h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Multimodal Audio-Visual Sensor Fusion for Multi-Object Tracking of Speakers in Indoor Environments

Thesis submitted for the degree of
Master of Science (M.Sc.)

Laurens Sillekens

laurens.w.g.sillekens@stud.h-da.de
Student ID: 758676

Supervised by: Prof. Dr.-Ing. Jens-Peter Akelbein
Stephan Gimbel

Submission date: December 01, 2025

ERKLÄRUNG DER URHEBERSCHAFT

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.
Der Inhalte der digitalen und gedruckten Version sind identisch.

Ort, Datum	Laurens Sillekens

ABSTRACT

Motivation: Accurate indoor speaker tracking is important for new applications such as smart environments in building automation. While audio and video sensors each provide useful but incomplete information, combining the two types of sensor and modalities makes the system more robust when one is temporarily not available or unreliable. Progress in multimodal tracking, however, is hindered by the absence of realistic audio–visual datasets with precise ground-truth tracks of speakers. Creating such datasets is costly and difficult due to complex room acoustics, dynamic speaker motion, calibration demands, and synchronization challenges. This thesis is therefore motivated by the need for a reproducible way to simulate realistic multimodal datasets and to benchmark audio-only, video-only, and fused tracking methods, providing a scalable basis for future research in multimodal detection and tracking.

Content: The thesis starts with an extensive literature review about Room Acoustics, Visual Sensing, as well as Sensor Fusion and Tracking to identify the latests state of research about audio-visual simulation, localization and tracking methods suited for indoor environments. An overview about promising state-of-the-art algorithms and software tools is gathered. This overview allows to selected algorithms which fullfill useability, high accuracy, as well as computational efficiency. Next an online and modular software architecture with standardized and streaming compatible interfaces is designed. An experimental setup implements each software module of the proposed architecture using the preselected algorithms. It is shown how the architecture allows online multi-modal simulation, detection and tracking.

Results: Simulated indoor scenarios of varying audio and visual complexity are processed using the proposed framework, producing the datasets MOT25A, MOT25V, and MOT25AV. Audio-only tracking yields high localisation accuracy but low number of detections during speech pauses or acoustic occlusion. Video-only tracking performs reliably in most scenarios, with strong detection and association except under visual occlusion. Audio–visual fusion achieves the highest overall HOTA by combining complementary cues from both modalities and increasing robustness when one modality becomes unreliable. Fusion does not always exceed the best unimodal performance in scenes dominated by strong visual information, but overall provides the most balanced and robust tracking. These results validate the fusion architecture and highlight the benefits of combining audio and visual sensing for multi-speaker tracking in indoor environments.

Keywords: Audio-visual simulation, sound event detection, sound source localization, visual object detection, feature fusion, multi-object tracking, hota

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Tracking multiple speakers in indoor environments is essential for applications such as smart environments in building automation. These systems can improve awareness of people, enable new interaction options, and support energy savings, for example through demand-based air conditioning. Both audio and visual information provide complementary cues about speaker positions and activities, but relying on a single modality is often insufficient. Video tracking can be affected by occlusions, lighting changes, or limited viewpoints, while audio tracking is sensitive to reverberation, noise, and microphone placement. Multimodal sensor fusion combines both modalities to achieve more robust and accurate tracking.

The goal of this work is to develop and evaluate methods for multimodal audio-visual sensor fusion for multi-object tracking of speakers in indoor environments. This includes simulating realistic audio-visual sensor data, designing a feature-level fusion strategy, and analyzing tracking performance compared to unimodal approaches. The following research questions guide this investigation:

Research Question 1. *How can realistic and reproducible audio-visual sensor data for multi-speaker indoor scenarios be simulated to support the evaluation of sensor fusion approaches?* Since collecting large-scale annotated datasets with synchronized audio and video is time-consuming and costly, simulation provides a flexible alternative. The challenge lies in generating realistic acoustic and visual conditions that reflect indoor environments while remaining reproducible for systematic evaluations.

Research Question 2. *How can a feature-level sensor fusion software architecture be designed and realized to effectively combine audio and video data for multi-speaker tracking?* This thesis explores the design and realization of a modular software architecture that enables to localize speakers with audio and video detectors, and fuse the positions as features within a unified tracking pipeline. The focus lies on defining appropriate data representations, synchronization strategies, and to identify detection algorithms and a feature-level fusion mechanism that balances accuracy, robustness and computational efficiency.

Research Question 3. *To what extent does audio-visual sensor fusion improve accuracy and robustness in multi-speaker tracking compared to unimodal tracking in indoor environments?* Unimodal detection and tracking highly depends on consistent availability of the modality. The audio modality can be temporarely unavailable the because the person doesnt speak at all or because of natural pauses in human speech. The video modality can be temporarely or partially unavailable due to visual occlusion by objects in the scene. This thesis quantifies and evaluates the tracking performance of unimodal and multi-

modal tracking, to quantify the performance gains of audio-visual sensor in indoor environments.

In summary, the motivation of this thesis arises from the need for reliable multi-speaker tracking in indoor scenarios. By addressing these three research questions, this work contributes to the development of simulation tools, fusion strategies, and evaluation methods that advance multimodal tracking technologies.

2

BASICS AND RELATED WORK

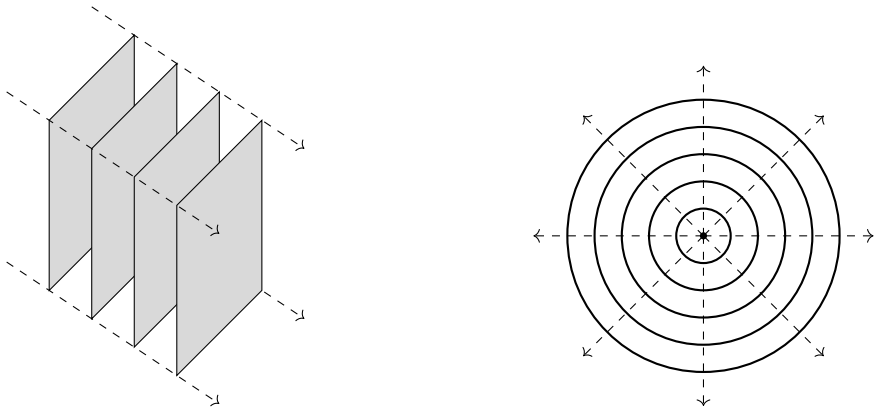
This chapter presents foundational concepts across the subdomains of Room Acoustics in Section 2.1, Visual Sensing in Section 2.2, and Sensor Fusion & Tracking in Section 2.3. Each section follows a wedge-shaped structure, beginning broadly with existing or, as part of this work, proposed classification schemes to provide a comprehensive overview of the respective fields. The discussion then narrows in focus to highlight promising state-of-the-art algorithms relevant to audio and video simulation and localization, as well as for sensor fusion and tracking. This approach enables a structured understanding of each domain and subdomain, while also offering references to related research, research applications and directions for future work.

2.1 ROOM ACOUSTICS

The field of research that investigates the behavior of sound in enclosed or partially enclosed spaces is known as Room Acoustics. This section provides an overview of the fundamentals of sound waves and propagation. It continues with the modeling of room acoustics using room impulse response and classifies virtual acoustics simulation tools. The section concludes with a presentation and comparison of different positional sound source localization techniques.

2.1.1 *Wave-based Acoustics and Geometrical Acoustics*

In room acoustics, the sound field can be considered as a superposition of simple sound waves, e.g., plane and spherical waves, both illustrated in Figure 2.1. According to [1], a plane wave is a progressive wave where the



(a) Plane Wave (b) Spherical Wave

Figure 2.1: Simple types of sound waves [1].

sound pressure is constant in any plane perpendicular to the propagation direction. A spherical wave on the other hand is a sound wave that is propagated uniformly in all directions as concentric wave fronts. Sound waves typically emerge as spherical wavefronts near the source, but as they travel farther, their curvature decreases and they take on the characteristics of planar waves. Since room acoustics deals only with air as the medium, the considerations can be restricted to sound propagation in gases. The sound waves in gases are described by the so-called *wave equation* which depends on the speed of sound [1][2][3]. In large indoor environments, variations in the temperature dependend speed of sound c cannot be entirely avoided. However, the effects caused by such variations are small therefore the speed of sound is often assumed to be constant $c = 343.0$ m/s. [1]

The propagation of sound waves in rooms is affected by multiple basic laws of physics, such as: (1) Reflection, (2) Refraction, (3) Absorption, (4) Diffraction, (5) Superposition and Interference. For more information about each of the basic laws, refer to [1] and [2].

Wave-based Acoustics

In wave-based room acoustics, the primary goal is to solve the *wave equation*. These methods typically subdivide the spatial domain into small elements, allowing the differential wave equations to be approximated by a system of algebraic equations that can be solved using established numerical techniques. Since the element size must be smaller than the shortest wavelength being simulated, these methods are generally limited to low-frequency applications [1].

Geometrical Acoustics

In geometrical acoustics, the concept of a wave described previously is replaced by the examination of single sound rays. A sound ray is considered an infinitesimal part of a spherical wave emitted from a point source. These rays propagate along well-defined directions and follow the same principles as light rays in optics, with the exception of their different propagation speeds. The energy carried by a ray decreases proportionally to the square of the distance traveled, while the pressure amplitude decreases linearly with distance. Rays emitted from a source may carry different amounts of energy, allowing the modeling of source directivity, such as that of the human voice. The straightforward linear propagation of a sound ray can be altered by obstacles or variations within the medium. In such cases, phenomena based on basic laws of physics including reflection, refraction, and diffraction occur [1]. The two types of reflections of sound at surfaces from a geometrical perspective are visualized in Table 2.2.

(a) Specular reflections (b) Diffuse reflections

Figure 2.2: Types of reflections of sound at surfaces from a geometrical perspective [4].

2.1.2 *Room Impulse Response Modeling*

A Room Impulse Response (RIR) is defined as the acoustic response of a room to a unit pulse, which is an infinitely short, high-energy pulse also known as Dirac pulse. It is propagated from the sound source and received by the receiver. It contains the direct sound from the source as well as the sound reflected from the room boundaries. [1] A RIR is therefore the most common way to describe sound propagation between two points in a static environment [4]. The convolution of the RIR with an anechoic signal results in a signal perceived at a given receiver position, which allows to simulate receiver signals in room acoustics [1] [5]. According to [6] and [7], the RIR can be separated in three perceptually and physically distinct components: direct sound, early reflections, and late reverberation as shown in Figure 2.3. The direct sound is the first wavefront to reach the receiver, traveling directly from the source without encountering any surfaces. It provides the most accurate information about the sound source's identity and timing and plays a critical role in speech intelligibility and source localization. Following the direct sound, the early reflections phase consists of sparse, high-energy wavefronts that arrive after reflecting once or twice off nearby surfaces. These reflections

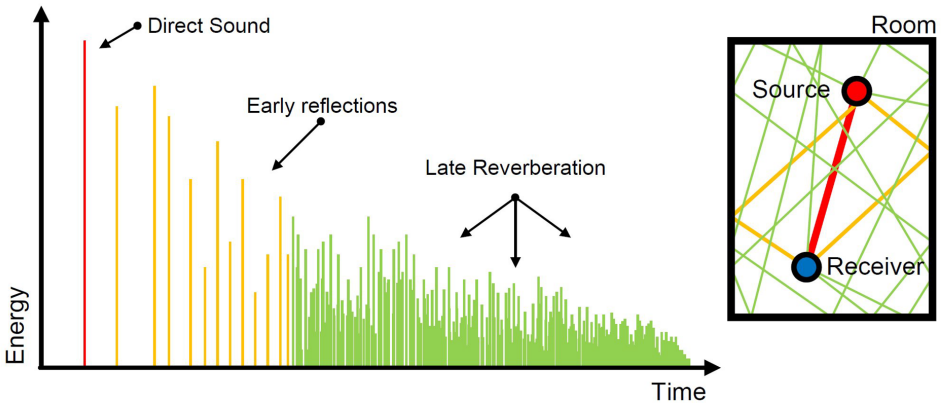


Figure 2.3: Simplified Room Impulse Response of a sound source and receiver in a shoebox shaped room [8].

are typically perceived as distinct echoes and contribute to the perception of spatial cues, including the location and distance of the sound source. Finally, the late reverberation phase comprises a dense and diffuse field of low-amplitude wavefronts resulting from multiple reflections. These wavefronts are no longer perceived individually but are fused into a continuous reverberant tail. Perceptually, late reverberation conveys global characteristics of the acoustic environment, such as room size, geometry, furnishing, and overall absorptive properties. In practice, there are two ways to measure an RIR, as described in the ISO 3382 measurement standard [9]. One way is to excite an impulse, which involves exposing all frequencies at once. Examples of impulses include a loud clap or a balloon pop. The other way is to perform a sine sweep, which involves exposing one frequency at a time over a period. In both cases, the sound pressure is recorded at a specific receiver location. [9][4]

2.1.3 *Synthetic Audio Data Generation*

Room acoustics has traditionally focused on the design of spaces with optimized sound characteristics such as theaters, recording studios, classrooms, and railway stations. The goal has been to predict and shape how sound behaves within physical environments to ensure clarity, intelligibility, and overall acoustic quality. [6] In parallel, the field of virtual acoustics has gained increasing importance, driven by the rapid increase in computational power. These developments now allow highly detailed simulations of sound propagation, not only to support the design and improvement of real-world spaces, but also to enhance immersive virtual reality experiences. [1] Equivalent to the two perspectives in Section 2.1.1, the main approaches used to simulate virtual acoustics in rooms are also differentiated into numerical acoustics and geometrical acoustics [6][7][1].

Numerical Acoustics (NA) try to directly solves the *wave equation*, the fundamental differential equation describing sound propagation in a homogeneous, lossless (non-energy-absorbing) medium such as air, and is therefore relevant for almost all acoustic phenomena [2]. To achieve this, NA discretizes space and time, replaces continuous derivatives with numerical approximations, and then computes step by step how the sound field evolves throughout the domain. With sufficient computational power, this allows NA to capture all wave phenomena, including diffraction and scattering, even in complex environments. Unlike GA, NA is not affected by scene complexity or polygon count but instead scales with the physical dimensions of the problem. While NA provides very accurate results, it is significantly more computationally expensive than GA.

Geometrical Acoustics (GA), in contrast, treats sound as rays that propagate in straight lines and neglects all wave properties. This assumption is valid primarily at mid to high frequencies, where the wavelengths are short

compared to the size of surfaces and the overall dimensions of the room. Unlike wave equation-based NA methods, GA cannot simulate low-frequency effects, such as diffraction. Separate methods are needed for diffraction simulation. Nevertheless, GA is widely used in practice due to its computational efficiency. However, its accuracy decreases at lower frequencies where wave effects dominate, and its computational complexity can increase significantly with complex geometries or large numbers of reflections. [1][10]

Room Impulse Response Generators

Different methods exist to generate RIRs, so-called Room Impulse Response generators, which are further used to convolute with an anechoic signal to simulate the receiver signal for specific combination of sound source position and receiver position. A classification scheme for RIR generators is proposed in Figure 2.4 which contains the most widely used RIR generator methods discussed in the literature. Recently, a new class of neural network-based RIR generators has gained ground this domain. [1, 6, 7, 11, 12].

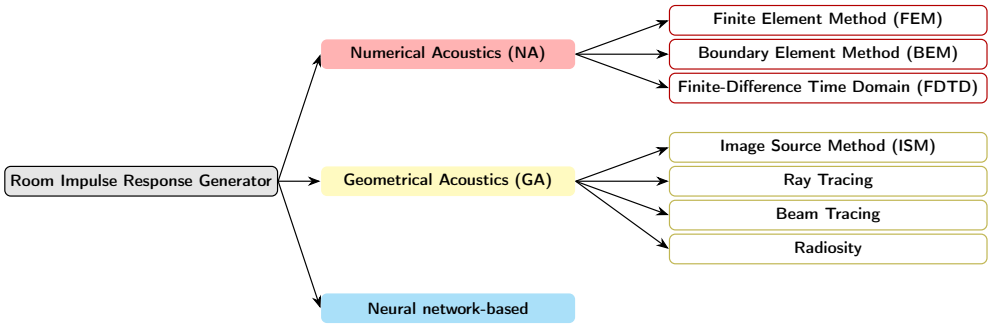


Figure 2.4: Classification of Room Impulse Response generator methods.

Numerical Acoustics RIR generators compute the RIR by numerically solving the *wave equation* within the modeled space [1][11][10]:

The Finite Element Method (FEM) divides the entire volume of the domain into non-overlapping small elements ('finite elements') and solves the wave equation in frequency domain or the time domain by approximating it with a system of equations. It provides high accuracy, especially for complex geometries, but requires fine meshes and large computational resources to simulate higher frequencies.

The Boundary Element Method (BEM) is similar to the Finite Element Method (FEM). The main difference is that the surface of the boundary of the acoustic space is divided into elements, rather than the entire space. These elements can be either continuous or discontinuous. BEM can be applied to both interior and exterior problems. By performing calculations on the boundary instead of the entire acoustic space, the problem is reduced from three dimensions to two. This reduction leads to a smaller boundary element matrix and requires fewer elements to achieve the same accuracy

The Finite-Difference Time Domain (FDTD) Method doesnt solve problems in the spatial domain, but instead discretizes the wave equation directly as a Cartesian grid in the time domain. It computes sound pressure and particle velocity at each grid point by replacing the derivatives in the equations with finite difference approximations. The method is accurate and simple to implement but computationally expensive for large or complex spaces.

Among the mentioned NA-based methods, the FEM can model physical phenomena such as diffraction. However, when applied to room acoustics modeling, its resolution is insufficient. The BEM is preferred for modeling exterior problems, but it does not perform well in complex, inhomogeneous acoustic spaces. In comparison, the FDTD method is more widely applied to room acoustics due to its higher modeling resolution and computational efficiency

Geometrical Acoustics RIR generators estimate the room impulse response (RIR) by simulating sound propagation using simplified geometric models [6][10]:

The Image-Source Method (ISM), proposed by Allen and Berkley in 1979 [13] is a widely used GA method which works by recursively mirroring the sound source across the surfaces of the modeled environment. Each reflection creates an image source, and these are subsequently mirrored again against all surfaces. This recursive process continues until a termination criterion is met, such as a maximum reflection order or a desired response length. The result is a hierarchical tree of image sources, with the original source as the root and each branch representing a reflection path. The final impulse response is computed by summing the contributions of all valid ISs in the tree. ISM can model the Time-of-Arrival of the direct path and specular reflections accurately, but it is restrited to rectangular room shapes. Furthermore, it cannot model frequency-dependent reflection coefficients (portion of the reflected energy to the incident energy) of walls. Despite its limitations, its computational efficiency makes it widely used for generating large-scale databases and even for real-time applications.

Ray Tracing, in contrast involves emitting numerous rays from the sound source and simulating their reflections off room surfaces. Valid acoustic paths are detected by tracking these rays through the environment. This allows to handle specular and also diffuse reflections. However, Ray Tracing methods have a detection problem because the receiver point must be assumed to be a finite size in order to be hit by rays which results in misidentification of rays or duplicated registered rays. Furthermore, Ray Tracing suffers from limited spatial resolution because the number of traceable rays is also limited.

Beam Tracing can either be built upon the idea of ISM or Ray Tracing. As an improvement to ISM, Beam Tracing focuses on eliminating invalid image sources as early as possible to minimize the number of beams. Beam Tracing methods based on Ray Tracing use volumetric beams instead of individual rays. This allows for the detection of specular reflections with improved

efficiency and accuracy in complex geometries, facilitating even real-time applications.

Radiosity methods are view-independent approaches. In acoustics, they are based on an energy balance assuming purely diffuse reflections, where surfaces reflect sound uniformly in all directions. The scene is divided into patches, and energy exchange between patches is determined by form factors and air attenuation factors. Unlike ray-based methods, the form factors are computed once and remain valid for different listener positions, which makes handling moving sources and receivers more efficient. However, radiosity methods are computationally expensive due to the calculation of form factors, and they cannot model specular reflections, diffraction, or transmission accurately. As a result, their application to large or complex environments is limited.

In summary, the ISM is highly efficient and accurately models specular reflections, but it is limited to rectangular rooms and cannot handle frequency-dependent surface properties. Ray Tracing supports arbitrary geometries and diffuse reflections, yet suffers from limited resolution and receiver detection issues. Beam Tracing improves efficiency and accuracy in complex scenes, enabling even real-time use. Radiosity is receiver-independent and models diffuse reflections, but is computationally expensive and cannot capture specular reflections, diffraction, or transmission.

Neural network-based RIR generators have emerged as a new promising approach for simulating acoustic environments. These models leverage deep learning techniques to generate RIRs, offering advantages in terms of speed and adaptability compared to previous explained traditional methods.

IR-GAN is a model that uses a generative adversarial network to synthesise realistic RIRs by learning from real-world data. By parametrically control different acoustic parameters it can imitate new or different acoustic environment. [14]

FAST-RIR is a new method, which uses acoustic environment details like rectangular room dimensions, reverberation time and sender and receiver position as input to rapidly generate diffuse RIRs. This model has demonstrated significant speed improvements over existing GPU-based generators. [12]

MESH2IR is another notable model that extends the approach of FAST-RIR to complex 3D indoor scenes represented using meshes [15].

These neural network-based generators are typically trained on large datasets of measured or simulated RIRs and can generalize across various room configurations offering more flexibility in modeling complex acoustic phenomena.

Virtual Acoustics Simulation Tools

Many virtual acoustics simulation tools implement the RIR generation methods described previously. Tables 2.1 and 2.2 summarize a selection of widely used, actively maintained tools that support multiple sound sources, includ-

ing both research-focused and commercial audio solutions. The tools vary in terms of platform integration, supported domains, source and receiver configurations, simulation methods, computation type, and pricing. For instance, Python-based research libraries such as *Pyroacoustics*, *GSound-SIR*, and *SoundSpaces 2.0* primarily perform offline RIR computation using either the ISM or ray tracing. In contrast, hybrid or specialized tools such as *Virtual Acoustics*, *RAVEN*, and *Treble SDK* provide more sophisticated simulation methods like FDTD or hybrid approaches, though often at the cost of increased computation time and complexity. Commercial solutions such as *Meta XR Audio SDK* and *Wwise + Reflect Plugin* are optimized for real-time interactive applications, mainly in gaming and VR. However, they only support single receivers.

Table 2.1: A comparison of maintained virtual acoustics simulation tools (part 1).

Tool	**Integration**	**Sources**	**Receivers**
Pyroacoustics [16]	Python	Static (Multiple)	Static (Multiple)
GSound-SIR [17]	Python	Static (Multiple)	Moving (Single)
SoundSpaces 2.0 [18]	Python	Static (Multiple)	Moving (Single)
gpuRIR [5]	Python	Moving (Multiple)	Moving (Multiple)
Virtual Acoustics [19]	Python, Unity, UE	Moving (Multiple)	Moving (Single)
RAVEN [20]	Python, Unity, UE	Moving (Multiple)	Moving (Single)
Meta XR Audio SDK [21]	Unity, UE	Moving (Multiple)	Moving (Single)
Wwise + Reflect Plugin [22]	Unity, UE	Moving (Multiple)	Moving (Single)
Treble [23]	Standalone, Python	Static (Multiple)	Static (Multiple)

Table 2.2: A comparison of maintained virtual acoustics simulation tools (part 2).

Simulation Method	**Computation**	**Domain**	**Pricing**
ISM	Offline	Research	Free
Ray Tracing	Offline	Research	Free
Ray Tracing	Offline	Research	Free
ISM (with GPU acceleration)	Real-time	Research	Free
FDTD	Offline	Research	Free
Hybrid (ISM + Ray Tracing)	Offline	Research	Free (Research)
Ray Tracing	Real-time	Gaming	Free
Hybrid (ISM + Ray Tracing)	Real-time	Gaming	Commercial
Hybrid (Ray Tracing / Radiosity + FEM)	Real-time	Room Optimization	Commercial

Among these options, *gpuRIR* stands out due to several key advantages for research-oriented applications requiring flexible and efficient simulation. Unlike many offline tools, the library leverages GPU acceleration to perform ISM-based RIR computation in real time, allowing for multiple moving sources and receivers. It is fully open-source, Python-based, and does not depend on a game engine, making it highly accessible for experimental setups. Its combination of efficient computation, support for multiple moving sources and receivers, ease of integration and free usage made *gpuRIR* the preferred choice over other simulation tools for the experiments conducted in this thesis.

2.1.4 *Positional Sound Source Localization*

To localize sound sources like speakers, the 3D-SSL algorithm from the category Positional Sound Source Localization (PSSL) can be used, which can determine the position of a sound source under near-field conditions. Under near-field conditions, sound waves propagate in a spherical shape as shown in Figure 2.1. Consequently, the acoustic signal arrives at the microphones at slightly different times. The relative time difference between the arrival of the signal at two microphones is referred to as the time difference of arrival (TDOA). In 2-dimensional space, the TDOA between two microphones defines a hyperbola, while in 3-dimensional space it corresponds to a hyperboloid surface, as shown in Figure 2.5. By intersecting several independent hyperbolas, the position of a sound source in 2D space can be estimated. Similarly, the intersection of three or more hyperboloids enables the estimation of a sound source position in 3D space, making 3D PSSL possible. With PSSL, the position of sound sources is always determined in Euclidean coordinates (x,y,z). Prior research conducted by me in [24] evaluated the localization error of the PSSL algorithm. The 3D-SSL algorithm introduced in that work is summarized below and employed as an audio detector later in this thesis.

3D-SSL: 3D Localization Solution in Closed-Form

An analytical approach is used to determine the 3D position of the sound source by solving a non-linear system of equations. At least four audio signals are required to determine the three spatial coordinates (x, y, z), one of which is used to linearize the equations. Since an additional microphone serves as a reference for the TDOA calculation, at least five microphones are required to determine the 3D coordinates. [25][26] An analytical solution for 3D PSSL using five or microphones is proposed in [27]. A 3D sound source position x, y, z is determined using the closed-form solution and at least five non-coplanar microphones. This method is referred to as 3D Sound Source Localization (3D-SSL). To assess the reliability of the 3D-SSL solution, it is essential to evaluate the sensitivity of the system to errors in TDOA estimates. A common measure for this is the condition number, which indicates the stability index of the solution [25]. Under the same noise conditions, the stability of the solution and thus the localization performance are related. Small values for the condition number indicate that the influence of measurement errors on the solution is low and the localization is therefore stable. Large values indicate that measurement errors have a significant influence on the solution and the localization is unstable. [28] The condition number is calculated as part of the localization in order to filter out unstable solutions of the 3D-SSL. To further increase the robustness of the localizations, the localization results are spatially filtered based on the known spatial size. Furthermore, a combinatorial method based on the binomial coefficient is used to select 4 out of 7 TDOAs, resulting in 35 localizations of a sound source. This approach mitigates the effects of outlier TDOAs, which are more likely

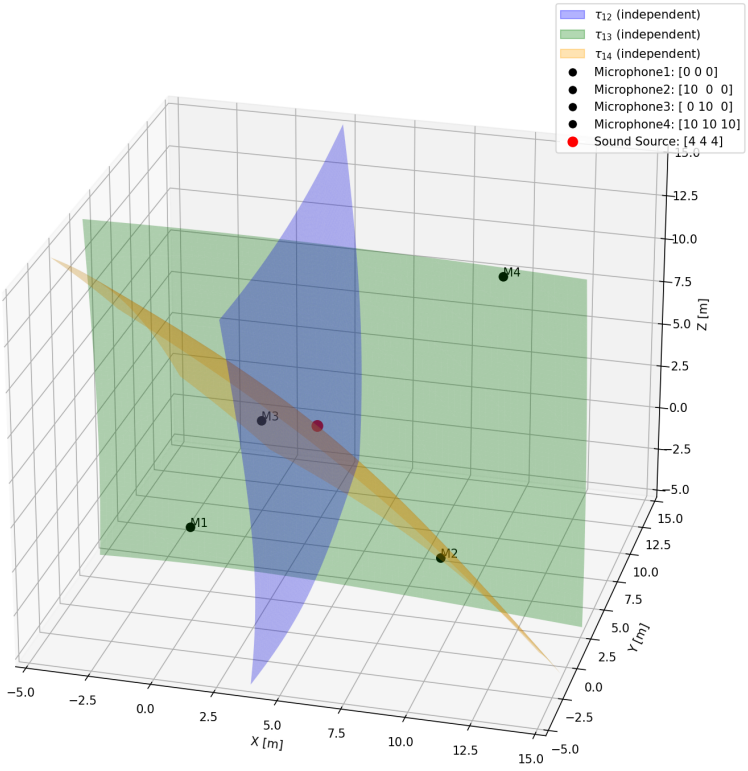


Figure 2.5: The 3D position of a single sound source is derived using multilateration of four microphones.

to not falsify all combinations, as opposed to using all TDOAs at the same time.

2.2 VISUAL SENSING IN INDOOR ENVIRONMENTS

Visual sensing plays a crucial role in indoor environments, providing complementary information to audio-based methods for localization, mapping, and tracking. In combination with audio data, visual information enhances the accuracy and robustness of multi-object tracking systems, especially in complex or cluttered spaces. This section introduces key concepts, methodologies, and challenges associated with visual sensing in indoor settings.

2.2.1 *Video Measurements*

Video measurements form the foundation of visual sensing. Cameras capture the dynamic state of indoor environments and serve as primary data sources for localization and tracking. Different types of cameras can be employed, including monocular, stereo, depth (RGB-D), and omnidirectional cameras, each with specific advantages in terms of resolution, field of view, framerate and depth perception [29][30].

Key challenges in video measurements include (1) variations in lighting, (2) occlusion of objects or individuals, and (3) motion blur. Evaluation metrics for video-based systems typically consider localization accuracy, temporal resolution, robustness to occlusion, and computational efficiency. Proper understanding of these factors is essential for designing reliable visual sensing pipelines.

Many real-world image datasets using wide field-of-view cameras, such as FRIDA [31], DETRAC [32], or WEPDTOF [33], exist. However, these datasets are typically annotated in image space only, using bounding boxes. LOAF [34] is the first real-world dataset providing annotated metric 2D human positions obtained through camera calibration. This lack of annotated metric 3D positions of humans, motivates the generation of synthetic image data. Moreover, synthetic image data can be combined with synthetic audio data generation approaches described in Subsection 2.1.3, enabling the creation of synthetic multimodal datasets containing both audio and video modalities.

2.2.2 *Synthetic Image Data Generation*

The literature research of this work identifies three review papers that cover the research literature on synthetic image data generation in computer vision [35], indoor scenes [36], and video surveillance [37]. The reviews identify two synthetic data generation methods, *3D modeling-based methods*, and *3D computer graphics engine-based methods* to generate images from scratch. Other methods like generative networks, which are not able to provide annotations or image composition which past or fuse images and require previous image data, are therefore considered outside the scope of this work.

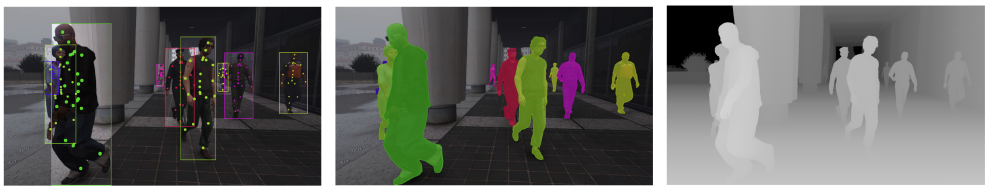
3D modeling-based methods, such as the open-source software Blender, enable the creation of 3D virtual environments from basic code and leverage modelling tools equipped with integrated physics engines.

3D computer graphics engine-based methods on the other hand repurpose engines initially designed for game development to facilitate data generation. Rendering engines such as *Unity*, *Unreal*, *Rockstar Advanced Game Engine (RAGE)* are powerful tools for building virtual worlds with precise geometry, material and object placement. These environments are used to perform controlled experiments and generate synthetic data.

Synthetic data generation is widely used in recent research for evaluating algorithms in general but also for training and validating neural networks. Synthetic data is used in the related domains of human or pedestrian detection [38] [39] [40] [41] [42], human pose estimation [43] [44] [45] [46], human activity tracking or safety monitoring [47] [48], and human tracking [49] [50]. In the publication [50], the authors present, *MOTSynth*, the largest synthetic video dataset to date for the purposes of pedestrian detection, re-identification, segmentation and tracking in urban scenarios. An image sample of the dataset and its annotations is shown in Figure 2.6. By employing Computer Vision-based Multi-Object Tracking models, it is possible to train such models purely on synthetic data and achieve state-of-the-art results for real, crowded scenes. The experimental evaluation in [50] confirms that the *MOTSynth* dataset is a valid replacement for real data, with diversity playing a pivotal role in bridging the synthetic-to-real gap. *THEODORE* [41] and *THEODORE+* [46] are two other synthetic video datasets simulating top-down omnidirectional fisheye cameras in indoor scenes using the rendering game engine Unity. An image sample of the dataset and its annotations is shown in Figure 2.7.

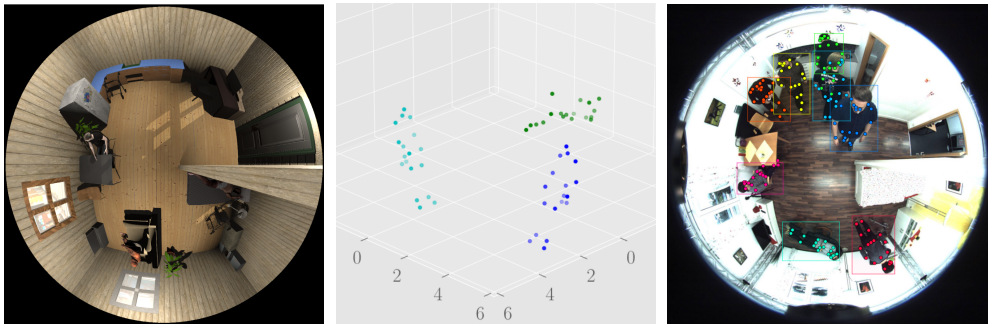
An overview, comparing large scale synthetic video datasets is shown in Table 2.3 and can be used as a starting point to identify a suited dataset based on the target application.

While synthetic image data generation has seen widespread adoption in research, several key challenges and research opportunities, collectively known as the *Simulation-to-Reality Gap*, remain. These include [35]: *Domain Gap* refers to the discrepancy between datasets collected from different domains. *Data Diversity* concerns the quality of data and the ability of models trained on a given dataset to generalise to other domains. *Photorealism* aims to make synthesised data appear as close as possible to real-world captures,



(a) Bounding boxes and pose (b) Segmentation masks (c) Depth maps

Figure 2.6: The *MOTSynth* dataset as an example for a synthetic, annotated, large and diverse video dataset using the highly photorealistic graphics engine *RAGE* [50].



(a) Fisheye RGB Image (b) 3D Skeleton Keypoints (c) Bounding boxes + 3D Skeleton Keypoints

Figure 2.7: The *THEODORE+* dataset as an example for a synthetic, annotated, top-down fisheye video dataset using the graphics engine *Unity* [46].

Dataset	Simulator	Camera Projection	#Frames	3D	Pose	Segmentation	Depth
VIPER [51]	RAGE	Perspective	254k	√	–	√	–
GTA [52]	RAGE	Perspective	250k	–	–	√	√
JTA [49]	RAGE	Perspective	460k	√	√	–	–
MOTSynth [50]	RAGE	Perspective	1,382k	√	√	√	√
THEODORE [41]	Unity	Fisheye	100k	–	–	√	–
THEODORE+ [46]	Unity	Fisheye	50k	√	√	√	–

Table 2.3: A comparison of labeled large-scale synthetic video datasets.

with the core idea that training on photorealistic data will reduce the domain gap to the target domain. *Data Benchmarking* often lacks a central method to evaluating the effectiveness of synthetic data.

To ensure consistency with an existing real-world setup and to minimize the *Simulation-to-Reality Gap*, custom scenes are generated using *Unity*. This choice is driven by several factors, including the need to simulate a camera with the same field of view as the real setup. Precise camera calibration data is used to distort fisheye images in order to obtain realistic distorted camera images for object detection. The same parameters are required to undistort the fisheye images and to extract accurate geometric information for mapping localization from image coordinates to real-world coordinates. Custom simulation experiments furthermore allow to start with scenes with lower complexity and increase as necessary. Recent, Unity-based datasets, such as *THEODORE* and *THEODORE+* have demonstrated the success of the computer graphics rendering engine in top-down object detection for indoor environments. This makes *Unity* the preferred choice over *Unreal* or *RAGE* for synthetic image data generation in this work.

2.2.3 *Camera Calibration and Distortion Correction*

A camera model is a mathematical representation of a real camera that describes the projection, sometimes called mapping, of a three-dimensional scene point (world coordinates) to a two-dimensional image point (image

coordinates). The pinhole camera model is considered the ideal model for an imaging device and is used in most computer vision tasks. It suites well for narrow-angle and wide-angle cameras. [53]

Geometric camera calibration is the acquisition process of information regarding the extrinsic and intrinsic parameters. Those parameters are then used to apply distortion correction. A schematic representation of the pinhole model and both parameter types is shown in Figure 2.8. The camera clibration process typically consists of four steps: data collection, target extraction, calibration optimization and result verification [53].

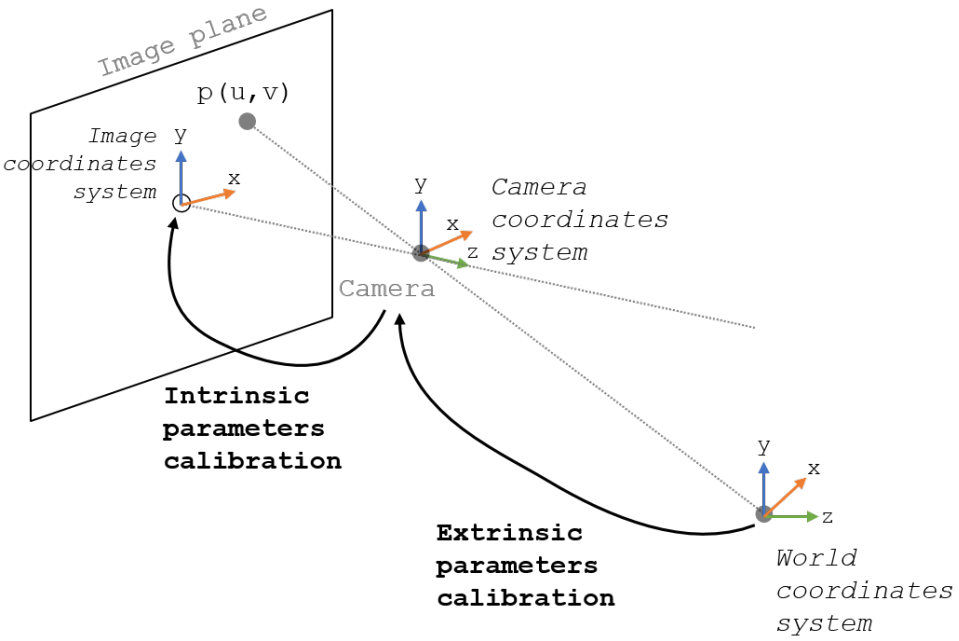


Figure 2.8: A schematic representation of the pinhole camera model. Its perspective projection is described by extrinsic and intrinsic parameters [54].

Extrinsic parameters describe be the position and orientation of the camera relative to the world coordinate system and are used to transform points from the object space to the camera coordinate system. The included parameters are camera orientation and camera translation.

Intrinsic parameters describe the camera's internal characteristics that affect how rays are projected in the camera frame to the imaging sensor. [53] The parameters include the focal length, which defines the distance between the optical center of the camera and the sensor. It is scaled by pixel size annotated as f_x and f_x in the x- and y-direction. The optical axis is the line passing through the optical center and orthogonal to the sensor chip. The foot of optical axis on the sensor chip is the principal point. The latter is scaled by pixel size and annotated as c_x and c_y in the x- and y-direction. This parameter set $i = [fx, fy, cx, cy]$ fully describe the distortion-free pinhole camera model. [53] Distortion coefficients denoted $k_1, ...k_n$ describe mathematically how the camera lens distorts light rays. The amount of distortion coefficients highly

depends on the camera model. The survey [53] provides presents existing geometrical camera calibration tools to estimate the intrinsic parameters and evaluates the most representative ones for wide-angle cameras. The survey covers the camera models, calibration targets and algorithms used in these tools. This thesis is exclusively concerned with intrinsic calibration, as the extrinsic parameters can be obtained directly from simulation or measured in real-world with minimal effort.

Photogrammetric calibration is a conventional method of geometric camera calibration that uses a calibration target with a known 3D geometry. Features such as corners, dots, lines, and circles are extracted from the target. This approach is accurate, stable, and simple, making photogrammetric calibration one of the most widely used methods. Other approaches include *self-calibration* without a calibration target or prior knowledge of the scene [55] [56]. In the last ten years also deep learning-based camera calibration gained a rapid increase in research and are presented in the survey [57]. The subsequent text of this subsection focuses on a conventional photogrammetric calibration method that utilises a calibration object with a planar pattern whose geometry is known. The more self-calibration and deep learning-based methods are outside the scope of this thesis due to their higher complexity.

Fish-eye cameras, unlike narrow-angle and wide-angle cameras, use extremely wide-angle fisheye lenses to cover the entire hemispherical area in front of the camera. Therefore, they typically have a field of view of close to 180 degrees or higher. Unfortunately those lenses leads to the undesirable effect that they deviate significantly from the pinhole model, introducing high levels of geometric non-linear distortion. Distortion causes straight lines in images to bend.[58] [55] There are several types of distortion, including: Radial distortion, Tangential distortion, as well as non-linear distortion. The purpose of camera distortion correction is to transform the distorted image captured by the camera into an image that resembles the ideal image produced by a pinhole camera. Distortion correction transform the distorted view of wide-angle cameras into a pinhole perspective. Consequently they are important pre-processing tasks for computer vision applications that require geometric information to be extracted from the scene [55] [56].

Kannala-Brandt Model

The Kannala-Brandt (KB) model, proposed by Juho Kannala and Sami Brandt in [59], is a generic camera model that well fits regular, wide angle and fisheye lenses. The KB model assumes that the distance from the optical center of the image to the projected point is proportional to the polynomial of the angle between the point and the principal axis. Kannala-Brandt characterizes distortion as a function of the incidence angle of the light passing through the lens. This is done because the distortion function is smoother when parameterized with respect to this angle, which makes it easier to model as a power-series. The full KB model defined in [59] has 23 parame-

ters, where four describe the affine transform, five describe an equidistant radial-symmetric distortion, and the other 14 describe the asymmetric distortion. In practice, most systems will characterize Kannala-Brandt distortions purely in terms of the symmetric radial distortion, as that distortion is significantly larger in magnitude and will be the leading kind of distortion in extremely wide-angle fisheye lenses. This results in the commonly used KB-8 model that is radially symmetric and has 8 intrinsic parameters, $i = [f_x, f_y, c_x, c_y, k_1, k_2, k_3, k_4]$ [60]. Each intrinsic parameter has the following meaning:

- f_x and f_y are the focal lengths along the x and y axes, expressed in pixels.
- c_x and c_y represent the coordinates of the principal point, typically near the image center.
- k_1 to k_4 are the coefficients of the polynomial distortion function, modeling radial distortion as a function of the ray angle.

Figure 2.9 provides a schematic representation of the KB and pinhole model, visualizing the projection of a 3D point P in world-coordinates (XYZ) to the image projection point p in image-coordinates (xy) using the KB model and comparing it to image projection point p' of the pinhole model. It furthermore illustrates essential geometric parameters used in the subsequent text which provides a detailed explanation of the mathematical definitions of the KB-8 forward and backward projection based on [53].

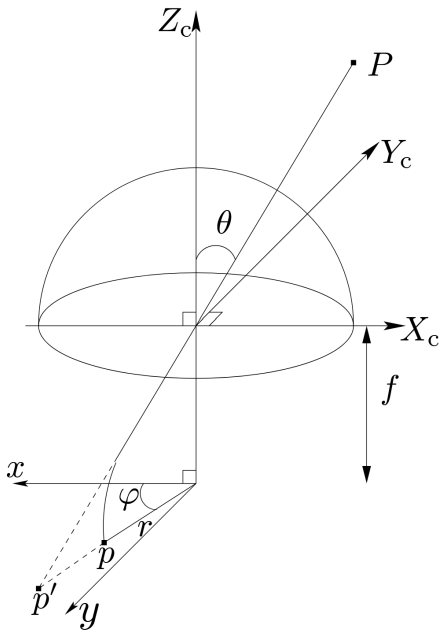


Figure 2.9: A schematic representation of the Kannala-Brandt and pinhole projection through a camera lens. The 3D point P is projected to p whereas it would be p' by a pinhole camera [59].

Forward Projection
In the KB-8 forward projection, a 3D point in camera coordinates

$$P = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

is projected to 2D image point in pixels

$$p = \begin{bmatrix} u_m \\ v_m \end{bmatrix}$$

using the the forward transformation function $\pi(P, i)$:

$$p = \pi(P, i) = \begin{bmatrix} \frac{f_x d(\theta) X_c}{r_c} + c_x \\ \frac{f_y d(\theta) Y_c}{r_c} + c_y \end{bmatrix} \tag{2.1}$$

The angle θ between the incoming ray and the optical axis is given by:

$$\theta = \arctan\left(\frac{r_c}{Z_c}\right) \tag{2.2}$$

The radial distance from the optical axis in the XY plane is defined as:

$$r_c = \sqrt{X_c^2 + Y_c^2} \tag{2.3}$$

The radial distortion function $d(\theta)$ is modeled as a polynomial in θ:

$$d(\theta) = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9 \tag{2.4}$$

Backward Projection
The backward projection maps a 2D image point p back to a unit vector or ray in the 3D camera coordinate system. The process involves inverting the distortion function $d(\theta)$ to recover the angle θ, which is not analytically invertible and must be solved numerically [59]. The following equations are based on [61] and [62]:
First, the image pixel coordinates are normalized:

$$x_d = \frac{u_m - c_x}{f_x}, \quad y_d = \frac{v_m - c_y}{f_y} \tag{2.5}$$

The distorted radial distance is then computed:

$$r_d = \sqrt{x_d^2 + y_d^2} \tag{2.6}$$

To find θ, we solve the following equation:

$$d(\theta) = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9 = r_d \tag{2.7}$$

This is a high-order polynomial equation and is typically solved using iterative methods such as Newton-Raphson [59]. Once θ is estimated, the resulting direction vector can be computed using the backward transformation function $\pi^{-1}(p, i)$:

$$P = \pi^{-1}(p, i) = \begin{bmatrix} \frac{x_d}{r_d} \cdot \sin(\theta), \\ \frac{y_d}{r_d} \cdot \sin(\theta), \\ \cos(\theta) \end{bmatrix} \tag{2.8}$$

While there are some other camera models then the KB model suited for the calibration of fisheye lenses and distortion correction, these are beyond the scope of this thesis. Additional camera models and comprehensive reviews can be found in [53] and [60].

2.2.4 *Deep Learning-based Object Detection*

Object detection is a fundamental task in computer vision that involves identifying and localizing objects within an image or video frame. It extends beyond simple image classification by not only recognizing object categories but also determining their spatial positions using bounding boxes. The primary objective of object detection is to accurately detect multiple objects of varying scales, orientations, and occlusions, while maintaining real-time performance and robustness across diverse visual conditions. This capability is critical for numerous applications, including video surveillance, autonomous driving and robotics. [63]

Over the past two decades, significant progress has been made in the field of object detection, transitioning from traditional machine learning-based methods to deep learning (DL) architectures. Traditional approaches relied on hand-crafted features, which were limited in their ability to generalize across complex visual environments. The emergence of deep learning fundamentally transformed this landscape. Deep learning-based detectors learn hierarchical representations directly from raw pixel data, enabling superior performance in accuracy, robustness, and computational efficiency. [63]

Recent surveys [64] [65],[66] have reviewed state-of-the-art DL object detection methods and proposed a categorisation of these methods into one-stage and two-stage detectors, as illustrated in Figure 2.10. As DL-based object detection is a research field of rapid development, other classification schemes and models are frequently published, often outperforming existing ones. However, it should be noted that, as part of this work, the focus is exclusively on the aforementioned two types of detectors. The differences between these detector classes are explained in a demonstrative manner, and it is important to acknowledge that they can be replaced by a wide variety of DL-based object detectors using bounding boxes to detect objects.

The introduction of *Convolutional Neural Networks (CNNs)* marked a pivotal moment in the development of object detection. CNNs automatically extract spatial hierarchies of features, allowing models to learn discriminative object representations without manual feature engineering. This innova-

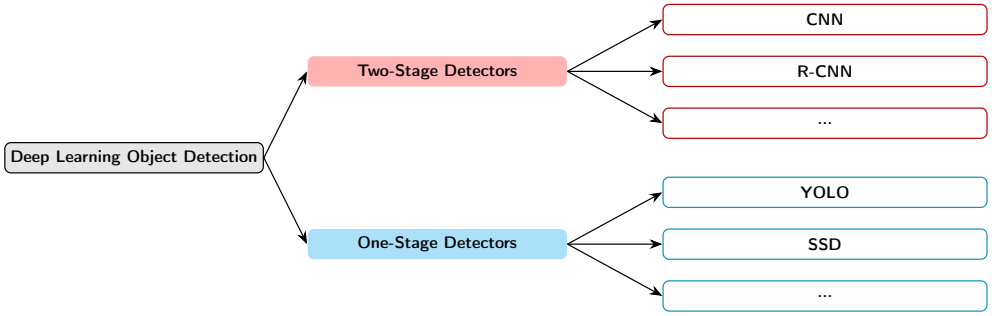


Figure 2.10: Classification of object detection methods.

tion laid the groundwork for subsequent advancements, such as *region-based CNNs (R-CNN)* and their successors, which significantly improved localization precision and object recognition accuracy. [66]

The Two-Stage Detectors include a first stage which generates a set of candidate regions. The second stage then classifies these proposals and refines their bounding boxes. While this approach yields high accuracy and strong localization performance, it is computationally more intensive and less suited for time-critical applications.

The need for faster and more efficient object detection models has led to the development of One-Stage Detectors, such as the *You Only Look Once (YOLO)* and *Single Shot MultiBox Detector (SSD)* family. These architectures redefined real-time object detection by removing the need for intermediate region proposal stages. YOLO, in particular, reformulated object detection as a single regression problem, predicting bounding boxes and class probabilities directly from entire images in one unified process. This end-to-end design enables high-speed inference while maintaining competitive accuracy, making such models ideal for real-time applications. [66]

The research work preceding this thesis in [67] examined various state-of-the-art object detection model families evaluating their detection accuracy and inference time. The gained results confirmed the theoretical superiority of the One-Stage Detector *YOLO* family by demonstrating the highest accuracy with the lowest inference time. Further research results of the preceding paper showed that a small "nano"-size YOLO modell using only 8 MB of memory in Float32 format was able to be deployed on a low-cost embedded device equipped with an neural processing unit, showing real-time object detection performance. [67]

2.3 SENSOR FUSION AND TRACKING

This section describes what sensor fusion is about. It also explains which types of sensor fusions can be differentiated according to their configuration (complementary/competitive/cooperative). The motivation behind sensor fusion is shown and the goals which can be achieved with it.

2.3.1 *Motivation and Categorization of Sensor Fusion*

Definition 1 *„**Sensor fusion** is the combination of sensory data or data derived from sensory data in such a way that the resulting information is in some way better than the information that would be possible if these sources were used individually.“[68]*

There are a number of characteristics that demonstrate the relevance and motivation of sensor fusion [69][70]:

1. **Single point of failure**: A single failing sensor can lead to a loss of perception of complete objects.
2. **Limited spatial coverage**: One sensor usually only covers a very limited range compared to multiple sensors.
3. **Limited time coverage**: Sensors require a certain set-up time to perform and transmit a measurement. This maximum frequency of measurements limits the time coverage.
4. **Limited accuracy**: A measurement of an individual sensor is limited to the accuracy of the used sensor.
5. **Limited certainty**: Certainty describes how sure the system is about the perception of a sensed object. Multiple sensors can give contradictory measurements, which can be interpreted as uncertainty.

Sensor fusion categorization types

Sensor fusion can be categorized by different aspects resulting in two common categorization types which are explained in the following.
The *Three-Level Categorization* distinguishes sensor fusion methods according to the system level in which the sensor fusion is implemented [71][72][68]:

1. **Low-level fusion / Raw data fusion** combines several sources of raw data to produce new data that is expected to be more informative than the inputs.
2. **Intermediate-level fusion / Feature level fusion** combines various features such as edges, corners, lines, textures, or positions into a feature map that may then be used for segmentation and detection.

3. **High-level fusion / Decision fusion** combines decisions from several experts. Methods of decision fusion include voting, fuzzy-logic, and statistical methods.

The *Categorization by Sensor Configuration* distinguishes between the following three types [73][74][68]:

A **Complementary**
A sensor configuration is called complementary if the sensors are not directly dependent on each other, but can be combined to obtain a more complete picture of the phenomenon being observed. Each sensor observes a different part of the surrounding space.

B **Competitive / Redundant**
Sensors are described as competing if each sensor provides independent measurements of the same property. A distinction is made between the fusion of data from different sensors, and the fusion of measurements from a single sensor performed at different points in time.

C **Cooperative**
A cooperative sensor network uses the information provided by two independent sensors to derive information that would not be available from the individual sensors, and is the most difficult to design because the resulting data is sensitive to inaccuracies in any of the individual sensors involved. Therefore, unlike competitive fusion, cooperative sensor fusion generally reduces accuracy and reliability.

In this thesis a complementary sensor setup is implemented that consists of a distributed microphone array together with a fisheye camera. Feature-level sensor fusion is selected because state-of-the-art unimodal audio and video detectors show strong individual performance, making the combination of their extracted features a suitable choice. Nevertheless, both raw data fusion using the original audio and image signals and decision-level fusion based on complete object tracks or trajectories would also be valid approaches for this task. In the present work, features correspond to audio or video detections in Euclidean coordinates. Feature-level sensor fusion combines time-dependent audio and video features into a two- or three-dimensional map. Multi-object tracking builds on this unified feature map, associating features over time to create object tracks (linked time-dependent positions) of speakers. It is worth mentioning that the predicted speaker tracks naturally include information about the number of speakers in the measurement area.

2.3.2 *Classification of Multi-Object Tracking Methods*

Multi-object tracking (MOT) involves detecting multiple objects in data, such as video or audio, and maintaining consistent identities for those objects across frames of the data. Since many components interact, such as detection,

association, appearance, and motion modeling, the literature uses several different classification schemes. Figure 2.11 provides a compact overview that is useful for classifying different MOT methods. The four common perspectives that appear across multiple surveys and reviews [75–78] are summarized to propose an classification overview which is explained in more detail in the following.

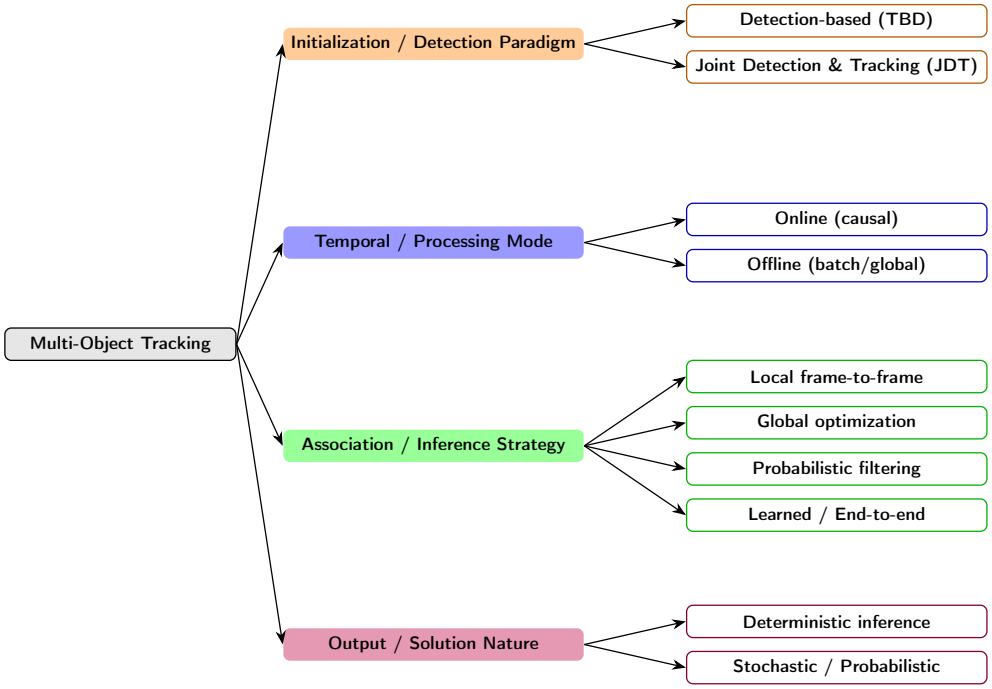


Figure 2.11: Classification perspectives for Multi-Object Tracking.

1. INITIALIZATION / DETECTION PARADIGM One way to distinguish MOT approaches is by how objects are initialized and detected across time:

- *Detection-based or Tracking-by-Detection (TBD)*: Each frame is first processed by an object detector. Then, the detections are linked across time. This paradigm is the dominant, widely used approach in modern MOT because of its modularity, how it naturally handles object entry and exit and takes advantage of strong single-frame detectors.
- *Joint Detection and Tracking (JDT)*: These end-to-end deep-learning approaches simultaneously detect and track objects in a unified framework. They take advantage of temporal and spatial information for robust tracking but often increase computational complexity compared to TBD.

2. TEMPORAL / PROCESSING MODE Another perspective is how temporal information is processed:

- *Online (causal)*: Only past and current frames are used, making it suitable for real-time and latency-sensitive applications.

- *Offline (batch / global)*: Several or all frames are processed together. While this method typically yields cleaner and more consistent tracks, it requires more memory and is not suitable for real-time applications.

3. ASSOCIATION / INFERENCE STRATEGY A third dimension is how data association and inference are carried out:

- *Local (frame-to-frame)*: A simple, fast matching between detections in adjacent frames using metrics such as Euclidean distance or IoU. It is good for real-time systems but prone to identity switches when objects are close to each other.
- *Global optimization*: The decisions are made jointly over many frames, often by solving an optimization problem. While this reduces the number of identity switches, it is more computationally expensive.
- *Probabilistic filtering*: These methods maintain probability distributions over the possible states of each object. They are useful when uncertainty is high, for example in noisy or cluttered environments.
- *Learned / End-to-end association*: Recent deep learning approaches learn the association step directly inside a network, which allows them to use rich spatial and temporal information.

4. OUTPUT / SOLUTION NATURE Finally, approaches can be distinguished by the nature of their output:.

- *Deterministic inference*: A fixed output for a fixed set of inputs (e.g., Hungarian matching based on fixed scores).
- *Stochastic / Probabilistic inference*: Outputs vary between runs for methods that explicitly model uncertainty, such as particle filters and stochastic sampling.

Table 2.4 shows a selection of state-of-the-art MOT algorithms, classified by detection paradigm, processing mode, inference strategy, and nature of solution.

Table 2.4: Overview and comparison of multi-object tracking algorithms.

Algorithm	**Detection Paradigm**	**Processing Mode**	**Inference Strategy**	**Solution Nature**
SORT [79]	TBD	Online	Local	Deterministic
DeepSORT [80]	TBD	Online	Local	Deterministic
ByteTrack [81]	TBD	Online	Local	Deterministic
FairMOT [82]	JDT	Online	Learned	Deterministic
MOTR [83]	JDT	Offline	Learned	Deterministic
TrackFormer [84]	JDT	Offline	Learned	Deterministic
MS-GLMB [85]	TBD	Online	Probabilistic Filtering	Stochastic
HybridTrack [78]	JDT	Online	Local + Learned	Stochastic

Among the selection of MOT methods, MS-GLMB was selected for this thesis to implement an online tracking-by-detection (TBD) approach. Current state-of-the-art joint detection and tracking algorithms (JDT) are typically limited to video-only modalities. Compared to learned MOT algorithms, MS-GLMB offers several advantages for research-oriented applications. Its efficient probabilistic filtering naturally handles sensor noise and clutter, resulting in robust tracking performance, even when there is uncertainty. These properties make the MS-GLMB algorithm a promising choice for audio-visual speaker tracking.

2.3.3 *Multi-Sensor Generalized Labeled Multi-Bernoulli Filter*

The MS-GLMB algorithm is a multi-object tracking approach designed for linear Gaussian models and multiple sensors. It's efficient multi-object tracking pipeline combines the following three main ideas: the GLMB recursion provides a principled Bayesian framework for representing and propagating multiple labeled objects [86]. Next, the sampling-based update step makes the data association problem tractable by generating data association hypotheses. This works even when using several sensors and large measurement sets [85]. Finally, the adaptive birth mechanism ensures that newly appearing objects are detected in a data-driven and reliable way [87]. This subsection introduces these key components in a conceptual way. For detailed mathematical derivations refer to the original papers [86] [85] [87].

Labeled Random Finite Sets and the GLMB Filter

A labeled Random Finite Sets (RFS) is a mathematical representation to model systems where the number of objects and their states are both uncertain. The labeled RFS is a set for timestep k which describes each object not only by its object state (e.g. 6D [x, vx, y, vy, z, vz] using 3D position and velocity), but also by a unique label. Linked object states of positions and velocities are called trajectories. Figure 2.12 shows an example of time-dependend object states and the labeled trajectories. One labeled RFS per timestamp describes the objects present at that time, their object states X_k and assigns as unique label l to each object. This label acts as a stable identifier across time, which is crucial for trajectories. Labeled RFSs are used in MOT to mathematically efficient describe estimated object states, the so-called State Space, as well as noisy sensor observations, the Observation Space. [88] A Multi-Object System describes both spaces and is visualized in Figure 2.13. It can be seen that the number of states X and observations Z vary from time $k-1$ to the next time frame k. Existing objects may not be detected, false observations may occur, and it is not known which observations originated from which objects.

The GLMB is a probabilistic model for labeled RFS that provides a closed-form solution for multi-object system. It does this by represents the multi-object state as a set of hypotheses. Each hypothesis corresponds to a particular combination of existing object labels and their measurement associa-

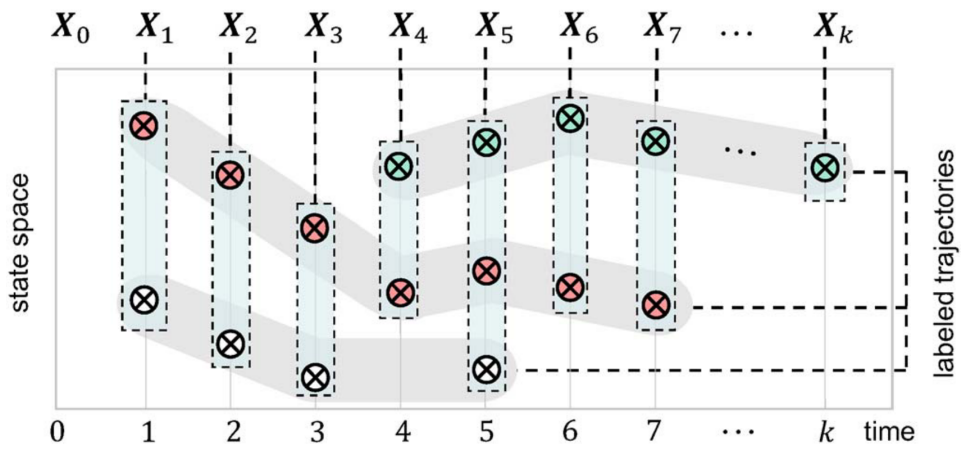


Figure 2.12: Labeled assignments in Random Finite Sets represent multi-object states and trajectories. [88]

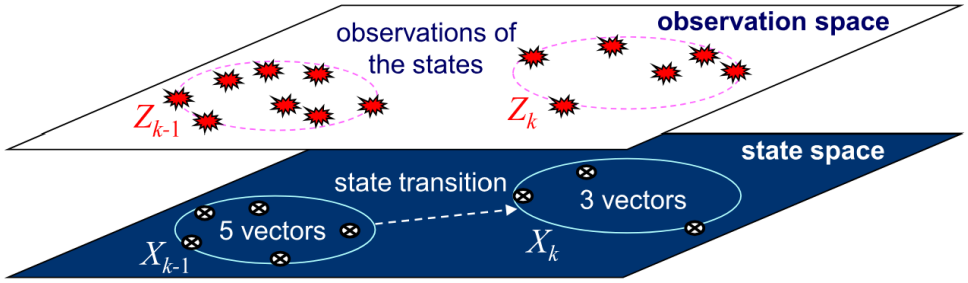


Figure 2.13: The State Space Model and Observation Space Model of a Multi-object system. [88].

tion histories. Each hypothesis has a weight that reflects its probability, and each labeled object carries a spatial probability distribution representing its estimated state. The GLMB furthermore keeps track of measurement association histories to ensure that objects are consistently matched with their measurements across time steps. The GLMB form is preserved during both prediction and update, providing a principled Bayesian framework:

- The **Prediction** step is performed at each time step, each labeled object can either survive and evolve to a new state, or disappear. New objects can also be born. The GLMB prediction step combines these possibilities to generate the predicted multi-object state while preserving object labels. This ensures that each object maintains its identity over time.
- An **Update** step is performed when measurements are received, the GLMB update step adjusts the weights and spatial distributions of the hypotheses based on which measurements are likely to have originated from which objects. False alarms and missed detections are naturally

handled in this process. After the update, the multi-object state remains a GLMB, allowing the next prediction step to proceed recursively.

Sampling-Based Approximation of Data Associations

The drawback of GLMB is, that when the number of labels or measurements grows, the number of hypotheses can become very large. To make the Update step more efficient, the MS-GLMB algorithm replaces deterministic ranking with a sampling-based approximation. Instead of computing a complete ordered list of association hypotheses, the method draws samples that are more likely when they explain the sensor data well and less likely otherwise. This is sampling method is named Gibbs sampler [85]. It produces high-quality hypotheses without enumerating the full combinatorial space. The strategy is particularly effective in multi-sensor scenarios, where each object may be observed by several sensors simultaneously. The sampler naturally incorporates information from all sensors without significantly increasing computational cost. Since the GLMB filter does not require hypotheses to be strictly ordered, generating them through sampling is entirely sufficient and leads to substantial speed-ups.

Adaptive Birth based on Gaussian Likelihoods

Another important part of the MS-GLMB algorithm is the adaptive birth process, which introduces new object hypotheses when the current measurement set cannot be fully explained by existing tracks. Instead of relying on predefined birth regions, the adaptive birth mechanism analyses the incoming measurements from all sensors and identifies those that are unlikely to originate from known objects. These unexplained measurements serve as candidates for new object births. The adaptive birth method uses Gaussian likelihoods to propose reasonable initial states for these new tracks and then applies Gibbs sampling to select plausible birth hypotheses. As a result, the filter can react quickly to newly appearing objects while avoiding unnecessary false births.

The MS-GLMB filter is an online multi-sensor multi-object tracker with linear complexity in the total number of measurements across the sensors, and quadratic in the number of hypothesized tracks [85]. It is used in diverse, complex environments that benefiting from the MS-GLMB principled handling of object identities, birth and death, and measurement associations. It is successfully applied across a wide range of real-world domains, including 3D tracking of people [89] and vehicles [90]. For a comprehensive overview and comparison of labeled RFS MOT algorithms, refer to [88].

2.3.4 *Multi-Object Tracking Performance Metrics*

In multi-object tracking (MOT), performance metrics are designed to quantify how accurately a tracker estimates both the number and the states (e.g.,

positions, velocities) of objects over time. These metrics can generally be categorized into sequence-based metrics, such as Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP), which evaluate tracking performance over entire sequences, and set-based metrics, such as the Optimal SubPattern Assignment (OSPA) metric, which compare sets of estimated and ground-truth objects at each frame. More recently, the Higher Order Tracking Accuracy (HOTA) metric has been introduced, providing a unified framework that addresses both object detection quality and temporal association, which are critical for comprehensive MOT evaluation.

Higher Order Tracking Accuracy (HOTA)

is a metric that explicitly balances detection accuracy and association accuracy in multi-object tracking. It addresses limitations of MOTA/MOTP by separately measuring the quality of object detection (DetA) and association (AssA), then combining them into a single comprehensive score. HOTA ranges from 0 to 100 and decomposes into the submetrics: *Detection Accuracy (DetA)*, *Association Accuracy (AssA)*, and localization measures, allowing a more detailed analysis of tracking performance. Unlike traditional metrics, HOTA explicitly addresses both object detection quality and temporal association, which are critical for comprehensive MOT evaluation. The HOTA metric and its sub-metrics, along with their mathematical definitions and interpretations, are listed in Table 2.5. Readers who are interested in the mathematics, but who are unfamiliar with the terms true positive (TP), false negative (FN), false positive (FP), true positive associations (TPA), false negative assocications (FNA) and false positive associations (FPA), should refer to the explanations in the original HOTA paper [91]. This thesis uses the official HOTA evaluation code, TrackEval [1], which is used by many well-known MOT benchmarks. In the present work, it enables MOT results based on custom datasets to be quantified and interpreted.

Similarity Score S

Most tracking metrics require the definition of a similarity score between two detections, denoted as S. The similarity score should be chosen according to the detection representation, and it is constrained to lie between 0 and 1. When $S = 1$, the predicted detection and ground-truth detection perfectly align, and when $S = 0$, there is no overlap between the detections. For point representations, such as 3D object positions, a similarity score based on Euclidean distance, short S_{Euclid}, can be used. [92] Let $d(p_{pr}, p_{gt})$ denote the Euclidean distance between the predicted and ground-truth points, and let d_0 be the selected zero-distance threshold of the Euclidean distance where similarity score becomes zero. The similarity score is then defined as [91]:

$$S_{\text{Euclid}}(p_{pr}, p_{gt}) = \max\left[0, 1 - \frac{d(p_{pr}, p_{gt})}{d_0}\right] \tag{2.9}$$

1 https://github.com/JonathonLuiten/TrackEval

In this formulation, points are considered to have zero similarity when their distance is equal to or greater than d_0, and perfect similarity when they coincide.

Table 2.5: The HOTA metric and its submetrics

Metric	Mathematical Definition	Description
HOTA	$\text{HOTA} = \int_0^1 \text{HOTA}_\alpha \, d\alpha$, $\text{HOTA}_\alpha = \sqrt{\text{DetA}_\alpha \cdot \text{AssA}_\alpha}$	HOTA measures overall tracking accuracy by balancing detection and association performance. Integrates performance over localisation thresholds α.
DetA	$\text{DetA}_\alpha = \frac{\lvert TP \rvert}{\lvert TP \rvert + \lvert FN \rvert + \lvert FP \rvert}$	Detection Accuracy measures how well objects are detected regardless of association.
AssA	$\text{AssA}_\alpha = \frac{1}{\lvert TP \rvert} \sum_{c \in TP} A(c)$, $A(c) = \frac{\lvert TPA(c) \rvert}{\lvert TPA(c) \rvert + \lvert FNA(c) \rvert + \lvert FPA(c) \rvert}$	Association Accuracy measures how well predicted trajectories match ground-truth trajectories.
DetRe	$\text{DetRe}_\alpha = \frac{\lvert TP \rvert}{\lvert TP \rvert + \lvert FN \rvert}$	Detection Recall measures the fraction of correctly detected ground-truth objects. Its important for avoiding missed objects.
DetPr	$\text{DetPr}_\alpha = \frac{\lvert TP \rvert}{\lvert TP \rvert + \lvert FP \rvert}$	Detection Precision measures the fraction of predicted detections that are correct. Its important for avoiding false alarms.
AssRe	$\text{AssRe}_\alpha = \frac{1}{\lvert TP \rvert} \sum_{c \in TP} \frac{\lvert TPA(c) \rvert}{\lvert TPA(c) \rvert + \lvert FNA(c) \rvert}$	Association recall measures how well predicted trajectories cover the ground-truth trajectories. Its low if objects are split into multiple predicted tracks.
AssPr	$\text{AssPr}_\alpha = \frac{1}{\lvert TP \rvert} \sum_{c \in TP} \frac{\lvert TPA(c) \rvert}{\lvert TPA(c) \rvert + \lvert FPA(c) \rvert}$	Association Precision measures how well predicted trajectories maintain identity. Its low if a track spans multiple objects.
LocA	$\text{LocA} = \int_0^1 \frac{1}{\lvert TP_\alpha \rvert} \sum_{c \in TP_\alpha} S(c) \, d\alpha$	Localisation Accuracy evaluates the spatial accuracy of matched detections integrated over multiple localisation thresholds α.

Integrating over Localisation Thresholds

To account for localisation, the final HOTA score is computed as the integral of the HOTA score over the valid range of localisation thresholds α between 0 and 1. In practice, this integral is approximated by evaluating HOTA_α at a discrete set of thresholds, from 0.05 to 0.95 in steps of 0.05, and averaging the results. For each threshold α, the matching between ground-truth detections and predicted detections is performed independently. Formally, this can be written as:

$$\text{HOTA} = \int_0^1 \text{HOTA}_\alpha \, d\alpha \approx \frac{1}{19} \sum_{\alpha \in \{0.05, 0.1, \ldots, 0.95\}} \text{HOTA}_\alpha \tag{2.10}$$

This procedure ensures that the final HOTA score reflects both detection and association performance across multiple localisation thresholds.

3

CONCEPT DEVELOPMENT

This chapter presents the concept development of audio–visual multi-speaker tracking system. Section 3.1 introduces the fundamental problem of multi-modal detection fusion and outlines a principled formulation and solution. Section 3.2 proposes a modular audio–visual software architecture design which supports audio-visual simulation, detection and sensor fusion to achieve multi-speaker tracking. Finally, Section 3.3 formulates hypotheses regarding the expected tracking performance of speakers in indoor environments. Hypothesis consider the modalities audio-only, video-only and the combined audio-visual modality.

3.1 PROBLEM FORMULATION AND SOLUTION

Tracking multiple sources using audio and video data poses a significant challenge. A central problem is that measurements from different modalities like sound and image cannot be clearly assigned. It is unclear which measurement must be assigned to which source, as both signal types usually exist in different "spaces". While video data, for example, captures two-dimensional or three-dimensional coordinates in the camera's field of view, audio data provides information about sound sources, which is captured in the form of time-difference-of-arrival or directional differences between microphones. The target system to be realized, consisting of several microphones and a extreme wide-angle fisheye camera, should capture both the audio and visual modalities of speakers in a room. The fusion of these modalities is crucial to enable precise and robust tracking of multiple people. The underlying problem can be described as a *Multimodal Space-Time Permutation Problem*, which requires an optimal mapping of detections across time and modalities. [77] MOT algorithms from the RFS filter family are one option to solve this problem. These state-of-the-art trackers combine multimodal detections, like speaker positions, to address several key challenges:

1. **Data Association** is the identification and correct assignment of detections from both modalities to the respective sources.

2. **Sensor Fusion** combines information from different sensor types to reduce uncertainties and provide a more robust estimate of speaker positions and movements.

3. **Tracking** is used to predict speaker trajectories over time, including accounting for new or disappearing sources and based on noisy and missing sensor detections.

3.2 AUDIO-VISUAL SOFTWARE ARCHITECTURE DESIGN

A modulare audio-visual software architecture with standardized interfaces between its software modules is designed and proposed in Figure 3.1. Each software module contains a state-of-the-art simulation, detection or multi-object tracking algorithm. The modularity and the standardized interfaces of the design allows the software architecture to work with any current or future state-of-the-art algorithm, which makes it future proof. The five software modules of the proposed audio–visual software architecture, along with their interfaces, are outlined in the following. A dedicated git repository is created for each module to make its source code available; these repositories are listed in Table 3.1.

(1) The Audio Simulation Software Module simulates the audio modality. The Geometrical Acoustics–based algorithm *gpuRIR* is used to model how sounds from multiple moving sources propagate and how multiple receivers capture these sounds in indoor environments. The inputs are the static scene description in JavaScript Object Notation (JSON) format, `config.json`, containing the room and sensor configuration, and the time-dependent ground-truth sound-source positions in the streamed JSON Lines (JSONL) file `groundtruth_sources.jsonl`. Both inputs are provided by the Video Simulation Software Module described later. The outputs consist of one or more multi-channel receiver signals in the lossless Waveform Audio File (WAV) format, stored as `multichannel_audio_<ts>.wav`.

(2) The Audio Detector Software Module detects sound-source positions in the multi-channel audio signals. The Positional Sound Source Localization algorithm `3D-SSL` is employed to estimate the 3D Cartesian coordinates of

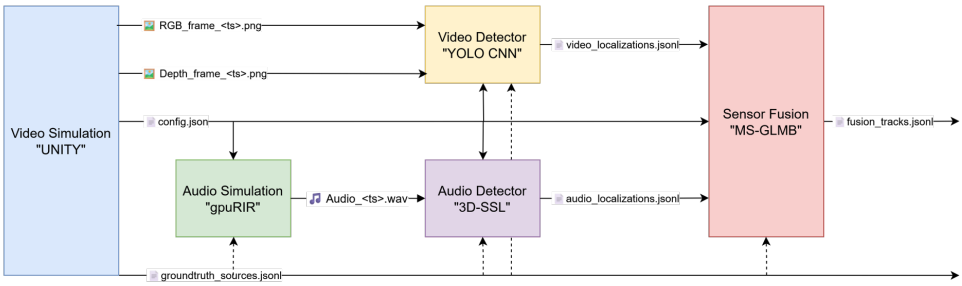


Figure 3.1: The proposed modular audio-visual software architecture design.

Software Modules	Git Repository URL
1. Unity Simulation	`https://code.fbi.h-da.de/est/est-workgroup/visualsimulationunity`
2. Audio Simulation	`https://code.fbi.h-da.de/est/est-workgroup/gpuRIR`
3. Video Detector	`https://code.fbi.h-da.de/est/est-workgroup/cnnvideodetektor`
4. Audio Detector	`https://code.fbi.h-da.de/est/est-workgroup/ssl4ips`
5. Sensor Fusion	`https://code.fbi.h-da.de/est/est-workgroup/audiovisualsensorfusion`
Evaluation	`https://code.fbi.h-da.de/est/est-workgroup/trackeval`

Table 3.1: Git repositories for the main components of the software architecture and for evaluation.

moving sound sources. The inputs are the scene description `config.json` and the multi-channel receiver signals `multichannel_audio_<ts>.wav` produced by the Audio Simulation Software Module. The output is a stream of localized time-dependent 3D sound-source positions written to `audio_localizations.jsonl`.

(3) The Video Simulation Software Module simulates the video modality. The 3D graphics engine `Unity` is used to simulate an indoor environment with multiple moving speakers and to model how the scene is captured by a fisheye RGB and Depth camera. This module does not require external input; scenarios are configured directly within Unity. The outputs are streamed RGB and Depth camera frames in the lossless Portable Network Graphics (PNG) format, `RGB_frame_<ts>.png` and `Depth_frame_<ts>.png`. In addition, it provides the static scene description `config.json` and the ground-truth positions of persons in `groundtruth_sources.jsonl`, both required by other modules.

(4) The Video Detector Software Module detects persons in the camera frames. A Convolutional Neural Network of the `YOLO` family is used to detect persons and localize them in 3D Cartesian coordinates. The inputs are the scene description `config.json` and the RGB and Depth frames supplied by the Video Simulation Software Module. The output is a stream of localized time-dependent 3D person positions written to `video_localizations.jsonl`.

(5) The Sensor Fusion Software Module fuses detection features from different modalities in accordance with the Tracking-by-Detection paradigm. The MOT algorithm `MS-GLMB` is used to associate detections across time and to predict object trajectories while handling uncertainty, object birth, and object death. The required inputs are the audio detections `audio_localizations.jsonl` and the video detections `video_localizations.jsonl`. The ground-truth positions `groundtruth_sources.jsonl` may optionally be provided for evaluation. The output consists of predicted tracks (linked time-dependent positions) of persons in the scene, streamed to `fusion_tracks.jsonl`. Although the Sensor Fusion Software Module internally predicts trajectories (linked time-dependent positions and velocities), this thesis, like many MOT benchmarks, evaluates only tracks.

The evaluation of the predicted tracks is performed using the TrackEval HOTA reference implementation, as previously described in Subsection 2.3.4. It compares the output tracks of the Sensor Fusion Software Module, `fusion_tracks.jsonl`, with the ground-truth tracks of persons in the scene, `groundtruth_sources.jsonl`, to quantify detection, association, and localisation errors.

To systematically organize and store the data for each experiment, a directory structure is defined, as illustrated in Figure 3.2. This structure includes the scene description, ground-truth annotations, raw sensor data, detection results, and audio–video recordings. Examples of each of the JSONL files are provided in the appendix. The format of the scene configuration file is shown in Listing .1, the format of the audio and video detection files is

shown in Listing .4, and the format of the ground-truth and sensor fusion tracking files is shown in Listing .5.

```
experiment_001/
├── config.json
├── groundtruth_sources.jsonl
├── audio/
│   └── wav/
│       └── multichannel_audio_<timestamp>.wav
├── video/
│   ├── rgb/
│   │   └── RGB_frame_<timestamp>.png
│   └── depth/
│       └── Depth_frame_<timestamp>.png
├── localization/
│   ├── audio_localizations.jsonl
│   └── video_localizations.jsonl
└── tracking/
    ├── audio_tracking.jsonl
    ├── video_tracking.jsonl
    └── audio_video_tracking.jsonl
```

Figure 3.2: Directory structure for a single audio-visual simulation experiment.

3.3 RESEARCH HYPOTHESES

The following research hypotheses describe the expected outcomes of the multi-object tracking evaluation for the three modality configurations investigated in this thesis: audio-only, video-only, and combined audio–video fusion. These hypotheses reflect the theoretical advantages and limitations of each modality and serve as a scientific baseline against which the experimental results presented later in this thesis can be compared.

H1: AUDIO-ONLY MOT PERFORMANCE. It is expected that the audio-only modality will achieve comparatively low MOT performance. My previous work in [24] has shown that the PSSL algorithm 3D-SSL can localize sound sources, but the localization accuracy and especially the number of detections decreases significantly with increasing room reverberation. Consequently, it is hypothesised that audio-only tracking will show a high number of missed track detections. This will result in split tracks and incorrect associations, leading to low HOTA scores.

H2: VIDEO-ONLY MOT PERFORMANCE. The video-only modality is expected to outperform the audio-only modality. My previous work in [67] has shown the reliability and high accuracy of the YOLO visual detector. Provided that persons remain at least partially visible, object localisation

and association are typically reliable. However, performance is expected to degrade under occlusions or high lens distortion. Overall, video-only tracking is hypothesised to achieve high HOTA scores, but with noticeable drops in scenarios involving significant occlusion.

H3: AUDIO–VIDEO SENSOR FUSION MOT PERFORMANCE. The combined modality is expected to achieve the highest MOT performance. Sensor fusion using the MS-GLMB algorithm can exploit complementary strengths: video provides spatial precision, while audio offers robustness in visually occluded or low-visibility situations. It is therefore hypothesised that the fused system will reduce detection gaps, improve association consistency, and achieve more stable trajectories over time. Consequently, audio–video fusion is expected to outperform both unimodal approaches and achieve the highest HOTA scores across all evaluated scenarios.

4

EXPERIMENTAL SETUP

In this chapter the role and processing steps of each of the five software modules: (1) Audio Simulation Software Module, (2) Audio Detector Software Module, (3) Video Simulation Software Module, (4) Video Detector Software Module, (5) Sensor Fusion Software Module, in the proposed audio-visual software architecture from Section 3.2 is explained in detail one by one.

4.1 ROOM ACOUSTICS SIMULATION

The Audio Simulation Software Module combines the steps of (1) Room Acoustic Simulation, (2) Room Impulse Response (RIR) Generation, and (3) Audio Rendering, to generate synthetic multi-channel microphone recordings for multiple moving speakers within the virtual room environment. It utilizes the scene description and time-dependent 3D source positions exported by the *Unity* engine, as described in Section 4.3. The resulting synthetic microphone signals with 32-bit depth and a 44.1kHz represent physically plausible sound propagation in the same environment as the visual simulation, enabling synchronized and reproducible audio-visual experiments.

Microphone Simulation

The open-source Python library *gpuRIR* [5] is utilised as the virtual acoustics simulation tool. It implements the Image Source Method (ISM) using the parallel programming language CUDA to achieve real-time performance using available Graphical Processing Unit (GPU) resources. *gpuRIR* models sound reflections within rectangular rooms by recursively mirroring sound sources along the room boundaries. This approach allows efficient computation of high-order reflections and reverberation effects for multiple sources and receivers. The simulation assumes a static room geometry, where both the sound source and receiver positions are specified in Cartesian coordinates relative to the global room reference frame.

Eight static microphones are defined in the scene, corresponding to the physical microphone array used in the laboratory. Their 3D positions are read from the configuration file `config.json`. The simulated speakers correspond to moving sources, whose 3D trajectories are continuously read from the `groundtruth_sources.jsonl` file generated by the visual simulation module. Each trajectory point contains a timestamp and the 3D position of a speaker in the simulated room.

Room Impulse Response Generation

For each discrete source position, the acoustic propagation between the speaker and each microphone is modeled by a Room Impulse Response (RIR). The RIR represents the temporal evolution of sound energy arriving at a receiver as a combination of the direct sound, early reflections, and late reverberation. The RIR is determined by the room geometry, the wall absorption coefficients, and the reverberation time T_{60}.

The reverberation time is a key parameter defining how long it takes for the sound energy to decay by 60 dB. It is estimated using the Sabine equation:

$$T_{60} = 0.161 \frac{V}{A} \tag{4.1}$$

where V is the room volume in cubic meters and A is the equivalent total absorption area in square meters. The latter is computed as the weighted sum of the surface areas S_i of all six room walls and their frequency-independent absorption coefficients α_i:

$$A = \sum_{i=1}^{6} \alpha_i S_i \tag{4.2}$$

The *gpuRIR* library internally estimates the reflection coefficients β_i for each wall from the specified reverberation time and applies them to compute the time-domain impulse responses. In this work a common reverberation time of $T_{60} = 0.35$ s for office spaces according to DIN18041 (Acoustic Quality in Rooms) is used [93].

Once the RIRs are computed for all source–receiver pairs, the simulated microphone signals are obtained by convolving the dry (anechoic) source signals with their corresponding impulse responses. Mathematically, this can be expressed as:

$$y_m(t) = \sum_{n=1}^{N_s} \big(s_n(t) * h_{n,m}(t)\big), \tag{4.3}$$

where $y_m(t)$ is the signal received at the m-th microphone, $s_n(t)$ denotes the n-th source signal, $h_{n,m}(t)$ is the RIR between source n and microphone m, and $*$ represents time-domain convolution. The summation accounts for all sound sources simultaneously active in the room.

Audio Rendering of Moving Speakers

In the case of a static speaker, each source-receiver pair is represented by a single RIR that remains constant over time. However, to account for moving speakers along continuous trajectories, *gpuRIR* divides the source signal into short segments and convolves each segment with the RIR corresponding to the source position at that time . The resulting convolved segments are then concatenated to form a continuous microphone signal as shown in Figure 4.1, where they x-axis represents time in samples. The example shows the result of convolving the RIR with a dry male speaker saying, “Can you keep

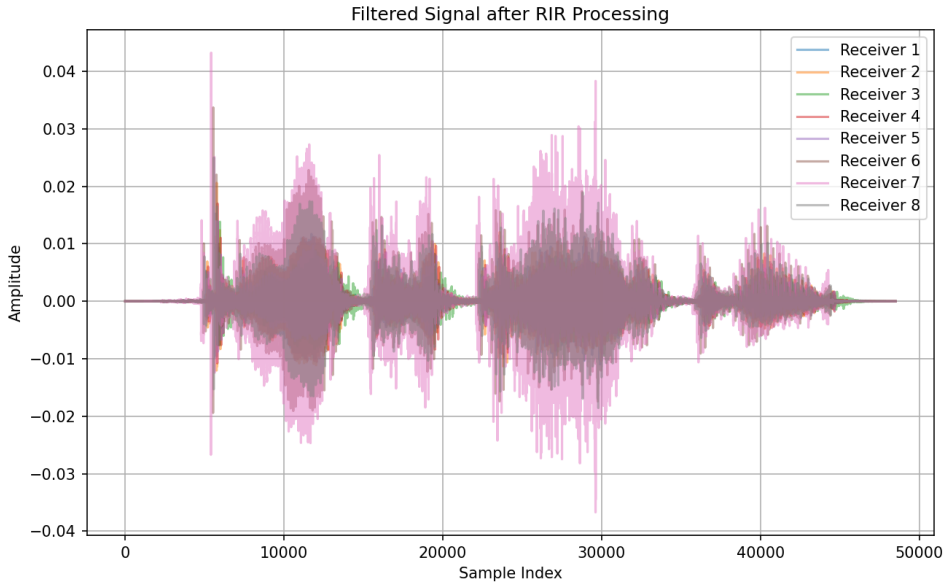


Figure 4.1: Synthetic microphone signals of a single moving male speaker generated with gpuRIR.

a secret?". The resulting signals represents the sound perceived by multiple receivers, e.g. a distributed microphone array. When approximating the motion of a sound source as a sequence of static, discrete positions, the time and frequency resolution of the receiver signal depends heavily on the number of positions. As part of this work, linear interpolation between a sequence of 3D discrete positions is proposed. The interpolated source trajectories maximise the time and frequency resolution of the receiver signals. This is crucial for accurately localising sound sources later on. By approximating the motion of sound sources as a sequence of interpolated discrete positions, this method can efficiently reproduce the time-varying acoustic characteristics associated with moving sound sources. This work uses an interpolation time step size of $t_{interpolate} = 0.01$ s, which results in speaker movements of ≤ 1 cm assuming a maximum speaker velocity of $v = 1$ m/s.

Limitations of the Room Acoustics Simulation

To identify the frequency range for a realistic Room Acoustic simulation using the ISM method, the Schroeder frequency is typically chosen in recent research work [94] and [95]. This presents the transition frequency between low frequencies, which are described by wave theory, and high frequencies, which follow the ray-based concept of Geometrical Acoustics. The Schroeder Frequency is defined in [96] as

$$f_S = 2000\sqrt{\frac{T_{60}}{V}}\ \text{Hz} \tag{4.4}$$

and is a function of reverberation time, which makes it frequency-dependent. A higher reverberation time results in a higher cross-over frequency.

In my research work preceding this thesis [24], reverberation time measurements were performed the laboratory room with a volume of approximately 65.15 m^3. The dominant frequencies the male The lowest frequency used by the Audio Detector is 200 Hz. At this frequency the minimum RT60 of approximately 0.80 s, results in a Schroeder frequency of $f_S = 2000\sqrt{\frac{0.80}{65.15}}$ Hz $= 221.63$ Hz. Therefore, it can be concluded that the frequency range simulated via ISM follows the concept of Geometrical Acoustics, making it accurate and realistic. If lower frequencies have to be simulated, hybrid simulation methods and tools that simulate wave phenomena [94] [95] are required, as discussed in Subsection 2.1.3.

4.2 AUDIO DETECTOR

The Audio Detector Software Module combines two main steps: (1) Sound Event Detection and (2) Positional Sound Source Localization, to localize moving speakers in the synthetic multi-channel microphone recordings introduced in Section 4.1 to obtain their positions in 3D world coordinates of the virtual room environment. The selection of the PSSL algorithm *3D-SSL* is motivated by the low computation complexity of it's closed-form analytical solution and its high localization precision even in highly reverberant indoor environments. The PSSL algorithm is implemented in my research work [24] preceding this thesis and using the MATLAB programming language. The paper also examines tuning of parameter used in the following,

Sound Event Detection

For localization, selecting a reference microphone from the array is required for TDOA computation and is performed first. This is done by computing the Short-Time Energy (STE) for each sample of the signal. The reference is chosen as the microphone with the highest short-time energy, under the assumption that it is closest to the sound source and thus captures the cleanest signal. Once a predefined emprical STE threshold is exceeded in the reference microphone signal, a sound event is assumed to be detected. A symmetric window of for of 42 ms is spanned from there for each of the time-synchronized audio channels to extract multi-channel signal windows for further processing. Figure 4.2 visualises the sound event detection processing steps. It starts with the synthetic reference microphone signal at the top. The STE computation and the STE threshold are shown in the middle. The final sound events are visualised at the bottom, with each one starting green and ending red.

Positional Sound Source Localization of Moving Speakers

The PSSL algorithm *3D-SSL* uses the multi-channel signal windows to obtain the 3D position of detected sound events. To mitigates the negative influence of room reverberation on the localization accuracy the algorithm uses the well-known *β-GCC-PHAT* frequency weighting. The weighting factor β is empirically set to 0.7. The speed of sound is 343.0 m/s. Knowing the synthetic microphone position, the resulting 3D world coordinates of a speaker can be solved in a closed-form. During runtime, the Audio Detector Software Module writes the detected 3D speaker positions in the scene to the JSONL file `audio_localizations.jsonl`. Figure 4.3 visualises the ground-truth and estimated positions of a single male speaker repeatedly saying, "Can you keep a secret?" while walking in a straight line. It demonstrates the high accuracy of 3D-SSL in localising a moving speaker at very low reverberation. Simulating the same scene with higher room reverberation times T_{60} decreases the number of PSSL detections and also the localization accu-

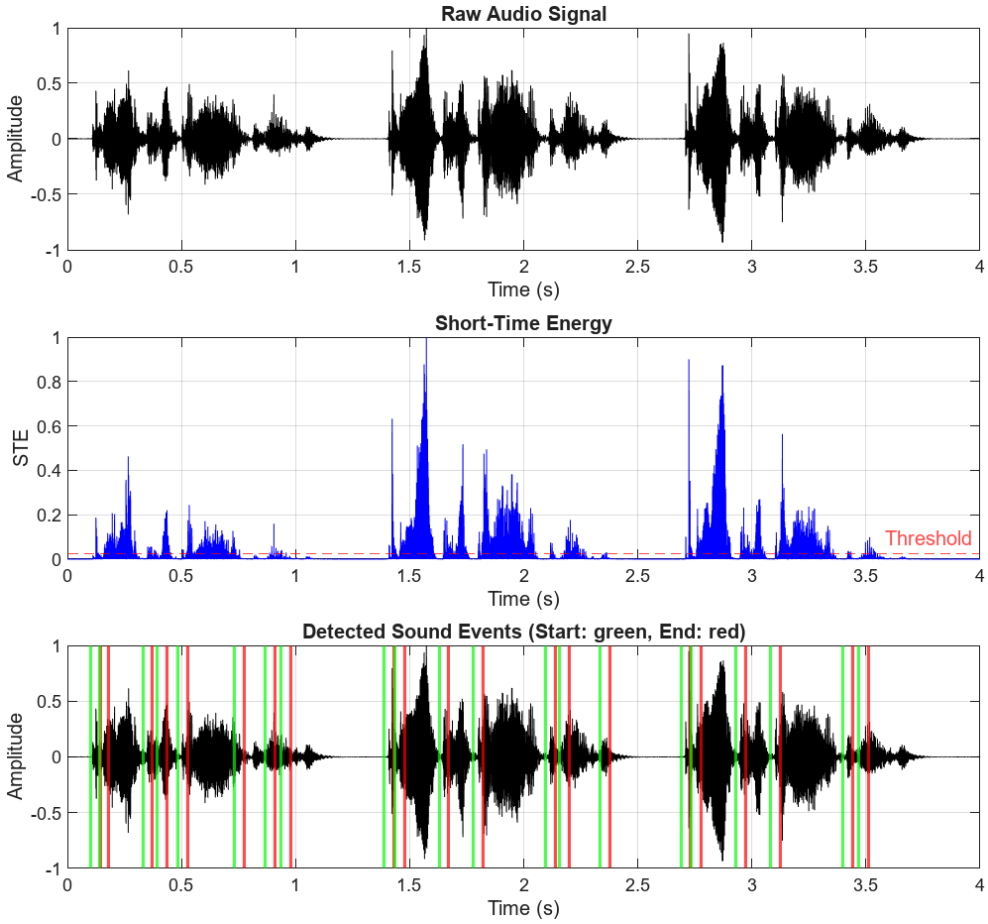


Figure 4.2: Processing steps of sound event detection: raw audio signal (top), short-time energy (center) and detected events (bottom).

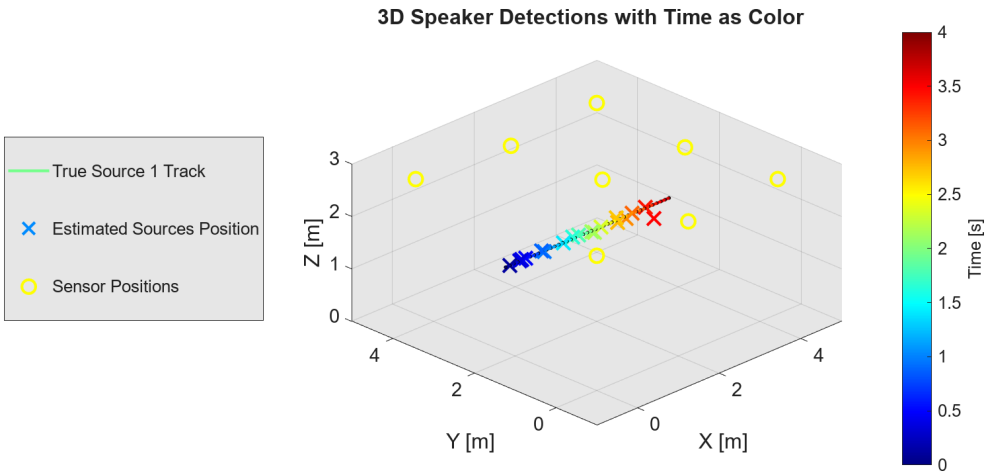


Figure 4.3: 3D-SSL detections of a moving speaker in a very low reverberation indoor environment (T60 = 0.1 s).

racy. These results correspond with the evaluation results of 3D-SSL in my previous paper [24].

4.3 VISUAL SCENE SIMULATION

The Video Simulation Software Module combines the three steps (1) Camera Calibration, (2) Camera Simulation, and (3) Scene Simulation to render realistic synthetic fisheye camera images and provide a scene discription that is used by all other software modules of the proposed audio-visual software architecture.

Camera Calibration

The *OpenCV* library was selected as the fisheye camera calibration tool based on the comparative overview presented in [53]. The open-source Python library implements the KB-8 model and supports standard checkerboards as calibration targets. It also provides visualization tools such as corner detection overlays to verify calibration quality and computes the root mean square (RMS) projection error in pixels to quantify the accuracy of the estimated intrinsic parameters.

A dataset of calibration images was created by connecting the 170 degrees Diagonal angle of view (DAOV) fisheye camera *ELP-USBFHD06H-L170* via USB to a host PC and recording a 5:39 min video of a planar checkerboard calibration target from different angles and distances by moving the calibration target. The open-source software *OBS Studio* is used for creating the recording. The main specifications of the fisheye camera are summarized in Table 4.1. Further details can be found on the camera manufacturer's website[1] and the imager datasheet[2]. A large planar calibration checkerboard made of metal is used with a generated and printed pattern of 6×8 patches created with calib.io[3]. Each patch is squared and has a edge size of 95 mm. A Python script, `calibration.py`, is implemented

Table 4.1: Main specifications of the ELP-USBFHD06H-L170 fisheye camera.

Parameter	**Specification**
Lens size	1/2.9 inch
Lens DAOV	170 degrees
Sensor	Sony IMX322
Sensor coverage by lens	Full-frame format
Pixel size	2.8 μm (H) $\times$ 2.8 μm (V)
Recording pixels	1920 (H) $\times$ 1080 (V) $\approx$ 2.07 Mpx
Maxiumum resolution	1920 $\times$ 1080 (1080p)
Frame rate	30 fps @ 1920 $\times$ 1080
Interface	USB 2.0

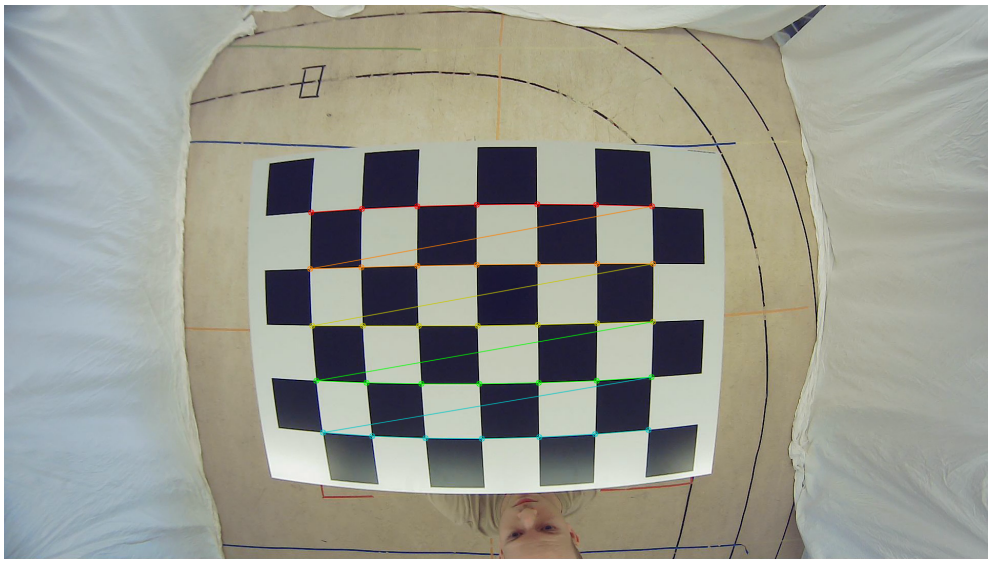
1 http://www.elpcctv.com/elp-h264-low-illumination-sony-imx323-1080p-170-degree-wide-angle-usb-camera-module-p-332.html
2 https://dashcamtalk.com/cams/lk-7950-wd/Sony_IMX322.pdf
3 https://calib.io/pages/camera-calibration-pattern-generator

to automate the calibration workflow. Initially, two frames per second of the recorded `.mp4` video are extracted as `.png` image, thus increasing the diversity of the calibration images. Next, checkerboard corner detection is performed, and poorly detected frames are discarded to prevent calibration failures. A first OpenCV fisheye calibration is performed using all 407 calibration images and now an empiric RMS projection threshold of 0.75 px is used to filter out images suffering from motion blur, out of focus, or overexposure. Finally, the resulting dataset of 383 valid calibration images are used for OpenCV fisheye calibration to estimate the eight intrinsic parameters $i = [f_x, f_y, c_x, c_y, k_1, k_2, k_3, k_4]$ and compute the mean RMS projection error. The obtained intrinsic parameter set of the KB-8 model is: $i = [862.27px, 862.70px, 1049.47px, 552.76px, -0.0269, -0.0038, 0.0016, -0.0006]$ and calibration achieved an mean RMS projection error of $\epsilon_{\text{RMS}} = 0.29$ px for the image resolution of 1920x1080 px. A significantly smaller checkerboard with 8×11 squared patches with an edge size of only 25 mm was tested also but resulted in more than three times higher mean RMS projection error of $\epsilon_{\text{RMS}} = 1.07$ px. An RMS projection error for KB-8 of approximately 0.3 px can be considered a very good calibration accuracy for fisheye lenses when comparing it to the calibration results of 164° DAOV fisheye camera in [53] achieved by using different calibrations tools like Babel, Basalt, Camodocal, Kalibr and OpenCV.

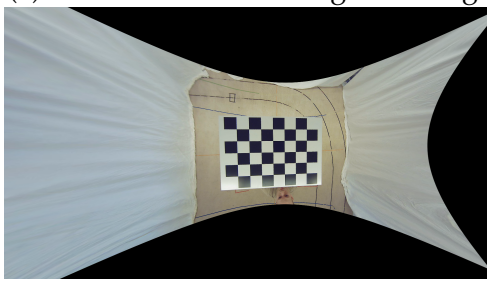
A second script, `undistort.py`, is developed to apply distortion correction using the estimated camera intrinsics for visual verification. Figure 4.4 left shows one of the calibration images with detected checkerboard corners, while Figure 4.4 right shows the same image after distortion correction. The rectified image demonstrates straightened checkerboard lines confirming successful correction. However, a noticeable cropping of the image occurs because the projection from the fisheye to the pinhole camera model introduces non-rectangular pixel distributions. To maintain the image dimensions of the input image also for the output image and to avoid black regions, OpenCV automatically crops the valid image area during undistortion.

Camera Simulation

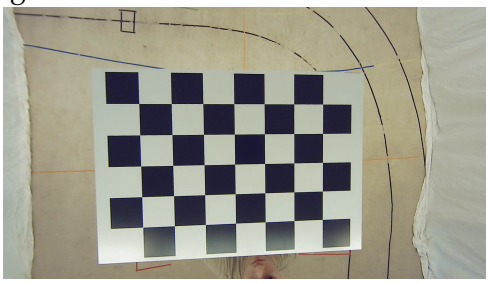
The 3D computer graphics engine *Unity* is used to generate synthetic fisheye camera images. The intrinsic and extrinsic parameters previously obtained for the real camera *ELP-USBFHD06H-L170* are used to simulate the fisheye camera projection as realistically as possible. Two custom fisheye shaders, `KannalaBrandt_KB8_EquidistantProjection.shader` and `BrownConradyProjection.shader`, have been implemented using *Unity* fragment shaders, which allow efficient pixel-wise post-processing. The first shader implements an equidistant projection using KB-8 intrinsic parameters. The second shader implements the Brown-Conrady projection using Brown-Conrady intrinsic parameters [53] [60]. Excerpts from both forward projection implementations can be found in the appendix as Listing .2 and



(a) Distorted camera image with highlighted checkboard corner detections



(b) Undistorted camera image



(c) Undistorted and cropped camera image

Figure 4.4: Camera Calibration and Distortion Correction using the Kannala-Brandt model in OpenCV results in straight lines in the camera image.

Listing .3. Each shader is applied to one *Unity Material*, which changes the visual appearance of gameobjects such as virtual cameras. A custom C# script, `PhysicalCameraSimulator.cs`, is created and attached to a virtual camera in the scene. This script uses the GPU-accelerated function call `Graphics.Blit()` to create fisheye post-processing effects for the virtual camera using the material that includes the custom fisheye shader. This enables the synthetic fisheye camera images to be rendered in real time during simulation. Figure 4.5 shows a visual comparison of images from the real camera and the rendered images from the virtual cameras using different projections. The columns show camera images that are (i) distorted, (ii) undistorted and (iii) undistorted and cropped. The first row shows the real camera images in the laboratory for general reference. A similar scene with a different calibration target was created in *Unity* for the virtual cameras. Although there was no Unity asset available for the exact same calibration target, it allows a general visual comparison. The perspective projection of the scene in the second row has no distortion and represents the ground-truth after undistortion. The Kannala-Brandt equidistant projection in row three uses the same

intrinsic parameters as a real camera. As can be seen, the camera centre in (i) is not centred, resulting in a horizontally and vertically shifted image. This is expected, given that the principal point coordinates of the camera intrinsics, $\frac{c_x}{w} = \frac{1049.47px}{1920px} = 0.54660$ and $\frac{c_y}{h} = \frac{552.76px}{1080px} = 0.51182$, are not perfectly centred in the image plane. Furthermore, it can be seen in (ii) and (iii) that the undistorted image still shows distortion, resulting in bent lines that are not straight. This issue must be caused by an errors in the intrinsic parameters or a bug in the fisheye shader implementation of the Kannala–Brandt KB-8 forward projection, since the same backward projection is used as in the real camera. The exact cause could not be resolved within the timeframe of this work.

The Brown-Conrady projection in row four uses the same focal length as the real camera, but with a centred principal point of $c_x = 960px$ and $c_y = 540px$, and with approximated coefficients $k1 = -0.2, k2 = 0, k3 = 0, k4 = 0$. As can be seen, the camera image in (i) is perfectly centred. The undistortion result of the backward projection from (i) to (iii) is much better than that of the Kannala-Brandt projection. Visually, it matches the perspective projection in the second row perfectly. While the Brown-Conrady forward projection is only an approximation of the real camera, its backward projection is very precise. Motivated by these results, the Brown-Conrady forward projection is used for the virtual camera for the remainder of this work, and the backward projection is used to map detections very precisely from image space to world space.

The camera images are provided to other applications on the same machine or to other machines in the local area network using the written C# script `TcpDataSender.cs`. It implements the client of the server-client

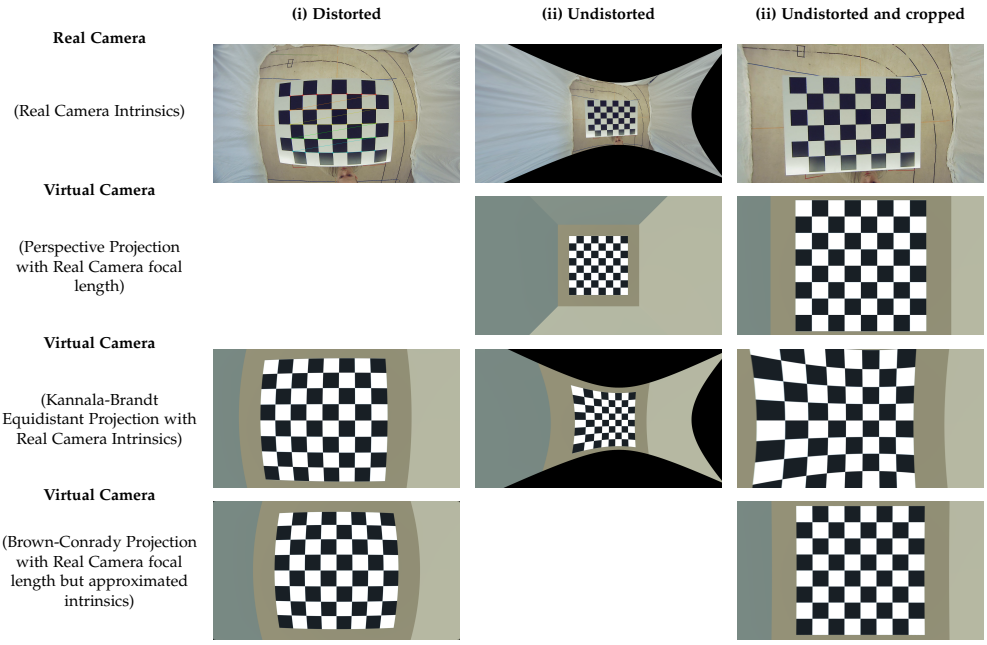


Figure 4.5: Visual comparison of real and virtual camera images after (i) distortion, (ii) undistortion and (iii) undistortion and cropping.

based, connection-oriented *Transmission Control Protocol and Internet Protocol (TCP/IP)*. At simulation startup it creates a TCP client and establishes a connection to a server with IP adress 127.0.0.1 and port 25001.

Scene Simulation

A virtual scene is created in *Unity* by designing the same room dimensions of $4.66 \times 4.66 \times 3.00$ m as in the laboratory and approximating curtain wall colors and floor colors manually. Eight virtual microphone gameobject are created and positioned at the exact 3D position in the real scene. Sound propagation is not simulated in the Video Simulation Software Module but the 3D positions of the microphones are used in the later Audio Simulation Software Module. Two virtual cameras are created and placed in the scene with the same 3D position and top-view orientation as in the laboratory. Three photorealistic and freely available 3D models were obtained from the commercial platform *RenderPeople*[4]. The two female models and one male model include idle and walking animations. Within the scope of this work, their position is controlled by a simple *Unity* C# script. It translates the speakers along the x-axis through the simulated scene according to the constant velocity equation $x = x + v \cdot t$, using a fixed velocity of $v = 1.0\,\mathrm{m/s}$. By combining walking animations and translations, realistic movement of all speakers is achieved. The simulated indoor environment in *Unity* is shown in Figure 4.6 and includes the audio sensors visualized as cubes, three speakers and two tables with chairs. As the central video sensor position overlaps with the central audio sensor position, it is not visualised. The C# script `RGBD_Capture_And_Send.cs` acts as main script during simulation runtime

4 https://renderpeople.com/free-3d-people/

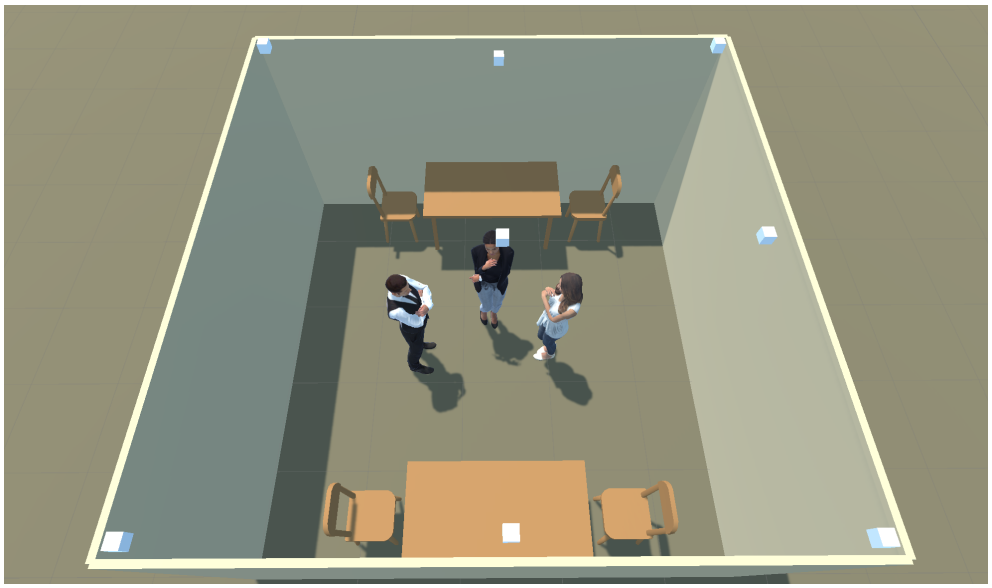


Figure 4.6: The indoor environment with three speakers simulated in Unity.

to sent any data from *Unity* to other applications. The script first exports positions and properties of the synthetic audio and video sensors to provide a scene discription that is used by all other software modules of the proposed audio-visual software architecture. The scene description in JSON format is named `config.json`. The script then predefines the simulation update period T of 0.1 s and the function call `FixedUpdate()` to achieve fixed simulation update frequency of 10 Hz. This implementation is essential to achieve reproducable experiments independent from CPU and GPU timings. In each simulation update the following steps are performed:

1. The time since simulation start in seconds is received as double precision timestamp directly the simulation.
2. The current synthetic fisheye camera image is rendered and sent to the connected TCP Server
3. The current 3D Speaker positions in the scene are written to the JSONL file `groundtruth_sources.jsonl`.

The combination of (1) Camera Calibration, (2) Camera Simulation, and (3) Scene Simulation of this Section achieve the generation of synthetic fisheye camera images during simulation time. A comparision of the synthetic fisheye camera image rendered in *Unity* and a real scene captured in the laboratory is visualized in Figure 4.7.

Limitations of the Visual Scene Simulation

The visual scene simulation assumes idealized optical and environmental conditions. The fisheye projection implemented in *Unity* approximates the real camera model but neglects effects such as sensor noise, chromatic aberration, and exposure dynamics. Room geometry and surface materials are simplified, and global illumination or dynamic lighting variations are not simulated. Speaker motion is modeled as constant velocity and may not perfectly reflect real physical dynamics. Nevertheless, the simplified synthetic images produced are sufficiently realistic and geometrically consistent to allow reproducible visual experiments.

Synthetic fisheye camera image of the simulated scene

Real fisheye camera image of the laboratory

Figure 4.7: The synthetic fisheye camera image of the simulated scene (top) and the real camera fisheye camera image of the laboratory (bottom).

4.4 VIDEO DETECTOR

The Video Detector Software Module combines two main steps: (1) Object Detection and (2) 2D-to-3D Back-Projection, to detect humans in the synthetic fisheye camera images introduced in Section 4.3, and map their position to 3D world coordinates.

Object Detection

A CNN from the *YOLO* family is selected based on its superior performance, as explained in Subsection 2.2.4. The neural network model was trained in research work preceding this thesis [67] using a combination of standard images from the COCO dataset and a custom fisheye image dataset collected using the same fisheye camera *ELP-USBFHD06H-L170*. The YOLO model performs object detection of humans in the 2D image plane by outputting bounding boxes, confidence scores, and class probabilities. The python script `video_detector.py` is written that starts with loading the locally saved *YOLO* object detection model `yolov5nu.pt` using the *ultralytics* library. Once finished, it reads the scene configuration `config.json` to receive the camera intrinsic parameters $i = [f_x, f_y, c_x, c_y, k_1, k_2, k_3, k_4]$ and the camera extrinsic parameters to map points from the camera coordinate system to the world coordinate system. Next a TCP Server is created using the *socket* library to connect to the running 3D computer graphics engine *Unity* to start receiving synthetic RGB and Depth fisheye camera images. Per frame image inference is performed using the *YOLO* model to get rectangular bounding box object detections in the image coordinates. Postfiltering is used to consider only object detections of class person and with confidence values equal or higher than a predefined threshold. As shown in Figure 4.8 and 4.9 the center of each rectangular bounding box is used to represent the detected person as a point detection in image coordinates.

2D-to-3D Back-Projection

To obtain the 3D position of detected persons, the 2D bounding boxes are mapped to world coordinates using the known intrinsic and extrinsic parameters of the camera. Specifically, the center point of each bounding box is back-projected into the 3D world coordinate system. This process uses the inverse camera projection along with depth information. The resulting 3D coordinates allow for spatial localization of each detected person within the simulated room environment. The same python script `video_detector.py` performs computational efficient undistortion of the point detections by using a precomputed Look-Up Table of the inverse KB-8 camera model using the previously obtained intrinsics parameters $i = [f_x, f_y, c_x, c_y, k_1, k_2, k_3, k_4]$. The resulting undistorted point detections in 2D image coordinates are now backprojected to rays in 3D world-coordinates using the camera extrinsic parameters $E = [R, t]$, where $R \in \mathbb{R}^{3\times3}$ is the rotation matrix and $t \in \mathbb{R}^3$

Figure 4.8: Bounding box detections and point abstractions in the synthetic RGB fisheye camera image using the YOLO model.

Figure 4.9: Bounding box detections and point abstractions in the synthetic Depth fisheye camera image using the YOLO model.

is the translation vector. Due to the fact that rays in 3D world coordinates can not specify a point position additonal depth information needs to be incorporated. This is done by extracting the pixel value of the center of each rectangular bounding box but from the synthetic Depth fisheye camera image instead of the RGB camera. As the synthetic Depth fisheye camera shares the same intrinsics as the RGB fisheye camera, backward projection of depth image is not needed. By geometrically intersecting each rays in 3D world coordinates with its corresponding depth value, each person detection can be localized in 3D world coordinates. Figure 4.10 illustrates the same speaker point representation previously shown in Figure 4.8. Here the point as well

Figure 4.10: Undistorted 3D video detections of speakers in the indoor environment.

as the image are undistorted. During the simulation, the Video Detector Software Module writes the detected 3D speaker positions in the scene to the JSONL file `video_localizations.jsonl`.

4.5 AUDIO-VISUAL SENSOR FUSION

The Audio-Visual Sensor Fusion Software Module performs multi-object tracking on positions previously determined by the Audio Detector Software Module and the Video Detector Software Module. The module combines detections from both modalities to improve robustness and accuracy in estimating tracks of multiple speakers.

Multi-object Tracking of Moving Speakers

The MS-GLMB algorithm [85] was selected as the multi-object tracking method which implements the TBD paradigm, while handling uncertainties in detections. Its low computational complexity allows online processing of detection stream. The Python script `audio_visual_sensor_fusion.py` reads streams of audio localizations and video localizations from the respective JSONL files `audio_localizations.jsonl` and `video_localizations.jsonl`. The implementation of MS-GLMB[5] follows the methods presented in [86] [85] and [87]. It employs a Bayesian framework with a constant velocity model, performing both prediction and update steps while assuming Gaussian likelihoods. To further increase robustness, audio and video detections are pre-filtered based on the two-dimensional size of the measurement area. Any detections outside the room boundaries are discarded and not processed by the tracking algorithm.

Resolving the Multimodal Space-Time Permutation Problem

Tracking multiple speakers with audio and video data poses the challenge that detections from different modalities exist in separate measurement spaces. This makes it unclear which detection belongs to which source. The MS-GLMB algorithm addresses this problem by representing multiple sources with unique labels within the labeled Random Finite Set framework. Measurements from both audio and visual sensors are associated with these labels in a unified probabilistic model, allowing the algorithm to combine complementary information from the two modalities.

The algorithm performs recursive Bayesian estimation to predict and update the states of each speaker while accounting for uncertainties and possible errors in both audio and video measurements. Likelihood models specific to each modality are used to integrate detections systematically. This fusion reduces ambiguities in speaker associations and can improve tracking robustness in scenarios such as partial video occlusions or noisy and reverberant audio.

MS-GLMB enables online tracking by continuously estimating speaker tracks over time. By jointly modeling spatial and temporal relationships of audio and visual measurements, the algorithm provides consistent and accurate multi-object tracking and effectively resolves the Multimodal Space-

5 https://github.com/linh-gist/labeledRFS/tree/main/ms_glmb_gms

Time Permutation Problem [77]. Figure 4.11 shows the MS-GLMB tracking result for tracking a single moving speaker using audio and video detections. The Audio-Visual Sensor Fusion Software Module writes the predicted speaker tracks to the JSONL file `fusion_tracks.jsonl`.

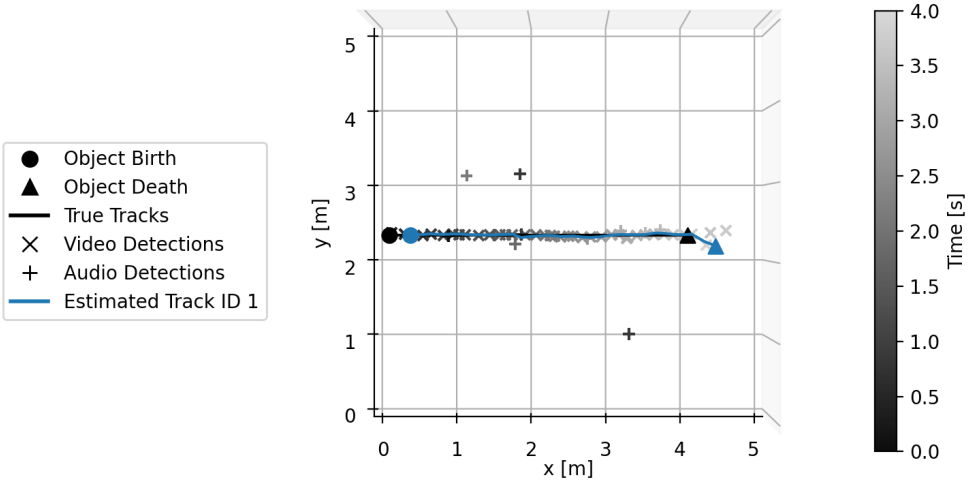


Figure 4.11: MS-GLMB speaker tracking results for tracking a single moving speaker using the audio and video detections.

5 EVALUATION

In this chapter the evaluations for the proposed multimodal audio-visual sensor fusion method for human speakers in an acoustic room is presented. First the simulated scenarios and modalities, as well as the parameters of each algorithm are defined in Section 5.1. The tracking performance depending on the available modalities is evaluated for each scenario separately in the Sections 5.2, 5.3, 5.4, 5.5. The combined results for each modality are evaluated in Section 5.6.

5.1 SIMULATION SCENARIOS, MODALITIES AND PARAMETERS

To evaluate the performance of the proposed multimodal sensor fusion approach for human speaker tracking, three multiple scenarios are defined and simulated using the developed audio-visual software architecture. The scenarios are designed to represent different levels of the scene, visual and acoustic complexity. such as:

(1) Varying number of people in the scene.

(2) Varying walking directions of people in the scene.

(3) Temporal and partial occlusion of the audio modality.

(4) Temporal and partial occlusion of the visual modality.

By systematically varying these conditions, the complementarity of the audio and visual modalities and the robustness of the tracking approach can be effectively assessed. The scenarios involve human subjects who walk across the monitored area while one individual speaks at a time.

1. **Scenario S1:** One person walks across the scene and is speaking continuously.

2. **Scenario S2:** One person walks across the scene and speaks continuously, but is visually temporarily occluded by an object.

3. **Scenario S3:** Three people walk across the scene in parallel. The person in the center walks from the other side and speaks continuously.

4. **Scenario S4:** Three people walk across the scene in parallel. The person in the center walks from the other side. They speak alternately.

It is important to note that the audio modality generally incorporates temporal occlusion, as pauses are a natural part of human speech, even when it is continuous. The remainder of this work refers to this as natural temporal

occlusion of the audio modality. The defined scenarios impose the constraint that only one person speaks at any given time. This restriction arises from the chosen PSSL algorithm, specifically the 3D-SSL method implemented in the Audio Detector Software Module. The TDOA approach employed in this algorithm provides a valid closed-form solution for the three-dimensional localization of a sound source only when a single acoustic source is active within the scene. To investigate the MOT performance of each sensed modality, the following configurations are evaluated:

1. **Modality M1:** Audio modality.
2. **Modality M2:** Video modality.
3. **Modality M3:** Audio and video modality combined.

Algorithmic Configuration and Parameterization of Software Modules

This subsection provides a detailed description of the algorithmic configuration and parameterization used for each software module. The corresponding parameter sets are summarized in Tables 5.1, 5.2, 5.3, 5.4, and 5.5. These specifications ensure the reproducibility of all simulated scenarios. Unless explicitly stated, default parameters of the respective algorithms are used. Most parameter choices are based on my previous research presented in [24] and [67], while the remaining parameters were chosen empirically. Due to the limited length of this thesis, not all parameters can be explained in detail. The interested reader is encouraged to consult the cited publications and their associated code repositories.

Table 5.1: Audio Simulation Software Module: Algorithms and Parameters

Parameter Class	**Parameter**	**Value**
Acoustic Scene Simulation	Virtual Acoustics	gpuRIR [5]
	Number of Audio Channels N_m	8
	Microphones Directivity	Omni-Directional
	3D Position of Microphone 1	[0.115, 4.545, 2.493] m
	3D Position of Microphone 2	[2.401, 4.498, 2.385] m
	3D Position of Microphone 3	[4.545, 4.545, 2.484] m
	3D Position of Microphone 4	[4.498, 2.342, 2.380] m
	3D Position of Microphone 5	[4.545, 0.115, 2.489] m
	3D Position of Microphone 6	[2.395, 0.162, 2.378] m
	3D Position of Microphone 7	[0.115, 0.115, 2.489] m
	3D Position of Microphone 8	[2.386, 2.253, 2.489] m
	Room Size	$[4.66, 4.66, 3.00]$ m
	Speed of Sound c	343.0 m/s
	Source Signal	speech.wav (male speaker)
	Reverberation Time T_{60}	0.35 s
	Interpolation Time Step Size $t_{interpolate}$	0.01 s
	Audio Output Bit Depth	32 bit
	Audio Output Sample Rate f_s	44.1 kHz
	Audio Output Format	8-channel WAV

Table 5.2: Audio Detector Software Module: Algorithms and Parameters

Parameter Class	**Parameter**	**Value**
Signal Pre-Processing	Audio Bit Depth	32 bit
	Audio Sample Rate f_s	44.1 kHz
	High-Pass Filter Cutoff Frequency f_c	200 Hz
	Window Size L	42 ms
	Window Function	Hann Window
Sound Event Detection	Detection Algorithm	Short-Time Energy
	Normalized Detection Threshold	0.01
	Minimum Time Between Events	$T_{60} \cdot L$
Positional Sound Source Localization	Sound Source Localization Method	3D-SSL [24]
	Cross-Correlation Method	β-GCC-PHAT with $\beta = 0.7$
	Speed of Sound c	343.0 m/s
	Localizations per Sound Event	35
	Fusion Method for Localizations	Euclidean Mean

Table 5.3: Video Simulation Software Module: Algorithms and Parameters

Parameter Class	**Parameter**	**Value**
Fisheye Camera Model	Forward Projection Model	Brown–Conrady
	Focal Length f_x, f_y	$[1049.47, 1049.47]$ px
	Principal Point c_x, c_y	$[960, 540]$ px
	Distortion Coefficients k_1, k_2, k_3, k_4	$[-0.2, 0, 0, 0]$
	Camera 3D Position	$[2.33, 2.33, 2.5]$ m
	Camera 3D Rotation	$[90°, 0, 0]$ (top-down view)
	Image Output Format	24-bit RGB, 8-bit Depth
	Image Output Dimensions w_{px}, h_{px}	1920 px, 1080 px
Simulation Settings	Rendering Engine	Unity
	Simulation Time t_{max}	4 s
	Simulation Update Period T	0.1 s ($\hat{=}$10 Hz)
	Speaker Moving Velocity v_x	1.0 m/s

Table 5.4: Video Detector Software Module: Algorithms and Parameters

Parameter Class	**Parameter**	**Value**
Object Detection	Convolutional Neural Network	YOLOv5n (Fisheye trained) [67]
	Image Input Format	24-bit RGB fisheye image
	Inference Image Dimension w, h	640 px, 384 px
	Minimum Confidence Threshold	0.5
	Intersection-over-Union Threshold	0.7
	Target Class	Person
2D-to-3D Backprojection	Backward Projection Model	Inverse Brown–Conrady
	Depth Information	8-bit Depth fisheye image

Table 5.5: Audio–Visual Sensor Fusion Software Module: Algorithms and Parameters

Parameter Class	**Parameter**	**Value**
Sensor Fusion	Multi-object Tracking Algorithm	MS-GLMB [86] [85] [87]
	Motion Model	Constant-velocity model
Object State Model	Sampling Period T	0.1 s
	Object State Vector X	6D $[x, v_x, y, v_y, z, v_z]$
	Process Noise Std. Dev. σ_v	0.15
Sensor Observation Models	Sensor 1 Dimensions $z1_{dim}$	Video 3D $[x, y, z]$
	Sensor 2 Dimensions $z2_{dim}$	Audio 3D $[x, y, z]$
	Sensor 1 Noise Std. Dev. σ_{s1}	$[0.20, 0.20, 0]$ m
	Sensor 2 Noise Std. Dev. σ_{s2}	$[0.25, 0.25, 0]$ m
	Sensor 1 Detection Probability P_{D1}	0.5
	Sensor 2 Detection Probability P_{D2}	0.25
	Sensor 1 Clutter Rate λ_{c1}	0.01
	Sensor 2 Clutter Rate λ_{c2}	0.01
Adaptive Birth	Initial Object State Covariance Matrix	$\mathrm{DIAG}(1.0, 0.5, 1.0, 0.5, 0, 0)$
	Birth Probability r_{birth}	0.001
	Birth Gaussian Mean	$[2.33, 0.0, 2.33, 0.0, 1.5, 0.0]$
	Birth Gaussian Std. Dev	$[2.33, 1.0, 2.33, 1.0, 1.5, 1.0]$
	Survival / Death Probability P_S, Q_S	$0.99, 1 - P_S$

5.2 TRACKING RESULTS OF SCENARIO S1

Scenario S1 represents the simplest configuration, with a single continuously speaking person and no occlusions apart from the natural temporal occlusions of the audio modality. Representative video frames and detections for this scenario are visualized in Figure .1 in the appendix. As discussed in Section 2.3.4, HOTA balances detection accuracy (DetA) and association accuracy (AssA). It furthermore quantifies localization accuracy (LocA). By decomposing HOTA in its sub-metrics enable a structured interpretation of the tracking results visualized in Figures 5.1–5.3 and Table 5.6 summarizes the results from a HOTA perspective. It is important to note that HOTA and its sub-metrics compare the predicted track detections and the ground-truth track detections. This means that the accuracy and precision of audio and video detection can only be evaluated indirectly. Each Figure visualizes the 2D room dimensions of 4.66×4.66 m. All audio and/or video detections are visualized as symbols ($\times$ or $+$) and are color-coded over time, from black to light grey, to indicate detection time relative to the simulation's start and end. The ground-truth birth, track, and death of objects in the scenario are colored black to enhance visibility. As speakers in the scenario walk at a constant speed of 1 m/s, the ground-truth tracks have linear track times from the simulation start to end. The estimated/predicted birth, track, and death of objects are color-coded in a single color based on their track ID for clarity and clean visualization.

By using the audio-only modality (M1), the MOT algorithm successfully predicts and maintains a continuous speaker track over time, but it only matches half of the ground-truth track. The high LocA of 95.6 indicates a high overall accuracy of audio-only of true positive track detections. The low DetRe of 48.0 implies many missed track detections, especially in the object birth area. This can be attributed to the fact that the uniform microphone distribution lacks a sensor at that position. Another cause is the natural temporal occlusions of human speech. On the other hand, the high DetPr of 98.4 indicates that not too many track detections are found. The DetRe and DetPr lead to an overall low DetA of 47.7, implying high uncertainty in the number of track detections. The low AssRe of 48.0 but high AssPr of 98.4 imply that only half of the associations to ground-truth are found, but not too many associations. The AssRe and AssPr result in an overall low AssA of 47.7, implying high uncertainty in the number of track associations. As a result of low DetA and AssA, the overall tracking accuracy measured by HOTA of 47.7 is also low.

By using the video-only modality (M2), the MOT algorithm successfully predicts and maintains a continuous speaker track over time, matching the whole ground-truth track very well. The high LocA of 94.0 indicates a high overall accuracy of video-only true positive track detections. The high DetRe of 81.6 implies very few missed track detections, which mainly happen in the object birth area. This can be attributed to the fact that the adaptive birth process of the MOT algorithms requires a number of detections before an

object is be born. On the other hand, the high DetPr of 95.6 indicates that not too many track detections are found. The DetRe and DetPr lead to an overall high DetA of 80.0. The high AssRe of 81.6 and high AssPr of 95.6 imply that not too few and not too many associations to ground-truth are found. The AssRe and AssPr result in an overall high AssA of 80.0. As a result of low DetA and AssA, the overall tracking accuracy measured by HOTA of 80.0 is also high.

By using the audio and video modality (M3), the MOT algorithm successfully predicts and maintains a continuous speaker track over time, matching the whole ground-truth track very well. The high LocA of 94.0 indicates a high overall accuracy of audio and video true positive track detections. The high DetRe of 81.9 implies very few missed track detections, which mainly happen in the object birth area. This can be attributed to the fact that the adaptive birth process of the MOT algorithms requires a number of detections before an object is be born. On the other hand, the high DetPr of 95.9 indicates that not too many track detections are found. The DetRe and DetPr lead to an overall high DetA of 80.3. The high AssRe of 81.9 and high AssPr of 95.9 imply that not too few and not too many associations to ground-truth are found. The AssRe and AssPr result in an overall high AssA of 80.3. As a result of low DetA and AssA, the overall tracking accuracy measured by HOTA of 80.3 is also high.

To summarise the findings of scenario S1, tracking the audio-only modality (M1) shows a high degree of uncertainty in the number of track detections, resulting in many missed associations and the lowest HOTA. Nevertheless, the MOT algorithm was able to track the speaker on half its track with high LocA. Tracking the video-only modality (M2) shows a high degree of certainty in the number of track detections and associations, resulting in a high HOTA. The MOT algorithm was able to track the speaker on its whole track with high LocA. Tracking the audio and video modality (M3) shows a high degree of certainty in the number of track detections and associations, resulting the highest HOTA, but being just very slightly better than the M2 result. This indicates that in scenario S1 the MOT algorithm best performs on the M2 modality and barely benefits from also using the M1 modality.

Table 5.6: MS-GLMB tracking results of scenario S1

Scenario S1: One person walks and speaks								
Modality	**HOTA↑**	**DetA↑**	**AssA↑**	**DetRe↑**	**DetPr↑**	**AssRe↑**	**AssPr↑**	**LocA↑**
M1 (Audio)	47.7	47.7	47.7	48.0	98.4	48.0	98.4	95.6
M2 (Video)	80.0	80.0	80.0	81.6	95.6	81.6	95.6	94.0
M3 (Audio+Video)	80.3	80.3	80.3	81.9	95.9	81.9	95.9	94.0

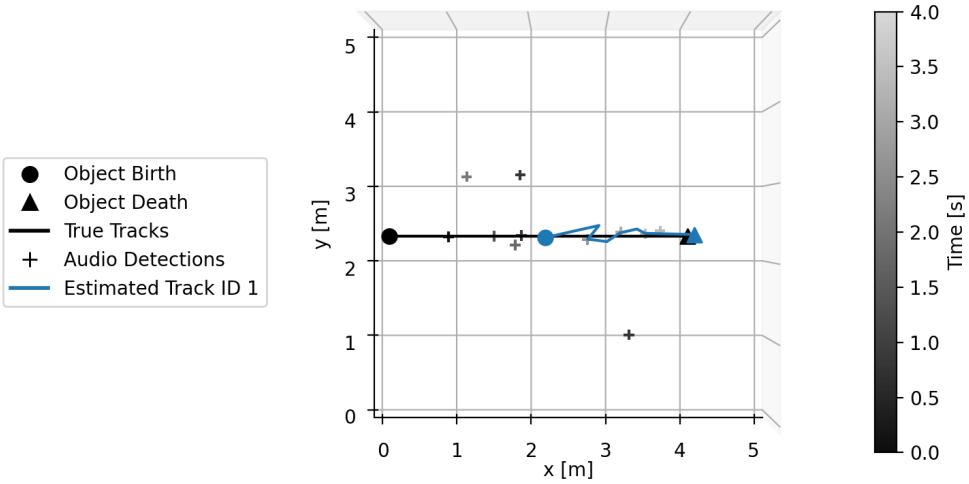


Figure 5.1: Tracking results of scenario S1 and modality M1.

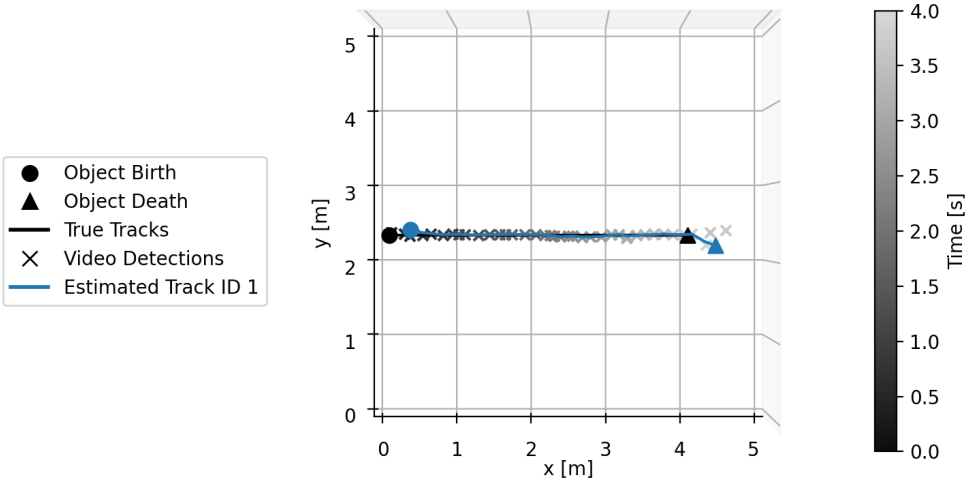


Figure 5.2: Tracking results of scenario S1 and modality M2.

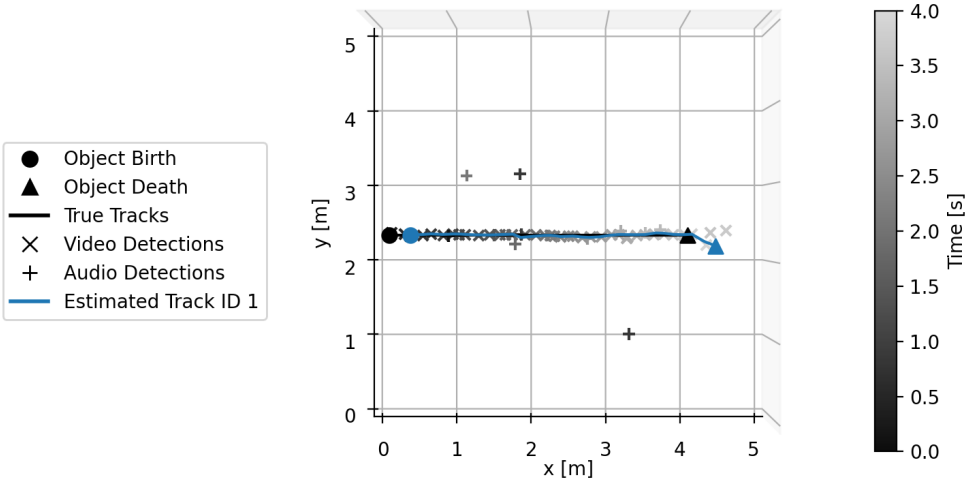


Figure 5.3: Tracking results of scenario S1 and modality M3.

5.3 TRACKING RESULTS OF SCENARIO S2

Scenario S2 introduces a temporary visual occlusion, allowing us to analyse how HOTA and its submetrics behave when the speakers visual modality (M2) is temporarily not available. The audio modality (M1) remains unobstructed. Representative video frames and detections for this scenario are visualized in Figure .2 in the appendix. The tracking results are visualized in Figures 5.4–5.6 and Table 5.7 summarizes the results from a HOTA perspective.

The audio-only modality (M1) performs identically to Scenario S1, since audio detections remain unaffected by the visual occlusion. By using the audio-only modality (M1), the MOT algorithm successfully predicts and maintains a continuous speaker track over time, but it only matches half of the ground-truth track. As a result of low DetA and AssA, the overall tracking accuracy measured by HOTA of 47.7 is also low.

By using the video-only modality (M2), the MOT algorithm predicts and maintains two continuous speaker tracks over time, instead of only one. The two tracks match the whole ground-truth track very well while the visual modality is available, but once occluded the track dies. Once the visual modality is available again a second track is born. The high LocA of 91.1 indicates a high overall accuracy of video-only true positive track detections. The low DetRe of 49.6 implies many missed track detections, which mainly happen when the speaker is occluded. On the other hand, the high DetPr of 92.5 indicates that not too many track detections are found. The DetRe and DetPr lead to an overall low DetA of 49.1. The low AssRe of 25.3 but high AssPr of 93.4 imply that one quarter of the associations to ground-truth are found, but not too many associations. The AssRe and AssPr result in an overall very low AssA of 25.1. As a result of low DetA and very low AssA, the overall tracking accuracy measured by HOTA of 35.1 is also low.

By using the audio and video modality (M3), the MOT algorithm successfully predicts and maintains a continuous speaker track over time, matching the whole ground-truth track very well. The high LocA of 92.7 indicates a high overall accuracy of audio and video true positive track detections. The high DetRe of 80.8 and the high DetPr of 94.7 imply very few missed track detections and at the same time not too many track detections are found. The DetRe and DetPr lead to an overall high DetA of 79.4. The high AssRe of 80.8 and high AssPr of 94.7 imply that not too few and not too many associations to ground-truth are found. The AssRe and AssPr result in an overall high AssA of 79.4. As a result of high DetA and AssA, the overall tracking accuracy measured by HOTA of 79.4 is also high.

To summarise the findings of scenario S2, tracking the audio-only modality (M1) shows the same results as in scenario S1 as the audio modality remained unaffected. The MOT algorithm was able to track the speaker on half its track with high LocA. Tracking the video-only modality (M2) shows a high degree of uncertainty in the number of track detections and associations, resulting in a low HOTA. The MOT algorithm lost track of the

first speaker when occlusion happend and tracked a second false speaker after occlusion. Speakers are tracked with high LocA. Tracking the audio and video modality (M3) shows a high degree of certainty in the number of track detections and associations, resulting in the highest HOTA. The results in scenario S2 show that the MOT algorithm best performs on the combined M3 modality. It substantially benefits from also using the M1 modality.

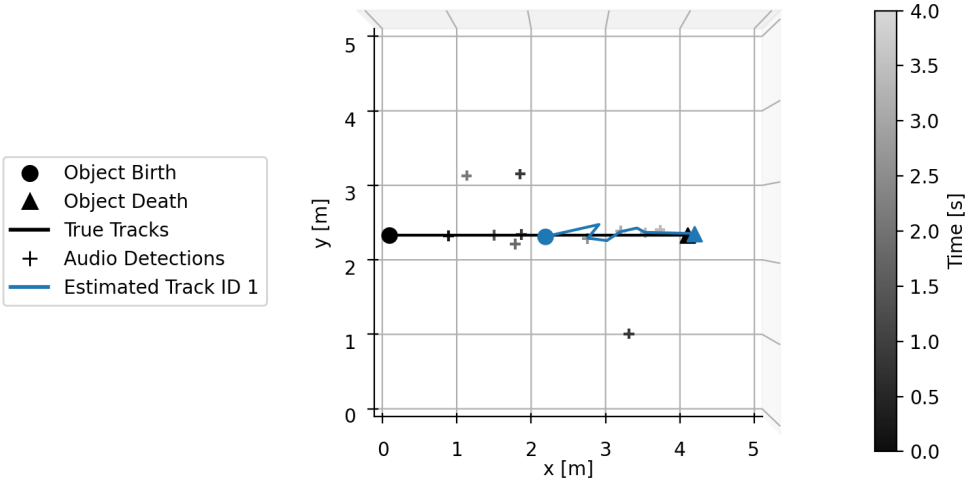


Figure 5.4: Tracking results of scenario S2 and modality M1.

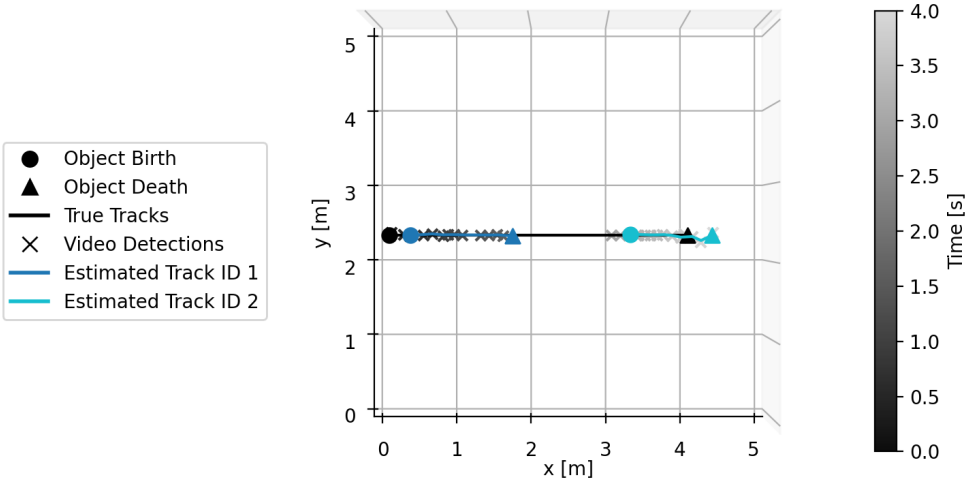


Figure 5.5: Tracking results of scenario S2 and modality M2.

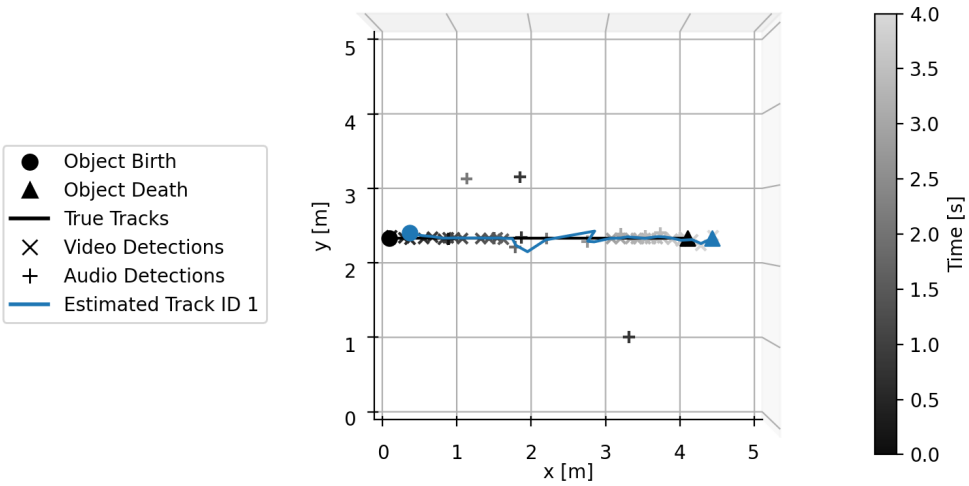


Figure 5.6: Tracking results of scenario S2 and modality M3.

Table 5.7: MS-GLMB tracking results of scenario S2

Scenario S2: One person walks and speaks; Occluded visually temporarily								
Modality	**HOTA↑**	**DetA↑**	**AssA↑**	**DetRe↑**	**DetPr↑**	**AssRe↑**	**AssPr↑**	**LocA↑**
M1 (Audio)	47.7	47.7	47.7	48.0	98.4	48.0	98.4	95.6
M2 (Video)	35.1	49.1	25.1	49.6	92.5	25.3	93.4	91.1
M3 (Audio+Video)	79.4	79.4	79.4	80.8	94.7	80.8	94.7	92.7

5.4 TRACKING RESULTS OF SCENARIO S3

Scenario S3 increases the complexity by introducing three people walking in parallel. The person in the center person is walking from the opposite side and is speaking continuously. This scenario tests how HOTA reacts to multiple speakers crossing the scene with a distance of only 1 m between their center positions. Representative video frames and detections for this scenario are visualized in Figure .3 in the appendix. The tracking results are visualized in Figures 5.7–5.9 and Table 5.8 summarizes the results from a HOTA perspective.

By using the audio-only modality (M1), the MOT algorithm predicts the center speaker track very late and the track barely matches the ground-truth track of the speaker. The high LocA of 91.2 indicates a high overall accuracy of audio-only of true positive track detections. The very low DetRe of 9.0 implies that most track detections are missed. This is expected as no audio detections are available for the upper and lower non-speaking people in the scene. For the speaking person in the center, natural temporal occlusions of human speech results in only a few audio detections. Those are too sparse for the MOT to be able predict and maintain a track. The high DetPr of 98.4 indicates that not too many track detections are found. The DetRe and DetPr lead to an overall low DetA of 9.0, implying very high uncertainty in the number of track detections. The very low AssRe of 27.2 but high AssPr of 92.9 imply that only a few of the associations to ground-truth are found, but also not too many associations. The AssRe and AssPr result in an overall very low AssA of 26.9, implying very high uncertainty in the number of track associations. As a result of very low DetA and AssA, the overall tracking accuracy measured by HOTA of 15.6 is also very low.

By using the video-only modality (M2), the MOT algorithm successfully predicts and maintains three continuous speaker tracks over time, matching the whole ground-truth tracks very well. The high LocA of 91.4 indicates a high overall accuracy of video-only true positive track detections. The high DetRe of 71.5 implies few missed track detections, which mainly happen at object births. This can be attributed again to the adaptive birth process. Another cause are missing detections of the detector for the upper speaker at the beginning. The detector shows lower performance for object in image areas with high distortion. The high DetPr of 92.6 indicates that not too many track detections are found. The DetRe and DetPr lead to an overall high DetA of 69.5. The high AssRe of 72.8 and high AssPr of 93.1 imply that a few associations to ground-truth are not found, but also not too many associations to ground-truth. The AssRe and AssPr result in an overall high AssA of 71.0. As a result of high DetA and AssA, the overall tracking accuracy measured by HOTA of 70.3 is also high.

By using the audio and video modality (M3), the MOT algorithm successfully predicts and maintains three continuous speaker tracks over time, matching the whole ground-truth tracks very well. The high LocA of 91.5 indicates a high overall accuracy of audio and video true positive track de-

tections. The high DetRe of 69.4 implies few missed track detections. As previously explained, this can be attributed to the adaptive birth process and missing audio and video detections. The high DetPr of 92.8 indicates that not too many track detections are found. The DetRe and DetPr lead to an overall high DetA of 67.6. The high AssRe of 70.9 and high AssPr of 93.4 imply that a few associations to ground-truth are not found, but also not too many associations to ground-truth. The AssRe and AssPr result in an overall high AssA of 69.2. As a result of high DetA and AssA, the overall tracking accuracy measured by HOTA of 68.4 is also high, but slightly lower than for modality M2.

To summarise the findings of scenario S3, tracking the audio-only modality (M1) shows a very high degree of uncertainty in the number of track detections, resulting in many missed associations and the lowest HOTA. The MOT algorithm was able to identify the one speaker in the center, but the predicted track barely matches the ground-truth. Tracking the video-only modality (M2) shows a high degree of certainty in the number of track detections and associations, resulting in a high HOTA. The MOT algorithm was able to track all three speaker on their track with high LocA. Only the upper person was identified later due to missing video detections. Tracking the audio and video modality (M3) shows a high degree of certainty in the number of track detections and associations, resulting in a high HOTA, but which is slightly lower than the M2 result. This indicates that in scenario S3 the MOT algorithm best performs on the M2 modality and doesn not benefits from also using the M1 modality.

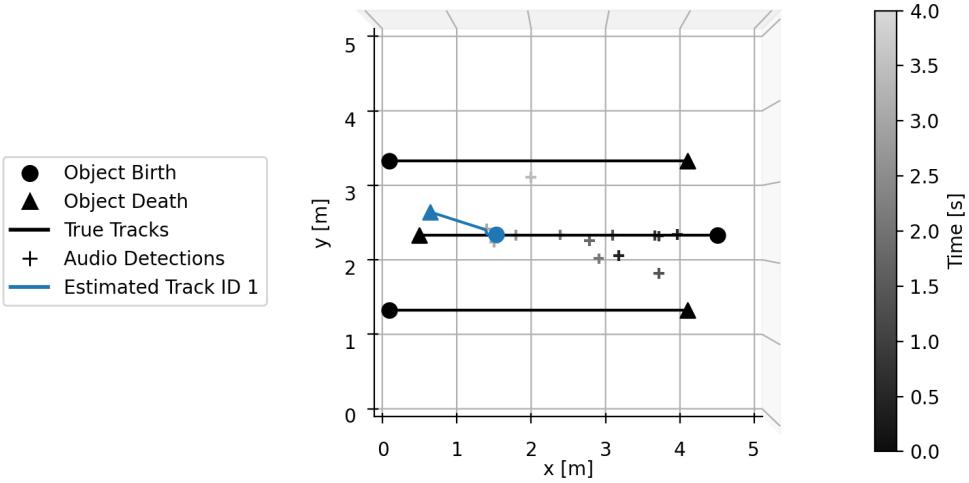


Figure 5.7: Tracking results of scenario S3 and modality M1.

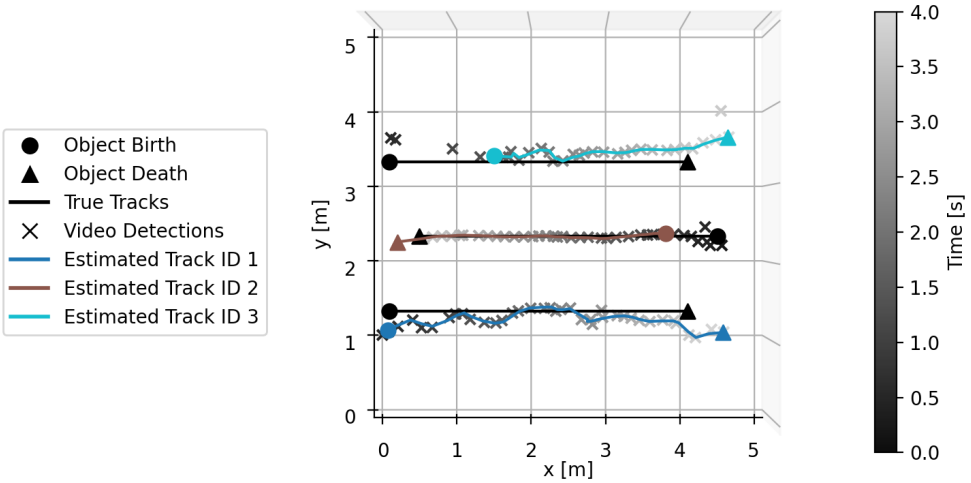


Figure 5.8: Tracking results of scenario S3 and modality M2.

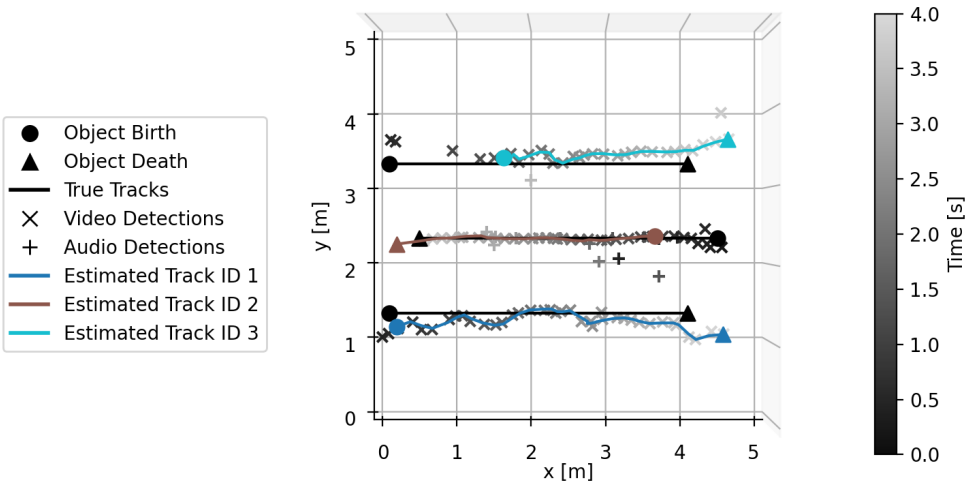


Figure 5.9: Tracking results of scenario S3 and modality M3.

Table 5.8: MS-GLMB tracking results of scenario S3

Scenario S3: Three people walk in parallel; Person in center speaks								
Modality	**HOTA↑**	**DetA↑**	**AssA↑**	**DetRe↑**	**DetPr↑**	**AssRe↑**	**AssPr↑**	**LocA↑**
M1 (Audio)	15.6	9.0	26.9	9.0	92.9	27.2	92.9	91.2
M2 (Video)	70.3	69.5	71.0	71.5	92.6	72.8	93.1	91.4
M3 (Audio+Video)	68.4	67.6	69.2	69.4	92.8	70.9	93.4	91.5

5.5 TRACKING RESULTS OF SCENARIO S4

Scenario S4 represents the most challenging scenario: three poeple walking in parallel. From lower to upper person each one speaks sequentially 1/3 of the time. This introduces frequent changes in the active audio source while the visual modality stable. Representative video frames and detections for this scenario are visualized in Figure .3 in the appendix. The tracking results are visualized in Figures 5.10–5.12 and Table 5.9 summarizes the results from a HOTA perspective.

By using the audio-only modality (M1), the MOT algorithm predicts the center speaker track very late and the track doesnt matches the ground-truth track of the speaker. The high LocA of 83.4 indicates a high overall accuracy of audio-only of true positive track detections. The very low DetRe of 7.6 implies that most track detections are missed. This is generally expected as the audio modality is only available 1/3 of the time for each speaker. The center speaker is detected by the audio detector but the other two speakers are barely detected by the audio detector while speaking. This shows the performance degradation in detecting people which are not in the center of the room. Once again, the audio modality is affected by the natural temporal occlusions of human speech, resulting in only a few audio detections. Those are too sparse for the MOT to be able properly predict the three tracks. The uncertainty results in a short living jumping object track. The high DetPr of 78.0 indicates that not too many track detections are found. The DetRe and DetPr lead to an overall low DetA of 7.5, implying very high uncertainty in the number of track detections. The very low AssRe of 12.2 and the low AssPr of 41.9 imply that only a few associations with the ground-truth are found, and these span multiple objects incorrectly. The AssRe and AssPr result in an overall very low AssA of 10.7, implying very high uncertainty in the number of track associations. As a result of very low DetA and AssA, the overall tracking accuracy measured by HOTA of 8.9 is the lowest so far.

The video-only modality (M2), the MOT algorithm performans identically to modality M2 in the previous scenario S3, since alternating speakers do not affect video detections. By using the video-only modality (M2), the MOT algorithm successfully predicts and maintains three continuous speaker tracks over time, matching the whole ground-truth tracks very well. As a result of high DetA and AssA, the overall tracking accuracy measured by HOTA of 70.8 is also high.

By using the audio and video modality (M3), the MOT algorithm successfully predicts and maintains three continuous speaker tracks over time, matching the whole ground-truth tracks very well. The high LocA of 91.4 indicates a high overall accuracy of audio and video true positive track detections. The high DetRe of 65.9 implies few missed track detections. Again, this can be attributed to the adaptive birth process and missing audio and video detections. The high DetPr of 92.4 indicates that not too many track detections are found. The DetRe and DetPr lead to an overall high DetA of 64.2. The high AssRe of 67.7 and high AssPr of 93.1 imply that a few associ-

ations to ground-truth are not found, but also not too many associations to ground-truth. The AssRe and AssPr result in an overall high AssA of 65.9. As a result of high DetA and AssA, the overall tracking accuracy measured by HOTA of 65.1 is also high, but slightly lower than for modality M2.

To summarise the findings of scenario S4, tracking the audio-only modality (M1) shows a very high degree of uncertainty in the number of track detections, resulting in many missed associations and the lowest HOTA. The MOT algorithm was able to identify a speaker in the center, but the predicted track did not matches the ground-truth at all. Tracking the video-only modality (M2) shows a high degree of certainty in the number of track detections and associations, resulting in a high HOTA. The MOT algorithm was able to track all three speaker on their track with high LocA. Only the upper person was identified later due to missing video detections. Tracking the audio and video modality (M3) shows a high degree of certainty in the number of track detections and associations, resulting in a high HOTA, but which is a bit lower than the M2 result. This indicates that in scenario S3 the MOT algorithm best performs on the M2 modality and does not benefits from also using the M1 modality.

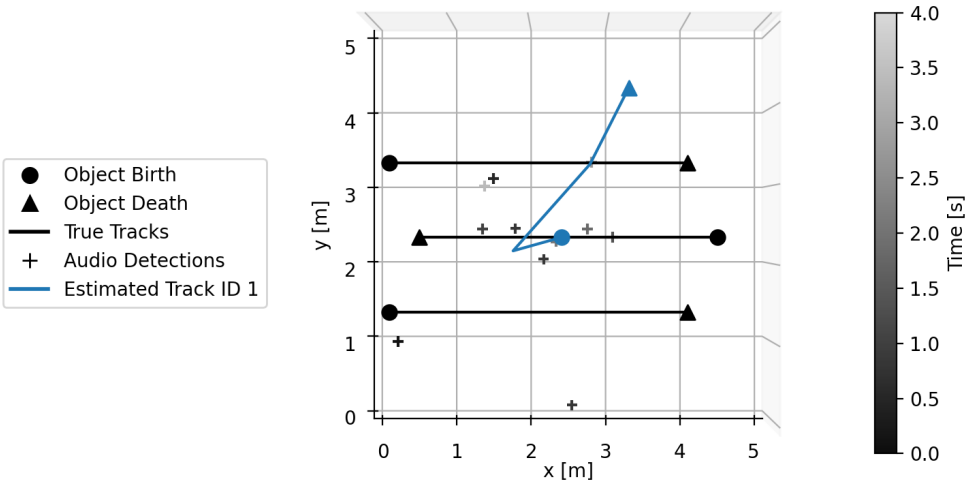


Figure 5.10: Tracking results of scenario S4 and modality M1.

Table 5.9: MS-GLMB tracking results of scenario S4

Scenario S4: Three people walk in parallel and speak alternately								
Modality	**HOTA↑**	**DetA↑**	**AssA↑**	**DetRe↑**	**DetPr↑**	**AssRe↑**	**AssPr↑**	**LocA↑**
M1 (Audio)	8.9	7.5	10.7	7.6	78.0	12.2	41.9	83.4
M2 (Video)	70.8	70.3	71.4	72.4	92.7	73.2	93.3	91.5
M3 (Audio+Video)	65.1	64.2	65.9	66.1	92.4	67.7	93.1	91.4

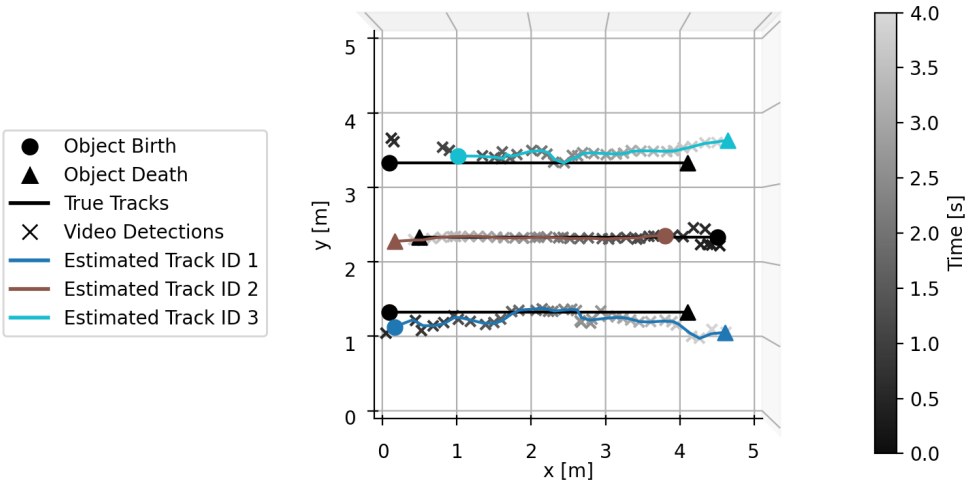


Figure 5.11: Tracking results of scenario S4 and modality M2.

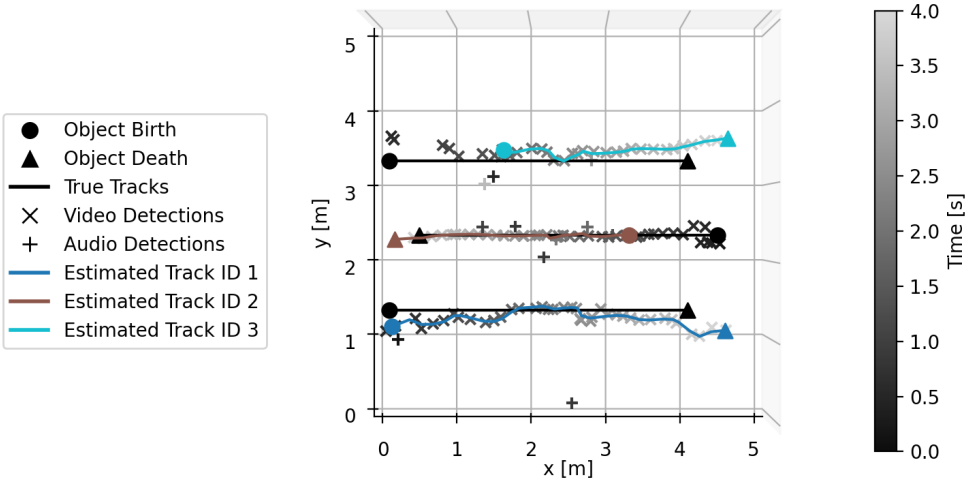


Figure 5.12: Tracking results of scenario S4 and modality M3.

5.6 TRACKING RESULTS ACROSS ALL SCENARIOS

The aggregated results across all scenarios, shown in Table 5.10, reflect the cumulative behaviour of the HOTA metric. Since HOTA integrates performance across localisation thresholds and explicitly balances DetA and AssA, it provides a clear measure of the overall robustness of each modality in the three datasets MOT25A, MOT25V and MOT25AV.

The audio-only modality (M1) exhibits low overall HOTA of 26.32, primarily due to limited DetRe caused by missing detections and its general inability to detect silent participants or in temporal audio occlusion, which is natural part of human speech, even when it is continuous. However, its high LocA of 92.66 indicates that when audio detections are available, they are highly reliable. Although AssPr and AssRe are high when a single speaker is present, association quality cannot compensate for missing detections. For the MOT25A dataset, the MOT algorithms that relies on the selected audio detector, is not able to robustly predict and maintain speaker tracks solely using the audio modality (M1).

The video-only modality (M2) provides consistently strong performance. Its high HOTA value of 68.4 reflect the robustness of video detections in all scenarios except those involving occlusion. High DetRe and DetPr reflect reliable detection, while strong AssPr and AssRe indicate stable identity over time. LocA of 91.83 indicates that when audio detections are available, they are also highly reliable. For the MOT25V dataset, the MOT algorithms that relies on the selected video detector, is able to robustly predict and maintain speaker tracks solely using a continously available video modality (M2).

The combined audio and video modality (M3) achieves the highest average HOTA across all scenarios. This demonstrates that multimodal fusion improves robustness whenever modalities are complementary, such as under occlusion or when one modality becomes unreliable. Its advantage is strongest in situations where one modality loses reliability—particularly scenario S2, where continuous audio support preserves associations during visual occlusion. However, the fusion two modalities via sensor fusion does not always exceed the performance of the best single modality in every scenario. Particularly in Scenarios S3 and S4, the modality imbalance caused by speaker-dependent audio measurements can introduce small degradations relative to the high video-only performance.

Overall, the aggregated results demonstrate that audio-visual fusion yields the best performance due to a consistent balance between detection and association performance across scenarios of different complexity levels.

Table 5.10: MS-GLMB tracking results of all scenarios

Combined Results Over All Scenarios								
Modality	**HOTA↑**	**DetA↑**	**AssA↑**	**DetRe↑**	**DetPr↑**	**AssRe↑**	**AssPr↑**	**LocA↑**
M1 (Audio)	26.32	18.08	38.40	18.26	93.59	38.92	89.60	92.66
M2 (Video)	68.54	68.61	68.49	70.40	93.10	70.17	93.74	91.83
M3 (Audio+Video)	70.29	69.38	71.24	71.18	93.39	72.88	93.97	92.04

6 CONCLUSION

This thesis investigated the simulation, detection, and fusion of audio–visual sensor data for multi-speaker tracking in indoor environments. Building on concepts in room acoustics, visual sensing, and multi-object tracking, three research questions were addressed through literature review, system design, software implementation, and experimental evaluation.

Research Question 1 asked how realistic and reproducible audio–visual sensor data can be simulated to support the evaluation of sensor fusion approaches. To answer this, a complete simulation pipeline was developed for both audio and video. Video simulation combines camera calibration, fisheye rendering, and 3D scene modeling with animated speaker models, producing geometrically consistent and visually realistic images. Audio simulation uses virtual microphone positions and Geometrical Acoustics to generate synchronized multi-channel audio streams that reflect the movement and speech of the simulated speakers. The resulting datasets MOT25A, MOT25V, and MOT25AV provide multimodal data for controlled experiments, thereby avoiding the high cost and effort involved in collecting labeled real-world datasets.

Research Question 2 focused on designing and realizing a feature-level sensor fusion software architecture for multi-speaker tracking. A modular architecture with five software modules was implemented: Audio Simulation, Video Simulation, Audio Detector, Video Detector, and Audio-Visual Sensor Fusion. Each software module has standardized interfaces and streamable outputs, which allows easy replacement or extension of individual components. Feature-level fusion is realized by combining the 3D positions from the audio and video detectors within a unified tracking pipeline. This design ensures modularity, synchronization, and reproducibility, while maintaining a balance between accuracy, robustness, and computational efficiency.

Research Question 3 investigated how audio–visual fusion improves accuracy and robustness compared to unimodal tracking. The experimental results confirm the hypotheses defined earlier: H1, that audio-only tracking is less robust during speech pauses, is supported by the low detection recall in MOT25A. H2, that if video-only tracking is affected by visual occlusion, this results in drops in association performance. H3, that audio–visual fusion improves overall performance, is supported by the higher HOTA scores in MOT25AV across all scenarios. In particular, fusion shows clear advantages when one modality is temporarily unreliable, resulting in more balanced detection and association performance across all simulated indoor scenarios.

7

FUTURE WORK

Several directions arise for future work that can further improve the realism and applicability of the proposed audio–visual tracking framework. One direction is to replace the depth information with a radial distance approximation derived from the fisheye projection model, reducing the dependence on a depth camera while still enabling 3D localization from monocular video.

The created datasets can be be extended with controlled audio and video noise to study detector and tracking robustness under more complex conditions. A higher camera position would enlarge the observable indoor area and allow full-body visibility, which is beneficial for handling occlusions. Beyond these modifications, future datasets could contain longer scenarios with more people, richer motion patterns, and varied poses, which would provide a richer basis for evaluating multi-object tracking.

Beyond dataset extensions, additional MOT algorithms could be evaluated alongside MS-GLMB to compare their behaviour using the same HOTA metrics and scenarios. Finally, comparing synthetic and real recordings of identical scenes remains an important next step. While not achievable within the time constraints of this thesis, such a comparison would further validate the realism of the simulation pipeline and its transferability to future real-world applications like in building automation.

Part I

APPENDIX

```
{
  "session_id": "Scenario1",
  "camera_config": {
    "camera_position_m": [2.33, 2.33, 2.50],
    "camera_rotation_deg": [90.0, 0.0, 0.0],
    "fx_px": 596.43,
    "fy_px": 596.43,
    "cx_px": 960.00,
    "cy_px": 540.00,
    "k1": -0.20,
    "k2": 0.00,
    "k3": 0.00,
    "k4": 0.00
  },
  "microphone_config": {
    "audio_sample_rate_hz": 44100,
    "audio_bit_depth": 32,
    "mic_positions_m": [
      { "object_id": 1, "position": [0.12, 4.55, 2.49] },
      { "object_id": 2, "position": [2.40, 4.50, 2.39] },
      { "object_id": 3, "position": [4.55, 4.55, 2.48] },
      { "object_id": 4, "position": [4.50, 2.34, 2.38] },
      { "object_id": 5, "position": [4.55, 0.12, 2.49] },
      { "object_id": 6, "position": [2.40, 0.16, 2.38] },
      { "object_id": 7, "position": [0.12, 0.12, 2.49] },
      { "object_id": 8, "position": [2.39, 2.25, 2.49] }
    ],
    "mic_directivity": "omni"
  },
  "room_config": {
    "room_dimensions_m": [4.66, 4.66, 3.00]
  }
}
```

Listing .1: JSON format of config.json.

```
fixed4 frag(v2f i) : SV_Target
{
    // Convert UV [0,1] → normalized camera coordinates (for P1)
    float2 uv = i.uv;

    // --- 1. Undo projection P1: perspective projection of virtual unity camera
    float _Fx1 = 1.0;
    float _Fy1 = 1.0;
    float x_cam = (uv.x - _Cx) / _Fx1;
    float y_cam = (uv.y - _Cy) / _Fy1;
    float3 dir_cam = normalize(float3(x_cam, y_cam, 1.0));

    // --- 2. Compute geometric quantities ---
    float2 xy = dir_cam.xy / dir_cam.z; // normalized direction
    float l = length(xy);               // radial distance in camera plane
    float theta = tan(l);               // angle of incidence

    // --- 3. Define projection model for P2 Projection ---
    // equidistant projection
    float theta_d = theta + _K1*pow(theta,3) + _K2*pow(theta,5) +
    ↪  _K3*pow(theta,7) + _K4*pow(theta,9);
    float r_equidist = theta_d;

    // --- 4. Apply P2 projection: equidistant projection ---
    // Scale r by focal lengths, preserving angle
    float x = r_equidist * xy.x/l;
    float y = r_equidist * xy.y/l;
    float2 uv2;
    uv2.x = _Fx * x + _Cx;
    uv2.y = _Fy * y + _Cy;

    // If outside bounds, sample black
    if (uv2.x < 0.0 || uv2.x > 1.0 || uv2.y < 0.0 || uv2.y > 1.0)
        return fixed4(0,0,0,1);

    return tex2D(_MainTex, uv2);
}
```

Listing .2: Fisheye fragment shader implementation of the Kannala-Brandt KB-8 forward projection.

```
fixed4 frag(v2f i) : SV_Target
{
    // Normalize coords to range [-1, 1]
    float2 uv = i.uv * 2.0 - 1.0;

    // Radial distance
    float r2 = dot(uv, uv);

    // Radial distortion factor
    float distortionFactor = 1.0
                           + _K1 * r2
                           + _K2 * r2 * r2
                           + _K3 * r2 * r2 * r2
                           + _K4 * r2 * r2 * r2 * r2;

    // Apply distortion correction
    float2 distortedUV = uv / distortionFactor;

    // Map back to [0,1] texture space
    distortedUV = (distortedUV + 1.0) * 0.5;

    // If outside bounds, sample black
    if (distortedUV.x < 0.0 || distortedUV.x > 1.0 || distortedUV.y < 0.0 ||
    ↪  distortedUV.y > 1.0)
        return fixed4(0,0,0,1);

    return tex2D(_MainTex, distortedUV);
}
```

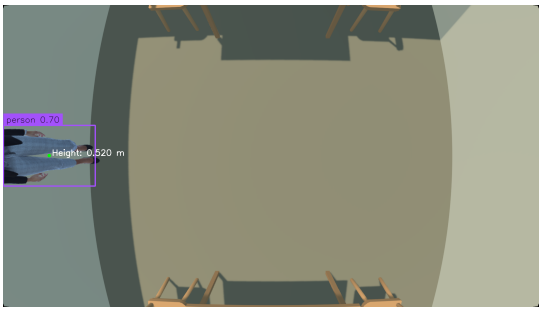
Listing .3: Fisheye fragment shader implementation of the Brown-Conrady forward projection.

```
{"timestamp": "0.0000000000", "objects": [{"position": [-0.63, 2.34, 0.52]}]}
{"timestamp": "0.1000000000", "objects": [{"position": [-0.62, 2.34, 0.46]}]}
{"timestamp": "0.2000000000", "objects": [{"position": [-0.44, 2.29, 0.35]}]}
{"timestamp": "0.3000000000", "objects": [{"position": [-0.12, 2.31, 0.83]}]}
{"timestamp": "0.4000000000", "objects": [{"position": [0.13, 2.37, 1.01]}]}
{"timestamp": "0.5000000000", "objects": [{"position": [0.30, 2.35, 0.97]}]}
{"timestamp": "0.6000000000", "objects": [{"position": [0.38, 2.32, 0.94]}]}
{"timestamp": "0.7000000000", "objects": [{"position": [0.54, 2.33, 1.06]}]}
{"timestamp": "0.8000000000", "objects": [{"position": [0.66, 2.35, 1.16]}]}
{"timestamp": "0.9000000000", "objects": [{"position": [0.82, 2.33, 1.25]}]}
{"timestamp": "1.0000000000", "objects": [{"position": [0.95, 2.35, 1.25]}]}
...
{"timestamp": "4.0000000000", "objects": []}
```

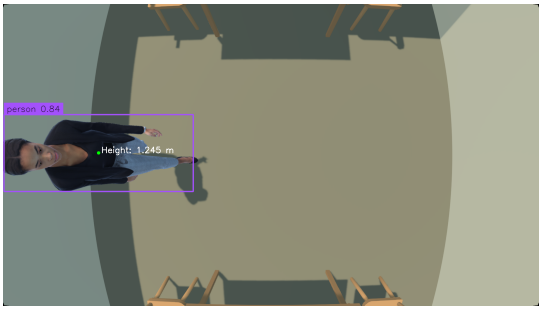
Listing .4: JSONL format of `audio_localizations.jsonl` and `video_localizations.jsonl`.

```
{
  "timestamp": "0.0000000000",
  "objects": [
    {
      "object_id": 0,
      "position": [0.0, 1.33, 0.0]
    },
    {
      "object_id": 1,
      "position": [0.0, 2.33, 0.0]
    },
    {
      "object_id": 2,
      "position": [0.0, 3.33, 0.0]
    }
  ]
}
{
  "timestamp": "0.1000000000",
    ...
}
```

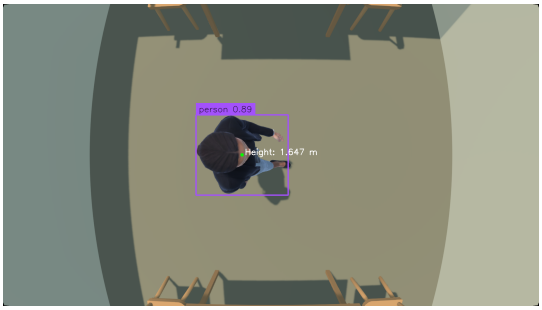
Listing .5: JSONL format of `groundtruth_sources.jsonl` and `fusion_tracking_sources.jsonl`.



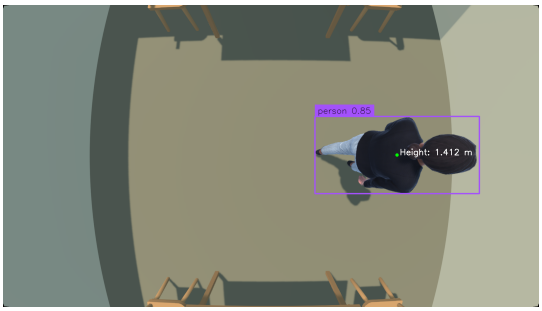
(a) t = 0.0 s



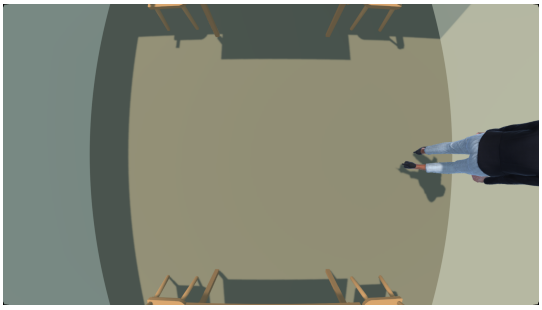
(b) t = 1.0 s



(c) t = 2.0 s

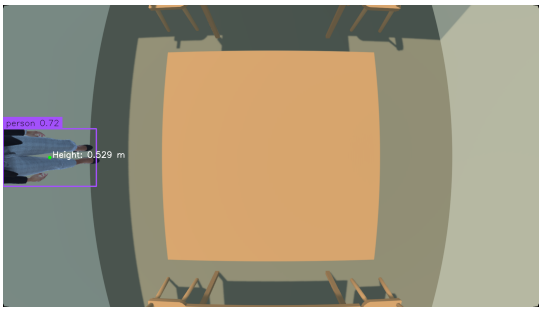


(d) t = 3.0 s

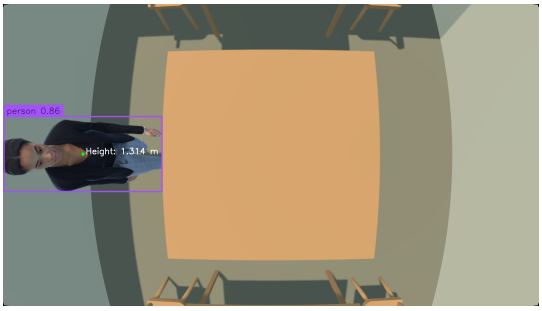


(e) t = 4.0 s

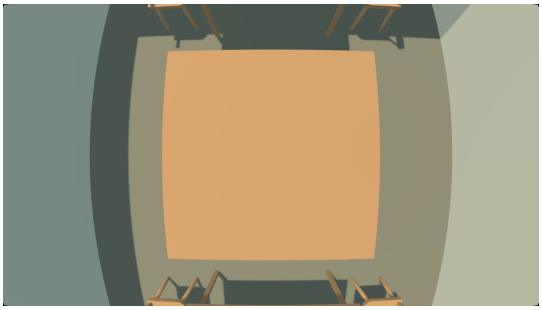
Figure .1: Representative camera frames from Scenario S1 showing video detections of the speaker in the simulated indoor environment.



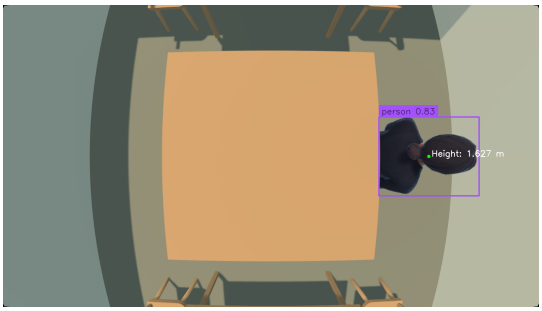
(a) t = 0.0 s



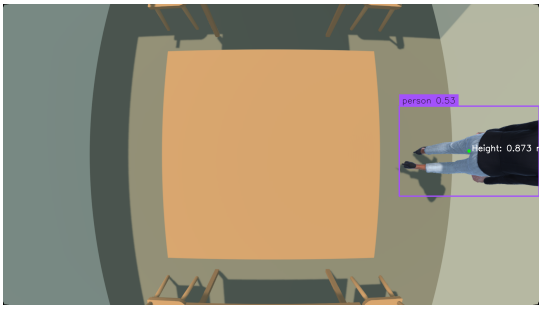
(b) t = 1.0 s



(c) t = 2.0 s



(d) t = 3.0 s

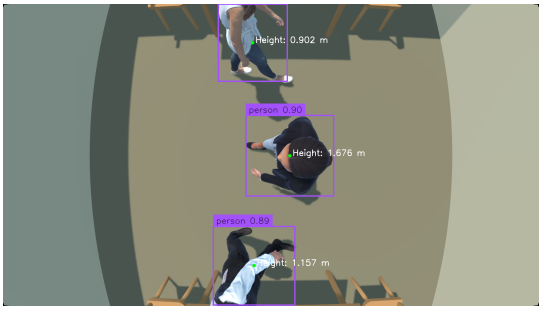


(e) t = 4.0 s

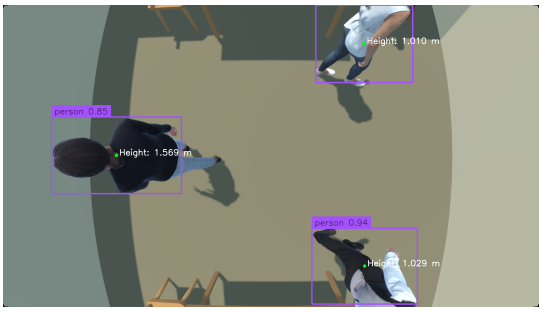
Figure .2: Representative camera frames from Scenario S2 showing video detections of the speaker in the simulated indoor environment.

(a) t = 0.0 s

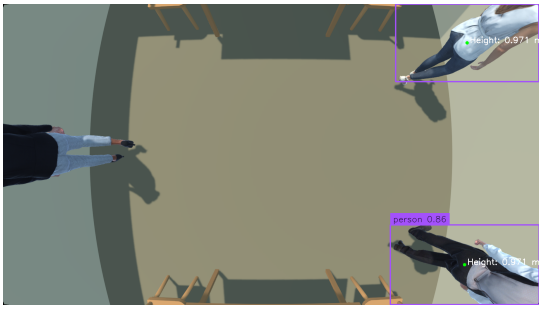
(b) t = 1.0 s



(c) t = 2.0 s



(d) t = 3.0 s



(e) t = 4.0 s

Figure .3: Representative camera frames from Scenario S3 and S4 showing video detections of the speaker in the simulated indoor environment.

BIBLIOGRAPHY

[1] Eduard Deines. "Acoustic simulation and visualization algorithms." PhD thesis. Technische Universität Kaiserslautern, 2008.

[2] Heinrich Kuttruff and Michael Vorländer. *Room acoustics*. Crc Press, 2024.

[3] Eugen Skudrzyk. *The foundations of acoustics: basic mathematics and basic acoustics*. Springer Science & Business Media, 2012.

[4] Zhenyu Tang, Lianwu Chen, Bo Wu, Dong Yu, and Dinesh Manocha. "Improving reverberant speech training using diffuse acoustic simulation." In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 6969–6973.

[5] David Diaz-Guerra, Antonio Miguel, and Jose R Beltran. "gpuRIR: A python library for room impulse response simulation with GPU acceleration." In: *Multimedia Tools and Applications* 80.4 (2021), pp. 5653–5671.

[6] Lauri Savioja and U. Peter Svensson. "Overview of geometrical room acoustic modeling techniques." In: *The Journal of the Acoustical Society of America* 138.2 (Aug. 2015), pp. 708–730. ISSN: 0001-4966. DOI: 10.1121/1.4926438. URL: https://doi.org/10.1121/1.4926438.

[7] Nikunj Raghuvanshi, John Snyder, Ravish Mehra, Ming Lin, and Naga Govindaraju. "Precomputed wave simulation for real-time sound propagation of dynamic sources in complex scenes." In: *ACM Siggraph 2010 papers*. 2010, pp. 1–11.

[8] Sönke Pelzer, Lukas Aspöck, Dirk Schröder, and Michael Vorländer. "Integrating Real-Time Room Acoustics Simulation into a CAD Modeling Software to Enhance the Architectural Design Process." In: *Buildings* 4.2 (2014), pp. 113–138. ISSN: 2075-5309. DOI: 10.3390/buildings4020113. URL: https://www.mdpi.com/2075-5309/4/2/113.

[9] International Organization for Standardization. *ISO 3382-1:2009: Acoustics – Measurement of room acoustic parameters – Part 1: Performance spaces*. https://www.iso.org/standard/40979.html. Accessed: 2025-09-01. 2009.

[10] W Yu. "The Estimation of Acoustic Parameters and Representations based on Room Impulse Responses." PhD thesis. Delft University of Technology, 2024.

[11] Albert G. Prinn. "A Review of Finite Element Methods for Room Acoustics." In: *Acoustics* 5.2 (2023), pp. 367–395. ISSN: 2624-599X. DOI: 10.3390/acoustics5020022. URL: https://www.mdpi.com/2624-599X/5/2/22.

[12] Anton Ratnarajah, Shi-Xiong Zhang, Meng Yu, Zhenyu Tang, Dinesh Manocha, and Dong Yu. "FAST-RIR: Fast neural diffuse room impulse response generator." In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 571–575.

[13] Jont B. Allen and David A. Berkley. "Image method for efficiently simulating small-room acoustics." In: *The Journal of the Acoustical Society of America* 65.4 (Apr. 1979), pp. 943–950. ISSN: 0001-4966. DOI: 10.1121/1.382599. eprint: https://pubs.aip.org/asa/jasa/article-pdf/65/4/943/11426543/943_1_online.pdf. URL: https://doi.org/10.1121/1.382599.

[14] Anton Ratnarajah, Zhenyu Tang, and Dinesh Manocha. "IR-GAN: Room Impulse Response Generator for Far-Field Speech Recognition." In: *Proc. Interspeech 2021*. 2021, pp. 286–290. DOI: 10.21437/Interspeech.2021-230.

[15] Anton Ratnarajah, Zhenyu Tang, Rohith Aralikatti, and Dinesh Manocha. "Mesh2ir: Neural acoustic impulse response generator for complex 3d scenes." In: *Proceedings of the 30th ACM International Conference on Multimedia*. 2022, pp. 924–933.

[16] Robin Scheibler, Eric Bezzam, and Ivan Dokmanić. "Pyroomacoustics: A python package for audio room simulation and array processing algorithms." In: *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2018, pp. 351–355.

[17] Yongyi Zang and Qiuqiang Kong. *GSound-SIR: A Spatial Impulse Response Ray-Tracing and High-order Ambisonic Auralization Python Toolkit*. 2025. arXiv: 2503.17866 [cs.SD]. URL: https://arxiv.org/abs/2503.17866.

[18] Changan Chen, Carl Schissler, Sanchit Garg, Philip Kobernik, Alexander Clegg, Paul Calamia, Dhruv Batra, Philip W Robinson, and Kristen Grauman. "SoundSpaces 2.0: A Simulation Platform for Visual-Acoustic Learning." In: *NeurIPS 2022 Datasets and Benchmarks Track*. 2022.

[19] Institute for Hearing Technology and Acoustics, RWTH Aachen University. *Virtual Acoustics – A real-time auralization framework for scientific research*. http://www.virtualacoustics.org/. Accessed on 2025-09-17.

[20] Dirk Schröder and Michael Vorländer. "RAVEN: A real-time framework for the auralization of interactive virtual environments." In: *Forum Acusticum*. Aalborg Denmark. 2011, pp. 1541–1546. URL: https://www2.ak.tu-berlin.de/~akgroup/ak_pub/seacen/2011/Schroeder_2011b_P2_RAVEN_A_Real_Time_Framework.pdf.

[21] *Meta XR Audio SDK for spatial audio and acoustics simulation*. https://developers.meta.com/horizon/documentation/unity/meta-xr-audio-sdk-features. Accessed on 2025-09-17.

[22] Audiokinetic. *Reflect Plugin — Wwise*. https://www.audiokinetic.com/en/wwise/plugins/reflect/. Accessed: 2025-09-17. 2025.

[23] Georg Götz, Daniel Gert Nielsen, Steinar Guðjónsson, and Finnur Pind. "Room-acoustic simulations as an alternative to measurements for audio-algorithm evaluation." In: *arXiv preprint arXiv:2509.05175* (2025).

[24] L. Sillekens, O. Rudolf, M. Thißen, I. Penner, S. Seyfarth, E. Hergenröther, and J.-P. Akelbein. "A Non-invasive Measurement System for Evaluating 3D Indoor Sound Source Localization Techniques." In: *2025 10th International Conference on Frontiers of Signal Processing (ICFSP)*. 2025.

[25] Yizhen Qiu, Bohan Li, Junchao Huang, Yandan Jiang, Baoliang Wang, and Zhiyao Huang. "An analytical method for 3-D sound source localization based on a five-element microphone array." In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–14.

[26] Nathan Cunanan, Nathaniel Dunn, Daniel Vicenti, Eric Watson, and Adam Bush. "Project minuteman." In: *Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida* (2019), pp. 32816–2450.

[27] Steven Li. "Tdoa acoustic localization." In: *Jul* 5 (2011), pp. 1–3.

[28] Yizhen Qiu, Yandan Jiang, Baoliang Wang, and Zhiyao Huang. "An analytical method for 3-D target localization based on a four-element ultrasonic sensor array with TOA measurement." In: *IEEE Sensors Letters* 7.5 (2023), pp. 1–4.

[29] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[30] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[31] Mertcan Cokbas, John Bolognino, Janusz Konrad, and Prakash Ishwar. "Frida: Fisheye re-identification dataset with annotations." In: *2022 18th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2022, pp. 1–8.

[32] Behzad Abdi, Zeynab Rokhi, Carlos Vidal, and Ali Emadi. "Scene-Centric Vehicle Trajectory Prediction at Cooperative Intersection Using Decision-Aware Attention Graph Transformer." In: *IEEE Transactions on Intelligent Transportation Systems* (2025).

[33] Ozan Tezcan, Zhihao Duan, Mertcan Cokbas, Prakash Ishwar, and Janusz Konrad. "Wepdtof: A dataset and benchmark algorithms for in-the-wild people detection and tracking from overhead fisheye cameras." In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2022, pp. 503–512.

[34] Lu Yang, Liulei Li, Xueshi Xin, Yifan Sun, Qing Song, and Wenguan Wang. "Large-scale person detection and localization using overhead fisheye cameras." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 19961–19971.

[35] Keith Man and Javaan Chahl. "A review of synthetic image data and its use in computer vision." In: *Journal of Imaging* 8.11 (2022), p. 310.

[36] Hannah Schieber, Kubilay Can Demir, Constantin Kleinbeck, Seung Hee Yang, and Daniel Roth. "Indoor synthetic data generation: A systematic review." In: *Computer Vision and Image Understanding* 240 (2024), p. 103907.

[37] Rita Delussu, Lorenzo Putzu, and Giorgio Fumera. "Synthetic data for video surveillance applications of computer vision: A review." In: *International Journal of Computer Vision* 132.10 (2024), pp. 4473–4509.

[38] Javier Marin, David Vázquez, David Gerónimo, and Antonio M López. "Learning appearance in virtual scenarios for pedestrian detection." In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 137–144.

[39] Jiaolong Xu, David Vázquez, Antonio M López, Javier Marín, and Daniel Ponsa. "Learning a part-based pedestrian detector in a virtual world." In: *IEEE Transactions on Intelligent Transportation Systems* 15.5 (2014), pp. 2121–2131.

[40] Giuseppe Amato, Luca Ciampi, Fabrizio Falchi, Claudio Gennaro, and Nicola Messina. "Learning pedestrian detection from virtual worlds." In: *International conference on image analysis and processing*. Springer. 2019, pp. 302–312.

[41] Tobias Scheck, Roman Seidel, and Gangolf Hirtz. "Learning from theodore: A synthetic omnidirectional top-view indoor dataset for deep transfer learning." In: *Proceedings of the IEEE/CVF Winter conference on applications of computer vision*. 2020, pp. 943–952.

[42] Xiaonan Pan, Qilei Sun, Jia Wang, and Eng Gee Lim. "Game Engine Based Multi-View Video Dataset Synthesis for Pedestrian Detection and Tracking." In: *2024 IEEE International Conference on Metaverse Computing, Networking, and Applications (MetaCom)*. IEEE. 2024, pp. 259–264.

[43] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, et al. "Efficient human pose estimation from single depth images." In: *IEEE transactions on pattern analysis and machine intelligence* 35.12 (2012), pp. 2821–2840.

[44] Thomas Golda, Tobias Kalb, Arne Schumann, and Jürgen Beyerer. "Human pose estimation for real-world crowded scenarios." In: *2019 16th IEEE international conference on advanced video and signal based surveillance (AVSS)*. IEEE. 2019, pp. 1–8.

[45] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Stefano Alletto, and Rita Cucchiara. "Compressed volumetric heatmaps for multi-person 3d pose estimation." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 7204–7213.

[46] Jingrui Yu, Tobias Scheck, Roman Seidel, Yukti Adya, Dipankar Nandi, and Gangolf Hirtz. "Human Pose Estimation in Monocular Omnidirectional Top-View Images." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2023, pp. 6410–6419.

[47] Shunkun Yang, Akshat Hans, Wenbing Zhao, and Xiong Luo. "Indoor localization and human activity tracking with multiple kinect sensors." In: *Smart Assisted Living: Toward An Open Smart-Home Infrastructure*. Springer, 2019, pp. 23–42.

[48] Heejae Lee, Jongmoo Jeon, Doyeop Lee, Chansik Park, Jinwoo Kim, and Dongmin Lee. "Game engine-driven synthetic data generation for computer vision-based safety monitoring of construction workers." In: *Automation in Construction* 155 (2023), p. 105060.

[49] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Andrea Palazzi, Roberto Vezzani, and Rita Cucchiara. "Learning to detect and track visible and occluded body joints in a virtual world." In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 430–446.

[50] Matteo Fabbri, Guillem Brasó, Gianluca Maugeri, Orcun Cetintas, Riccardo Gasparini, Aljoša Ošep, Simone Calderara, Laura Leal-Taixé, and Rita Cucchiara. "Motsynth: How can synthetic data help pedestrian detection and tracking?" In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10849–10859.

[51] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. "Video panoptic segmentation." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9859–9868.

[52] Philipp Krähenbühl. "Free supervision from video games." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2955–2964.

[53] Jianzhu Huai, Yuxin Shao, Grzegorz Jozkow, Binliang Wang, Dezhong Chen, Yijia He, and Alper Yilmaz. "Geometric wide-angle camera calibration: A review and comparative study." In: *Sensors (Basel, Switzerland)* 24.20 (2024), p. 6595.

[54] *An Overview of Different Camera Models*. https://kaizoudou.com/an-overview-of-different-camera-models/. Accessed: 2025-09-01.

[55] Ciaran Hughes, Martin Glavin, Edward Jones, and Patrick Denny. "Review of geometric distortion compensation in fish-eye cameras." In: *IET Irish Signals and Systems Conference (ISSC 2008)*. IET. 2008, pp. 162–167.

[56] Juho Kannala, Janne Heikkilä, and Sami S Brandt. "Geometric camera calibration." In: *Wiley encyclopedia of computer science and engineering* 13.6 (2008), pp. 1–20.

[57] Kang Liao, Lang Nie, Shujuan Huang, Chunyu Lin, Jing Zhang, Yao Zhao, Moncef Gabbouj, and Dacheng Tao. "Deep learning for camera calibration and beyond: A survey." In: *arXiv preprint arXiv:2303.10559* (2023).

[58] Kenro Miyamoto. "Fish eye lens." In: *Journal of the Optical Society of America* 54.8 (1964), pp. 1060–1061.

[59] Juho Kannala and Sami S Brandt. "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses." In: *IEEE transactions on pattern analysis and machine intelligence* 28.8 (2006), pp. 1335–1340.

[60] Jian Xu, De-Wei Han, Kang Li, Jun-Jie Li, and Zhao-Yuan Ma. "A comprehensive overview of fish-eye camera distortion correction methods." In: *arXiv preprint arXiv:2401.00442* (2023).

[61] Sangjun Lee. "Fisheye-Calib-Adapter: An Easy Tool for Fisheye Camera Model Conversion." In: *arXiv preprint arXiv:2407.12405* (2024).

[62] Vladyslav Usenko, Nikolaus Demmel, and Daniel Cremers. "The double sphere camera model." In: *2018 International Conference on 3D Vision (3DV)*. IEEE. 2018, pp. 552–560.

[63] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object detection in 20 years: A survey." In: *Proceedings of the IEEE* 111.3 (2023), pp. 257–276.

[64] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. "A survey of deep learning-based object detection." In: *IEEE access* 7 (2019), pp. 128837–128868.

[65] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee. "A survey of modern deep learning based object detection models." In: *Digital Signal Processing* 126 (2022), p. 103514.

[66] Maria Trigka and Elias Dritsas. "A comprehensive survey of machine learning techniques and models for object detection." In: *Sensors* 25.1 (2025), p. 214.

[67] O. Rudolf, R. Hecker, M. Thißen, L. Sillekens, I. Penner, J.-P. Akelbein, S. Seyfarth, and Elke Hergenröther. "Implementation of visual people counting algorithms in embedded systems." In: *Computer Science Research Notes* (2025). URL: http://www.doi.org/10.24132/CSRN.2025-4.

[68] Wilfried Elmenreich. "Sensor fusion in time-triggered systems." In: *Vienna University of Technology, Austria* (Jan. 2002). URL: https://api.semanticscholar.org/CorpusID:69628892.

[69] Eloi Bosse, Jean Roy, and Dominic Grenier. "Data fusion concepts applied to a suite of dissimilar sensors." In: *Proceedings of 1996 Canadian Conference on Electrical and Computer Engineering*. Vol. 2. IEEE. 1996, pp. 692–695.

[70] P Grossmann. "Multisensor data fusion." In: *GEC Journal of Technology* 15.1 (1998), pp. 27–37.

[71] Pek Hui Foo and Gee Wah Ng. "High-level information fusion: An overview." In: *J. Adv. Inf. Fusion* 8.1 (2013), pp. 33–72.

[72] D.L. Hall and J. Llinas. "An introduction to multisensor data fusion." In: *Proceedings of the IEEE* 85.1 (1997), pp. 6–23. DOI: 10.1109/5.554205.

[73] Matthias Schreier. "Data fusion for automated driving: An introduction." In: *at - Automatisierungstechnik* 70.3 (2022), pp. 221–236. DOI: doi:10.1515/auto-2021-0132. URL: https://doi.org/10.1515/auto-2021-0132.

[74] R.C. Luo, Chih-Chen Yih, and Kuo Lan Su. "Multisensor fusion and integration: approaches, applications, and future research directions." In: *IEEE Sensors Journal* 2.2 (2002), pp. 107–119. DOI: 10.1109/JSEN.2002.1000251.

[75] Suya Li, Hengyi Ren, Xin Xie, and Ying Cao. "A Review of Multi-Object Tracking in Recent Times." In: *IET Computer Vision* 19.1 (2025), e70010. DOI: https://doi.org/10.1049/cvi2.70010. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cvi2.70010.

[76] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. "Multiple object tracking: A literature review." In: *Artificial Intelligence* 293 (2021), p. 103448. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2020.103448. URL: https://www.sciencedirect.com/science/article/pii/S0004370220301958.

[77] Jonah Ong Soon Xuan. "Online Audio-Visual Multi-Source Tracking and Separation: A Labeled Random Finite Set Approach." In: (2021).

[78] Leandro Di Bella, Yangxintong Lyu, Bruno Cornelis, and Adrian Munteanu. "Hybridtrack: A hybrid approach for robust multi-object tracking." In: *IEEE Robotics and Automation Letters* (2025).

[79] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. "Simple online and realtime tracking." In: *2016 IEEE International Conference on Image Processing (ICIP)*. 2016, pp. 3464–3468. DOI: 10.1109/ICIP.2016.7533003.

[80] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple Online and Realtime Tracking with a Deep Association Metric." In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2017, pp. 3645–3649. DOI: 10.1109/ICIP.2017.8296962.

[81] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. "ByteTrack: Multi-Object Tracking by Associating Every Detection Box." In: (2022).

[82] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. "Fairmot: On the fairness of detection and re-identification in multiple object tracking." In: *International Journal of Computer Vision* 129 (2021), pp. 3069–3087.

[83] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. "MOTR: End-to-End Multiple-Object Tracking with TRansformer." In: *European Conference on Computer Vision (ECCV)*. 2022.

[84] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. "TrackFormer: Multi-Object Tracking with Transformers." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[85] Ba-Ngu Vo, Ba-Tuong Vo, and Michael Beard. "Multi-sensor multi-object tracking with the generalized labeled multi-Bernoulli filter." In: *IEEE Transactions on Signal Processing* 67.23 (2019), pp. 5952–5967.

[86] Ba-Ngu Vo, Ba-Tuong Vo, and Dinh Phung. "Labeled random finite sets and the Bayes multi-target tracking filter." In: *IEEE Transactions on Signal Processing* 62.24 (2014), pp. 6554–6567.

[87] Anthony Trezza, Donald J Bucci, and Pramod K Varshney. "Multi-sensor joint adaptive birth sampler for labeled random finite set tracking." In: *IEEE Transactions on Signal Processing* 70 (2022), pp. 1010–1025.

[88] Ba-Ngu Vo, Ba-Tuong Vo, Tran Thien Dat Nguyen, and Changbeom Shim. "An Overview of Multi-Object Estimation via Labeled Random Finite Set." In: *IEEE Transactions on Signal Processing* 72 (2024), pp. 4888–4917. DOI: 10.1109/TSP.2024.3472068.

[89] Ba-Ngu Vo Hyunsung Jang Moongu Jeon Linh Van Ma Tran Thien Dat Nguyen. "Track Initialization and Re-Identification for 3D Multi-View Multi-Object Tracking." In: *Information Fusion* 111 (2024).

[90] Linh Van Ma, Muhammad Ishfaq Hussain, Kin-Choong Yow, and Moongu Jeon. "3D Multi-Object Tracking Employing MS-GLMB Filter for Autonomous Driving." In: *2024 13th International Conference on Control, Automation and Information Sciences (ICCAIS)*. IEEE. 2024, pp. 1–6.

[91] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. "Hota: A higher order metric for evaluating multi-object tracking." In: *International journal of computer vision* 129.2 (2021), pp. 548–578.

[92] Yizhou Wang, Tim Meinhardt, Orcun Cetintas, Cheng-Yen Yang, Sameer Satish Pusegaonkar, Benjamin Missaoui, Sujit Biswas, Zheng Tang, and Laura Leal-Taixé. "MCBLT: Multi-Camera Multi-Object 3D Tracking in Long Videos." In: *arXiv preprint arXiv:2412.00692* (2024).

[93] Christian Nocke. "German DIN 18041 Acoustic Quality in Rooms." In: *The Journal of the Acoustical Society of America* 141.5_Supplement (2017), pp. 3686–3686.

[94] Wouter Wittebol, Huiqing Wang, Maarten Hornikx, and Paul Calamia. "A hybrid room acoustic modeling approach combining image source, acoustic diffusion equation, and time-domain discontinuous Galerkin methods." In: *Applied Acoustics* 223 (2024), p. 110068. ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2024.110068. URL: https://www.sciencedirect.com/science/article/pii/S0003682X24002196.

[95] Lloyd May, Nima Farzaneh, Orchisama Das, and Jonathan S. Abel. "Comparison of Impulse Response Generation Methods for a Simple Shoebox-Shaped Room." In: *Acoustics* 7.3 (2025). ISSN: 2624-599X. DOI: 10.3390/acoustics7030056. URL: https://www.mdpi.com/2624-599X/7/3/56.

[96] M. R. Schroeder and K. H. Kuttruff. "On Frequency Response Curves in Rooms. Comparison of Experimental, Theoretical, and Monte Carlo Results for the Average Frequency Spacing between Maxima." In: *The Journal of the Acoustical Society of America* 34.1 (Jan. 1962), pp. 76–80. ISSN: 0001-4966. DOI: 10.1121/1.1909022. eprint: https://pubs.aip.org/asa/jasa/article-pdf/34/1/76/18745094/76_1_online.pdf. URL: https://doi.org/10.1121/1.1909022.