Tracking People for an mmWave-Based Interactive Game

Reducing Stationary Target Noise in Tracking and Movement Reconstruction

Kaloyan Fachikov[1]

Supervisor(s): Assoc. Prof. Marco Zuñiga Zamalloa[1]**, Dr. Girish Vaidya**[1]

[1]**Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Kaloyan Fachikov
Final project course: CSE3000 Research Project
Thesis committee: Marco Zuñiga Zamalloa, Girish Vaidya, Michael Weinmann

Abstract

Interactive video games often use vision-based systems or wearables to track player movements. Vision-based systems are privacy-invasive, and wearables require frequent recalibration and recharging. Frequently-Modulated Continuous-Wave (FMCW) radars have been proposed as an alternative tracking solution addressing these problems. Working in the millimeter-wave (mmWave) range, they capture scenes as point clusters, ensuring privacy without attaching sensors to the user. Previous research has shown their applicability in rehabilitation, gait recognition, and smart home appliances. This study focuses on integrating an mmWave sensing device with an interactive version of the Breakout video game. We propose a general framework with three main modules – generating points, clustering them, and reconstructing the player's movements as in-game commands. Our system enhances an already existing Kalman filter-based tracking algorithm. Online experiments were conducted to compare the proposed system to the baseline algorithm. The proposed system decreases the standard deviation on the estimated target location by 33% against motionless targets, while maintaining the baseline accuracy when tested on moving targets. Furthermore, it allows a higher game refresh rate, thus smoothing in-game movements. These results demonstrate the potential of FMCW radars in enhancing interactive video game experiences.

Keywords - mmWave, tracking, stationary, interactive game

1 Introduction

Transparent screens are electronic displays that show content while allowing people to see through them. Installed in public areas, they function as versatile tools, blocking direct sunlight and distributing information [1]. Figure 1 illustrates a transparent screen deployment in the Rotterdam The Hague Airport. To increase the utility of these screens, we suggest adding interactive activities like video games. For instance, a simple interactive version of the Breakout video game [2], where the user navigates the paddle through physical movement, would provide entertainment to the passengers. An example game interface is shown in Figure 6c and a demonstration of the game played on a PC is available online [1].

Knowing the precise user location is crucial to facilitate such an interactive game. Tracking primarily relies on vision-based systems or standalone controllers. However, both introduce challenges that limit their applicability to public areas. Cameras are expensive, raise serious privacy concerns, and require strict lighting. At the same time, it is impractical to assume that all passengers have a controller compatible with our system. The limitations of these traditional tracking methods emphasize the requirement for an alternative solution that offers accurate and convenient tracking in an indoor setting.

Figure 1: Transparent screens installed in the Rotterdam The Hague Airport, displaying geometric figures and reducing the incoming sunlight. Courtesy of VideowindoW [3].

Recent progress in radio-based technology and its applications presents promising opportunities for an innovative approach to tracking users in an interactive video game environment. Frequently-Modulated Continuous-Wave (FMCW) radars transmit electromagnetic waves and analyse the reflected signals to determine an object's range, velocity, and angle [4]. Working in the millimeter wave (mmWave) range, these devices function indifferent to light. Furthermore, they ensure privacy by capturing the scene as a cluster of points (or a point cloud), as shown in Figure 6(a). Additionally, unlike controllers, they require no sensors attached to the user.

However, despite the extensive research on mmWave devices and their applications, there is a gap in their use for interactive video games. State-of-the-art (SoA) systems have three main limitations when considering a gaming application. First, some focus on gesture classification rather than continuous tracking [5], [6]. Second, others are vulnerable to multipath interference, inevitable in indoor environments [7], [8]. Lastly, there are also issues with the detection of stationary individuals, as systems lose track of the target when it is not moving sufficiently long [9].

Consequently, these limitations motivate our investigation into utilising an FMCW radar for real-time video game control. This paper aims to establish a novel approach to tracking people in real time for the interactive video game Breakout, using an mmWave radar. The main contributions of this work are as follows:

- We design an interactive game system using mmWave radar as a control module for Breakout. To understand our work better, please refer to a demonstration of the interactive game [2].
- We implement a novel technique reducing the variance in stationary target tracking by 33% [3].
- We evaluate various signal processing techniques to increase the game refresh rate.

This paper is structured as follows. Section 2 summarises video game tracking systems and relevant mmWave radar ap-

[1] https://www.youtube.com/watch?v=IK1wbIBbjyI

[2] https://youtu.be/O2M1cgoI-eU

[3] Measured in standard deviation in the target location.

plications. Section 3 introduces the challenges posed by using an mmWave radar as a tracking module in the interactive video game Breakout. Subsequently, Section 4 presents the proposed system design. The experimental setup is described in Section 5, followed by an evaluation of the results in Section 6. Section 7 discusses the limitations of this research, and Section 8 examines the reproducibility and integrity of our work. Finally, Section 9 concludes the paper by summarizing our findings and listing potential future improvements.

2 Related Work

To our knowledge, no previous research has been done on using an FMCW radar as a control module for a video game. The following subsections consider multiple tracking systems applied in the gaming industry, their limitations, and various mmWave applications, as the motivation for our research.

2.1 Tracking Solutions in Interactive Games

There are two primary tracking systems categories – vision- and controller-based. Here, we investigate the most popular devices in each category and their shortcomings.

Vision-Based Systems

Vision-based systems primarily rely on an RGB camera, a depth camera, or other motion capture modules [6]. A regular RGB camera has several disadvantages. First, it cannot sense the third dimension (depth), thus making it difficult to understand what a person is doing [10]. Second, a camera raises privacy concerns by capturing sensitive and identifiable information. Last, they are susceptible to poor lighting conditions, resulting in low-quality images in a dark environment.

Microsoft Kinect is an example of a motion capture system. It improves on the depth limitation posed by RGB technology, using a depth sensor that comprises an infrared (IR) projector and an IR camera [10]. Yet, it suffers a few major limitations, beyond privacy and strict lighting inherent to the RGB camera module. According to Zhang [10], heat or drift in the IR laser might invalidate the calibration between the IR projector and the IR camera, making the depth values inaccurate. Even though recalibration is not particularly complicated, it brings inconvenience to the user.

Standalone Controllers

The PlayStation Move Motion Controller [4] (or *wand*) works together with the PlayStation Eye camera [5], configured to detect the illuminating sphere attached to the wand. Knowing the sphere's size, it can accurately compute the controller's 3D coordinates. However, it tracks the player's hand(s) only. According to Tanaka et al., [11], upper-body motion can be estimated using Inverse Kinematics but the resulting accuracy would be poor. They also report that frequent recalibration is required for an accurate system performance. Even though quickly adopted by the users [12], recalibration would cause inconvenience in an intense gameplay. Finally, Teixeira et al. [12] show that button placement is a drawback, as users experience problems finding the appropriate button when prompted to.

[4] https://en.wikipedia.org/wiki/PlayStation_Move

[5] https://en.wikipedia.org/wiki/PlayStation_Eye

2.2 mmWave Radar Applications

In this section, we discuss several relevant studies on tracking and user identification, and gesture and whole-body movement recognition, using a single mmWave radar.

Tracking and User Identification

Zhao et al. [9] propose a tracking and identification system, called mID, consisting of three main modules. First, the generated points are grouped using DBSCAN, a density-based clustering algorithm. Second, it uses the Hungarian algorithm to associate identical objects across two consecutive frames and create object-corresponding tracks. In addition, it uses a Kalman filter as an unbiased estimator to predict the track's location when the target is "undetected due to occlusion or temporary loss from the sensing region" [9, p. 35-36]. Finally, the system uses a long short-term memory (LSTM) unit as an identity classifier. It uses a fixed-size bounding box to form an occupancy grid and encapsulate the body shape information, and sequential occupancy grids to infer movement characteristics. Their system achieved median position errors of $0.16\,\mathrm{m}$ and identification accuracy of 89% when tested with 12 participants.

As mentioned, mID uses the Hungarian algorithm to assign existing tracks to current measurements, with a step threshold defining the maximum distance between a track and its assigned measurement. However, a small threshold cannot support abrupt movements due to the large distance in the target's location. Conversely, a large threshold value could wrongly associate a new target with an old track. This makes the system less effective in highly dynamic environments.

Movement Recognition

Liu et al. [5] developed a four-module arm gesture recognition system, mmHomeGes, applied in smart home scenarios. First, it transforms raw mmWave signals into a *reflection point* cloud that depicts the performed gesture. Then, it removes interference by separating potential users from noise via *UDAN*, a novel user discovery algorithm [5]. Subsequently, a Hidden Markov Model determines whether the captured motion is among the defined arm gestures. Last, a shallow neural network and another Hidden Markov Model are used to respond to users' gestures in real time. mmHomeGes archives a recognition accuracy of 95.30% across various smart home scenarios, despite the impact of surrounding movements and concurrent gestures.

S. An and U. Ogras [6] proposed an assistive rehabilitation system, MARS. It preprocesses the point cloud data to a lower dimension and utilises a convolutional neural network (CNN) to reconstruct 19 human joints and skeletal movements. When evaluated against ten specific rehabilitation movements, MARS achieved an average mean absolute error of $5.87\,\mathrm{cm}$ for all joint positions, with a maximum error of 7° for the knee angle, and 13° for the elbow angle.

Singh et al. [7] develop a human activity recognition system focusing on five full-body movements – walking, jumping, jumping jacks, squats, and boxing. Their system, RadHAR, preprocesses the generated point cloud into a voxelised representation, standardising the input size. It then uses a sliding window to combine multiple frames before feeding

them into a classifier. The study examined how spatial and temporal dependencies in the data impact classifier performance. They concluded that the best results are achieved by combining time-distributed CNN layers for spatial feature learning and bi-directional LSTM layers for capturing temporal dependency. The results presented in this study reveal the importance of assuming temporal and spatial dependencies in sequential point cloud data.

The remarkable performance achieved in these studies demonstrates the capabilities of mmWave radars in various applications, ranging from user tracking to complex movement recognition. However, these applications are restricted to controlled environments such as smart homes, rehabilitation systems, or specific physical activities. Consequently, the reviewed works do not address the complexities introduced by a highly dynamic, real-time environment, such as those found in interactive video games. Video games involve rapid, unpredictable movements and periods of little-to-no movement, and require instantaneous feedback, posing significant challenges.

3 Three Problems in Using mmWave Technology in Breakout

We have identified three challenges in integrating an mmWave radar as a paddle control for the video game Breakout. The first problem is inherent to the radar technology – the sensor captures few points on stationary targets. This makes predicting the exact location of a static object difficult. The second challenge arises from integrating the sensing technology into the game environment. A mismatch between the radar sampling rate and the game refresh rate results in snappy paddle movement. The third problem is caused by wave reflections from static and dynamic objects, resulting in ghost clusters being captured. In the following subsections, we explore these issues and their impact on the system.

3.1 Few Cluster Points on a Static Target

A non-moving person produces a weaker signal, reducing the number of captured points and making it difficult to identify them as a human. In contrast to the general motionless object detection problem, this research focuses on continuously tracking a target that stops moving. Since the target is actively tracked until it stops, the processing unit retains knowledge of its last known position and covering region. This alleviates the higher-level detection issue but introduces another problem, shown on Figure 2. As we can observe, sampling a limited number of points within the area results in a high variance in the static target's expected center of mass. Integrated into Breakout, this system misbehaviour causes high variance in the paddle location, as illustrated by the blurred effect in Figure 3b. Thus, it becomes crucial to minimise the noise introduced by the low number of points.

3.2 Refresh Rate Mismatch

Another challenge is the discrepancy between the radar sampling rate and the refresh rate, required for a smooth user experience. The radar samples a point cloud every $100\,\mathrm{ms}$, while the game screen is updated every $50\,\mathrm{ms}$. Thus, every

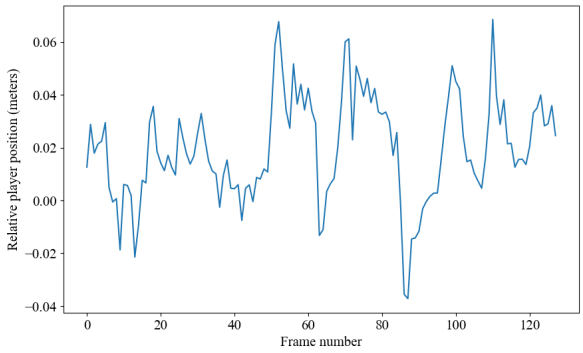
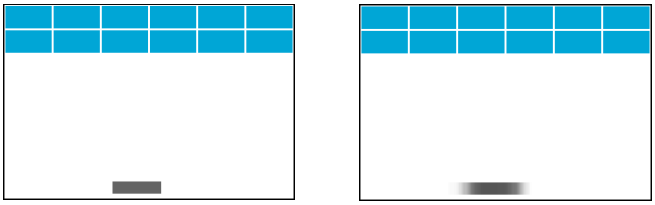


Figure 2: Tracking a stationary target, positioned right in front of the radar. The relative player position is measured in a dimension, perpendicular to the radar-to-player direction vector.



(a) (b)

Figure 3: Low-opacity snapshot overlay over 106 consecutive frames, throughout which the target is static. The darker the paddle color shade is, the more frequently the paddle appears there. Desired behaviour (a) and observed behaviour (b).

second game frame processes a point cloud. However, with no new information in the other frames, a naive reconstruction that uses the current user's position to place the paddle results in snappy movement. This behavior is illustrated in Figure 4. As we can observe, the paddle is moved only every second frame, disrupting the seamless interaction players expect.

3.3 Ghost Targets

In an environment with a substantial number of objects, wave reflections are common. They might result in the so-called *ghost objects* – nonexistent point clusters that the radar detects after multiple reflections of surrounding objects. Ghost objects can introduce a second target, which can mislead the system. If the ghost target is far from the real one, the system might track the ghost instead. If the ghost target is close, it can deceive the system into believing the tracked object is larger. When the real target is stationary, the expanded cluster dimensions caused by a nearby ghost could increase noise. However, if the real target starts moving, the system might continue to track that ghost.

In this paper, we focus on addressing the former two problems – reducing the variance in static target tracking and creating a framework that allows a high refresh rate.

4 System Design

In this section, we describe the design of our interactive video game system using the IWR1443BOOST radar [13]. First,

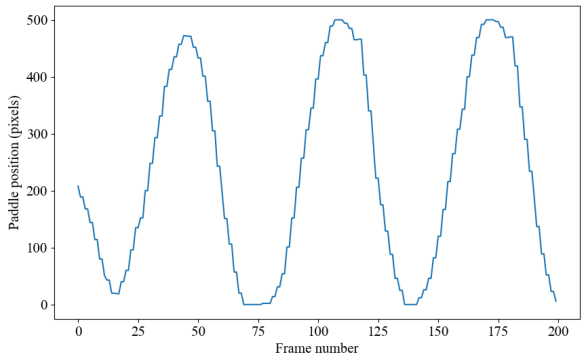


Figure 4: The paddle position at every frame. The radar generates a point cloud every 100 ms, whereas the game screen is updated every 50 ms. A reconstruction using the current player's position updates the paddle every 100 ms, resulting in snappy movement.

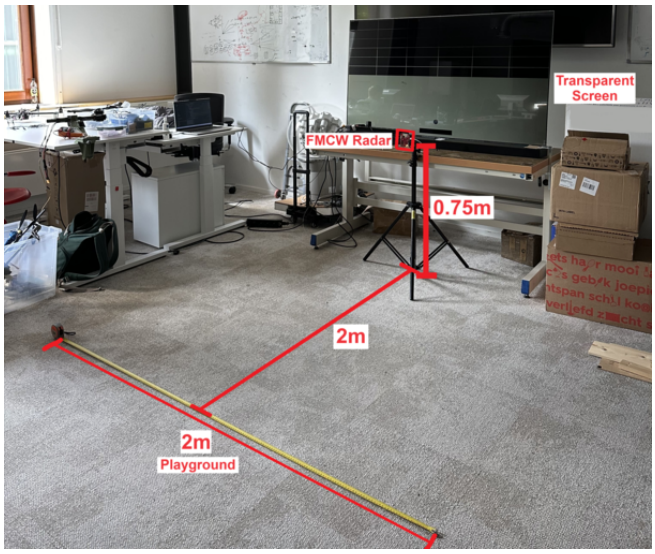


Figure 5: Environmental Setup. The labels indicate the radar, circled in red, the transparent screen, behind it, and the 2-m-wise playground.

we outline the development and evaluation environment, including radar placement and the recognised player area. Second, the radar configuration is detailed, specifying key parameters and their values. Last, we discuss the software component, tracking the user and reconstructing their movements.

4.1 Setup

The environment is depicted in Figure 5. The radar is placed at 0.75 m height. Other placement positions, specifically 1 m and 1.5 m, were considered, and it was empirically concluded that higher placement results in fewer generated points. The *playground* is a two-metre wide strip, positioned at a two metres distance to the radar, where the user is detected and can interact with the system. The transparent screen, visualising the game, is placed right behind the radar.

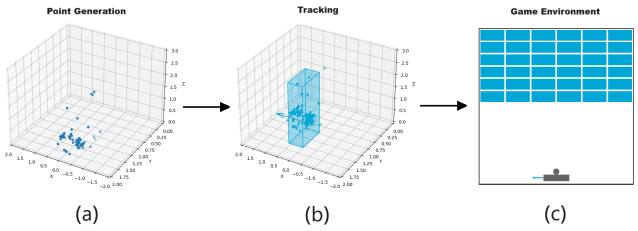


Figure 6: System Overview. Point generation module (a). Tracking module responsible for clustering and monitoring location and direction over time (b). The game environment controls the paddle using the *tracks* (c).

4.2 Radar Configuration

The configuration is generated using the Texas Instruments Demo Visualiser [6]. Further details on all configured parameters are shown in Table 1.

Table 1: Configuration Parameters

Parameter	Value
SDK version	2.1
Transmitting Antennas	3
Receiving Antennas	4
Maximum Unambiguous Range (m)	8.02
Peak Grouping in Range Direction	0
Peak Grouping in Doppler Direction	0
Clutter Removal	1
Threshold Scale in CFAR Message	1130

4.3 Software

To facilitate an interactive video game, our system requires three independent modules – point generation, tracking, and game environment, as illustrated in Figure 6. First, an FMCW radar captures the scene as a point cloud. The second module clusters the points into tracks. Each track represents an object (in our case, a person) and must be independently tracked throughout the subsequent frames. The third module is responsible to emulate the game and register appropriate user actions as control commands. The implementation is publicly available on our GitHub repository [14]. In the rest of this section, we describe the tracking and reconstruction modules in detail.

Tracking

The tracking algorithm extends a system developed by Parlitsis [15]. Its tracking module works as follows. Once the point cloud is generated, the points are clustered using DBSCAN. A track is created for each cluster, and its state is continuously monitored in the subsequent measurement frames. First, the target's location, direction, and speed are estimated using Kalman filter. Subsequently, the newly captured points are assigned to the closest track, based on the expected track positions. Points that are not within any track bounding box

[6] https://dev.ti.com/gallery/view/mmwave/mmWave_Demo_Visualizer/ver/2.1.0/

are considered outliers. They are processed using DBSCAN and new dynamic targets are created, for instance, on people entering the scene. Following, the Kalman filter is updated using the measurement. Despite being considered a more accurate estimator than single measurements, the Kalman filter easily loses a target that moves quickly during experiments. The root problem is not identified, as it was not the primary focus of this research. The existing system already overcomes this issue. It adjusts the updated state along the x-axis, considering the distance between the updated state and the measured target location. Specifically, it adds a fraction of this distance to the state estimate, placing greater importance on precise measurements

Inspired by the group tracking algorithm GTRACK [16] developed by Texas Instruments, we expanded the track state update step to reduce the variance on static targets. In particular, the track state is updated based on its previous status and estimated velocity, the points assigned in this measurement, and their properties. The flow diagram in Figure 7 shows the decisions and the corresponding state changes.

The decision process is divided into two main branches based on whether any measurement points are assigned to a target. If no new data is associated with a target, the current track status is considered. If the target is static, the update step is skipped. If the target is dynamic, its estimated speed determines the next step – if the speed is below a threshold, the target is assumed to have stopped, and otherwise, it is considered occluded by another target and assumed to continue moving. Conversely, when new points are associated with a target, the number of dynamic points among them is evaluated. If this number exceeds a threshold, the target is considered dynamic and updated accordingly. If the number of dynamic points is below the threshold, the track's current status is assessed. If the target is static, the update step is skipped, while if it is dynamic, its estimated speed is considered. Similar to the other branch, a speed lower than a threshold value suggests the target has stopped, while higher speeds indicate continued movement.

We explore the influence of three particular parameters on the system behavior. The first, the point velocity threshold V_p determines the minimal speed necessary to consider a point dynamic. The second, the dynamic points count threshold N_d, determines the number of dynamic points needed to classify a target as dynamic. The third one, the target velocity threshold V_t (or *minVelocityStopNoDyn* in Figure 7) defines the speed, below which a target with few dynamic points is considered stopped. Specific threshold values are further investigated in Section 5.1.

Reconstruction

As our system uses the Breakout video game, the reconstruction module focuses on positioning the paddle based on the player's location. To overcome the snappy movement problem discussed in Section 3.2, we implement a signal processing technique known as moving average [17]. This method uses a sliding window to process the most recent data points. In particular, we calculate the *most recent user location* at each frame. We either use a past measurement, when the frame is between measurements or the current measurement. We calculate the average of the most recent user locations over the last three frames. This average guides the paddle's placement. While effective in smoothing the movement, this approach introduces delay by considering past measurements when determining the current paddle location. In this paper, the delay measures the time required for the system to position the paddle according to the user's movement. Further information about the optimal window length W_l and detailed evaluation can be found in Section 5.2. In addition, Section 5.2 considers another smoothing technique, namely interpolation.

5 Evaluation

Four types of experiments are conducted. Each experiment follows one of the guidelines that specify the player's actions.

Type 1: Staying straight right in front of the radar, feet together. This experiment mimics a person about to start the game, or waiting for the ball to reach the paddle in a low-speed game.

Type 2: Staying straight right in front of the radar, feet apart. This experiment mimics a person waiting for the ball to reach the paddle in a high-speed game.

Type 3: Move sideways with feet apart, facing the radar, and staying within the designated playground. The actor should stop for a second every time they reach the playground border before they move to the other end. This experiment mimics a person trying to position the paddle where the ball would be in a low-speed game.

Type 4: Move sideways with feet apart, facing the radar, and staying within the designated playground. The actor should constantly move end-to-end and change their direction. This experiment mimics a person trying to position the paddle where the ball would be in a high-speed game.

Each conducted experiment is recorded and publicly available on our GitHub repository [18], stored in comma-separated value files (or *logs*). However, as the player starts and stops the system manually, the logs include data captured while the player moves towards the playground at the beginning, and towards the system to shut it down, at the end. To accurately assess the system behaviour, these preceding and succeeding frames are excluded during evaluation. These details and a label indicating the experiment type are specified in a separate text file within each experiment. In the following subsections, we consider how changing the tracking and reconstruction modules impacts the system performance.

5.1 Tracking

This section investigates different values for the three key parameters identified in Section 4.3 – the point velocity threshold V_p, the dynamic point count threshold N_d, and the target velocity threshold V_t.

Radial Velocity Threshold for Dynamic Points

To label a point as dynamic, its velocity must exceed V_p. The radar's resolution for radial velocity is $0.13\,\mathrm{m/s}$. In

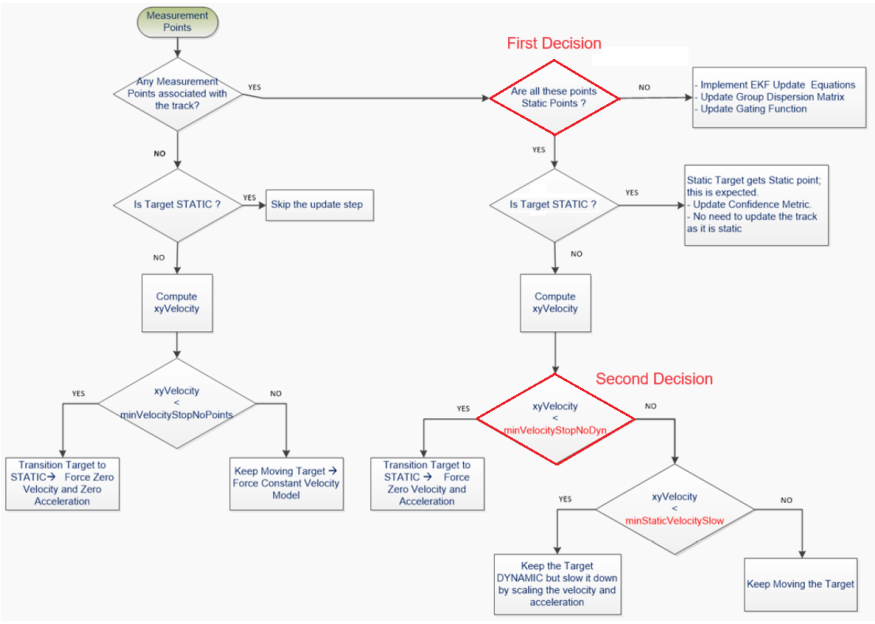


Figure 7: Flow Diagram: Tracker Update Steps. The decision points in red are examined. Image Courtesy of Texas Instruments [16]

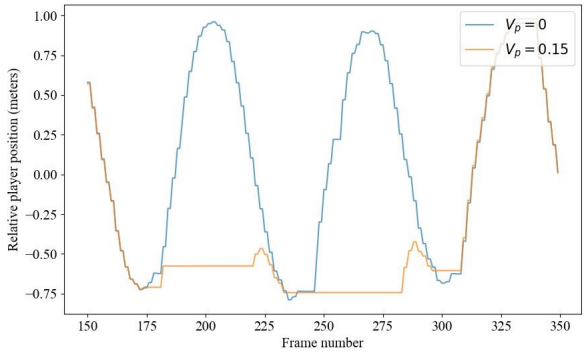


Figure 8: Comparison between the system behaviour across two threshold values V_p, used to determine whether a point is dynamic. Against a dynamic target, with $V_p = 0.15\,\mathrm{m/s}$, the system loses track of the target between frames 175 and 225, and 250 and 290.

other words, all recorded velocities are multiple of $0.13\,\mathrm{m/s}$. Therefore, the system's behaviour changes only once V_p surpasses the next multiple of $0.13\,\mathrm{m/s}$. For instance, $V_p = 0.14\,\mathrm{m/s}$ and $V_p = 0.20\,\mathrm{m/s}$ result in identical point labels, but a value greater than $0.26\,\mathrm{m/s}$ introduces a difference. However, as illustrated in Figure 8, any non-zero V_p results in poor performance when tracking a moving target.

Dynamic Point Count Threshold to Update Estimators

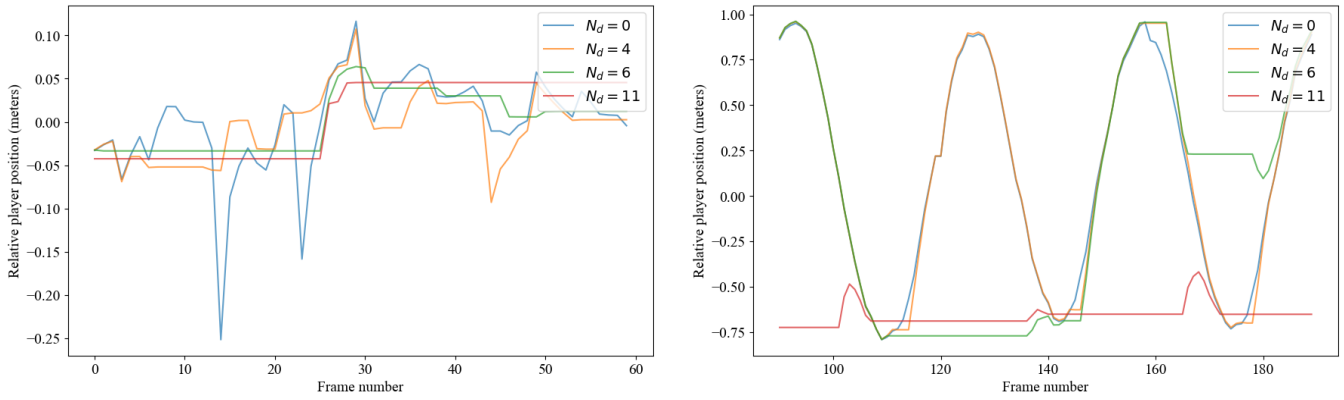
When points are associated with a track, the subsequent decision in Figure 7 is based on whether there are dynamic points among them. Even though the decision poses the question "Are all these points Static Points", GTRACK considers a non-zero threshold N_d on the minimal number of dynamic points necessary to update the state and the corresponding estimators. Intuitively, a higher value results in a more stable estimate on a static target, but introduces a delay or loses a dynamic target. Conversely, a smaller value responds to movements with little delay but produces a noisy estimate on a stationary target. This behavior is visualised in Figure 9, which compares four values. As we can observe, $N_d = 6$ is already large, as the system loses the moving target (b), whereas, with $N_d = 0$, the system performs badly against a non-moving target (a).

Maximal Velocity of Static Target

The velocity of a target is calculated as the average velocity of all points associated with that target. To label a target as dynamic when insufficient dynamic points are recorded, its velocity must be above V_t. Otherwise, it is considered stopping and labeled as static. We examine several threshold values – $0.5\,\mathrm{m/s}$, $0.15\,\mathrm{m/s}$, and $0.04\,\mathrm{m/s}$. While $V_t = 0.5\,\mathrm{m/s}$ is used in GTRACK, the latter values are empirically chosen based on observations of the velocities when a track status changes. For instance, when a moving target is considered stationary, or a static one is detected at a new location. A lower threshold value improves target tracking accuracy, while a higher threshold value reduces the variance in static target tracking. Figure 10 illustrates the system behaviour with the three different thresholds. The system performs worst with $V_t = 0.04\,\mathrm{m/s}$, while with $V_t = 0.15\,\mathrm{m/s}$ and $V_t = 0.5\,\mathrm{m/s}$ exhibits similar behavior in tracking a static target. However, any value higher than $0.04\,\mathrm{m/s}$ is shown to cause misbehavior on dynamic targets.

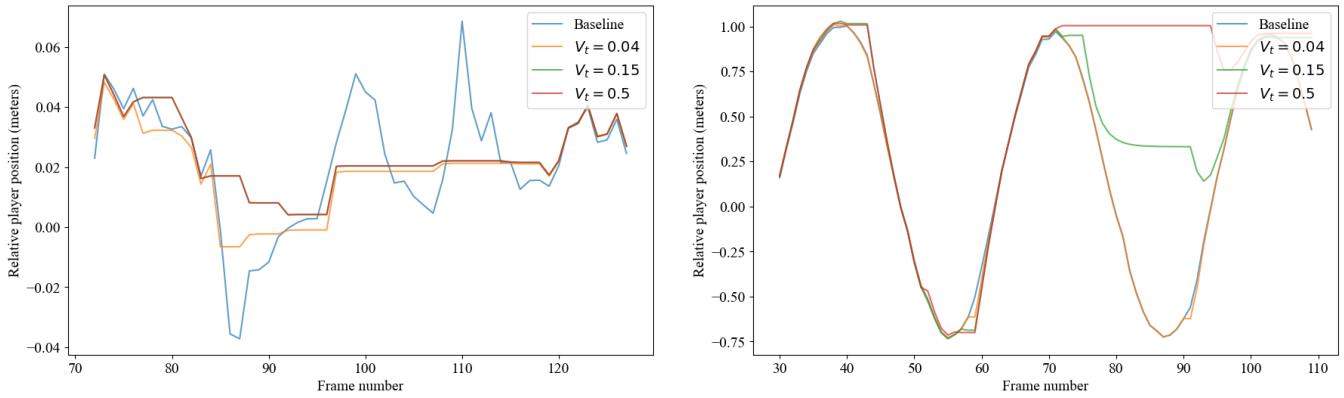
Post-Tuning Comparison

Based on the discussed hyperparameter tuning, $V_p = 0\,\mathrm{m/s}$, $N_d = 4$, and $V_t = 0.04\,\mathrm{m/s}$. Tested on stationary targets, the improved tracking module achieves an average of $3.02\,\mathrm{cm}$ standard deviation in static target tracking. In comparison, the baseline implementation, applied to the same experimental data, results in approximately 1.5 times larger standard deviation, specifically $4.52\,\mathrm{cm}$.



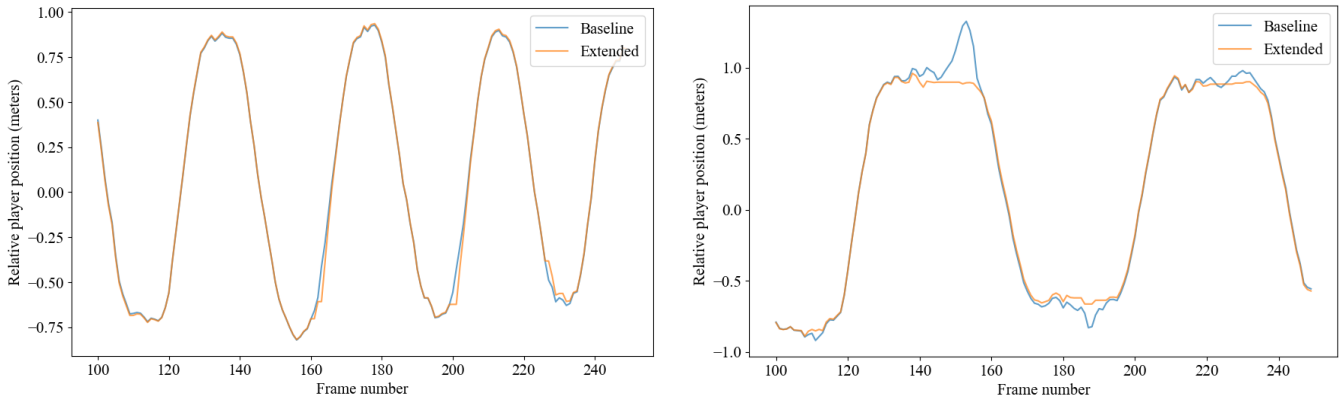
(a) (b)

Figure 9: Comparison of system behaviour across four different threshold values N_d, defining the number of dynamic points necessary to classify a target as dynamic. (a) For a static target, a higher threshold decreases variance. (b) For a dynamic target, a threshold over four leads to losing the target. The experiments are run using $V_t = 0.04\,\mathrm{m/s}$ and $V_p = 0\,\mathrm{m/s}$.



(a) (b)

Figure 10: Comparison between the system behaviour across three different threshold values V_t, defining the maximal velocity allowed on a stationary target. (a) For a static target, a higher threshold decreases variance. (b) For a dynamic target, a threshold value above $0.04\,\mathrm{m/s}$ leads to losing the target. The experiments are run using $V_p = 0.04\,\mathrm{m/s}$ and $N_d = 0\,\mathrm{m/s}$.



(a) (b)

Figure 11: Comparison between the baseline and extended algorithm in dynamic target tracking. (a) The extended algorithm accurately tracks a dynamic target in constant motion. (b) The extended algorithm signal is more stable when the dynamic target temporarily stops.

Evaluated on a dynamic target, the state machine does not hinder the tracking performance. As observed in Figure 11, the system accurately follows a target in motion. In addition, it shows significant improvement in stabilising a target track once it temporarily stops.

5.2 Reconstruction

As discussed in Section 3.1, a low sensor sampling rate and high in-game refresh rate result in snappy paddle movement. Here we investigate two solutions, aiming to smooth the user experience – interpolation and moving average window.

Inter-Frame Interpolation

The Kalman filter predicts where the player will be in the subsequent measurement. Based on the distance between the last measured player position and the prediction, we adjust the paddle, moving it towards the estimated location. Even theoretically sound, this approach shows limited practical improvement. Specifically, against a dynamic target, this technique moves the paddle by 0.97 pixels between measurement frames. It is observed that the velocity estimated by the Kalman filter in the x-axis is relatively small. We believe this discrepancy is the reason behind the poor performance of the regular tracking module, as discussed in Section 4.3.

Moving Average

In contrast to interpolation, averaging multiple past measurements does not rely on estimates. However, as discussed in Section 4.3, it introduces a response-time delay. As shown in Figure 12, using a larger window increases this delay and compromises the system's accuracy. In the experiment, the radar samples a point cloud every $100\,\mathrm{ms}$, while the game screen refreshes every $50\,\mathrm{ms}$. Therefore, there is a single intermediate frame between every two consecutive measurements. The minimal window size required to keep the paddle constantly in motion during user movement is $W_l = 2$, resulting in a 50-ms delay. Generally, updating the game screen every $\frac{1}{R} \times 100\,\mathrm{ms}$ while keeping the radar sampling rate results in $R-1$ intermediate frames. Thus, the minimal window length is $W_l = R$ and the introduced delay is $\frac{R-1}{R} \times 100\,\mathrm{ms}$. In summary, while this approach supports a higher refresh rate, it incurs a delay of up to $100\,\mathrm{ms}$, introducing a trade-off between smooth experience and responsiveness.

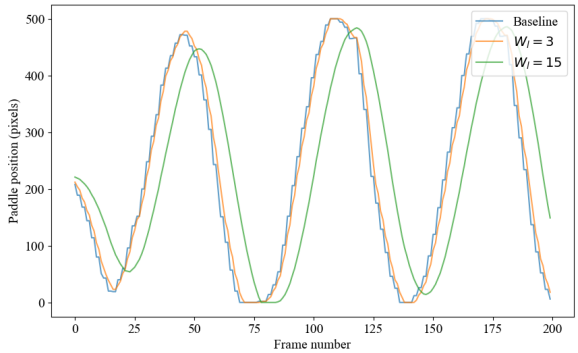


Figure 12: Comparison between paddle position over time across two moving window lengths W_l. In particular, the paddle position is averaged over the last W_l measurements. Against a dynamic target, $W_l = 3$ efficiently smooths the movement, introducing a small delay. The delay with $W_l = 15$ is already noticeable.

6 Discussion and Limitations

Despite the achieved improvements in the tracking module and the proposed interactive video game framework, this research is limited in three aspects. First, the experiments are conducted in a small room where the conditions are different in comparison to a big indoor public area. Fewer dynamic objects (e.g. people), closer distances, and potentially more reflections are closely related to the radar configuration and system implementation. Thus, the optimal parameters found are specific to the development environment, and applying the system elsewhere might require further tuning. Second, limited access to infrastructure, including FMCW radar and transparent screens, did not allow continuous live evaluation and hindered development. For instance, the experiments are conducted at 20 fps and can be only simulated at 10 or 20 fps. Furthermore, the game environment is predominantly visualized on a laptop screen. The visualisation disparity [7] might have hidden problems such as worse user experience due to bigger paddle displacements. Therefore, the game refresh rate should be adapted to foster a smooth user experience based on the screen size and resolution. Last, only one person interacted with the system, and the research work is based on their interactions, which do not represent diverse movement patterns.

7 Responsible Research

This research work is compliant with Chapters 2 and 3 from the Netherlands Code of Conduct [19], as conducted by Kaloyan Fachikov, a student at Technische Universiteit (TU) Delft, under the supervision of Dr. Girish Vaidya and Assoc. Prof. Marco Zuñiga Zamalloa.

Transparency is ensured by reporting the experimental setup, including the environment setting and the performed movement sequence. Elaborating on the process, and extensively explaining each configuration decision, together with open-sourcing the system [14], and the experiments and the evaluation framework (as a Jupyter Notebook) [18], makes the results verifiable. In addition, elaborating on the limitations that hindered our research and their impact on our work helps us determine whether our findings and conclusions are reliable and generalisable. Furthermore, this non-funded research is guided solely by scientific and scholarly considerations, as part of the *CSE3000 Research Project* course at TU Delft, and there are no conflicting interests that could compromise our findings.

As our research involves participants, i.e., people who interact with the system, it is important to ensure that they are aware of the information our system processes. Since mmWave technology uses millimeter waves and reflections, it cannot capture personal information. The sparse point clouds generated hinder accurate environmental reconstruc-

[7] Laptop screen resolution is 1920×1080 pixels, at size $34.5 \times 19.4\,\mathrm{cm}$, while transparent screen resolution is 3840×2160 pixels, at size $144.5 \times 81.3\,\mathrm{cm}$.

tion. Consequently, the experimental data stored during development does not contain identifiable information about the player or details about the surroundings Additionally, during real-time gameplay, the system does not store the generated points. Lastly, the research team is the only entity interacting with the system, thus consent is not required.

8 Conclusions and Future Work

In this paper, we propose an interactive video game system utilising an FMCW radar as a tracking device. Working in the millimeter-wave range, the sensor ensures privacy but generates a sparse point cloud. This introduces several challenges, including noise when tracking a stationary target. Integrating an mmWave radar in an interactive video game poses additional problems, such as lag and delay. Our work shows that a track-state transition module processing the points count and their velocities improves system accuracy. It reduces the standard deviation on a static target location from $4.52\,\mathrm{cm}$ to $3.02\,\mathrm{cm}$ while preserving tracking accuracy on dynamic targets. We compare two reconstruction policies, which allow a higher refresh rate. The moving average, which does not rely on the estimator accuracy, achieves superior improvements in user experience, particularly in smoothing paddle movement. Three possible extensions to our framework are identified. First, to extend the system's capability to support multiplayer video games, work in user identification is needed. Second, a ground truth dataset should be composed and used to quantitatively evaluate the system's performance. Finally, investigating ghost objects and interference removal techniques could significantly improve system performance in crowded public areas.

References

[1] J. Yang, T. Lim, S.-M. Jeong, and S. Ju, "Information-Providing Flexible and Transparent Smart Window Display," *ACS Applied Materials & Interfaces*, vol. 13, pp. 20689–20697, May 2021. Publisher: American Chemical Society.

[2] "*Breakout* (video game)," May 2024. Page Version ID: 1224017580, https://en.wikipedia.org/w/index.php?title=Breakout_(video_game) (accessed Jun. 16, 2024).

[3] "Videowindow." https://www.videowindow.eu (accessed Jun. 21, 2024).

[4] C. Iovescu and S. Rao, "The fundamentals of millimeter wave radar sensors," 2020. https://www.ti.com/lit/wp/spyy005a/spyy005a.pdf (accessed Jun. 16, 2024).

[5] H. Liu, Y. Wang, A. Zhou, H. He, W. Wang, K. Wang, P. Pan, Y. Lu, L. Liu, and H. Ma, "Real-time Arm Gesture Recognition in Smart Home Scenarios via Millimeter Wave Sensing," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, pp. 140:1–140:28, Dec. 2020.

[6] S. An and U. Ogras, "MARS: MmWave-based Assistive Rehabilitation System for Smart Healthcare," *ACM Transactions on Embedded Computing Systems*, vol. 20, no. 5s, 2021.

[7] A. D. Singh, S. S. Sandha, L. Garcia, and M. Srivastava, "RadHAR: Human Activity Recognition from Point Clouds Generated through a Millimeter-wave Radar," in *Proceedings of the 3rd ACM Workshop on Millimeter-wave Networks and Sensing Systems*, mmNets '19, (New York, NY, USA), pp. 51–56, Association for Computing Machinery, Oct. 2019.

[8] Z. Meng, S. Fu, J. Yan, H. Liang, A. Zhou, S. Zhu, H. Ma, J. Liu, and N. Yang, "Gait Recognition for Co-Existing Multiple People Using Millimeter Wave Sensing," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 849–856, Apr. 2020. Number: 01.

[9] P. Zhao, C. X. Lu, J. Wang, C. Chen, W. Wang, N. Trigoni, and A. Markham, "mID: Tracking and Identifying People with Millimeter Wave Radar," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 33–40, May 2019. ISSN: 2325-2944.

[10] Z. Zhang, "Microsoft Kinect Sensor and Its Effect," *IEEE Multimedia - IEEEMM*, vol. 19, pp. 4–10, Feb. 2012.

[11] K. Tanaka, J. R. Parker, G. Baradoy, D. Sheehan, J. R. Holash, and L. Katz, "A Comparison of Exergaming Interfaces for Use in Rehabilitation Programs and Research," *Loading...*, vol. 6, Feb. 2012. Number: 9.

[12] A. Teixeira, A. Assena, A. Santos, M. Moura, N. Gomes, and J. Orvalho, "Usability evaluation of playstation move motion controler," *Proceedings of the International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing*, pp. 276–280, 2014. Accepted: 2023-10-23T11:31:16Z Publisher: [IADIS].

[13] "IWR1443BOOST Evaluation board | TI.com." https://www.ti.com/tool/IWR1443BOOST (accessed Jun. 16, 2024).

[14] K. Fachikov, "kfachikov/mmWave-based-game," June 2024. https://github.com/kfachikov/mmWave-based-game (accessed Jun. 16, 2024).

[15] A. Parlitsis, "AsteriosPar/mmWave_msc," Apr. 2024. https://github.com/AsteriosPar/mmWave_MSc (accessed Arp. 30, 2024).

[16] "People Tracking," May 2023. https://dev.ti.com/tirex/explore/node?node=A__AAA1qiZfNyuf6Vuc4GJbiQ__radar_toolbox__1AslXXD__LATEST (accessed Jun. 16, 2024).

[17] R. J. Hyndman, "Moving averages.," 2011.

[18] K. Fachikov, "kfachikov/mmWave-based-game-experiments," June 2024. https://github.com/kfachikov/mmWave-based-game-experiments (accessed Jun. 16, 2024).

[19] KNAW, NFU, NWO, TO2-Federatie, Vereniging Hogescholen, and VSNU, "Nederlandse gedragscode wetenschappelijke integriteit," 2018.