory data.
- **Real-Time Data**: Limited data points for immediate visualization.
- **Full Trajectories**: Comprehensive data for detailed analysis for raw ‘Fig.1'and kalman‘Fig.2’ Trajectories.

3.3 Accuracy Metrics

The Accuracy Metrics system evaluates human motion tracking performance by comparing detected trajectories against ground truth data. It applies standardized metrics such as Mean Squared Error (MSE), which quantifies spatial accuracy using scaled coordinates, requiring valid frame overlap, with thresholds to classify accuracy,. and Intersection over Union (IoU, which Measures bounding box overlap, scale-independent, computed per frame with averages across sequences.

Implementation Details

Location: `project/src/core/metrics.py`



Evaluation Pipeline

1. **Ground Truth Loading**: Parses annotations, transforms coordinates, filters occluded frames, and validates input integrity.
2. **Trajectory Matching**: Uses greedy IoU-based one-to-one matching without thresholds for initial pairing.

3. **Metrics Calculation**: Computes MSE and IoU, stores results, and aggregates match statistics.
4. **Performance Classification**: Categorizes trajectories as matched, well-matched, or unmatched based on MSE and IoU thresholds.

```
=== Final Accuracy Metrics ===
Total ground truth objects: 61
Successfully matched objects: 59
Average IoU: 0.248
Average MSE: 10.033
Very Well matched objects: 51
Average MSE well matched: 1.865
Average IoU well matched: 0.280
Ground truth ID 0 -> Detected ID 25: MSE = 1.074, IoU = 0.227
Ground truth ID 1 -> Detected ID 59: MSE = 0.403, IoU = 0.316
Ground truth ID 2 -> Detected ID 31: MSE = 0.555, IoU = 0.320
Ground truth ID 3 -> Detected ID 14: MSE = 0.945, IoU = 0.173
Ground truth ID 4 -> Detected ID 35: MSE = 0.535, IoU = 0.261
Ground truth ID 5 -> Detected ID 36: MSE = 0.151, IoU = 0.465
Ground truth ID 6 -> Detected ID 57: MSE = 0.610, IoU = 0.210
```

Result:

- Includes the Final Accuracy Metrics as shown in the image.
- Also, it shows which object from the ground truth trajectory is matched with the detected trajectory like both images.
- The “Well matched objects” are those that have MSE <= Threshold (15), these objects are nearly the same and filter out the unmatched objects.

3.4 Frequent Path Analysis

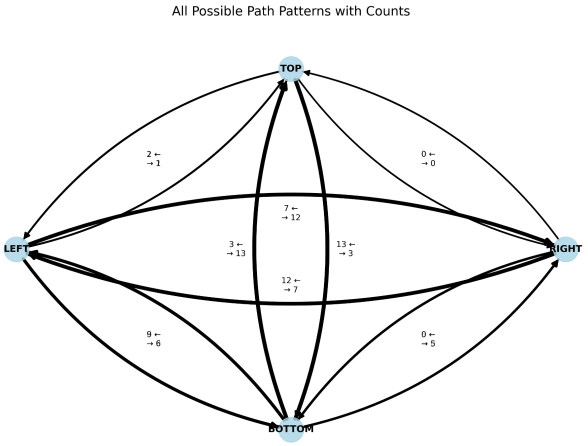
The entry and exit points of objects were analyzed to determine the most frequent paths. This involved mapping trajectories and identifying patterns based on observed movement directions.

Location: `project/src/core/annotations.py`

Result

Total path patterns found: 10 [Entry DIRECTION to Exit DIRECTION] with a graph to show the paths and counts well

- **BOTTOM to TOP:** 13 objects, Example objects: Object 9, 11, 46, 52, 53, 56, 0, 8, 23, 1, 55, 57, 59
- **LEFT to RIGHT**: 12 objects, Example objects: Object 2, 3, 10, 12, 25, 34, 35,17, 21, 22, 54, 24
- **BOTTOM to LEFT**: 9 objects, Example objects: Object 6, 13, 14, 30, 37, 48, 51, 29, 16
- **RIGHT to LEFT:** 7 objects, Example objects: 27, 28, 36, 47, 38, 39, 58
- **LEFT to BOTTOM:** 6 objects.Example objects: Object 26, 40, 41, 43, 44, 60
- **BOTTOM to RIGHT:** 5 objects. Example objects: Object 7, 31, 32, 33, 42
- **TOP to BOTTOM:** 3 objects. Example objects: Object 45, 49, 50
- **CENTER to CENTER:** 3 objects, Example objects: Object 15, 19, 20
- **TOP to LEFT:** 2 objects. Example objects: Object 4, 18
- **LEFT to TOP:** 1 object. Example objects: Object 5

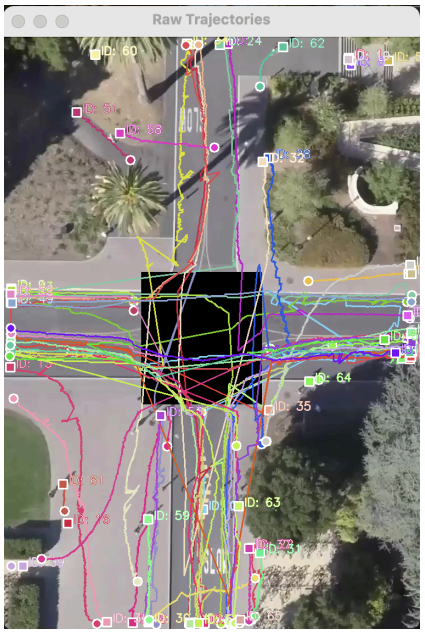


3.5 Trajectory Reconstruction (Bonus Task)

The occlusion tracking analysis is an advanced version of the motion analysis that specifically handles scenarios where objects can be occluded (hidden) by static obstacles in the scene. It uses two reference images(the one with

the hidden area and the original one) to compute an occlusion mask by subtracting them. and then tracks objects through the occluded area with the ground truth to complete the missing trajectories. It's done via some steps as:

1. **Occlusion Mask Computation**: Computes absolute difference between two reference images (occluded/non-occluded) and thresholds it
2. **Occlusion Detection and Tracking:** Monitors objects near occlusion areas, Tracks when objects first approach the occlusion, Objects stationary for 5+ frames are marked as "in occlusion" as it means it's inside this area.
3. **Object Matching During Occlusion:** When a new object appears after occlusion, the system tries to match it with previously occluded objects via a "Matching Strategy":
    a. **Trajectory Comparison:** Compares pre-occlusion trajectories with ground truth.
    b. **IoU Calculation:** Computes Intersection over Union for common frames
    c. **Frame Coverage:** Ensures sufficient overlap in time
    d. **New Detection Verification:** Checks if new detection matches ground truth
4. **Trajectory Reconstruction:** When a match is found, the system reconstructs the missing trajectory:
    a. Identifies frames between last detection and reappearance
    b. Uses ground truth data to fill missing trajectory points
    c. Scales coordinates from original resolution to video resolution
    d. Maintains trajectory continuity



Conclusion & Challenges:

The Human Motion Analysis System demonstrates significant advancements in the field of computer vision and object tracking. By employing a modular architecture, the system ensures maintainability, testability, and extensibility, while integrating advanced techniques such as Kalman filtering, multi-object tracking, and sophisticated occlusion handling. These capabilities have enabled robust trajectory analysis and behavioral pattern identification, even in complex environments.

Despite challenges such as occlusion detection, object ID consistency, and real-time performance optimization, the system successfully overcomes these limitations through innovative solutions like trajectory similarity matching, ground truth integration, and modular processing. However, areas such as detection robustness, 3D tracking, and real-time scalability offer opportunities for further improvement.

The system's contributions to understanding human motion patterns have broad applications in surveillance, traffic analysis, and behavioral research. With the incorporation of advanced deep learning techniques, multi-modal data analysis, and improved scalability, the system holds promise for evolving into a cutting-edge platform for comprehensive motion analysis. This project underscores the importance of modular design, parameter tuning, and real-world testing in building practical and impactful computer vision solutions.