University of Zurich[UZH]

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Identity-verified Gesture Recognition based on Time-of-Flight Sensors

Jorge Alejandro Ortiz Valerio
Zurich, Switzerland
Student ID: 23-743-958

Supervisor: Prof. Dr. Bruno Rodrigues, Prof. Dr. Burkhard Stiller
Date of Submission: March 15th, 2026

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Master Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

Smart indoor environments increasingly rely on natural, touch-free interfaces to enable seamless Human-Computer Interaction (HCI). Gesture recognition has emerged as an intuitive approach for triggering actions in such spaces. However, conventional RGB-based systems raise two fundamental concerns: privacy, since color cameras capture personally identifiable visual information, and access control, since gesture commands can be issued by any person in the scene regardless of their authorization.

This thesis proposes a privacy-conscious, identity-verified gesture recognition system for indoor smart environments. A single Time-of-Flight (ToF) depth camera mounted in an overhead position provides continuous depth sensing without capturing color or texture information. An Inertial Measurement Unit (IMU) worn on the wrist of the authorized person serves as a passive identity token. The system fuses depth-based person tracking with IMU motion signals to determine which tracked agent corresponds to the authorized user, enabling gesture commands to be accepted exclusively from that individual without requiring prior user enrollment or storing any kind of personal information of users.

The processing pipeline encompasses depth-based foreground segmentation, morphological filtering, blob detection, Kalman filter tracking, confidence-based IMU-vision fusion for authorization, IMU-driven gesture segmentation, and Dynamic Time Warping (DTW) template matching for gesture classification. The system supports both single-agent and multi-agent scenarios.

The system is evaluated on a dataset of 57 recorded scenarios across 11 participants. Offline gesture recognition achieves 87.0% overall accuracy with a micro F1-score of 0.906 and near-perfect precision of 0.997, confirming reliable rejection of out-of-distribution movements. In single-agent online scenarios, authorization achieves an F1-score of 0.951 (accuracy: 93.0%) with 99.7% of authorized time attributed to the correct agent. In multi-agent scenarios, authorization achieves an F1-score of 0.723 (accuracy: 73.3%) and close to 80% of authorized time attributed to the correct agent, with the reduction attributed primarily to transient false authorizations caused by simultaneous motion of multiple tracked agents. Across all scenarios, 88.7% of total authorized time is correctly attributed to the ground-truth agent. Online gesture precision is 1.000 across all scenarios, with recognition latency averaging 2.97 s end-to-end.

The results demonstrate the feasibility of privacy-preserving, identity-aware gesture interaction using a single ToF camera and a wearable IMU, without relying on RGB imagery or prior biometric registration.

The implementation is publicly available at:
`https://github.com/JorgeOrtizV/identity-verified-gesture-rec`

Zusammenfassung

Intelligente Innenraumumgebungen verlassen sich zunehmend auf natürliche, berührungsfreie Schnittstellen, um eine nahtlose Mensch-Computer-Interaktion (Human-Computer Interaction, HCI) zu ermöglichen. Die Gestenerkennung hat sich dabei als intuitiver Ansatz etabliert, um Aktionen in solchen Umgebungen auszulösen. Herkömmliche RGB-basierte Systeme werfen jedoch zwei grundlegende Probleme auf: zum einen den Datenschutz, da Farbbildkameras visuelle Informationen erfassen, die eine persönliche Identifizierung ermöglichen, und zum anderen die Zugriffskontrolle, da Gestenbefehle von jeder im Bild befindlichen Person ausgeführt werden können, unabhängig von ihrer Autorisierung.

Diese Arbeit schlägt ein datenschutzbewusstes, identitätsverifiziertes Gestenerkennungssystem für intelligente Innenraumumgebungen vor. Eine einzelne Time-of-Flight (ToF) Tiefenkamera, die in einer Überkopfposition montiert ist, ermöglicht eine kontinuierliche Tiefenerfassung, ohne Farb- oder Texturinformationen aufzuzeichnen. Eine Inertial Measurement Unit (IMU), die am Handgelenk der autorisierten Person getragen wird, dient als passives Identitätstoken. Das System kombiniert tiefenbasierte Personenverfolgung mit IMU-Bewegungssignalen, um zu bestimmen, welcher verfolgte Akteur dem autorisierten Benutzer entspricht. Dadurch können Gestenbefehle ausschließlich von dieser Person akzeptiert werden, ohne dass eine vorherige Benutzerregistrierung erforderlich ist oder personenbezogene Daten der Benutzer gespeichert werden.

Die Verarbeitungspipeline umfasst tiefenbasierte Vordergrundsegmentierung, morphologische Filterung, Blob-Erkennung, Tracking mittels Kalman-Filter, eine konfidensbasierte IMU-Vision-Fusion zur Autorisierung, IMU-gestützte Gestensegmentierung sowie Dynamic Time Warping (DTW)-Template-Matching zur Gestenklassifikation. Das System unterstützt sowohl Einzelpersonen- als auch Mehrpersonen-Szenarien.

Das System wird anhand eines Datensatzes mit 57 aufgezeichneten Szenarien von 11 Teilnehmenden evaluiert. Die Offline-Gestenerkennung erreicht eine Gesamtgenauigkeit von 87.0% mit einem Micro F1-Score von 0.906 und einer nahezu perfekten Präzision von 0.997, was eine zuverlässige Zurückweisung von Bewegungen außerhalb der gelernten Verteilung bestätigt. In Online-Szenarien mit einer einzelnen Person erreicht die Autorisierung einen F1-Score von 0.951 (Genauigkeit: 93.0%), wobei 99.7% der autorisierten Zeit dem korrekten Akteur zugeordnet werden. In Mehrpersonen-Szenarien erreicht die Autorisierung einen F1-Score von 0.723 (Genauigkeit: 73.3%) und nahezu 80% der autorisierten Zeit werden dem korrekten Akteur zugeordnet. Die Verringerung ist hauptsächlich auf kurzzeitige Fehl-Autorisierungen zurückzuführen, die durch gleichzeitige Bewegungen mehrerer verfolgter Akteure verursacht werden. Über alle Szenarien hinweg werden 88.7% der gesamten autorisierten Zeit korrekt dem Ground-Truth-Akteur zugeordnet. Die Online-Gestenpräzision beträgt in allen Szenarien 1.000, bei einer durchschnittlichen End-to-End-Erkennungslatenz von 2.97 s.

Die Ergebnisse demonstrieren die Machbarkeit einer datenschutzwahrenden, identitätsbewussten Gesteninteraktion mithilfe einer einzelnen ToF-Kamera und einer tragbaren IMU, ohne auf RGB-Bilddaten oder eine vorherige biometrische Registrierung angewiesen zu sein.

Die Implementierung ist öffentlich verfügbar unter:
`https://github.com/JorgeOrtizV/identity-verified-gesture-rec`

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Dr. Bruno Rodrigues and to the Communications Systems Group led by Prof. Dr. Burkhard Stiller for their continuous guidance, feedback, and support throughout this project. I also thank Tridonics AG for providing the necessary hardware that made this work possible.

I am deeply grateful to all participants who dedicated their time to the data collection sessions, without whom the evaluation of this system would not have been possible.

Finally, I thank my family and friends for their unconditional support, encouragement, and motivation throughout these years.

Contents

Abstract i

Zusammenfassung iii

Acknowledgments v

1 Introduction 1

1.1 Motivation 1

1.2 Thesis Goals 2

1.3 Methodology 3

1.4 Thesis Outline 3

2 Fundamentals 5

2.1 Background 5

2.1.1 RGB Cameras and Privacy Concerns 5

2.1.2 Time-of-Flight Cameras 6

2.1.3 Used Technologies 8

2.2 Related Work 10

2.2.1 Indoors Tracking and Depth Sensing Introduction 10

2.2.2 ToF for gesture recognition 11

2.2.3 Person identification and Time-of-Flight cameras as a privacy-compliant option 13

2.2.4 Discussion 15

3 Design **19**

3.1 System Overview 19

3.2 Hardware 21

3.2.1 Time-of-Flight Camera: Espros cam660 21

3.2.2 IMU Sensor 24

3.3 Platform Architecture 25

3.4 Preprocessing and Foreground Segmentation 25

3.5 Person Tracking 29

3.6 Identity Verification and Authorization 33

3.7 Gesture Segmentation 39

3.8 Gesture Recognition 41

4 Implementation **45**

4.1 Development Environment 45

4.2 System Utilities 48

4.3 Preprocessing 48

4.3.1 Preprocessing and Foreground Segmentation 48

4.3.2 Tracking 55

4.4 IMU-Vision fusion and authority selection 59

4.4.1 Reading data from IMU 59

4.4.2 IMU-Vision Fusion 61

4.5 Gesture Segmentation and Recognition 65

4.6 Visualizations 68

5 Evaluation **71**

5.1 Experimental Setup . . . 71

5.1.1 Recording Environment . . . 71

5.1.2 Data Collection . . . 72

5.1.3 Data Annotation . . . 76

5.1.4 Rosbag Playback Methodology . . . 77

5.2 Evaluation Metrics . . . 77

5.2.1 Authorization Evaluation Metrics . . . 77

5.2.2 Gesture Recognition Evaluation Metrics . . . 78

5.3 Offline Evaluation of the Gesture Recognition System . . . 78

5.4 Online System Evaluation . . . 80

5.4.1 Metrics Infrastructure . . . 80

5.4.2 Single-Agent Scenarios . . . 81

5.4.3 Multi-Agent Scenarios . . . 82

5.4.4 Combined Results . . . 83

5.5 Discussion . . . 85

5.5.1 Offline Gesture Recognition . . . 85

5.5.2 System Performance . . . 86

5.5.3 Single-Agent Performance . . . 87

5.5.4 Multi-Agent Performance . . . 88

5.5.5 Overall System Performance . . . 91

5.5.6 Improving Gesture Recognition Performance - Creation of a dataset for gesture recognition on an overhead perspective . . . 91

5.5.7 Privacy Analysis . . . 92

5.5.8 Limitations . . . 92

5.5.9 Comparison to Related Work . . . 93

6 Final Considerations **97**

6.1 Summary 97

6.2 Conclusions 98

6.3 Future Work 99

Bibliography **100**

Abbreviations **107**

List of Figures **107**

List of Tables **111**

List of Algorithms **113**

A System ROS Graph **117**

B ESPROS TOFcam660 calibration parameters **119**

C Repository Directory Structure **121**

D Tracking Node Parameters **123**

E Preprocessing Node Parameters **125**

F *tof_preprocessing* custom message definitions **127**

F.1 Blob 127

F.2 BlobArray 127

F.3 AgentMsg 127

F.4 AgentArray 128

G *mpu* custom message definitions **129**

G.1 Conf 129

G.2 ConfArray 129

H IMU Reader parameters **131**

I Fusion Node Parameters **133**

J Gesture Node Parameters **135**

Chapter 1

Introduction

1.1 Motivation

Smart environments such as homes, offices, or classrooms are increasingly adopting natural user interfaces to enable seamless, touch-free interaction with technology. Among these, gesture recognition has emerged as an intuitive and touch-free form of communication between humans and machines. Typical applications range from controlling lighting and multimedia systems to interactive displays and industrial automation.

Traditionally, gesture recognition systems rely on RGB cameras, leveraging rich visual information for detecting objects, poses, and gestures through computer vision algorithms. However, using RGB cameras for monitoring and surveillance of closed rooms like homes or offices raise privacy concerns where occupants may feel surveilled, exposed, or have identifiable visual details captured and stored without consent [7].

Time-of-Flight (ToF) cameras present a compelling alternative from a privacy-preserving perspective. Instead of capturing detailed visual imagery, ToF cameras construct depth maps by measuring the time it takes for an emitted light signal to bounce back from surfaces [13, 15]. This produces depth maps that represent the distance of objects from the sensor, preserving the geometric structure of the scene while omitting identifiable facial or clothing details. Such characteristics make ToF-based sensing particularly suitable for privacy-conscious gesture interfaces in smart-room environments.

Depth-sensing has already shown success in applications like environment mapping [45], gesture and pose recognition [17]. Prior studies also indicate that even low-resolution depth images can capture enough information for activity recognition while maintaining anonymity [7]. Furthermore, incorporating ToF data has been shown to improve tracking robustness in multi-person scenarios compared to RGB-only approaches [24].

In a typical smart office scenario [17], multiple people might be present, but only one or a few should be authorized to control environment settings through gesture commands, for example, turning lights on and off or adjusting AC temperature. This adds the complexity of correctly identifying the authorized person while recognizing gestures, and triggering

actions only when an authorized agent executes them. Multi-user presence introduces ambiguity: if two people perform similar gestures, a naive system could misinterpret an unauthorized user's movement as a valid command.

Recent research has demonstrated that gesture dynamics can be used as a biometric form of identity verification [70], motivating the idea of identity-verified gesture recognition: combining gesture detection with user identification so that only a specific person's gestures are recognized as commands. A key desirable property for such a system is that the identity verification is passively determined. The authorized user should be recognized without any prior enrollment, stored appearance models, or explicit registration step. This registration-free approach is essential to alleviate privacy concerns inherent to an identity-aware gesture recognition system, which implies active monitoring of indoor spaces.

The overarching aim of this thesis is therefore to investigate whether passive, identity-aware gesture recognition can be realized in a smart-room environment using privacy-preserving sensing, and to design, implement, and evaluate a concrete system that demonstrates this possibility.

1.2 Thesis Goals

The central goal is to investigate whether passive, identity-aware gesture recognition can be realized in a smart-room environment using privacy-preserving ToF depth sensing, and to design, implement, and evaluate a concrete system that demonstrates this possibility. To achieve this, the research pursues the following specific objectives.

First, it aims to develop a conceptual framework for passive identity-aware gesture recognition. This includes assessing whether ToF depth data alone is sufficient to passively identify an authorized user without prior enrollment. In case depth-only passive identification proves insufficient, the framework investigates the integration of a complementary sensing modality that enables registration-free authorization without storing personal biometric data.

Second, the framework will be implemented as a real-time processing pipeline, emphasizing low latency and modular software design. The entire processing workflow, from person detection and tracking to gesture classification and user authorization, operates locally to preserve privacy and eliminate network dependencies on external servers.

Finally, the system is quantitatively and qualitatively evaluated. The evaluation assesses gesture recognition accuracy, authorization performance, and end-to-end response time. A qualitative analysis examines privacy implications, discussing how the use of privacy-preserving depth sensing mitigates visual exposure risks while balancing identification reliability and anonymity.

1.3 Methodology

This research follows an hypothetico-deductive approach combining theoretical analysis with experimental validation. A comprehensive literature review is conducted to identify existing methods for gesture recognition, depth-based sensing, and privacy-preserving human-computer interaction, establishing the theoretical foundation and research gap. Based on these insights a proof-of-concept prototype is developed. The solution integrates depth-based gesture recognition with passive identity verification using a ceiling-mounted Time-of-Flight (ToF) depth camera and a wrist-worn Inertial Measurement Unit (IMU). The prototype serves to empirically evaluate whether privacy-preserving, identity-aware interaction can be achieved without relying on RGB imagery. Quantitative metrics such as recognition accuracy, false-acceptance and false-rejection rates, and latency, together with qualitative analysis of privacy implications, provide the evidence necessary to assess the proposed hypothesis.

1.4 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces the fundamental technical concepts relevant to this work and presents a comprehensive review of related research in gesture recognition, depth sensing, and privacy-preserving human-computer interaction, including re-identification approaches. Chapter 3 describes the conceptual design of the proposed system, outlining the architecture, data flow, and key algorithmic components. Chapter 4 details the implementation of the framework, including hardware setup, software modules, and optimization strategies for real-time performance.

Chapter 5 presents the experimental setup and a comprehensive evaluation of the proposed system using quantitative and qualitative metrics. Finally, Chapter 6 summarizes the conclusions drawn from the research, discusses practical insights and limitations, and outlines directions for future work.

Chapter 2

Fundamentals

2.1 Background

2.1.1 RGB Cameras and Privacy Concerns

RGB cameras capture visible light through a color filter array, most commonly using a Bayer BGGR pattern (cf. Fig. 2.1) that is sensitive to the primary colors, red, green, and blue (hence we refer to them as RGB cameras) [59]. In this arrangement, the number of green pixels is doubled compared to red or blue, since luminance is primarily defined by the green channel and human vision is more sensitive to spatial variations in luminance [59].

Once the light from the scene reaches the camera, this is reflected onto the image sensor through a lens system (Figure 2.2) [59]. The image sensor (array of photo-diodes) measure the pixel intensity of the incoming light filtered by the color array. Then, the missing color values are calculated by interpolation. This process is commonly known as demosaicing [59]. As a result, RGB cameras are able to produce color representations that closely resemble human perception.

However, because RGB cameras record detailed visual information, they can also capture uniquely identifiable characteristics of individuals such as facial features, clothing, and

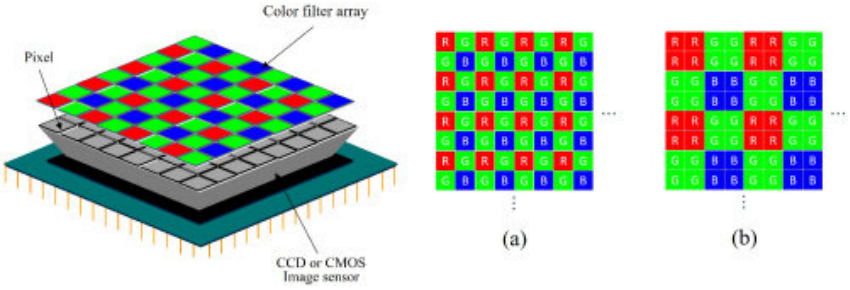


Figure 2.1: a) Bayer Filter; b) Quad Bayer Filter [28]

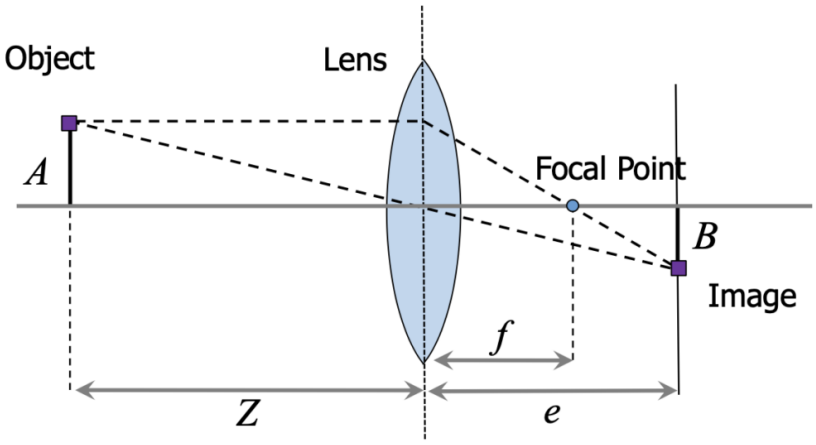


Figure 2.2: Thin Lens Equation [56]

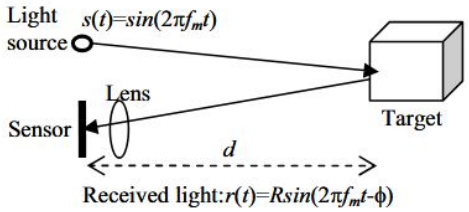
personal belongings. When such cameras are deployed in indoor environments for continuous monitoring, significant privacy concerns arise. It becomes necessary to establish clear boundaries on how visual data is collected, stored, and processed to avoid violating individual privacy rights. Within the European Union, the General Data Protection Regulation (GDPR) provides the legal framework governing this type of data processing. The GDPR enforces principles of transparency, purpose limitation, data minimization, and proportionality, as well as strict rules regarding storage, duration, employee notification, and access rights [69].

2.1.2 Time-of-Flight Cameras

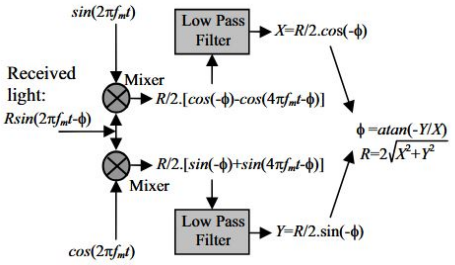
To address privacy concerns while still enabling intelligent sensing in smart environments, alternative sensing modalities have been explored. Time-of-Flight (ToF) cameras represent a privacy-conscious solution that captures depth information rather than full-color imagery. Since ToF sensors record only geometric distances rather than detailed visual textures, they can effectively capture human shape and motion while concealing personal appearance, thus enhancing user anonymity. It is true that it is also possible to obtain depth information using traditional cameras, for instance, using stereo vision or depth-from-X techniques [59]. However, the privacy issue still stands since traditional cameras capture personal identifiable features.

A ToF camera operates by emitting modulated light (typically infrared) and measuring the time or phase shift required for the reflected signal to return to the sensor. From this delay, the distance to each point in the scene is computed. In continuous-wave (CW) modulation systems, distance is estimated from the phase difference between the emitted and received signals, while in pulse-based systems it is derived from the direct travel time of light [15]. The sensor array consists of pixels capable of detecting phase information, from which depth and intensity maps are generated. Unlike stereo vision systems, ToF cameras do not require correspondence matching between multiple viewpoints and can thus provide dense, real-time depth measurements [13]. Common technologies based on

this principle include LiDAR and radar, which operate at different wavelength ranges but share the same underlying concept. Figure 2.3 illustrates the ToF working principle and distance calculation.



(a) ToF working principle [15]



(b) ToF distance calculation diagram [15]

Figure 2.3: Illustration of the Time-of-Flight (ToF) principle and phase-based distance calculation.

Despite their advantages, ToF cameras are not immune to environmental and technical limitations. Measurement accuracy can be affected by lighting conditions, surface reflectivity, and sensor noise. Moreover, phase-based systems can suffer from aliasing, where multiple distances produce the same measured phase shift. This issue can be mitigated by using multiple modulation frequencies, though this approach may introduce motion-related artifacts such as blur. Increasing frame rate or applying edge detection techniques can reduce these effects [15]. Overall, the system's robustness depends on both hardware stability and signal-processing strategies that compensate for such artifacts.

Although this thesis adopts ToF sensing, it is instructive to briefly compare it against other sensing modalities considered in the literature, since their respective trade-offs inform the design decisions made in Chapter 3. Beyond Time-of-Flight cameras, several alternative technologies have been explored for gesture recognition and indoor user monitoring. For example, Millimeter-Wave Radar (mmWave). mmWave sensors operate by emitting radio-frequency (RF) signals and measuring their reflections from objects in the environment [75]. Due to their working principle, mmWave can sense objects through certain materials, unlike optics-based sensors [75]. Fang et al [10] used mmWave radars to capture hand and arm motion, transforming the temporal signals into the frequency domain (Fast Fourier Transform) and classifying gestures with neural networks. Zhao et al [75] further demonstrated that human identification is possible by clustering radar point clouds and analyzing spatiotemporal movement characteristics with LSTM networks.

Other optics-based depth alternatives rely on ambient or reflected light variations rather than active depth measurement. Venkatnarayan et al [67] proposed a passive light-sensing system using an array of photo-diodes embedded in the floor. Gestures are recognized by analyzing changes in ambient light caused by the user's movements and classifying them with a support vector machine (SVM). Similarly, Yu et al [73] developed an infrared-based sensing system that infers gestures from reflected IR light signals, using k-nearest neighbors (KNN) and SVM classifiers on processed reflection patterns.

Despite the promising capabilities of these alternative technologies, ToF cameras remain highly suitable for the scope of this thesis. Although mmWave radars are robust to oc-

clusions and lightning conditions, typically they are very noisy and produce sparse point clouds with limited spatial resolution, which can hinder fine-grained gesture interpretation and precise hand trajectory analysis. RF-based sensing also raises concerns around privacy leakage through wall penetration and inadvertent detection beyond a room's boundaries, which is an undesirable property for a controlled indoor environment. Light-based sensing solutions, on the other hand, often require dense arrays of sensors to cover even small areas, resulting in substantial hardware installation effort and limited scalability [67]. Moreover, these sensors capture only indirect illumination changes rather than explicit geometric information, which restricts them to coarse gesture categories in restricted environments. In contrast, ToF cameras deliver dense, structured depth maps with consistent spatial resolution, enabling reliable gesture extraction while preserving privacy by excluding color and texture information. Their compact form factor, real-time performance, and established use in indoor human-computer interaction make ToF sensors a practical and balanced choice for privacy-conscious gesture-based access control systems.

2.1.3 Used Technologies

This section introduces the core algorithmic and sensing technologies that form the building blocks of the proposed system. Each is described at a conceptual level to give the reader sufficient background before the detailed design is presented in Chapter 3.

2.1.3.1 Image preprocessing and Foreground Segmentation

Image preprocessing is a common computer vision technique where an input raw image is transformed into a simplified representation while keeping meaningful information, saving this way computation time and achieving better performance. Common techniques include resizing, smoothing (noise reduction), thresholding, and morphological filtering [59]. Feature extraction identifies meaningful structures in the image, such as blob boundaries, centroids, or bounding boxes, that provide a compact, discriminative representation for downstream processing, reducing the computational burden compared to operating on raw pixel data [59].

Background subtraction is a standard technique for detecting moving objects in a static scene. A background model is first estimated from a set of frames in which no foreground objects are present. In subsequent frames, each pixel is classified as foreground if its value deviates significantly from the background model, and background otherwise [3]. Several background modeling strategies exist: frame differencing is the simplest approach, while running average or median methods accumulate statistics over multiple frames to produce a more robust estimate. Adaptive models can further update the background over time to account for gradual scene changes such as lighting drift.

In depth-image-based systems, background subtraction operates on distance values rather than color intensities. A pixel is labeled as foreground when its measured depth is meaningfully smaller than the background depth (i.e., the object is closer to the sensor than the background surface). This characteristic makes depth-based segmentation inherently more robust to illumination variation than color-based approaches.

2.1.3.2 Morphological Image Processing

Morphological operations are a family of image processing techniques that operate on the shape and structure of image regions. They are applied to binary images (such as foreground masks) using a structuring element, a small kernel that defines the local neighborhood [16, 61].

The two fundamental morphological operations are *erosion*, which shrinks foreground regions by removing pixels at their boundaries, and *dilation*, which expands them. These operations are commonly combined into compound operations: *opening* (erosion followed by dilation) removes small isolated noise blobs, while *closing* (dilation followed by erosion) fills small holes and gaps within foreground regions. In person detection from depth images, opening is typically applied to suppress sensor noise and spurious foreground pixels, while closing is used to consolidate the depth silhouette of a person into a single connected region, especially when the depth image contains shadow artifacts or thin gaps between body parts [16, 61].

2.1.3.3 Connected-Component Labeling

Connected-component labeling assigns a unique label to each distinct foreground region in a binary image [16]. Two pixels belong to the same connected component if they are both foreground and connected through a path of neighboring foreground pixels (using 4- or 8-connectivity). The result partitions the foreground mask into separate blobs, each representing a candidate object in the scene.

In person-tracking pipelines, connected-component labeling is applied after morphological post-processing to isolate individual person silhouettes. Each component can then be characterized by geometric properties such as bounding box, area, centroid, and height, which serve as features for downstream tracking and identification.

2.1.3.4 Kalman Filter

The Kalman filter is an optimal recursive state estimator for linear dynamic systems subject to Gaussian noise [25]. It operates in two alternating steps: *prediction*, in which the state is projected forward in time using a motion model, and *update*, in which the predicted state is corrected using a new measurement. The filter maintains a state estimate (e.g., position and velocity of a tracked person) together with an associated covariance matrix that quantifies estimation uncertainty.

In multi-person tracking, a Kalman filter is assigned to each tracked agent. When a new depth frame arrives, the filter predicts each agent's likely position and a nearest-neighbor assignment matches predictions to detected blobs. If no matching detection is found for a given filter (e.g., due to brief occlusion or segmentation failure), the filter continues to propagate the state estimate using only the motion model, allowing temporary loss of detection to be bridged without discarding the tracked identity.

2.1.3.5 Inertial Measurement Unit (IMU)

An Inertial Measurement Unit (IMU) is a sensor that measures linear acceleration (via an accelerometer) and angular velocity (via a gyroscope), typically along three orthogonal axes [71]. By integrating these signals over time, the IMU can estimate changes in velocity and orientation. IMUs are compact, wearable, and entirely self-contained, making them well-suited for capturing limb motion without any external infrastructure. In gesture recognition applications, IMUs mounted on the wrist capture the dynamic movement patterns of arm and hand gestures. A key property of IMU-based sensing is its independence from the visual scene: it produces a reliable motion signal regardless of occlusions, lighting conditions, or camera perspective.

2.1.3.6 Dynamic Time Warping

Dynamic Time Warping (DTW) is an elastic similarity measure for comparing two time series that may differ in length or exhibit local temporal distortions [55]. Unlike Euclidean distance, which requires a rigid one-to-one correspondence between time steps, DTW finds the optimal alignment between sequences by warping the time axis. This is achieved by computing a cost matrix over all pairwise distances between signal values and finding the minimum-cost monotonic path from the start to the end of both sequences.

DTW is particularly well-suited for gesture recognition, where the same gesture performed by different users or at different speeds produces time series of varying durations and tempos. To control computational complexity and prevent pathological alignments, the Sakoe-Chiba band constraint restricts the warping path to a diagonal corridor of fixed width [55]. In template-based classification, an incoming gesture sequence is compared against stored reference templates using DTW distance, and the gesture class with the nearest template is selected as the recognized gesture.

2.2 Related Work

2.2.1 Indoors Tracking and Depth Sensing Introduction

Indoor human tracking plays a key role in smart environments, enabling natural Human-Computer Interaction (HCI) by supporting applications such as people counting and occupancy monitoring [2, 11, 17, 24, 62, 64], activity understanding [7, 49], and gesture recognition (Section 2.2.2). Traditional systems rely on networks of RGB cameras, which offer rich visual information but introduce challenges related to visible-light changes, partial occlusions, and privacy concerns.

To improve robustness, early research combined RGB cameras with range sensing techniques. Guomundsson et al [17] demonstrated that augmenting an RGB camera network with a Time-of-Flight camera improves foreground-background segmentation and enables

real-time 3D scene reconstruction. This two-modal fusion leveraged depth information to enhance indoor tracking accuracy under challenging lightning and cluttered environments.

After a successful integration of ToF cameras in RGB-based tracking solutions, research shifted towards ToF-only solutions for indoor tracking. In the field of robotics and autonomous navigation, Nakagawa et al [45] used depth sensing for real-time mapping surfaces in indoor environments. Multiple authors have researched the use of ToF sensors for people tracking in indoor environments. Early systems primarily relied on background-foreground segmentation [3]. Jia et al [24] combined classical pre-processing and a Markov model for pose labeling and person tracking using depth images. Oprisescu et al [50] used ToF cameras for action recognition based on silhouette extraction and motion cues. Bevilacqua et al [2] performed people detection using a simple background-foreground segmentation and tracking based on connected components. Additionally, a Kalman filter was used in the case of occlusions. Similarly, Stahlschmidt et al [62] developed a depth-based person detection and tracking pipeline using Kalman filtering, demonstrating applications in people counting, occupancy density estimation, and crowd flow analysis. Tamas et al [64] developed a person recognition and tracking for automatic door-control and people counting through fine-tuning (transfer learning) of a RGB trained joint 2D-3D reconstruction network. Fernandez et al [11] implemented a people detection based on a feature vector obtained from depth measurements and a classification algorithm using Principal Component Analysis (PCA) and SVM. Finally, a particle filter was used for tracking. Other authors, like Tomasi et al [14] or Meers et al [42] focused on head detection and tracking using time-of-flight cameras by locating specific features of human face such as the nose tip.

While depth-based tracking systems demonstrate strong performance for tasks such as person localization, occupancy estimation, and motion analysis, tracking alone is insufficient for interactive smart-room applications. In many indoor environments, only specific individuals should be able to trigger actions, and system commands must be issued intentionally through explicit gestures rather than mere presence or movement. Therefore, this thesis focuses on two higher-level capabilities built on top of depth sensing: gesture recognition and depth-based person identification. The goal is to enable a privacy-conscious gesture-controlled smart-room system where only authorized users can activate functions, ensuring both intuitive interaction and access control without relying on RGB imagery.

2.2.2 ToF for gesture recognition

Gestures are expressive, meaningful body movements involving of the fingers, hands, arms, head, face, or full-body, often used to express information or interact with the environment [43]. Gestures can be static and dynamic. Static gestures represent fixed poses or configurations of the body captured in a single image frame, while dynamic gestures consist of a motion trajectories that evolve over time between defined start and end points [23, 43]. From a user experience perspective, static gestures tend to be less intuitive and require the user to memorize specific poses [23].

This thesis focuses primarily on dynamic hand, arm, and body gestures. Earlier systems relied on specialized hardware, such as instrumented gloves equipped with accelerometers

and flex sensors. However, the reliance on expensive hardware and the restriction of natural hand movement motivated the development of vision-based solutions [23]. Traditional computer vision techniques, including feature extraction, object detection, clustering, and classification, have been successfully used for gesture recognition. In recent years, machine learning (ML) and deep neural networks have been increasingly employed to enhance robustness and generalization, enabling real-time and user-independent gesture recognition systems [43].

The use of hand gestures is a natural way for humans to communicate, commands, intentions, actions, and emotions [50]. Gesture recognition enables intuitive HCI and plays an important role in smart-room applications such as contactless control interfaces and sign language translation. While gesture recognition from RGB images is a mature and well-established technology, its reliance on appearance cues introduces sensitivity to lightning, background clutter, and privacy risks.

To address these limitations, early work introduced Time-of-Flight (ToF) cameras as a complementary modality to enhance RGB systems' performance. Oprisescu et al. [50] combined depth data with RGB input to improve segmentation in cluttered scenes, achieving more robust gesture recognition than RGB-only approaches. Similarly, Takahashi et al. [63] used grayscale images together with depth information to extract keypoint trajectories (via Harris corner detection [20] and Lucas-Kanade tracking [34]), forming 4-D motion descriptors which were classified using an Support Vector Machine (SVM).

More recent work has explored ToF-based gesture recognition as a standalone solution. Depth sensing simplifies segmentation by eliminating factors such as color, lightning, reflection, and shadows while also reducing privacy concerns compared to RGB [44]. Additionally, they represent a robust solution to poor lightning conditions and high-speed motion situations [53].

In most ToF gesture recognition systems, the hand is assumed to be the closest object to the sensor, allowing segmentation by depth thresholding [5, 29, 44, 66, 74]. Uebersax et al. [66] performed palm localization and orientation after thresholding to identify gestures, while Molina et al. [44] extracted geometric features (e.g., number of finger maxima, angles, amplitude) and classified gestures via SVM. Kollorz et al. [29] used x- and y- projections of segmented depth data to classify gestures using a nearest-neighbor classifier.

However, threshold-based segmentation may fail when the hand is not the closest object to the sensor or when the gesture involves complex motion. To address this Kapuscinski et al. [26] proposed the use of Viewpoint Feature Histograms (VFH) which capture surface geometry and viewpoint, enabling robust nearest-neighbor classification without relying on proximity assumptions.

With the increasing success of deep learning, convolutional architectures have also been applied to ToF gesture recognition due to their feature learning and data fitting capabilities. One approach is to see gesture recognition as a multiclass classification problem. Ma, et al.[39] used depth measurements and a ResNet-50 model, reporting an accuracy of 98.5% on static hand gestures classification. For dynamic gestures, Gomaa et al [53] used a Convolutional Neural Network - Long Short Term Memory (CNN-LSTM) architecture

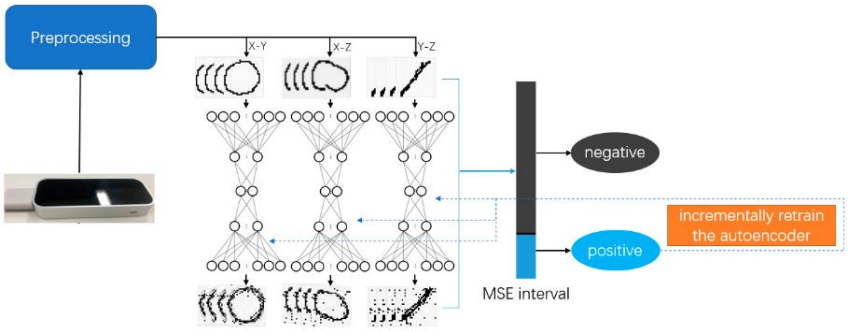


Figure 2.4: Authentication Procedure based on decomposition of Gesture movement in three planes [70].

to perform dynamic gesture classification inside a vehicle cockpit. After background-subtraction to isolate hand region, the CNN was used to extract features, then temporal modeling was done by the LSTM, and final classification was performed via a fully connected network. Incremental learning was additionally used to improve per-user performance.

Recent work suggests that gestures themselves can be used for biometric authentication. Wang, et al [70] developed a novel gesture-based authentication system using depth sensors. They argue that humans have specific habits and muscle memory, making their motion patterns difficult to imitate and therefore suitable as a unique fingerprint for authentication. A One-Class Classification (OCC) auto-encoder network is used for decision making. The network is initially trained by asking the user to perform the gesture a couple of times. The training set is then expanded through data augmentation. The model is continuously trained with incremental learning as the user interacts with the system (See Figure 2.4).

2.2.3 Person identification and Time-of-Flight cameras as a privacy-compliant option

Humans possess characteristics that make them uniquely or singularly identifiable, such as fingerprints, facial features [35], or even movement patterns [70], as described in 2.2.2. Earlier, facial identification was typically performed using classical computer vision techniques based on feature extraction and matching. Notable examples include Eigenfaces [65] and keypoint-based methods such as the Scale-Invariant Feature Transform (SIFT) [23, 33], Oriented FAST and Rotated BRIEF (ORB) [54], and Random Sample Consensus (RANSAC) [12]. These methods rely on detecting distinctive geometric or intensity features that can be compared against previously stored templates. Modern face recognition systems have largely transitioned to deep learning approaches, such as FaceNet [57], which learn compact feature embeddings through contrastive or triplet-loss training. These learned representations enable robust face matching and verification and have become widely adopted in everyday applications, such as smartphone authentication.

A problem of performing gesture recognition without user verification is that they could be triggered by any user knowing the gesture and, thus, making the system unreliable [23].

Solutions, such as [23] have developed successful user identification and gesture recognition using RGB cameras. However, smart environments rely on continuous sensing to enable seamless HCI. Constant visual monitoring in indoor spaces is often perceived as intrusive and raises privacy concerns, particularly when high-resolution RGB cameras are used [7, 24].

These concerns have motivated the adoption of depth-only sensing for indoor monitoring, since depth maps omit color and texture information and inherently reduce identifiable visual detail. Time-of-Flight sensors have been successfully used in different indoors monitoring applications while providing enhanced privacy to their users, e.g. people detection and tracking [2, 11, 14, 17, 24, 42, 50, 62, 64]. A particularly illustrative example is Chou et al. [7], who deployed depth sensors in hospital environments to monitor hygiene compliance and ICU activity while safeguarding the privacy of patients and staff. To further minimize privacy risk, the authors employed low-resolution depth sensing, applied private super-resolution via a Deep CNN Skip Connection and Network (DCSCN) before action recognition, and used public datasets disjoint from deployment data for model training.

Although ToF sensing conceals appearance attributes such as color, clothing, texture, and facial detail, it does not guarantee full anonymity. Depth images still encode geometric information, such as body shape, height, motion patterns, and coarse facial structure. These details may still be used to identify individuals under certain conditions. Accordingly, research has explored person re-identification from depth data alone as well as from combined RGB-D input.

Depth-only re-identification. Several works demonstrate re-identification using only depth images [6, 36, 37, 42, 72]. Cheng et al. [6] used Kinect 3D depth scans to extract 2D facial contour maps and match them to stored identity templates. Similarly, Meers et al [42] proposed people recognition by using face-prints, which contain depth information of a face in an spherical area surrounding the nose tip. Wu et al. [72] proposed a rotation-invariant Eigen-Depth descriptor for body-shape-based matching. Sanchez et al. [36] employed overhead depth cameras and geometric descriptors for multi-person re-identification in non-overlapping fields of view. Luna et al. [37] proposed a lightweight heuristic descriptor to rapidly match frontal depth images on low-power hardware.

RGB-D re-identification. When both modalities are available, fusing depth with RGB improves recognition accuracy [1, 31, 38]. Liciotti et al. [31] used an overhead RGB-D sensor and showed improved performance when depth complemented color information. Albiol et al. [1] performed re-identification by creating a body print using both depth and color information. Luna et al. [38] proposed the use of an averaged descriptor based on body features through different frames to improve robustness before matching.

Cross-modal depth-RGB re-identification. Recent work investigates whether depth images can be matched to RGB images of the same person. Lui et al. [32] explored cross-modal matching using CNNs. The first approach was fine-tuning LightCNN with RGB and depth images following a supervised learning objective to obtain the ID of the test subject the images belonged to. Similarly they trained a CNN using RGB-depth image pairs to identify if a RGB-depth image pair belonged to the same person [32]. Hafner et al. [18] used a two-network approach based on ResNet-50. The first network performed

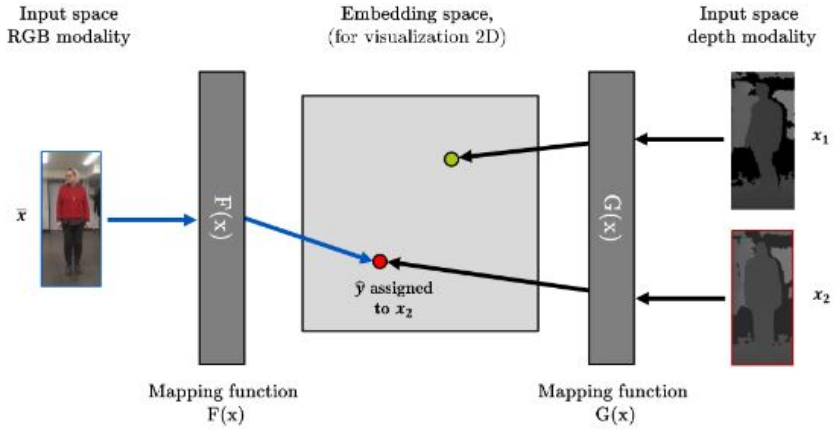


Figure 2.5: Depth-RGB re-identification [18]

single modal re-identification and a second identical network was used as a feature extractor of the second modality (color images if depth images were used for step I or vice versa). Figure 2.5 illustrates the process in further detail. Key was weight transfer from the first network to the second network to create a shared embedding space for RGB and depth inputs [18].

2.2.4 Discussion

The reviewed literature illustrates the versatility of Time-of-Flight (ToF) cameras for a wide range of indoor monitoring applications, such as, people tracking, gesture recognition, and person identification. Table 2.1 shows a summary of the most relevant tracking approaches. It is possible to notice that ToF-based systems tend to use an overhead setting. This choice is motivated primarily by its robustness to occlusions. If the sensor is placed above the scene, interactions between users, objects, and environment rarely block the line of sight, enabling consistent detection and tracking of individuals. Overhead positioning also simplifies background-foreground segmentation, due to the largely uniform and static background (in most cases). The main drawback of this setting is the reduced field of view, often requiring multiple sensors for full room coverage, but the trade-off is generally favorable for tracking stability in multi-person environments.

Table 2.2 shows that gesture recognition normally takes advantage of a frontal viewpoint. This is a logical design decision since a frontal placement captures the most information of hand poses, finger articulation, and arm motion. It also simplifies segmentation by assuming the hand is the closest object to the sensor (not always the case). However, this positioning reintroduces occlusion issues in multi-user settings and is less compatible with room-level monitoring, where users are not expected to face a specific direction.

ToF cameras have also proven effective for gesture recognition across a wide range of feature extraction and classification techniques. Traditional approaches rely on depth thresholding, silhouette segmentation, geometric descriptors, or hand-shape features, while recent work employs deep learning techniques, such as, CNNs or LSTMs. Yet, notably,

Authors	Camera Position	Camera Model	Application
Stahlschmidt et al [62]	Overhead	Not mentioned	People detection and tracking
Bevilacqua et al [2]	Overhead	Canesta Electronic Perception EP205	People detection and tracking
Jia et al [24]	Overhead	SwissRanger SR4000	People detection, tracking, and pose estimation
Fernandez et al [11]	Overhead	Kinect v2	People detection and tracking, classification
Chou et al [7]	Overhead	Kinect v2	Hygiene monitoring, activity recognition
Guomundsson et al [17]	Inclined	RGB + SwissRanger SR3100	People tracking
Tamas et al [64]	Inclined	Analog Devices (ADI) ToF CCD	People detection and counting
Opricescu et al [49]	Frontal	SwissRanger SR3000	Action recognition
Tomasi et al [14]	Frontal	Canesta ToF depth sensor	Head tracking
Meers et al [42]	Frontal	SwissRanger SR3000	Face recognition

Table 2.1: Summary of ToF sensing applications for indoors monitoring

none of these works employ an overhead viewpoint for hand-gesture interpretation, leaving open the question of whether reliable gesture recognition can be achieved under this more tracking friendly but gesture-ambiguous geometry.

Table 2.3 highlights the paradigm of person identification and re-identification using depth data. Many depth-only identification systems rely on frontal viewpoints, extracting biometric cues from facial geometry or upper-body shape. Such approaches cannot transfer to an overhead configuration because the head and face are not visible or are captured only as top-down silhouettes. Overhead identification therefore requires alternative cues. One option is to rely on global body-shape descriptors. Another option is to consider dynamic cues-motion patterns, gesture execution styles, or spatiotemporal trajectories, as demonstrated by Wang et al [70]. However, these options require storing personal identifiable data and previously setting up the system with the authorized agents and modifying it when changes in the authorized group, for example, someone leaving or joining the organization. A third option is to incorporate a complementary sensing modality such as Radio Frequency (RF) or IMU, in such a way that this additional modality would provide a passive identification way by just wearing a small sensor when in the smart room. These observations indicate that identification under overhead sensing remains underexplored, especially when constrained to privacy-preserving depth-only input.

Although prior work has demonstrated that person re-identification is possible using depth-only [6, 36, 37, 72], RGB-D [1, 31, 38], and even cross-modal depth-RGB information [18, 32], this does not imply that ToF-sensing should be discarded as a privacy-

Authors	Camera Position	Camera Model	Application
Oprisescu et al [50]	Frontal	Kinect	Static hand gesture recognition
Takahashi et al [63]	Frontal	Kinect	Dynamic hand gesture recognition
Molina et al [44]	Frontal	SwissRanger SR4000	Static and dynamic hand gesture recognition
Uebersax et al [66]	Frontal	MESA SR4000	Sign language recognition
Zahedi et al [74]	Frontal	Kinect	Sign language recognition
Breuer et al [5]	Frontal	SR2	Dynamic gesture recognition
Kollorz et al [29]	Frontal	Photonic-Mixer-Device (PMD)	Dynamic gesture recognition
Kapuscinski et al [26]	Frontal	SwissRanger SR4000	Static gesture recognition
Ma et al [39]	Frontal	VL53L5	Static gesture recognition (ResNet50-based)
Gomaa et al [53]	Frontal	BMW Natural User Interface	Dynamic gesture recognition (LSTM-based)
Wang et al [70]	Frontal	LeapMotion	Dynamic gesture recognition; user identification through gestures

Table 2.2: Summary of ToF sensing applications for gesture recognition.

conscious option. Depth cameras inherently remove texture, color, facial detail, and background information, significantly reducing direct identifiability. Furthermore, successful cross-modal re-identification [18, 32] typically requires paired RGB-depth training data, often collected under controlled settings with consistent viewpoints, environments, and subject poses.

In practical deployments where only depth sensors are installed, such paired multi-modal supervision is not available, and an adversary cannot trivially train a model to link ToF images to RGB footage from other cameras. To the best of our knowledge, no work has demonstrated reliable cross-modal matching between unpaired depth images and arbitrary RGB imagery of the same individuals across different scenes, viewpoints, or cameras. Achieving this would require an extremely generalizable biometric embedding that can bridge modalities without paired training data.

Therefore, ToF remains a privacy-enhancing sensing modality aligned with data minimization principles: it enables functional interaction while reducing the exposure of identifiable visual information. Nonetheless, it is important to underline that privacy is not absolute, depth data still contains geometric cues that can enable restricted forms of re-identification. These observations motivate the development of interaction systems that

Authors	Camera Position	Camera Model	Application
Meers et al [42]	Frontal	SwissRanger SR3000	Depth-only identification using face-prints descriptors
Wang et al [70]	Frontal	LeapMotion	Identification through gestures
Cheng et al [6]	Frontal	Kinect	Depth-only re-identification based on face features
Luna et al [37]	Frontal	Kinect	Depth-only re-identification using geometric descriptors
Lui et al [32]	Frontal	Not specified	Cross-modal re-identification
Sanchez et al [36]	Overhead	Kinect v2	Depth-only re-identification using geometric descriptors
Liciotti et al [31]	Overhead	Xtion Pro Live	RGB-D re-identification using KNN
Luna et al [38]	Overhead	Kinect v2	RGB-D re-identification using body-pose descriptors
Wu et al [72]	Inclined	Kinect	Depth-only re-identification using pose descriptors
Hafner et al [18]	Inclined	Kinect	Cross-modal re-identification using contrastive learning
Albiol et al [1]	Multiple	Kinect	RGB-D re-identification using body-prints

Table 2.3: Summary of ToF sensing applications for person identification.

deliberately use depth-only sensing while acknowledging and mitigating its privacy limitations rather than relying on RGB input. This thesis builds on this perspective by enabling gesture-based control with user authorization using ToF data, without introducing additional identity-revealing sensors such as RGB cameras.

Taken together, the tables show that: (i) ToF is robust and widely used for overhead tracking, (ii) gesture recognition research almost exclusively assumes a frontal viewpoint, (iii) person identification using depth typically depends on frontal biometric information or controlled movement patterns, and (iv) ToF is a technology used for privacy enhancing applications. This thesis aims to bridge these four research directions by developing a gesture-based interaction system that operates from an overhead viewpoint, unifying the robustness of overhead tracking with the expressiveness of gesture recognition, while ensuring that only authorized users can trigger actions in a privacy-conscious manner. The design of the proposed system is presented in Chapter 3.

Chapter 3

Design

3.1 System Overview

In smart rooms monitoring and gesture recognition systems camera positioning is a central design decision to achieve optimal system performance. Frontal camera placement is the most common choice in the gesture recognition literature [5, 26, 29, 39, 44, 53, 63, 66, 70, 74], as it provides rich visual cues for hand pose analysis, including finger articulation and relative hand orientation. However, this thesis adopts an overhead camera configuration for several reasons. An overhead camera is robust against inter-person occlusions, requires an unobtrusive fixed installation, and allows users to interact naturally from anywhere in the room without having to position themselves in front of a specific viewpoint to perform a gesture. Furthermore, overhead depth sensing directly benefits from an extensive and well-validated body of research in indoor monitoring, including person tracking and occupancy estimation [2, 11, 24, 62], activity monitoring [7], and person re-identification [31, 36, 38], all of which have been demonstrated under this configuration. This design decision allows us to research in this thesis the feasibility of overhead positioning for gesture recognition, since a top-down perspective does introduce a tradeoff: hand pose and finger articulation are not as distinguishable as in a frontal configuration, and even less distinguishable when using a Time of Flight sensor.

Although ToF cameras are more privacy-preserving than RGB cameras, prior research has shown that depth-based person identification remains feasible through template matching of stored biometric profiles, whether extracted from body shape [1, 6, 18, 31, 32, 36–38, 42, 72] or from personalized gesture-execution patterns [70]. While these approaches could in principle support an identity-aware gesture recognition system, they require prior enrollment of each authorized user and the retention of personal identifiable data, which conflicts with the privacy-preserving intent of the system. No existing research has demonstrated passive person identification from depth data alone without pre-stored user templates. Furthermore, even if such passive identification were to be achievable, it would require the authorized user to exhibit consistently distinctive movement patterns that no other person in the scene replicates. A fragile assumption in dynamic multi-user environments that introduces a spoofing risk: an adversary aware of the authorized user's movement patterns or behavior could attempt imitation to gain unauthorized access.

To avoid storing any personal information and enable a passive identity-verified gesture recognition system it is proposed the use a second sensor worn by the authorized user. This provides an additional information modality complimentary to the depth perception, which allows the system to take a more confident authorization decision. Wireless localization alternatives, such as Radio Frequency (RF) tags or Bluetooth Low Energy (BLE) beacons, could in principle allow the token's position to be triangulated relative to the camera, but they require additional fixed receiver infrastructure and their accuracy degrades in multi-camera deployments.

A wrist-worn IMU was therefore selected as the complementary sensing modality: it is self-contained, requires no additional infrastructure, and produces continuous motion measurements that can be correlated with the depth-derived displacement of tracked agents to passively identify the IMU wearer. In addition, the IMU provides precise gesture onset and offset information as well as a detailed three-axis record of arm trajectory, which can be used directly for gesture classification without requiring depth information to recognize a gesture. In this way the gesture recognition tradeoff of the overhead camera configuration can be overcome.

The IMU does introduce a known limitation. Although it operates at high frequency and provides accurate short-term measurements, position estimates derived from integrating acceleration accumulate drift over time due to sensor bias [60, 71]. IMU-vision fusion algorithms can mitigate this drift, but they require a calibrated shared reference frame between the camera and the IMU, which does not exist in the overhead configuration adopted here, making loop-closure-based error correction infeasible [27, 46]. The IMU is therefore used exclusively for motion correlation and short-duration gesture trajectory extraction, operating directly on raw acceleration and angular velocity signals without any integration step, avoiding this way the drift accumulation tradeoff.

The proposed system thus combines two complementary sensing modalities: a Time-of-Flight depth camera mounted in an overhead position monitors the scene, providing a continuous stream of depth images from which persons are detected, segmented, and tracked, and a wrist-worn IMU carried by the authorized user. IMU motion signals are used both to establish the correspondence between the IMU wearer and a tracked depth blob to determine authorization and to drive gesture segmentation and classification. Only the authorized user can trigger recognized gesture commands. All other persons present in the scene are tracked but their gestures are silently ignored, reducing unnecessary computation.

Figure 3.1 illustrates the high-level architecture of the system.

The processing pipeline consists of five sequential stages:

1. **Preprocessing and Foreground Segmentation.** A background model is estimated from the first frames of the depth stream. Each subsequent depth frame is compared against this model to produce a binary foreground mask, which is refined by morphological operations and partitioned into individual person blobs via connected-component labeling [3, 16, 24, 61].

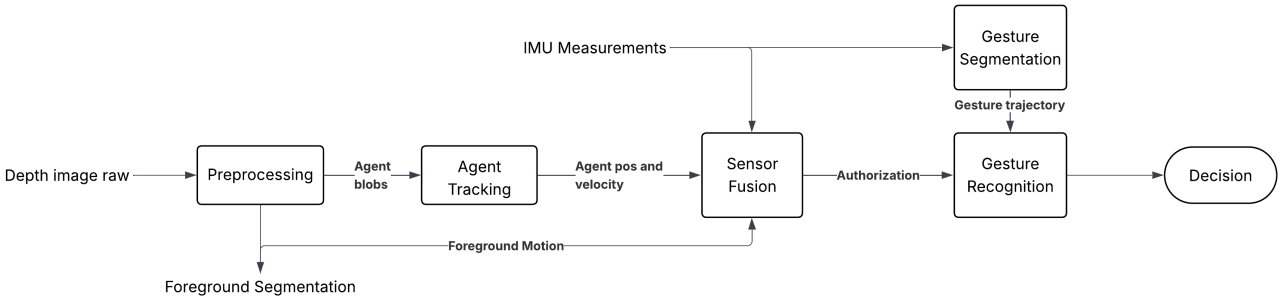


Figure 3.1: Overview of system architecture design

2. **Agent Tracking.** Each foreground blob is matched to an existing tracked agent using spatial proximity. A Kalman filter [25] maintains the state estimate (position and velocity) of each agent across frames and bridges temporary detection gaps caused by occlusions, segmentation failures, or flickering.

3. **Identity Verification and Authorization: IMU-vision fusion.** The IMU worn by the authorized person produces a continuous motion signal. The system correlates this signal with the motion patterns of tracked agents to accumulate a confidence score for each agent. The agent whose motion best matches the IMU signal is declared the authorized agent and receives a positive authorization flag.

4. **Gesture Segmentation.** A state machine running on the IMU signal detects the onset and offset of a gesture. When sustained arm motion above a threshold is detected, the state transitions from *IDLE* to *ACTIVE*, and the 3D trajectory of the agent's motion is recorded. When motion falls below the threshold, the accumulated trajectory is passed for classification.

5. **Gesture Recognition.** The recorded trajectory is compared against stored gesture templates using Dynamic Time Warping (DTW) with a Sakoe-Chiba band constraint [55]. The gesture class whose nearest template yields the smallest DTW distance below a rejection threshold is published as the recognized gesture command, provided the agent was authorized at the time the gesture began.

3.2 Hardware

3.2.1 Time-of-Flight Camera: Espros cam660

The depth sensing component of the system is the *ESPROS TOFcam660* Time-of-Flight camera [9], which can be observed in Figure 3.2. It is a continuous-wave phase-based ToF sensor operating in the near-infrared range, producing per-pixel depth maps at a resolution of 320px × 240px and up to 160 frames per second. The sensor also outputs an amplitude (intensity) image that reflects signal strength and could be used for quality filtering of depth measurements, though this work uses only the raw distance image.

Figure 3.2: ESPROS TOFcam660

As discussed in Section 3.1, the camera is mounted in an overhead position looking downward onto the scene floor. Figure 3.3 provides an example diagram of this setup in an office environment. This placement minimizes inter-person occlusions and provides a largely static background (the floor and room furniture) simplifying foreground segmentation. As a direct consequence of this viewpoint, hand pose and finger articulation are not visible from depth images, so gesture sensing is delegated to the wrist-worn IMU described in Section 3.7. Although the TOFcam660 sensor can provide different data formats (see figure 3.5). It was decided to use only the distance image raw for the development of this project, since it provides the required depth map, satisfy the requirements for privacy enhancing underlined previously (Section 2.1.2), and computation is less expensive than using point clouds. Considering the used setting, figure 3.4 displays an example of what a raw distance image looks like in this identity-aware gesture recognition system.

Despite their advantages, Time-of-Flight cameras are inherently noisier than other depth sensing technologies such as structured light or stereo vision, and this has direct implications on system design. The most immediate effect is per-pixel depth jitter: individual measurements fluctuate frame-to-frame even for static scenes, causing person centroids derived from foreground blobs to vary slightly between consecutive frames. This noise motivates the use of a Kalman filter for tracking (Section 3.5), which smooths the state estimate by fusing noisy centroid measurements with a constant-velocity motion model prediction [13, 62, 63].

A second limitation is the appearance of *flying pixels* at depth discontinuities [13]. At the boundary between a person and the background, some pixels receive a mixed reflected signal from both surfaces, producing intermediate and physically meaningless depth values. These artifacts manifest as noisy fringes along the silhouette boundary, fragmenting the foreground mask. Morphological post-processing (Section 3.4) is applied to suppress isolated noise and consolidate the body silhouette into a single coherent region. Additionally, continuous-wave ToF sensors are susceptible to multi-path interference: in indoor environments with reflective floors and walls, the emitted IR signal can reach a pixel after

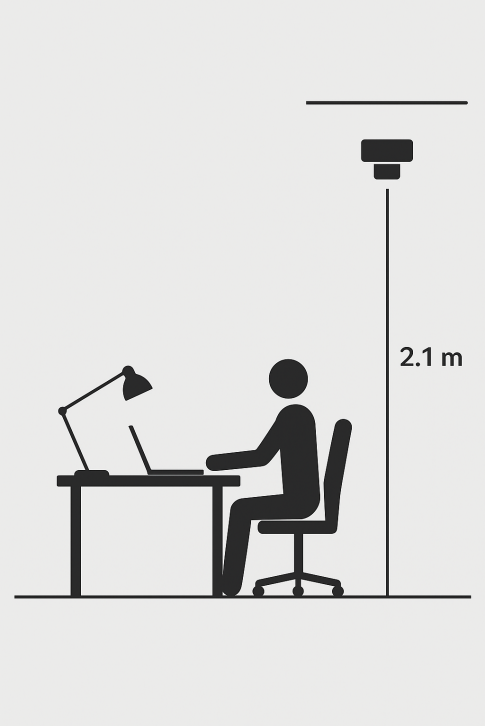


Figure 3.3: Overhead camera positioning in an office environment

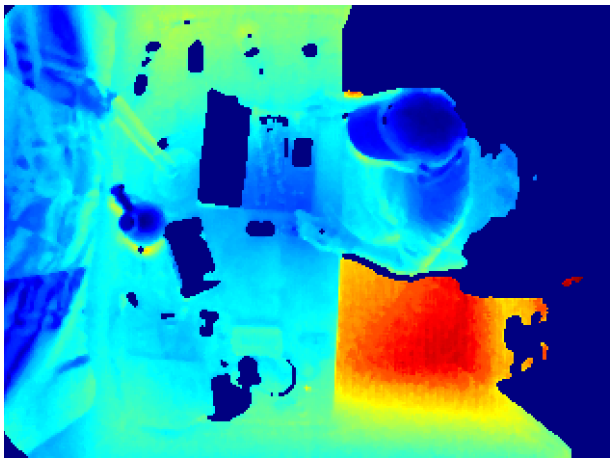
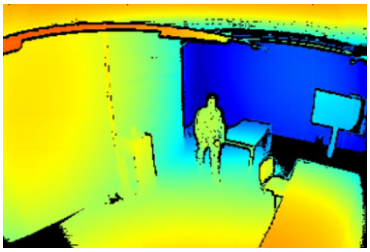
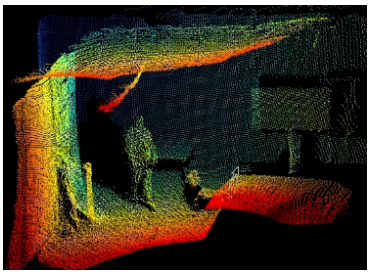


Figure 3.4: Camera point of view - Overhead configuration for this identity-aware gesture recognition system

(a) Grayscale amplitude image



(b) Color coded raw distance image



(c) Point cloud

Figure 3.5: Example of data acquirable using ESPROS TOFcam660 [9]

reflecting off multiple surfaces, introducing systematic depth errors [13]. This effect is amplified when multiple people are in close proximity, as each body partially redirects IR illumination toward neighboring sensor regions, increasing depth noise in crowded scenes.

3.2.2 IMU Sensor

An Inertial Motion Unit (IMU) sensor based on the MPU-6050 [22] was selected to provide a complementary motion modality to the system. This specific sensor measures three-axis linear acceleration and three-axis angular velocity, providing a 6-dimensional observation of the user's wrist movement. As motivated in Section 3.1, the IMU serves a dual role in the system. First, it acts as a passive identity token: the authorized person wears the device, and the system uses its motion signal to identify which tracked depth blob corresponds to that person, establishing authorization without any stored biometric profile. Second, it is the primary gesture sensing modality: arm trajectories reconstructed from IMU signals are used directly as input to the DTW classifier, bypassing the ambiguity of top-down depth-based gesture interpretation.

The selection of a 6-axis sensor is critical for the proposed fusion logic. The accelerometer data allows for translational correlation between inertial acceleration and visual agent speed, while the gyroscope data enables rotational correlation between angular velocity and foreground pixel changes. Furthermore, the availability of both modalities is necessary to satisfy the requirements of the Madgwick orientation filter [40] used to perform gravity compensation, essential for isolating pure linear motion from the effects of gravity.

To avoid limiting the natural movements of the user, a wireless communication channel was prioritized to transmit measurements from the wristband to the system server. The design adopts a minimal embedded architecture comprising a power source, the MPU-6050 sensor, and a microcontroller to ensure the device remains unobtrusive and non-invasive during extended periods of interaction.

3.3 Platform Architecture

Since the system is expected to handle large amounts of information received at high frequency (depth images and IMU measurements) and to perform authorized agent selection and gesture recognition in real time while preserving software modularity, ROS (Robot Operating System) [47] is a natural design platform. Communication in ROS is handled with a publish-subscribe message-passing architecture that decouples producers from consumers, allowing each processing stage to be developed, tested, and replaced independently.

Furthermore, ROS enables the synchronous recording of high-frequency multi-modal data through the use of rosbags. This is a central design requirement for this thesis, as it allows for the construction of a unique dataset comprising agents interacting in a multi-user environment while performing gestures under an overhead configuration. By fusing depth maps and IMU measurements into a single time-stamped archive, the system supports repeatable offline experimentation and ensures that the fusion and recognition algorithms are validated against identical, real-world data sequences, and provides a foundation stone for future work under this setup.

More details about implementation using ROS can be found in Section 4.

3.4 Preprocessing and Foreground Segmentation

The preprocessing node receives a raw depth frames acquired by the TOFcam660. Each of them has to be first cleaned before any preprocessing is applied due to the known limitations of ToF sensors [13]. Depth readings outside the valid sensor range are discarded and replaced with invalid markers, eliminating both near-range reflections from the sensor housing and far-range noise beyond the scene boundaries. The remaining valid depth values are then smoothed with a median filter [15, 16], which selects the median value in a ranked neighborhood, making it robust to impulse noise while preserving sharp depth discontinuities, which are critical for accurate foreground segmentation. Figure 3.6 shows the overall process flow of the image preprocessing stage for feature extraction, all the components are then described in detail during this section.

Background Model Initialization

Before foreground detection can begin, a background model of the empty scene must be established. During the initialization phase, the first N frames are collected and their per-pixel median is computed, ignoring invalid measurements. The median is preferred over the mean because it is robust to outlier depth readings caused by transient noise or partial sensor warm-up during the early frames. The result is a stable reference image of the static scene against which all subsequent frames are compared. Accumulating an initial buffer of N frames is a highly recommended practice for mitigating the inherent distance-proportional noise and invalid pixels of ToF sensors [2, 17, 24, 36, 38]. This

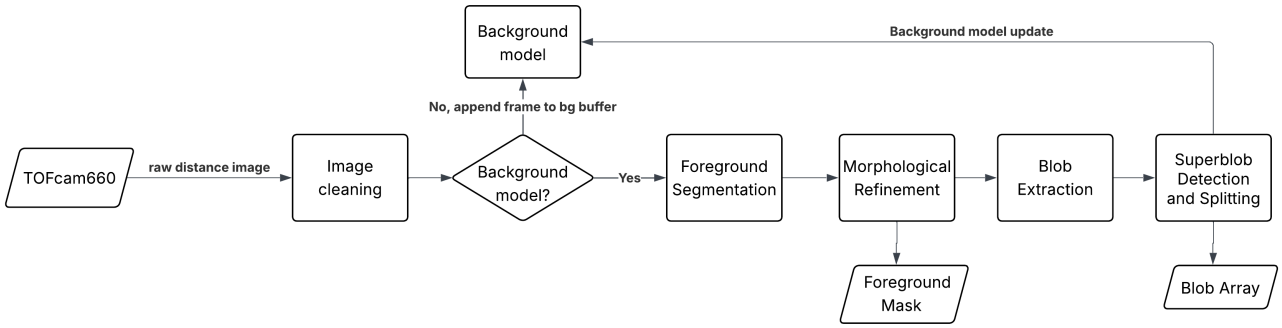


Figure 3.6: Diagram flow of the raw image preprocessing process to extract a foreground mask and region of interests (blobs)

initialization phase runs once at startup and requires the scene to be free of moving agents.

Foreground Segmentation

Once the background model is available, foreground detection is performed per-frame by computing the absolute per-pixel difference between the current cleaned depth image and the background model:

$$\text{diff}_t = |\text{depth}_t - \text{background}_t| \tag{3.1}$$

A pixel is classified as foreground if its difference exceeds a fixed threshold τ:

$$\text{fg}_t = \text{diff}_t > \tau \tag{3.2}$$

where invalid depth readings are treated as zero difference. Additionally, foreground pixels whose depth values exceed a secondary threshold are suppressed. This removes low-height objects in the scene, such as chairs or tables, that would otherwise produce valid foreground detections but do not correspond to the upper-body of a person (no more information than the one provided by the upper-body is needed). This background subtraction and thresholding approach is a standard technique in overhead depth-based tracking and monitoring systems [2, 17, 24].

Morphological Refinement

The raw binary foreground mask produced by depth thresholding typically contains noise artifacts due to sensor jitter and flying pixels. Two morphological operations are applied sequentially to refine the mask, an approach that has proven highly effective for isolating human silhouettes and mitigating artifacts in ToF depth maps [16, 24, 61]. First, an *opening* operation (erosion followed by dilation) removes small isolated foreground blobs that result from sensor noise, while leaving larger person-shaped regions intact. The kernel

size is chosen to preserve thin structural features such as arms, which occupy only a few pixels in the overhead depth image.

Second, a *closing* operation (dilation followed by erosion) is applied to fill internal gaps and holes within each person silhouette, which arise from shadow artifacts and regions of low signal amplitude. It is important to mention that the closing operation is applied per component rather than globally: each foreground region is processed independently within its bounding box. This prevents the closing operation from merging the silhouettes of two agents who are in close proximity, which would otherwise cause them to be detected as a single blob.

Blob Extraction and Matching

The refined foreground mask is then passed through connected-component labeling [16] to partition it into individual foreground regions, referred to as *blobs* [1, 2, 24]. A dual size filter is applied: very small blobs below a minimum pixel area are removed immediately as residual noise artifacts, while larger blobs below a second threshold are retained in the cleaned mask but excluded from agent tracking, since they are likely to correspond to small moving objects rather than persons. Blobs that pass both thresholds are considered candidate person detections and are characterized by their centroid and bounding box.

Each detected blob is then matched to an existing tracked agent using nearest-neighbor association based on centroid distance. If the closest previous track falls within a proximity threshold, the blob is assigned to that track and its state is updated. If no sufficiently close track is found, a new track is created. Tracks that receive no matching blob for several consecutive frames are removed after a timeout period, preventing ghost agents from persisting in the scene.

Each track additionally maintains a *static flag*, determined by monitoring centroid displacement and bounding box area changes over a sliding history window. A track is marked static when both its position and silhouette size have remained stable for a sufficient number of frames, indicating that the corresponding agent is not moving.

Superblob Detection and Splitting

The merging of closely interacting individuals into a single foreground detection is a recognized challenge in overhead ToF tracking, often referred to as the "touching objects" problem [1, 2, 24, 62]. When two agents move into close proximity, the morphological refinement may fail to keep their silhouettes separated, causing them to merge into a single large connected component, referred to as a *superblob*. This is detected by checking whether an unmatched blob is unusually large and has an elongated bounding box aspect ratio, suggesting that it contains more than one person.

When a superblob is detected, a splitting algorithm is triggered. The primary strategy uses the depth image to locate local maxima within the merged region (in overhead depth imagery, the heads of standing persons appear as local minima, which become maxima

when the depth map is inverted). These depth peaks are matched to previously known track centroids to establish seed points, and a marker-guided Watershed transform [1, 16] segments the superblob into sub-regions, one per agent. Using known track centroids as markers rather than applying the watershed directly prevents over-segmentation, as recommended by Gonzalez et al [16]. As a fallback strategy, superblobs can be split using the distance transform [16], which finds geometric peaks inside the binary silhouette. The boundary region between the two sub-blobs, which corresponds to the physical gap between the agents, is identified as a *bridge* and immediately force-updated into the background model to prevent it from persisting as a foreground artifact.

To improve robustness against superblobs, the tracking algorithm gives more importance to the motion model than to the ToF observations when a superblob is detected, a strategy analogous to the Kalman-prediction-based blob assignment used by Bevilacqua et al. [2] under merging conditions. This mechanism is described in further detail in Section 3.5.

Background Update

Since agents move through the scene over time, the background model must be updated continuously to account for gradual changes in the static environment. Pixels are eligible for background update if they are currently not classified as foreground, or if they have been consistently classified as foreground for long enough that the corresponding agent is considered to have become part of the static scene. This is tracked per-pixel through a *foreground age* counter: the age of a pixel increments each frame it remains foreground and resets when it transitions to background. When the age exceeds a threshold, the pixel is included in the background update mask even if the foreground segmentation still flags it as foreground. The mask of currently active tracked blobs is explicitly excluded from the update region to prevent the background model from absorbing agents who are temporarily stationary but still tracked. The background update region is then defined as follows,

$$\text{bg\_mask}_t = \left[\text{fg}_t = 0 \vee (\text{age}_t > \theta_{\text{age}} \wedge \neg\text{track\_mask}_t)\right] \wedge \text{valid}_t \tag{3.3}$$

The update itself is performed using an Exponential Moving Average (EMA) [17], giving higher weight to the accumulated background history than to the current observation. Only regions of the background included in the background update region *bg_mask* are updated:

$$\text{background}_t = \beta \cdot \text{background}_{t-1} + (1 - \beta) \cdot \text{depth}_t \tag{3.4}$$

Two update rates are used. Under normal conditions, a slow rate (β close to 1) is applied to track gradual lighting and sensor drift without over-fitting to moving agents. When all tracked blobs are simultaneously detected as static for an extended period, indicating a truly empty or fully stationary scene, an aggressive update rate (β significantly lower) is triggered to rapidly reclaim stationary foreground regions back into the background model.

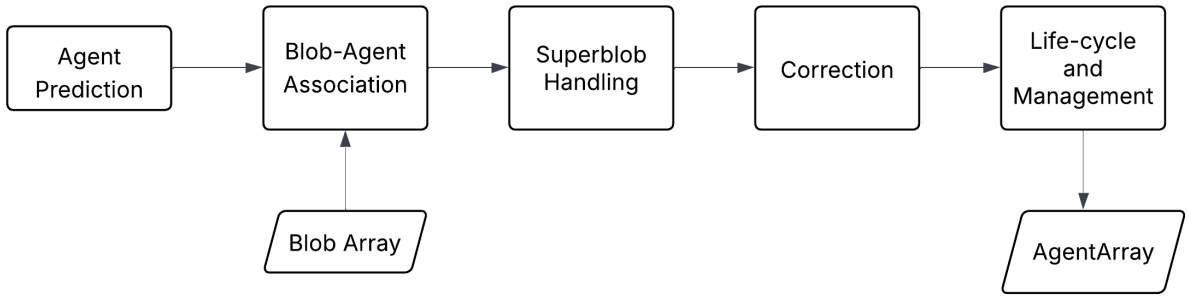


Figure 3.7: Diagram flow of the tracking process for robust evaluation of agents' position and velocity based on visual input.

Stage Outputs

The preprocessing stage produces two outputs consumed by downstream nodes. The *foreground mask* is a binary image published per frame that indicates which pixels belong to detected person regions. It is then used by the fusion node for spatial reasoning about agent positions. The *blob array* is the primary output: a list of detected agent descriptors, each containing the agent's centroid coordinates, bounding box, pixel area, and static flag. This array is forwarded to the tracking node (Section 3.5), which maintains persistent agent identities across frames using Kalman filtering, and subsequently to the identity verification and authorization stage (Section 3.6).

3.5 Person Tracking

The preprocessing node provides per-frame blob information containing the centroid and bounding box of each detected person region. These measurements are inherently unreliable in isolation due to ToF jitter, transient segmentation failures, noisy measurements, and possible partial occlusions. A dedicated tracking algorithm consumes the blob array information published by the preprocessing node and computes agent state estimates across frames, bridging missed detections and providing stable identities required by downstream identity verification and gesture recognition nodes. A Kalman Filter [25] is chosen as the core estimation component, fusing noisy centroid measurements with physics-based motion prior to compute smooth position and velocity estimates for each agent. This choice is consistent with prior overhead ToF tracking systems, where Kalman filtering has been validated as an effective solution for handling detection noise and bridging occlusion gaps [2, 3, 62], and has been identified as a foundational tool for visual tracking in gesture recognition pipelines [43]. Figure 3.7 show the interaction of the main components for the people tracking block, which are in detail described throughout the rest of this section.

State Representation

Each tracked agent is modeled by a four-dimensional state comprising its 2D image-plane position and velocity:

$$x_k = \begin{bmatrix} x \\ y \\ vx \\ vy \end{bmatrix} \tag{3.5}$$

Only position is directly observed through measurements (blob centroid):

$$z_k = \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.6}$$

Velocity is then implicitly inferred through filter dynamics. While centroid positions can be reliably extracted from the foreground mask, instantaneous velocity cannot be measured from a single depth frame. The process noise covariance Q encodes this asymmetry in measurement availability by assigning a low uncertainty to position and high uncertainty to velocity. The filter then trusts position measurements closely while letting velocity estimates adapt freely to observed motion.

Motion Model

A constant-velocity model is adopted [58, 62], under which position is propagated by the current velocity estimate and velocity is held constant between frames:

$$x_{k+1} = x_k + v_x \cdot \Delta t, \qquad y_{k+1} = y_k + v_y \cdot \Delta t \tag{3.7}$$

This can be written compactly as $\mathbf{x}_{k+1|k} = F\,\mathbf{x}_{k|k}$, where the transition matrix is:

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

F propagates position by the current velocity estimate scaled by Δt and leaves velocity unchanged. At typical processing rates, human walking acceleration between frames is small relative to the measurement noise, making the constant velocity assumption an acceptable approximation. The time step Δt is computed from the actual timestamp difference between consecutive blob messages rather than a fixed rate, ensuring correct behavior under variable processing times.

Agent-Blob Association

Blobs are assigned to existing agents by calculating a cost metric using the *Mahalanobis distance* between the blob centroid and the filter's predicted position [41]:

$$d_M = \sqrt{\mathbf{y}^\top S^{-1}\mathbf{y}} \tag{3.9}$$

where equation 3.10 defines the innovation and equation 3.11 defines the innovation covariance by combining the filter's current error covariance P with the measurement noise covariance R, and the current measurement matrix H. Unlike Euclidean distance, Mahalanobis distance [41] scales the gate with the filter's current uncertainty, a gating approach proven highly effective for overhead depth tracking [3, 62]. For instance, an agent that has been unobserved for several frames accumulates larger P, expanding its gate and enabling re-association at larger distances. On the other hand, a well-tracked agent maintains a tight gate, reducing the risk of erroneous cross-assignments. Pairs exceeding a fixed gate threshold are excluded from the cost matrix entirely.

$$\mathbf{y} = \mathbf{z} - H\mathbf{x}_{k+1|k} \tag{3.10}$$

$$S = HPH^\top + R \tag{3.11}$$

The gate threshold is grounded in the statistical distribution of d_M^2. Under the Kalman filter's Gaussian noise assumption, the squared Mahalanobis distance $d_M^2 = \mathbf{y}^\top S^{-1}\mathbf{y}$ follows a chi-squared distribution with n degrees of freedom, where n is the measurement dimension [58]. Since each observation is a 2-D centroid, $n = 2$ and therefore $d_M^2 \sim \chi_2^2$. The threshold $d_M^2 \leq 9.21$ corresponds to the 99th percentile of this distribution.

The globally optimal assignment over all valid agent–blob pairs is found using the *Hungarian algorithm* [30]. The problem is formulated as a linear assignment: given the binary assignment matrix $A \in \{0,1\}^{|\mathcal{A}|\times|\mathcal{B}|}$, find the assignment that minimises the total cost:

$$\min_A \sum_{i,j} C_{ij}\, a_{ij} \quad \text{subject to} \quad \sum_j a_{ij} \leq 1 \;\; \forall i, \quad \sum_i a_{ij} \leq 1 \;\; \forall j, \quad a_{ij} \in \{0,1\} \tag{3.12}$$

where C_{ij} is the Mahalanobis distance between agent i and blob j, set to infinity for pairs outside the gate. The two inequality constraints enforce that each agent is matched to at most one blob and each blob to at most one agent. Unlike a greedy nearest-neighbor approach, this formulation minimizes the total cost simultaneously across all agents and blobs, which is important in crowded scenes where a locally greedy choice can strand a nearby agent without a match.

Superblob Handling

Even when the preprocessing node splits a superblob succesfully at the mask level, the tracker must handle the residual case where the Hungarian assignment maps multiple agents to a single blob. This is used as a fallback mechanism complementary to the

preprocessing splitting since blob splitting might not define with complete accuracy the region belonging to each agent, which could happen if agents remain closely merged for several frames. This scenario can be straight forward detected when multiple agents are assigned to the same blob while other agents are left unmatched. Combined with a spatial check, it is possible to check if all candidate agent's predicted centroids lie within the blob's bounding box and rule out an inflated gate falsely drawing in a distant agent.

When a superblob event is confirmed at the tracker level, the preferred strategy is to trust the Kalman prediction rather than the corrupted observation, keeping each agent's state estimate independent. This is only feasible when the agents have accumulated sufficient tracking confidence, otherwise the merged observation is used as a fallback to prevent track loss.

Agent life-cycle

Agent creation, state transitions, and deletion are parts of a life-cycle designed to suppress spurious measurements while retaining real agents through brief detection gaps [62].

When an unmatched blob cannot be associated with any existing track, a new agent may be created. However, blobs appearing suddenly in the interior of the scene during a static period are more likely to be sensor noise artifacts or superblob fragments than genuine new persons, since real persons typically enter the field of view from the image boundary. Spawn location is therefore classified as either *BOUNDARY* (near the image edge) or *CENTER* (interior). Center-spawned blobs are suppressed from creating a new agent if the scene is currently static or if an existing agent is already in close proximity. Boundary-spawned blobs are always accepted as legitimate person entries.

New agents begin with a *CANDIDATE* state and are not reported to downstream nodes until they have accumulated a minimum number of consecutive successful detections, after which they are promoted to *ACTIVE*. This prevents transient single-frame detections from propagating through the system. When an agent is not matched for a frame, its confidence decays and a miss counter increments. When the miss counter exceeds a threshold the agent state changes to *LOST*, and it is deleted after a brief grace period that accommodates momentary occlusions.

The bounding box dimensions are not updated directly from each blob measurement but smoothed using an Exponential Moving Average:

$$w_k = \alpha\, w_{k-1} + (1-\alpha)\, w_{\text{blob}}, \qquad h_k = \alpha\, h_{k-1} + (1-\alpha)\, h_{\text{blob}} \tag{3.13}$$

to avoid abrupt size jumps caused by segmentation jitter, since downstream stages use bounding box information to determine the region of interest for motion computation.

A duplicate-merging step runs each frame as a final safeguard against residual fragment agents that escape the creation guards [3]. Two agents are candidates for merging based on a combination of criteria: bounding-box overlap (Intersection over Union), centroid proximity, and size ratio between the two bounding boxes. The last criteria specifically targets the case where a small fragment blob is tracked alongside the primary agent it

belongs to. When a merge is triggered, the agent spawned at the boundary or with greater age is retained, absorbing the confidence of the discarded duplicate.

Stage Outputs

The tracking stage produces an *agent array*. For each tracked agent the following information is obtained: estimated centroid position $[c_x, c_y]$, estimated velocity $[v_x, v_y]$, an EMA-smoothed bounding box, and a lifecycle state (CANDIDATE, ACTIVE, or LOST). The identity verification node (Section 3.6) consumes this array to extract per-agent motion energy and correlate it with the IMU signal. The gesture recognition node (Section 3.7) uses the bounding box of the authorized agent to define the region of interest for segmentation.

3.6 Identity Verification and Authorization

The Authorization step correlates IMU measurements with vision data in order to determine the authorized agent. Figure 3.8 provides a high level overview of the different blocks interacting together to determine authorization confidence. Each of them is described in detail throughout the rest of this section.

IMU data acquisition

An Inertial Measurement Unit was selected as second modality to complement perception with the ToF sensor and through sensor fusion identify the selected agent. The additional hardware the user has to wear is limited to an ESP32 and IMU 6050 that can easily be fitted in a wrist band. Wireless communication through WiFi is used to avoid disturbing natural hand movements of the user.

Once the data is received, integrity of the package is reviewed (Linear Acceleration on three axis: AcX, AcY, AcZ and Angular Velocity on three axis: GyX, GyY, GyZ) this is published as a ROS topic downstream so that it can be used by the fusion and gesture (Section 3.7 and Section 3.8).

To correctly determine linear acceleration a Madgwick Filter [21, 40] is used. Then the linear acceleration and angular velocity magnitudes are obtained to be processed and determine authority together with vision insights.

Available measurements and correlation

The MPU-6050 sensor provides raw 16-bit signed integer readings for three-axis linear acceleration and three-axis angular velocity. The accelerometer is configured at its default

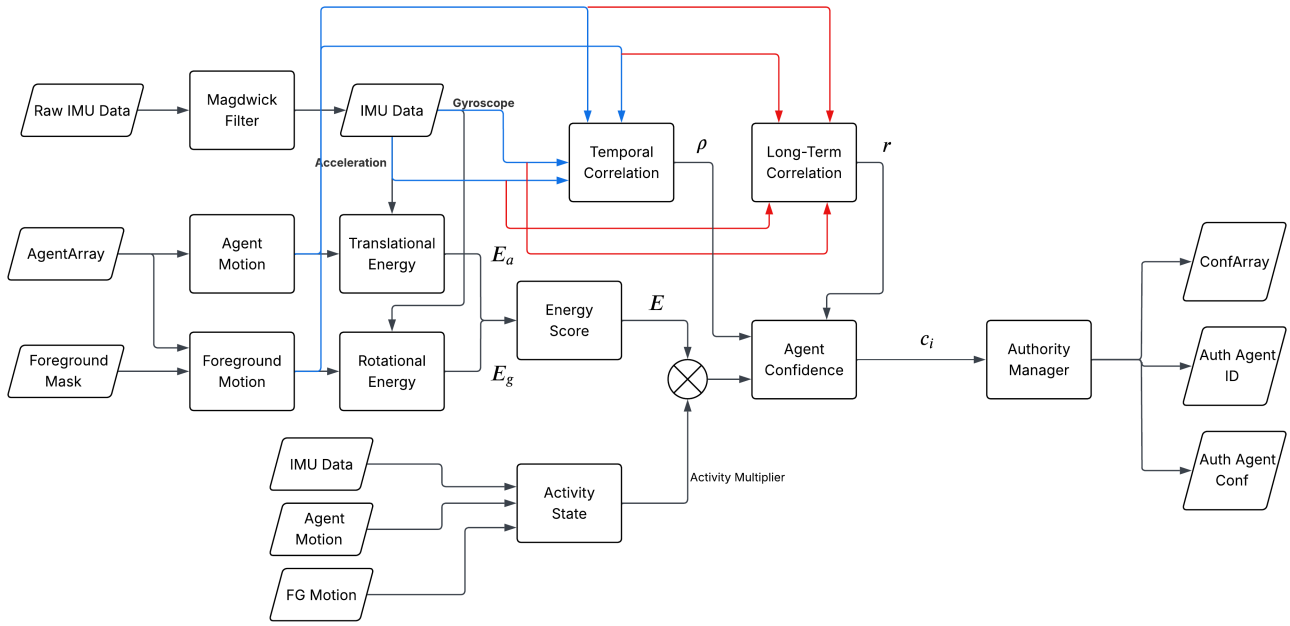


Figure 3.8: Functional architecture of the Identity Verification and Authorization stage. The pipeline fuses inertial data (compensated via a Madgwick filter) with visual tracked agent data. It computes instantaneous energy scores (E), short-term temporal correlations (ρ), and long-term activity bonuses (r) across translational and rotational channels to maintain a per-agent confidence score (c_i). Final authorization is determined by the Authority Manager.

full-scale range of $\pm 2\,g$, corresponding to a sensitivity of $S_a = 16384\,\mathrm{LSB}/g$ and readings are converted to $\mathrm{m/s^2}$

$$a_i = \frac{\mathrm{Ac}_i}{S_a} \cdot g_0, \quad g_0 = 9.80665\ \mathrm{m/s^2}, \quad i \in \{x, y, z\} \tag{3.14}$$

The gyroscope is configured at $\pm 250\,^\circ/\mathrm{s}$, giving $S_\omega = 131\,\mathrm{LSB}/(^\circ/\mathrm{s})$, and readings are converted to rad/s:

$$\omega_i = \frac{\mathrm{Gy}_i}{S_\omega} \cdot \frac{\pi}{180} \tag{3.15}$$

Since the accelerometer measures total specific force, including the gravity acceleration, the obtained measurement has to be compensated (removing gravity term) according to the sensor orientation to represent pure motion. The Madgwick filter [40] provides an orientation estimate as a unit quaternion q, from which the rotation matrix $R(q)$ is derived. The gravity vector in the world frame $\mathbf{g}_w = [0,\ 0,\ g_0]^T$ is rotated into the sensor frame and subtracted:

$$\mathbf{g}_s = R(q)^T\,\mathbf{g}_w, \qquad \tilde{a}_i = a_i - g_{s,i} \tag{3.16}$$

The gravity-compensated acceleration vector $\tilde{\mathbf{a}}$ and angular velocity vector $\boldsymbol{\omega}$ are accumulated following a time window technique with buffer duration of T_w. Motion energy over the window is represented as Root Mean Square (RMS) scalar magnitudes:

$$\bar{a} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} \|\tilde{\mathbf{a}}_k\|^2}, \qquad \bar{\omega} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} \|\boldsymbol{\omega}_k\|^2} \tag{3.17}$$

On the visual side, two complementary measurements are extracted per tracked agent. First the agent's linear velocity is derived from the Kalman Filter estimate $[v_x, v_y]$. Kalman Filter provides this measurement in pixels/seconds. Through camera calibration, extracting the horizontal Field of View (θ_{FOV}), and using the camera mounting height (h) it is possible to transform this measurement to m/s

$$v_{\mathrm{m/s}} = \sqrt{v_x^2 + v_y^2} \;/\; \frac{W_{\mathrm{px}}}{2h\,\tan(\theta_{\mathrm{FOV}}/2)} \tag{3.18}$$

Additionally, visual motion energy is captured calculated by measuring frame-to-frame change in foreground pixel count within the agent's bounding box region of interest. This measurement is intended to spike when an agent triggers a gesture (arm-hand movement detected in foreground).

$$\Delta f_t = |f_t - f_{t-1}| \tag{3.19}$$

where f_t is the foreground pixel count inside the bounding box at time t. The RMS of Δf over a short history window is used as the in-place motion energy.

This measurements can be correlated and form the basis for person identification.

- **Translational channel**: IMU acceleration magnitude $\bar{a}$ should correlate with visual agent speed $\bar{v}$, since walking and arm displacement produce both measurable inertial acceleration and centroid displacement in the depth image.

- **Rotational channel**: IMU angular velocity magnitude $\bar{\omega}$ should correlate with foreground pixel change $\overline{\Delta f}$ within the agent's ROI, since gestures and body rotation deform the silhouette without necessarily displacing the centroid.

All four measurements are normalized to $[0, 1]$ using adaptive ceilings that track the maximum observed value with a slow exponential decay, preventing the normalization from becoming anchored to extreme one-off transients. In this way the system can adapt to movement patterns of different users and avoid being biased towards certain group.

Determining agent confidence

The instantaneous cross-modal similarity between the IMU and the visual agent is quantified through an *energy score* using a Gaussian kernel applied independently to each channel

$$E_a = e^{-\lambda(\hat{v}-\hat{a})^2}, \qquad E_g = e^{-\lambda(\hat{f}-\hat{\omega})^2} \tag{3.20}$$

where $\hat{v}$, $\hat{a}$, $\hat{f}$, $\hat{\omega} \in [0, 1]$ are the normalized visual speed, accelerometer magnitude, foreground energy, and gyroscope magnitude respectively, and λ is the Gaussian scale parameter. The kernel assigns maximum similarity when both signals are equal and decays smoothly as their discrepancy grows. The combined energy score is

$$E = w_a\, E_a + (1 - w_a)\, E_g \tag{3.21}$$

where w_a controls the relative weight of translational versus rotational channel.

To capture temporal structure beyond instantaneous energy, two time-correlations are used to evaluate IMU and vision measurements similarities over different time spans. The system computes *temporal cross-correlation score* over a short term window (IMU window), intended to detect if the shape and timing of motion bursts match. This measurement is sensitive to gesture pattern motion, a spike in IMU should produce a matching spike in the visual signal. Both IMU and visual signals are interpolated to a common time grid (since camera and IMU operate at different rates), and the normalized cross-correlation is evaluated at integer sample lags τ within a tolerance window:

$$\rho(\tau) = \frac{\sum_t \big(a_t - \bar{a}\big)\big(b_{t-\tau} - \bar{b}\big)}{n\, \sigma_a\, \sigma_b} \tag{3.22}$$

The best non-negative correlation across all tested lags is returned as the temporal score. Lag tolerance accommodates communication latency between the IMU and the camera pipeline, as well as natural causal delays between arm onset and body centroid displacement. The score is computed independently on both channels and averaged.

Similarly, a *long-term activity correlation* is computed over a larger rolling history of length L. Using the Pearson correlation coefficient between the dual channel, IMU accelerometer history and agent's visual speed history and IMU angular velocity history and foreground

change history, a reward bonus is defined for agents whose activity patterns have co-varied with the IMU signal over many frames.

$$r = \gamma \left(\rho_{\text{Pearson}}^{(\text{accel, speed})} + \rho_{\text{Pearson}}^{(\text{gyro, fg})} \right) \tag{3.23}$$

where γ is a scaling factor. The three components are combined into a final per-frame score s:

$$s = \left[(1 - w_t)\, E + w_t\, \rho \right] \cdot (1 + r) \tag{3.24}$$

where w_t is the temporal weight. The score is then accumulated into a per-agent *confidence* via an Exponential Moving Average (EMA):

$$c_i \leftarrow \alpha\, c_i + (1 - \alpha)\, s_i \tag{3.25}$$

The EMA factor α acts as an implicit memory that prevents the system from being fooled by a single coincidental motion burst from a non-authorized agent, requiring sustained correlation evidence for confidence to build.

These two correlation measurements were selected for robustness against coincidental motion matches (two agents moving in a similar way in a short time span) and associate more precisely different movement patterns that can be observed in this thesis scope. For instance, gestures are short term and high intensity movements, while walking is continuous sustained through time.

Before computing per-agent scores, a global activity check is performed using balanced IMU energy:

$$E_{\text{bal}} = w_a\, \hat{a} + (1 - w_a)\, \hat{\omega} \tag{3.26}$$

If $E_{\text{bal}} < \epsilon$, the IMU signal is insufficiently informative for agent discrimination and the scoring step is skipped entirely, avoiding spurious confidence degradation during periods when the authorized user is at rest.

To further increase score discriminability, each agent is classified into an activity state from visual observations alone, deliberately excluding the IMU from this classification to avoid a circular dependency with the IMU-weighted scoring. Deriving discrete behavioral labels from tracked blob features such as centroid speed and silhouette motion energy is an established paradigm in overhead depth tracking [24, 49, 62]. Four states are defined: WALKING (high translational speed), GESTURE (low speed with elevated foreground change), IDLE (low speed and low foreground change), and AMBIGUOUS. State-specific multipliers modulate the energy score to reward correct modality agreement:

- **WALKING**: the score is amplified when both visual speed and IMU acceleration are simultaneously elevated, confirming that centroid displacement matches wrist inertia.
- **GESTURE**: the score is amplified when the agent is nearly stationary visually but shows elevated foreground change within its bounding box and the gyroscope registers significant angular motion, confirming an in-place gestural activity.
- **IDLE**: the score is amplified when IMU energy, visual speed, and foreground change are all simultaneously low, confirming mutual stillness; it is penalized otherwise, since a visually idle agent inconsistent with an active IMU strongly suggests the agent is not the IMU wearer.

Determining authorization

The agent with the highest accumulated confidence above an authorization threshold θ_{auth} is declared the authorized agent. If no agent reaches this threshold, the system either retains the previously authorized agent provided its confidence remains above a relaxed threshold $\theta_{\text{relaxed}} < \theta_{\text{auth}}$, or de-authorizes entirely. This hysteresis band prevents oscillation between authorized and de-authorized states due to momentary score drops during brief occlusions, motion interruptions, or noisy/inaccurate measurements.

Probabilistic approach

Several mechanisms are combined to ensure that the authorization decision is stable and robust against transient misclassifications.

Hysteresis and switch margin. Once an agent has been authorized, a minimum authorization duration T_{hyst} must elapse before a different agent can take over. Furthermore, the challenger's score must exceed the current authorized agent's score by a multiplicative margin $m > 1$, requiring clear qualitative improvement. An additional cooldown T_{cool} after each authorization change further suppresses rapid oscillations.

Reentry logic. When the authorized agent's track is deleted, indicating the agent has physically left the scene, an exit event is flagged. Any re-authorization within a subsequent reentry window T_{re} requires a stricter threshold $\theta_{\text{re}} > \theta_{\text{auth}}$ preventing a non-authorized person who remained on the scene or enters immediately after the authorized user left the scene is incorrectly promoted.

Authority decay. When no valid scores can be computed, because the IMU signal is below the activity threshold or no candidate agents pass the age filter, all confidence scores are multiplied each callback by a decay factor δ close to 1. This provides a gradual response to inactivity rather than an abrupt de-authorization.

Exploration–exploitation trade-off. When a strong authorized agent exists, evaluating all competing agents each frame wastes computation and introduces unnecessary confidence volatility. A probabilistic scheduling policy assigns each non-authorized agent an evaluation probability that decreases as the authorized agent's confidence increases, balancing the need to detect legitimate challengers against the efficiency gain of concentrating resources on the confirmed authorized agent.

Stage Outputs

The identity verification and authorization stage produces a *confidence array*, which communicates the confidence and the activity state for each agent of the scene. This information is relevant for visualization and metrics purposes. Additionally, to simplify gesture triggering, this stage communicates the *authorized agent id* (if any) and the *authorized agent confidence* downstream. The gesture node (Section 3.8) consumes this information

and, if a gesture is detected while an agent is authorized, the gesture is triggered under the authorized agent authority.

3.7 Gesture Segmentation

Accurate gesture recognition requires knowing precisely when a gesture begins and ends, so that only the motion samples belonging to the gesture trajectory are used for classification. The wrist-worn IMU on the gesturing hand provides a natural segmentation signal, when performing a gesture, the authorized user is in a relative rest posture before and after a gesture, with a burst of arm motion in between. A two-state finite state machine (2-FSM) running directly on the IMU signal detects these transitions in real time, separating the continuous IMU stream into discrete gesture episodes.

Motion Energy and Segmentation Signal

Each IMU sample provides a 6-dimensional observation (Section 3.6). These are reduced to a single scalar *energy* value using the same balanced combination defined for the fusion node (Section 3.6).

$$E = w_a\,\hat{a} + (1 - w_a)\,\hat{\omega}, \qquad \hat{a} = \min\left(\frac{\|\tilde{\mathbf{a}}\|}{a_{\max}},\, 1\right), \quad \hat{\omega} = \min\left(\frac{\|\boldsymbol{\omega}\|}{\omega_{\max}},\, 1\right) \tag{3.27}$$

This energy measurement captures combined translation and rotational wrist activity and forms the input to the segmentation state machine.

Segmentation State Machine

IDLE → ACTIVE (onset detection). Each sample in IDLE state is compared against an onset activity threshold θ_{onset}. A consecutive counter increments when $E > \theta_{\text{onset}}$ and resets to zero on any value below threshold. The transition to ACTIVE is triggered only when the counter reaches a minimum N_{onset}, requiring uninterrupted sustained energy above the threshold. This prevents transient noise spikes from falsely triggering the ACTIVE state.

ACTIVE → IDLE (offset detection). While in ACTIVE state, each incoming sample is appended to the gesture buffer. On a similar way, a consecutive counter increments when $E < \theta_{\text{offset}}$ and resets to zero on any above threshold value. The system transitions back to IDLE when the counter reaches $N_{\text{offset}} > N_{\text{onset}}$. This asymmetry is chosen intentionally, onset detection should be responsive to sudden motion, while offset detection should be conservative to avoid prematurely cutting off the settling phase at the end of a gesture.

The energy thresholds, $\theta_{\text{onset}} > \theta_{\text{offset}}$, create an hysteresis band. This prevents rapid state toggling when energy fluctuates near a single boundary value.

Lookback Buffer and Authorization Snapshot

There's an uncertainty moment between every IDLE→ACTIVE transition. To avoid loosing gesture trajectory samples, when a transition is triggered, gesture buffer is pre-loaded with a short lookback segment from a rolling buffer that continuously stores recent IMU samples. This captures the pre-onset ramp that precedes threshold crossing. Without a lookback buffer the samples before the N_{onset} crosses the threshold would be lost and the recorded trajectory would start mid-motion.

Similarly, at the transition moment, the current authorized agent ID is captured into a snapshot variable. This decouples the authorization state at gesture onset from any subsequent changes that may occur while the gesture is being recorded. Without this snapshot, a mid-gesture authorization change could be applied retroactively to a gesture whose onset was recorded under a valid authorization, causing incorrect rejection.

Duration Constraints and Tail Trimming

Two duration bounds restrict valid gesture length. A minimum L_{min} filters out very short sequences resulting from accidental micro-movements or segmentation noise, since they carry insufficient information for reliable recognition. A maximum L_{max} discards sequences that have grown too long to represent a spontaneous single gesture, such as sustained arm motion during walking.

When the offset transition is confirmed, the N_{offset} low-energy tail samples that triggered it are trimmed from the captured sequence. These samples represent the post-gesture settling period. Retaining these samples would append a near-zero energy suffix that would distort DTW alignment (all samples ending with the same pattern).

Graphic Representation

Figure 3.9 summarizes the complete state machine, combining the onset and offset transitions, the lookback capture, the authorization snapshot, and the duration guards into a single diagram.

Stage Outputs

The segmentation stage publishes the current *gesture state* (IDLE or ACTIVE) after every transition. Every ACTIVE→IDLE transition, the validity of the collected gesture trajectory is evaluated. If satisfied, this transition triggers a flag for gesture recognition (Section 3.8). Additionally, this information is used for visualization purposes for real-time monitoring and for system metrics (measure gesture latency from onset to recognition).

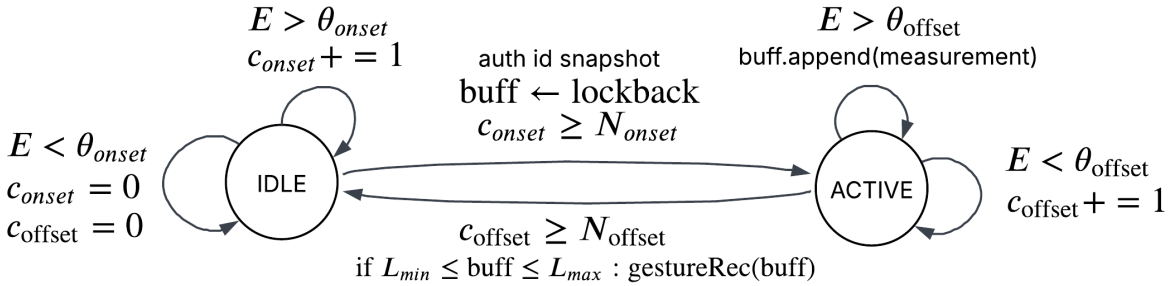


Figure 3.9: Two-state Finite State Machine (2-FSM) for gesture segmentation. IDLE→ACTIVE transition requires N_{on} consecutive samples above θ_{on}. ACTIVE→IDLE transition requires N_{off} consecutive samples below θ_{off}. A captured segment is forwarded for classification only if its length falls within $[L_{\min}, L_{\max}]$. Segments outside this range are discarded. The authorization snapshot is taken at the IDLE→ACTIVE transition and is held fixed for the duration of the gesture.

3.8 Gesture Recognition

Gesture recognition is performed using mathematical approach Dynamic Time Warping (DTW) [55], a similarity measure for time series that finds the minimum-cost of alignment between two sequences by allowing elastic warping of the time axis. DTW is well suited to wrist gesture recognition because gestures of the same class may be executed at different speeds or motion width by different users, and a rigid sample-by-sample comparison would penalize this natural variability, an advantage of DTW that has been effectively utilized in other sensing applications [73]. Compared to neural network classification approaches, DTW requires no training beyond a small set of recorded exemplars per class (not necessarily all the intended authorized agents have to be recorded), offers built-in out-of-distribution rejection through a distance threshold, and incurs negligible computation costs, plus it is not necessary to invest resources for training. Besides, neural networks are not inherently good rejecting OOD samples without techniques as open-world classification or prediction confidence estimation.

Feature Representation and Normalization

Each segmented gesture is represented as a matrix $G \in \mathbb{R}^{N\times 6}$ where N is the number of IMU samples captured during the ACTIVE state and the six columns are the gravity-compensated acceleration and angular velocity components

$$[\tilde{a}_x,\, \tilde{a}_y,\, \tilde{a}_z,\, \omega_x,\, \omega_y,\, \omega_z] \tag{3.28}$$

Before classification or template storage, the matrix is normalized per axis using z-score standardization

$$\hat{g}_{i,k} = \frac{g_{i,k} - \mu_k}{\sigma_k}, \qquad \mu_k = \frac{1}{N}\sum_i g_{i,k}, \quad \sigma_k = \sqrt{\frac{1}{N}\sum_i (g_{i,k} - \mu_k)^2} \tag{3.29}$$

Axes with $\sigma_k < 10^{-6}$ are left unchanged to avoid division by near-zero. This normalization removes inter-user amplitude differences while preserving the temporal trajectory shape, making the template library regardless of which person recorded the exemplars.

DTW Distance with Sakoe-Chiba Band

Given a normalized query $G \in \mathbb{R}^{N\times 6}$ and a template $T \in \mathbb{R}^{M\times 6}$, the frame-level cost is the Euclidean distance between rows

$$d(i,j) = \|G_i - T_j\|_2 \tag{3.30}$$

The DTW alignment cost is computed by dynamic programming

$$\mathrm{DTW}(i,j) = d(i,j) + \min \begin{cases} \mathrm{DTW}(i-1,\, j) \\ \mathrm{DTW}(i,\, j-1) \\ \mathrm{DTW}(i-1,\, j-1) \end{cases} \tag{3.31}$$

with $\mathrm{DTW}(0,0) = 0$ and all other border cells initialized to infinity. To prevent pathological warping and limit computation, a Sakoe-Chiba band of width r [55] constrains the search to cells satisfying $|i-j| \le r$

$$r = \lfloor \rho \cdot \max(N, M) \rfloor \tag{3.32}$$

where $\rho \in (0,1]$ is the band ratio parameter. The band also implicitly rejects templates whose length differs from the query by more than r samples. The final score is normalized by path length to the comparable across gesture pairs of different durations

$$d_{\mathrm{DTW}} = \frac{\mathrm{DTW}(N,M)}{N+M} \tag{3.33}$$

Template Library and Classification Rule

Templates are pre-recorded samples stored as normalized 6-channel time series. Multiple samples per class cover intra-class variability. The predicted class is the one whose nearest template achieves the globally minimum DTW

$$c^* = \arg\min_{c} \; \min_{k} \; d_{\mathrm{DTW}}(G,\, T_k^{(c)}) \tag{3.34}$$

The gesture is recognized as c^* if the best-match distance falls below a classification threshold $\theta_{\mathrm{DTW}}^{(c)}$, which can be fine-tuned per class.

$$\hat{c} = \begin{cases} c^* & \text{if } \min_k d_{\mathrm{DTW}}(G, T_k^{(c^*)}) < \theta_{\mathrm{DTW}}^{(c^*)} \\ \text{reject} & \text{otherwise} \end{cases} \tag{3.35}$$

If no class meets its threshold, the gesture is rejected as OOD. This is a structural advantage of DTW over softmax-based classifiers, since the distance to templates has a natural

dissimilarity interpretation and single per-class scalar suffices to exclude unknown gestures without an explicit negative class or probability calibration. This property is analogous to One-Class Classification (OCC), where a model is trained exclusively on positive examples and rejects anything outside the learned boundary [70]. DTW achieves the same effect non-parametrically through a distance threshold.

To avoid unnecessary computation, an early stopping optimization is implemented. If any distance falls below an strict confidence threshold $\theta_{\text{strict}} \ll \theta_{\text{DTW}}$, evaluation terminates immediately. Since DTW reliably assigns very low distances to strongly matching pairs, this early exit rarely affects recognition correctness while significantly reducing computation.

Template Quality Control

Recording templates in practice occasionally produces very short files corresponding to accidental micro-movements or premature trigger events. These degrade matching because their length mismatch with typical queries places them outside the Sakoe-Chiba band, silently reducing the effective template count per class. A utility template cleaning script was used to scan the template directory, identify files below a configurable minimum sample count, report them grouped by class, and remove them after explicit user confirmation. A dry-run mode is available to preview deletions without modifying any files.

Stage Outputs

When a candidate segment is accepted, the *recognized gesture* class name is communicated downstream, being the terminal output of the pipeline. Rejected segments (those whose minimum DTW distance exceeds all per-class thresholds) produce no output. The *recognized gesture* information is collected and used for system evaluation, overall metrics (Section 5), and visualization purposes.

Chapter 4

Implementation

4.1 Development Environment

The whole system is built relying on open-source software. As motivated in Section 3.3, ROS is adopted as the implementation platform for its publish-subscribe decoupling, node modularity, and capability of handling real-time high frequency sensor reading and synchronization. ESPROS provides a ROS driver for the TOFcam660 [8]. For this reason ROS Noetic (the last major ROS 1 release) was selected, running on Ubuntu 20.04 for full compatibility with the driver and the ROS distribution. The system is hosted on an HP Omen 15 laptop with 16 GB RAM.

The previously defined System Components 3.1 are implemented in four ROS packages, all built inside a single catkin workspace:

- `tof_preprocessing`: Performs background modeling, foreground segmentation, blob detection, and Kalman tracking. It is the core processing node and the primary consumer of the cam660 depth stream.
- `gesture`: Implements the IMU-driven gesture segmentation state machine and DTW-based gesture classification. It subscribes to IMU data and the authorized agent topic.
- `mpu`: Driver and interface for the wrist IMU. Publishes raw inertial measurements and defines the custom message types used across the system. Implements confidence-based IMU-vision fusion to determine authorized-agent selection
- `metrics`: Passive monitoring node that subscribes to all output topics and collects per-agent and scene-level statistics, writing a structured report on shutdown.

The interaction between the different ROS nodes be visually represented in a ROS graph, which shows the complexity of the implemented system (see Appendix A). The expected system inputs are a depth image and raw IMU measurements, which can either be provided

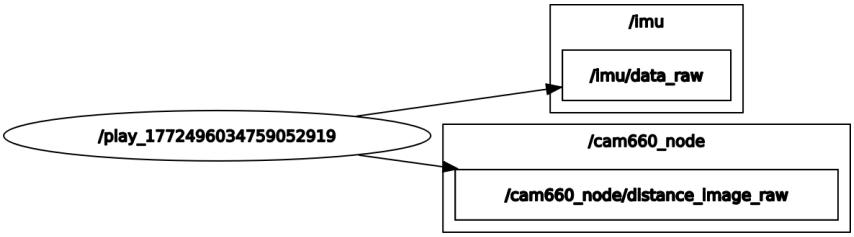


Figure 4.1: Expected system inputs. In this specific example obtained from rosbag replay.

from a prerecorded rosbag (as shown in Figure 4.1) or being transmitted online by the TOFcam660 and MPU-6050.

Additionally, Table 4.1 lists the main ROS topics through which the nodes exchange data.

Topic	**Message Type**	**Publisher → Subscriber(s)**
`/cam660_node/distance_image_raw`	sensor_msgs/Image	cam660_driver → /preproc
`/imu/data_raw`	IMU	/mpu/mpu_reader → /imu_filter_node
`/imu/data`	IMU	/imu_filter_node → /fusion, /gesture
`/preproc/foreground_mask`	sensor_msgs/Image	/preproc → /fusion, /vis_*
`/preproc/blobs`	BlobArray	/preproc → /tracking
`/tracking/kalman`	AgentArray	/tracking → /fusion, /metrics
`/fusion/confidence`	ConfArray	/mpu → /metrics
`/fusion/authorized_agent_id`	Int32	/fusion → /gesture, /metrics
`/fusion/authorized_agent_confidence`	Float32	/fusion → /metrics
`/gesture/state`	String	/gesture → /metrics
`/gesture/recognized`	String	/gesture → /metrics

Table 4.1: Main ROS topics exchanged between system nodes.

The pipeline is implemented in Python, using `rospy` [48] for ROS communication, `numpy` [19] and `scipy` [68] for numerical computation, and `cv2` [4] for image processing and on-screen visualization. The catkin build system manages the workspace and places all packages on the `ROS_PACKAGE_PATH`. Shell scripts provide system-level wrappers that start the full pipeline with a single terminal command, as described in Section 4.2.

Hardware Setup

The TOFcam660 is mounted in an overhead position approximately 1.9 m above the floor, pointing straight down, and connected to the host laptop through Ethernet communication protocol. At this height, the 108° horizontal field of view covers a ground area of

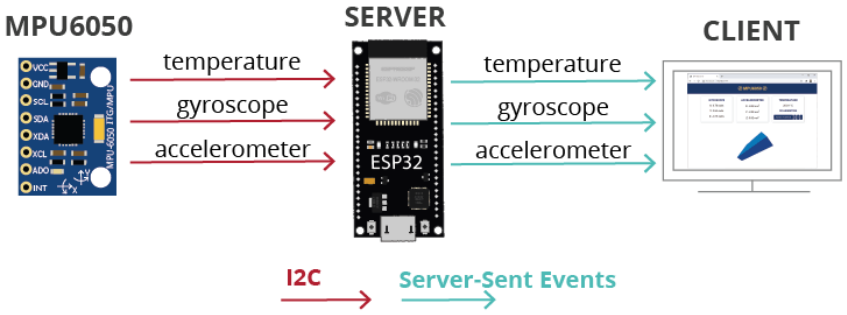


Figure 4.2: ESP32, IMU, and server interaction [52]

roughly 4.0 m × 3.0 m, sufficient to monitor a typical desk workspace or small room area. Table 4.2 describes the ROS interface provided in the ESPROS Software Development Kit (SDK), the obtainable information through rostopics, and data types. The specific calibration used in this project can be found in Appendix B.

Topic	Message Type	Description
`cam660_node/distance_image_raw`	sensor_msgs/Image	Sends the distance image for image type parameters which include distance
`cam660_node/amplitude_image_raw`	sensor_msgs/Image	Sends the grayscale or amplitude image according the selected image type parameter
`cam660_node/dcs_image_raw`	sensor_msgs/Image	Sends 4 DCS images
`cam660_node/points`	sensor_msgs/PointCloud2	Sends the point cloud image for image type parameters which include distance

Table 4.2: ROS topics and functions for the cam660_node [9].

The motion sensing component is a wrist-worn IMU based on the MPU-6050 sensor [22], which measures three-axis linear acceleration and three-axis angular velocity. The sensor communicates with an ESP32 microcontroller via I2C, and inertial data is transmitted wirelessly over Wi-Fi to the processing host at 50 Hz, where it is published as a ROS topic for downstream nodes. Figure 4.2 shows a diagram of the ESP32 and IMU interaction.

The ESP32 microcontroller code was implemented using micropython and transmits inertial packets over UDP as described in Section 4.4.1.

Launch Infrastructure

Each package defines its own per-node launch file, allowing any individual node to be started and tested in isolation. Two top-level launch files combine these into a complete pipeline:

- `system.launch`: full live operation, including the camera driver and the `mpu_reader` node communicating with the ESP32 over Wi-Fi.
- `system_rosbag.launch`: replay from a pre-recorded rosbag. Identical to the live variant except that the `mpu_reader` node is omitted, since IMU data is replayed from the bag.

Both files expose the four tunable preprocessing parameters as command-line arguments that can be overridden at launch without modifying any source file or rebuilding the workspace. For example,

```
roslaunch tof_preprocessing system.launch \
    open_kernel_size:=5 median_kernel:=5 \
    fg_depth_thresh:=1400 close_kernel_size:=19
```

A dedicated `show_vis.launch` starts all six visualization nodes simultaneously; it is included by both top-level launch files and can also be run independently. The full repository directory structure is provided in Appendix C.

4.2 System Utilities

A `system_commands.txt` file at the repository root collects the most commonly used terminal commands for operating the system: building the catkin workspace, sourcing the environment, launching the system live or from a rosbag, recording a new rosbag, and starting the gesture template recorder. This file serves as a quick-reference cheat sheet that avoids consulting the full documentation for routine operations. Complete documentation as well as system access can be found on the publicly accessible project repository (`https://github.com/JorgeOrtizV/identity-verified-gesture-rec`) [51].

4.3 Preprocessing

4.3.1 Preprocessing and Foreground Segmentation

The `tof_preprocessing` package is the primary consumer of the depth stream and the most computationally intensive component of the pipeline, implementing the preprocessing and foreground segmentation pipeline designed in Section 3.4 and the tracking pipeline designed in Section 3.5. It subscribes to raw 320×240 pixel depth images from the TOFcam660 driver and is responsible for depth cleaning, background modeling, foreground segmentation, morphological refinement, blob extraction, nearest-neighbor track association, superblob splitting, and adaptive background update. Its two outputs are a per-frame binary foreground mask published to `/preproc/foreground_mask` and a blob descriptor array published to `/preproc/blobs`, both consumed by downstream nodes.

The node is configurable through a set of ROS parameters summarized in Table E.1. Two parameters differ between single-agent and multi-agent configurations: `median_kernel` and `open_kernel_size`. In multi-agent scenarios, the simultaneous presence of multiple IR-emitting persons amplifies multi-path interference, generating a noisier depth signal than in single-agent conditions. Larger kernels for both the median smoothing and the morphological opening step are therefore required to suppress the additional artifacts without sacrificing blob separation quality.

Other tunable parameters are `fg_depth_thresh` and `close_kernel_size`. Depending on the height of the agents, camera height, pose of the agent (sitting or standing), and height of the chairs, a larger or smaller `fg_depth_thresh` can help to effectively isolate the upper-body. Additionally, due to the opening kernel size, blurring, and noisy measurements, tuning the `close_kernel_size` can result in a better quality representation of the agents' arms in the segmentation mask, which allows more precisely measurement of foreground energy when the agent is relatively static but gesturing (Section 4.4.2).

Besides multi-path interference in multi-agent scenarios, it was observed that parameter tuning could yield better results even for rosbags recorded back to back under nominally similar conditions. Possible explanations include thermal drift as the sensor heats up during operation, which induces temperature-dependent phase offsets in the demodulation electronics and increases photodetector dark current, as well as small variations in ambient lighting that add background IR signal to the sensor [13]. Both effects can generate small but consistent shifts in depth readings. Overall, tuning only these four parameters is sufficient to achieve high performance across different recording conditions.

Algorithm 1 presents the top-level per-frame processing callback. During the initialization phase the node accumulates the first `bg_frames` cleaned depth images and computes a per-pixel *nanmedian* to form the initial background model, after which normal processing begins. The `ExtractBlobs` step applies connected-component labeling with a two-tier size filter: components below `min_blob_size` are removed from the mask as residual noise and components above `min_blob_size` but below `min_blob_area` are retained in the cleaned mask but excluded from agent tracking.

Algorithm 2 details the foreground segmentation sub-routine. The morphological opening is applied globally to remove isolated noise pixels. The subsequent closing is performed *per connected component*: each component is processed independently within its local bounding box extended by a small padding margin, preventing the dilation step from merging the silhouettes of nearby agents.

Figure 4.3 illustrates the background initialization and foreground segmentation stages. After `bg_frames` are accumulated and the background model is complete, the foreground mask is empty in the absence of motion (Figure 4.3a). When agents enter the scene, the segmentation isolates their upper-body silhouettes in both single- and multi-agent conditions (Figures 4.3b and 4.3c).

Figure 4.4 shows blob extraction results using connected components, in both single- and multi-agent conditions.

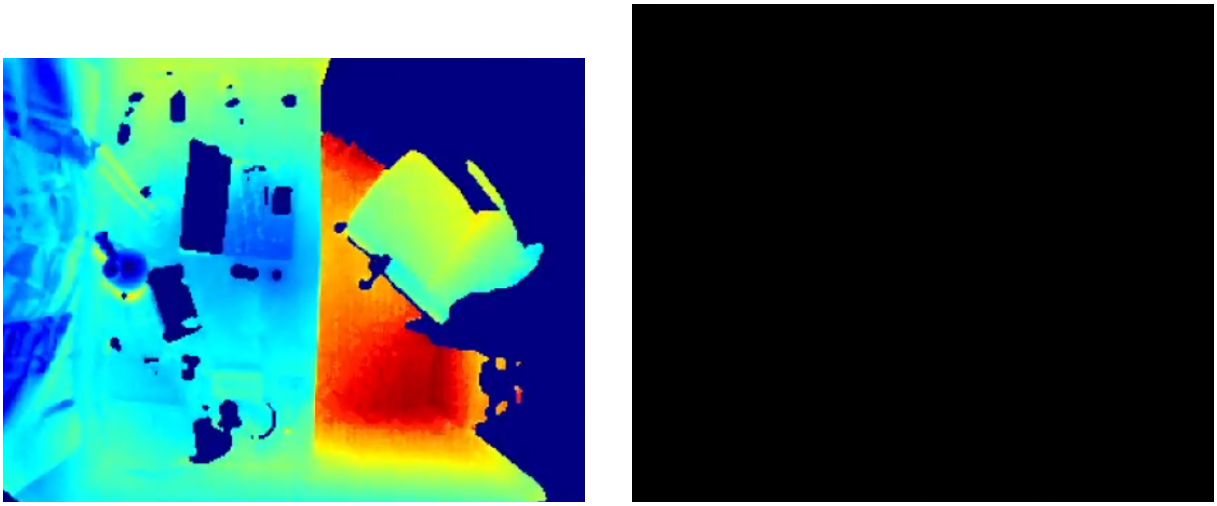
Blob-to-track association uses a greedy nearest-neighbor strategy. A blob is treated as a *superblob* (two or more merged agents) when its area exceeds `split_area_thresh` and its

Algorithm 1 tof_preprocessing — main per-frame callback

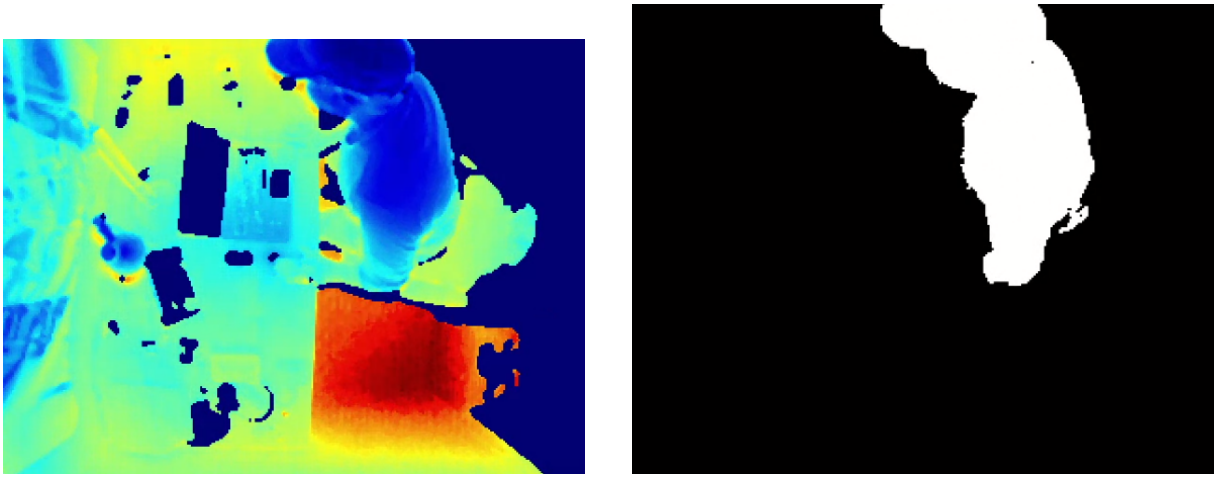
Input: `depth_raw` ▷ raw depth image from TOFcam660
`depth` ← DepthClipping(`depth_raw`, [`min_depth`, `max_depth`]) ▷ Out-of-range → NaN
`depth` ← Denoise(`depth`, `median_kernel`) ▷ Median smoothing. NaN → 0 before filtering
if not `bg_initialized` **then**
    Append `depth` to `buffer`
    if |`buffer`| ≥ `bg_frames` **then**
        `background` ← nanmedian(`buffer`)
        `bg_initialized` ← True
    end if
    return
end if
`fg_mask` ← ForegroundSegmentation(`depth`) ▷ Algorithm 2
`fg_mask`[(`depth` > `fg_depth_thresh`) ∨ (`depth` = 0)] ← 0 ▷ Remove floor-level objects
`fg_mask`, `blobs` ← ExtractBlobs(`fg_mask`) ▷ Two-tier size filtering + blob descriptors
UpdateForegroundAge(`fg_mask`) ▷ Per-pixel age: +1 if fg, reset to 0 if bg
`UpdateTracks`(`blobs`, `depth`) ▷ Algorithm 3
if `bridge_mask` ≠ ∅ **then**
    `background`[`bridge_mask`] ← `depth`[`bridge_mask`] ▷ Force-update inter-agent gap
    `fg_mask`[`bridge_mask`] ← 0
end if
`UpdateBackground`(`depth`, `fg_mask`) ▷ Algorithm 4
Publish `/preproc/foreground_mask` and `/preproc/blobs`

Algorithm 2 ForegroundSegmentation

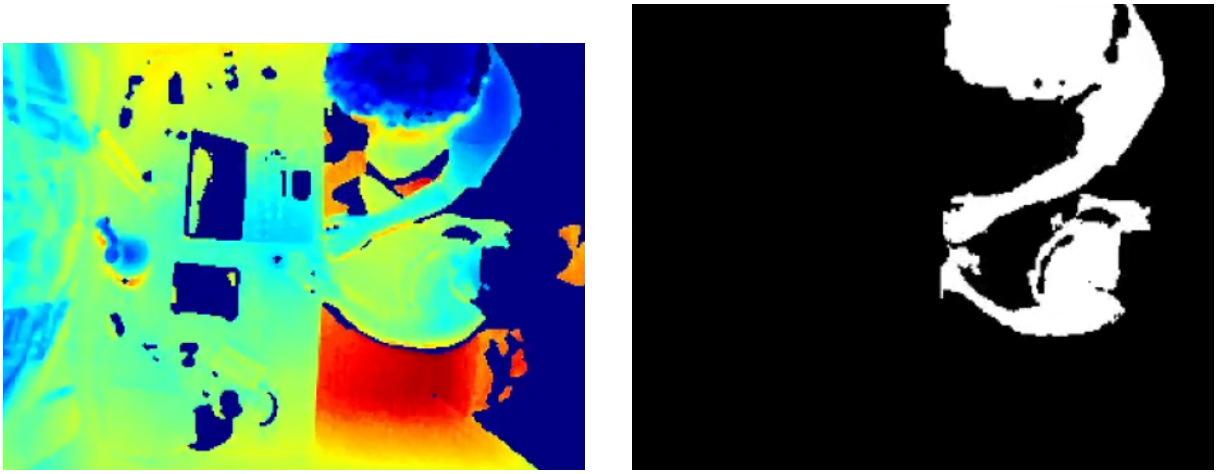
Input: `depth`, `background`
`diff` ← |`depth` − `background`|
`diff`[`invalid`] ← 0
`fg` ← (`diff` > τ)
`fg` ← `MorphOpen`(`fg`, `open_kernel_size`) ▷ Remove isolated noise blobs
$\mathcal{C}$ ← ConnectedComponents(`fg`); `result` ← **0**
for each $c \in \mathcal{C}$ with area(c) ≥ `min_blob_size` **do**
    `roi` ← BoundingBox(c) ⊕ pad ▷ Expand by padding margin
    `closed` ← MorphClose($c \cap$ roi, Ellipse(`close_kernel_size`))
    `result`[`roi`] ← max(`result`[roi], `closed`)
end for
return `result`



(a) Static scene and foreground segmentation after background initialization.

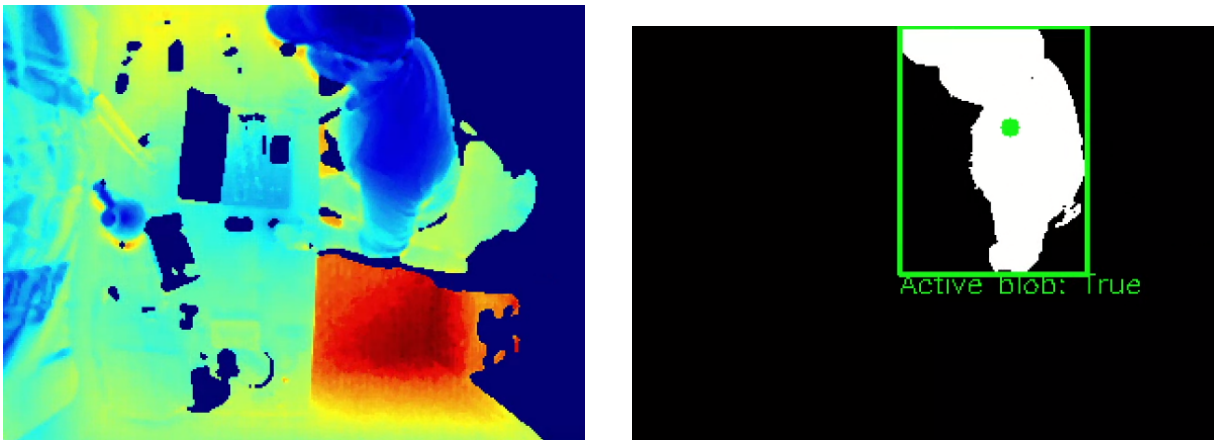


(b) Foreground segmentation with a single moving agent in the scene.

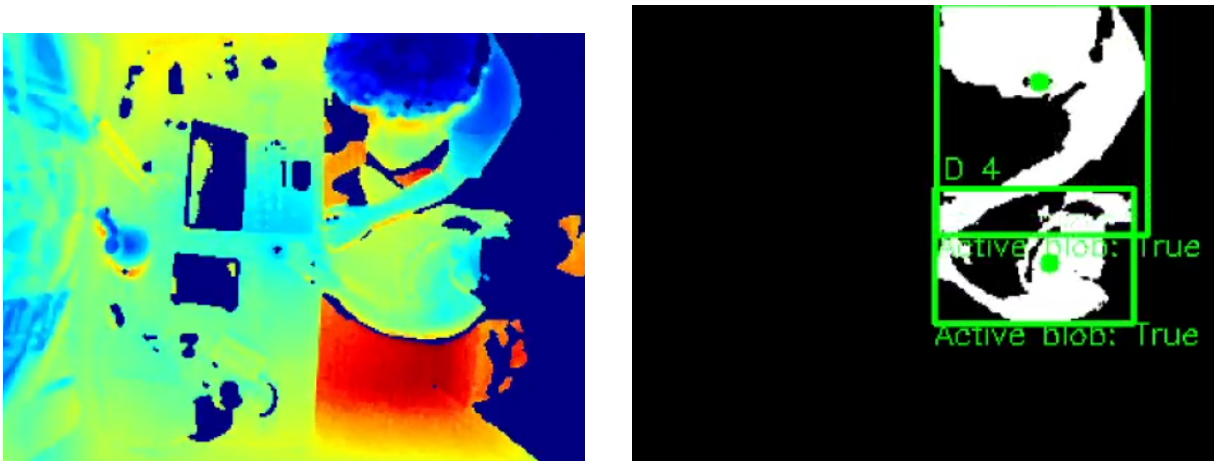


(c) Foreground segmentation with two moving agents in the scene.

Figure 4.3: Background model and foreground segmentation results. Left column shows the cleaned depth image and right column shows the output of the segmentation step.



(a) Blob extraction in the single-agent case.



(b) Blob extraction in the multi-agent case.

Figure 4.4: Blob extraction using connected components. Left column shows the cleaned depth image and right column shows the detected blobs after connected-component labeling and size filtering.

bounding-box aspect ratio exceeds `split_ratio_thresh`. In that case a splitting routine is triggered: it first attempts a depth-based strategy by inverting the depth within the blob region, locating local maxima that correspond to the heads of standing persons, matching these peaks to previously known track centroids, and applying a Watershed transform to partition the blob. If the depth variation within the region is insufficient to locate distinct peaks, a fallback based on the distance transform of the binary silhouette is used. Both strategies return a set of sub-blobs attributed to existing tracks and a *bridge mask* identifying the inter-agent gap region, which is subsequently force-updated in the background model. Algorithm 3 summarizes the full track update logic.

Figure 4.5 shows the splitting routine in action during a handshake: the foreground mask presents a continuous superblob connecting both agents, yet the algorithm recovers separate bounding regions for each agent through depth local-maxima extraction and Watershed partitioning. This splitting process is complemented by superblob handling in the Kalman filter (Section 4.3.2).

Figure 4.5: Superblob splitting during a handshake using depth local-maxima extraction and Watershed. Left: foreground mask showing a continuous superblob. Right: individual sub-blobs recovered for each agent.

The background update sub-routine is presented in Algorithm 4. The eligibility mask `background_mask` restricts updates to pixels that are currently non-foreground or have persisted as foreground beyond the age threshold, while explicitly excluding the footprint of all active tracked blobs to prevent absorbing agents who are temporarily stationary. The update rate switches between a slow rate β_1 under normal conditions and a fast rate $\beta_2 \ll 1$ when the entire scene has been detected as static for a sustained period.

To facilitate parameter tuning, the preprocessing node can be launched via `preprocessing.launch`, which exposes the four tunable parameters as command-line arguments, allowing them to be overridden at launch time without modifying any source file or rebuilding the catkin workspace.

Algorithm 3 UpdateTracks

Input: `blobs`, `depth`
`updated` $\leftarrow \{\}$; `static_count` $\leftarrow 0$; `bridge_mask` $\leftarrow \emptyset$
for each blob $b \in$ `blobs` **do**
  $j^* \leftarrow \arg\min_j \|b.\text{centroid} - \text{track}_j.\text{centroid}\|$ subject to distance $<$ `track_dist_thresh`
  if $j^* = \varnothing$ **or** (IsSuperblob(b) **and** $|\text{tracks}| > |$`blobs`$|$) **then**
    if IsSuperblob(b) **then**
      (`sub_blobs`, `bridge`) $\leftarrow$ SplitBlob(b, tracks, `depth`) ▷ Depth-based; DT fallback
      if split succeeded **then**
        for each (`sub`, `tid`) in `sub_blobs` **do**
          Update $\text{track}_{\texttt{tid}}$ with sub
        end for
        `bridge_mask` $\cup=$ `bridge`; **continue**
      end if
    end if
    Create Track(next_id, b); next_id += 1
  else
    Update track_{j^*} with b ▷ Centroid, bbox, history
  end if
  Update static flag of track ▷ Centroid motion and bbox-area change over sliding window
  if track is static **then** static_count += 1
  end if
end for
Drop tracks where `frame_count` $-$ `last_seen` $\geq$ `track_timeout`
full_update $\leftarrow$ (`static_count` $=$ $|$`blobs`$|$ sustained for $>$ `static_scene_thresh` frames)

Algorithm 4 UpdateBackground

Input: `depth`, `fg_mask`
`background_mask`$\leftarrow$ $\big[(\texttt{fg\_mask} = 0) \vee ((\texttt{fg\_age} > \theta_{\text{age}}) \wedge \neg \texttt{track\_mask})\big] \wedge$ valid(`depth`)
$\beta_t \leftarrow \beta_2$ **if** *full_update* **else** β_1
`background`[`background_mask`] $\leftarrow \beta_t \cdot$ `background`[`background_mask`] $+ (1 - \beta_t) \cdot$ `depth`[`background_mask`]

4.3.2 Tracking

The `tof_preprocessing` tracking node subscribes to the `/preproc/blobs` blob array and publishes Kalman-filtered agent states to `/tracking/kalman`. Each agent encapsulates a `cv2.KalmanFilter` instance initialized with position and zero velocity, and is updated each frame with the matched blob centroid as measurement. The bounding box is maintained separately via EMA smoothing ($\alpha = 0.7$) rather than through the Kalman state, since box dimensions do not follow a linear motion model. Table D.1 lists the configurable parameters of the node.

The Kalman Filter is initialized with a diagonal process noise covariance

$$Q = \text{diag}(5,\ 5,\ 1000,\ 1000) \tag{4.1}$$

and measurement noise covariance

$$R = \text{diag}(10,\ 10) \tag{4.2}$$

The high velocity entries in Q reflect that velocity is unobservable at initialization and adapts over time, while the position entries are kept small since centroid measurements are available each frame. The error covariance is initialized to

$$P_0 = I_4 \tag{4.3}$$

Algorithm 5 describes the top-level per-frame callback. The prediction step runs before association and uses the actual Δt from the message timestamp, clamped to $[0.01,\ 0.2]$ s. The cost matrix is built by evaluating the Mahalanobis distance for every agent-blob pair (Algorithm 6). Pairs exceeding `maha_thresh` receive infinity cost and are excluded from assignment. The Hungarian algorithm then finds the globally optimal matching.

Algorithm 7 details agent creation. Before spawning a new agent, the blob's centroid is classified as BOUNDARY or CENTER based on its proximity to the image edge. For CENTER-spawned blobs, two additional guards prevent spurious agent creation. A scene-motion check that suppresses creation when all existing agents are nearly static (the blob is most likely a sensor artifact), and a proximity check that suppresses creation when an existing agent is already nearby (the blob is likely a fragment of that agent, i.e. an arm). BOUNDARY-spawned blobs bypass these guards since they represent persons entering the field of view.

After all assignments are resolved, a duplicate-merging pass inspects all agent pairs and removes redundant tracks. Three main triggers are checked: (i) bounding-box IoU exceeds `iou_thresh` and centroids are within `duplicate_centroid_thresh` (significant overlap with nearby centroid); (ii) centroid distance is below `strict_close_centroids_thresh` regardless of overlap (agents have converged to the same position); (iii) the size ratio between the two bounding boxes is below `size_ratio_thresh` and centroids are within `loose_close_centroid_thresh` (one agent is a small fragment of the other). For CENTER-spawned agents with age below `agent_legal_age`, the thresholds are relaxed

Algorithm 5 Tracking node — main per-frame callback

Input: `BlobArray` ▷ Custom ROS message. See Appendix F for message definitions.
$\Delta t \leftarrow$ clamp(`now` − `agent.last_seen`, 0.01, 0.2) for each agent
Predict all agents: $\mathbf{x}_{k+1|k} \leftarrow F(\Delta t)\,\mathbf{x}_{k|k}$
if no agents **then** create one agent per blob (Algorithm 7); **return**
end if
if no blobs **then** mark all agents missed; `Cleanup`; **return**
end if
$C \leftarrow$ `BuildCostMatrix`(agents, blobs) ▷ Algorithm 6
(rows, cols) $\leftarrow$ `HungarianAssignment`(C)
`blob_to_agents` $\leftarrow$ group valid assignments ($C[r,c] \leq$ `maha_thresh`) by blob index
if |agents matched by Hungarian| < |agents| **then** ▷ Possible superblob
    for each blob j **do**
        `close` $\leftarrow \{i \mid C[i,j] \leq$ `maha_thresh`$\}$
        if |`close`| ≥ 2 **and** all centroids of `close` inside $\texttt{blob}_j$`.bbox` **then**
            `blob_to_agents`$[j] \leftarrow$ `close` ▷ Override: confirmed superblob
        end if
    end for
end if
for each (j, `agent_list`) in `blob_to_agents` **do**
    if |`agent_list`| = 1 **then** `agent.correct`($\texttt{blob}_j$) ▷ 1-to-1 match
    else ▷ Superblob
        if $\overline{\texttt{conf}} >$ `avg_conf_thresh` **then** keep predictions; decay confidence
        else correct all agents with $\texttt{blob}_j$; decay confidence
        end if
    end if
end for
Mark unmatched agents missed (decay confidence, increment miss counter)
for each unmatched blob b **do**
    if superblob detected this frame **then** skip ▷ Suppress fragment creation
    else `CreateAgent`(b) (Algorithm 7)
    end if
end for
`MergeDuplicateAgents`(); `Cleanup`(); Publish `/tracking/kalman`

Algorithm 6 BuildCostMatrix

Input: agents $\mathcal{A}$, blobs $\mathcal{B}$
$C \leftarrow \infty^{|\mathcal{A}|\times|\mathcal{B}|}$
for each agent $i \in \mathcal{A}$, blob $j \in \mathcal{B}$ **do**
    $\mathbf{y} \leftarrow \mathbf{z}_j - H\,\mathbf{x}_{k+1|k}^{(i)}$ ▷ Innovation
    $S \leftarrow H P^{(i)} H^\top + R$ ▷ Innovation covariance
    $d \leftarrow \sqrt{\mathbf{y}^\top S^{-1}\mathbf{y}}$ ▷ Mahalanobis distance
    if $d \leq$ `maha_thresh` **then** $C[i,j] \leftarrow d$
    end if
end for
return C

Algorithm 7 CreateAgent

Input: blob b, current agents $\mathcal{A}$
`spawn_type` $\leftarrow$ BOUNDARY if centroid within `boundary_thresh` of any edge, else CENTER
if `spawn_type` = CENTER **then**
    $v_{\text{scene}} \leftarrow \text{mean}(\|\mathbf{v}_i\|)$ over all agents $\mathcal{A}$ ▷ Average scene velocity
    if $v_{\text{scene}} <$ `static_scene_threshold` **and** $|\mathcal{A}| > 0$ **then return** ▷ Static scene, likely artifact
    end if
    if $\min_i \|b.\text{centroid} - \mathbf{x}_i\| <$ `proximity_thresh` **then return** ▷ Too close to existing agent
    end if
end if
Initialize Agent(next_id, b) in CANDIDATE state with $\mathbf{x}_0 = [b.c_x,\ b.c_y,\ 0,\ 0]^\top$
next_id += 1

further to aggressively absorb suspicious early-birth agents. In all cases, the boundary-spawned or older agent is retained and absorbs a fraction of the discarded agent's confidence.

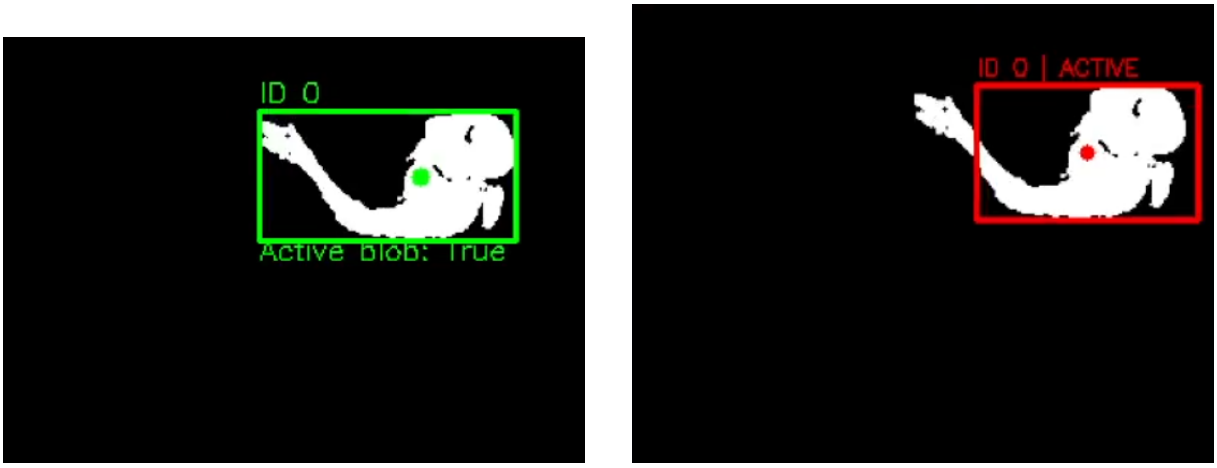
The `Cleanup` step removes agents that have been in LOST state for more than 2 s, whose confidence has fallen below `agent_cleanup_conf`, or who are young CENTER-spawned agents with confidence below `agent_low_conf_thresh`.

Figure 4.6 compares the blob detections from the preprocessing stage with the corresponding Kalman tracker output. The bounding boxes differ slightly due to EMA smoothing applied to the tracked dimensions. Figure 4.7 shows stable tracking of a walking agent across time, demonstrating consistent identity assignment and correct motion prediction.

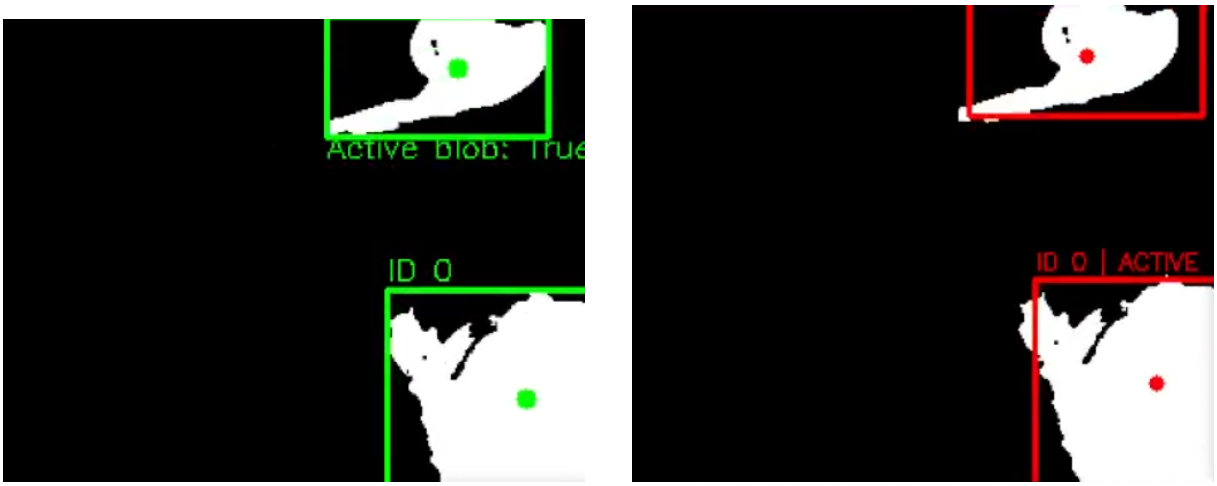
Figure 4.8 illustrates agent persistence under missed detections. Figure 4.8a shows the preprocessing stage failing to detect an agent, while the Kalman filter continues estimating its position from the motion model alone. After a few frames the agent reappears in the foreground mask and the filter resumes correction based on observations (Figure 4.8b).

Finally, Figure 4.9 shows the necessity of the superblob handling routine in the Kalman filter. When a prolonged handshake prevents the preprocessing stage from splitting the merged blob, a superblob is forwarded to the tracker. The filter detects the superblob condition, discards the observation, and trusts the motion model to continue estimating the position of both agents independently.

As with the preprocessing node, to maintain a modular design and easier parameter modification, the tracking node can be launched via `tracking.launch`.



(a) Kalman tracking in the single-agent case.



(b) Kalman tracking in the multi-agent case.

Figure 4.6: Kalman tracker output. Left column shows the input detected blobs and right column shows the Kalman filter estimates.

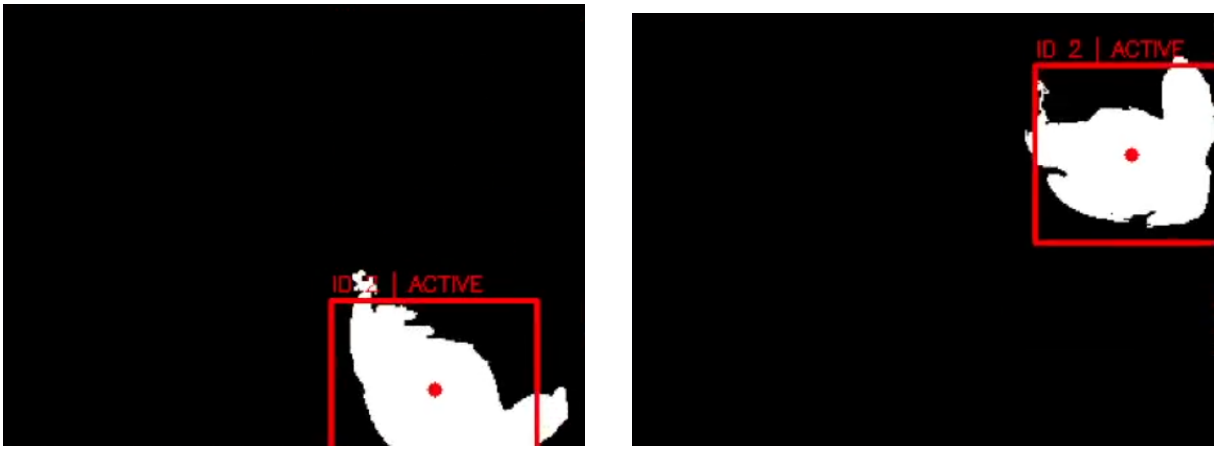


Figure 4.7: Kalman tracker estimating the changing position of a moving agent across time.

(a) Missed blob in the foreground mask. The Kalman filter continues estimating from the motion model.

(b) The agent is detected again and the filter resumes correcting the motion model from observations.

Figure 4.8: Track persistence through missed detections and spontaneous disappearances of agents.

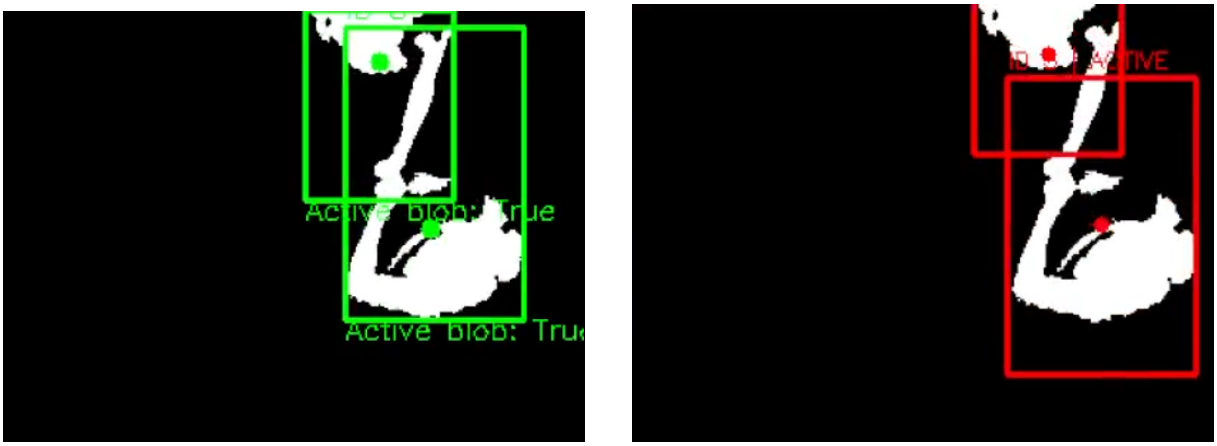
4.4 IMU-Vision fusion and authority selection

4.4.1 Reading data from IMU

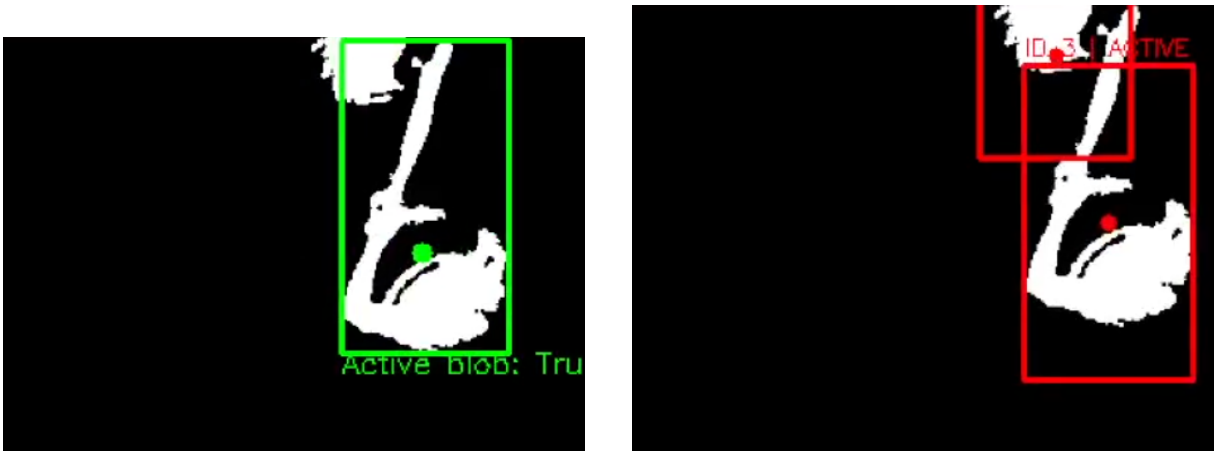
The ESP32 microcontroller runs a MicroPython script that reads raw 16-bit integer values from the MPU-6050 over I2C and transmits them as a JSON-encoded UDP datagram at approximately 50 Hz. The `mpu_reader` ROS node listens on a configured UDP port, validates each received packet, converts raw ADC values to physical SI units, and publishes a `sensor_msgs/Imu` message to `/imu/data_raw`. Table H.1 lists its configurable parameters.

Algorithm 8 presents the main receive loop. Source-IP filtering rejects packets from unexpected senders, providing a basic integrity guard in environments with multiple wireless devices on the same network. Packet integrity is validated by checking that all six required sensor keys are present before attempting any conversion. Diagonal covariance matrices are populated in the published message as required by the ROS IMU convention. When `esp32_ip` is left empty, packets from any source are accepted, which is useful during development.

The raw IMU message is passed through the external `imu_filter_madgwick` node [21], which estimates sensor orientation and republishes a completed message to `/imu/data` with the orientation quaternion populated. This filtered topic is the input consumed



(a) Kalman tracker response when blob splitting succeeds.



(b) Kalman tracker response when blob splitting is unsuccessful.

Figure 4.9: Superblob handling in the Kalman tracker. Left column shows the preprocessing blob output and right column shows the tracker response.

Algorithm 8 mpu_reader — main receive loop

Bind UDP socket to `0.0.0.0:udp_port` with 1 s timeout
while not shutdown **do**
    (`data`, `addr`) ← `socket.recvfrom`(1024) ▷ retry on timeout
    if `esp32_ip` set **and** `addr` ≠ `esp32_ip` **then reject**
    end if
    `raw` ← JSON.parse(`data`)
    if any of {AcX, AcY, AcZ, GyX, GyY, GyZ} missing **then skip**
    end if
    $a_i \leftarrow (\texttt{raw}[\mathrm{Ac}_i]/16384.0) \cdot g_0, \quad i \in \{x, y, z\}$
    $\omega_i \leftarrow \mathrm{radians}(\texttt{raw}[\mathrm{Gy}_i]/131.0), \quad i \in \{x, y, z\}$
    Build `Imu` msg: set `linear_acceleration`, `angular_velocity`, diagonal covariances
    Publish to `/imu/data_raw`
end while

by both the fusion and gesture nodes. It is worth noting that Madgwick filter can yield even more accurate results through magnetometer compensation (not available with MPU 6050).

4.4.2 IMU-Vision Fusion

This node implements the authorization logic described in Section 3.6. The `fusion` node subscribes to `/imu/data` at full rate and to a synchronized pair of `/preproc/foreground_mask` and `/tracking/kalman` via `ApproximateTimeSynchronizer`. The IMU subscription uses a large queue to ensure that fast 50 Hz messages are never dropped. The node publishes the authorized agent and confidence to `/fusion/authorized_agent_id` and `/fusion/authorized_agent_confidence`, and per-agent scores `/fusion/confidence`. A camera-geometry calibration step at startup computes the pixels/m conversion factor from the configured camera height and field of view, used to express agent speed in m/s. Table I.1 summarizes all configurable parameters.

Algorithm 9 describes the IMU callback. It runs at the full IMU rate and is responsible for maintaining the rolling buffer in a consistent state. The actual IMU rate is computed through IMU message timestamps and used downstream to detect sparse windows.

Algorithm 9 fusion — IMU callback

Input: `Imu` msg ▷ Content: orientation q, linear acceleration $\mathbf{a}$, angular velocity $\boldsymbol{\omega}$
Update IMU rate estimate from inter-arrival interval ▷ Validated range: 10–100 Hz
$R \leftarrow \text{QuaternionToRotation}(q)$
$\mathbf{g}_s \leftarrow R^T [\,0,\ 0,\ g_0\,]^T$ ▷ Gravity vector rotated into sensor frame
$\tilde{\mathbf{a}} \leftarrow \mathbf{a} - \mathbf{g}_s$ ▷ Gravity-compensated acceleration
`imu_buffer`.append$\big(t,\ \|\tilde{\mathbf{a}}\|,\ \|\boldsymbol{\omega}\|\big)$
Prune entries where $t_k < t - T_w - T_{\text{margin}}$

Algorithm 10 presents the agent callback, which drives the full scoring and authorization pipeline. The foreground energy for an agent is computed by extracting the bounding box ROI from the foreground mask, counting active foreground pixels, and tracking the frame-to-frame absolute change against an adaptive normalization ceiling that tracks the maximum observed value and decays slowly. The temporal cross-correlation is computed independently on both channels by interpolating the visual signals to the IMU time grid before evaluating the normalized cross-correlation.

The `SelectAuthorized` procedure implements the authorization logic described in Section 3.6. All evaluated agents' EMA confidence scores are updated first. The best-scoring agent is identified, and hysteresis and switch-margin checks are applied before accepting a potential switch. Overall six cases are handled in order:

- No scores: Decay all confidences, retain or de-authorize current agent.
- Reentry window active: Require θ_{re}.

Algorithm 10 fusion — agent callback

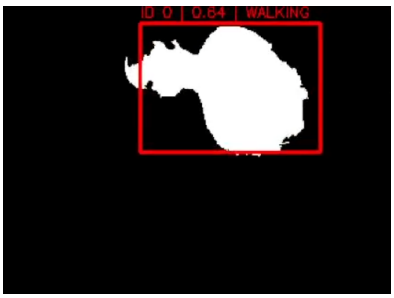
Input: synchronized `fg_msg`, `agent_msg`; $t \leftarrow$ `agent_msg.stamp`
`imu_win` $\leftarrow \{(t_k, a_k, \omega_k) \in$ `imu_buffer` $\mid t - T_w \le t_k \le t\}$
if $|$`imu_win`$| < 5$ **then return** ▷ Insufficient IMU data
end if
$\bar{a} \leftarrow \mathrm{RMS}(\{a_k\})$; $\bar{\omega} \leftarrow \mathrm{RMS}(\{\omega_k\})$
$\hat{a} \leftarrow \min(\bar{a}/a_{\max}, 1)$; $\hat{\omega} \leftarrow \min(\bar{\omega}/\omega_{\max}, 1)$
$E_{\mathrm{bal}} \leftarrow w_a\,\hat{a} + (1 - w_a)\,\hat{\omega}$
if $E_{\mathrm{bal}} < \epsilon$ **then**
    SelectAuthorized($\varnothing$, E_{bal}); **return** ▷ IMU inactive: skip scoring
end if
Append $\hat{a}$, $\hat{\omega}$ to long-term IMU history deques
for each agent i with `age` $\ge$ `min_agent_age` **do**
    if not ShouldEvaluate(i) **then skip** ▷ Exploration–exploitation
    end if
    Update speed buffer; $\hat{v} \leftarrow \min(\mathrm{RMS}(\{v_k\})/v^i_{\max}, 1)$; update adaptive $v^i_{\max}$
    $\hat{f} \leftarrow$ ComputeFGEnergy(i, `fg_mask`) ▷ ROI fg change, adaptive normalization
    `state` $\leftarrow$ ClassifyActivity($\hat{v}$, $\hat{f}$) ▷ Vision-only: WALKING/GESTURE/IDLE/AMBIGUOUS
    $E \leftarrow w_a\, e^{-\lambda(\hat{v}-\hat{a})^2} + (1 - w_a)\, e^{-\lambda(\hat{f}-\hat{\omega})^2}$
    $E \leftarrow E \cdot$ ActivityMultiplier(`state`, $\hat{a}$, $\hat{\omega}$, $\hat{v}$, $\hat{f}$)
    $\rho \leftarrow$ ComputeTemporalScore(i, t) ▷ Dual-channel cross-correlation with lag tolerance
    $r \leftarrow$ ComputeActivityCorrelation(i) ▷ Long-term Pearson correlation bonus
    if $\rho \neq \varnothing$ **then**
        $s \leftarrow (1 - w_t)\,E + w_t\,\max(\rho,\ E \cdot 0.5)$ ▷ Floor temporal score to avoid dragging
    else $s \leftarrow E$; $r \leftarrow \min(r, r_{\max})$ ▷ Cap bonus when no temporal score
    end if
    `scores`$[i] \leftarrow \min\big(s \cdot (1 + r),\ s_{\max}\big)$
end for
SelectAuthorized(`scores`, E_{bal}); Publish all results; Cleanup

- Hysteresis or cooldown active: Require m-fold margin for a switch.
- Best agent above θ_{auth}: Authorize.
- Current agent confidence between θ_{flexible} and θ_{auth}: Retain.
- Nobody above any threshold: De-authorize.

The `ShouldEvaluate` function implements the exploration–exploitation policy. When the authorized agent's confidence exceeds `exploit_high_conf`, competitors are evaluated with probability `exploit_high_prob`. If confidence is high but there's still a degree of uncertainty then competitors are evaluated with `exploit_med_prob`. Otherwise, all agents are always evaluated.

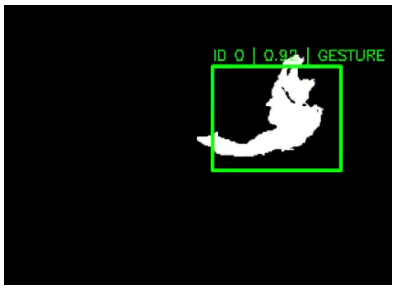
A `Cleanup` routine runs each callback to delete agents unseen for a timeout clock, flagging an exit event if the deleted agent was the authorized one. Adaptive normalization ceilings for inactive for a time are reset to their defaults preventing stale ceilings from persisting into subsequent tracking sessions.

Figure 4.10 shows the confidence build-up process for an agent. As the agent enters the scene confidence is low (Figure 4.10a), but as the agent moves and IMU–vision correlation evidence accumulates the system builds confidence and grants authorization (Figure 4.10b). Once authorized, the agent retains its status provided it remains active; prolonged static presence risks absorption into the background model (Figure 4.10c).



(a) Agent enters the scene. Confidence is low.

(b) Agent builds up confidence and gets authorized.



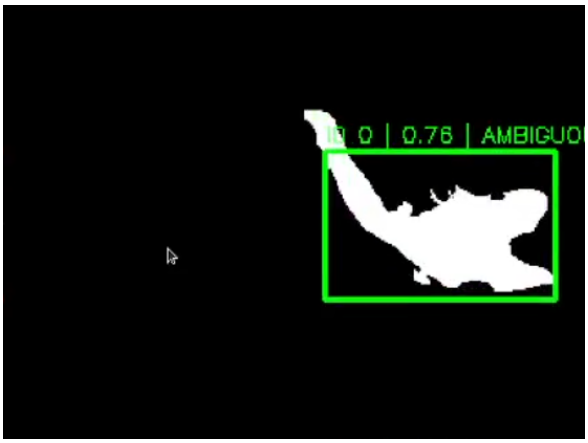
(c) Agent keeps authorized.

Figure 4.10: Confidence build-up process as the agent interacts with the environment.

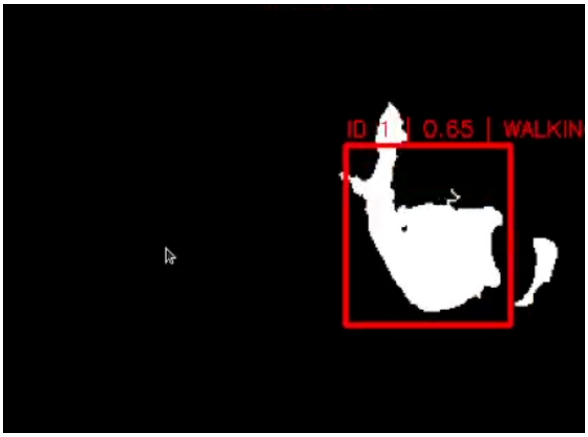
Figure 4.11 shows the system behavior in the single-agent case. When the authorized agent is in the scene it is granted authorization after building up confidence (Figure 4.11a). If the agent present has no IMU or its movement does not correlate with any IMU signal, it remains unauthorized (Figure 4.11b).

Figure 4.12 shows the multi-agent case. When agents enter the scene the system cannot yet determine who is authorized, so the safe default is to keep all agents unauthorized (Figure 4.12a). As correlation evidence accumulates, the agent with the strongest confidence is authorized provided it satisfies the authorization policies (Figure 4.12b).

Figure 4.13 shows an example of correct authorization handling in a three-agent scenario. Segmentation quality degrades with more agents present, and overall performance tends



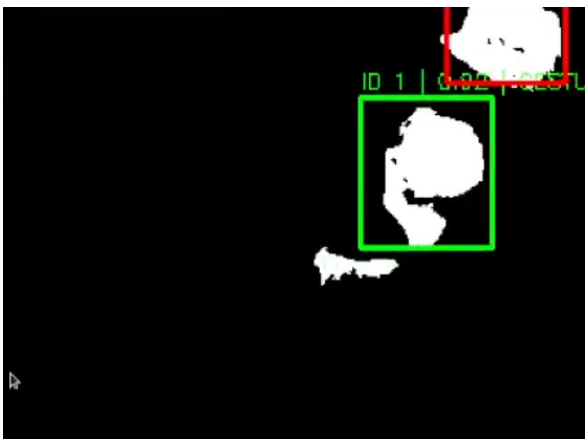
(a) Only authorized agent in scene.



(b) Only unauthorized agent in scene.

Figure 4.11: Single-agent interaction with the system.

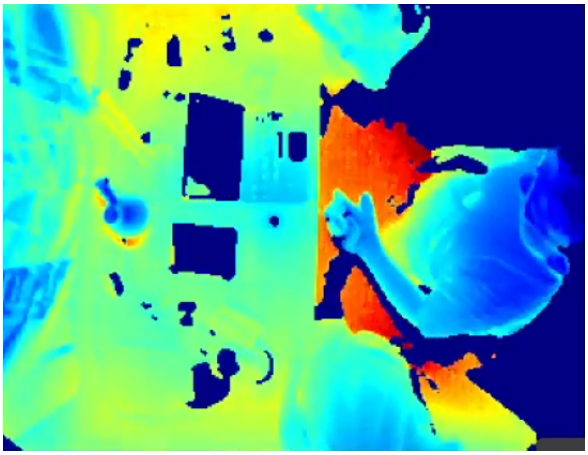
(a) Multi-agent case: no agent authorized.



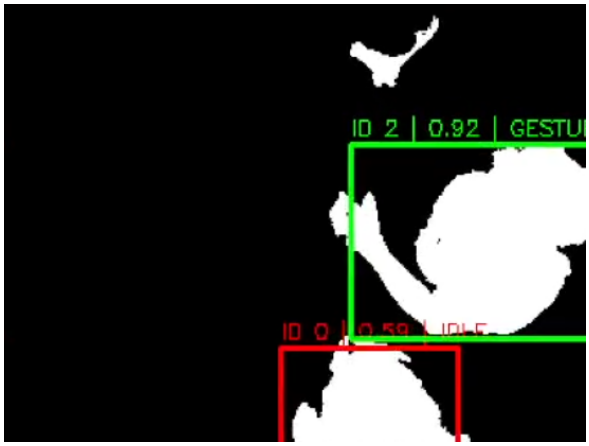
(b) Multi-agent case: one agent authorized.

Figure 4.12: Multi-agent interaction with the system.

to be lower than in the two-agent and single-agent cases. This behavior is discussed in detail in Section 5.5.



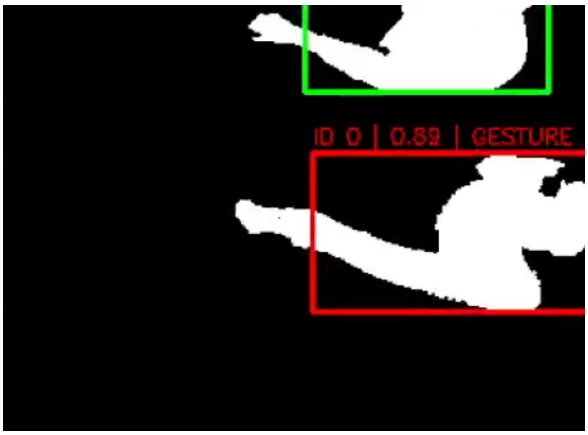
(a) Three-agent case: no agent authorized.



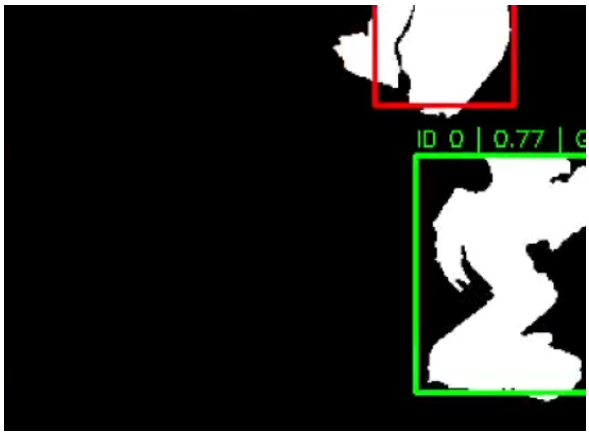
(b) Three-agent case: one agent authorized.

Figure 4.13: Authorization handling in a three-agent scenario.

Figure 4.14 shows the most challenging scenario the system faces: two agents performing similar movements simultaneously. In Figure 4.14a the synchronized motion transfers authorization to the unauthorized agent. In Figure 4.14b the same situation occurs but this time the authorized agent successfully retains authority. This behavior is discussed in detail in Section 5.5.



(a) Authorization transferred through similar simultaneous movements.



(b) Authorized agent retains authorization successfully.

Figure 4.14: Authorization handling under similar simultaneous movement patterns.

4.5 Gesture Segmentation and Recognition

The `gesture` node implements both gesture segmentation and DTW-based recognition in a single callback-driven node, corresponding to the designs described in Sections 3.7

and 3.8. It supports two operating modes selected at launch: *recognition*, where captured trajectories are classified against the loaded template library, and *record*, where captured trajectories are saved to disk as new templates for specified gesture class. When `identity_verification` is enabled, the node additionally subscribes to `/fusion/authorized_agent_id` and discards any gesture whose onset snapshot carried no authorized agent. Table J.1 lists all configurable parameters.

Algorithm 11 shows the IMU callback. Gravity compensation follows the same quaternion rotation used in the fusion node. The 6D sample is appended to the rolling buffer and the per-sample energy is forwarded immediately to the segmenter, keeping the callback lightweight.

Algorithm 11 gesture — IMU callback

Input: `Imu` msg ▷ Content: orientation q, $\mathbf{a}$, $\boldsymbol{\omega}$
Update IMU rate from inter-arrival time ▷ Validated range: 10–100 Hz
$R \leftarrow \text{QuaternionToRotation}(q)$; $\mathbf{g}_s \leftarrow R^T[0, 0, g_0]^T$
$\tilde{\mathbf{a}} \leftarrow \mathbf{a} - \mathbf{g}_s$; `sample` $\leftarrow [\tilde{a}_x, \tilde{a}_y, \tilde{a}_z, \omega_x, \omega_y, \omega_z]$
`buffer`.append(t, `sample`) ▷ Rolling deque, maxlen $= \lceil T_{\text{buf}} \cdot f_{\text{IMU}} \rceil$
$\hat{a} \leftarrow \min(\|\tilde{\mathbf{a}}\|/a_{\max}, 1)$; $\hat{\omega} \leftarrow \min(\|\boldsymbol{\omega}\|/\omega_{\max}, 1)$
$E \leftarrow w_a\,\hat{a} + (1 - w_a)\,\hat{\omega}$
UpdateSegmenter(E, `sample`)

Algorithm 12 details the segmentation state machine. At the IDLE→ACTIVE transition, the gesture buffer is initialized with a lookback slice from the rolling buffer, capturing the pre-onset ramp. The authorization snapshot is taken at the same instant.

`ProcessGesture` normalizes the captured sequence per-axis and either saves it as a numbered `.npy` file under `template_dir/<gesture_name>/` in record mode, or forwards it to the classifier in recognition mode. In the second case, if `identity_verification` is enabled, online gesture recognition is performed. When `auth_snapshot` equals -1, the gesture is silently discarded before classification.

Algorithm 13 presents the DTW classification procedure. All pairwise Euclidean distances between the query and each template are precomputed in a single vectorized cdist call before the dynamic programming loop, avoiding redundant per-cell distance evaluation inside the band.

Per-class thresholds are loaded at startup for any class whose parameter `dtw_threshold_<name>` is set, with the global `dtw_threshold` as fallback. This lets individual gestures be tuned independently after observing recognition performance, without modifying any source file or rebuilding the workspace.

After performing classification the output of the DTW algorithm is logged using `rospy.loginfo`. The logged information includes score per class, class achieving the lowest score, and the matching template if cost is below the threshold. In case of match, a string with the matched class is published downstream through the topic `/gesture/recognized`.

Figure 4.15 illustrates the gesture segmentation and recognition working together with the authorization system. When an authorized agent executes a gesture, the FSM begins in

Algorithm 12 UpdateSegmenter

Input: energy E, current sample `s`
if `state` = IDLE **then**
  if $E > \theta_{\text{onset}}$ **then** `onset_counter` += 1
    if `onset_counter` $\geq N_{\text{onset}}$ **then**
      `state` ← ACTIVE; `onset_counter, offset_counter` ← 0
      `auth_snapshot` ← `authorized_agent_id` ▷ Snapshot before motion ends
      `gesture_samples` ← `buffer`$[-(N_{\text{onset}} +$ `lookback`$)$:].`samples`
      Publish `"ACTIVE"` to `/gesture/state`
    end if
  else `onset_counter` ← 0
  end if
else if `state` = ACTIVE **then**
  `gesture_samples`.append(`s`)
  if |`gesture_samples`| $\geq L_{\max}$ **then**
    Reset to IDLE; discard; Publish `"IDLE"`; **return** ▷ Too long: not a gesture
  end if
  if $E < \theta_{\text{offset}}$ **then** `offset_counter` += 1
    if `offset_counter` $\geq N_{\text{offset}}$ **then**
      `state` ← IDLE; Publish `"IDLE"`
      `gesture` ← `gesture_samples`$[: -N_{\text{offset}}]$ ▷ Trim low-energy tail
      if |`gesture`| $\geq L_{\min}$ **then** ProcessGesture(`gesture`)
      end if
    end if
  else `offset_counter` ← 0
  end if
end if

Algorithm 13 Classify

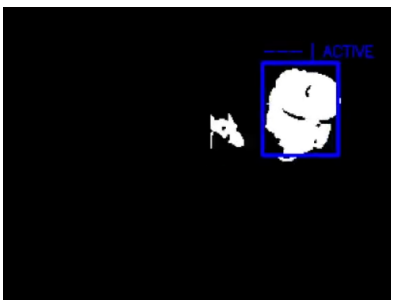
Input: normalized query $G \in \mathbb{R}^{N\times 6}$; $r \leftarrow \max(1, \lfloor \rho N \rfloor)$
`best_dist` $\leftarrow \infty$; `best_class` $\leftarrow \varnothing$; `early_stop` $\leftarrow$ False
for each class c (excluding `ood`) **do**
  for each template $T \in \mathcal{T}^{(c)}$ **do**
    if $|N - |T|| > r$ **then skip** ▷ Length outside band: DTW $= \infty$
    end if
    $D \leftarrow$ `cdist`(G, T, euclidean) ▷ $(N \times |T|)$ pairwise distance matrix
    $d \leftarrow$ BandDTW(D, r) / $(N + |T|)$ ▷ Band-constrained DP, path-normalized
    if $d <$ `best_dist` **then** `best_dist` $\leftarrow d$; `best_class` $\leftarrow c$
    end if
    if $d < \theta_{\text{strict}}$ **then** `early_stop` $\leftarrow$ True; **break**
    end if
  end for
  if `early_stop` **then break**
  end if
end for
$\theta \leftarrow$ `dtw_thresholds`.get(`best_class`, θ_{DTW})
if `best_class` $\neq \varnothing$ **and** `best_dist` $< \theta$ **then**
  Publish `best_class` to `/gesture/recognized`
else Log rejection ▷ OOD: no class below threshold
end if

IDLE state (Figure 4.15a), which transitions to *ACTIVE* as soon as enough consecutive onset measurements are detected (Figure 4.15b). After the segmenter moves the state back to *IDLE*, the recognition system is triggered with the captured trajectory as input. If the trajectory is matched to an existing gesture class, the visualization shows the detected class name (Figure 4.15c). In the OOD case the placeholder text "---" is displayed along with the *IDLE* state indicator. Since classification only runs after the segment is complete, the recognized class and the *ACTIVE* state cannot appear simultaneously in the visualization (*ACTIVE*→*IDLE* transition).

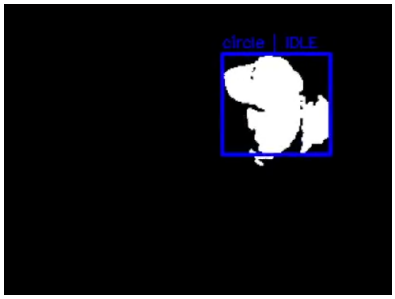
4.6 Visualizations

Six dedicated visualization nodes are provided for system monitoring and debugging. Table 4.3 summarizes the six nodes, their host package, the topics they consume, and the information they render. All screenshots in this report were captured from this visualization layout.

(a) Agent in *IDLE* state before triggering a gesture.



(b) *IDLE*→*ACTIVE* transition upon N_{on} consecutive onset measurements.



(c) *ACTIVE*→*IDLE* transition upon N_{off} consecutive offset measurements and gesture recognition output.

Figure 4.15: Gesture segmentation FSM states and recognition output in the online system.

Table 4.3: Visualization nodes and their properties.

Script	Package	Topic(s)	What is rendered
vis_depth_clean.py	tof_preprocessing	/preproc/depth_clean	Cleaned depth frame colormapped with COLORMAP_JET; invalid pixels set to black.
vis_fg_mask.py	tof_preprocessing	/preproc/foreground_mask	Binary foreground mask converted to BGR for display.
vis_blobs.py	tof_preprocessing	/preproc/foreground_mask, /preproc/blobs	Foreground mask with each detected blob outlined and labelled with its bounding-box dimensions.
vis_kalman.py	tof_preprocessing	/preproc/foreground_mask, /tracking/kalman	Foreground mask with a bounding rectangle drawn around every tracked agent; each box is annotated with the agent ID.
vis_conf.py	mpu	/preproc/foreground_mask, /tracking/kalman, /fusion/confidence, /fusion/authorized_agent_id	Foreground mask with per-agent bounding boxes colored green for the authorized agent and red for all others; each box is labeled with its ID, confidence score, and authorization state.
vis_gesture.py	gesture	/preproc/foreground_mask, /tracking/kalman, /fusion/authorized_agent_id, /gesture/recognized, /gesture/state	Foreground mask with the authorized agent's bounding box and a text overlay showing the most recently recognized gesture class (held for gesture_display_duration seconds) alongside the current segmentation FSM state.

Chapter 5

Evaluation

5.1 Experimental Setup

This chapter evaluates the end-to-end performance of the identity-verified gesture recognition system described in Chapters 3 and 4 (see Section 3.1 for a system-level overview). To quantify how well the system performs, data was collected from multiple participants during live interactions and stored as ROS bag recordings (`.bag`), which capture all the topics published from the TOFcam660 (see Table 4.2) and raw IMU measurements, stored as `IMU messages`, in parallel. These recordings can be replayed deterministically at any time, enabling controlled and reproducible evaluation without requiring the hardware to be present.

The evaluation is divided into three parts: offline evaluation of the DTW gesture classifier in isolation, an online evaluation of the full pipeline for both single- and multi-agent scenarios using pre-recorded rosbag replays, and a qualitative discussion of observed performance, edge cases, and privacy properties.

5.1.1 Recording Environment

All recordings were made in an indoor room of approximately $13m \times 13m$ with a ceiling height of approximately $2.5m$. The room has both natural light through a window and artificial light from a ceiling bulb. Recordings were made under a combination of both sources, artificial light only, and natural light only, providing this way diversity in illumination conditions. The TOFcam660 was mounted overhead at approximately 1.9 m, centered on the interaction zone, and oriented perpendicular to the floor. The IMU unit was worn on the right wrist of each participant and connected to the host PC via Wi-Fi UDP at 50 Hz. Since gesture authorization relies on wrist-worn IMU data, all registered gestures were performed with the IMU-wearing arm.

The camera coverage of the scene is approximately $5.23m \times 3.02m$. Considering the Wide Field-of-View (WF) configuration used in the TOFcam660 with Field-of-View (FoV) of

108×77 (horizontal FoV $\times$ vertical FoV) and the mounting height h, the coverage can be computed as shown in equations 5.1 and 5.2.

$$W = 2h \tan\left(\frac{\theta_H}{2}\right) = 2 \times 1.9 \times \tan(54) \approx 5.23\text{m} \tag{5.1}$$

$$H = 2h \tan\left(\frac{\theta_V}{2}\right) = 2 \times 1.9 \times \tan(38.5) \approx 3.02\text{m} \tag{5.2}$$

5.1.2 Data Collection

Eleven participants took part in the data collection, with a gender split of 45% of female and 55% male (5F / 6M). Since the TOFcam660 is a depth sensor, the participant height is an important factor. Participant heights ranged from $1.58m$ to $1.93m$, this range is also relevant for gesture recognition because height correlates with arm length and therefore with gesture trajectory amplitude. The z-score normalization described in Section 4.5 is intended to provide robustness against this variability. The participant pool also included diversity in body type, which affects foreground blob dimensions.

A total of 57 rosbags were selected for system evaluation: 28 for single-agent scenarios and 29 for multi-agent scenarios. Bags in which the agent entered the scene during background model acquisition were discarded, as the agent would not be correctly separated from the background (see Sections 5.1.2.2 and 5.1.2.3). For the gesture classifier, 493 template recordings were retained after quality control (Section 5.1.2.1).

5.1.2.1 Gestures

Two gesture classes were defined for the proof-of-concept evaluation. The *circle* class is a closed circular movement in the plane parallel to the camera frame, performed in a clockwise direction. Figure 5.1 illustrates a sequence of this gesture from the TOFcam660 perspective. The *arm_up_down* class is a full upward then downward sweep in the plane perpendicular to the camera frame. Figure 5.2 shows the gesture sequence from the camera perspective and Figure 5.3 provides a visualization from a different perspective. These two classes were selected because they provide complementary motion cues (one is primarily planar and rotational, the other is primarily vertical), making them distinguishable both visually and inertially.

For template recording, each participant was asked to sit in front of a desk positioned beneath the TOFcam660, simulating a typical office environment. Each participant was asked to perform 20 *circle* gestures, 20 *arm-up-down* gestures, and 10 out-of-distribution (OOD) gestures of their own choice, with the instruction that the OOD movements shouldn't resemble either authorized class. The OOD category serves to evaluate the system's ability to reject gestures that do not belong to the authorized set.

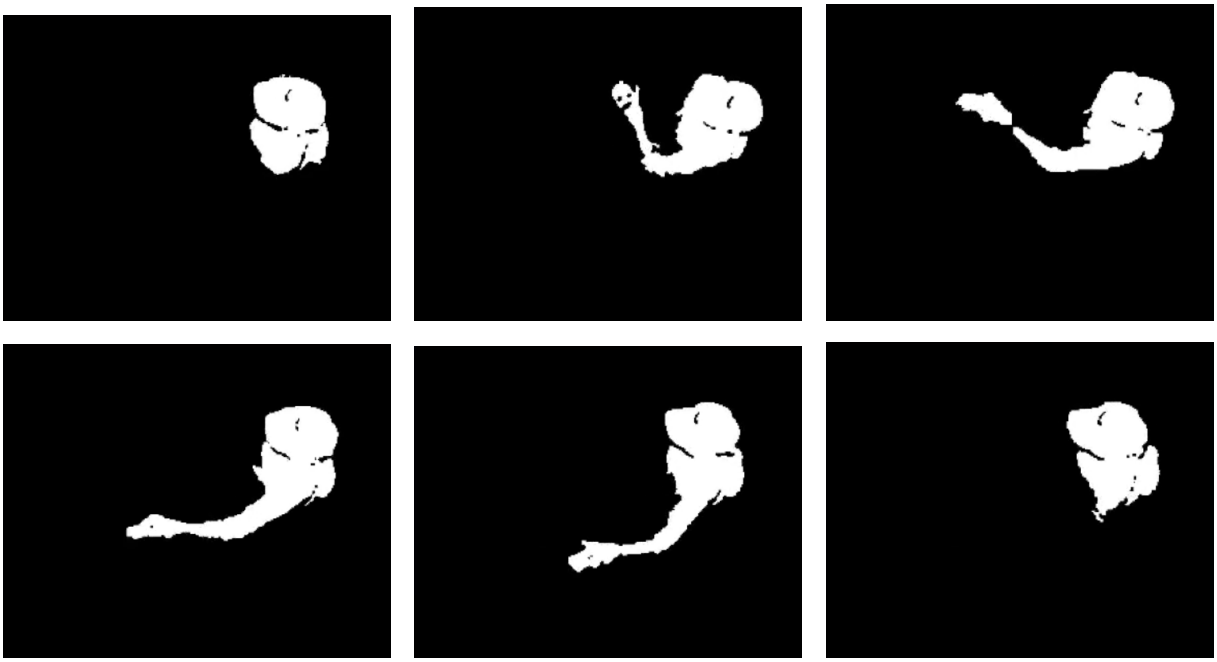


Figure 5.1: Sequence of the *circle* gesture class from the camera perspective. Images were obtained from the foreground segmentation visualization.

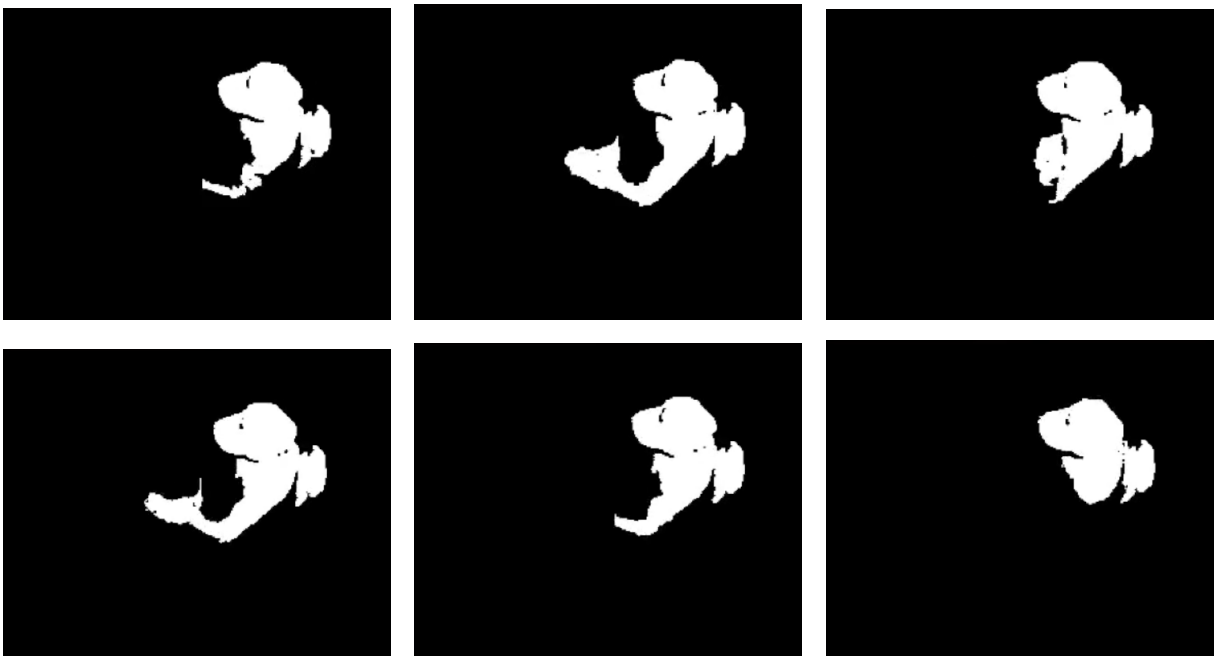


Figure 5.2: Sequence of the *arm_up_down* gesture class from the camera perspective. Images were obtained from the foreground segmentation visualization.

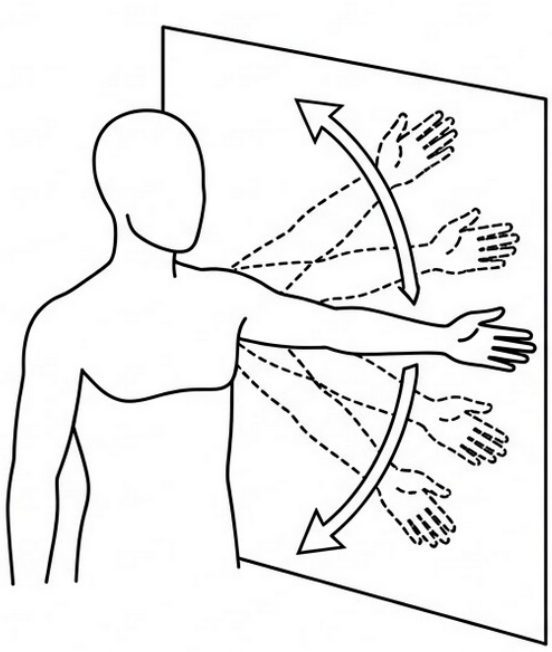


Figure 5.3: Illustration of the *arm_up_down* gesture class movement pattern from a front perspective.

During recording, the TOFcam660 was collecting depth information while the IMU was providing movement measurements (the gesture was always triggered with the IMU wearing arm). The data was stored in a rosbag (`.bag`) and the gesture templates were preprocessed (Section 4.5) and stored as numpy arrays (`.npy`).

It is important that the participant leaves a rest interval of a few seconds between consecutive gestures so that the *ACTIVE→IDLE* transition can complete and the `.bag` and `.npy` files can be correctly written to disk. Gestures performed continuously without sufficient rest time in between are merged into a single long trajectory, which is discarded by the upper-length constraint (Section 3.7). A subset of recorded templates was filtered out during post-processing because they were to short to carry meaningful information (Section 3.7). This occurred when the participant paused mid-gesture (e.g. moving arm up, pause, then moving arm down), splitting a single motion into short sub-motions or when they made accidental small movements that triggered the onset detector (e.g. changing sitting position).

Template recording was automated and launched with a single command:

```
roslaunch gesture record_gesture.launch \
record_gesture_name:=<circle/arm_up_down/ood>
```

5.1.2.2 Single agent collection

For the single-agent scenario, each participant stepped into the scene after background model initialization and interacted with the environment freely, simulating a realistic

office task (e.g. working at a computer). No fixed script was imposed, participants were encouraged to behave naturally, alternating between sitting and standing positions, in order to test the system under real-life conditions rather than controlled poses.

At their own discretion, participants performed authorized gestures (*circle* or *arm-up-down*) with a rest interval between them, as well as OOD gestures throughout the recording. At the end of the session the participant left the scene and the recording was stopped.

Data recording was automated as well through a single command line command:

```
./record_sys.sh
```

5.1.2.3 Multiple agent collection

Multi-agent recordings involved two or three participants simultaneously on scene. After the background model was acquired, participants were free to enter either together or sequentially. They were asked to interact naturally among themselves as they would in a shared office, again without a fixed script.

All participants on scene performed gestures of all three categories (*circle*, *arm-up-down*, and *ood*) throughout the recording. To emulate dynamic environments where the composition of the scene changes, participants were asked to leave and re-enter the field of view at intervals, producing scenarios where the authorized agent is temporarily absent. The following motion edge cases were deliberately simulated to stress test the IMU-vision fusion:

- One Agent moving and the other not moving.
- Both agent moving at the same time.
- Not authorized agent imitating authorized agent movements.

Scenarios with more than three agents were not collected. The $5.23\,\mathrm{m} \times 3.02\,\mathrm{m}$ scene coverage (Section 5.1.1) is further reduced by the presence of furniture, leaving limited usable floor area. Adding more agents would have resulted in excessive crowding and occlusion rather than realistic multi-person interaction.

Similarly, data recording was automated through a single command line command:

```
./record_sys_multi.sh
```

5.1.3 Data Annotation

For gesture template collection, annotation is implicit: the gesture class is selected as a launch parameter before recording begins, so each stored template carries an unambiguous class label. The participant is then asked to perform only the movement belonging to the collected class to avoid dataset contamination.

For system recordings, a ground-truth file (`ground_truth.json`) was created manually upon manual inspection of each file after recording (see Section 5.1.4). For every bag, the Kalman tracker ID(s) associated to the authorized agent was identified by inspecting the `vis_kalman` visualization window. The number of authorized gestures of each class performed by the authorized agent was counted from the `vis_depth_clean` visualization window. Only gestures performed by the authorized agent are annotated. OOD gestures and gestures by unauthorized agents were excluded. The following illustrates an example entry of the ground-truth file:

```
{
  "single_bag_001": {
    "authorized_agents": [1],
    "expected_gestures": {
      "1": {"circle": 2, "arm_up_down": 1}
    }
  },
  "multi_bag_001": {
    "authorized_agents": [1],
    "expected_gestures": {
      "1": {"circle": 1, "arm_up_down": 1}
    }
  }
}
```

Bag names are prefixed with `single_` or `multi_` so that `compute_accuracy.py` can partition the results automatically when computing aggregate statistics.

Data Privacy

Beyond the sensor recordings themselves (depth frames from TOFcam660 and raw IMU measurements) no personal data was stored. In particular, no information linking a recording to a specific participant is retained. Participant identity appear only in visualizations as anonymous integer IDs assigned by the Kalman tracker within a session and reset between sessions. This design prioritizes data minimization and removes any risk of inadvertent re-identification (Section 2.2.3).

5.1.4 Rosbag Playback Methodology

Evaluation is performed by replaying the recorded bags through the full pipeline rather than processing live sensor streams. All bags were replayed at real-time rate (no modifying the replay ratio), so that pipeline latency and temporal computations match the conditions experienced during recording and a reliable evaluation of live model performance can be obtained. The use of the `-clock` flag becomes relevant when bags are replayed at a different rate other than real-time.

5.2 Evaluation Metrics

5.2.1 Authorization Evaluation Metrics

Authorization is treated as a binary-classification over agent ID's. Consider (A) the set of agents in the scene, $\mathcal{P}$ the set of authorized agents during evaluation, and $\mathcal{G}$ the set of agents that should have been authorized according to the ground-truth. Then the evaluation metrics for a binary classification model can be obtained.

$$\begin{aligned} \mathrm{TP} &= |\mathcal{P} \cap \mathcal{G}|, & \mathrm{FP} &= |\mathcal{P} \setminus \mathcal{G}|, \\ \mathrm{FN} &= |\mathcal{G} \setminus \mathcal{P}|, & \mathrm{TN} &= |\mathcal{A} \setminus (\mathcal{P} \cup \mathcal{G})|, \end{aligned} \tag{5.3}$$

Precision, Recall, F1, and Accuracy are then derived from this results. Aggregate figures use micro-averaging: raw counts are summed across all bags before rates are computed.

Correct authorization time (t_{correct}) is the fraction of total authorized duration that is correctly attributed to ground-truth agents

$$t_{\text{correct}} = \frac{\sum_{i \in \mathcal{G}} t_{\text{auth},i}}{\sum_{i \in \mathcal{A}} t_{\text{auth},i}}, \tag{5.4}$$

where $t_{\text{auth},i}$ is the cumulative time agent i was in the authorized state. This metric is complementary to the binary classification metrics and allows to obtain better insights about the performance of the system, since authorization switches may happen but it is also important how long these switches last.

Scene authorization coverage (c_{scene}) measures what fraction of the scene duration had any authorized agent active.

$$c_{\text{scene}} = \frac{\sum_{i \in \mathcal{A}} t_{\text{auth},i}}{t_{\text{scene}}}, \tag{5.5}$$

where t_{scene} is the total scene duration.

Time to authorization (Δt_{auth}) is the time elapsed from the moment was first detected by the Kalman Tracker to the time it was first authorized. This metric is expected to be low for ground-truth agents and high for falsely authorized agents.

5.2.2 Gesture Recognition Evaluation Metrics

A gesture is considered a true positive when it is detected for the correct agent and belongs to a system gesture class. Gestures attributed to a wrongly authorized agent are considered false positives for that agent and false negatives to the ground truth agent.

For each authorized agent i, $\hat{c}_{i,k}$ and $c_{i,k}$ represent the number of detected and expected instances of the gesture class k. Then the metrics are computed on the following way:

$$\text{TP}_i = \sum_k \min(\hat{c}_{i,k},\, c_{i,k}), \quad \text{FP}_i = \sum_k \max(0,\, \hat{c}_{i,k} - c_{i,k}), \quad \text{FN}_i = \sum_k \max(0,\, c_{i,k} - \hat{c}_{i,k}). \tag{5.6}$$

Precision, Recall, and F1 metrics are computed based on this information. Micro-averaging is applied when aggregating across agents and bags.

Gesture latency (Δt_{gest}) is the time it takes the FSM transition to move from *ACTIVE* (gesture onset) to the publication of the recognition output on the `/gesture/recognized` topic.

5.3 Offline Evaluation of the Gesture Recognition System

Before evaluating the full pipeline, the DTW classifier was assessed in isolation using the classification procedure defined in Algorithm 13. The used technique was leave-one-template-out cross validation. For each template in the library, it was temporarily removed and tested against all the remaining templates in the library (otherwise the template would always match with itself).

A total of 493 templates were evaluated: 180 *circle*, 193 *arm_up_down*, and 120 OOD. The OOD templates are used exclusively to test rejection behavior. Table 5.1 summarizes the obtained results.

For the gesture class true positives show the correctly recognized instances while false negatives the missed gestures. For OOD, the true positives reflect the correctly rejected samples and the false positives the samples incorrectly accepted as a system gesture.

The aggregate results over the gesture classes are: Accuracy = 0.870, Precision = 0.997, Recall = 0.831, and F1 = 0.906. Figure 5.4 provide a graphic representation of per class and aggregate results. The almost perfect precision indicates that the selected DTW

Table 5.1: Leave-one-template-out offline gesture recognition results.

Class	Templates	TP	FP	FN	Recall
`circle`	180	145	0	35	0.806
`arm_up_down`	193	165	0	28	0.855
Aggregate (gesture)	**373**	**310**	**0**	**63**	**0.831**
OOD (rejection)	120	119	1	0	1.000

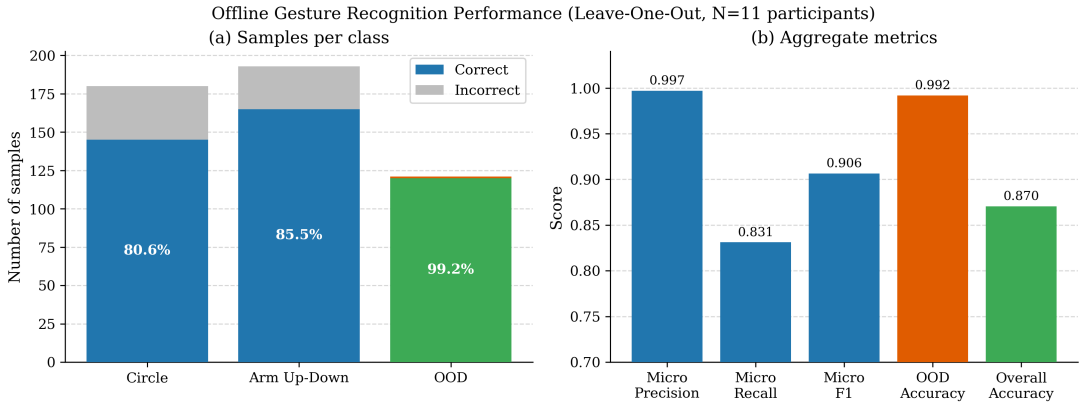


Figure 5.4: Obtained results for the offline evaluation of the Gesture Recognition System.

thresholds are conservative. When the classifier accepts a gesture it is almost always correct. The primary observed error is false negatives (rejected gestures because their normalized DTW exceeds the matching threshold) rather than misclassifications between classes. This asymmetric behavior is intentional, since from a security and access-control perspective it is preferable that the user has to repeat a gesture than to accept an incorrect one.

Rejection of OOD gestures is highly effective, with only one false positive observed out of 120 OOD queries. This confirms that the distance thresholds provide an effective out-of-distribution rejection (Recall = 1.000, F1 = 0.996).

Early stopping. As described in Section 3.8, an early-stopping mechanism was introduced to improve computing time of gesture classification when the DTW returns a confidence match. If the matching distance is below an strict threshold $\theta = 0.6$, the scan terminates immediately. This behavior was observed 78 times during the offline testing and only triggered when testing on system classes (not when testing OOD samples), which means no impact on system accuracy due to early stopping was observed. Additionally, competing-class distances are consistently above a 1.2 distance, which is well above the strict and the regular thresholds for the DTW matching algorithm.

5.4 Online System Evaluation

The online evaluation runs the complete identity-verified gesture recognition pipeline (preprocessing, tracking, IMU-vision fusion, gesture segmentation, and recognition) on all the pre recorded rosbags through file replaying. Metrics are collected automatically by an evaluation script `metrics_node` during each replay upon system shutdown and aggregated offline by a python script (`compute_accuracy.py`).

5.4.1 Metrics Infrastructure

The `metrics node` (package `metrics`) is a passive ROS node that subscribes to system topics and writes a structured report when it receives the ROS shutdown signal, without interfering with any other node.

Subscribed topics. Table 5.2 lists the topics consumed and the information extracted from each.

Table 5.2: Topics subscribed by `metrics_node`.

Topic	Information extracted
`/tracking/kalman`	Agent presence windows (first and last seen stamps); scene start time from bag clock.
`/fusion/authorized_agent_id`	Authorization transition events (agent ID, wall-clock stamp); cumulative authorized time per agent.
`/fusion/authorized_agent_confidence`	Confidence value at each authorization event and throughout authorized windows.
`/fusion/confidence`	Per-agent confidence over the entire scene.
`/gesture/state`	FSM transitions from IDLE to ACTIVE; records wall-clock onset time and the currently authorized agent.
`/gesture/recognized`	Recognized gesture name; used to compute latency from ACTIVE onset to recognition.

Clock handling. Scene duration and agent presence are measured in bag clock time, derived from the `header.stamp` of `/tracking/kalman` messages. This ensures that coverage statistics are independent of playback speed. Authorization duration and gesture latency are measured in wall-clock time, so they reflect actual pipeline latency including processing delays introduced during replay.

Report format. On shutdown, the metric nodes writes two files to `metrics/results/`. A human-readable `<bag_name>.txt` summary and a machine readable `<bag_name>.json` file. The per-bag result JSON files are consumed by the `compute_accuracy.py`, which compares the obtained results with the manually annotated ground truth (`ground_truth.json`). The script then produce per-bag and aggregate authorization and gesture metrics via micro-averaging.

5.4.2 Single-Agent Scenarios

The 28 single-agent bags contain recordings of only one participant each. Authorization and gesture metrics are reported in Table 5.3.

Table 5.3: Online evaluation results: single-agent scenarios (28 bags, 886.9 s total).

Metric	**Value**
Authorization	
Accuracy	0.930
Precision	0.935
Recall	0.967
F1	0.951
System behavior	
Scene auth coverage	67.7%
Correct authorization time	99.7%
Avg. time to authorization(GT agents)	3.03 s
Avg. time to authorization (FP agents)	0.93 s
Avg. scene confidence (GT agents)	0.788
Avg. scene confidence (non-GT agents)	0.439
Gesture recognition	
Precision	1.000
Recall	0.521
F1	0.685
Avg. gesture latency	2.85 s

Authorization F1 of 0.951 and correct authorization time of 99.7% indicate that the system reliably assigns authority to the correct person and that, when a false authorization does occur, it is very brief. The high recall of 0.967 means that in nearly all cases the expected authorized agent does receive an authorization event during the session. The small precision drop reflects two bags where a transient detection caused a non-ground-truth agent to be momentarily authorized (see Section 5.5 for a detailed analysis of this behavior). In average the system was fast to authorize, taking on average 3.03s to decide that the agent in scene should be authorized. Authorization was also stable, with minimal switching (0.04 $\text{events}_{\text{auth}}$/s). On average scene confidence for ground-truth-agents was 0.788 and for non-ground-truth-agents 0.439. In total only 1.6s a non-ground-truth agent was authorized.

Gesture precision of 1.000 confirms that every accepted gesture is attributed to an authorized agent and classified correctly. The observed average latency for gesture recognition was 2.850s, with a minimum of 0.302s and maximum of 4.621s. However, the 0.521 recall reveals that roughly half of the gestures are not detected (38 out of 73 expected), a consequence of the conservative DTW thresholds described in Section 3.8. This behavior is discussed in detail in Section 5.5.

5.4.3 Multi-Agent Scenarios

The 29 multi-agent bags contain two or three participants simultaneously. Results are shown in Table 5.4.

Table 5.4: Online evaluation results: multi-agent scenarios (29 bags, 1521.1 s total).

Metric	**Value**
Authorization	
Accuracy	0.733
Precision	0.600
Recall	0.911
F1	0.723
System behavior	
Scene auth coverage	71.9%
Correct authorization time	82.6%
Avg. time to authorization (GT agents)	3.26 s
Avg. time to authorization (FP agents)	4.67 s
Avg. scene confidence (GT agents)	0.757
Avg. scene confidence (non-GT agents)	0.523
Gesture recognition	
Precision	1.000
Recall	0.487
F1	0.655
Gestures attributed to correct agent	92.1%
Avg. gesture latency	3.08 s

Authorization F1 of 0.723 and correct authorization time of 82.6% indicate that, while the ground-truth agent is identified in most recordings, 17.4% of the cumulative authorized time is attributed to non-ground-truth agents. The high recall of 0.911 shows that the legitimate agent receives at least one authorization event in the vast majority of sessions. Additionally, the precision of 0.600 shows transient authorization switching between agents on the scene. The average time to first authorization is 3.26 s, for ground-truth agents and 4.67 s for non-ground-truth agents. The authorization switching rate is 0.12 $\text{events}_{\text{auth}}$/s. Finally, the average confidence of ground-truth agents was 0.757 and 0.523 for non authorized agents. A detailed analysis of these results is provided in Section 5.5.

Gesture precision of 1.000 shows that every detected gesture is classified correctly and attributed to an authorized agent. Of the 119 expected gestures, 58 were correctly detected and attributed to the ground-truth agent (Recall = 0.487). An additional 5 gestures were detected but attributed to a falsely authorized agent, bringing the total number of detected gestures to 63, of which 92.1% were correctly attributed. The average gesture latency was 3.077 s, with a minimum of 1.068 s and a maximum of 4.865 s.

Detailed analysis of the multiagent performance and comparison with single agent performance can be found in Section 5.5.

5.4.4 Combined Results

Table 5.5 aggregates all 57 bags (28 single-agent and 29 multi-agent) into a single set of system-level metrics.

Table 5.5: Online evaluation results: all scenarios combined (57 bags, 2408.0 s total).

Metric	**Value**
Authorization	
Accuracy	0.778
Precision	0.690
Recall	0.930
F1	0.792
System behavior	
Scene auth coverage	70.4%
Correct authorization time	88.7%
Avg. time to authorization (GT agents)	3.24 s
Avg. time to authorization (FP agents)	4.44 s
Avg. scene confidence (GT agents)	0.768
Avg. scene confidence (non-GT agents)	0.512
Gesture recognition	
Precision	1.000
Recall	0.500
F1	0.667
Gestures attributed to correct agent	95.0%
Avg. gesture latency	2.97 s

Across all 57 sessions, the system achieves an authorization F1 of 0.792 and a correct authorization time of 88.7%, meaning that 11.3% of the total authorized duration was incorrectly attributed to non-ground-truth agents. The high recall of 0.930 confirms that the legitimate agent receives at least one authorization event in the vast majority of recordings. Figure 5.5 shows the obtained metrics for single- and multi-agent scenarios and the aggregated results.

Ground-truth agents are authorized faster (3.24 s) than FP agents (4.44 s), consistent with the expectation that genuine IMU-vision correlation accumulates confidence more quickly.

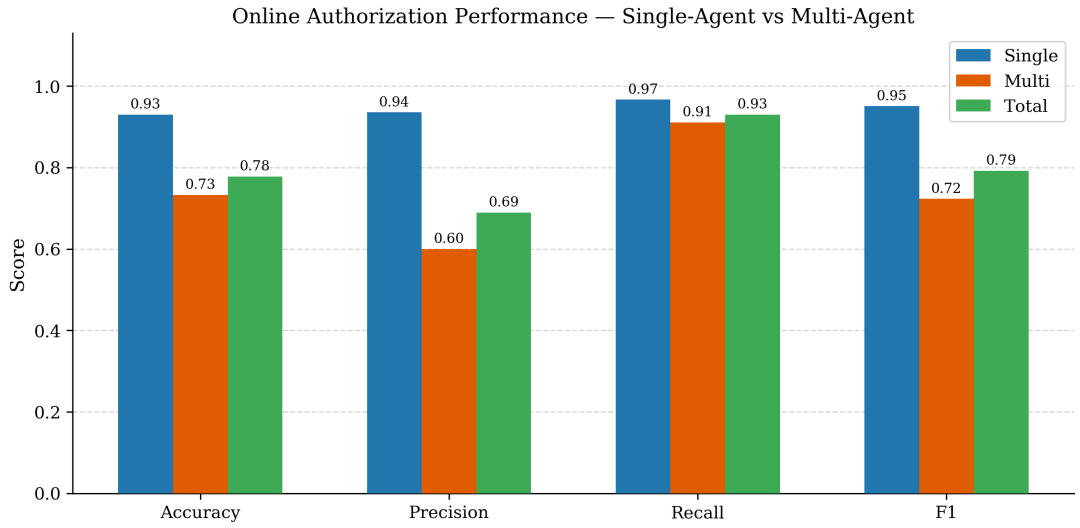


Figure 5.5: Accuracy, Precision, Recall, and F1 scores for online system evaluation.

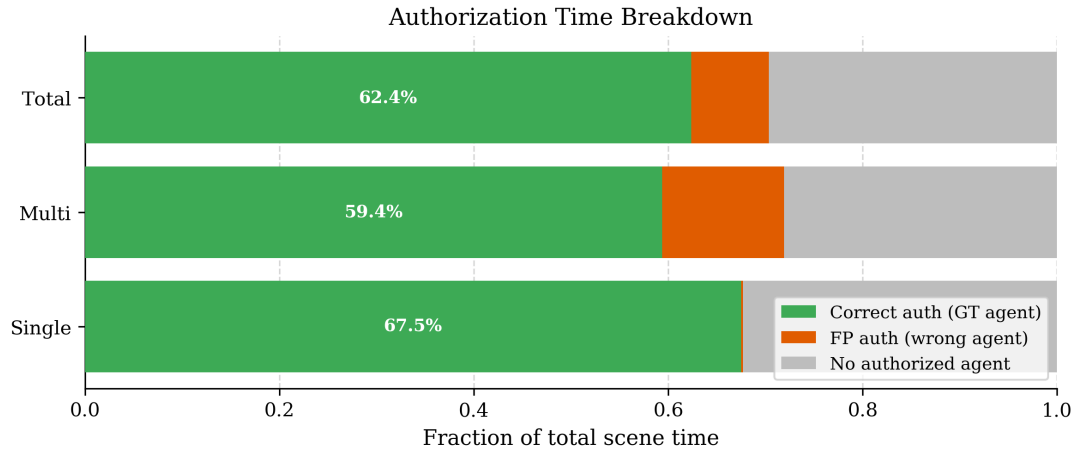


Figure 5.6: Correct authorization time, incorrect authorization time, and overall scene time.

The mean scene confidence is 0.768 for GT agents and 0.512 for non-GT agents, reflecting a clear separation between the two groups. Figures 5.6 and 5.7 show graphically the time ground-truth agents get authorized throughout scene time and the confidence values ground-truth agents and non-ground-truth agents get over time.

Of the 192 expected gestures, 101 were detected, all with gesture precision of 1.000. Of those, 96 (95.0%) were correctly attributed to the ground-truth agent and 5 to a falsely authorized agent, yielding a recall of 0.500. The average gesture recognition latency was 2.97 s. A detailed discussion of these aggregate trends is provided in Section 5.5.

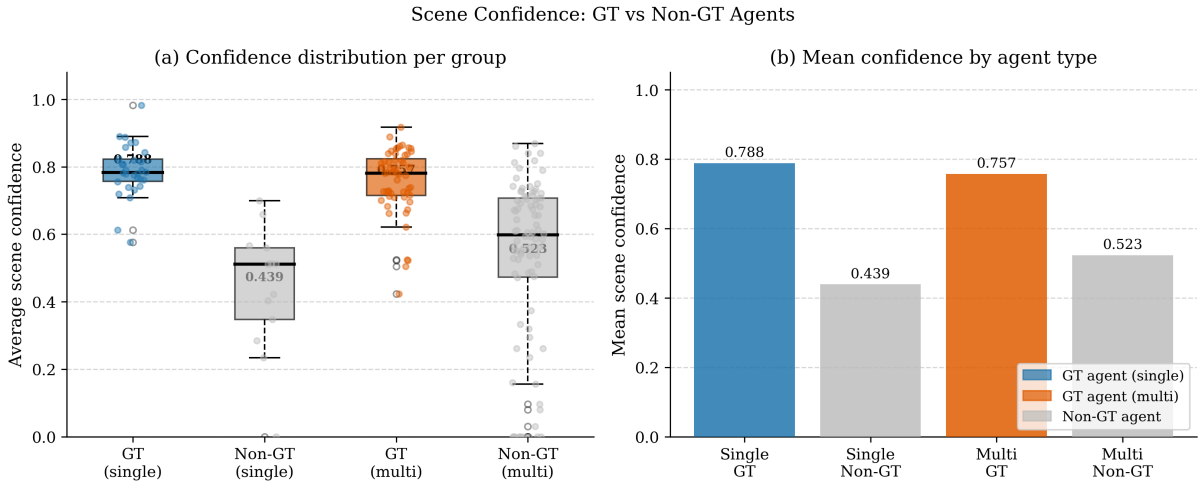


Figure 5.7: Average confidence obtained for agents on the scene.

5.5 Discussion

5.5.1 Offline Gesture Recognition

Based on the offline classifier results (Section 5.3), it is possible to conclude that DTW in isolation achieves very good performance. The relatively high number of false negatives (see Table 5.1) shows that it is a very sensitive algorithm that heavily penalizes small deviations between gesture trajectories to classify and stored templates. This explains why recall (0.831) was the lowest scoring metric. The fact that a test query was not matched against with templates recorded by the same participant further demonstrates this sensitivity concern. On the other hand, the OOD rejection rate is very high, which demonstrates the robustness of the classifier against falsely recognizing out-of-distribution gestures. It is important to emphasize that although missing gestures (large false negatives) is undesirable, this behavior is preferable from a security and access-control point of view over classifying out-of-distribution gestures as in-distribution.

It is worth noting that although the offline gesture testing approach compared each sample against all other templates, including those recorded by the same participant, in some cases test queries achieved minimum distance when compared to templates from other participants. This suggests that the system may be robust enough to recognize gestures even if the authorized agent had not previously contributed any sample to the template library. This hypothesis however requires further testing and results may depend on the quality and diversity of templates in the library.

It is important to mention that results are favorable considering a small template library was used (373 templates). Performance could potentially be improved by adding more templates to the library. However, this comes with the tradeoff of increasing worst-case computing time, since DTW complexity is defined as:

$$\mathcal{O}(|\mathcal{T}| \cdot N^2) \tag{5.7}$$

where $|\mathcal{T}|$ is the size of the template library and N is the length of the longest of the two trajectories being compared. Early stopping helps reduce computing time in optimistic scenarios. However, as described in Section 5.3, the mechanism is not constantly triggered due to the DTW's inherent sensitivity.

Overall, isolation results provide good justification for continuing to use DTW as the system's gesture recognition method and results confirm the efficacy of the selected thresholds. While relaxing thresholds could reduce false negatives, it would come at the cost of accepting more OOD gestures as valid, which is undesirable from an access-control perspective.

5.5.2 System Performance

5.5.2.1 Preprocessing Performance

Across all 57 recordings, the preprocessing and tracking pipeline demonstrated robust performance: no bag exhibited a complete agent detection failure, and all expected agents were tracked throughout their time in scene. The observed authorization switches and true negative counts confirm that non-ground-truth agents were also reliably detected, ruling out tracking failures as a source of false negatives in the authorization results.

The Kalman filter (Section 3.5) was central to this robustness. Time-of-Flight sensors are inherently noisy and prone to depth flickering and transient misdetections. The Kalman filter allows the system to continuously estimate each agent's position and velocity from the motion model alone during brief measurement outages, maintaining track continuity through superblob events and unstable depth frames.

Additionally, the preprocessing demonstrated reliable upper-body segmentation, avoiding false detections from moving objects such as chairs. The superblob splitting routine successfully separated overlapping agent blobs, which was particularly relevant during multi-agent testing when participants walked close to each other.

5.5.2.2 Parameter configuration

As described in Section 4.3.1, four preprocessing parameters were tuned across bags to achieve satisfactory foreground segmentation: `fg_depth_thresh`, `median_kernel`, `open_kernel_size`, and `close_kernel_size` (see Table E.1). The full per-bag configurations used to obtain the reported results are documented in `configs_sys_rosbags.txt`. Despite covering diverse participants and lighting conditions, the parameter space explored was narrow: 14 out of 57 bags ran with fully default settings, and the adjusted values were conservative departures from the defaults rather than large reconfigurations. Specifically, `fg_depth_thresh` was modified in 43 bags (range 1300-1750), `close_kernel_size` in 23 bags (range 19-25), and `median_kernel`/`open_kernel_size` were raised together from 3 to 5 in 7 multi-agent bags only.

`fg_depth_thresh`. This parameter sets the maximum depth value accepted as foreground, effectively clipping the segmentation to the upper-body region of interest. A lower value tightens the depth cut and works better for taller participants (whose upper body is closer to the camera), while a higher value is necessary for shorter participants ($\approx 1.6\,\mathrm{m}$) or sitting postures, where the torso and arms are farther from the sensor. This was the most frequently adjusted parameter, required in 43 out of 57 bags.

`close_kernel_size`. The morphological opening operation used to remove noise also erodes thin foreground structures such as outstretched arms. A larger closing kernel compensates for this erosion and consolidates the agent blob, including the arm region. This is critical for capturing foreground energy during gestures (Section 3.4). The need to increase this value from its default was observed mainly in single-agent bags, likely because arm segmentation quality is more critical when no other agents are present to help disambiguate foreground energy.

`median_kernel` **and** `open_kernel_size`. These two parameters were always adjusted together, as both control the aggressiveness of smoothing and noise suppression. Their default value of 3 was sufficient in single-agent conditions, but in 7 multi-agent bags a value of 5 produced cleaner segmentation. The likely cause is IR multi-path interference: with multiple people in the scene, ToF IR signals reflect off several surfaces simultaneously, increasing depth measurement noise and requiring stronger smoothing to obtain stable foreground masks.

Additional factors influencing parameter sensitivity across sessions include varying lighting conditions and ToF sensor thermal drift during long recordings (see Section 4.3.1). Current parameter tuning was sufficient for this proof-of-concept evaluation. For a production deployment, an online calibration algorithm would be necessary, capable of automatically adjusting parameters based on live measurements. Alternatively, active per-session TOFcam660 recalibration could reduce the sensitivity to these variations, as a fixed factory calibration was used throughout all evaluations.

5.5.3 Single-Agent Performance

Single agent performance showed that the idea of correlating IMU measurements with depth information is a feasible strategy for determining an authorized agent. In the single-agent case there is no correlation ambiguity between IMU measurements and different agents' motion, so the system showed high authorization performance, authorizing the correct agent 99.7% of the time (see Table 5.3 and Figure 5.5). Authorization did not reach a perfect score due to two specific cases (*single_028.bag* and *single_029.bag*), where the authorized agent left the scene and the person responsible for data collection stepped in briefly to stop the recording and was detected by the Kalman tracker. The hypothesis is that during this time the authorized agent, while outside of the scene, was performing some movements with the IMU-wearing arm, which rapidly correlated with the new agent on scene. The resulting rapid confidence increase of the new agent led to an authorization

event, since the former authorized agent was not on scene and therefore its confidence was decaying but had not yet been marked as lost by the cleanup routine, so the strict reentry threshold was not yet active (Section 4.4.2). The True Negative count of 11 shows that in other recordings the data collector was detected as an agent but never authorized. Additionally, the fact that the non-ground-truth agent was authorized for only 1.6 s in total, and that its average scene confidence was of 0.439 (a clear gap relative to the GT average of 0.788, see Figure 5.7), further supports this implementation.

Gesture recognition achieved a precision of 1.000, meaning that all detected gestures were correctly classified and attributed to the authorized agent (there were no competing agents in this scenario). The low recall of 0.521 means that only roughly half of the expected gestures were detected (38 out of 73). This can be due to the following reasons:

- **Tracking loss.** The system briefly misses the tracked agent. This can be due to sensor flickering or to agent long static periods, resulting on the agent being partially absorbed to the background reducing then the segmentation quality.
- **Premature gesture.** If the participant performs a gesture before the fusion confidence crosses the authorization threshold, the gesture segmentation FSM never enters ACTIVE for an authorized agent and the gesture is not classified.
- **Merged segments.** Back-to-back gestures performed without sufficient rest between them can be merged into a single segment that exceeds `max_gesture_len` and is discarded, or that produces a trajectory too dissimilar from any template to pass the DTW threshold (OOD rejection).
- **Threshold values.** It was observed that in general the best DTW distances achieved during online evaluation tend to be higher than during offline evaluation. It was decided not to increase the threshold from a security and access-control perspective, where it is preferable that the user retries a gesture than to accept an OOD movement.
- **Gesture quality.** When participants record gestures in isolation, they tend to perform them carefully and consistently. When interacting with the environment (and other agents), attention is divided, which may lead to lower quality executions. DTW is a sensitive algorithm that penalizes even slight trajectory deviations. Combined with the conservative thresholds, results in more gestures being rejected.
- **System bottleneck.** The gesture node may be unable to process a new gesture segment if the DTW classifier is still busy evaluating a previous one (even if gestures are triggered with in theory enough rest time for allowing FSM transitions) causing it to miss the incoming segment. This bottleneck is most likely due to DTW complexity (see Section 5.3).

5.5.4 Multi-Agent Performance

It is possible to see that the multi-agent metrics are lower in comparison with the single agent case, achieving an authorization accuracy of 0.733 and a precision of 0.600 (see

Table 5.4 and Figure 5.5). This reduced metrics indicate that multiple times non-ground-truth agents where authorized. Specifically, a low Precision is impacted by a high number of FP (34). A higher FP count is due to a more unstable system (in terms of authorization) since multi-agent evaluation present short transient switches between the authorized agent and competing candidates. A more revealing metric is that 82.6% of the time the ground-truth agents are in fact authorized. Meaning that although the system present short authority switches, most of the time the right authorization decision is made.

Additionally, is important to consider that the average time to authorize ground truth agents is 3.26 s and the the time to authorize non-ground-truth agents is 4.67 s. The fact that the GT time to authorize is faster means that the system does tend to correlate easier the IMU measurements to the vision data corresponding to the expected GT agent. Finally, the average scene confidence for GT agents is 0.757 and for non-GT agents is 0.523 which confirms a trend to decide for the right agent (see Figure 5.7).

The hypothesis on why there is transient authorization of non-ground-truth agents is that IMU motion, agent motion, and foreground motion are only energy measurements expressing how much activity there is in the respective domain. The logic is that when the IMU is active there should be motion in the foreground when gesturing, and when the IMU is active there should be motion of the agent's centroid when walking. Conversely, when the IMU is static, both agent and foreground motion should be low (see Section 3.6). This works well in the single agent case since there are no competing agents, and works to a certain degree in the multiagent case, especially when agents are moving differently (e.g. agent-1 walking while agent-2 is static). When another agent imitates the authorized agent's movement patterns (even when they are not exactly the same) the energy measurements may correlate equally with the non-ground-truth agent, because they are not discriminative in movement direction, only in magnitude.

Additionally, it is important to note that although this reasoning holds, measurement units are comparable but not directly equivalent. IMU provides linear acceleration in m/s^2 and angular velocity in rad/s, while vision provides agent centroid speed in m/s and foreground energy as a dimensionless normalized activity score (see Section 3.6). A natural first instinct would be to integrate acceleration to obtain IMU velocity and compare it directly to agent velocity, but it is not possible to obtain an accurate velocity measurement from the MPU 6050 due to integration drift (Section 3.1). While angular velocity spikes while gesturing, as does foreground energy, the two signals are not expressed on the same scales (Section 3.6). To overcome this, all measurements are normalized using adaptive RMS ceilings to obtain instantaneous energy scores over a time window in $[0, 1]$. Similarity is quantified using a Gaussian kernel over a dual-channel (see Section 3.6). To further improve robustness, measurements are evaluated through both short-term temporal cross-correlation and long-term activity correlation, with energy scores additionally modulated by per-agent activity state classification (see Section 3.6).

A more discriminative approach would be to correlate IMU measurements per axis (rather than scalar magnitudes) with directional variation of the depth map, effectively extracting a 3D motion trajectory (x-, y-, and z-axis). However, this would require reliably isolating each agent's arm (specifically the wrist region), so that their directional movement patterns could be extracted and matched against the three-axis IMU signal. Given that

this project is an initial proof of concept for an identity-aware gesture recognition system, which as the best of our knowledge is an under-researched topic, this approach was not pursued due to the substantially increased system complexity and the time constraints of this thesis. This opens the debate if it is worth the effort pursuing this alternative (per-axis correlation) or sacrificing the IMU advantage in gesture recognition and pick a different complementary sensing modality that can be fused with ToF measurements to determine authority and delegate gesture recognition entirely to the depth sensor, accepting the limited visual cues available from the overhead perspective.

It is also worth highlighting a hardware limitation that contributes to the observed imperfections. The MPU-6050 is a low cost IMU and it is inherently noisy, prone to temperature-dependent drift, and lacks a shared reference frame with the depth camera. Similarily, the TOFcam660 produces relatively noisy depth measurements compared to higher-end sensors, leading to foreground segmentation instabilities. Despite these hardware constraints, the overall results are encouraging, demonstrating that even with inexpensive and imprecise sensors, the fusion strategy produces meaningful authorization decisions for most of the scene duration (Table 5.4 and Figure 5.6).

On a final note on the system activity correlation, the solution showed to correctly determine the expected authorized agent for most of the time. The natural limitations described above explain why performance is not perfect and this discussion constitutes a first step towards future improvements of this identity-aware gesture recognition system.

On the gesture recognition side, the system presents the same limitations as in the single agent setup, missing almost half of the expected gestures (Recall 0.487). However, on the same way as in the single agent case, all the gestures attributed to authorized agents are correctly assigned to the expected system class (Precision 1.000) and OOD gestures were always rejected, confirming the robustness against false positives of DTW. Causes on why gestures are missed in the multi-agent case are the same as in the single agent scenario (Section 5.5.3).

It is important to note that in the multi-agent scenario 5 gestures were attributed to a falsely authorized agent. This means that on 5 occasions an unauthorized agent triggered a gesture event. The primary cause was movement imitation: an unauthorized agent intentionally replicated the GT agent's motion, including performing gestures at the same time. The previously discussed limitations of magnitude-only energy measurements resulted that under these circumstances the system occasionally switched authority to the unauthorized agent, causing the gesture to be attributed to the non-GT participant. This raises the question of whether this constitutes a genuine vulnerability in this specific setup, since the authorized agent also performed the gesture at the same time, the smart-room command itself would have been correctly intended. The main practical implication is that the activity log would recorded the action as triggered by an unexpected user, which is undesirable from an audit perspective.

5.5.5 Overall System Performance

Overall, it is possible to say that the gesture recognition system is very robust against false positives and achieves high precision on the classification of detected gestures that belong to a recognized system class. Additionally, the authorization module correctly identifies the right agent for most of the scene duration. Figure 5.6 is particularly revealing, showing that across single and multiagent scenarios, the overwhelming majority of authorized time is assigned to the ground-truth agent, with the FP authorization slice being visibly small in single-agent conditions and slightly larger the multi-agent case. The large no-authorized time is expected, since there are times when the scene is empty. Additionally, the system requires time to build up sufficient confidence before declaring an authorized agent, which increases as well the no-authorized time.

Figure 5.7 shows the average scene confidence for GT and non-GT agents across scenarios, explicitly demonstrating that the system consistently assigns higher confidence to ground-truth agents with a clear gap relative to non-ground-truth agents. The reason why average GT confidence is not higher is that the agents start with zero confidence each time they enter or re-enter the scene and must gradually accumulate evidence. This phenomena also happens when an agent is lost due to a long static period, resulting in a partial background absorption. This requires as well to rebuild confidence from scratch. Non-GT agents are subject to the same conditions with the difference that in most of the time, they tend to not build up as much confidence as GT-agents, which is the expected behavior.

Finally, Table 5.5 shows encouraging results for this proof of concept, achieving correct authorization 88.7% of the time. The recall of 0.930 means the expected agent receives at least one authorization event in the vast majority of recordings. The slightly lower accuracy of 0.778 is mostly attributable to transient authorization switching in the multi-agent case (Section 5.5.4).

Overall, the system results are promising for a proof-of-concept given the hardware and algorithmic constraints described throughout this discussion. It is hoped that this work will encourage further research on identity-verified gesture recognition systems, and that future efforts on this topic will yield improved performance.

5.5.6 Improving Gesture Recognition Performance - Creation of a dataset for gesture recognition on an overhead perspective

The main reason DTW was selected for gesture recognition was the absence of an existing overhead-perspective gesture dataset, combined with the decision to use a wrist-worn IMU as the identity-correlation modality. To the best of our knowledge, no publicly available dataset covers gesture recognition from an overhead depth sensor configuration. DTW enabled a robust recognition method without requiring large amounts of training data, correctly classifying in-distribution gestures and reliably rejecting OOD movements.

As with any design decision, DTW has its own tradeoffs. First, as discussed in Section 5.5.3, it is a sensitive algorithm that heavily penalizes time-series mismatches, resulting in a higher number of false negatives (see Section 5.3). Second, complexity scales

linearly with template library size and quadratically with trajectory length (Equation 5.7), contributing to an average gesture recognition latency of 2.97 s during online testing (Table 5.5). Improving recognition quality by increasing the template library would increase this latency further, making the system bottleneck worse and likely causing more gestures to be missed.

An additional contribution of this project is a small, annotated dataset of synchronized depth measurements (see Table 4.2) and IMU recordings of participants performing gestures in isolation, captured from an overhead configuration. Using classical ML approaches such as SVMs, or more recent neural network architectures, this dataset (combined with data augmentation techniques) could be used to train a gesture-recognition model (bimodal, IMU-only, or vision-only) that would be substantially faster than DTW at inference time.

The tradeoff of ML-based approaches is that training cost scales with model complexity. The zero-shot capability of DTW would be lost (DTW can begin recognizing a new gesture class by only adding one sample to the template library), while ML models require retraining whenever there's a change in the authorized gesture classes. Additionally, ML-approaches are not inherently robust at rejecting OOD sampling as DTW is.

5.5.7 Privacy Analysis

The use of a ToF depth camera rather than an RGB camera provides inherent privacy advantages. The sensor produces a low-resolution (320×240) depth map in which no color, texture, or facial features are captured. Identities are represented only by blob silhouette and approximate height, making inadvertent facial recognition or biometric profiling impossible from the stored data. This aligns with the GDPR principle of data minimization [69]. The evaluation recordings were made with informed participant consent, and no information linking a recording to a specific individual is stored beyond the anonymous tracker ID assigned by the tracking algorithm (Section 4.3.2), which is reset at the start of each session.

5.5.8 Limitations

- **Background absorption of stationary users.** The EMA background model (Section 3.4) slowly incorporates stationary foreground blobs. The `bg_beta2` parameter controls this rate; a value that is too high causes loss of stationary participants, while a value that is too low prevents the model from adapting to genuine changes in the scene.
- **Tuning parameters across diverse conditions.** As discussed in Section 5.5.2.2, four preprocessing parameters required per-session tuning to achieve satisfactory foreground segmentation. The main drivers were varying illumination conditions, differences in participant height, and ToF sensor thermal drift during long recording sessions. For a production deployment, an online calibration mechanism would be required.

- **IMU-vision confusion under simultaneous motion.** The energy correlation used for fusion is based on scalar magnitudes rather than directional measurements, making it susceptible to false authorizations when multiple agents move synchronously or intentionally imitate each other's movement patterns, as observed in the multi-agent results.

- **Sensor hardware quality.** The MPU-6050 is a low-cost consumer-grade MEMS sensor prone to noise and temperature-dependent bias drift. Notably, integrating its acceleration measurements to estimate velocity is not viable due to rapid error accumulation, which is why visual centroid speed is used for the translational channel instead. The TOFcam660 similarly produces noisy depth measurements, particularly when multiple people are present due to IR multi-path interference, contributing to foreground segmentation instabilities. Both sensors lack a shared physical reference frame, requiring normalization-based comparison rather than direct unit-equivalent correlation.

- **Agent height and scene boundary sensitivity.** Participants shorter than average required a larger `fg_depth_thresh` to correctly capture their upper body and arm region. Additionally, agents near the edges of the camera's field of view are more susceptible to partial occlusion and blob fragmentation, which can temporarily degrade tracking quality.

- **Gesture recognition latency and bottleneck.** DTW complexity scales linearly with the template library size and quadratically with trajectory length (Equation 5.7), resulting in an average gesture recognition latency of 2.97 s. This creates a processing bottleneck: if a gesture is triggered while the classifier is still evaluating a previous segment, the new segment may be missed. Enlarging the template library to improve recognition quality would worsen this bottleneck further.

- **Out-of-scene IMU activity.** Because the IMU transmits over WiFi UDP regardless of the wearer's physical location, the fusion module receives and process inertial data even when the authorized agent has left the camera's field of view. This is a more subtle vulnerability than movement imitation. From the system's perspective the scene appears normal (an agent is visible and moving) and the IMU is active, so there is no observable signal that the two belong to different physical locations. If the out-of-scene authorized agent happens to move similar in terms of energy to an in-scene bystander, the fusion module may incorrectly accumulate confidence for that bystander and eventually authorize it. The single-agent anomaly cases (*single_028* and *single_029*) are examples of this exact failure mode. Per-axis directional correlation would reduce the likelihood of such coincidental matches but cannot fully eliminate the vulnerability, since directional agreement by chance remains possible.

5.5.9 Comparison to Related Work

The authorization concept of granting gesture control rights to a specific individual identified through wrist-worn inertial sensing rather than appearance is, to the best of our

knowledge, novel within the ToF-based gesture recognition literature. The closest analogue is Wang et al [70], who combine gesture recognition with user authentication using a LeapMotion depth sensor and a one-class autoencoder, achieving a FAR of 0.69% and FRR of 1.05% after incremental learning. However, their approach requires prior enrollment and continuous model retraining, whereas this system performs passive authorization without storing any biometric profile.

Table 5.6 places the offline gesture recognition accuracy of 87.0% (F1 = 0.906, precision = 0.997) in context with ToF-based gesture recognition systems from the literature.

Authors	**Method**	**Camera pos.**	**Accuracy**	**Dataset size**
Molina et al. [44]	SVM	Frontal	90.0%	∼600 samples/class
Uebersax et al. [66]	Thresholding	Frontal	88.3–89.6%	≥50 samples/letter, 7 users
Takahashi et al. [63]	SVM + BoF	Frontal	95.0%	∼2 700 frames/gesture/user
Gomaa et al. [53]	CNN-LSTM	Frontal	90.4%†	7 288 samples, 83 users
Ma et al. [39]	ResNet-50	Frontal	98.5%	600 samples, 4 users
This work	**DTW**	**Overhead**	**87.0%**	**373 templates, 11 users**

Table 5.6: Comparison of ToF-based gesture recognition systems. †After personalized incremental learning; baseline accuracy without personalization is 66.9%.

Two contextual factors are essential for interpreting this comparison. First, every prior work in Table 5.6 uses a *frontal* camera positioning, which provides rich cues on hand pose and motion, and finger articulation. This thesis is, to the best of our knowledge, the first to use an *overhead* ToF configuration in a gesture recognition system, where these cues are absent and classification relies entirely on the wrist-worn IMU trajectory. The 87.0% accuracy is therefore competitive given this geometric constraint.

Second, the high-accuracy deep learning entries [39, 53, 63] require between hundreds and thousands of annotated samples per class and per user. Our DTW classifier achieves comparable performance to non-ML baselines (Molina 90.0% [44] and Uebersax 88.3% [66]) using only 373 templates across all classes and users combined, with no training phase. This minimal-data property is a deliberate design choice aligned with the proof-of-concept scope of this work, and positions the system as a practical baseline for future overhead gesture recognition research.

Regarding identity verification, none of the gesture recognition works cited above include any mechanism to restrict commands to an authorized user. The body of work closest to this aspect is person re-identification from depth data [1, 6, 18, 31, 32, 36, 37, 42, 72], where agents are identified by matching stored geometric or appearance descriptors (body shape, head geometry, anthropometric features) against a pre-enrolled template database. Three of these works operate from an overhead configuration [31, 36, 38], which is the most comparable setting to this thesis. However, all of them share a fundamental requirement absent from this work: prior registration of each authorized user and persistent storage of personal biometric data. This conflicts with the privacy-preserving design intent and precludes deployment in scenarios where the authorized user set is dynamic or biometric

enrollment is not acceptable. This thesis fulfills these requirements by correlating a wrist-worn IMU signal with tracked visual motion, which is inherently passive. The system then identifies the authorized agent through continuous sensor agreement without ever extracting or storing a biometric descriptor.

Chapter 6

Final Considerations

6.1 Summary

This thesis investigated whether privacy-preserving, identity-aware gesture recognition can be achieved in a multi-user smart-room environment using only a ceiling-mounted Time-of-Flight depth camera and a wrist-worn Inertial Measurement Unit. The motivation was twofold: the growing adoption of gesture interfaces in shared smart environments and the privacy concerns associated with RGB camera-based motioning. The central challenge was not only to recognize gestures, but to do so selectively, accepting commands only from a specific authorized user while rejecting those from other people in the scene.

To address this, a modular real-time processing pipeline was designed and implemented within the ROS framework. The pipeline comprises five main stages operating in sequence. First, a background subtraction and morphological operations are used to segment the upper-body foreground of people on scene in each depth frame. Second, a Kalman filter assigns persistent identities to detected blobs and maintains tracks through occlusions and sensor noise. Third, an IMU-vision fusion module correlates wrist-worn inertial measurements with the tracked agents' visual motion signatures to accumulate a per-agent confidence score and determine which agent is authorized. Fourth, a finite-state evaluates the authorized agent IMU measurements and segments candidate gesture intervals. Fifth, a DTW classifier with a Sakoe-Chiba band compares each segmented trajectory against a template library to recognize gestures or reject out-of-distribution movements.

The system was evaluated over 57 pre-recorded ROS bag sessions (28 single-agent, 29 multi-agent) totaling 2408 s of interaction. An offline evaluation of the gesture classifier in isolation was also conducted on a leave-one-template-out basis across 11 participants. Together, these evaluations provide both a component-level and an end-to-end characterization of the system's performance.

6.2 Conclusions

The three specific objectives stated in Chapter 1.2, developing a conceptual framework, implementing it as a real-time pipeline, and evaluating it quantitatively and qualitatively, were achieved.

Regarding the conceptual framework, the thesis first established that passive identification from ToF depth data alone is not achievable without prior user enrollment. There is no existing approach that demonstrates registration-free person identification from depth imagery, and any depth-based re-identification would require storing biometric profiles. This motivated the investigation of a complementary sensing modality. The thesis demonstrated that passive identity verification through IMU-vision energy correlation is a viable strategy for authorizing gesture commands in a smart-room context. The key insight is that a wrist-worn IMU and a depth camera observe the same underlying physical activity from different modalities. By normalizing each signal to a comparable energy scale and measuring their agreement through instantaneous Gaussian scoring, short-term temporal cross-correlation, and long-term Pearson correlation, the system builds a confidence metric that discriminates the authorized agent from bystanders without requiring any appearance-based identification or previous system registration.

Regarding implementation, the pipeline was locally implemented, using a modular ROS system with no dependency on external servers or RGB imagery. All processing runs on a standard desktop PC, and the system operates in real time at the camera's frame rate. The modular architecture allowed independent development and testing, and the use of rosbag recordings enabled reproducible evaluation without requiring participants to be present at each test run.

Regarding evaluation, the results are encouraging for a proof of concept. In single agent scenarios the system authorized the correct agent 99.7% of the time, with an authorization F1 of 0.951 and gesture precision of 1.000. In multi-agent scenarios, where the disambiguation problem is inherently harder, the system achieved an authorization F1 of 0.723 and correct authorization time of 82.6%, with gesture precision again of 1.000. Across all 57 sessions combined the system authorized the correct agent 88.7% of the time and achieved a gesture recall of 0.500, with all detected gestures correctly classified. These numbers confirm that the core hypothesis of IMU-vision correlation being sufficient to identify the authorized user in the majority of conditions is valid.

Several important lessons were learned during the execution of this work. First, energy-based scalar correlation is effective in single-agent conditions but inherently limited in multi-agent settings. The system cannot distinguish between two agents who move with similar overall energy, since direction information is discarded by the magnitude normalization. The multi-agent performance gap relative to the single-agent case can be largely attributed to this limitation. Second, the DTW classifier proved a strong choice for the available data, achieving 87.0% offline accuracy with only 373 templates (and no training required). Its sensitivity, however, carries a cost in recall, even slight deviations in gesture trajectory or execution speed can cause rejection. Third, the TOFcam660 and MPU-6050, being the last one inexpensive and consumer-grade, are sensors that individually exhibit noise, drift, and a lack of shared spatial reference frame. The fact that meaningful identity

discrimination is achievable with such hardware is arguably the strongest indicator of the concept's robustness. Finally, the overhead camera configuration, while advantageous for coverage, privacy, and occlusion handling, is genuinely novel in the gesture recognition literature, and the absence of a reference dataset was the primary driver behind choosing DTW over a machine learning approach.

On the privacy dimension, the use of a ToF depth camera was confirmed to align well with the principle of data minimization. The system operates entirely on low-resolution depth maps from which facial recognition, texture identification, and clothing based profiling are structurally impossible. The anonymous Kalman tracker IDs, which are reset at each session, ensure that no persistent biometric record associated to a participant is created or stored.

6.3 Future Work

The results of this thesis and the limitations identified in Chapter 5 open several concrete directions for future research.

Directional IMU-vision correlation. The most impactful improvement to the identity verification module would be replacing scalar magnitude correlation with per-axis directional comparison. Correlating the three-axis IMU acceleration and angular velocity with the directional variation of the depth map within the agent's bounding box would make the fusion substantially more discriminative, as two agents moving with similar energy but in different directions would no longer produce ambiguous confidence scores (the primary failure mode identified in Section 5.5.4). This requires reliably isolating the wrist and arm region of each tracked agent, which is a non-trivial segmentation problem at the camera's resolution. This improvement would also partially mitigate the out-of-scene IMU activity vulnerability described in Section 5.5.8. However, per-axis correlation cannot fully eliminate the vulnerability since directional agreement by chance remains possible. A robust solution would additionally require a proximity signal, such as BLE from the ESP32, to determine the physical presence of the authorized agent in the room.

Machine Learning based gesture recognition. A small annotated dataset of synchronized depth and IMU recordings from 11 participants performing gestures from an overhead configuration was collected as part of this thesis. This dataset, augmented with standard data augmentation techniques, provides the foundation for training supervised classifiers (SVMs, CNNs, or small transformer-based models) that would be substantially faster at inference time than DTW and potentially more robust to execution variability. Achieving this would eliminate the gesture latency bottleneck and improve recall.

Automated preprocessing calibration. Four preprocessing parameters required manual per-session tuning in the current evaluation. An online calibration module that monitors segmentation quality metrics, such as blob size consistency, foreground stability, arm detection metrics, and heights of the people on scene, could then adjusts parameters adaptively, removing this way the parameter tuning dependency and making the system deployable without expert intervention.

Larger gesture vocabulary. The current system recognizes two gesture classes (*arm_up_down* and *circle*) and an out-of-distribution rejection class. Extending the vocabulary to five or more gestures would increase the system's utility in smart-room applications, and the collected dataset provides a starting point for developing and evaluating additional classes.

Alternative identity verification approaches. The results suggest that IMU-vision correlation works well but is fundamentally limited by the scalar nature of the comparison. Alternative approaches worth exploring include using the depth-image silhouette dynamics for identification based on agents' movement patterns (similar to Wang et al [70] but for whole body dynamics and using an overhead configuration), or combining the current IMU approach with a height and body-proportion estimate derived from the preprocessing pipeline. These two approaches would require registration of users and storing biometric enrollment data, introducing re-identification risks absent from the current passive approach (Section 2.2.3). Additionally, different sensor alternatives to IMU that provide a different and probably better performing modality should be explored, e.g. UWB for precise localization, or a second wrist-worn camera.

Extended multi-agent evaluation. The current multi-agent dataset covers scenarios with two or three participants. Evaluating the system with more agents, increasing diversity in movement patterns, and deliberate adversarial imitation in a controlled setting would provide a more complete characterization of the system-robustness limits and guide the prioritization improvements listed above.

Bibliography

[1] A. Albiol et al. “Who is who at different cameras: people re-identification using depth cameras”. In: *IET Computer Vision* 6 (5 2012), pp. 378–387.

[2] Alessandro Bevilacqua, Luigi Di Stefano, and Pietro Azzari. “People Tracking Using a Time-of-Flight Depth Sensor”. In: *2006 IEEE International Conference on Video and Signal Based Surveillance*. 2006, pp. 89–89.

[3] Luca Bianchi et al. “Tracking without Background Model for Time-of-Flight Cameras”. In: *Advances in Image and Video Technology*. Ed. by Toshikazu Wada, Fay Huang, and Stephen Lin. 2009, pp. 726–737.

[4] Gary Bradski. “OpenCV: Open Source Computer Vision Library”. In: *Dr. Dobb's Journal of Software Tools*. 2000.

[5] Pia Breuer, Christian Eckes, and Stefan Müller. “Hand Gesture Recognition with a Novel IR Time-of-Flight Range Camera–A Pilot Study”. In: *Computer Vision/Computer Graphics Collaboration Techniques*. Ed. by André Gagalowicz and Wilfried Philips. 2007, pp. 247–260.

[6] Zijun Cheng et al. “3D face recognition based on kinect depth data”. In: *2017 4th International Conference on Systems and Informatics (ICSAI)*. 2017, pp. 555–559.

[7] Edward Chou et al. “Privacy-Preserving Action Recognition for Smart Hospitals using Low-Resolution Depth Images”. In: *CoRR* abs/1811.09950 (2018).

[8] *ESPROS-660FPGA ROS Package*. `https://wiki.ros.org/ESPROS-660FPGA`. 2026.

[9] ESPROS Photonics Corporation. *Installation and Operation Manual for TOFcam-660*. ESPROS Photonics Corporation. 2021.

[10] Qun Fang, YiHui Yan, and GuoQing Ma. *Gesture Recognition in Millimeter-Wave Radar Based on Spatio-Temporal Feature Sequences*. 2023. URL: `https://arxiv.org/abs/2309.09528`.

[11] Alvaro Fernandez-Rincon et al. “Robust People Detection and Tracking from an Overhead Time-of-Flight Camera”. In: Jan. 2017, pp. 556–564.

[12] Martin A. Fischler and Robert C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Commun. ACM* 24.6 (June 1981), 381â395.

[13] Sergi Foix, Guillem Alenya, and Carme Torras. “Lock-in Time-of-Flight (ToF) Cameras: A Survey”. In: *IEEE Sensors Journal* 11.9 (2011), pp. 1917–1926.

[14] S.B. Gokturk and Carlo Tomasi. “3D head tracking based on recognition and interpolation using a time-of-flight depth sensor”. In: vol. 2. Jan. 2004, pp. II–211.

[15] S.B. Gokturk, H. Yalcin, and C. Bamji. “A Time-Of-Flight Depth Sensor - System Description, Issues and Solutions”. In: *2004 Conference on Computer Vision and Pattern Recognition Workshop*. 2004, pp. 35–35.

[16] Rafael Gonzalez, Richard Woods, and Barry Masters. “Digital Image Processing, Third Edition”. In: *Journal of Biomedical Optics* 14 (Mar. 2009), pp. 029901–029901.

[17] Sigurjon Arni Guomundsson et al. “TOF imaging in Smart room environments towards improved people tracking”. In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 2008, pp. 1–6.

[18] Frank M. Hafner et al. “RGB-Depth Cross-Modal Person Re-identification”. In: *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2019, pp. 1–8.

[19] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, et al. “Array programming with NumPy”. In: *Nature* 585.7825 (2020), pp. 357–362.

[20] Christopher G. Harris and M. J. Stephens. “A Combined Corner and Edge Detector”. In: *Alvey Vision Conference*. 1988.

[21] *imu_filter_madgwick ROS Package (imu_tools)*. `https://index.ros.org/r/imu_tools/`. 2026.

[22] InvenSense (TDK). *MPU-6000/MPU-6050 Datasheet*. Tech. rep. PS-MPU-6000A-00. TDK InvenSense, 2015.

[23] Ankit Jha et al. *GesSure- A Robust Face-Authentication enabled Dynamic Gesture Recognition GUI Application*. 2022. URL: `https://arxiv.org/abs/2207.11033`.

[24] Li Jia and Richard J. Radke. “Using Time-of-Flight Measurements for Privacy-Preserving Tracking in a Smart Room”. In: *IEEE Transactions on Industrial Informatics* 10.1 (2014), pp. 689–696.

[25] Rudolph Emil Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[26] Tomasz Kapuściński, Mariusz Oszust, and Marian Wysocki. “Hand Gesture Recognition Using Time-of-Flight Camera and Viewpoint Feature Histogram”. In: *Intelligent Systems in Technical and Medical Diagnostics*. Ed. by Jozef Korbicz and Marek Kowal. 2014, pp. 403–414.

[27] Marco Karrer, Patrik Schmuck, and Margarita Chli. “CVI-SLAMâCollaborative Visual-Inertial SLAM”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 2762–2769.

[28] Hansol Kim, JongEun Park, and Minsung Kang. “Multi-frame demosaicing for the Quad Bayer CFA in the color difference domain”. In: *Optics Express* 32 (Apr. 2024), pp. 14223–14239.

[29] Eva Kollorz et al. “Gesture recognition with a Time-Of-Flight camera”. In: *IJISTA* 5 (Jan. 2008), pp. 334–343.

[30] Harold W. Kuhn. “The Hungarian method for the assignment problem”. In: *Naval Research Logistics (NRL)* 52 (1955).

[31] Daniele Liciotti et al. “Person Re-identification Dataset with RGB-D Camera in a Top-View Configuration”. In: *Video Analytics. Face and Facial Expression Recognition and Audience Measurement*. Ed. by Kamal Nasrollahi et al. 2017, pp. 1–11.

[32] Han Liu et al. “Matching Depth to RGB for Boosting Face Verification”. In: *Biometric Recognition*. Ed. by Jie Zhou et al. 2017, pp. 127–134.

[33] D.G. Lowe. “Object recognition from local scale-invariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2.

[34] Bruce Lucas and Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)”. In: vol. 81. Apr. 1981.

[35] Teghan Lucas and Maciej Henneberg. “Are human faces unique? A metric approach to finding single individuals without duplicates in large samples”. In: *Forensic Science International* 257 (2015), 514.e1–514.e6.

[36] Sara Luengo SÃ¡nchez et al. “Re-identificaciÃ³n*dePersonasUtilizandonicamenteInformaci*³*n*
In: *teleÃ¡tica* (2019).

[37] Carlos A. Luna et al. “Fast heuristic method to detect people in frontal depth images”. In: *Expert Systems with Applications* 168 (2021), p. 114483.

[38] Carlos A. Luna et al. “People re-identification using depth and intensity information from an overhead camera”. In: *Expert Systems with Applications* 182 (2021), p. 115287.

[39] Yuqian Ma et al. “Gesture Recognition Based on Time-of-Flight Sensor and Residual Neural Network”. In: *Journal of Computer and Communications. Vol.12* (2024).

[40] Sebastian O. H. Madgwick, Andrew J. L. Harrison, and Ravi Vaidyanathan. “Estimation of IMU and MARG orientation using a gradient descent algorithm”. In: *2011 IEEE International Conference on Rehabilitation Robotics*. 2011, pp. 1–7.

[41] Prasanta Chandra Mahalanobis. “On the Generalized Distance in Statistics”. In: *Proceedings of the National Institute of Sciences of India* 2.1 (1936), pp. 49–55.

[42] Simon Meers and Koren Ward. “Face Recognition Using a Time-of-Flight Camera”. In: *2009 Sixth International Conference on Computer Graphics, Imaging and Visualization*. 2009, pp. 377–382.

[43] Sushmita Mitra and Tinku Acharya. “Gesture Recognition: A Survey”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.3 (2007), pp. 311–324.

[44] Javier Molina et al. “Real-time user independent hand gesture recognition from time-of-flight camera video using static and dynamic models”. In: *Machine Vision and Applications* (2013).

[45] Masafumi Nakagawa and Tamaki Kobayashi. “Real-Time Floor Recognition in Indoor Environments Using ToF Camera”. In: *Assian Association of Remote Sensing* (2016).

[46] D. Nister, O. Naroditsky, and J. Bergen. “Visual odometry”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. 2004, pp. I–I.

[47] Open Robotics. *ROS Noetic Ninjemys*. `https://wiki.ros.org/noetic`. 2020.

[48] Open Source Robotics Foundation. *rospy: Python Client Library for ROS*. `https://wiki.ros.org/rospy`. 2026.

[49] Serban Oprisescu, Constantin Burlacu, and Vasile Buzuloiu. “Action recognition using time of flight cameras”. In: *2010 8th International Conference on Communications*. 2010, pp. 153–156.

[50] Serban Oprisescu, Christoph Rasche, and Bochao Su. “Automatic static hand gesture recognition using ToF cameras”. In: *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*. 2012, pp. 2748–2751.

[51] Jorge Ortiz. *Identity-Verified Gesture Recognition System*. https://github.com/JorgeOrtizV/identity-verified-gesture-rec. 2026.

[52] Random Nerd Tutorials. *MPU-6050 âHow It Worksâ diagram (ESP32)*. https://randomnerdtutorials.com/wp-content/uploads/2021/01/MPU-6050-How-It-Works-ESP32.png. 2021.

[53] Guillermo Reyes, Amr Gomaa, and Michael Feld. "Itâs all about you: Personalized in-Vehicle Gesture Recognition with a Time-of-Flight Camera". In: *Proceedings of the 15th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. AutomotiveUI â23. Sept. 2023, 234â243.

[54] Ethan Rublee et al. "ORB: an efficient alternative to SIFT or SURF". In: Nov. 2011, pp. 2564–2571.

[55] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (1978), pp. 43–49.

[56] Davide Scaramuzza. *Image Formation*. 2022. URL: https://rpg.ifi.uzh.ch/docs/teaching/2022/02_image_formation.pdf.

[57] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, 815â823.

[58] Yaakov bar shalom, X.âRong Li, and Thia Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. Jan. 2004.

[59] Roland Siegwart, Ilia Noourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. 2011.

[60] Joan SolÃ. *Quaternion kinematics for the error-state Kalman filter*. 2017. URL: https://arxiv.org/abs/1711.02508.

[61] Ravi Srisha and Am Khan. "Morphological Operations for Image Processing : Understanding and its Applications". In: Dec. 2013.

[62] Carsten Stahlschmidt et al. "Applications for a people detection and tracking algorithm using a time-of-flight camera". In: *Multimedia Tools and Applications* (2016).

[63] Masaki Takahashi et al. "Human gesture recognition system for TV viewing using time-of-flight camera". In: *Multimedia Tools and Applications* (2013).

[64] Levente Tamas and Andrei Cozma. "Embedded real-time people detection and tracking with time-of-flight camera". In: Apr. 2021, p. 10.

[65] Matthew Turk and Alex Pentland. "Eigenfaces for Recognition". In: *Journal of Cognitive Neuroscience* 3.1 (Jan. 1991), pp. 71–86.

[66] Dominique Uebersax et al. "Real-time sign language letter and word recognition from depth data". In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 2011, pp. 383–390.

[67] Raghav H. Venkatnarayan and Muhammad Shahzad. "Gesture Recognition Using Ambient Light". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2.1 (Mar. 2018).

[68] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272.

[69] Paul Voight and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR). A Practical Guide*. 2024.

[70] Xuan Wang and Jiro Tanaka. "GesID: 3D Gesture Authentication Based on Depth Camera and One-Class Classification". In: *Sensors* 18.10 (2018).

[71] Oliver J. Woodman. "An introduction to inertial navigation". In: 2007.

[72] Ancong Wu, Wei-Shi Zheng, and Jian-Huang Lai. "Depth-based person re-identification". In: *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*. 2015, pp. 026–030.

[73] Li Yu et al. "Gesture Recognition Using Reflected Visible and Infrared Lightwave Signals". In: *IEEE Transactions on Human-Machine Systems* 51.1 (2021), pp. 44–55.

[74] Morteza Zahedi and Ali Reza Manashty. "Robust Sign Language Recognition System Using ToF Depth Cameras". In: *CoRR* abs/1105.0699 (2011).

[75] Peijun Zhao et al. "mID: Tracking and Identifying People with Millimeter Wave Radar". In: *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 2019, pp. 33–40.

Abbreviations

CNN	Convolutional Neural Network
ToF	Time-of-Flight
RGB	Red Green Blue, often referred to color cameras
TPR	True Positive Rate
FPR	False Positive Rate
TNR	True Negative Rate
FNR	False Negative Rate
GDPR	General Data Protection Regulation
DCSCN	Deep CNN skip Connection and Network in Network
HCI	Human Computer Interface
OCC	One Class Classification
SVM	Support Vector Machine
IaaS	Infrastructure as a Service
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPFS	Inter Planetary File System
ISP	Internet Service Provider
NFV	Network Function Virtualization
P2P	Peer to Peer
PoA	Proof-of-Authority
PoW	Proof-of-Work
REST	Representational State Transfer
RTT	Round Trip Time
SDN	Software-Defined Networking
SLA	Service Level Agreement
VNF	Virtualized Network Function

List of Figures

2.1 a) Bayer Filter; b) Quad Bayer Filter [28] 5

2.2 Thin Lens Equation [56] 6

2.3 Illustration of the Time-of-Flight (ToF) principle and phase-based distance calculation. 7

2.4 Authentication Procedure based on decomposition of Gesture movement in three planes [70]. 13

2.5 Depth-RGB re-identification [18] 15

3.1 Overview of system architecture design 21

3.2 ESPROS TOFcam660 22

3.3 Overhead camera positioning in an office environment 23

3.4 Camera point of view - Overhead configuration for this identity-aware gesture recognition system 23

3.5 Example of data acquirable using ESPROS TOFcam660 [9] 24

3.6 Diagram flow of the raw image preprocessing process to extract a foreground mask and region of interests (blobs) 26

3.7 Diagram flow of the tracking process for robust evaluation of agents' position and velocity based on visual input. 29

3.8 Functional architecture of the Identity Verification and Authorization stage. The pipeline fuses inertial data (compensated via a Madgwick filter) with visual tracked agent data. It computes instantaneous energy scores (E), short-term temporal correlations (ρ), and long-term activity bonuses (r) across translational and rotational channels to maintain a per-agent confidence score (c_i). Final authorization is determined by the Authority Manager. 34

3.9 Two-state Finite State Machine (2-FSM) for gesture segmentation. IDLE→ACTIVE transition requires N_{on} consecutive samples above θ_{on}. ACTIVE→IDLE transition requires N_{off} consecutive samples below θ_{off}. A captured segment is forwarded for classification only if its length falls within $[L_{\min}, L_{\max}]$. Segments outside this range are discarded. The authorization snapshot is taken at the IDLE→ACTIVE transition and is held fixed for the duration of the gesture. 41

4.1 Expected system inputs. In this specific example obtained from rosbag replay. 46

4.2 ESP32, IMU, and server interaction [52] 47

4.3 Background model and foreground segmentation results. Left column shows the cleaned depth image and right column shows the output of the segmentation step. 51

4.4 Blob extraction using connected components. Left column shows the cleaned depth image and right column shows the detected blobs after connected-component labeling and size filtering. 52

4.5 Superblob splitting during a handshake using depth local-maxima extraction and Watershed. Left: foreground mask showing a continuous superblob. Right: individual sub-blobs recovered for each agent. 53

4.6 Kalman tracker output. Left column shows the input detected blobs and right column shows the Kalman filter estimates. 58

4.7 Kalman tracker estimating the changing position of a moving agent across time. 58

4.8 Track persistence through missed detections and spontaneous disappearances of agents. 59

4.9 Superblob handling in the Kalman tracker. Left column shows the preprocessing blob output and right column shows the tracker response. 60

4.10 Confidence build-up process as the agent interacts with the environment. . 63

4.11 Single-agent interaction with the system. 64

4.12 Multi-agent interaction with the system. 64

4.13 Authorization handling in a three-agent scenario. 65

4.14 Authorization handling under similar simultaneous movement patterns. . . 65

4.15 Gesture segmentation FSM states and recognition output in the online system. 69

5.1 Sequence of the *circle* gesture class from the camera perspective. Images were obtained from the foreground segmentation visualization. 73

5.2 Sequence of the *arm_up_down* gesture class from the camera perspective. Images were obtained from the foreground segmentation visualization. . . . 73

5.3 Illustration of the *arm_up_down* gesture class movement pattern from a front perspective. 74

5.4 Obtained results for the offline evaluation of the Gesture Recognition System. 79

5.5 Accuracy, Precision, Recall, and F1 scores for online system evaluation. . . 84

5.6 Correct authorization time, incorrect authorization time, and overall scene time. 84

5.7 Average confidence obtained for agents on the scene. 85

A.1 System Ros Graph 118

List of Tables

2.1 Summary of ToF sensing applications for indoors monitoring 16

2.2 Summary of ToF sensing applications for gesture recognition. 17

2.3 Summary of ToF sensing applications for person identification. 18

4.1 Main ROS topics exchanged between system nodes. 46

4.2 ROS topics and functions for the cam660_node [9]. 47

4.3 Visualization nodes and their properties. 70

5.1 Leave-one-template-out offline gesture recognition results. 79

5.2 Topics subscribed by `metrics_node`. 80

5.3 Online evaluation results: single-agent scenarios (28 bags, 886.9 s total). . . 81

5.4 Online evaluation results: multi-agent scenarios (29 bags, 1521.1 s total). . 82

5.5 Online evaluation results: all scenarios combined (57 bags, 2408.0 s total). . 83

5.6 Comparison of ToF-based gesture recognition systems. †After personalized incremental learning; baseline accuracy without personalization is 66.9%. . 94

B.1 Parameters for the CAM660 sensor. 119

D.1 Configurable parameters of the `tof_preprocessing` tracking node. 124

E.1 Key configurable parameters of the `tof_preprocessing` node. 126

H.1 Configurable parameters of the `mpu_reader` node. 131

I.1 Configurable parameters of the `fusion` node. 133

J.1 Configurable parameters of the `gesture` node. 136

List of Algorithms

1 tof_preprocessing — main per-frame callback 50
2 ForegroundSegmentation 50
3 UpdateTracks 54
4 UpdateBackground 54
5 Tracking node — main per-frame callback 56
6 BuildCostMatrix 56
7 CreateAgent 57
8 mpu_reader — main receive loop 60
9 fusion — IMU callback 61
10 fusion — agent callback 62
11 gesture — IMU callback 66
12 UpdateSegmenter 67
13 Classify 68

Appendix A

System ROS Graph

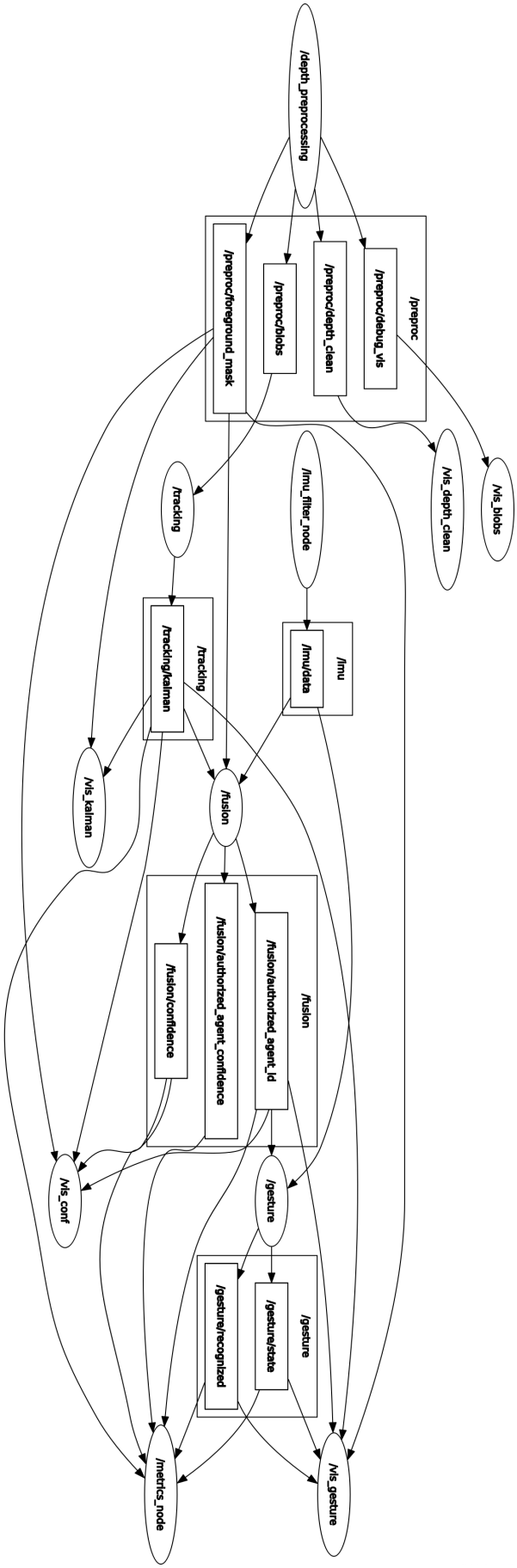


Figure A.1: System Ros Graph

Appendix B

ESPROS TOFcam660 calibration parameters

Parameter	Value	Description
image_type	1	Selected image output mode
integration_time_tof_1	60	Primary ToF integration time (μs)
integration_time_tof_2	480	Secondary ToF integration time (μs)
integration_time_tof_3	1000	Tertiary ToF integration time (μs)
integration_time_gray	3000	Ambient gray image integration time
min_amplitude	70	Minimum signal amplitude threshold
average_filter	false	Spatial averaging filter state
median_filter	false	Median noise reduction filter state
temporal_filter_factor	0.0	Exponential moving average factor
temporal_filter_threshold	0	Threshold for temporal filtering
edge_filter_threshold	0	Depth discontinuity filter threshold
roi_left_x	0	Region of Interest left boundary
roi_right_x	319	Region of Interest right boundary
roi_height	240	Region of Interest total height
lens_type	0	Lens model/calibration index
frequency_modulation	1	ToF modulation frequency channel
channel	0	Sensor communication channel
cartesian	true	Output in Cartesian coordinates
publish_point_cloud	true	Enable PointCloud2 publishing

Table B.1: Parameters for the CAM660 sensor.

Appendix C

Repository Directory Structure

The catkin workspace is rooted at `cam660_apps/`. Only source files relevant to the system are listed; build artefacts (`build/`, `devel/`) and rosbag recordings are omitted. For more details visit the project repository [51].

```
cam660_apps/
|-- src/
|   |-- tof_preprocessing/
|   |   |-- scripts/
|   |   |   |-- depth_subscriber.py
|   |   |   |-- preprocessing.py
|   |   |   |-- tracking.py
|   |   |   |-- vis_depth_clean.py
|   |   |   |-- vis_fg_mask.py
|   |   |   |-- vis_blobs.py
|   |   |   `-- vis_kalman.py
|   |   |-- launch/
|   |   |   |-- system.launch
|   |   |   |-- system_rosbag.launch
|   |   |   |-- preprocessing.launch
|   |   |   |-- tracking.launch
|   |   |   |-- show_vis.launch
|   |   |   |-- record_sys.launch
|   |   |   `-- record_sys_multi.launch
|   |   |-- msg/
|   |   |   |-- Blob.msg
|   |   |   |-- BlobArray.msg
|   |   |   |-- AgentMsg.msg
|   |   |   `-- AgentArray.msg
|   |   `-- package.xml
|   |
|   |-- mpu/
|   |   |-- scripts/
```

```
|   |   |   |-- mpu_reader.py
|   |   |   |-- fusion.py
|   |   |   `-- vis_conf.py
|   |   |-- launch/
|   |   |   |-- mpu_reader.launch
|   |   |   `-- fusion.launch
|   |   |-- msg/
|   |   |   |-- Conf.msg
|   |   |   `-- ConfArray.msg
|   |   `-- package.xml
|   |
|   |-- gesture/
|   |   |-- scripts/
|   |   |   |-- gesture_node.py
|   |   |   |-- vis_gesture.py
|   |   |   |-- clean_templates.py
|   |   |   `-- test.py
|   |   |-- launch/
|   |   |   |-- gesture.launch
|   |   |   |-- gesture_recognition.launch
|   |   |   `-- record_gesture.launch
|   |   |-- templates/
|   |   |   |-- arm_up_down/    [193 .npy template files]
|   |   |   `-- circle/         [180 .npy template files]
|   |   |-- clean_bags.sh
|   |   `-- package.xml
|   |
|   `-- metrics/
|       |-- scripts/
|       |   |-- metrics_node.py
|       |   |-- compute_accuracy.py
|       |   `-- plots.ipynb
|       |-- launch/
|       |   `-- metrics.launch
|       |-- ground_truth/
|       |   `-- ground_truth.json
|       |-- results/            [57 .json + .txt per-bag result files]
|       `-- package.xml
|
|-- system_launch.sh
|-- system_launch_rosbag.sh
|-- record_sys.sh
|-- record_sys_multi.sh
`-- system_commands.txt
```

Appendix D

Tracking Node Parameters

Parameter	Default	Description
`maha_thresh`	9.21	Mahalanobis gate; corresponds to 99% confidence ellipse under χ^2_2
`hit_thresh`	10	Consecutive detections required to promote CANDIDATE → ACTIVE
`miss_count_thresh`	5	Consecutive misses before transitioning to LOST
`bbox_alpha` (α)	0.7	EMA weight for bounding box smoothing
`boundary_thresh`	40 px	Pixel margin defining the BOUNDARY spawn zone
`proximity_thresh`	80 px	Centroid distance below which a CENTER-spawn blob is suppressed
`static_scene_threshold`	20 px/s	Mean agent velocity below which the scene is considered static
`avg_conf_thresh`	0.7	Confidence above which superblob agents keep their prediction
`agent_cleanup_conf`	0.3	Confidence below which an agent is unconditionally deleted
`agent_legal_age`	10 frames	Frames below which a CENTER-spawn agent is subject to strict cleanup
`iou_thresh`	0.6	IoU threshold for conservative duplicate merge
`duplicate_centroid_thresh`	30 px	Centroid distance threshold for IoU-based merge
`strict_close_centroids_thresh`	20 px	Centroid distance for unconditional merge
`size_ratio_thresh`	0.35	Bounding box area ratio below which smaller agent is treated as fragment
`loose_close_centroid_thresh`	60 px	Centroid distance for size-ratio-based fragment merge

Table D.1: Configurable parameters of the `tof_preprocessing` tracking node.

Appendix E

Preprocessing Node Parameters

Parameter	Single	Multi	Description
`bg_frames`	50		Frames used to initialise background model
`fg_threshold` (τ)	200 mm		Depth-change threshold for foreground classification
`fg_depth_thresh`	1300-1700 mm		Maximum foreground depth; removes floor-level objects
`min_depth`	150 mm		Minimum valid depth; discards near-range noise
`median_kernel`	3	5	Kernel size for median denoising
`open_kernel_size`	3	5	Kernel size for morphological opening
`close_kernel_size`	17-23 px		Kernel size for per-component closing
`min_blob_size`	300 px		Minimum area retained in the foreground mask
`min_blob_area`	3000 px		Minimum area to register as a tracked agent
`track_dist_thresh`	50 px		Maximum centroid distance for blob–track matching
`track_timeout`	15 frames		Frames before dropping an unmatched track
`fg_age_thresh` (θ_{age})	50 frames		Age before a persistent foreground pixel re-enters background update
`bg_beta` (β)	0.995		EMA weight for normal background update
`bg_beta2` (β_2)	0.9		EMA weight for fast update when scene is static
`static_scene_thresh`	30 frames		Consecutive static frames before triggering fast update

Table E.1: Key configurable parameters of the `tof_preprocessing` node.

Appendix F

tof_preprocessing custom message definitions

F.1 Blob

```
int32 id
float32 cx
float32 cy
int32 x
int32 y
int32 w
int32 h
float32 area
bool is_static
```

F.2 BlobArray

```
std_msgs/Header header
Blob[] blobs
```

F.3 AgentMsg

```
int32 id
float32 cx
float32 cy
float32 vx
float32 vy
int32 x
```

```
int32 y
int32 w
int32 h
int32 age
string state
```

F.4 AgentArray

```
std_msgs/Header header
AgentMsg[] agents
```

Appendix G

mpu custom message definitions

G.1 Conf

```
int32 id
float32 conf
string state
```

G.2 ConfArray

```
std_msgs/Header header
Conf[] confs
```

Appendix H

IMU Reader parameters

Parameter	Default	Description
`esp32_ip`	*empty*	Expected source IP of the ESP32; packets from any other address are silently dropped
`udp_port`	5005	UDP port to listen on

Table H.1: Configurable parameters of the `mpu_reader` node.

Appendix I

Fusion Node Parameters

Table I.1: Configurable parameters of the `fusion` node.

Parameter	Default	Description
`camera_height`	1.9 m	Camera mounting height above ground
`fov_horizontal`	108°	Horizontal field of view
`image_width`	320 px	Image width for px/m calibration
`imu_window_duration` (T_w)	0.4 s	IMU integration window
`imu_safety_margin`	0.3 s	Extra buffer margin beyond T_w
`imu_queue_size`	200	IMU buffer capacity
`max_accel` ($a_{\max}$)	9.0 m/s^2	Normalization ceiling for acceleration
`max_gyro` ($\omega_{\max}$)	3.5 rad/s	Normalization ceiling for angular velocity
`imu_thresh` (ϵ)	0.05	Minimum balanced IMU energy to trigger scoring
`vision_window_duration`	1.0 s	Agent speed history window
`min_agent_age`	5 frames	Minimum track age before evaluation
`default_max_agent_vel`	5.0 m/s	Initial speed normalization ceiling
`agent_vel_alpha`	0.98	EMA weight for adaptive speed ceiling
`default_max_fg_change`	100 px	Initial FG change normalization ceiling
`fg_update_alpha`	0.99	EMA weight for adaptive FG change ceiling
`fg_change_history_len`	10	Frames for FG change RMS history
`gaussian_scale` (λ)	1.0	Gaussian kernel scale for energy score
`energy_accel_weight` (w_a)	0.5	Weight for translational channel
`temporal_weight` (w_t)	0.6	Weight of temporal vs. energy score
`temporal_lag_tolerance`	0.1 s	Max cross-correlation lag search
`activity_history_len` (L)	30	Long-term activity history length

Continued on next page

Table I.1 – Continued from previous page

Parameter	Default	Description
`corr_bonus_scale` (γ)	0.25	Long-term correlation bonus scale
`correlation_min_history`	10	Min history before computing bonus
`max_fusion_score` ($s_{\max}$)	2.0	Per-frame score cap
`ema_alpha` (α)	0.9	EMA factor for confidence update
`confidence_thresh` (θ_{auth})	0.8	Threshold to authorize
`tolerant_confidence_thresh` (θ_{tol})	0.6	Threshold to retain current agent
`reentry_confidence_thresh` (θ_{re})	1.2	Threshold after scene exit
`reentry_window` (T_{re})	5.0 s	Window duration after exit
`no_scores_decay` (δ)	0.9995	Confidence decay per callback when no scores
`hysteresis_delay` (T_{hyst})	5.0 s	Min authorization duration before switching
`switch_cooldown` (T_{cool})	3.0 s	Cooldown after each switch
`switch_margin` (m)	1.3	Required score improvement to trigger a switch
`exploit_high_conf`	1.3	Confidence above which eval. probability is low
`exploit_high_prob`	0.2	Evaluation probability at high confidence
`exploit_med_conf`	0.8	Confidence for medium evaluation probability
`exploit_med_prob`	0.5	Evaluation probability at medium confidence

Appendix J

Gesture Node Parameters

Parameter	Default	Description
`imu_rate`	50 Hz	Expected IMU rate; auto-updated from inter-arrival times
`buffer_duration`	3.0 s	Duration of the rolling lookback buffer
`onset_thresh` (θ_{onset})	0.30	Energy threshold to increment onset counter
`offset_thresh` (θ_{offset})	0.15	Energy threshold to increment offset counter
`onset_count` (N_{onset})	3	Consecutive samples to confirm IDLE→ACTIVE
`offset_count` (N_{offset})	8	Consecutive samples to confirm ACTIVE→IDLE
`lookback_samples`	5	Samples prepended from rolling buffer at onset
`min_gesture_len` ($L_{\min}$)	15	Minimum samples ($\approx$0.3 s) for a valid gesture
`max_gesture_len` ($L_{\max}$)	200	Maximum samples ($\approx$4.0 s) before discarding
`energy_accel_weight` (w_a)	0.5	Translational weight in segmentation energy
`max_accel` ($a_{\max}$)	9.0 m/s^2	Normalization ceiling for acceleration
`max_gyro` ($\omega_{\max}$)	3.5 rad/s	Normalization ceiling for angular velocity
`dtw_threshold` (θ_{DTW})	0.82	Global per-class recognition threshold
`strict_threshold` (θ_{strict})	0.60	Early-stop threshold for strong matches
`dtw_band_ratio` (ρ)	0.5	Sakoe-Chiba band ratio
`template_dir`	`../templates`	Path to template library root
`dtw_threshold_<name>`	—	Per-class threshold override
`record_mode`	False	Enable template recording mode
`record_gesture_name`	*empty*	Class name to record into
`identity_verification`	False	Require authorized agent at gesture onset

Table J.1: Configurable parameters of the `gesture` node.