3D Motion Tracking and Analysis with Dual-Camera

David Femi Lamptey
College of Engineering
Northeastern University
Boston, US
lamptey.f@northeastern.edu

Ryan Walton
College of Engineering
Northeastern University
Boston, US
walton.ry@northeastern.edu

Chaohan Li
College of Engineering
Northeastern University
Boston, US
li.chaoh@northeastern.edu

***Abstract*—This project built a dual-camera system aimed at detecting and quantifying the position and velocity of moving objects using stereo techniques. The primary focus is on applying computer vision and image processing algorithms to video data collected from two statically mounted cameras with overlapping fields of view. Our method involves calibrating the cameras to get the camera matrix and correct for optical distortions. Subsequent to calibration, the system employs motion segmentation to isolate moving objects in the video frames. Feature extraction techniques are then used to identify corresponding points between the two camera views. By employing stereo vision algorithms, we calculate the three-dimensional coordinates and trajectories of the detected objects.**

Initial experiments are conducted using a controlled setup with a single slow-moving object, gradually escalating to more complex scenarios. The base objective is to accurately determine the position and velocity of a person walking across the cameras' field of view. Success is measured by the precision of the location and velocity estimates relative to ground truth measurements. As stretch goals, we aim to extend the system's utility to high-speed, smaller objects such as a plastic ball in flight.

This project makes an executable implementation in the realms of security, surveillance, and motion analysis, offering a robust tool for spatial monitoring and dynamic object tracking.

***Index Terms*—camera calibration, motion segmentation, feature extraction, object Tracking**

I. Introduction

A typical application of a statically mounted security or surveillance camera is to detect moving objects in a static background. It may also be interesting to determine the location of the moving object in world coordinates. In some applications, security or surveillance cameras may be placed such that adjacent cameras have overlapping fields of view. In such a configuration, stereo vision methods can be used to determine the 3D location of objects detected in both frames.

An additional attribute of the statically mounted security camera case, is that background subtraction can be applied to perform basic motion segmentation in the image. For detecting known objects, however, there are existing algorithms that can be leveraged and directly applied. For example, human detection algorithms are common and robust.

In this paper, we describe the work done for this project: We collected data using iphone cameras statically mounted. We tackle the major calibration steps needed for the stereo system with statically mounted cameras. We apply basic motion segmentation and existing human recognition algorithms to detect people. Finally, we use the calibrated system to estimate a person's location and velocity in world coordinates.

First, we will discuss the problem at a high level. Then we will discuss the technical approach we executed. Finally, we will show results and draw conclusions.

II. Problem Formation

The primary challenge addressed in this project is the accurate detection and quantification of the position and velocity of moving objects within the field of view of statically mounted surveillance cameras using stereo vision techniques. In surveillance systems, it is crucial not only to detect motion but also to accurately determine the spatial coordinates and movement characteristics of objects in real-world units.

Stereo vision involves capturing the same scene from two different points of view, allowing for the extraction of 3D information from 2D captures. This process requires precise camera calibration to correct for lens distortions and to align the cameras effectively. Motion segmentation then isolates moving objects from the static background, and feature extraction techniques are applied to find corresponding points between the images captured from each camera. These points are used to compute the position and trajectory of the objects in three-dimensional space.

This problem is fundamental in enhancing the functionality of surveillance systems, which are increasingly relied upon for security monitoring and management. The ability to accurately track the velocity and trajectory of objects can significantly improve response strategies in real-time surveillance operations.

III. Technical Approach

The technical approach of the project can be divided into two main sections, "System Calibration" and "System Appli-

cation". The first section covers the initial step of characterizing and calibrating the two camera system, the second covers the algorithms developed and used to perform object detection, tracking, and velocity estimation.

A. System Calibration

We utilized two iPhone cameras, mounted to ensure overlapping fields of view. The efficacy of stereo vision systems hinges on the precise calibration of the cameras used. Calibration is the process by which we determine the intrinsic (camera-specific) and extrinsic (position and orientation) parameters, which allow us to accurately relate the 2D image points to 3D world coordinates.

1) Intrinsic Calibration: Intrinsic parameters describe the camera's internal characteristics, such as its focal length and optical center. These parameters are encapsulated in the camera matrix, a fundamental element for mapping 3D points to 2D coordinates. The intrinsic camera matrix is derived using the pinhole camera model, represented by the equation:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

where f is the focal length of the camera, and (x_c, y_c, z_c) are the coordinates of a point in camera-centric space. The coordinates (x, y) are then scaled and translated into pixel coordinates (u, v) using the equations:

$$\begin{cases} u = u_0 + \frac{x}{dx} \\ v = v_0 + \frac{y}{dy} \end{cases}$$

Here, (u_0, v_0) represents the optical center of the camera, and (dx, dy) are the physical dimensions of the camera's sensor pixel.

Lens distortion affects image quality, with radial distortion causing images to appear distorted at the edges. To correct for radial and tangential lens distortion, we use the following equations:

For radial distortion correction:

$$x_{\text{corrected}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{\text{corrected}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

For tangential distortion correction:

$$x_{\text{corrected}} = x + \left[2p_1 y + p_2(r^2 + 2x^2)\right]$$
$$y_{\text{corrected}} = y + \left[2p_2 x + p_1(r^2 + 2y^2)\right]$$

where x and y are the coordinates of a point in a distorted image, r is the distance from the center of the image to the point (x, y), and k_1, k_2, k_3 are the radial distortion coefficients, and p_1, p_2 are the tangential distortion coefficients. These coefficients are obtained during the calibration process.

These corrections are applied to each image captured by the cameras, ensuring that the extracted feature points are as accurate as possible before proceeding with stereo matching and 3D reconstruction.

For calibration, we captured multiple images of a standard chessboard pattern placed at different orientations and distances from the cameras. Utilizing the captured images, we computed the intrinsic and extrinsic parameters using the calibration functions available in the OpenCV library. This process was repeated for each camera to ensure precision.

Once calibrated, the camera matrices and distortion coefficients were used to rectify the images captured from each camera, ensuring that the corresponding points in the images aligned on the same rows. This rectification process is crucial for accurate depth estimation in subsequent stereo matching algorithms.

2) Extrinsic Calibration: Extrinsic calibration establishes the two cameras' relative position and orientation. The final result of extrinsic calibration is a rotation matrix R and a translation vector T, which describes the transformation between the two camera frames. This allows us to translate points from camera 1 coordinates (x_{c1}, y_{c1}, z_{c1}) to camera 2 coordinates (x_{c2}, y_{c2}, z_{c2}) using the equation:

$$\begin{bmatrix} x_{c2} \\ y_{c2} \\ z_{c2} \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{c1} \\ y_{c1} \\ z_{c1} \\ 1 \end{bmatrix}$$

The first step in finding the rotation and translation is to determine the system's fundamental matrix. This matrix provides a transformation from points in one camera's image to epipolar lines in the other camera's image.

In order to do this, we manually found point correspondences in the image, and applied the 8-Point [4] algorithm in order to estimate the fundamental matrix, using an OpenCV implementation.

Next step is to calculate the essential matrix, which relates the relative location of the two camera frames in the world. It can be expressed as the following:

$$E = RS$$

Where R is the relative rotation between the camera frames and S is the matrix form of the cross product of the translation between the camera frames.

The essential matrix (E) can be calculated directly from the camera intrinsic matrices (K_1 and K_2) and the fundamental (F) matrix, both described above.

$$E = K_2^T F K_1$$

Finally, the rotation and translation between the two camera frames must be recovered from the essential matrix. In order to do this, we follow an algorithm outlined in section 9.6.2 of Hartley [5].

First, and SVD of the essential matrix is taken as $U\sigma V^T$. The translation is determined up to a scale and a sign as the last column of U. The rotation matrix is given as either UWV^T or UW^TV^T, where W is:

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The correct translation sign and rotation matrix are determined by manually comparing to the approximate known scene parameters.

However, the translation is still only know up to a scale. In order to finalize the calibration, this scale must be determined. In the scene we placed a tape measure and we found the image coordinates of two locations on the tape measure in both image frames. These points now have a know 3D displacement.

We swept a translation scale parameter from a low value to a high value in small increments. At each translation scale, we attempted 3D triangulation of the tape measure locations in image coordinates to 3D coordinates compared the distance between the 3D coordinates with the known 3D displacement of the tape measure. The translation scale that minimized this error we took to be our best estimate.

B. System Application

Once the system was properly calibration, data it produces is now ready to be processed by application level algorithms.

1) Tracking: In order to perform tracking, we made use of YOLOv8 [1], a state-of-the-art deep learning model, for the task of object tracking across two simultaneous video streams. This implementation leverages the robust feature extraction capabilities of YOLOv8 to maintain continuity and accuracy in tracking an object as it moves from the field of view of one camera to another. For the implementation, the model's input layer receives extracted frames from each video stream. The model predicts bounding boxes, object classes, and confidence scores for each detected object.

To maintain object identities as they move between camera views, we integrate ByteTrack, a highly efficient tracking algorithm. ByteTrack [2] exploits the bounding boxes and class predictions provided by YOLOv8 to assign and maintain unique IDs for each detected object using both appearance and motion information.

ByteTrack employs a combination of IoU scores and a Kalman filter to link object identities across frames effectively. This methodology ensures that objects retain their IDs even when transitioning between streams, using spatial and temporal cues to facilitate re-identification in the new camera view.

Our experiments involve the synchronized operation of two calibrated iPhone cameras in a stereo rig. The two video streams from these cameras are passed into two YOLO models that share memory, allowing for identified targets to be shared between the two models and streams of videos. The YOLO model then returns bounding boxes for the identified targets in each of the video frames, from which we extract the center of the bounding box for each target in each video. We then store the center coordinates for the target, for each frame in each video for later use in point triangulation to determine the position of the target in 3D space.

2) Point Triangulation and Velocity Estimation: Finally, we can use the extrinsic parameters of the camera to determine the projection matrix for both cameras. This allows us to project real world locations onto the image, and conversely to triangulate the real life coordinates of a target given image points of the target in two cameras. Each set of 2D coordinates from the cameras undergoes smoothing using the Savitzky-Golay filter [3] to reduce noise and improve the accuracy of the triangulation. The smoothed coordinates are then used to compute the 3D positions through the triangulatePoints function, which applies stereo triangulation. The resulting homogeneous coordinates are converted to Euclidean coordinates by normalizing with respect to the fourth dimension. This procedure is repeated for each frame, resulting in a sequence of 3D points that describe the trajectory of the target over time.

Then, with information from the videos about the times the target was at each position, we can calculate the velocity by taking the first derivative of the positional data with respect to time, which is approximated by the frame rate of the video capture (30 frames per second in this case). The differences between successive 3D positions yield displacement vectors, which are divided by the time interval to estimate the velocity vectors in three-dimensional space.

To refine the velocity estimates and mitigate any fluctuations due to discrepancies in the frame-to-frame triangulation, we again apply the Savitzky-Golay filter. This smoothed velocity data is then parsed into its component vectors x, y, z and the magnitude of the overall velocity is calculated using the Euclidean norm.

IV. Results

We recorded two intrinsic camera calibration videos with each camera, pointed at checkerboard pattern. We also recorded two test videos, one in which a target was walking, and another in which the target was running, and then applied the discussed techniques in order to calculate the speed of the target in each video.

A. System Calibration

1) Camera Matrix Estimation: In performing system calibration, the first step was to calculate the camera's intrinsic matrices. This was done be taking a video of a checkerboard pattern displayed on a laptop screen. We developed a simple script using OpenCV functions to find corners in the checkerboard for each frame in the video, and perform the calibration calculation.

We found that we had to be careful to select the correct capture type using the iPhone camera. There are HD and 4K options for video, as well as other settings for still images. We made sure to use the same settings for the calibration video capture as the data collection video capture in order to ensure that the camera calibration matrix would be consistent.

Figure 1 shows the camera matrix calibration setup, where a checkerboard pattern on a laptop screen is imaged.

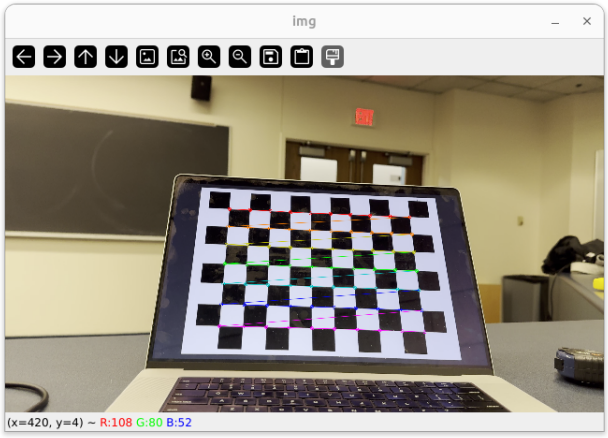


Fig. 1: Camera Matrix Estimation

2) Fundamental Matrix Estimation: After calibrating the camera matrices for each iPhone, the cameras were mounted slightly apart from each other, both facing similar sections of the camera scene. Common points in the scene, mostly corners of objects in the image, were manually selected in both camera frames with a python tool that was developed.

Then, the OpenCV function findFundamentalMat was used to estimate an optimal fundamental matrix to match the set of collected points. Care was taken to ensure the set of correspondence points have enough depth variation, since set of co-planar points cannot estimate the fundamental matrix. This is the reason for the backpack and wooden object in the foreground of the image.

Figure 2 shows the two camera frames with the set of correspondence points appearing as circles in the two images, with consistent coloring between corresponding points. In order to verify and visualize the calculation of the fundamental matrix, epipolar lines corresponding to points in the other camera were also plotted in Figure 2.

Notably, the epipolar lines in the lower image in Figure 2 are less parallel than those in the upper image. This is due to the fact that the camera corresponding to the lower image was pointed slightly towards the other camera, while the camera corresponding to the upper image was not.

3) Rotation and Translation Estimation: After estimating the fundamental matrix, rotation and translation direction were estimated. As described in the technical approach section, this involves first determining the essential matrix, and performing an SVD and analyzing the results.

In order to verify the estimation, a new fundamental matrix can be directly computed from the the camera matrices, the rotation, and the translation direction. Figure 3 shows the same epipolar line verification plot that Figure 2 does. Notably, there is some error between the points and corresponding epipolar lines now.

This is due to measurement error, and the constraint on the SVD that assumes equal valued entries for the first two coefficients, and a zeroed 3rd coefficient.

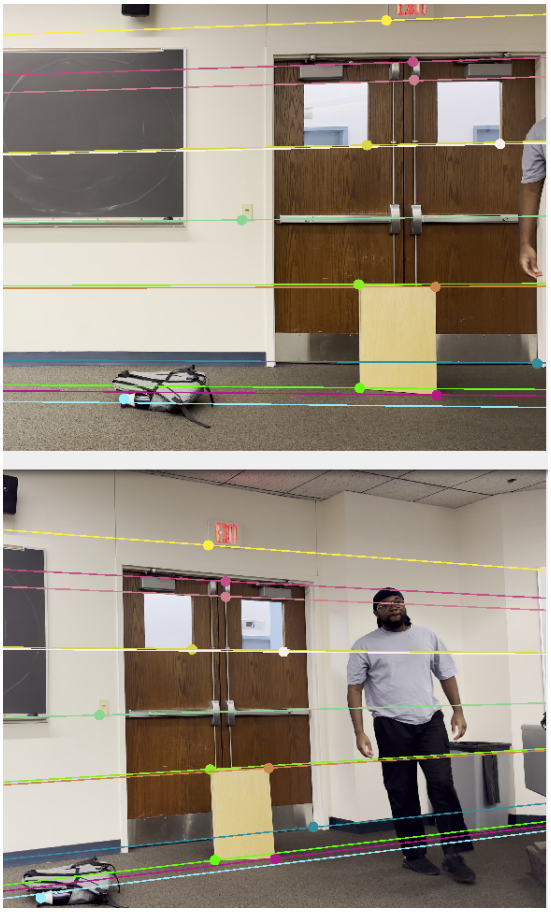


Fig. 2: Fundamental Matrix Estimation

4) Tracking: Figures 4 and 5 show the results of using YOLO to perform simultaneous tracking in 2 video feeds in a stereo rig. The path that a target has travelled in the video is denoted by a trail behind the target, and the pixel coordinates of the center of the tracked target is stored and used to perform triangulation.

5) Point Triangulation and Velocity Estimation: Figures 6 and 7 show the calculations of the target's velocity using our method, separated into x, y, z components, and the magnitude of the speed velocity. In Figure 7, we can observe a dip in the target's velocity, this is due to the target tripping over an object in the video, reducing their speed, although they did pick up speed again.

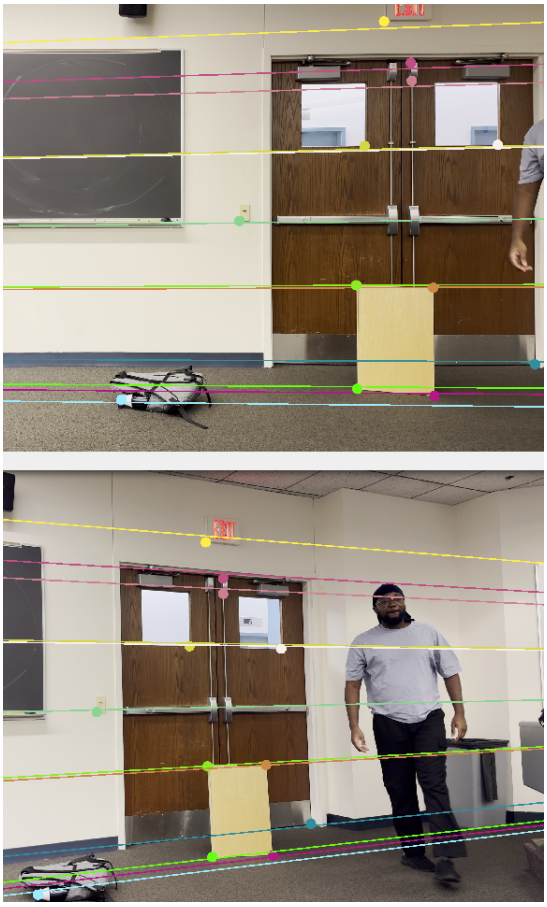


Fig. 3: Essential Matrix Estimation

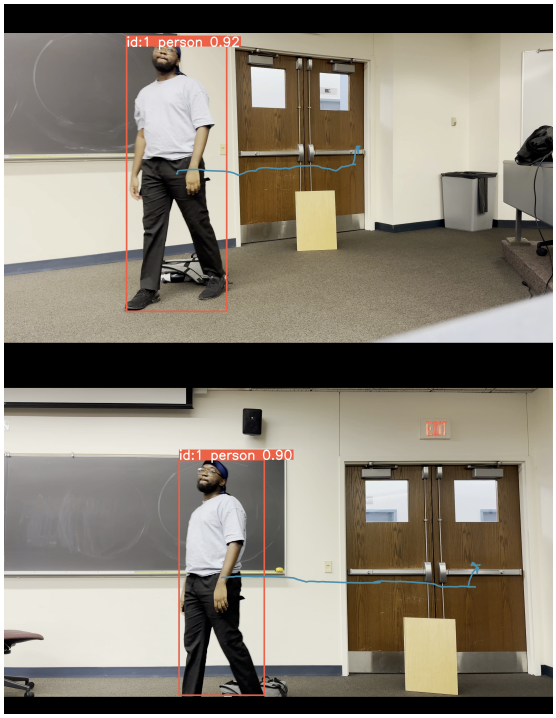


Fig. 4: Tracking the trajectory of a walking target in a stereo rig. The trail behind the target shows the path the target has taken in the image

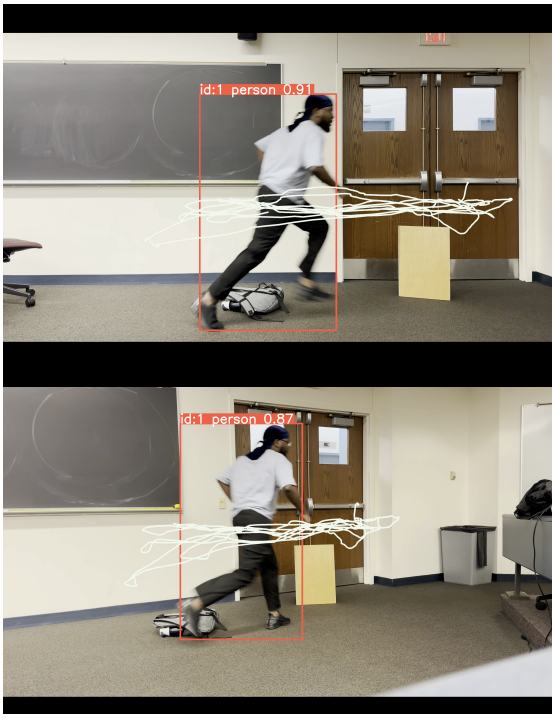
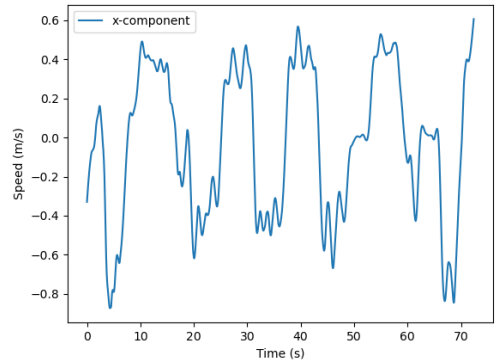
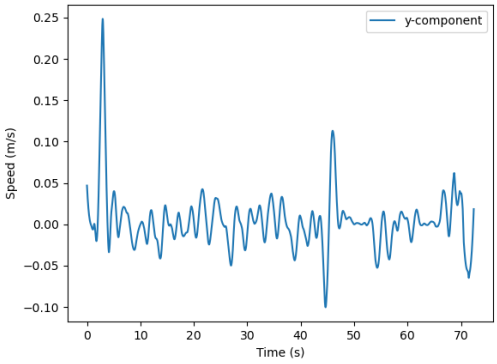
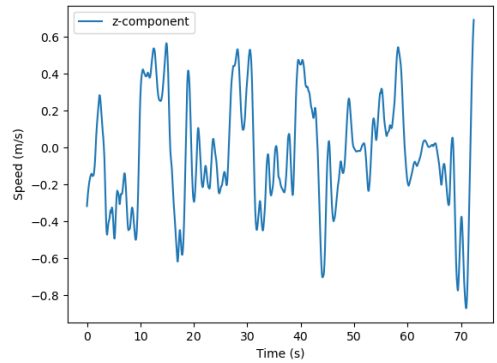


Fig. 5: Tracking the trajectory of a running target in a stereo rig. The trail behind the target shows the path the target has taken in the image







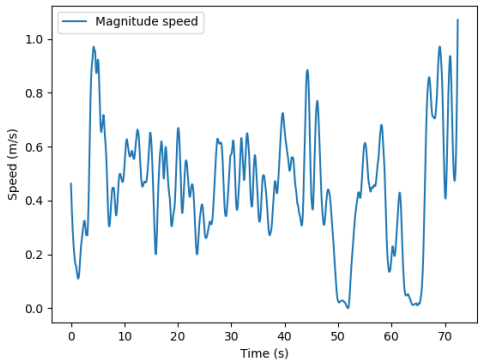
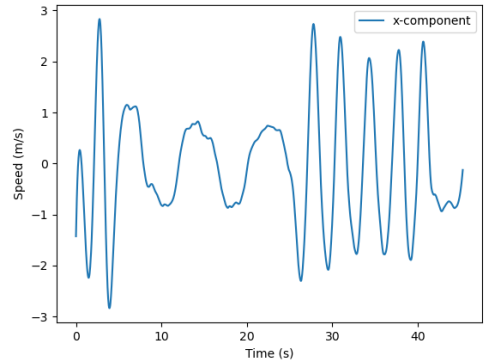
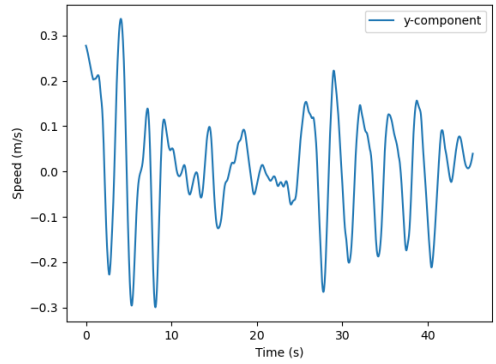
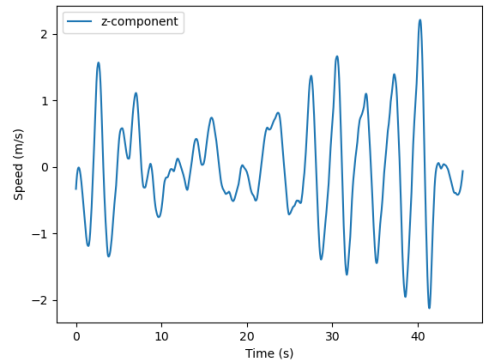


Fig. 6: Graphs showing the calculated x, y, z velocity components of the target in the test video with the target walking, and the Eucledian norm of all three velocity components







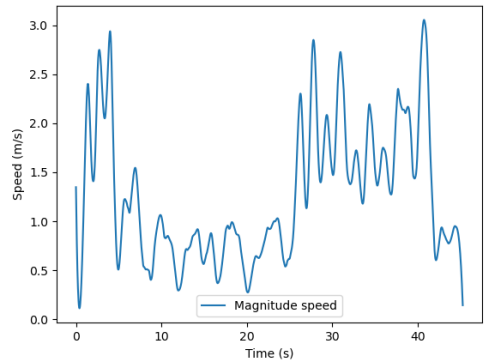


Fig. 7: Graphs showing the calculated x, y, z velocity components of the target in the test video with the target running, and the Eucledian norm of all three velocity components

V. Conclusion

The research presented in this paper demonstrates a robust system for the detection, tracking, and velocity estimation of objects using stereo vision techniques with statically mounted cameras. Our approach effectively integrates advanced methods in camera calibration, motion segmentation, feature extraction, and object tracking to achieve precise measurement of object positions and velocities in real-world coordinates.

The system's efficacy was validated through a series of controlled experiments, ranging from tracking a single slow-moving person to monitoring high-speed objects like a plastic ball in flight. The experiments confirmed the system's capability to accurately estimate the position and velocity of various moving targets. Notably, the implementation of YOLOv8 and ByteTrack algorithms provided reliable and continuous tracking across the dual-camera setup, essential for maintaining object identification and trajectory consistency.

Our calibration process, utilizing a standard chessboard and leveraging the intrinsic capabilities of iPhone cameras, enabled precise alignment of the stereo camera setup. This calibration was crucial for reducing optical distortions and enhancing the accuracy of triangulated 3D points. Furthermore, the application of the Savitzky-Golay filter smoothed the positional and velocity data, resulting in more consistent and reliable measurements.

This project contributes to the field of surveillance and motion analysis by providing a detailed methodology for setting up and calibrating a dual-camera system capable of performing high-precision object tracking and velocity estimation. The system's flexibility and accuracy suggest its applicability in various practical scenarios, including but not limited to security surveillance, sports analytics, and traffic monitoring.

For future enhancements, we plan to explore the integration of real-time data processing technologies to reduce latency and increase the throughput of the system, making it viable for more dynamic and uncontrolled environments. Additionally, expanding the dataset to include more diverse scenarios and object types will help improve the robustness and adaptability of the tracking algorithms.

In conclusion, the successful implementation of this stereo vision system not only underscores the potential of statically mounted cameras in complex surveillance tasks but also sets the groundwork for further innovations in the field of computer vision and real-time object tracking.

References

[1] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023, [Online]. Available: https://github.com/ultralytics/ultralytics. Version: 8.0.0. [Accessed: April 23, 2024].

[2] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," arXiv preprint arXiv:2110.06864, 2022.

[3] A. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures," Analytical Chemistry, vol. 36, no. 8, pp. 1627-1639, 1964. [Online]. Available: https://doi.org/10.1021/ac60214a047.

[4] Longuet-Higgins, H., A computer algorithm for reconstructing a scene from two projections. Nature 293, 133–135 (1981). https://doi.org/10.1038/293133a0

[5] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge: Cambridge University Press, 2004.