Master of Science Thesis in Embedded Systems
and Computer Engineering

Real-time mmWave Multi-Person Pose Estimation System for Privacy-Aware Windows

Asterios Parlitsis

TU Delft Delft University of Technology

Real-time mmWave Multi-Person Pose Estimation System for Privacy-Aware Windows

Master of Science Thesis in Embedded Systems and Computer Engineering

Networked Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Asterios Parlitsis

July 5, 2024

Author:

Asterios Parlitsis

Title:

Real-time mmWave Multi-Person Pose Estimation System for Privacy-Aware Windows

MSc Presentation Date:

July 5, 2024

Graduation Committee:

Dr. M.A. Zuñiga Zamalloa (Direct Supervisor), Faculty EEMCS, TU Delft
Dr. Rihan Hai, Faculty EEMCS, TU Deflft
Girish Vaidya, PostDoc EEMCS, TU Delft

Preface

This thesis is the result of a challenging yet fulfilling journey, and I could not have been more pleased with the outcome and the entire process. First and foremost, I would like to express my gratitude towards Marco Zuñiga, my supervising professor, for his constant support throughout my final year of studies, his enthusiasm and motivation for my thesis and his exceptional character within our research group. I owe special thanks to Girish Vaidya, my advisor, for his guidance and meticulous attention to detail. His intuitive recommendations significantly eased my final year's journey, and without his guidance, the results of my work would not have been the same. Lastly, I would not want to forget my friends and colleagues and thank them all for their involvement in my research and their ongoing encouragement. Our collaborative interactions have enabled me to surpass my own expectations. I cannot thank each of them enough.

Abstract

Modern building facades and indoor partition walls feature large amounts of transparency for sufficient lighting and social safety. However, this transparency leads to concerns about privacy invasion, as sensitive objects, such as computer monitors, are exposed to onlookers. The advent of advanced screen technology has introduced VideowindoW, a smart installation capable of adjusting the transparency of its pixels to create a self-fading window, potentially addressing these privacy concerns. This thesis investigates the combined use of these smart windows together with mmWave radar technology, as a non-intrusive method to perform human posture estimation among multiple people. It develops a real-time system that detects passersby, localizes their head/eyes and obstructs their line-of-sight to sensitive indoor content, by projecting opaque squares on the smart screen. A notable gap in the mmWave literature is the insufficient handling of posture estimation challenges posed by multiple and dynamically moving targets. To address this gap, we propose the first, to our knowledge, mmWave-based Multi-Person Pose Estimation (MPPE) system. This system combines and improves two state-of-the-art methods for tracking and posture estimation and introduces a novel dataset for dynamic targets, including ground truth data for 19 human joints. Our solution demonstrated a 20% improvement in joint localization Mean Average Error (MAE) over the baseline system, in offline experiments with a single dynamic target. Furthermore, it achieved a mean blocking accuracy of 92% in online evaluations involving multiple people and varying environment. These results highlight a promising application in privacy shielding and lay the groundwork for further research in mmWave posture estimation in more unconstrained scenarios.

Contents

1 | Introduction

In modern architectural design, facades of buildings and indoor partition walls are often kept transparent to ensure sufficient lighting, thermal comfort and enhance social safety [32]. However, this transparency might lead to privacy concerns for the people inside, as passersby can view sensitive areas and objects, such as monitor screens. Permanently and completely blocking the transparency of facade windows or interior glass partitions would defeat their intended purpose and therefore, a more adaptive solution is needed; a solution that detects the presence of people and temporarily obscures their view of the sensitive object. This thesis addresses this problem and breaks it down to two main challenges. First, how to detect the people passing outside the transparent surface and second, how to block their sight, ensuring privacy over the sensitive interior content.

VideowindoW [45] has introduced a line of smart window products that offers a great solution to our second challenge; to block the view of onlookers. These smart windows are capable of projecting media on their surface by controlling the opacity of individual pixels. In other words, they can display videos, similarly to conventional computer monitors, however, their darkest pixels become opaque and the lightest pixels remain transparent. By utilizing their capability to make regional adjustments to their material's opacity, we can use these screens to interactively and locally project fading squares, obscuring the view of passersby over sensitive interior content, while maintaining the transparency of the rest of the window intact.

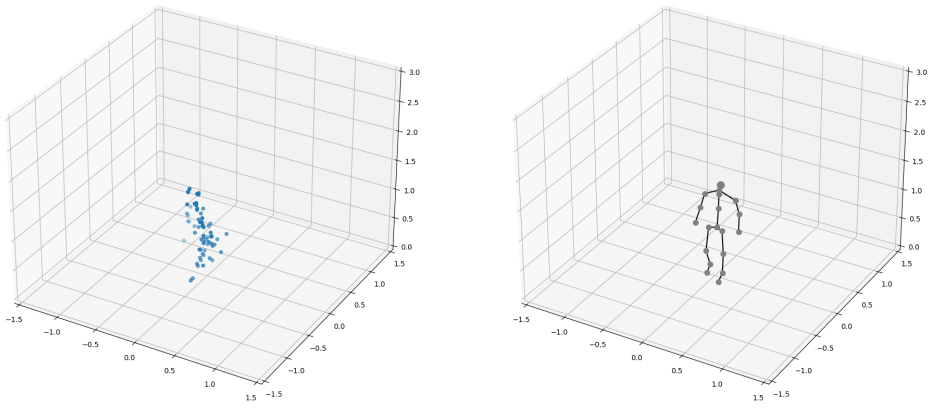
Additionally, in order to detect the people passing by, a human sensing technology is required. Human sensing refers to detecting, tracking and interpreting human presence, characteristics, postures and activities. For instance, a system that identifies a person from their walking pattern or another that detects falls in elderly homes are both examples of human sensing applications. Technologies exploited to perform human sensing include the conventional RGB cameras, RGB-D systems, such as the Microsoft Kinect sensor [49] and RF radars. Recent years have also witnessed the emergence of new sensing technologies such as LIDAR [29] and mmWave radars [48].

So far, optical sensing through cameras has been the most prevalent technology in human sensing applications and its widespread use has led to the establishment of computer vision as a dedicated field. The extensive use of cameras, however, has raised some notable privacy concerns for a wide range of use-cases, especially the ones intended for indoor areas. Their capability to capture identifiable characteristics of a person, beyond mere presence, posture or activity, poses a substantial privacy risk and as a result, people are willing to switch to alternative, privacy-preserving technologies [38]. Moreover, the General Data Protection Regulation (GDPR) poses restrictions over capturing identifiable personal information [26], encouraging the use of more privacy friendly technologies.

mmWave radar technology appeared on the foreground of technological advances during the last decade [48] and successfully addressed the concern of privacy intrusion. Operating on the RF spectrum with wavelength less than a centimeter, mmWave sensors are radar devices that abstract scenes in the form of pointclouds (Figure 1.1). As mmWave generated pointclouds are generally sparse, with resolution lower than that of cameras, they do not portrait faces or distinctive characteristics of individuals, making them better suited for a wide range of human-sensing applications. Additionally, mmWave shows notable advantages against other competitive technologies, as its is more cost-effective and environmentally resilient than LIDAR and provides better resolution than RF-WiFi sensors.

Current research on mmWave based human sensing has already reached some important milestones. By incorporating a variety of refined signal processing principles and prediction models, research has managed to create dependable human tracking systems for human presence and people counting applications [51] [44] [19]. Models targeting close range interaction have been developed to perform gesture recognition [33] and achieve seamless human-computer interaction. Intrusion detection systems able to identify individuals through their gait patterns [24] shed light on the classification challenges of the mmWave pointclouds. Extended research focused on human-joint (keypoint) detection [39] [5] [40] [9] has managed to estimate human posture with mean average error of only a few centimeters. Finally, various benchmarking datasets have been proposed [11] [5] [39] [28] addressing three major human sensing fields; human identification, posture estimation and activity recognition, allowing researchers to compare their results to other existing models and state-of-the-art research.

Through this thesis we will attempt to solve the problem at hand by investigating the potential of mmWave



(a) mmWave pointcloud (b) ground truth of human figure

Figure 1.1: Representation of a mmWave pointcloud frame

technology in estimating the posture of multiple people in the scene. Since the intended application focuses on maintaining most of the smart window surface area transparent, we aim to keep the fading squares size as small as possible. Challenges that arise, such as varying individual heights, leaning, waving or ducking movements, require more than merely tracking the human targets. We aim to localize the head/eyes of the over-lookers and accurately block their line-of-sight, therefore transitioning the problem into the field of Multi-Person Posture Estimation (MPPE). The concept of this research is portrayed in Figure 1.2.

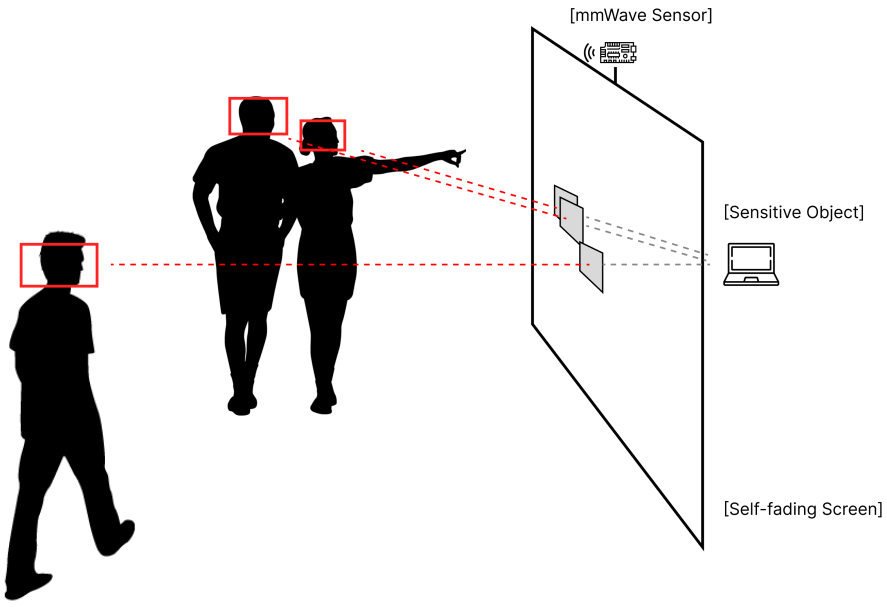


Figure 1.2: Representation of this paper's concept. A mmWave sensor captures the multiple people in scene and localizes their head/eyes. A smart window installation then projects fading squares at the appropriate position to block the line-of-sight between the target eyes and the sensitive object.

1.1. Research Challenges

The realisation of the proposed application comes with a variety of challenges that current state-of-the-art research fails to address.

- **Real time system:** The first challenge appears due to the need of our system to operate in real time. The real time constraints imposed by potential swift human movements require an approach that is lightweight in terms of memory and with low computational load, for faster execution. As we will see in Section 2.3.1, many of the proposed approaches in current literature follow processing methods that introduce heavy computational load, as they focus in optimizing performance for offline scenarios.

- **Dynamic movements:** Current research in pose estimation focuses predominantly on stationary targets performing static movements[1], while our application heeds the need of detecting the posture of individuals moving freely in the scene. This introduces challenges associated with varying and sometimes insufficient pointcloud density when people move further from the sensor, as well as noise interference in the target's localization. Additionally, most of the publicly available datasets also lack dynamic movements in their samples, revealing the need for more complete and extended training sets.

- **Multiple Targets:** Finally, so far there have been no known attempts to create a mmWave-based posture estimation system for multiple targets. The significant difference in resolution between camera-based and mmWave systems raises questions about the transferability of existing methods and techniques used in optical systems to mmWave technology. Handling multiple people simultaneously presents additional challenges, such as human-to-human occlusions and point clouds from multiple individuals merging into a single cluster. Moreover, the increased number of points in the point cloud and human joints requires precise resource management, as the computational intensity and memory requirements escalate.

1.2. Problem Statement

In order to sufficiently address the concept's challenges through the proposed solution, we restate them into this thesis problem statement. Through this research we aim to:

> "Develop a real-time mmWave-based system, able to perform pose estimation on multiple, dynamic targets, in order to localize their heads and obscure their view over privacy sensitive content through the use of self-fading VideowindoW smart windows."

1.3. Thesis Contributions

The major contributions of this thesis are threefold and listed below:

- We present a fully integrated system for the intended application of privacy shielding. It successfully blocks the gaze of passers-by over sensitive objects by utilizing a self-fading smart window and non-intrusive mmWave sensing. Through online evaluation, the system achieved 92% mean blocking accuracy for scenarios involving up to three people in the scene and opaque blocks with dimensions $20cm\ x\ 20cm$. It still managed to maintain acceptable accuracy upon increasingly challenging scenarios.

- We introduce the first, to the best of our knowledge, real-time top-down MPPE system operating on mmWave radar pointclouds, achieving precise positioning and posture estimation over multiple dynamic targets. It seamlessly combines and modifies two existing systems for tracking and posture estimation to predict the targets pose (Section 3.1) and improves upon the system's localization accuracy, posture estimation ability and pointcloud sparsity (Section 4). System testing reports a mean average joint localization error (MAE) of $13.1cm$, outperforming the selected baseline approach by approximately 20% on a single dynamic target.

- We establish a new dataset including highly dynamic and weakly supervised movements. The dataset is focused on posture estimation performed by 16 individuals of varying height, gender, body types and clothing. It comprises 88.851 labeled frames containing 5D mmWave point vectors and ground truth data for 19 human joints. The mmWave point vectors are detected through a IWR1443 mmWave sensor and include the 3D position, velocity and intensity information of the captured pointcloud. The dataset

[1] Static movements involve posture changes without resulting in changes of the overall location of the person in space. Examples of static movements are waving motions, in-position jumps, squatting and leaning motions. Dynamic movements on the other hand involve disposition, including movements such as walking and running.

will be made available to the public to promote further research on mmWave posture estimation over freely moving targets.

1.4. Thesis Outline

This thesis is structured as follows: In Chapter 2 we will explore the state-of-the-art (SoA) relevant technologies and methodologies, their principles, and perform research comparison. Furthermore, we will present the related work that will be used as a starting point for this research and identify the research gaps in their practises. In Chapter 3 we will delve into the design and implementation details of the proposed solution's fundamental structure and explore the role of each module in its chain. In Chapter 4 we will analyze the innovative improvements made over the proposed system and describe the motivation behind them. Chapter 5 will introduce our new dataset which includes dynamic and free human movements and will go through the data collection procedure, as well as the necessary preprocessing steps, before training the system. In Chapter 6 we will present the experimental results, ablation studies and the full system evaluation, followed by a discussion of their impact. The thesis concludes with Chapter 7, where we summarize the work and significant contributions, as well as suggestions for future work, limitations and potential fields of improvement.

2 | Theoretical Background

2.1. Introduction to mmWave radar technology

mmWave radar sensing is a recent radar technology with various advantageous characteristics. An mmWave radar typically operates in the high-frequency band between the ranges of 30 GHz and 300 GHz and utilize wavelengths of less than a centimeter. The short wavelength allows for significantly shorter antennas compared to other RF radar systems [2], making the design of mmWave hardware compact and cheap, as well as improving the accuracy of the system. Indeed, with the right configurations a mmWave sensor is able to detect movements of size less than a millimeter [23].

Additionally, mmWave maintains privacy due to the nature of its captured data. The received signals appear in the form of vectorized points in space, and a collection of those points is called pointcloud. Pointclouds typically capture the rough silhouettes and external contours of objects in the scene. Apart from positional data, mmWave pointclouds include attributes such as radial velocity and signal-to-noise (SNR) ratio.

mmWave technology has been embraced in applications across a wide variety of fields. Healthcare and in-house monitoring applications such as elderly critical movement recognition and rehabilitation movements correction [4] [3] [25] are already migrating to mmWave for non-intrusive solutions. Its robustness in various environmental conditions like poor lighting, fog and dust interference makes it an appealing solution for future Advanced Driver-Assistance Systems (ADAS) in autonomous vehicles [15]. The low device and installation cost[1] of mmWave allow its easy adaptation to industrial applications for material and object detection [16], to security applications such as identification, user authentication [24], human activity [11] and posture recognition [5] [1]. Finally, the increasing needs of occupancy detection for public and private smart applications [47] [44] deem the mmWave versatile nature a perfect fit.

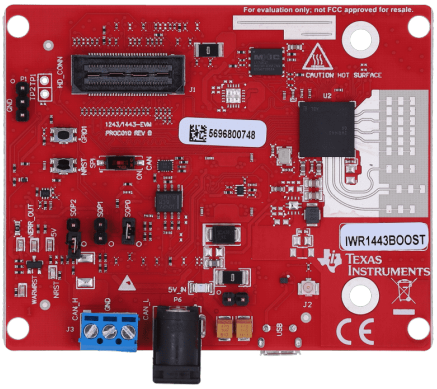


Figure 2.1: Top view of IWR1443-BOOST mmWave device by Texas Instruments

2.1.1. Why mmWave? - A comparative analysis

mmWave offers various advantages, but it is important to assess it against other sensing technologies and their respective benefits. In this section we perform a comparative analysis between mmWave and technologies such as RGB and RGB-D cameras, LiDAR, RF-WiFi sensors and wearable device-based systems for posture estimation applications.

Traditional Optical Cameras

Conventional cameras which include RGB, monochromatic and depth sensors have defined the field of computer vision, where a significant amount of progress has already been achieved. This technology's abundance of visual information has lead to the development of new human sensing methods, efficiently utilizing AI

[1] The price of a commercial mmWave radar sensor typically ranges between 10-50 euros, depending on its intended application and quality. Evaluation boards from Texas Instruments, designed for developers, are naturally more expensive, with prices varying significantly between automotive and industrial-specific sensors. [43]

state-of-the-art models and yielding highly accurate results. Furthermore, the conventional use of cameras naturally led to a plethora of labeled datasets [10] [7] [36] which further boosts future research endeavors.

On the other hand, optical cameras hold the risk of privacy intrusion due to their detailed scene representation. Studies performed on individuals reveal lower user acceptance to camera based applications in comparison to their radar based alternatives [38]. Additionally, GDPR regulations on gathering identifiable personal data introduce legal limitations to the use of camera systems for indoor applications. Finally, unlike mmWave sensors, optical cameras are highly dependent on lighting conditions, while their performance can also be negatively affected by environmental interference such as dust or fog.

Light Detection And Ranging (LIDAR)

LIDAR sensors are devices that transmit laser signals and use the principles of Time of Flight (TOF) or FMCW to abstract the scene in the form of a pointcloud. Their laser-based technology allows for higher resolution due to its significantly denser pointclouds compared to those produced by mmWave sensors. They are already heavily endorsed in automotive systems [29], achieving a precise, yet non-intrusive sensing solution.

The use of laser as an operating signal, however, comes with some concerning limitations. Its high cost and computational requirements for data processing makes this solution unsuitable for most indoor applications. Texas Instruments reported that the cost of a complete LIDAR module was estimated to be just under 10,000 USD. They predicted that with advances in technology, this price will drop to a few hundred dollars in the coming years [21]. Additionally, laser can be significantly affected by environmental conditions like fog and rain, which trigger scattering to the light beam and introduce noise to the system. Millimeter waves are generally more robust to interference and can penetrate through adverse weather conditions [15].

WiFi radar systems

Radio Frequency (RF) Wi-Fi systems developed for human sensing possess some interesting abilities. Researchers have proposed a system that is able to recreate a human's skeleton through heatmaps of received signal intensity [50]. This system is robust to environmental interference and can even detect a human posture by penetrating through walls. While this feature might unlock possibilities for novel applications, it involves the risk of becoming privacy intrusive. Furthermore, the high frequency of WiFi signals, at around 2.4G Hz, limits the system's available bandwidth and thus hinders its resolution. Alternatively, the system's large wavelength requires the use of large antennas for sufficient resolution, introducing spatial limitations. On the contrary, mmWave sensors are more compact with size of only a few centimeters. During the last years they have also become more accessible to the public with various manufacturers providing standardized designs for a variety of applications [43].

Wearable and personal sensing systems

Wearable systems rely on devices either owned by the user, such as personal cellphones, or provided by the application, such as wearable Inertial Measurement Unit (IMU) sensors. Systems providing the user with wearable devices come with the downside that their applicable use-cases are limited. They can not perform sensing on non-involved passersby and their scalability to multiple people depends on hardware availability. On the other hand, personal device dependent sensing systems assume that every person entering the scene bears a personal cellphone or smart-watch device. This assumption makes such systems unreliable sensing approaches, especially for indoor applications. Additionally, these systems are mostly suited to positioning and tracking [46] rather than human activity or posture recognition.

Table 2.1: Comparison of proposed human sensing technologies. NOTE: The operation requirements field refers to all the environmental (e.g. lighting condition, fog, dust) and hardware (e.g. wearables, phones) requirements of the technologies

	Camera	**LiDar**	**WiFi**	**Wearables**	**mmWave**
Resolution	High	High	Low	-	Medium
Operation requirements	High	Medium	Low	High	Low
Privacy-Intrusive	High	Medium	Medium	Low	Low
Cost	Low	High	Low	-	Low

2.1.2. Methodology of mmWave

In this section we delve into the principles behind the operation of mmWave technology. We present the procedures used in mmWave radars to capture and process data to set the fundamental knowledge and later discuss about the challenges that this technology introduces in the context of our application.

Frequency Modulated Continuous Waves (FMCW)

mmWave's detection capabilities lie on its ability to both transmit and receive reflected signals, carrying useful information. mmWave radars continuously transmit radar signals of modulating frequency to embed range and velocity information in the phase and velocity of the reflected signal. These radar signals are called Frequency Modulated Continuous Waves (FMCW) and an isolated FMCW segment of increasing frequency is known as a chirp (Figure 2.2a).

A chirp signal is characterized by its starting frequency fc which is the lowest transmitted frequency, the signal's bandwidth B and period Tc (Figure 2.2b). The bandwidth-to-duration ratio gives the chirp's slope S which determines the rate of increase in frequency. mmWave radar devices use these characteristics, collectively known as the chirp profile, to determine attributes such as the maximum detectable range and velocity, as well as the system's resolution.

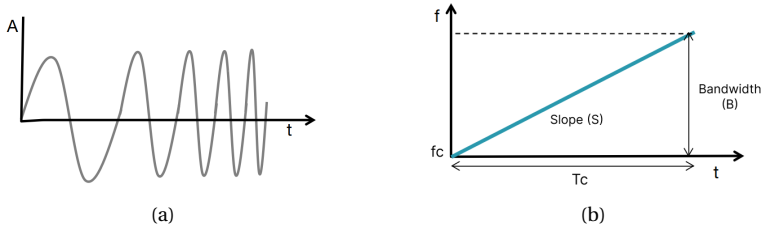


Figure 2.2: (a) FMCW chirp's amplitude as a function of time, (b) chirp signal's frequency as a function of time [23]

Analog Data processing

To extract useful information from the transmitted and received chirps, an mmWave device performs analog processing on the two signals. It first mixes them into an intermediate frequency (IF) signal, with frequency and phase equal to the difference between the instantaneous frequencies and phases of the transmitted and received signals. The IF signal is then processed through a low-pass filter (LPF) and an ADC converter (Figure 2.3). Excluding the angle information, which is determined through multiple antennas, the range and velocity information can now be extracted by the digitized IF signal.

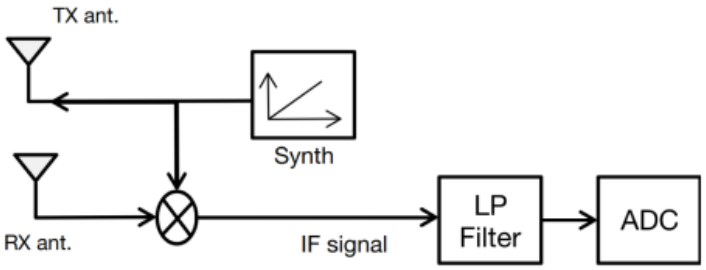


Figure 2.3: High-Level overview of the mmWave analog processing path [23]

Digital Data processing

The digital data processing stage is responsible for extracting all the useful information from the detected signals, and outputting the complete pointcloud data frame. As shown in Figure 2.4, the system first detects the range and velocity of objects by performing two Fast Fourier Transforms (FFT), one over the frequency and the second over the phase of the signals, respectively.

Then, Constant False Alarm Rate (CFAR) filtering and a collection of other post-processing methods are performed to eliminate unimportant, noisy or abundant data. The CFAR method is applied to filter out points

with low signal-to-noise (SNR) ratio. CFAR imposes a dynamic filter around a set threshold, effective maintaining a constant noise level to the pointcloud. Other post-processing methods include static clutter removal, which filters out the signal from static objects and peak grouping, which clusters points that appear very close to each other.

Finally, we combine the signal from different antennas to estimate the objects' angle and therefore have sufficient information on the objects' range, radial velocity, azimuth and/or elevation angles. Some mmWave devices further transform these angles and range into Cartesian 3D coordinates, while others use the raw values as outputs to their interface. In the next section we will analyze how the methods used in the analog and digital processing can introduce limitations to the design of our system.

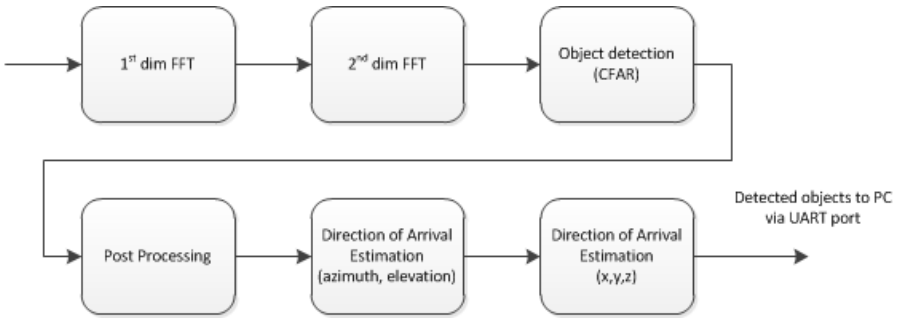


Figure 2.4: High-Level overview of the IWR1443 digital data path [23]

2.2. Challenges of mmWave

After setting the foundation on the methods used in mmWave technology, we discuss potential limitations and challenges of mmWave sensing for our intended application. We detect trade-offs between system configurations and identify potential weaknesses such as the sensor's inability to detect static points.

2.2.1. The trade-off between range and resolution

To acquire point clouds that are sufficiently dense to extract meaningful posture estimation, good spatial resolution is essential. The resolution of a mmWave sensor is defined by its ability to detect reflected signals from objects that are very close to each other. Objects are only separable if their frequency difference exceeds a specific threshold. This threshold is called unambiguous range and, as shown in Figure 2.5, it is only determined by the chirp bandwidth B. Therefore, in theory, setting a high bandwidth should provide our system with enough points to estimate a human silhouette.

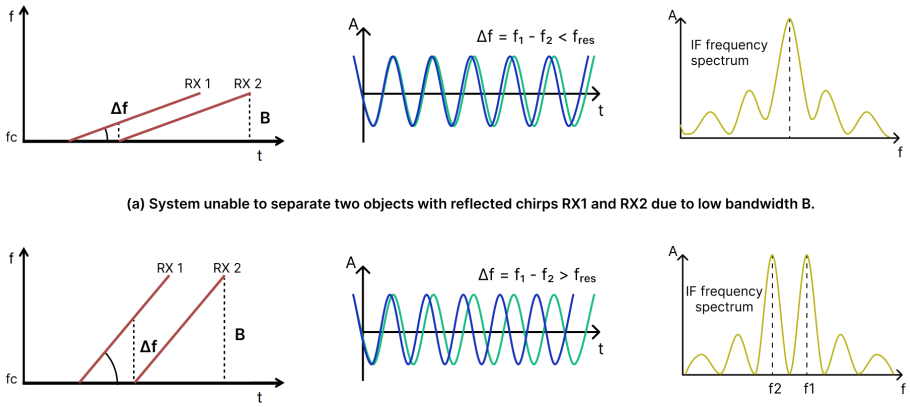


Figure 2.5: Comparison between two systems with low and high bandwidth. The first column demonstrates the frequency representation of multiple received signals over time, the second column demonstrates the chirp's amplitude over time and the last column shows the FFT result, after the signals' analog processing. NOTE: The time difference between the two received chirps stays the same in both scenarios.

A challenge arises when the application demands detection over larger ranges. The Low Pass Filter (LPF) cutting frequency and the digitization sampling frequency of the IF signal in the analog processing stage, impose constraints on the maximum allowed bandwidth by introducing a trade-off between the system's maximum detection range and its spatial resolution. In other words, to detect objects farther from the sensor, we must sacrifice some point cloud density. In our case, where we aim to detect people at over a few meters range, there is a need to look for alternative ways to enhance the pointcloud density.

2.2.2. The trade-off between noise interference and pointcloud density

Another trade-off targeting the pointcloud density can be found in the digital data processing stage (Section 2.1.2), at the CFAR filter setting. CFAR accepts a parameter to adjust the noise filtering level, which allows us to enhance the pointcloud density by also introducing more noise to the scene. However, phenomena such as multi-path might have a large negative impact on the system by introducing ghost targets and interfering with the detected target pointclouds. A careful adjustment of CFAR is necessary to avoid diminishing performance.

2.2.3. The challenge of static points detection

In the same chain of operations with CFAR, we mentioned the static clutter removal procedure. This filter detects the points with zero Doppler velocity (velocity on the radial axis of the sensor) and removes them from the scene. Unless static clutter removal is performed, static objects such as furniture and walls would appear in the pointcloud, making the detection of people a significantly harder task. The challenge that appears due to static clutter removal in our intended application is, therefore, the detection of static people. We can not depend on sufficient reflections of the targets at every frame instant and certainly, the pointcloud density of static people is expected to be considerably sparser, even temporarily non-existent.

2.3. The step after sensing: Utilizing pointcloud data

At this point we have analyzed most of the major methods and principles of mmWave radars. The next step for a mmWave-based system is to efficiently use the pointcloud information and perform some analysis, which in our case is posture estimation. In this section, we present the prominent methods through which systems process the pointcloud information.

2.3.1. Heat map and regression-based approaches

Two of the most commonly used approaches for handling the incoming pointcloud are the heat map and the regression based methods (Figure 2.6). Regression based systems follow an end-to-end approach using the detected pointclouds as direct vector inputs to the posture estimation models. Deep learning architectures such as Graph Neural Networks (GNNs) are developed to directly handle the radar points for feature extraction, while other techniques aim to make regression compatible to Convolutional Neural Networks (CNN), by reshaping the data into a matrix.

Radar heat map based approaches do not directly use the pointcloud vectors, but they first generate 2D or 3D heat maps instead. These heat maps depict local instances of point density, concentrated signal intensity or other metrics such as mean average velocity. Since the heat maps are graphic representation of fixed resolution, they are ideal inputs for CNN models. AI models have generally shown better performance on heat map based methods [7] as it is proven easier for them to learn from a map-image rather than from raw information.

On the other hand, heat map processing comes with some significant shortcomings in inference speed and computational resources. Since they abstract the scene in fixed size images, they introduce a quantization error, affecting the system's resolution [41]. To improve the resolution, we need to reduce the size of the heat-map's pixels, quadratically increasing the system's memory and computational requirements. Moreover, a 2D heat-map can represent only three attributes at once, so a typical mmWave system would further require three heat-maps per frame to include the position, velocity $\dot{r}$ and signal intensity i pointcloud information (One approach for the three heat-maps would be: $x-y-z$, $x-y-\dot{r}$, $x-y-i$)

Due to these large delays introduced by heat map generation, the increased memory and computational intensity, regression based approaches are preferred for real time and embedded applications, offering high inference speed and limited hardware demands. Since a large constraint for our application is its real-time operation, we will therefore base our solution only on regression-based approaches.

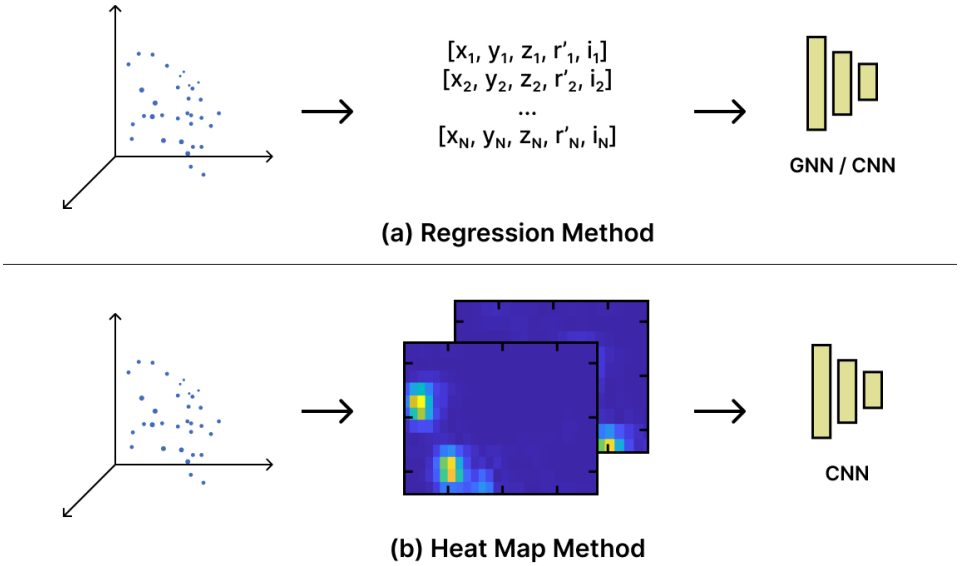


Figure 2.6: Comparison between regression and heat map based methods on pointcloud data

2.4. Related Work

Having established some of the fundamental methods used in mmWave sensing and pointcloud processing, we can now broaden our perspective and examine the state-of-the-art research over human sensing, that will serve as a starting point for this thesis. Since our proposed solution is the first attempt of a mmWave-based Multi-Person Posture Estimation (MPPE) system, our literature study first explores the MPPE methods already used in computer vision, in order to identify which of these methods can transfer to mmWave-based systems. Then, it addresses the state-of-the-art research in mmWave Single-Person Pose Estimation (SPPE) and Multiple Target Tracking (MTT) systems, in an attempt to combine the two into an integrated system able to track and estimate the pose of multiple people.

2.4.1. Advances in computer vision

Latest advances in AI have significantly contributed to the evolution of human posture estimation systems, particularly in computer vision. Recent findings have evolved past performing posture and activity detection in monitored environments to what is known as human sensing applications "in the wild". They have successfully addressed the challenges introduced by multiple targets and unforeseen objects in the scene, occlusions, varying body shapes, clothing, etc. [36] Two dominant approaches categorizing the existing work on MPPE systems are the top-down [34] [17] [12] and the bottom-up methods [37] [20] [8] [50].

Top-down and Bottom-up systems

The top-down method first tracks the human targets and creates bounding boxes around them, effectively isolating them from their surrounding environment. Then, it applies pose estimators at each one of them individually and re-positions their predictions back to the scene. On the other hand, the bottom-up approach, which is gaining increasing attention in recent work, first detects all the key-point proposals (joints) in the scene and then connects them into potential human-target instances. A representation and comparison of both methods over optical inputs is made in Figure 2.7

Both methods have their advantages and disadvantages, depending on the system's setting and requirements. Bottom-up methods outperform the ones utilizing the top-down method in popular datasets like COCO [30] [10]. They also appear to be less computationally intense and faster than top-down methods, since they avoid iterating the posture estimator model over every single detected target. On the other hand, top-down methods have the ability to predict the movement of targets, since they use a tracking system to create bounding boxes. Tracking systems typically use motion models and can therefore continue estimating the targets' position during large scale occlusions. More importantly, top-down approaches do not require high sensing resolution because they provide a rough pose estimate over the entirety of the target's figure, compared to bottom-up that identify every human joint separately.

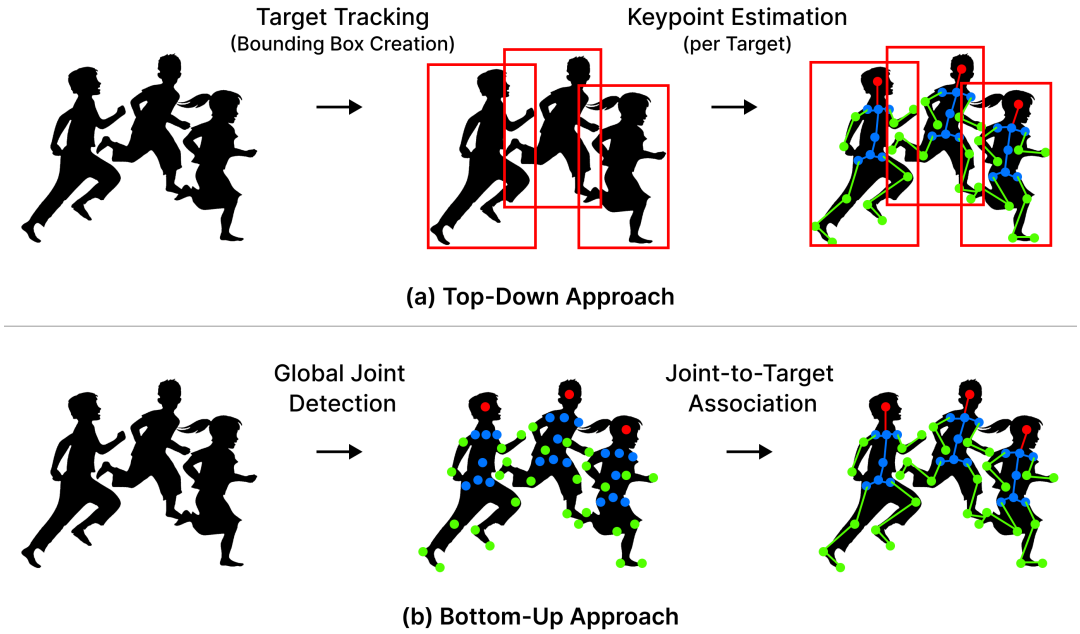


Figure 2.7: Comparison of (a) top-down approach and (b) bottom-up approach for Multi-Person Pose Estimation

Transitioning to mmWave technology

The top-down method appears best for mmWave-based applications. Compared to the abundance of information in image or video frames obtained through commercial cameras, the radar pointcloud frames typically face the problem of sparsity. For example, the limited antennas on commercial mmWave PCBs only allow up to a total of hundreds of points per frame [6] and are limited even further due to the datapath filtering. This sparsity, together with the lack of significant distinguishing features, such as color, and the low resolution (joints like wrists or ankles are really hard to tell apart) in mmWave pointclouds, make the implementation of a bottom-up system considerably challenging. Therefore, given these constraints, the top-down method proves to be the most viable option for mmWave point cloud data.

2.4.2. Advances in radar technologies

The only approach, so far, for radar-based MPPE has been made through WiFi radar. In the field of mmWave, attempts have been made to combine tracking with activity recognition [25] [3] and human identification [24], yet, to the best of our knowledge, there has been no related work in the field of MPPE. Moreover, research in mmWave-based SPPE primarily focuses on static targets, neglecting the challenge of localization in human sensing. By examining the mmWave literature over SPPE and tracking systems, we envision combining the most appropriate findings into a fundamental MPPE system over which we will improve in this thesis. Finally, we will examine the existing datasets and identify shortcoming for their use in our application.

WiFi Multi-Person Pose Estimation

The first attempts of detecting the posture of people through radar technology have been focused on WiFi signals. In [2] the researchers managed to capture the silhouette of a human target, even behind a wall, by using a modified WiFi antenna setup. The authors in [50] built upon the previous findings and proposed a bottom-up MPPE, heat-map based system, utilizing the same WiFi setup and a teacher-student network where a camera provided the ground truth of the joint's location during training. The system was able to detect a human with an accuracy of 83%, but since no tracking was performed, the errors were mainly caused by false key point detection and low capacity in dealing with occlusions. Furthermore, the system was able to gain sufficient resolution for a bottom-up approach through the use of a large setup with multiple antenna arrays, which raised the question of deployment suitability for such a device.

mmWave Single-Person Pose Estimation

mmPose [39] was proposed in 2020 and was the first mmWave based system able to detect more than 15 human keypoints from a single target. The researchers used two mmWave radars to address the sparsity of

information and enhance the resolution in both axes. mmPose followed the heat map method by generating two 2D range maps, one for each sensor with different orientation, and then incorporated a forked-CNN architecture to fuse them and perform joint localization. The model came with high computational cost and storage requirements due to the large CNN model and the projection of point cloud data into two different planes. Furthermore, their dataset was inaccessible to the public, leading to further needs for public benchmarking sets.

More heat map based approaches followed right after. mPose [40] first performed human detection to remove environmental noise and introduced a domain discriminator to remove subject-specific characteristics. HuPR [28] proposed a novel model as well as a new preprocessing approach at the on-chip data path to better extract velocity information. It used cross and self attention modules for data fusion and downsampling and finally a GCNN to estimate the pose. The authors in [9] introduced a model architecture of two streams combining temporal and spatial data, one learning the joint constraints and the other finding patterns of joint movements between frames. The results from all approaches were robust but they still did not manage to address the high computational intensity of the heat map approach and the complexity of model architecture.

In order to solve this problem for real time applications, researchers developed MARS [5], a simple yet effective baseline model which operates on raw pointcloud data. MARS used sorting of the 3D coordinates as a way to spatially align them and eventually pass them through a CNN-based estimator. Moreover, a raw pointcloud data benchmark dataset was released, but as its use case was rehabilitation exercise, it referred only to a single person performing static movements. FUSE [6] built upon MARS and was the first paper that addressed the sparsity of information by merging consecutive frames into a single pointcloud. The results showed that in a system operating at 10 fps, the combination of approximately three frames yields the best results. Due to its regression-based approach and architecture simplicity, we choose the MARS model as the posture estimation foundation of our system and we set the frame combination approach of FUSE as one of the baselines for the evaluation of our system (Section 6).

Human Detection and Tracking

mID [51] first presented a comprehensive Multiple Target Tracking (MTT) system for mmWave, which made use of DBSCAN clustering and a Kalman Filter (more in Appendix A.1 and A.2). It proposed an altered Euclidean distance metric to better detect human silhouettes and performed a cluster-driven temporal track association. It reapplied the clustering algorithm on every frame and used the Hungarian algorithm, an optimization algorithm, to associate the detected clusters to the existing system tracks by minimizing the combined distance loss. This approach put too much trust on the cluster detection, expecting that in the majority of pointcloud frames there will be sufficient information to re-detect the clusters. In reality, occlusions, signal loss, even the increasing detection range can negatively impact the density of the received signal and the signal-to-noise ratio, resulting in performance loss.

Another approach [19] attempted to provide a high performing tracking system by enhancing its clustering detection ability. The researchers proposed two alternatives to the DBSCAN algorithm together with a recursive estimation method utilizing a Kalman Filter. They still did not address the cases of insufficient pointcloud data as they based their results on a mmWave sensor able to generate denser pointclouds at the expense of lower processing speed. To address the problem of occlusion, the same research group later proposed adding a second sensor to the system with an almost perpendicular Field of View (FOV) and performing data fusion [18]. Despite their results demonstrating great performance, their multi-sensor system setup raises questions about its suitability for different applications.

Other tracking approaches appearing in the literature try to approach the detection component of the system through different technologies. Researchers developed a system that performs tracking through the radar method CA-CFAR [24]. CA-CFAR tracks a target by comparing its reported signal's energy to the instantaneous CFAR threshold, removing noise and other interference. Apart from not addressing the problem of occlusions, this method solely operates on SNR heat maps, losing a lot of information from the pointcloud data. Finally, [35] proposed a robust Single Person Tracking system using an end-to-end Deep Neural Network for detection and tracking. It outperformed other SoA approaches and managed to deal with hard-to-model radar reflections and sources of noise. The system's ability, however, to adapt to new environments, generalize to different users and most importantly scale to a MTT system is not evaluated and would require extensive training resources.

Eventually, the Texas Instruments team introduced GTRACK [44], a tracking algorithm well adapted for some of the TI mmWave sensors. It used a vanilla DBSCAN algorithm for clustering but proposed a funda-

mentally different approach on tracking through gating. Instead of re-discovering all the clusters in the scene at every iteration, the already observed tracks formed ellipsoid gates in the 3D space and the points that fell within them were assigned to their respective track. This means that even a small amount of detected points could keep a track active and provide feedback to its motion model. This gating system manages to address a big challenge in our intended application; the pointcloud sparsity caused by human-to-human occlusions, large detection range and static people. Thus, we choose GTRACK as our baseline system for people tracking.

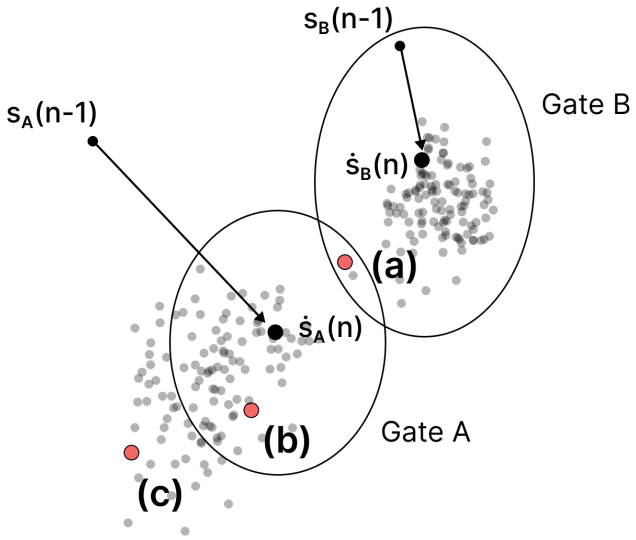


Figure 2.8: GTRACK's point-to-track association through gating. Point (a) is ambiguous between the two gates and we be associated to the track with the lowest distance metric score. Point (b) will be associated to track A without any ambiguities. Point (c) is not included in any gate, so it will be labeled as unassigned and passed through DBSCAN clustering to detect new clusters in the scene.

Pose Estimation Datasets

In the context of dataset creation, as mentioned in Section 2.4.2, mmPose was the first approach that provided a model and dataset including more than 15 human joints, however their resources remained unavailable to the public. HuPR [28] followed up and provided a publicly available dataset with generated heat maps for posture estimation, at the same time expanding the scope of research to movements featuring global displacement. It included the action of walking, introducing the concept of positioning error in the model benchmarking. However, it focused only on the velocity information of the dataset, excluding attributes like range and signal intensity. Furthermore, the real-time limitations of its heat map approach make it an unsuitable set for our intended application.

MARS suggested a dataset of raw pointcloud data, focusing on rehabilitation exercises and notably increasing the number of distinct actions performed by the participants. However, these actions were performed in a strictly static manner, where the participants had to stand within a fixed circle of 1m diameter for all the experiments. The mRI dataset [4] attempted to include multiple technologies in synergy in their dataset to address different applications. They extended the MARS dataset activities to include some very basic dynamic movement and the increased dataset size allowed for more thorough training. Finally, MiliPoint [11] attempted to establish a benchmark which addressed not only posture estimation but other human sensing tasks too. It provided a dataset with significantly larger size than that found in the existing related work, and incorporated a wide variety of 49 human movements. Unfortunately, this research was once again conducted in a relatively stable indoor environment and focused on static movements, capturing data over a strictly designated area.

For our privacy-shielding application, the appropriate dataset would require raw point cloud data to facilitate regression-based processing and include dynamic movements, such as walking, which is the primary action of interest. However, the proposed benchmark datasets either utilize heat-map formatting or fail to address dynamic movements adequately. Moreover, an additional limitation of the proposed datasets is that they consist of sanitized frames with minimal environmental noise and multi-path phenomena. Consequently, issues such as environmental noise, multipath, human-to-human occlusions, and foreign objects

are rarely addressed and their impact is hardly evaluated. These observations highlight the necessity for a new dataset that addresses these shortcomings and broadens the scope of existing literature. Table 2.2 provides a comparative overview of all the proposed posture estimation datasets, including our own.

Table 2.2: Comparison of existing mmWave datasets for the task of posture estimation. (*) In the mRI dataset only 1 of the 12 movements features global displacement in a strictly straight line (**) Our dataset includes 11 distinct actions at different orientations, as well as unsupervised, free movements.

	Movement Type	**Movements**	**Processing Type**	**Participants**	**Size (frames)**	**Publicly Available**
mmPose	STATIC	4	heat map	2	15k	No
MARS	STATIC	10	regression	4	40k	Yes
MiliPoint	STATIC	49	regression	11	213k	Yes
mRI	STATIC*	12	regression	20	160k	Yes
HuPR	DYNAMIC	3	heat map	6	141k	Yes
Our dataset	DYNAMIC	11**	regression	16	89k	Yes

3 | Design and Implementation

The proposed solution for the top down MPPE system combines and improves upon two of the existing systems described in the literature analysis, the G-Track algorithm proposed by Texas Instruments [44] for tracking applications and the MARS model [5] for posture estimation. In this chapter we will describe the design and implementation details of the baseline system that integrates these two approaches into a seamless MPPE pipeline. The code base of this project is publicly available at github.com/AsteriosPar/mmWave_MSc.

3.1. System Overview

The full pipeline of the system consists of four basic modules:

- **Interface Processing Module:** This subsystem handles the mmWave radar configurations and outputs, applying post-processing to transform the data into state vectors.
- **Tracking Module:** This module integrates GTRACK to detect and track the targets in the scene, outputting a target list with their attributes to the next stage.
- **Posture Estimation Module:** Utilizing the MARS model, this module receives the target list and produces a list of 3D estimated coordinates for all detected human joints.
- **Visualization Module:** This final module uses the list of 3D coordinates to determine the sizing and position of fading square blocks on the screen.

A high-level overview of the entire system, as well as the hardware chain, is shown in Figure 3.1.

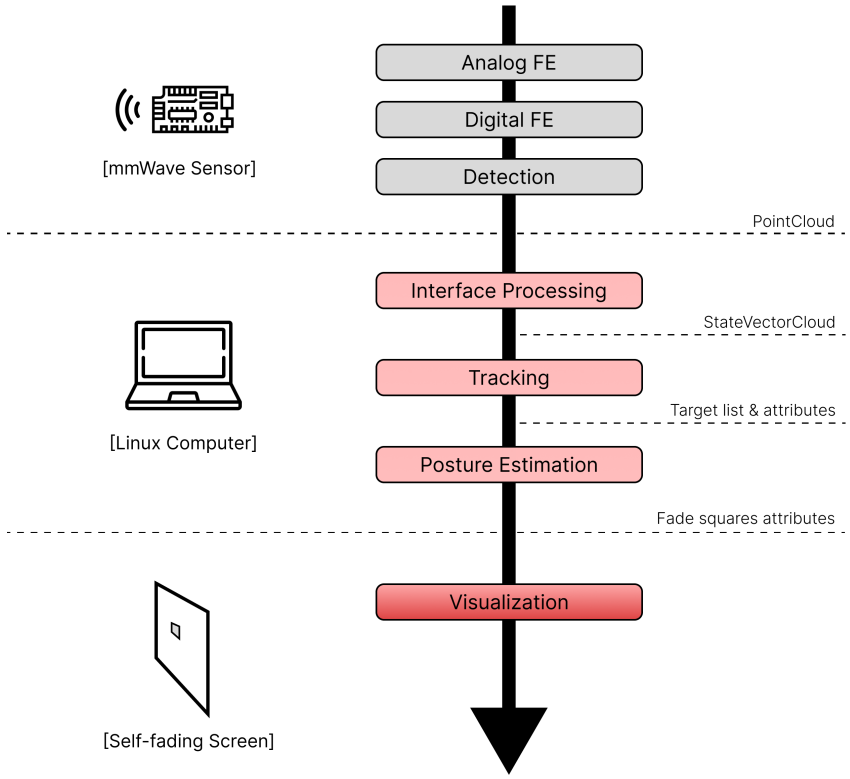


Figure 3.1: Overview of the hardware setup and software pipeline of the proposed system.

3.2. Interface Processing Module

3.2.1. Choice of mmWave sensor

The selected mmWave sensor for this thesis is the IWR1443 by Texas Instruments [43]. IWR1443 operates in the 76 GHz to 81 GHz range and features 3 transmitting and 4 receiving antennas arranged both horizontally and vertically. Unlike most devices in the adjacent production series, IWR1443 does not own a dedicated on-chip DSP, which introduces hardware limitations on the maximum FFT size and decreases resolution [22]. However, it compensates with much lower processing latency and power consumption, making it ideal for embedded, real time applications.

An additional difference appears in the processing datapath stage of the IWR1443, particularly in its output formatting. The IWR1443 converts the typical mmWave outputs—range, azimuth, and elevation values—into Cartesian coordinates. Such changes might impact the transferability of systems between different sensors. In Section 3.3.2, we will address significant modifications needed to adapt a system designed for a different interface to our application platform.

3.2.2. Radar setup and configurations

Starting with the first module in the system's pipeline, it is crucial to provide the appropriate configurations for the sensor, clearly defining the constraints and scope of the target application.

To sufficiently monitor the area of interest, the sensor should have a wide Field of View (FOV) and be capable of operating over larger ranges. The antenna configuration on the IWR1443 sensor allows for higher angle resolution along one axis. Given that the main activity of interest involves horizontal motion, we assign the highest resolution to azimuth angle detection. Additionally, considering the limits of human vision, we assume that a maximum range of 8-10 meters is appropriate for the application.

The sensor should also be able to capture and keep track of swift human movement, as well as detect minor movements of static people. These call for a sensing frequency high enough to not miss significant changes in postures, yet, accounting for the full system's computational delay. Furthermore, the CFAR setting, which sets a dynamic threshold to the noise, should be adjusted low enough for the sensor to capture sufficient data even from minor movements, while maintaining the environmental interference to moderate levels. Finally, the system should enable the static clutter and peak grouping functions to avoid processing irrelevant and abundant data and adhere to its real time constraints. Table 3.1 shows some major configurations for the mmWave sensor used in our application.

Table 3.1: List of major configuration parameters related to mmWave in this work.

Description	Values
Range Resolution (m)	0.044
Max. unambiguous Range (m)	8.02
Max. Radial Velocity (m/s)	1
Radial velocity resolution (m/s)	0.13
Frame Duration (msec)	100
CFAR threshold scale	1130
Range Peak Grouping	enabled
Doppler Peak Grouping	enabled
Static clutter removal	enabled

3.2.3. Radar's Interface

With the right configurations passed into the radar, an interface is now needed to transfer the detected point-clouds from the sensor to our system. The sensor's output including positional information together with radial velocity and signal intensity data is being temporarily placed in an on-chip buffer. The rest of the system can extract this information through a UART port. In our system we use the API provided by [13] to communicate with the buffer and handle cases like packet loss. We have, furthermore, adjusted the system's reading frequency through timing functions, in order to avoid overflows in the buffer.

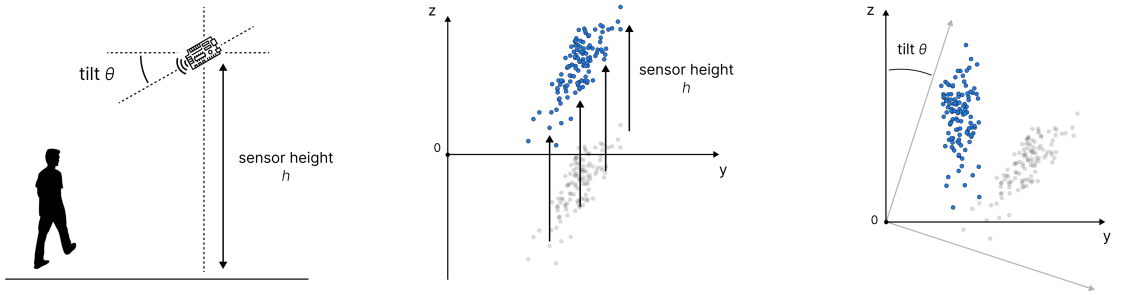
3.2.4. Point cloud data Post-Processing

The final step of the interface processing module includes some layers of data processing to format the point-cloud data into appropriate state vectors. The post-processing chain begins with decomposing the radial ve-

locity $\dot{r}$ into Cartesian velocities v_x, v_y and v_z. The pointcloud's default format is $[x\ y\ z\ \dot{r}\ i]$, where x, y, z are the point coordinates, $\dot{r}$ the radial velocity and i the signal intensity. The tracking system will be operating only on Cartesian coordinates and thus the modules output needs to be formatted accordingly. Applying the Equation 3.1 for all axes, the module's output formatting will be $[x\ y\ z\ v_x\ v_y\ v_z\ \dot{r}\ i]$, including sufficient information for all the next sub-systems.

$$v_i = \dot{r} \cdot \frac{i}{\sqrt{x^2 + y^2 + z^2}} \quad \text{for} \quad i = \{x, y, z\} \tag{3.1}$$

The next stage in the post-processing chain includes translating and performing axis normalization (rotation) to the pointcloud's positional and velocity information. The pointcloud's Cartesian coordinates are representing the scene with the sensor as the origin point. Scene dependencies like the sensor's height and tilt should be removed to utilize a ground-referenced coordinate system which will provide ease of use for the next subsystems. Figure 3.2 provides a comprehensive representation of the points' translation and axis systems normalization.



(a) Radar Setup (b) Point Cloud Translation (c) Axes System Normalization

Figure 3.2: Removing scene constraints from pointcloud data

To remove scene dependencies from the pointcloud data, we translate and rotate the axis system associated with it. We utilize a Translation Matrix T to account for the sensor height offset and a Rotation Matrix R_{inv} to rotate (roll) the axis system over the x-axis. If h_{sensor} is the sensor's height and t_{sensor} is the tilt of the sensor in radians, the new coordinates and velocities of the system are derived in Equations 3.4 and 3.5 respectively. The normalization results will be arrays of 4 elements, where only the first 3 elements represent the normalized coordinates and velocities.

$$\text{Translation Matrix } (T) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & h_{sensor} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$\text{Rotation Matrix } (R_{inv}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(t_{sensor}) & -\sin(t_{sensor}) & 0 \\ 0 & \sin(t_{sensor}) & \cos(t_{sensor}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$\text{Normalized Coordinates } (\text{coords}_{norm}) = T \cdot R_{\text{inv}} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.4}$$

$$\text{Normalized Velocities } (\text{vel}_{norm}) = T \cdot R_{\text{inv}} \cdot \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix} \tag{3.5}$$

3.3. Tracking Module

The tracking module is one of the most important parts of the system as it is responsible for detecting and tracking the position of all people in the scene. It receives the state vectors formatted by the interface processing stage and returns a list of all targets with their attributes. The implementation of the tracking system begins by integrating the GTRACK algorithm to the system requirements introduced by the IWR1443 sensor and then continues to improve certain aspects of it considering the application use case.

3.3.1. GTRACK overview

GTRACK is a system that follows a standard high level tracking approach shown in Figure 3.3. It tracks already seen targets through gate association and detects new targets through basic DBSCAN clustering over the remaining points. It goes on to update an Extended Kalman Filter to deduct the tracks' true position and handles unseen tracks through a maintenance step. Finally, it predicts the targets' next position through a constant acceleration motion model. These predictions are used over the next frame to drive the EKF's output.

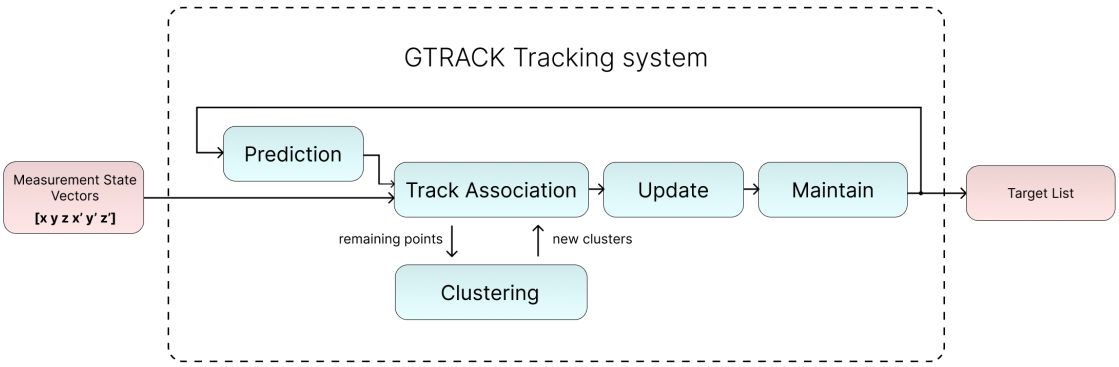


Figure 3.3: Overview of the GTRACK system pipeline

3.3.2. GTRACK adaptation

Transferring to the IWR1443

GTRACK was originally designed as an embedded demo application over the xWR68xx series of mmWave sensors. These provide higher pointcloud resolution and contain an on-chip DSP board at the expense of processing speed and power consumption. Their original data processing path also varies from the one found on IWR1443 which translates the spherical point coordinates to Cartesian. As a result, the TLV (Type Length Value) items [1] exchanged through the UART port are also inherently different. For the above reasons, adjustments focused on GTRACK's input formatting and data sparsity should be made.

Modifying the GTRACK to accept Cartesian coordinates as input, allows for a major simplification over its Kalman filter. The purpose of an Extended Kalman Filter is to approximate a non-linear relationship between the filter's measurement inputs and internal state vectors through a linear function. This non linearity in the case of GTRACK appears due to the spherical-to-Cartesian positional transformations as well as the radial velocity decomposition. In our approach, these transformations have already been performed during the on-chip data processing and the interface post-processing. We can thus simplify the system by using a classic Kalman filter, reducing the computational intensity and the system's complexity.

Redefining the Measurement Matrix

In our system, the input measurement and the state vectors of the Kalman filter share similar values. The measurement matrix H (more in Section A.2) can thus be simplified. It can take the form of an identity matrix for all the measurement data that remains identical to the state vectors. In our case, the input measurement vector $u(n)$ has the format of six values $[\, x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\,]$, which is extracted from the interface module's output $[\, x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \dot{r}\ i\,]$. The state vector $s(n)$ is also defined in Cartesian coordinates and for a constant acceleration

[1] TLV items allow for optional information to be exchanged in data communication protocols.

filter it also includes the target's acceleration data, resulting in a vector with 9 attributes: [$x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \ddot{x}\ \ddot{y}\ \ddot{z}$]. Equation 3.6 shows the relationship between the input and the state vectors through the measurement matrix H and through the interference of measurement noise v. The new measurement matrix for the Kalman filter has dimensions 6x9 and is:

$$u(n) = H(s(n)) + v(n) \tag{3.6}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.7}$$

Choosing a Motion Model

The state vector is dependent on the type of motion model used and the choice of that model is associated to the nature of the problem we try to solve. In our application, we expect people to move in arbitrary tracks with different speeds and accelerations. They might even stay still in an attempt to outsmart the system's fading squares blocking their sight.

Considering this, we choose the constant acceleration model to better approximate the human movements. The constant acceleration model will converge faster during changes in velocity and avoid lagging behaviour that would otherwise appear through a constant velocity model. A comparative analysis of the constant velocity and constant acceleration motion models is performed in Section 6.3.1.

In order to incorporate the motion model to the Kalman filter, we need to define the respective State Transition Matrix F (more on Section A.2). The State Transition matrix constitutes the prediction $\hat{s}$ for the state at time n using the state of time $n-1$ and we can calculate it through the state transition kinematic equations:

$$\hat{x} = 1x + 0y + 0z + \Delta t\dot{x} + 0\dot{y} + 0\dot{z} + 0.5\Delta t^2\ddot{x} + 0\ddot{y} + 0\ddot{z} \tag{3.8}$$

$$\hat{\dot{x}} = 0x + 0y + 0z + 1\dot{x} + 0\dot{y} + 0\dot{z} + \Delta t\ddot{x} + 0\ddot{y} + 0\ddot{z} \tag{3.9}$$

$$\hat{\ddot{x}} = 0x + 0y + 0z + 0\dot{x} + 0\dot{y} + 0\dot{z} + 1\ddot{x} + 0\ddot{y} + 0\ddot{z} \tag{3.10}$$

The same equations apply for both the y and z axis predictions and thus, transforming them into matrix-vector format we can derive the State Transition Matrix F in equation 3.11.

$$\hat{s}(n) = Fs(n-1) + w(n) => \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \hat{\dot{x}} \\ \hat{\dot{y}} \\ \hat{\dot{z}} \\ \hat{\ddot{x}} \\ \hat{\ddot{y}} \\ \hat{\ddot{z}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} + w(n) \tag{3.11}$$

Fine-tuning the filter

The final tuning of the Kalman filter requires the definition of the noise matrices in the system. In Equation 3.11 we add a vector of process noise w which has a 9x9 covariance matrix $Q(n)$. This noise represents the uncertainty of the motion model, in other words, it provides an approximation of how much the true motion will deviate from its prediction. In our solution we estimate this noise through a discrete constant white noise model with a set variance. Moreover, the noise introduced by the sensor itself during the detection phase (Equation 3.6) is called measurement noise v and it has a covariance matrix $R(n)$. In our filter implementation, $R(n)$ has dimensions 6x6, corresponding to the measurement vector sizing. We assume independent noise between the measurement vector's variables and as such, we calculate $R(n)$ by scaling the identity matrix with a standard deviation set value. Both the variances from matrices Q and R are empirically chosen.

3.4. Posture Estimation Module

The third basic subsystem of the proposed solution is that of posture estimation. Some of the major contributions of this thesis lie in the way the tracking and posture estimation modules are structured to compliment each other, as well as in the posture estimation model itself. The baseline model architecture will be the one proposed by MARS [5], operating on raw pointcloud data following a regression-based approach.

3.4.1. MARS model overview

MARS utilizes a unique regression approach to pass the raw track pointcloud data as input into the first CNN layer. It follows a preprocessing method which sorts the point vectors according to their spatial coordinates. It first sorts the vectors over the x-axis coordinate values and in the case where points share the same x-axis value, it proceeds to sort these by their y and z-axis coordinates. To maintain a constant input size, it adjusts the pointcloud data vectors to 64, either randomly removing points if they exceed the threshold or padding zeros to reach it. Finally, it reshapes the 64-entry array into an 8x8 matrix which is used as the input to the CNN.

The architecture of MARS maintains a straightforward structure, following a sequential execution of layers. It first passes the preprocessed 8x8 matrix input through two stages of CNN with 16 and 32 output channels and intermediate dropout layers. It then applies Batch Normalization (BN) and flattens the CNN's output. The flattened vector goes through two Fully Connected (FC) layers, reducing the vector's size to 512 and 57 neurons respectively. Each FC layer also includes intermediate dropout layers and Batch Normalization to avoid internal covariate shift and achieve more robust training. The model's output corresponds to the estimated 3D coordinates of 19 human joints with format [x_1, x_2, ... x_n, y_1, y_2, ... y_n, z_1, z_2, ... z_n]. The total number of parameters in the model is 1,095,115 which is approximately half of the ones appearing in the mmPose model, further emphasizing its computational and performance gains.

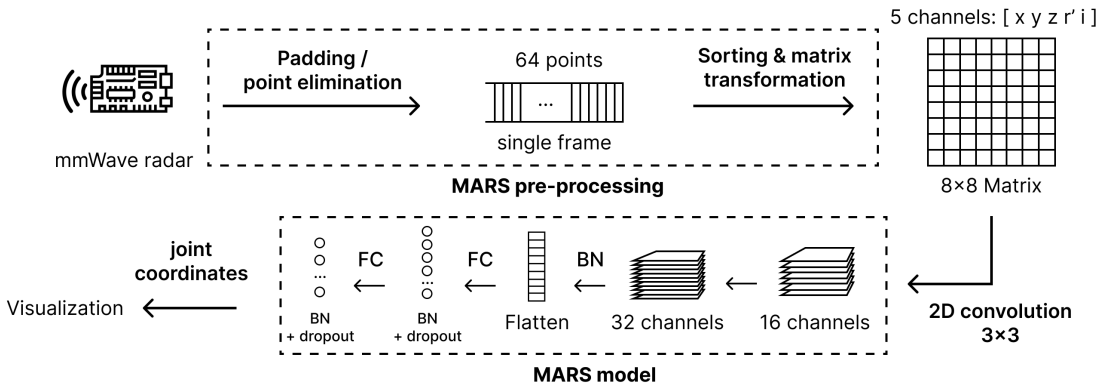
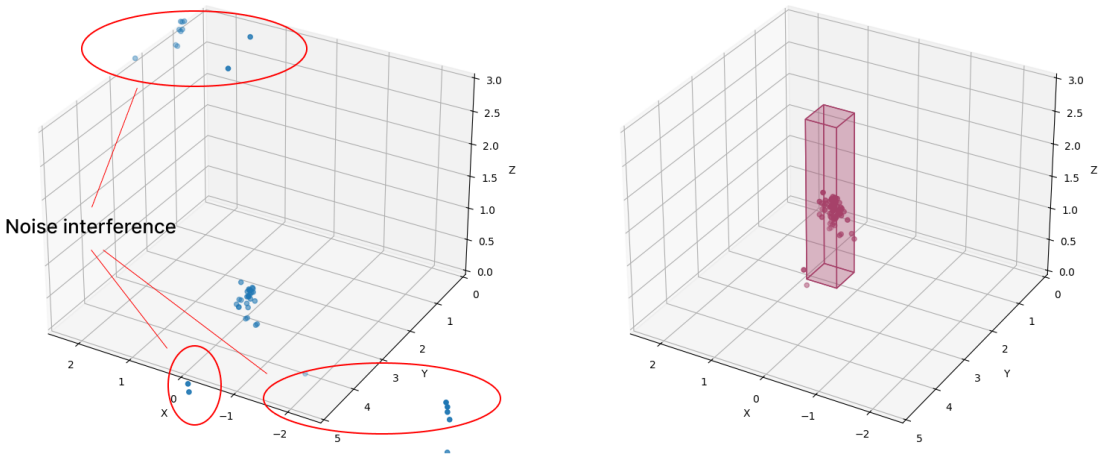


Figure 3.4: MARS pointcloud pre-processing and CNN architecture

3.4.2. Combining MARS and Tracking module

For the combination of the tracking and posture estimation systems, we follow the top-down approach. We perform posture estimation by creating point clusters for each track and then iteratively applying the MARS model to each of those clusters. Unlike the standalone MARS, which operates on the full point cloud, the integrated MARS model focuses on pointclouds for each individual track.

Isolating track point clouds improves the baseline model's accuracy by filtering out noise. In the baseline MARS approach, the model processes the full point cloud, including environmental interference and multipath noise. By using the tracking system, we can isolate surrounding noise from the active track, effectively filtering the model's input and enhancing accuracy, as shown in Figure 3.5. Additionally, combining the model with the tracking system enables posture estimation for multiple people. The top-down approach allows for simple training by abstracting the scene to a single person, enabling the same model architecture to be reused for an arbitrary number of targets and use single-person dataset samples to train the model.



(a) Raw pointcloud with noise interference

(b) Filtered scene after post-processing and tracking

Figure 3.5: Representation of the noise filtering introduced by the tracking system.

3.5. Visualization Module

The visualization module handles the logic of creating the fading blocks that is critical for this application. It interacts with the smart screen as the end node of the system and handles parameters like the number of fading squares, their position and size. To operate properly, it requires sufficient information over the scene setup and the location of the eyes of all targets in the scene.

3.5.1. Configuring the scene

In order for the system to adapt to new setups and environments, we have featured a configuration file including all the required scene dependencies. The system needs these configurations to create spatial relations between the individual components of the scene; the sensor, the smart screen and the sensitive object. The scene configuration parameters are listed in Table 3.2 and a comprehensive representation of the scene is provided in Figure 3.7.

Table 3.2: List of scene configuration parameters.

Scene parameters	Description
h_{sensor}	Sensor installation height (m)
t_{sensor}	Sensor down tilt (degrees)
h_{screen}	Smart screen installation height (m)
$size_{screen}$	Smart screen height & width (m)
x_o, y_o, z_o	Sensitive objects coordinates (m)

3.5.2. Projecting the fading squares

The posture estimation module outputs the location estimates of every human joint. Using this information, the visualization module combines the head coordinates with the scene parameters to calculate each person's gaze vector. This gaze vector represents the line extending from an individual's eyes to the sensitive object behind the smart window. The visualization module then employs trigonometric equations to determine the points where these gaze vectors intersect with the smart window. These points will serve as the centers of the projected fading squares.

The visualization module also handles the sizing of the fading squares to ensure privacy. To address the risk of inaccurate square positioning when a target is too close to the window, a responsive sizing system has been developed. While we aim to keep the original size of the squares as small as possible for practicality and overall solution elegance, this system adjusts the size of the squares based on the target's distance from the sensor. The closer the target is to the window, the larger the squares become. We set a minimum $size_{min}$ and

maximum $size_{max}$ threshold for the size of the squares, a distance threshold d_{min} that defines the furthest distance where we require the maximum square size and a size reduction rate β, ensuring a smooth size change for the squares. If y_{head} is the y-coordinate of the head joint, the square size is derived by Equation 3.12. This approach limits the square size between the two upper and lower thresholds as shown in Figure 3.6. The constant values chosen for our application will be discussed later, in Chapter 6.

$$size_{rect} = \max \begin{cases} size_{min} \\ \min \begin{cases} size_{max} \\ size_{max} - \beta * (y_{head} - d_{min}) \end{cases} \end{cases} \tag{3.12}$$

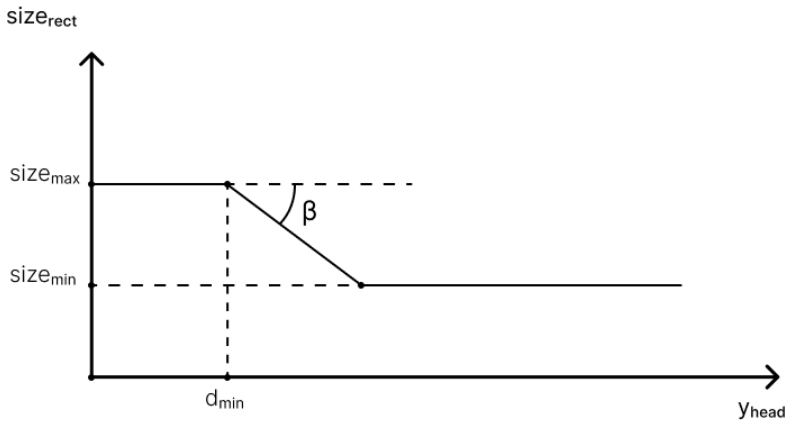


Figure 3.6: Square size over a target's distance from the sensor. The square sizing is always limited between $size_{min}$ and $size_{max}$.

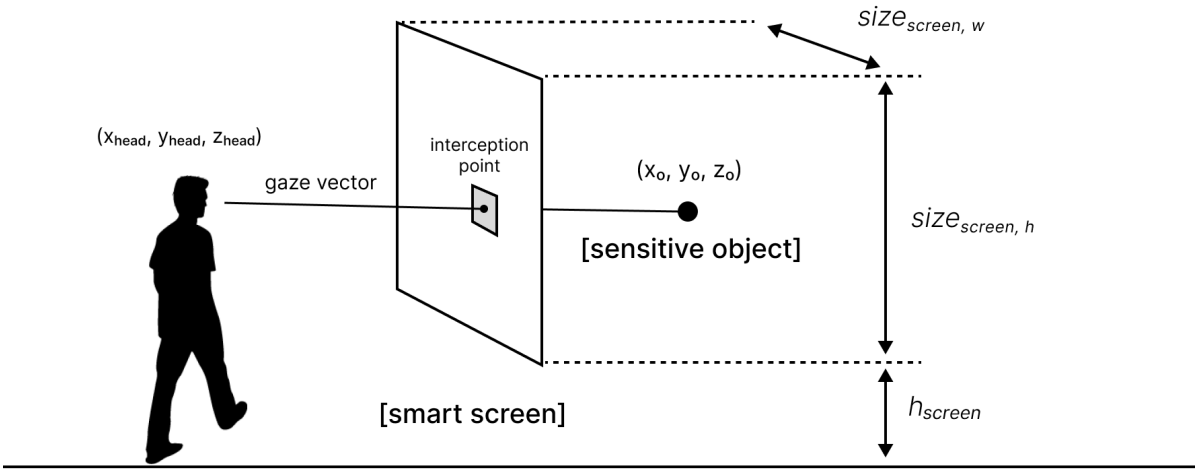


Figure 3.7: Representation of the gaze vector and the scene configuration parameters involving the sensitive object and the smart window

4 | System Improvements

In this section we demonstrate the improvements made over the baseline system, developed during the previous chapter. The improvements made are targeting the clustering logic of GTRACK, the localization error introduced by the MARS model and the posture estimation learning capabilities.

4.1. Improving the clustering logic

The first improvement we performed addresses the target detection capability of the GTRACK algorithm. GTRACK utilizes a clustering algorithm almost identical to the DBSCAN, with the only difference that the cluster creation is not only dependent on spatial constraints, but also on signal intensity and radial velocity thresholds. We will attempt to increase the effectiveness of DBSCAN by making modifications to the distance metric, and therefore, redefining the clustering neighbourhood of points.

One of the most commonly used metrics for measuring the spatial proximity in DBSCAN is the Euclidean distance. The Euclidean distance is defined as the square root of the the sum of squared differences between corresponding elements from each axis. In a 3D coordinate system, the Euclidean distance formula between two points A, and B is:

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2} \tag{4.1}$$

Differences along different axes impact the Euclidean distance metric equally and result in a spherical geometric locus when a maximum threshold is set. This sphere represents the clustering neighbourhood of the central point, in other words, the DBSCAN algorithm counts all points within this sphere to determine if a cluster should be initialized. However, in our intended application where the interest is people detection, the sensed pointclouds will rarely resemble spherical clusters. Instead, using a distance metric specialized in detecting clusters exhibiting characteristics similar to an average human body, would be a better suited approach.

The average human silhouette generally appears more compact in the horizontal plane, while the height proportion appears noticeably larger. In that line, we will follow the approach of [51] and alter the Euclidean distance to represent a vertically stretched ellipsoid geometric locus. We can achieve that by introducing a diminishing weight z_{weight} over the z-axis' contribution, where $z_{weight} \in (0,1)$. All the constant values we choose for our application will be presented in Chapter 6.

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + z_{weight}(z_A - z_B)^2} \tag{4.2}$$

Another challenge that the Euclidean clustering metric fails to address is the effect of a pointcloud's range on its density. As a target moves further from the sensor, the sensor's resolution decreases, resulting, in a proportionally sparser pointcloud [6]. An adaptive metric is necessary to allow for more relaxed constraints when the combined range of two points gets larger. We introduce the damping factor $\alpha(y_\mu)$ to adjust the clustering neighbourhood taking into consideration the mean range value y_μ of the two points. $\alpha(y_\mu)$ is limited at the range (0,1) and the further the two points are from the sensor, the smaller its value, allowing for a larger point neighbourhood. The derived distance metric is presented in Equation 4.3. Finally, a visual comparison between the three described metrics are shown in Figure 4.1.

$$d = \alpha(y_\mu)\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + z_{weight}(z_A - z_B)^2} \tag{4.3}$$

4.2. Position vector elimination

The second improvement aims to keep the localization and posture estimation functionalities clearly separated. To explain the improvement we provided, we first need to identify the errors in the system. The system's absolute error is a combination of two distinct factors:

- **Localization error** ϵ_l: This is the error between the estimated and actual skeletal position, measured as a difference between the estimated and true position of the spine base of a person. In other words, when the spine base between the prediction and ground-truth is aligned, that is assuming ϵ_l to be zero.

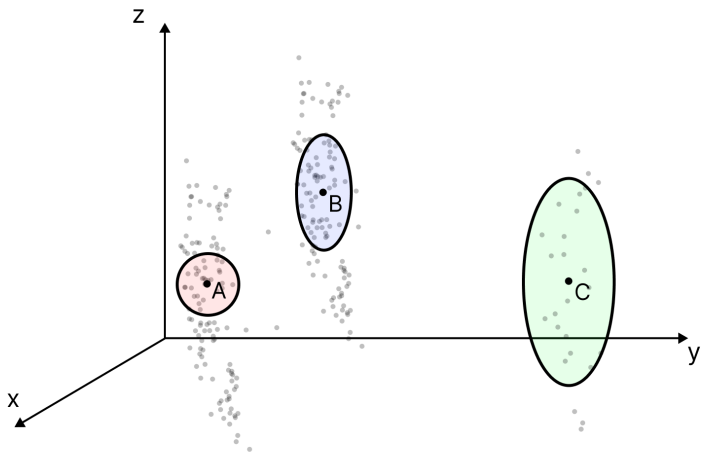


Figure 4.1: Representation of clustering neighborhoods of different points. The neighbourhood of point A is acquired through the classic Euclidean distance metric, the neighbourhood of point B through the Euclidean distance with diminishing weight on the z-axis and the neighbourhood of point C through the addition of a range-based damping factor over the metric of B. The neighbourhood of C is larger than the one from B because $y_C > y_B$. **NOTE:** A clustering neighborhood differs from its final cluster. Once a cluster is initiated, it expands by iteratively including all adjacent points found within the neighborhoods of any associated points. Therefore, the initial clustering neighborhood may be just a small subset of the final cluster.

- **Posture estimation error** ϵ_p: This error captures the difference between the true and estimated posture of a person. total deviation of the joint localization estimates when the spine bases are aligned.

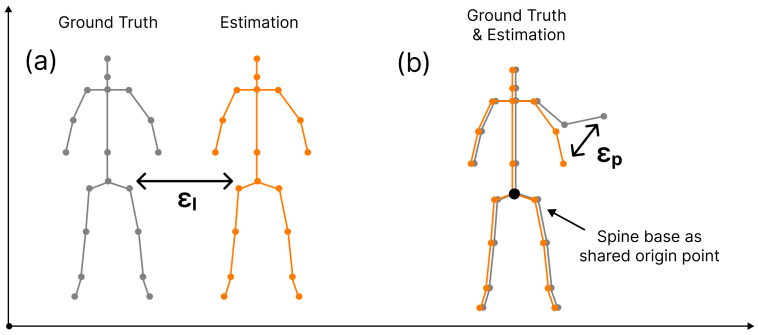


Figure 4.2: Representation of (a) localization error and (b) posture estimation error

As per the original work, MARS did not differentiate between the two errors as it was designed for operating on static movements. Thus, in our baseline design, despite localizing the tracks through the tracking system, MARS would still attempt to re-position their estimated posture in space. This unnecessary repositioning incorrectly attributes the localization error ϵ_l to MARS rather than the dedicated tracking algorithm, GTRACK.

Localization through MARS has a lot of limitations. First, it highly depends on the model's prior training. If MARS has not been trained for certain scenarios or locations, there is a risk it might not generalize properly and diminish the system's performance. Secondly, it does not handle cases of occlusion and temporal signal loss as well. The model's architecture lacks memory and would fail to operate in these cases, compared to the tracking system which would continue to perform localization through the motion prediction model.

Our proposed approach tackles this problem by clearly separating between the systems that perform localization and posture estimation. The localization error is attributed solely to the sensor resolution and the GTRACK system, while MARS is only responsible for the posture estimation error. We achieve this by repositioning the center of mass of every pointcloud to the origin, and thus eliminating their position vector. As shown in Figure 4.3, we translate the track centroid to the coordinate system origin to remove any position

information. The translated data is then passed through the posture estimator to deduct only the human pose and finally the created human skeleton is re-positioned back to its original position.

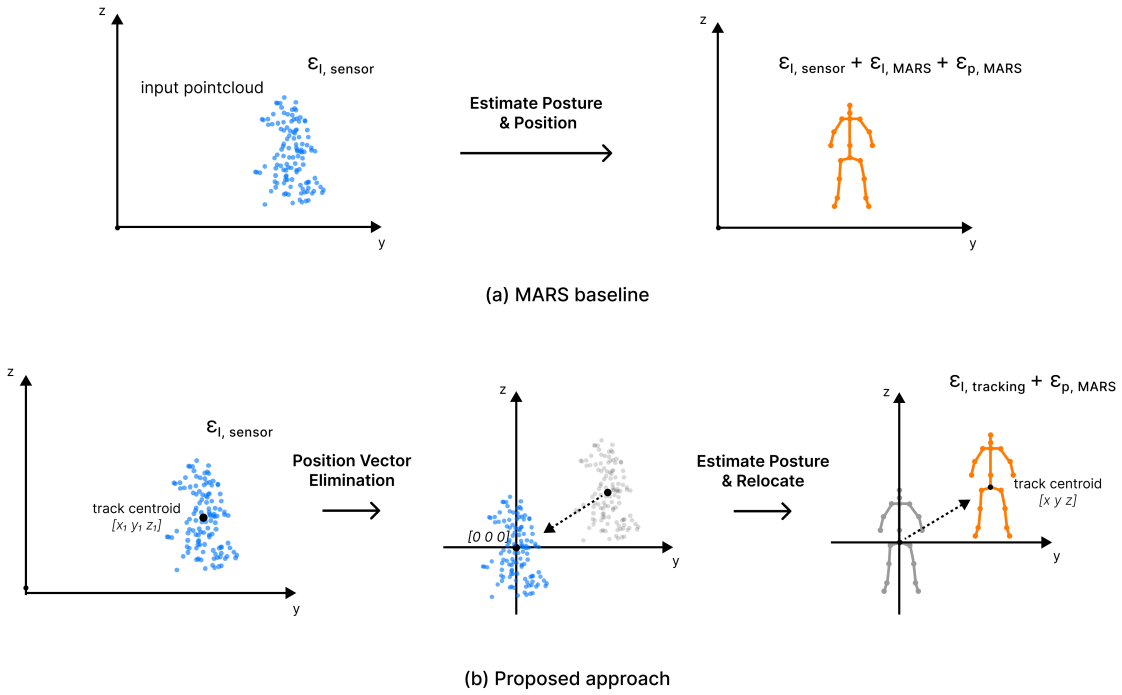


Figure 4.3: Comparison of the localization errors between the MARS baseline and our proposed approach. The proposed approach eliminates the position vector of the pointcloud before estimating its posture.

This method yields a number of benefits. A comparative analysis (Section 6.4.2) shows that by removing the localization error factor $\epsilon_{l,MARS}$, this method provides better accuracy. It also eliminates the need of exhaustive training, as the model does not need to generalize to unforeseen locations. Finally, the model's decision complexity is reduced by having the clear objective of sole posture estimation.

4.3. Creating temporal connections

The third and final improvement we implemented on the baseline system aims to:

- Address the sparsity problem of the mmWave pointcloud by efficiently combining frames to improve signal density.
- Use these frames to improve the posture estimation accuracy and introduce temporal learning in the MARS model architecture.

Interestingly, the steps we took to achieve this improvement had positive impact in other parts of the system, such as the clustering and the track maintenance components of GTRACK.

4.3.1. Addressing the sparsity problem

To address pointcloud sparsity we will combine multiple consecutive frames. In our implementation we make use of two different frame combination methods; **frame fusion** and **frame stacking**. Frame fusion refers to the integration of multiple frames into a single frame entity. It yields the benefit of higher pointcloud density and given that the frames are observed in short time indices, it does not introduce significant noise. On the contrary, frame stacking handles multiple frames as an ensemble of layers, preserving their distinct characteristics. This approach can be efficiently used in combination with a Deep Neural Network (DNN) capable of learning temporal associations. We will later see that we utilize frame fusion for improved clustering, while we perform frame stacking over the CNN model's input for improved posture estimation.

Our proposed solution suggests a Ringbuffer class to temporally store the new frames while overwriting the outdated ones. The class is equipped with methods to perform both frame fusion and frame stacking over the buffer's effective data and a method to adjust the inner buffer's size. The frames in the Ringbuffer are being stored as separate entities and are only combined (fused or stacked) upon retrieval by other fuctions. Given our system's sensing frequency (10fps), the optimal number of frames to combine is 3 [6] to avoid introducing significant noise due to human movement. Every detected track owns their respective Ringbuffer instance, while a universal Ringbuffer for the unassigned points of each frame is also being used.

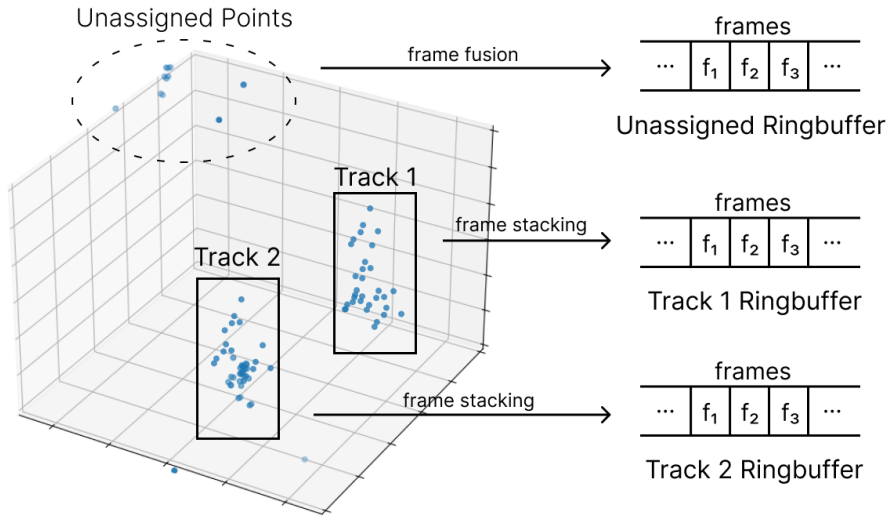


Figure 4.4: Representation of different Ringbuffers updated after Tracking

Increasing density for clustering performance
The system's clustering performance is positively affected by the Ringbuffers, as performing **frame fusion** over the unassigned points from each frame can be quite beneficial. The accuracy of mmWave is high, so the targets in the scene will continuously spawn points at approximately the same location through their presence. Even at larger ranges where the pointcloud tends to be sparser, merging those points from different frames should result in sufficiently dense clusters for the system to detect. Furthermore, noise interference which tends to appear at random, does not accumulate into clusters and thus is filtered out.

Maintaining stationary tracks
mmWave pointcloud density is significantly diminished during the observation of static points. To address this problem we have included a lifetime scheme allowing the tracks to remain undetected for short periods of time without being discarded as inactive. Each track is being labeled as STATIC or DYNAMIC depending on the absolute velocity of their centroid and they are assigned a maximum lifetime accordingly. The lifetime assigned to STATIC tracks is typically larger than that of the DYNAMIC targets to allow for greater detection probability. Furthermore, the size of the Ringbuffer associated with STATIC tracks increases too, allowing more frames to merge and eventually leading to a more substantial pointcloud scene. The detailed pointcloud can be later used for recognizing static human postures.

4.3.2. Learning temporal constraints

FUSE [6] was the first system that addressed the problem of signal sparsity through combining frames. It followed the method of frame fusion to merge frames, acquire a pointcloud with more distinct human characteristics and pass it into the CNN model for posture estimation. In our approach we follow the frame stacking approach instead, in order to maintain the order of the consecutive frames and learn from their temporal connections.

The critical part of this improvement modifies the architecture of MARS. It aims to use the frames in the track Ringbuffers to train and learn temporal associations amongst them. As such, the system extracts the

three latest frames of the track's Ringbuffer and performs **frame stacking**. After removing the position vector (Section 4.2), we limit or pad the stacked frames to 64 points each and reshape the resulting arrays into a 8x8x3 vortex. The vortex will eventually serve as the model's new input with 5 channels [x y z $\dot{r}$ i].

The architecture changes involve primarily the convolutional layers of the CNN. In order to learn the temporal associations between the stacked frames, a sliding 3D kernel of dimensions 3x3x3 is utilized at each convolutional layer. The kernel's two first dimensions refer to the spatial characteristics of the frames while the third is used to extract temporal connections from the data. The convolutional layer output channel dimensions are kept at 16 and 32 in sequential order, while the output size of the flattening layer and the first Fully Connected (FC) layer is tripled to handle the data of all frames. A full overview of the posture estimation system including the 3D kernel modifications on MARS is provided in Figure 4.5

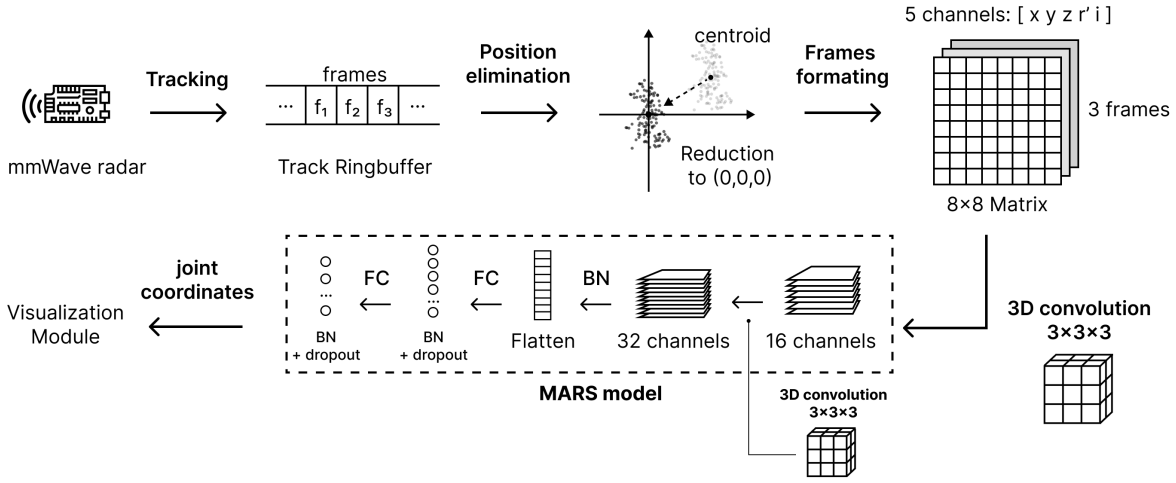


Figure 4.5: Track multi-frame pointcloud pre-processing and MARS architecture with 3D kernels for temporal connections

Finally, we want to emphasize this improvement's contribution in the bigger picture. In our intended application it significantly improves the consistency of blocking people's gaze, without interruptions. The instantaneous sparsity and irregular shape of the mmWave point cloud were issues that the baseline MARS, operating on a single frame, could not address and led to large variations between consecutive predictions. Our proposed solution links information from consecutive frames, making it resilient to such irregularities.

5 | Introducing a new dataset

A big part of this project is the creation of a new dataset that addresses the challenge of real time posture estimation over dynamic targets. So far, HuPR [28] has been the only publicly available dataset that included a noticeable share of dynamic movements, yet, the heat-map representation of the mmWave data hinders its usability for real time applications. By maintaining the raw pointclouds, we propose a novel dataset targeted for real time applications, with sufficient size and diverse samples of highly dynamic targets performing weakly supervised movements.

5.1. Dataset Overview

5.1.1. Hardware and Devices

For the pointcloud collection we use the IWR1443 mmWave sensor with the configurations provided in Section 3.2.2 to capture the pointcloud data. The mmWave sensor is connected to a Linux operating laptop through the UART port, where the data is processed and stored. The joint location ground truth of the system is captured by a Microsoft Kinect v2 sensor, a device able to capture color and depth information through an RGB and a laser-based Time-of-Flight (ToF) sensor (more on section 5.2.2). The Kinect is attached to a second laptop running on a Windows operating system, providing support for the sensor.

5.1.2. Participants and experimental process

The dataset creation process involved 16 volunteering participants, where 10 of them were males and 6 females. The participants represented different body types, heights and displayed diverse clothing and accessories. The mean height among the participants was 177 cm, with a standard deviation of 8.6 cm, while the total range of heights was 35 cm, indicating sufficient variation to avoid overfitting. Every participant had voluntarily consented to contribute towards data collection by signing a personal data consent form approved by the TU Delft Ethics committee.

Since the data is meant for a top-down MPPE application, the dataset's scope is limited to capturing **one person at a time**, in the premise that the trained model can be iteratively used to estimate the postures of all targets in the scene. Moreover, the dataset aims to capture people at different ranges and orientations to include pointclouds of different sparsity and body part occlusions.

Towards this, each participant was asked to participate alone in 5 experiments, which represented different sets of movements. Each experiment lasted for approximately 2 - 2.5 minutes, and the mean duration for completing all the experiments between the participants was approximately 12 minutes. Before every experiment, the participants received a brief explanation over the details entailing the area of operation and the desired movements.

The five experiments are divided into distinct movement sets:

- **Static Waving:** Participants performed static waving with the right arm, left arm, and both arms.
- **Normal Walking:** Participants walked continuously in various directions relative to the sensor—parallel, perpendicular, and along two diagonal lines.
- **Combined Walking and Waving:** Participants walked in the same directions as the previous experiment while waving their arms.
- **Static Movements:** Participants performed static movements, including leaning to the right, leaning to the left, and leaning forward into a squat.
- **Free Movement:** Participants had a 2-minute window to move freely in space, combining elements from the previous experiments. The specific combination of movements was left to the participants' judgment.

5.1.3. Environment setup

In order to capture data from people at different ranges and orientations we set an environment with 9 designated markers on the floor. We asked the participants to navigate between those markers when walking,

(a) Experimental scene defined by markers

(b) mmWave and Kinect v2

(c) Top view of experiment room

Figure 5.1: Environment setup

while all the static movements were repeated over all markers for the completion of the experiments. Specifically for the static movements, the participants were also randomly asked to face at different orientations for a better capture of different orientations.

The Kinect sensor's limited field of view (FOV) defined the setup of the rest of the experimental environment. We placed the Kinect sensor at the height of 1m with no tilt and after performing some FOV analysis we decided that the optimal range for the closest marker would be at 2m, in order to fully capture the silhouette of people in the scene. Given that the Kinect has a maximum detection range at 4.5-5 meters, we spread the rest of the markers accordingly. The mmWave sensor was mounted at the height of 165cm with a down-tilt of 5 degrees and was positioned at the same vertical plane as the Kinect, with a small offset of 5cm in the x-axis which will be accounted for in the dataset pre-processing stage. The environment setup is clearly visualized in Figure 5.1.

5.2. Capturing the Dataset

The data gathering process uses the mmWave and Kinect sensors only as front-end devices. The observed data from the two sources had to be synchronized and saved in a way that would be useful for later processing. In this section we will explain all the methods and techniques used to gather the data for the dataset creation.

5.2.1. Logging mmWave frames

In order to save the observed mmWave frame pointclouds, we created a lightweight frame logging module for offline processing. The logging system consists of two main threads for reading and writing. The reading thread has the task of reading the latest observed pointcloud. It periodically retrieves the data from the UART port through the use of the API and publishes it into an internally shared queue. The writing thread reads the contents of the queue and when the frames exceed a certain number, it writes the data into a *.csv* file.

Through that approach, the logging system is able to maintain a consistent reading frequency equal to the radar's sensing frequency, which is 10fps, and introduce minimal jitter through its computational delay. Additionally, a file manager ensures that the frame data of an experiment is split and saved in *.csv* files of relatively small, fixed size in order to avoid later parsing delays.

Each line in the saved files contains 7 attributes for a single point. The first attribute is the point's frame-number, which is shared across all the points of the same pointcloud. Then, the 5D point vector is listed, encapsulating the point's position, radial velocity and signal intensity information. The last attribute contains the timestamp from when the point was observed and follows the POSIX formatting in the scale of milliseconds[1]. This timestamp will later be used to couple the pointclouds to their ground truth.

[1] A POSIX timestamp represents the number of seconds that have elapsed since the Unix epoch, which is defined as January 1st, 1970, at 00:00:00 UTC

5.2.2. Capturing and localizing joints through Kinect

Microsoft Kinect v2 [49] is a system that utilizes multiple sensors to capture information. To estimate depth it uses a Time of Flight system comprised of three lasers and a monochrome camera with resolution of 512×424 pixels and FOV of 70°x 60°. It can extract spatial information of the scene either through the monochrome channel or through an equipped RGB camera. For the experiments we will only utilize the monochrome camera.

The Kinect v2 poses as a cheap, yet accurate solution for capturing the ground truth of human posture. Compared to its predecessor, the Kinect v1, the newer version appears to be more robust in diverse lighting conditions and its accuracy is almost double in applications like object identification [27]. The sensor comes together with an SDK containing tools to visualize and perform accurate real-time posture estimation for 25 human joints in the scene, through a computer vision model.

For the ground truth logging we used the system developed by [42], which captures the depth stream of the sensor and extracts an accurate skeleton with the 3D locations of the multiple human joints. The system is run through the Visual Studio 2015 tool and saves the joint coordinates of each frame into a *.csv* file. We added a POSIX timestamp entry for each frame and removed the unnecessary joints' data to limit them to the 19 relevant ones for our application. The frame rate of this system is 30fps.

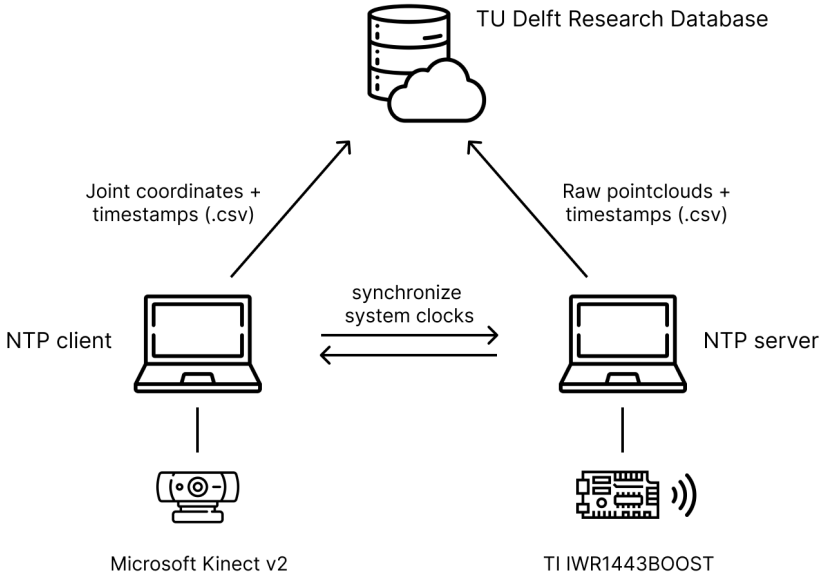


Figure 5.2: Experiment's data logging system

5.2.3. Devices synchronization

Synchronizing the device clocks and acquiring the frames' timestamp is important for coupling the point-cloud observations to their posture ground truth. We achieved the synchronization of the clocks of the two laptops through the use of the Network Time Protocol (NTP). We setup the laptop connected to the mmWave device to be a NTP server, broadcasting its system clock status and the one connected to the Kinect to function as an NTP client. We established a connection between the two and proceeded to synchronize the two clocks before every experiment session. Figure 5.2 shows the experiment logging setup and synchronization system between the hardware components of the system.

5.3. Dataset processing

To transform the unrelated files of point cloud data and keypoint ground truth into a comprehensive dataset, several steps are required. These steps include associating the frames, cleaning the data, and performing various preprocessing and formatting tasks. In this section, we will describe each of these stages the dataset goes through to become suitable for model training.

5.3.1. Data pairing

In order to pair the frames of the two sources we use the POSIX timestamps found in the logged files of both systems. We parse through all the frames and create 1-on-1 associations between the frames that have the smallest absolute time difference between the timestamps. We also set a maximum threshold of 20ms to prevent the connection of frames with larger time difference. This process outputs the synchronised pairs of frames and at the same time filters out the abundance of frames occurring due to the different sensing frequencies of the two systems.

5.3.2. Data pre-processing

Since the mmWave and the Kinect sensors did not capture the targets from the same perspective, the logged files of the two systems contain data that relate to coordinate systems with different origin points. Our pre-processing algorithm aligns both of the system axes and removes their spatial offset to correspond to the ground-referenced coordinate system.

Secondly, in order to bring the pointcloud to the appropriate format for the posture estimation model, the system needs to isolate the track associated pointclouds. For that it utilizes the tracking module of the proposed system pipeline. The tracking system detects the cluster associated to the person in the scene and tracks it, also removing excessive noise from the background. The tracking module's output should be a single track and its points are preserved as the only valid points of the frame.

Finally, the pre-processing system performs signal intensity normalization, which is essential for limiting the impact of arbitrarily large intensity values on the model's training and predictions. To achieve this, we first measured the mean and standard deviation of the signal intensity from a set covering the entire area of interest, providing a clear overview of the values' distribution. Using these two constants, we apply the normalization formula $i_{norm} = \frac{i - mean}{std}$ to limit most of the intensity values between (-1, 1).

5.3.3. Data cleaning

In order to ensure that all the frame pairs contain the desired information and not faulty measurements, we perform a stage of manual data cleaning. We remove the pairs of which the ground truth skeletal reconstruction shows anomalies or does not represent a human silhouette. The posture ground truth captured by Kinect appeared sensitive to the colors of the scene and resulted in errors when the clothing of the participants was of similar color to the walls or floor of the experiment room. Furthermore, we remove the pairs in which we observe clustering or tracking errors. In a few frames, the tracking module had misclassified noise as a human cluster which led to a completely irrelevant part of the pointcloud being preserved for training.

5.3.4. Data partitioning

After the abovementioned steps, the dataset includes 88,851 labeled frames, which need to be divided into training, validation, and testing sets. We split the dataset approximately into a 60/20/20 ratio, ensuring that individual participants are kept separate across the sets. This approach ensures that the testing and validation sets will include different individuals from the training set, which will help prevent overfitting, enhance the model's ability to generalize and eventually provide more representative testing results.

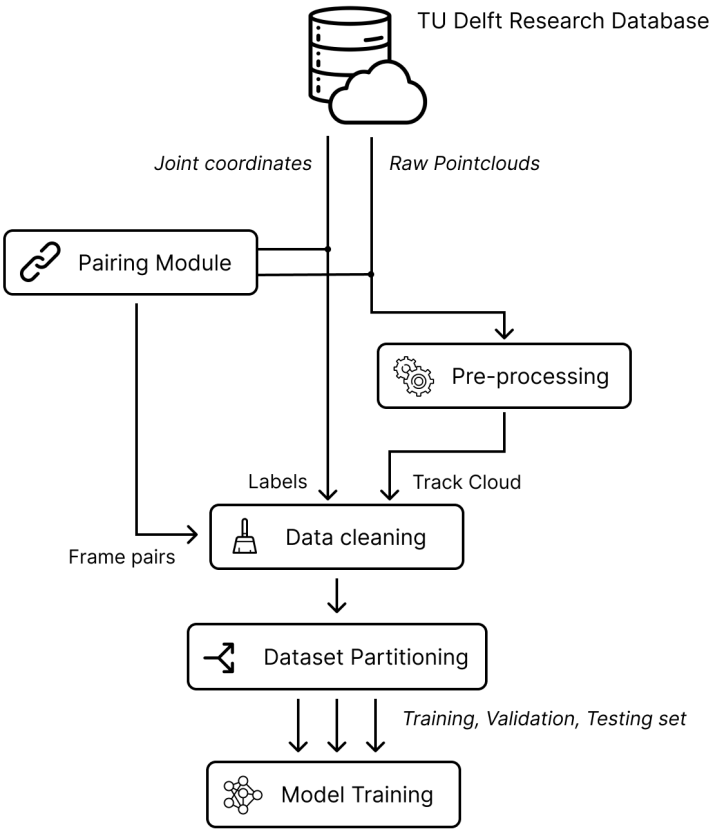


Figure 5.3: Overview of the dataset processing system

6 | Results

In this section we will discuss how the proposed solution is a better fit for the intended application compared to its baseline by performing analysis on results and metrics. We will also justify the use or rejection of specific methods and techniques (Section 6.3) and study their impact through thorough ablation studies (Section 6.4).

6.1. Training Tools and Details

For the creation of our system we used Python version 3.10 and implemented all the baseline approaches, the improvements and the ablation studies of the posture estimation model using Tensorflow and Keras versions 2.15.0. The quantitative analysis of the results was performed while using the proposed dataset for training and testing. For the experimentation we utilized the cloud-based resources provided by Google Colab Pro [14], taking advantage of the Nvidia A100 GPU.

We configured the tracking system by empirically optimizing its detection and tracking performance. For the DBSCAN algorithm we chose the eps to be 0.3, almost an order of magnitude higher than the sensor's unambiguous range detection and the minimum number of points to form a cluster at 40. We also used the the proposed altered Euclidean Distance as the distance metric and set its parameter z_{weight} to be 0.4. Finally, the Kalman standard deviations for the measurement and process noise are set at 0.1 and 1.

For the posture estimation training we adopted many of the parameters used for training the MARS CNN model. Specifically, we used the Adam optimizer with a learning rate of 0.001, a batch size of 128, and ran the training for 150 epochs. On average, the validation loss converged to 0.01 after the 130th epoch for the proposed model, and a little later for the MARS baseline. The function used to define validation loss of the system is the Mean Squared Error (MSE), Equation 6.1. We chose to validate our system through a metric where all 19 human joints have equal impact on the loss -and not applying more weight to the head which is our primary target- to achieve fair comparison between the MARS model and to provide a new general purpose benchmark for dynamic target posture estimation.

$$loss_{MSE} = \frac{\sum_{i=0}^{18}(x_i - \hat{x}_i)^2 + \sum_{i=0}^{18}(y_i - \hat{y}_i)^2 + \sum_{i=0}^{18}(z_i - \hat{z}_i)^2}{3 * 19} \tag{6.1}$$

To obtain representative results, we finally utilized a method similar to 10-fold cross validation. We performed 10 full training iterations for every baseline and improvement implementation, using 10 different instances of the dataset with the sets randomly split. The same 10 dataset split instances were used for every training case to ensure consistency. The combined loss is calculated as the average of all the iterations and the trained parameters achieving the highest accuracy are kept as the model's default.

6.2. System Evaluation

To thoroughly evaluate our proposed solution's performance, we followed an approach of two steps. In the first step we evaluated its posture estimation capabilities over a single dynamic target. We performed an offline analysis over the proposed dataset and compared the results against the baseline of MARS to have a clear overview of our system's improved modular performance. The second step focused on scaling the system's evaluation by testing the performance of its complete pipeline in an online, multi-person scenario.

6.2.1. Offline Evaluation over a Single Person

Following the top-down principle, our posture estimator is iterated over multiple targets in the scene, thus we can evaluate its performance only over a single one.

At a first glance, the proposed system, trained on the new dataset, has been able to identify multiple poses appearing in the proposed testing sets. Figure 6.1 visualizes the model's posture predictions in comparison to their ground truth, as well as it provides the track pointcloud serving as the input of the model.

To quantify the system's accuracy we calculate the Mean Average Error (MAE) (6.2) and Root Mean Squared Error (RMSE) (6.3), averaged across the entire testing set. We use the combination of the MAE and RMSE metrics to assess the variance between the errors of the frames within the set. The closer the values of RMSE and MAE are, the more consistent the magnitude of error between the samples and the lower the error of the

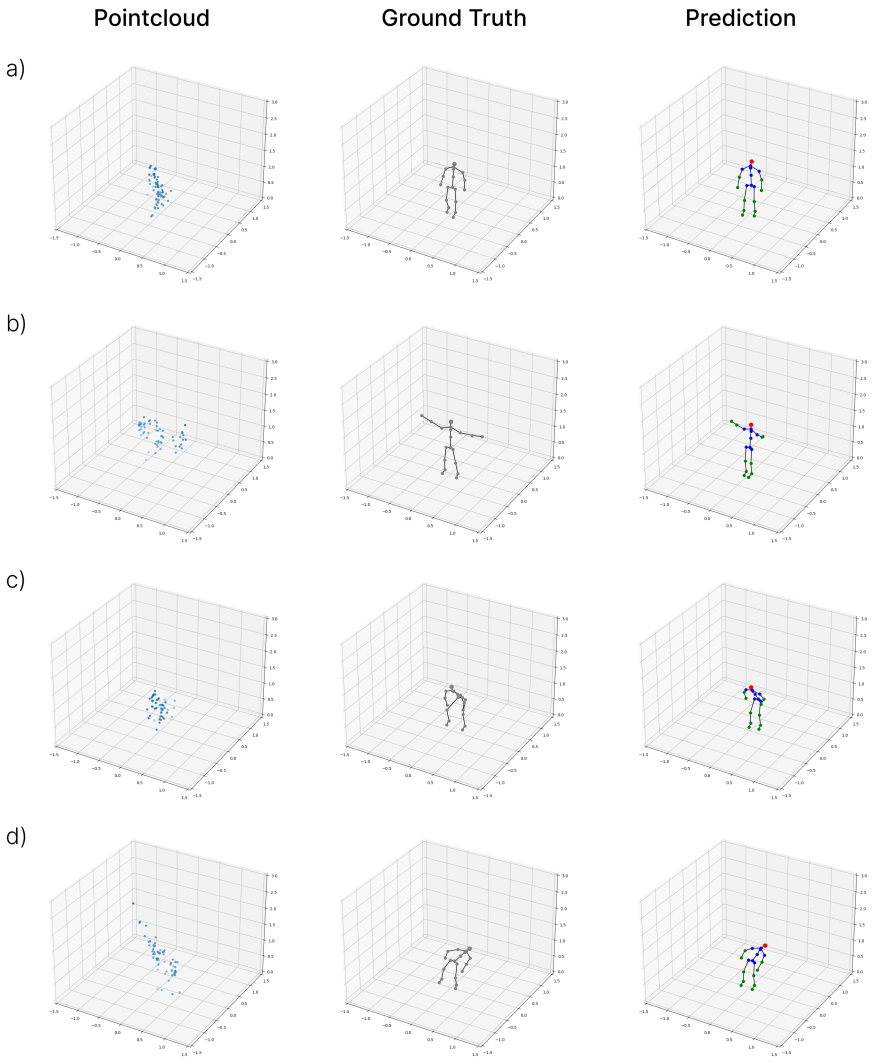


Figure 6.1: Final solution's pose prediction samples. The movements presented are a) walking, b) waving arms, c) squatting, d) leaning sideways

worst-case predictions. We especially care about keeping the MAE and RMSE values close, as this also indicates how "smoothly" the estimations of posture change over time. This smoothness can eventually translate in consistent movement of the fading blocks on the smart screen.

$$loss_{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{6.2}$$

$$loss_{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{6.3}$$

In Figure 6.3 we visualize the MAE error measured at every joint index, comparing the results of our proposed solution against the MARS baseline. The connection between the joint indexes and the unique joints are indicated in Figure 6.2. Finally, the overview of the average errors measured in both systems can be found in Table 6.1. A separate field focusing on the errors of the head joint in both approaches evaluates the system's ability to accurately estimate the location of the eyes which will then be used to block the target's line-of-sight.

Overall, our system outperforms the baseline by approximately 20%, achieving a mean average error of 13.1cm for all 19 joints and 13cm for the human joint representing the head. One significant improvement

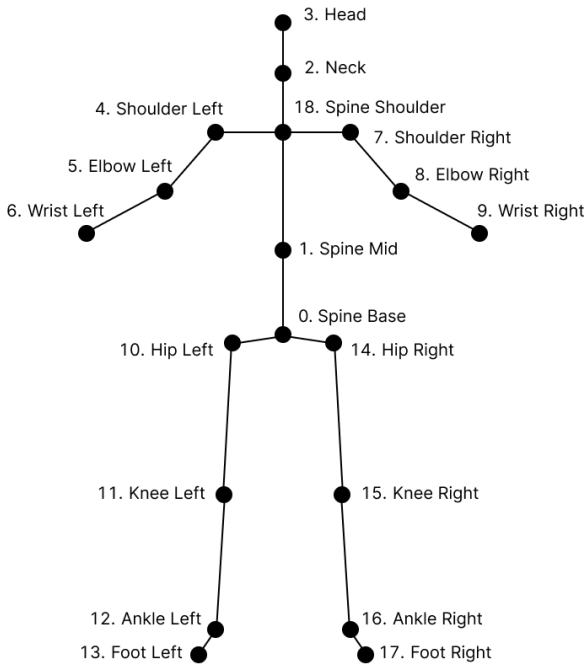


Figure 6.2: Skeleton diagram illustrating the mapping between indexes and corresponding joints

Table 6.1: Average MAE and RMSE error (in cm) for the combined 19 joints and the head joint. The analysis is over the baseline MARS model and the final proposed solution.

	All joints		Head	
	MAE	RMSE	MAE	RMSE
Baseline MARS	**16.43**	**251.87**	**17.26**	**279.32**
Our approach	**13.15**	**16.21**	**13.00**	**15.74**

is also noted in the RMSE error, with our system managing to reduce it by an order of magnitude, drastically decreasing the worst-case prediction error and ensuring consistency between the frames. Through the ablation study in section 6.4.1 we show that the baseline's poor RMSE performance is mainly caused due to noise interference, leading to large instantaneous errors in both localization and posture estimation.

6.2.2. Online Evaluation

After evaluating the posture estimation capabilities of the system over a single, dynamic target, we proceeded to test the whole pipeline with the smart screen as the system's end node, shielding the privacy of a sensitive object. To conclude into realistic results, we performed an online evaluation, in real time and in a different environment from the one the dataset was created in. In the evaluation experiments we had multiple people simultaneously at a scene larger than the 3m x 2m space we used at the dataset capture.

The goal of the evaluation was to determine the smart screen's successful visual blocking rate of a marker placed at 1m distance behind the smart screen. We performed experiments with volunteering participants who were instructed to move freely into space and try to see the marker which was hidden behind the fading squares. Every experiment lasted 3 minutes and it was repeated over different people or combinations of them to ensure consistent results.

We defined the system's accuracy as the average percentage of time the marker was blocked for the people in the scene. At every instant, each person was assigned a binary state; BLOCKED or NOT BLOCKED, depending on whether they were able to see the marker. We then measured the average duration each person spent in BLOCKED state and compared it against the fixed experiment duration of 3 minutes. To determine the

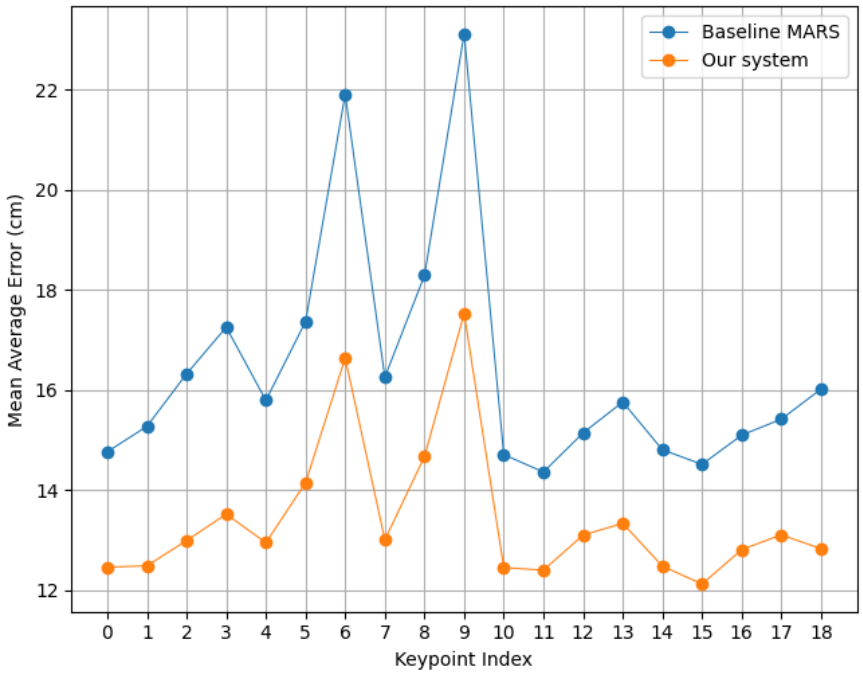


Figure 6.3: MAE comparison between the end-to-end baseline MARS and the proposed system approach over a single dynamic target.

participants' state, each person was given a smartphone with an application that allowed them to switch between states with a simple touch in real time.

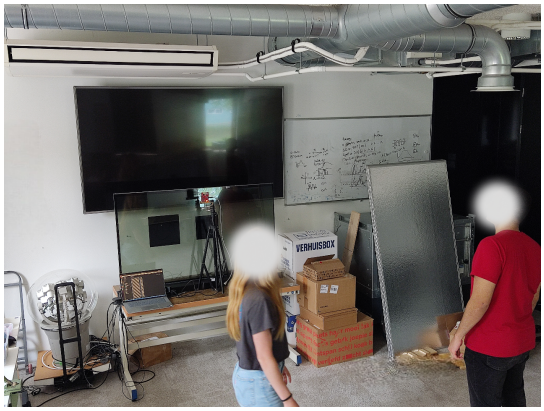
The parameters we experimented upon where the number of people in the scene and the size of the fading squares used to block the line-of-sight. Given the size of the smart screen installation, we decided to limit the number of people moving simultaneously in the scene to no more than 3. This decision was made to prevent the spawning of too many squares that would result in an overcrowded screen. Additionally, we set a lower bound to the size of the fading squares to 15cm x 15cm, as we deemed any size smaller than this, incapable of physically blocking a sensitive object like a computer's monitor.

Table 6.2: Accuracy of the system based on square size and number of people in the scene.

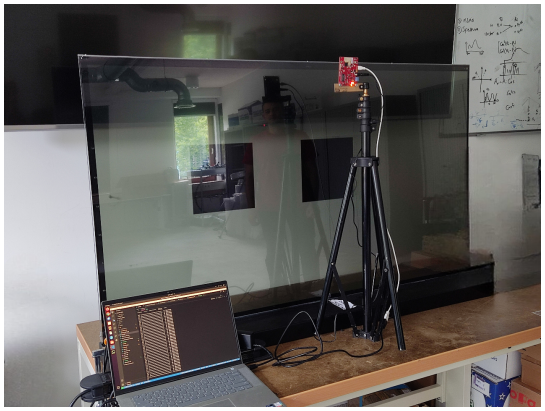
Size of squares	1 person	2 persons	3 persons
30cm	96%	95%	98%
20cm	93%	92%	90%
15cm	88%	81%	79%

The evaluation results are presented in Table 6.2. The general trend was that the larger the size of the squares, the higher the system's accuracy, which was expected since larger squares allow for larger margin of error in the localization and posture estimation system. In the extreme case were the squares would cover the whole screen, the system's accuracy would be 100%. On the contrary, the more the number of people increased, the more the accuracy dropped. That is because the system's load became more computationally intense and occurrences like human-to-human occlusions appeared, negatively affecting the tracking system. In the outlier case of 30cm squares tested with 3 people, the large squares covered a major percentage of the screen's surface. The significant coverage blocked people's sight, even when the originally assigned squares failed to do so themselves. Through the experiments we decided that the fading squares of size 20cm x 20cm provide a good trade-off between accuracy, system's practicality and elegance of design.

The major causes of system inaccuracies included lagging behaviour, tracks becoming undetected and incorrect square positioning. The system's computational procedures introduced jitter in real time operation,



(a) Evaluation area with two participants in the scene



(b) mmWave and smart screen setup

Figure 6.4: Online evaluation setup

especially when there were multiple people in the scene. The multiple presences spawned more data points, created multiple tracks and eventually caused the jitter to grow, which led to lagging behaviour of the squares. The rest of the issues were primarily linked to the natural limitations of the tracking and posture estimation systems to accurately handle static targets and temporal signal loss from the sensor.

6.3. Intermediate Design Choices

In this section we justify some of the design choices we made on the system and the dataset. We delve into the reasoning behind choosing constant acceleration as a motion model for our Kalman Filter (Section 3.3.2) and present an attempt of ours to augment the proposed dataset.

6.3.1. Motion Model: Constant velocity vs constant acceleration

One of first challenges that appeared while designing the MPPE system was the choice of the most appropriate motion model. We performed an analysis between the constant velocity model and the constant acceleration model to identify which of the two yields the best results over the intended application. The main difference between the two is that the state vector of the filter using constant velocity, includes only the position and velocity information of points, while the one using constant acceleration also includes the acceleration vectors. As a result, the first assumes constant speed between two consecutive frames while the second assumes constant acceleration.

To compare the two models we used samples from the created dataset. More specifically, we used the dynamic experiment where all participants performed plain walking to evaluate the motion models. We avoided mixing in the experiment where waving movements were involved, to avoid shifting the pointcloud's center of mass which would introduce noise to our study. We measured the motion model's performance over the x and y axes to focus on the target's horizontal disposition and measured each motion model's error by comparing the coordinates of the target's waist (ground truth keypoint with index 0) against the Kalman filter's output.

Table 6.3: Comparison of mean average errors between the constant velocity and the constant acceleration model. The errors are defined by the difference between the Kalman filter output and the ground truth coordinates of the target's waist (keypoint index 0).

	Constant Velocity Model	Constant Acceleration Model
X-axis MAE:	12.79 cm	10.17 cm
Y-axis MAE:	11.53 cm	6.13 cm
Average MAE:	**12.17 cm**	**8.15 cm**

Upon observing the behaviour of the Kalman model's prediction, we noticed that the constant velocity motion model appeared to lag and overshoot more than the constant acceleration model, especially in swift movement changes. Furthermore, in some cases, the constant velocity model lost track of the target, forcing the system to detect the track from scratch through the DBSCAN algorithm. The results show a clear accuracy

boost when using the constant acceleration model, which reduced the mean tracking error by approximately $4cm$. Consequently, we choose the constant acceleration model as our proposed system default approach.

6.3.2. Data Augmentation: Introducing Noise

An attempt to improve our solution had the goal to strengthen the proposed dataset by performing data augmentation. We placed the data augmentation module after the dataset partitioning stage (see Section 5.3), right before the model's training. In this approach we introduced noise to the training set by randomly dispositioning the points of its pointclouds. We generated random offsets in all three axes by using a Normal distribution with a mean value of 0 and a standard deviation of 0.02m, which is half of the unambiguous range specified in the radar configurations. We performed this procedure once for every frame, effectively doubling the size of the training set.

We assumed that introducing this variation in the dataset would allow the CNN model of our proposed system to generalize better in scenarios that it had never seen before. On the contrary, the system's performance did not gain any notable improvement and at specific joints at even appeared to perform slightly worse. We speculate that the great sparsity of data and the overall random shapes of the pointclouds have resulted in low sensitivity of the trained model over slight changes in the data point coordinates. Due to the results of this study we refrain from using this augmentation method in our dataset.

6.4. Ablation Studies

In this section we perform an isolated evaluation of the improvements of their impact to the system's performance. The design choices that will be discussed include:

- The baseline MPPE system design, combining MARS and GTRACK (Chapter 3). For simplicity we will refer to this system as "MARS w/ Tracking".
- The elimination of the position vector from the CNN model's input (Section 4.2), referred to as "Improvement 1".
- The addition of temporal connections between stacked frames in the posture estimation model (Section 4.3), referred to as "Improvement 2".

We will **not** analyze the modification of the DBSCAN distance metric improvement, because it does not aim to increase the system's accuracy, rather to recover from tracking losses faster. Finally, the last two improvements will be evaluated against our baseline MPPE system design (MARS w/ Tracking) as both depend on the existence of a Tracking system.

6.4.1. Impact of the Baseline MPPE system

As described in Section 3.4.2, incorporating the tracking system in posture estimation results in isolated track pointclouds that are also filtered from noise. This functionality should allow the CNN based estimator to perform better and more consistently without the influence of phenomena such as multi-path which creates ghost targets in the scene.

We compared the baseline top-down MPPE system we developed against the baseline of MARS, essentially comparing the performance of MARS over filtered and unfiltered input sets. Table 6.4 reveals that the Tracking system significantly lowers the RMSE error of the system and thus improves the accuracy of the worst case predictions. The filtered input allows the CNN model to perform more consistent localization without identifying noise interference as a human target.

Table 6.4: MAE and RMSE comparison between the baseline MARS model and our proposed baseline MPPE system.

Baseline MARS				MARS w/ Tracking			
All joints		Head		All joints		Head	
MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
16.43	251.87	17.26	279.32	14.55	19.08	15.51	20.03

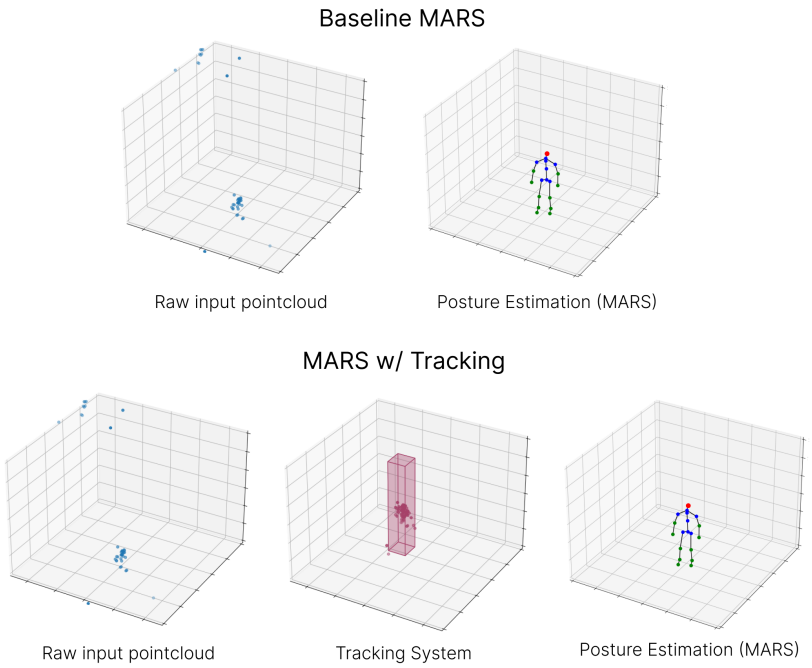


Figure 6.5: Comparison of the chain of modules between the baseline MARS model and the proposed baseline MPPE system

6.4.2. Improvement 1: Eliminating the position vector

The first improvement that we performed on top of the fundamental top-down structure utilizing MARS w/ Tracking, was eliminating the position vector from the track pointclouds and in this section we will present its isolated impact on the system's accuracy. Eliminating the position vector allows us to clearly separate the functions of localizing a target and estimating their posture between different dedicated systems. In our case we exclusively assign the Tracking system to perform localization and MARS to estimate posture. As a result, the errors due to these functions, namely ϵ_l and ϵ_p are discrete between the tracking and posture estimation system components.

The accuracy gained by this improvement and through the localization capabilities of the tracking system is presented in Table 6.5. Indeed, the tracking system was able to perform better localization than the CNN model from MARS, reducing the mean average error by 1.1cm and the RMSE error by 2.4cm, which is a reduction of approximately 10%. We expect this accuracy gap to widen in real-world scenarios with larger area of interest, where MARS generalization capability would have a more significant effect on the results.

Table 6.5: Comparison of errors between MARS combined with the Tracking module and the improvement achieved by removing the position vector.

	All joints		Head	
	MAE	RMSE	MAE	RMSE
MARS w/ Tracking	**14.55**	**19.08**	**15.51**	**20.03**
Removing Position Vector	**13.42**	**16.68**	**13.52**	**16.37**

6.4.3. Improvement 2: Adding Temporal Learning

The second improvement we implemented was to combine consecutive track frames and pass them as a vortex into a modified MARS model, which utilized 3D convolution. Since all the previous improvements and system modifications were implemented on stages prior to the module of posture estimation, we will evaluate the contribution of this improvement both on top of Improvement 1 and individually.

Frame Stacking vs Frame Fusion

First, we want to investigate the impact of stacking consecutive frames over Improvement 1. FUSE [6] proposed the approach of fusing frames to enhance the pointcloud density and has already performed an analysis that 3 frames give the best results without introducing noise. Along those lines, we will compare:

- the proposed approach of FUSE, fusing 3 frames into a single entity
- our approach of stacking 3 frames while maintaining their order and identities to learn from their temporal connections.

Since Improvement 1 removes the localization function from MARS, the impact of MARS to the full system includes only the posture estimation error ϵ_p and the differences of accuracy in the results account only for that error. Table 6.6 provides both the results of the total system error as well as the isolated ϵ_p.

Table 6.6: MAE and RMSE comparison between the FUSE approach and our proposed frame stacking approach. The second large column indicates the isolated posture estimation error ϵ_p for both approaches.

	System's error (ϵ_l+ϵ_p)		Isolated ϵ_p	
	MAE	RMSE	MAE	RMSE
FUSE approach (frame fusion)	13.37	16.64	6.34	8.77
Our approach (frame stacking)	13.15	16.21	5.96	8.20

The comparative results of this study are graphically presented in Figure 6.6. We observe that the approach of FUSE does not give much improvement compared to the baseline. It exhibits a drop in MAE of only a few millimeters, indicating that after filtering environmental noise from the scene, a frame carries sufficient information for the CNN model to extract a representative instantaneous human posture. On the other hand, adding temporal learning through frame stacking and usage of 3D kernel layers in the CNN improves more accurate predictions of continuous movements. It drops both the MAE and RMSE errors of the FRAME approach by approximately 0.4cm and 0.6cm respectively.

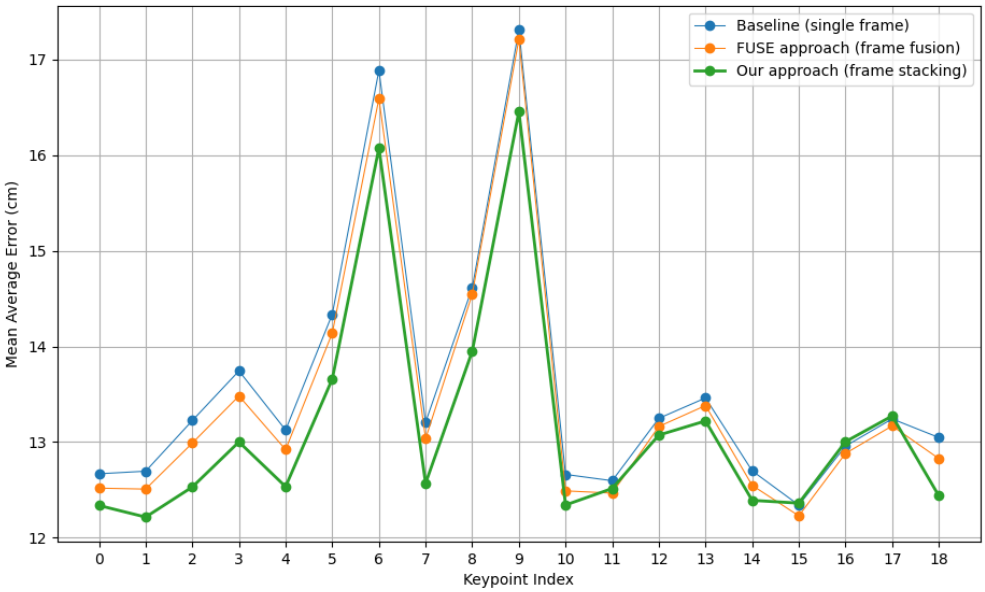


Figure 6.6: MAE comparison between the baseline, the FUSE approach and our proposed frame stacking approach.

Isolating the impact of 3D convolution

Our second study on the improvement of temporal learning focuses on its individual impact over the system's accuracy. To measure the isolated accuracy of this improvement we will remove the Improvement 1 from the system, effectively allocating both of the localization and posture estimation tasks to the CNN model.

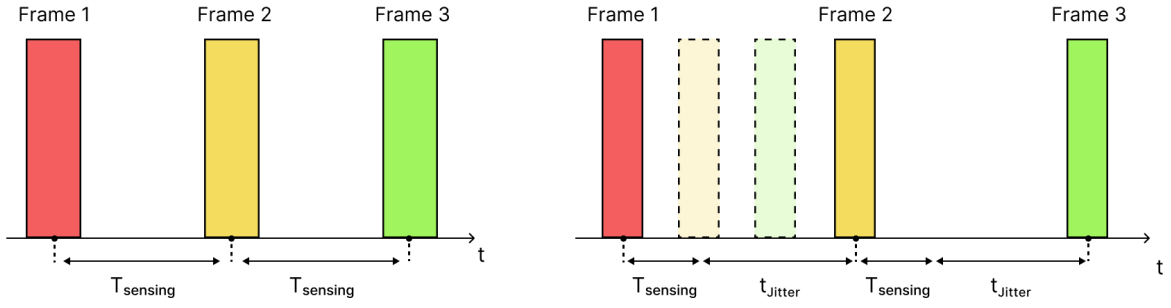
The results of this approach (Table 6.7) appear really interesting. This approach, when isolated, drops the total system MAE to approximately 10cm, which is even better than the MAE measured in our final solution. It also succeeds in reducing the localization error at the x-axis, where the tracking module was suffering most, down to half. The high x-axis error in the proposed solution is mainly caused due to the shift of the track's centroid when a person is performing movements that include waving. Moreover, the isolated improvement appears to be more robust in the RMSE field as well, indicating that the temporal misjudgements, from which the CNN model used to suffer when localizing a target, have been drastically reduced.

Table 6.7: Comparison of the per-axis MAE and RMSE errors over the Improvement 2 and the Final Approach of the system.

	Improvement 2				Final Approach			
	All joints		Head		All joints		Head	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
X axis	9.29	12.69	9.95	13.40	18.51	22.01	18.31	21.86
Y axis	13.56	19.40	12.62	18.41	9.30	14.83	10.66	13.19
Z axis	7.47	10.44	9.23	12.59	11.66	11.80	10.03	12.17
Average Error	**10.10**	**14.17**	**10.60**	**14.80**	**13.15**	**16.21**	**13.00**	**15.74**

Since the isolated approach yields better localization results than the tracking system, the question that is raised is why don't we use this approach on our final solution. Upon performing online tests we came across some limitations on the sole use of CNN for localization. First, when the system is operating in real time, it introduces Jitter due to its computational intensity. This jitter slows down the sensing frequency and eventually prolongs the duration between the stacked frames (Figure 6.7), introducing prediction errors.

Secondly, localization through a CNN is highly affected by the sparsity or absence of points. When the track's associated points are below 5-10, the CNN struggles to localize the target as much as it struggles to estimate its posture. Moreover, its brief memory of 30 ms (it only takes into account 3 frames through this improvement) makes it sensitive to human-to-human occlusions in a multi-person scenario. Finally, since the dataset was captured strictly in a designated area of 2m x 3m, we question the model's ability to generalize in really close or larger ranges.



(a) Frames during Training (a) Frames during Online operation

Figure 6.7: Stacked frame period differences between the training set and online operation. Jitter t_{jitter} is introduced in the second case due to computational delays.

In conclusion, we choose to limit the impact of this improvement to affect only the posture estimation error, by combining it with Improvement 1. Figure 6.8 presents the total system's error including both ϵ_l and ϵ_p and the isolated ϵ_p drop that we gain through our final solution. Through the observation of its promising results, though, we acknowledge the possibilities of using an end-to-end NN approach for localization and posture estimation and we leave it as a proposal for future work.

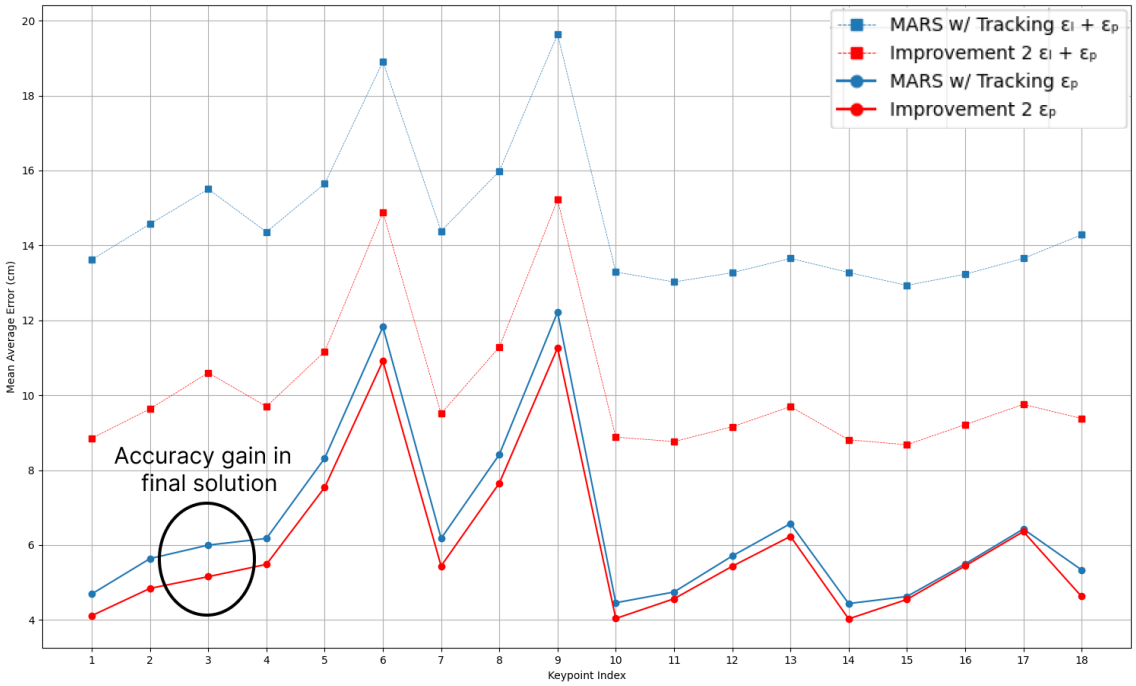


Figure 6.8: Separation of the gained posture estimation accuracy ϵ_p observed in the final solution. NOTE: The posture estimation error at keypoint-index 0 is zero.

6.5. Results Summary

In this section we summarize all the results and graphs from the proposed approaches and improvements. Table 6.8 includes the average MAE and RMSE errors of all 19 human joints as well as separated the ones for associated with the head to assess its usability for the full system. Figure 6.9 shows the MAE error of every posture estimation approach followed during the ablation studies.

Table 6.8: Average MAE and RMSE error (in cm) for the combined 19 joints and the head joint over all the analyzed approaches.

	All joints		Head	
Approaches	MAE	RMSE	MAE	RMSE
Baseline MARS	16.43	251.87	17.26	279.32
MARS w/ Tracking	14.55	19.08	15.51	20.03
Improvement 1	13.42	16.68	13.52	16.37
Improvement 2	10.10	14.17	10.60	14.80
Proposed System (Improvements 1 & 2)	**13.15**	**16.21**	**13.00**	**15.74**

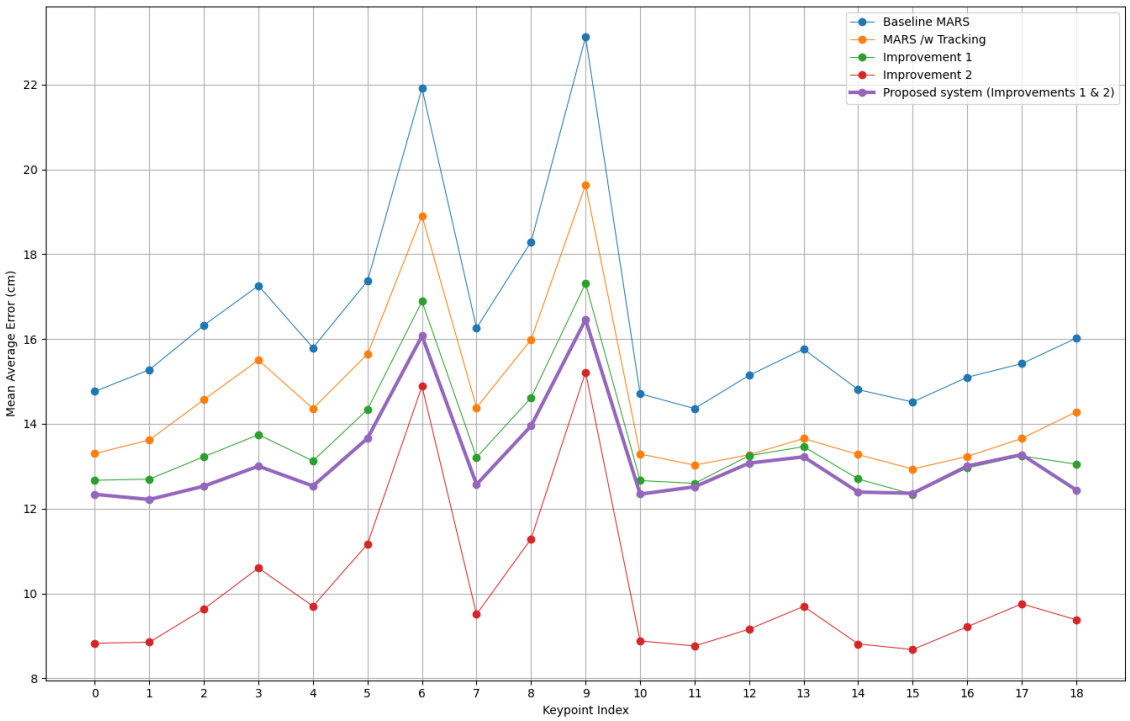


Figure 6.9: MSE comparison of all analyzed approaches. **NOTE:** The improvements 1 and 2 have been made over the MARS w/ Tracking approach.

7 | Discussion

The aim of this thesis was to investigate and design a real-time MPPE system that is able to detect dynamic targets, in order to achieve privacy shielding. On the basis of this, we performed multiple evaluations, comparing approaches by analyzing their individual impact on the system's performance. We decided to eliminate some proposed approaches such as augmenting the dataset through noise addition and made decisions with due consideration over which motion model and which frame combination method to use in the system. Overall, our final solution outperformed the baseline of MARS end-to-end approach by 20% in MAE for posture estimation on a single, dynamic target and achieved a mean accuracy of 92% when it was blocking the view of multiple targets over a sensitive object.

The results show that integrating tracking into posture estimation yields more consistent predictions by significantly reducing the baseline's RMSE error. We found out that filtering the input of the CNN model from noise interference really improves its worst case predictions. Additionally, the introduction of the two improvements of position vector elimination and temporal learning over stacked frames, result in more robust and accurate predictions by avoiding generalization problems and temporal pointcloud issues. Combined, the two improvements produce better results than the baseline MPPE system design, reducing the MAE by 2.50cm and the RMSE by 4.50cm.

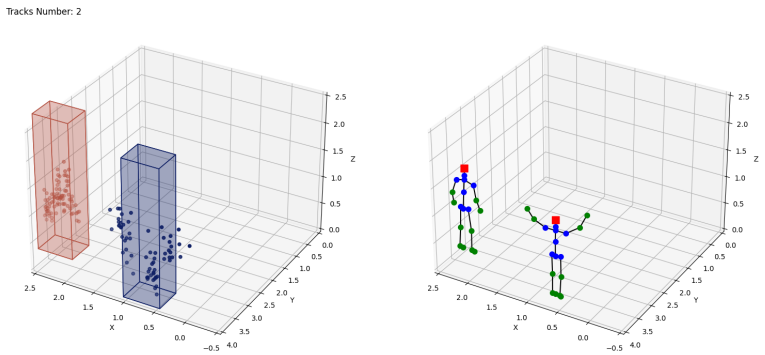


Figure 7.1: Final MPPE system's tracking and posture predictions over multiple people.

Our research contributes insights with regard to methods and techniques for mmWave-based posture estimation system designs. Most importantly, this research extends the literature of mmWave by proposing the first top-down MPPE system for privacy shielding applications. Previous studies had only focused on single targets, making their findings applicable to only a limited range of applications. Additionally, through this thesis we facilitate research on dynamic target posture estimation which was inadequately addressed by previous mmWave-based single-person posture estimation studies. The practical implications of this thesis in this area, lie in the separation of localization and posture estimation as two different functionalities and the proposal of the new dataset which includes highly dynamic and weakly supervised movements.

Apart from its impact on research, this thesis delivers a complete system-application for privacy shielding. Achieving high real-time accuracy for multiple people in a scene with considerable noise interference, it poses as a suitable solution for a variety of indoor and possibly outdoor spaces. The system's inherent ability to filter out noise and isolate the targets' "bounding boxes" allow it to adapt into different environmental conditions, while its installation becomes effortless through scene configuration parameters.

7.1. System Limitations

Of course, the proposed system comes with its limitations. The shortcoming of mmWave technology over capturing stationary targets impacts our system's robustness in scenarios where the passersby briefly remain perfectly still. In that case, after the track lifetime times out, the tracking system will assume the track inactive and will remove the fading square associated with it, exposing the sensitive object. Additionally, since we approach the problem at hand through a top-down MPPE approach, the computational demands of the full pipeline increase proportionally with the number of people in the scene. Additionally, in cases of multiple people, the system's performance deteriorates as it introduces jitter. The jitter affects the temporal associations of frames, as discussed in Section 6.4.3, interfering with the accuracy of posture estimation and introducing lag to the projections on the smart window. If the fading square size is not large enough to account for that lag, the system might instantaneously reveal the shielded object. We currently attribute this issue to the computational resources of the machine running our system, which are responsible for the amount of jitter introduced.

Finally, a limitation that appeared during the dataset creation phase revealed that the static movements performed during the data collection spawned less points than the dynamic ones. As a result, frames with insufficient points were discarded, causing occasional imbalances in the number of data samples associated with every movement. Our model trained on the created dataset, although reporting great accuracy, tends to have higher success rate in predicting walking or waving motions than sideways leaning or squatting static positions.

7.2. Future Work

During the development of our system, we identified several potential areas for future work that could significantly enhance and complement our efforts, as well as those of related research.

- **System Optimization:** First, optimizing the system at a firmware level could result in significant benefits. By improving memory allocation, reducing the use of dynamic-length lists and utilizing multithreading and multi-processing capabilities for parallel execution of tasks, we could reduce system delays and jitter, as well as facilitate the system's software on embedded platforms, offering a more integrated solution.

- **Static Target Sensing:** Secondly, research on detecting static targets through mmWave technology is still insufficiently addressed. We encourage endeavours that focus on combining other non-intrusive technologies, such as thermal sensing, with mmWave to complement each other's limitations. Additionally, we wish to highlight the potential benefits of dynamically changing the sensor's configurations mid-operation. With appropriate firmware flexibility, we assume that online configuration changes of the chirp's profile could be catalytic for static target sensing and tracking.

- **AI in posture estimation:** Lastly, we wish to recognize the potential of using Deep Neural Networks. We are confident that with appropriate foundational research on the application of AI in mmWave technology, end-to-end approaches could outperform state-of-the-art methods in both dynamic and multiple target posture estimation. It would be particularly interesting to see efforts aimed at developing a bottom-up approach over mmWave pointcloud.

8 | Conclusion

Through this thesis we developed a novel application for shielding sensitive objects from the sight of passersby, through the use of a smart window which projects fading squares, aligned with their gaze vectors. The privacy shielding functionality was achieved using of a mmWave-based Multi-Person Posture Estimation (MPPE) system, alongside the proposal of a new associated dataset. Building on top of the foundational systems GTRACK and MARS, the research investigated improvements for posture estimation on dynamic targets. Finally, it successfully achieved high overall performance in shielding sensitive objects from the sight of multiple people, providing a complete, non-intrusive and appealing solution for privacy protection applications.

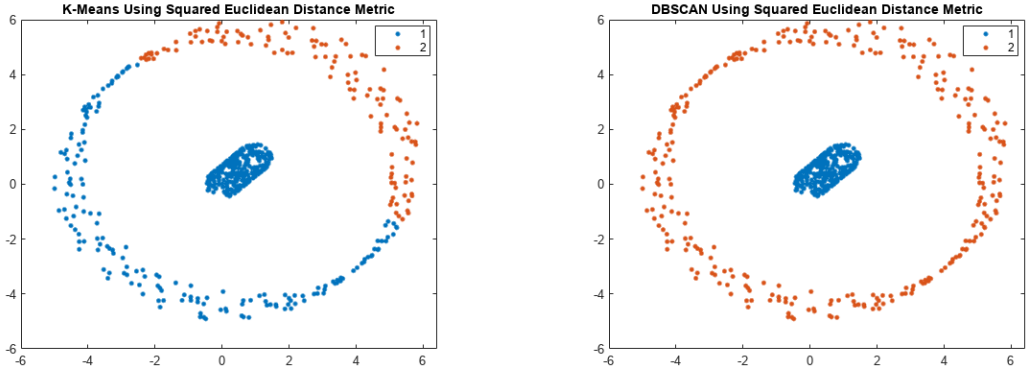
To conclude, this paper innovates by providing new insight on real-time and dynamic target posture estimation and sets the first baseline of mmWave-based MPPE systems for privacy shielding applications. The findings of this study set the foundation for further progress in the field of mmWave human sensing and guide the literature to explore more real-world applications, including multi-person and out-of-the-lab scenarios.

A | Appendix

A.1. DBSCAN Clustering

A large majority of the proposed mmWave Multiple Target Tracking (MTT) systems have been utilizing a clustering algorithm called DBSCAN for human detection [51] [44] [25]. DBSCAN is a density-based algorithm and it is preferred in tracking applications as it can handle a non-predefined number of noisy and non spherical clusters, as well as outlier points. Figure A.1 shows the comparison of DBSCAN to the K-Means clustering algorithm over arbitrary cluster shapes.

The algorithm requires no more than two parameters, the epsilon value (ϵ) and the minimum number of points required to form a cluster. It starts by a random starting point and calculates the amount of points that exist within the region dictated by ϵ. In the case that the number of points exceeds the minimum threshold, a cluster is created. Otherwise it is considered noise unless it belongs in the region ϵ of another point that has sufficiently dense neighbourhood. The ϵ region is calculated by a metric function such as the euclidean distance and can thus become ellipsoid in the 3D space if weights are applied to certain axes.



(a) (b)

Figure A.1: Comparison of (a) K-means and (b) DBSCAN algorithms performed by [31]

A.2. Kalman Filters

A common approach for tracking systems is to utilize a Kalman filter for temporal target associations. A Kalman filter is an estimation algorithm that predicts system parameters and is extensively used in systems with inaccurate or inconsistent measurements. In human tracking, the filter employs a theoretical motion model to predict the next position of targets. It then combines this prediction with the measured location, along with estimation error and noise covariances, to update the filter's gain and estimate the new state of the targets.The new state s at time instant n is defined as

$$s(n) = Fs(n-1) + w(n) \tag{A.1}$$

where F is a transition matrix representing the motion model and w is a vector of process noise. Kalman Filter produces estimates that tend to be more accurate than the measurements of a single, noisy sensor and can additionally handle temporal signal loss, target occlusions and measurement noise.

A Kalman filter estimates a new state vector for the system that might be different from its input measurement vector. For instance, a sensor might only be able to detect the position and velocity of an object, while the system also predicts information about its acceleration state given some previous measurements. It is also not uncommon for a system to perform transformations to the measurement vector like axis normalization. Let the input measurement vector for instance n be u(n), there is a measurement matrix H:

$$u(n) = H(s(n)) + v(n) \tag{A.2}$$

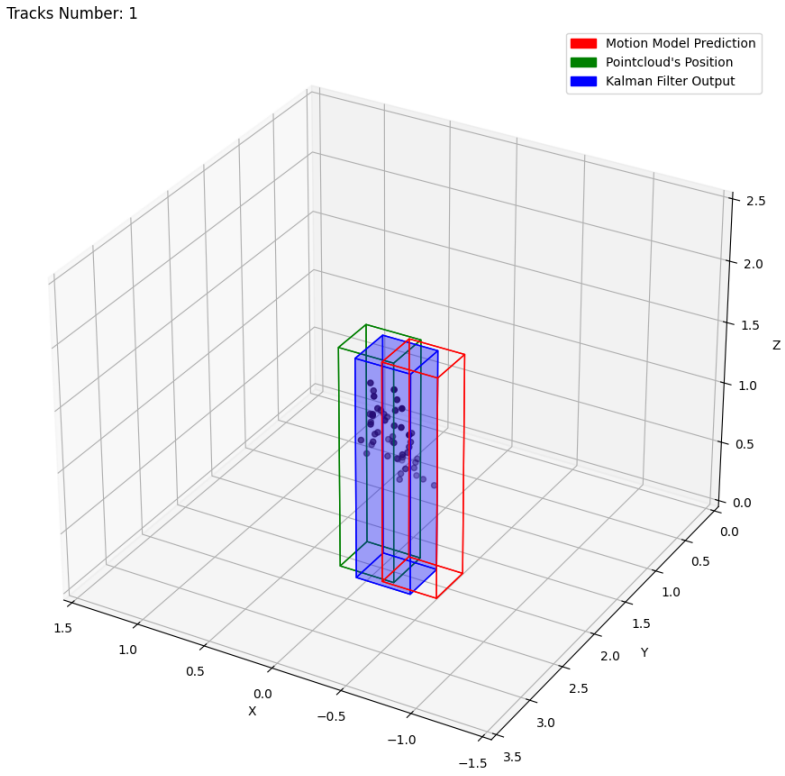


Figure A.2: Kalman output over varying motion model predictions and sensor measurements. The difference might have occurred due to swift target acceleration or noisy measurements affecting the cluster's centroid.

where v is a vector of measurement noise.

There are cases when the relation between u(n) and s(n) is not linear, like when the system needs to perform transformation from spherical to Cartesian coordinates. This non-linearity introduces complexity to the system and in result, some Kalman Filter alterations have been proposed to simplify this problem. Filters like the Extended Kalman (EKF) and Unscented Kalman (UKF) appear in the literature and propose solutions such as approximating a linear behaviour to avoid more complexities.

Finally, for the robust operation of a Kalman Filter, a motion model that produces realistic predictions over the system's use case is needed. Approaches modeling targets with noticeable patterns in their movement have incorporated constant velocity, acceleration or turning rate motion models to fit the system's needs. In this thesis we evaluate the constant velocity and constant acceleration motion models for unsupervised human motion in Section 6.

References

[1] Aakriti Adhikari et al. “MiShape: Accurate Human Silhouettes and Body Joints from Commodity Millimeter-Wave Devices”. In: Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 6.3 (Sept. 2022). doi: 10.1145/3550300. url: https://doi.org/10.1145/3550300.

[2] Fadel Adib et al. “Capturing the Human Figure through a Wall”. In: ACM Trans. Graph. 34.6 (Nov. 2015). issn: 0730-0301. doi: 10.1145/2816795.2818072. url: https://doi.org/10.1145/2816795.2818072.

[3] Abdullah K. Alhazmi et al. “Intelligent Millimeter-Wave System for Human Activity Monitoring for Telemedicine”. In: Sensors 24.1 (2024). issn: 1424-8220. doi: 10.3390/s24010268. url: https://www.mdpi.com/1424-8220/24/1/268.

[4] Sizhe An, Yin Li, and Umit Ogras. mRI: Multi-modal 3D Human Pose Estimation Dataset using mmWave, RGB-D, and Inertial Sensors. 2022. arXiv: 2210.08394 [cs.CV].

[5] Sizhe An and Umit Ogras. “MARS: mmWave-based Assistive Rehabilitation System for Smart Healthcare”. In: ACM Transactions on Embedded Computing Systems 20 (Oct. 2021), pp. 1–22. doi: 10.1145/3477003.

[6] Sizhe An and Umit Y. Ogras. “Fast and scalable human pose estimation using mmWave point cloud”. In: Proceedings of the 59th ACM/IEEE Design Automation Conference. DAC ’22. ACM, July 2022. doi: 10.1145/3489517.3530522. url: http://dx.doi.org/10.1145/3489517.3530522.

[7] Mykhaylo Andriluka et al. “2D Human Pose Estimation: New Benchmark and State of the Art Analysis”. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2014.

[8] Z. Cao et al. “OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields”. In: IEEE Transactions on Pattern Analysis amp; Machine Intelligence 43.01 (Jan. 2021), pp. 172–186. issn: 1939-3539. doi: 10.1109/TPAMI.2019.2929257.

[9] Lin Chen et al. “Spatial-temporal Multi-scale Constrained Learning for mmWave-based Human Pose Estimation”. In: IEEE Transactions on Cognitive and Developmental Systems PP (Nov. 2023), pp. 1–13. doi: 10.1109/TCDS.2023.3334302.

[10] Common Objects in Context. COCO: Common Objects in Context. Website. 2/2024. n.d. url: https://cocodataset.org/.

[11] Han Cui et al. MiliPoint: A Point Cloud Dataset for mmWave Radar. 2023. arXiv: 2309.13425 [cs.LG].

[12] Hao-Shu Fang et al. RMPE: Regional Multi-person Pose Estimation. 2018. arXiv: 1612.00137 [cs.CV].

[13] FmmW-Group. IWR1443-Python-API. https://github.com/FmmW-Group/IWR1443-Python-API. Accessed: 2024-06-02. 2024.

[14] Google. Google Colaboratory. https://colab.research.google.com. Accessed: 2024-06-09. 2024.

[15] Junfeng Guan et al. “Through Fog High-Resolution Imaging Using Millimeter Wave Radar”. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020, pp. 11461–11470. doi: 10.1109/CVPR42600.2020.01148.

[16] Guorong He et al. “Fusang: Graph-Inspired Robust and Accurate Object Recognition on Commodity MmWave Devices”. In: Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services. MobiSys ’23. Helsinki, Finland: Association for Computing Machinery, 2023, pp. 489–502. isbn: 9798400701108. doi: 10.1145/3581791.3596849. url: https://doi.org/10.1145/3581791.3596849.

[17] Kaiming He et al. Mask R-CNN. 2018. arXiv: 1703.06870 [cs.CV].

[18] Xu Huang, Joseph K. P. Tsoi, and Nitish Patel. “mmWave Radar Sensors Fusion for Indoor Object Detection and Tracking”. In: Electronics 11.14 (2022). issn: 2079-9292. doi: 10.3390/electronics11142209. url: https://www.mdpi.com/2079-9292/11/14/2209.

[19] Xu Huang et al. “Indoor Detection and Tracking of People Using mmWave Sensor”. In: Journal of Sensors 2021 (Feb. 2021), pp. 1–14. doi: 10.1155/2021/6657709.

[20] Eldar Insafutdinov et al. DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model. 2016. arXiv: 1605.03170 [cs.CV].

[21] Texas Instruments. An Introduction to Automotive Lidar. Accessed: 2024-06-17. 2017. url: https://www.ti.com/lit/wp/slyy150b/slyy150b.pdf?ts=1718643960167&ref_url=https%253A%252F%252Fwww.google.com%252F.

[22] Texas Instruments. High Accuracy Range Measurement Lab. Accessed: 2024-05-22. n.d. url: https://dev.ti.com/tirex/explore/node?node=A__AF860eUj4a7vJ8WpT57LiQ__com.ti.mmwave_industrial_toolbox__VLyFKFf__LATEST.

[23] Texas Instruments. The Fundamentals of Millimeter Wave Radar Sensors. Accessed: 2024-05-22. n.d. url: https://www.ti.com/lit/wp/spyy005/spyy005.pdf.

[24] Prabhu Janakaraj et al. "STAR: Simultaneous Tracking and Recognition through Millimeter Waves and Deep Learning". In: 2019 12th IFIP Wireless and Mobile Networking Conference (WMNC). IEEE. 2019, pp. 211–218. doi: 10.23919/WMNC.2019.8881354.

[25] Feng Jin et al. "Multiple Patients Behavior Detection in Real-time using mmWave Radar and Deep CNNs". In: Apr. 2019, pp. 1–6. doi: 10.1109/RADAR.2019.8835656.

[26] Richie Koch. What is considered personal data under the EU GDPR? Accessed: 2024-06-16. 2024. url: https://gdpr.eu/eu-gdpr-personal-data/.

[27] Gregor Kurillo et al. "Evaluating the Accuracy of the Azure Kinect and Kinect v2". In: Sensors (Basel) 22.7 (2022), p. 2469. doi: 10.3390/s22072469.

[28] Shih-Po Lee et al. HuPR: A Benchmark for Human Pose Estimation Using Millimeter Wave Radar. 2022. arXiv: 2210.12564 [cs.CV].

[29] You Li and Javier Ibanez-Guzman. "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems". In: IEEE Signal Processing Magazine 37.4 (2020), pp. 50–61. doi: 10.1109/MSP.2020.2973615.

[30] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: Computer Vision – ECCV 2014. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755. isbn: 978-3-319-10602-1.

[31] MathWorks. DBSCAN - Density-Based Spatial Clustering of Applications with Noise. MathWorks. 2024. url: https://ww2.mathworks.cn/help/stats/dbscan.html.

[32] Zichuan Nie et al. "Adaptive Façades Strategy: An architect-friendly computational approach based on co-simulation and white-box models for the early design stage". In: Energy and Buildings 296 (2023), p. 113320. issn: 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2023.113320. url: https://www.sciencedirect.com/science/article/pii/S0378778823005509.

[33] Sameera Palipana et al. "Pantomime: Mid-Air Gesture Recognition with Sparse Millimeter-Wave Radar Point Clouds". In: Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 5.1 (Mar. 2021). doi: 10.1145/3448110. url: https://doi.org/10.1145/3448110.

[34] George Papandreou et al. Towards Accurate Multi-person Pose Estimation in the Wild. 2017. arXiv: 1701.01779 [cs.CV].

[35] Jacopo Pegoraro and Michele Rossi. "Human Tracking with mmWave Radars: a Deep Learning Approach with Uncertainty Estimation". In: 2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC). 2022, pp. 1–5. doi: 10.1109/SPAWC51304.2022.9833987.

[36] Leonid Pishchulin et al. "Articulated people detection and pose estimation: Reshaping the future". In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. 2012, pp. 3178–3185. doi: 10.1109/CVPR.2012.6248052.

[37] Leonid Pishchulin et al. DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation. 2016. arXiv: 1511.06645 [cs.CV].

[38] Robert Savage-Beringer et al. "The "Acceptance" of Ambient Assisted Living: Developing an Alternate Methodology to This Limited Research Lens". In: vol. 6719. June 2011, pp. 161–167. isbn: 978-3-642-21534-6. doi: 10.1007/978-3-642-21535-3_21.

[39] Arindam Sengupta et al. “mm-Pose: Real-Time Human Skeletal Posture Estimation using mmWave Radars and CNNs”. In: IEEE Sensors Journal PP (May 2020), pp. 1–1. doi: 10.1109/JSEN.2020.2991741.

[40] Cong Shi et al. “mPose: Environment- and subject-agnostic 3D skeleton posture reconstruction leveraging a single mmWave device”. In: Smart Health 23 (Nov. 2021), p. 100228. doi: 10.1016/j.smhl.2021.100228.

[41] Xiao Sun et al. Integral Human Pose Regression. 2018. arXiv: 1711.08229 [cs.CV].

[42] Swapnil. Kinect Skeleton Tracking. Accessed: 2024-06-09. 2019. url: https://github.com/swapnil0908/Kinect-Skeleton-Tracking.

[43] Texas Instruments. Texas Instruments - Analog, Embedded Processing, Semiconductor Company. https://www.ti.com/.

[44] Texas Instruments. Tracking Radar Targets with Multiple Reflection Points. 1.8. https://dev.ti.com/tirex/explore/content/radar_toolbox_1_00_01_07/source/ti/examples/People_Counting/docs/Tracking_radar_targets_with_multiple_reflection_points.pdf. Texas Instruments. Feb. 2022.

[45] VideowindoW. Transforming Windows into Screens. https://www.videowindow.eu/.

[46] Jie Xiong and Kyle Jamieson. “ArrayTrack: A Fine-Grained Indoor Location System”. In: 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). Lombard, IL: USENIX Association, Apr. 2013, pp. 71–84. isbn: 978-1-931971-00-3. url: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/xiong.

[47] Ehsan Yavari et al. “Is There Anybody in There?: Intelligent Radar Occupancy Sensors”. In: IEEE Microwave Magazine 15.2 (2014), pp. 57–64. doi: 10.1109/MMM.2013.2296210.

[48] Jia Zhang et al. “A Survey of mmWave-Based Human Sensing: Technology, Platforms and Applications”. In: IEEE Communications Surveys amp; Tutorials 25.4 (2023), pp. 2052–2087. issn: 2373-745X. doi: 10.1109/comst.2023.3298300. url: http://dx.doi.org/10.1109/COMST.2023.3298300.

[49] Zhengyou Zhang. “Microsoft Kinect Sensor and Its Effect”. In: IEEE MultiMedia 19.2 (2012), pp. 4–10. doi: 10.1109/MMUL.2012.24.

[50] Mingmin Zhao et al. “Through-Wall Human Pose Estimation Using Radio Signals”. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018, pp. 7356–7365. doi: 10.1109/CVPR.2018.00768.

[51] Peijun Zhao et al. “mID: Tracking and Identifying People with Millimeter Wave Radar”. In: 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS). 2019, pp. 33–40. doi: 10.1109/DCOSS.2019.00028.