D-Track α: Multi-Object Tracking (MOT) System Using Distributed Cameras

A Project Report for Mobile and Wireless Systems Course Under the Guidance of Dr. Damith Ranasinghe

Evam Kaushik
School of Computer Science, University of Adelaide
Adelaide, Australia
evam.kaushik@adelaide.edu.au

Abstract

This report presents D-Track α, a distributed multi-object tracking system utilizing a network of Raspberry Pi cameras with non-overlapping fields of view (FOV). The system employs the YOLOv4 model for local person detection, integrates instance fusion using DBSCAN clustering, and applies Kalman filtering for trajectory tracking. A proof of concept is demonstrated using the WILDTRACK dataset.

This project was developed collaboratively by Evam Kaushik, Tom Zhu, and Lennox Avdiu, however, this report specifically reflects Evam Kaushik's individual contributions. The code and project details are available at `https://github.com/kickereb/D-Track-Alpha`

Keywords: Multi-Object Tracking, YOLOv4, DBSCAN, Kalman Filter, Non-Overlapping Field of View, Raspberry Pi, WILDTRACK

1 Aim

D-Track α project is created in order to provide a reliable and cost-effective way to perform multi-camera tracking with non-overlapping fields of view.

There are numerous benefits of having such systems, such as:

- Safety enhancement for sensitive public spaces, especially involving minors and the senior population.
- Equipment cost reduction by deploying fewer cameras to cover indoor spaces.
- Generating a reliable knowledge base for the study of human behaviour in public spaces.
- Cost-effective and reliable person-of-interest tracking.

2 Introduction

Multi-Object Tracking (MOT) aims to solve the detection and consistent tracking of multiple objects across consecutive frames in a video sequence, preserving their identities over time. We are specifically interested in tracking dynamic objects[1]. Since most computer vision-based tracking solutions are stochastic [11, 15], we face two major challenges when performing MOT: the presence of a high amount of noise in the sensor inputs and the accuracy of data fusion.

Multiple Hypothesis Tracking (MHT), Joint Probabilistic Data Association (JPDA) [2], and Random Finite Sets (RFS) are prominent frameworks in multi-object tracking. Both MHT, which is the more established approach, and JPDA solve the data association issue first and then proceed to estimate the states of the objects in

[1] Dynamic objects refer to objects of interest whose position changes with respect to the environment with each consecutive frame, e.g., pedestrians walking on the roadside, vehicles driving on the road, etc.

the observation. This method has shortcomings in managing the birth and death of objects. RFS helps overcome this limitation by providing a new perspective: rather than explicitly associating data points to each object, the space of all possible locations is modeled as a set-valued random variable. The number of objects becomes dynamic, varying over time with the use of Probability Hypothesis Density (PHD), Cardinalized Probability Hypothesis Density (CPHD), Multi-Bernoulli (MB), Poisson Multi-Bernoulli Mixture (PMBM), Labeled Multi-Bernoulli (LMB), and Generalized Labeled Multi-Bernoulli (GLMB) filters [5].

Furthermore, when we consider Distributed Multi-Object Tracking (DMOT), we also encounter the problem of limited computational resources and channel bandwidth.

3 Literature Review

3.1 Distributed Multi-Object Tracking

Chen et al. [5]: This paper maps out the work in designing a distributed multi-object tracking algorithm that makes it possible to work with heterogeneous sensor networks where the fields of view are limited and are non-uniform. The method utilizes cluster-based analysis for track fusion and graph-based algorithms for label consistency. These allow the method to alleviate issues of limited computational resources as well as low levels of communication bandwidth in order to enhance the system's robustness in areas where many resources are constrained.

Jia et al. [12]: Presents a Bayesian multi-object tracking framework with a single filter that encompasses track initialization, termination, re-identification, and handling occlusion, making it efficient for real-time applications. The method enables effective dealing with dynamic changes of the environment with multi-camera setups, and there is no need for detector retraining.

Robootx [13]: This GitHub repository presents a technique for multi-camera object tracking via deep metric learning by transferring representations to a top-view perspective. The method uses a neural network-based feature map representation to learn instance metrics and applies Principal Component Analysis (PCA) on the learned embeddings to cluster features together. This approach reduces inference time and CPU load.

3.2 Camera Calibration

Zhang [16]: Proposes a flexible camera calibration approach that overlays pattern images of a planar shape that has been tilted, thereby minimizing the complexity involved in having to use other supports at the beginning. This technique does not require much in terms of setup, and so is ideal for quick deployments that require adjustments of the camera since the final results are usually free from radial lens distortion effects.

3.3 Stereo Vision

Guo et al. [9]: The authors formulate the Group-wise Correlation (GWC) Stereo Network for stereo matching as a standardized hierarchical matching problem where depth maps are marked for viewing, and match measures between features are grouped into regions. This is achieved by decomposing images into planar patches for which matches over additional dimensions are known, thus yielding better depth data while requiring fewer comparisons. This component of D-Track relies on GWC's methods to improve tracking across numerous cameras especially when they are closely aligned and oriented.

Xu et al. [**?**]: This paper introduces HITNet, which is a Hierarchical Iterative Tile Refinement Network intended for real-time stereo matching. HITNet benefits from a cost volume upsampling module based on bilateral grids which allows for edge-avoiding high-resolution disparity estimation, significantly enhancing the speed of stereo net-matching networks without affecting precision. Xu et al. accumulate the effectiveness of multiple channels in the merging strokes and therefore have not presented the issue yet.

3.4 Pose Estimation

Geng et al. [8]: In their work, they show that realistic brute force methods in image capture and processing can be replaced by internal parameters and pose as Compositional Tokens (PCT) through their new formulation that uses discrete tokens to encapsulate joint dependencies. Such an approach lowers the number of erroneous pose

estimations which go above reasonable and provides comparable or even better outcomes relative to previously existing methodologies even in those cases when occlusions are a problem.

It proposed the greatest accuracy/complexity tradeoff. The more accurate a pose recognition model gets, the more computational resources it demands and hence, becomes that much less suitable for deployment on edge devices.

3.5 Multiple Object Tracking Methods

Chen et al. [6]: This paper provides information about a method to perform multiple object tracking through the use of Edge Multi-Channel Gradient Model (E-McGM) in combination with ORB.

3.6 Additional Tools and Resources

OpenTraj [7]: OpenTraj is an open-source toolbox for human trajectory analysis that supports multiple existing datasets for Human Trajectory Prediction (HTP) tasks. It provides tools to load, visualize, and analyze datasets, facilitating research and development in trajectory prediction and analysis. For D-Track, OpenTraj can be valuable for testing and evaluating tracking algorithms on established datasets like WILDTRACK.

4 Methodology

4.1 System Design

The final system architecture that we implemented is given below.

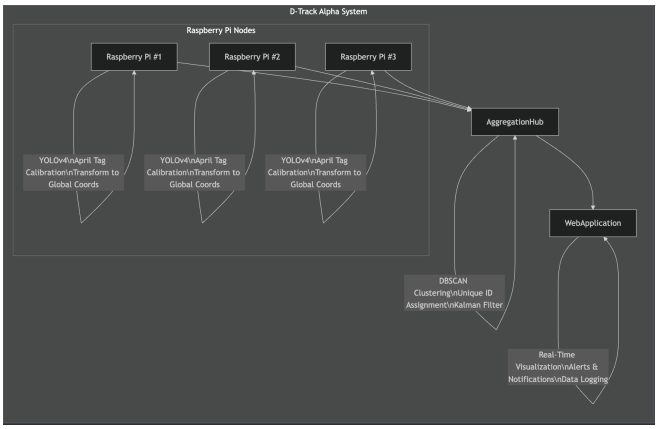

Figure 1: System Diagram for D-Track

- Each Raspberry Pi node is connected via ethernet to a routing hub, and has an attached camera module.
- My specific role in the project was to setup a person detection algorithm on each Pi cam which was achieved using the YOLOv4 architecture.

5 Implementation and Results

Herein, I will discuss some breakthroughs that propelled the project forward.

5.1 Probability Density Cloud Estimation

I was able to create a crude predictor of a person's trajectory by generating a Probability Density Point Cloud Estimation that associated the distance moved in each frame to their velocity.

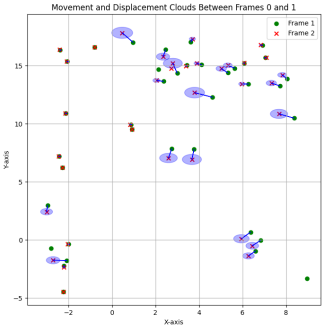

Figure 2: Probability Density Point Cloud Estimation on the WILDTRACK

5.2 Clustering for Resolution

I implemented a simple DBSCAN clustering to resolve multi-camera conflicts. This is inspired by Z.Cao et. al's work on Multi-view Person Association [4] who used t-SNE3.

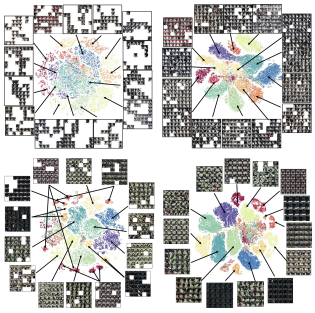

Figure 3: Multi-view Person Association resolves ID conflicts using t-SNE clustering

5.3 FPS Boosting

I was able to increase the YOLO inference time by upto 1.5 times by saving it as a NCNN file from the article *Getting Started with YOLO Object and Animal Recognition on the Raspberry Pi* [10] by Core Electronics.

5.4 Kalman Filter on WILDTRACK

I was able to apply Kalman Filter on WILDTRACK dataset's post-processed tags and generate a kinematic estimation of the a person's trajectory6. Code snippet is provided in the AppendixB

6 Conclusion

> "Turning over rocks and finding nothing is [still] progress."
> – *Karen, I Origins (2013)*

We implemented a Multi-Object Tracking (MOT) System Using Distributed Cameras using Raspberry Pi with camera modules and it worked satisfactorily during a live demo. It was able to get an input stream from 3 Pi cams, each of which would locally process the video stream to perform person detection using YOLOv4 model. The pointcloud was then processed distributedly using DBSCAN clustering to generate a fused frame.

There were many challenges that we had to overcome in order to get D-Track α up and running, the most notable of which, in the begining would be to get the cameras calibrated. Whilst I kept working parallely on the theoretical implementation of the tracking algorithm, the team was able to set up the prject space after several trail and error practices**??**.

Figure 4: Boosted YOLOv8-nano used to test NCNN model parameters

Figure 5: Tracker Output Visualization

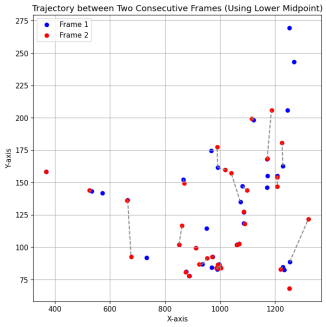

Figure 6: Kalman Filtering for upto 2 frames on the WILDTRACK dataset

7 Future Works

7.1 Model Fine-tuning

Ample literature suggests that large object detection models like YOLO can be fine-tuned to specific tasks to improve efficiency and reduce computational load. For instance, fine-tuning YOLO to detect only specific classes is a suggested practice for targeted applications [3]. In this project, though, I attempted to create a YOLO model that is specifically trained to detect only the 'Person' class instead of detecting all classes in the COCO dataset and then filtering the Person class.

7.2 Use of Track Fusion Algorithms instead of Object Detection Algorithms

- **Combine Sensor Data**: Track fusion algorithms merge observations from multiple sensors to improve the accuracy of object position and trajectory estimates in multi-object tracking [1].
- **Maintain Consistent Identities**: They ensure consistent object identities across different sensors or viewpoints, resolving ambiguities when objects move between fields of view [1].

References

[1] Y. Bar-Shalom and X. R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*, YBS Publishing, 1995.

[2] Y. Bar-Shalom, P. K. Willett, and X. Tian, *Tracking and Data Fusion*, vol. 11, Storrs, CT, USA: YBS Publishing, 2011.

[3] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.

[4] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *arXiv preprint arXiv:1805.08717*, 2018.

[5] F. Chen, H. V. Nguyen, A. S. Leong, S. Panicker, R. Baker, and D. C. Ranasinghe, "Distributed Multi-Object Tracking Under Limited Field of View Sensors via Data Clustering," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4929–4942, 2021.

[6] J. Chen, X. Li, Y. Li, and Y. Li, "Multiple Object Tracking Using Edge Multi-Channel Gradient Model with ORB Feature," *IEEE Access*, vol. 8, pp. 150214–150227, 2020.

[7] Crowdbot Project, "OpenTraj: A Toolbox for Human Trajectory Analysis," GitHub repository, 2020. [Online]. Available: `https://github.com/crowdbotp/OpenTraj`

[8] Z. Geng, C. Wang, Y. Wei, Z. Liu, H. Li, and H. Hu, "Human Pose as Compositional Tokens," *arXiv preprint arXiv:2303.11638*, 2023.

[9] X. Guo, K. Yang, W. Yang, X. Wang, and H. Li, "Group-wise Correlation Stereo Network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3273–3282.

[10] M. Horne, "Getting Started with YOLO Object and Animal Recognition on the Raspberry Pi," Core Electronics, 25-Jan-2023. [Online]. Available: `https://core-electronics.com.au/guides/raspberry-pi/getting-started-with-yolo-object-and-animal-recognition-on-the-raspberry-pi/`

[11] M. Isard and A. Blake, "CONDENSATION—Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[12] M. Jia, A. Tsokas, and A. H. Moore, "Track Initialization and Re-identification for 3D Multi-View Multi-Object Tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 12944–12953.

[13] Robootx, "Multi-Camera Object Tracking via Transferring Representation to Top View," GitHub repository, 2021. [Online]. Available: `https://github.com/Robootx/Multi-Camera-Object-Tracking-via-Transferring-Representation-to-Top-View`

[14] B. Xu, Y. Xu, X. Yang, W. Jia, and Y. Guo, "HITNet: Hierarchical Iterative Tile Refinement Network for Real-time Stereo Matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[15] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 13, 2006.

[16] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

A Project Repository

The code and project details are available at `https://github.com/kickereb/D-Track-Alpha`

B Collaborators

This project was developed collaboratively by:

- Evam Kaushik
  (a1909167, `evam.kaushik@adelaide.edu.au`)
- Tom Zhu
  (a1770422, `tom.zhu@adelaide.edu.au`)
- Lennox Avdiu
  (a1774765, `lennox.avdiu@adelaide.edu.au`)

Figure 7: D-Track α, before presentation 1.

Hardware Setup

My job was supposedly the least hardware-intensive when compared to Tom's or Lennox's. I was initially tasked with implementing a track fusion algorithm onto the PIs. Whilst I mostly developed Proof of Concepts (POCs) on the WILIDTRACK dataset, I did remain in touch with the team to parallely monitor the progress of the project.

Here's is the setup that I used to run all my proof of concepts.8

Code Snippets and Script Examples

Implementation of Kalman Filtering on 2 consecutive WILDTRACK Frames

```
    import matplotlib.pyplot as plt
import json
import numpy as np
from scipy.spatial import distance
```

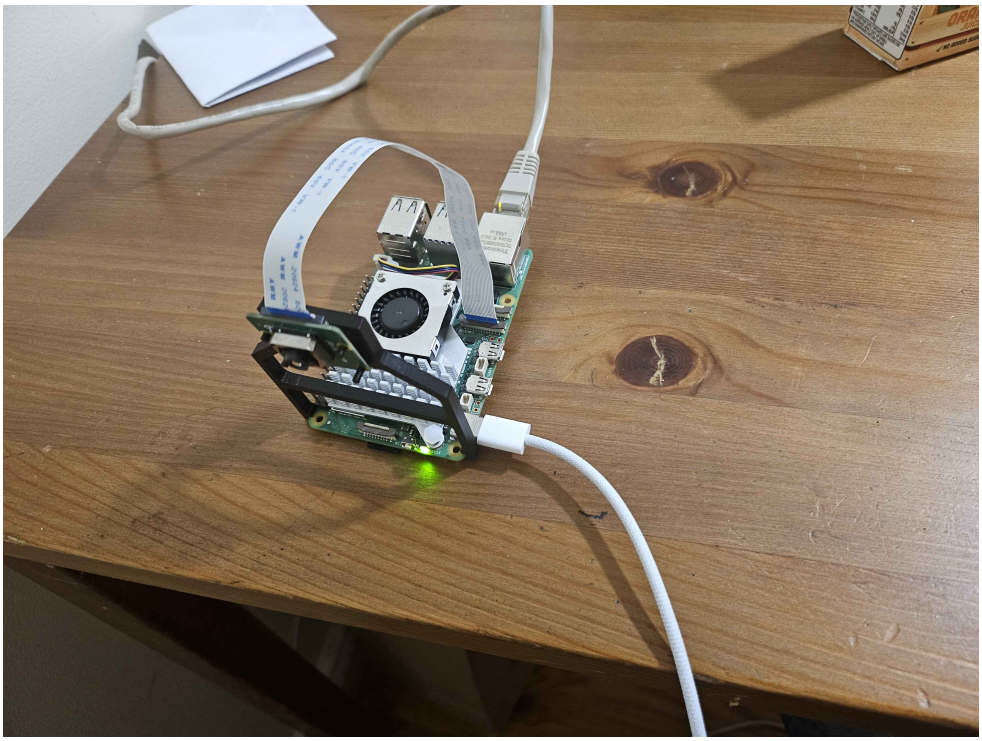

Figure 8: Raspberry Pi with a camera module

```
# Function to calculate the midpoint of the lower row of a bounding box
def calculate_lower_midpoint(view):
    if view['xmax'] == -1 or view['xmin'] == -1 or view['ymax'] == -1 or view['ymin'] == -1:
        return None
    x_mid = (view['xmax'] + view['xmin']) / 2  # Midpoint on the X-axis
    y_lower = view['ymin']  # Lower Y-axis bound
    return np.array([x_mid, y_lower])

# Function to get all valid positions for each person in a frame
def get_positions_from_frame(file_path):
    with open(file_path, 'r') as f:
        data = json.load(f)
    positions = []
    for person in data:
        midpoints = [calculate_lower_midpoint(view) for view in person['views'] if calculate_lower_midpo
        if midpoints:
            avg_position = np.mean(midpoints, axis=0)  # Average midpoint across views
            positions.append(avg_position)
    return np.array(positions)

# Function to match points between two frames and plot trajectories
def plot_trajectory_between_frames(file_path1, file_path2):
    # Get positions for each frame
    positions_frame1 = get_positions_from_frame(file_path1)
    positions_frame2 = get_positions_from_frame(file_path2)

    plt.figure(figsize=(8, 8))
```

```
    # Plot points for both frames
    plt.scatter(positions_frame1[:, 0], positions_frame1[:, 1], color='blue', label='Frame 1')
    plt.scatter(positions_frame2[:, 0], positions_frame2[:, 1], color='red', label='Frame 2')

    # For each point in frame 2, find the nearest point in frame 1 and draw a line
    for pos2 in positions_frame2:
        distances = distance.cdist([pos2], positions_frame1, 'euclidean')
        nearest_idx = np.argmin(distances)
        nearest_pos1 = positions_frame1[nearest_idx]

        # Draw trajectory line
        plt.plot([nearest_pos1[0], pos2[0]], [nearest_pos1[1], pos2[1]], color='gray', linestyle='--')

    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.title('Trajectory between Two Consecutive Frames (Using Lower Midpoint)')
    plt.legend()
    plt.grid(True)
    plt.show()

# Define file paths
file_path1 = '/content/annotations_positions/00000000.json'
file_path2 = '/content/annotations_positions/00000005.json'

# Plot trajectories between the frames
plot_trajectory_between_frames(file_path1, file_path2)
```

My Thoughts on the Process

It now feels like we could've completed this project within a week or two of having completely understood the domain inside-out. I would even compare it to learning French by a) Studying French in school for a year vs. b) living in France; there's an emergence of higher order principles as well as the setting in of foundational principles when you immerse youreslf in a field. I also attribute some hampering in my progress to the fact that I overloaded a course during this semester along with pursuing Research. Either way, the learnings will stay with and are best sumarised by the following.

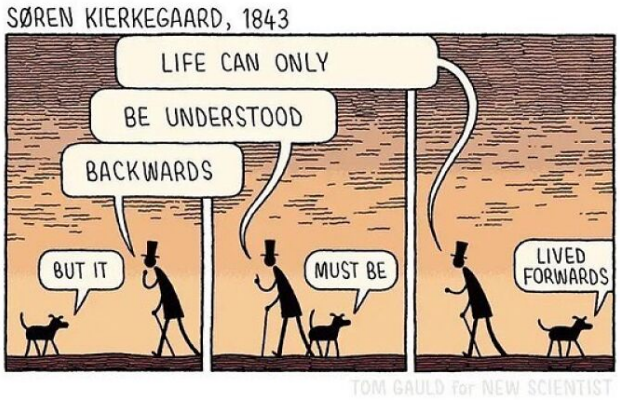

Figure 9: Søren Kierkegaard's quote illustrated by Tom Gauld for *New Scientist*