MIMO OFDM Radar for pose recognition

Tom Kroeze
Supervisors: Dr. Miao Yang and PhD. Nguyen Dao
Radio Systems (RS), University of Twente

July 15, 2025

Abstract

Orthogonal Frequency Division Multiplexing (OFDM) is a highly efficient technique of transmitting data digitally by utilizing multiple orthogonal subcarriers. By applying beamforming to OFDM it becomes achievable to focus the energy of the OFDM signal in specific directions. Performing range, Doppler and angle estimations allows the accurate estimation of target positions, which is further improved by beamforming. In this research, OFDM and beamforming will be jointly investigated to develop a system that is capable of accurately determining human limb motion and body pose. Modeling the interaction between human limbs and the OFDM signal with beamforming enables the extraction of meaningful data for pose recognition. The system successfully detected a squatting position using a coarse human model, showing that signal-processing applications have promising potential for pose recognition. Results demonstrate a computationally efficient method of pose recognition under constrained conditions. By using more advanced signal-processing techniques, current limitations in effective range and angular resolution can be mitigated.

1. Introduction

Human pose recognition is an area of research that has been growing due to the many possible applications, such as in health care, surveillance, behavioral monitoring and gesture interpretation [2]. The research of this thesis is part of bigger research, which investigates pose and breathing pattern recognition to make it possible to track respiratory diseases. Current systems often make use of regular and infra-red (IR) cameras to perform human pose recognition. However, the functionality of these systems depend on proper lighting conditions and especially regular cameras invade the privacy of humans. Unlike camera-based systems, radar based systems are not affected by lighting conditions and interfere less with the user's privacy [2].

When utilizing radar-based systems for human pose recognition, the human body is modeled as a set of points — hereafter referred to as targets — where each target corresponds to major joints such as the hips, the elbows and the shoulders. The radar transmits a signal, which will reflect off these targets and return to the receiver. The reflections of these targets can be processed to obtain information about the angle, range, and velocity of the joints.

Orthogonal frequency division multiplex-

ing (OFDM) is a digital modulation technique that uses many orthogonal subcarriers which are closely spaced in frequency and carry data at low symbol rates. The orthogonality prevents any interference between the subcarriers, making it possible to transmit many subcarriers on a specific bandwidth. Because OFDM signals can transmit efficiently on a large bandwidth, it is possible to achieve a high range resolution which is required for detailed pose recognition.

Additional to the high spectral efficiency of OFDM, a Multiple Input Multiple Output (MIMO) antenna-array is used. MIMO radar systems bring several advantages, mainly improving spatial resolution and offering the possibility to use beamforming for angle estimation — essential for pose recognition.

When transmitting an OFDM signal, it is essential that the subcarriers remain orthogonal and do not interfere with each other. This interference is called inter-carrier interference (ICI) and can be mitigated by using an interleaved structure. The technical details are described in section 2.2.4

Signal processing is utilized to interpret the receiver data. Fourier transforms make it possible to estimate the range and the velocity of targets. To obtain reliable target detection, data is processed further by filtering and Constant False Alarm Rate (CFAR) as detection method. The ability to apply signal processing brings several advantages compared to the data processing camera-based systems require. This enables using relatively simple computations to process data, resulting in a smaller data footprint compared to the large amount of data processing cameras rely on.

By applying beamforming techniques enabled by a MIMO system, it is possible to estimate angular position of received signals which is required for accurate target detection. Beamforming enables the radar to focus on specific directions by electronically steering the radiation pattern towards the direction of targets, making it possible to rapidly control it without any additional mechanical implementation.

Literature review Radar pose estimation has already been explored in literature such as in RF-Based 3D Skeletons [6] and Through-Wall Human Mesh Recovery Using Radio Signals [7]. In these previous works the benefits of using radar application are clearly proven and functional methods of pose recognition have been obtained. In this thesis the pose recognition relies heavily on signal processing, and does not depend on deep learning methods as described in other literature sources creating less dependency on the usage of large amounts of data. Additionally, in this thesis a dedicated OFDM radar system is used instead of a commercial WiFi-based system. This enables occupying a much larger bandwidth, improving specifications such as range- and spatial resolution.

Report structure The thesis is structured into multiple sections.

Section 2 contains the theoretical background where relevant literature is analyzed. Concepts such as DOA beamforming, range and Doppler estimation are explored to create a solid theoretical framework, which is used in later sections to successfully create a sufficiently realistic model to perform pose recognition. Building on the theoretical framework, section 3 contains the implementation of the OFDM MIMO radar system in MATLAB. Separate DOA estimation and range- and Doppler estimation pipelines have been created, both being able to function in reasonable error margins. Even though full integration has not been achieved, the implementation clarified challenges in the modeling of a practically realistic system.In section 4 human measurements were conducted. Although the human measurement were not

used to test the radar system on, conducting the human measurements still provided insights into what constraints would exist in a real-life application of the system. In section 5, validation of the DOA estimations and the range- and Doppler estimations are conducted. This section also investigates the ability to perform pose recognition by detecting a squatting pose of a coarse human model. Despite the lack of a fully integrated DOA and range-Doppler estimation pipeline, performing these estimations made it possible to detect the squatting pose within a limited range of approximately 0.9-1.3 meters due to constraints in angular resolution. In section 7, an overall summary of the quality and functionality of the developed pose-recognition system is provided by reflecting on the achieved result of estimating a squatting position. Furthermore, this summary and reflection relate to the potential future and feasibility of signal-processing models for pose recognition.

2. Theoretical Background

2.1 Notation

Bold symbols such as $\boldsymbol{A}$ denote matrices. When a matrix is written as $\boldsymbol{A}(i,j)$, the lowercase symbols represent the indices of these matrices. The corresponding uppercase symbols indicate the dimension of the matrix, so $\boldsymbol{A} \in \mathbb{C}^{I \times J}$, where each element of matrix $\boldsymbol{A}$ is described by $a_{i,j}$, denoting the entry at the i-th row and the j-th column. Vectors are described with either regular or capital letters with an arrow, such as $\vec{k}$ and $\vec{A}$. Furthermore, symbols with a hat such as $\hat{k}$ and $\hat{l}$ denote index positions of the corresponding vectors $\vec{k}$ and $\vec{l}$.

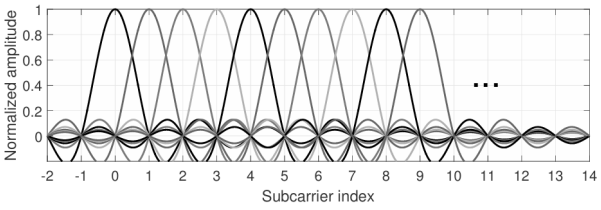


Figure 1: Visualization of OFDM subcarriers [4]

2.2 OFDM

2.2.1 General idea of OFDM

OFDM is a technique that is used for efficient transmission of data. By using multiple frequency bins that all represent a different frequency component and modulating multiple signals on these different sampling frequencies. To ensure that there is orthogonality between the subcarriers, the equation

$$f_n = n\Delta f = \frac{n}{T_0}, n = 0, 1...N-1 \quad (1)$$

must hold, where n is the subcarrier index, N is the total amount of subcarriers, Δf is the subcarrier spacing and f_n is the frequency of subcarrier n resulting in non-overlapping subcarriers and symbols [4]. This is properly depicted in Figure 1. At the peak of each individual subcarrier, all other subcarriers have a value of 0 1.

2.2.2 QAM and OFDM signal generation

To generate M OFDM symbols all carrying N subcarriers for transmission, a digital stream of bits is mapped onto a modulation scheme such as Quadrature Phase Shift Keying (QPSK) or Quadrature Amplitude Modulation (QAM). 16-QAM is used for this thesis which offers higher spectral efficiency than QPSK, transmitting 4 bits per symbol instead of 2. Modulation schemes with a higher order such as 64-QAM are avoided, which have increased complexity and SNR

demands. These QAM symbols are represented in matrix form [4].

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,M-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,M-1} \end{bmatrix} \in \mathbb{C}^{N \times M}$$

Where element $a_{n,m}$ denotes subcarrier n with $n = 0, 1, ..., N-1$ of OFDM symbol m where $m = 0, 1..., M-1$. By taking an Inverse Fast Fourier Transform (IFFT) the transmittable OFDM symbols are created. The time domain signal can be written as [4]

$$x(t) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} a_{n,m} e^{j2\pi f_n t} rect(\frac{t - mT_0}{T_0}) \tag{2}$$

Where T_0 is the OFDM symbol duration.

2.2.3 Cyclic prefix

Due to reflection from surrounding objects and the relative target positions, the transmitted signal could arrive at the receiver through multiple paths. This results in delays that create interference and overlap between the OFDM symbols. This phenomenon is called Inter Symbol Interference (ISI) and has a more significant effect at lower symbol durations. This is the case when OFDM symbols contain many subcarriers with fixed frequency spacing, which is required to obtain high range resolution.

causing a larger ISI. This can be prevented by adding a guard interval, by appending the OFDM symbols with a Cyclic Prefix (CP) before transmission and removing them at the receiver [3]. If the CP length is long enough, ISI will remain in the guard interval and will not distort the data-carrying part of the OFDM symbols. However, the CP data does not contain information thus the CP should not be too long. Furthermore, addition of CP does not change the signal model described in equation 2

2.2.4 MIMO and Interleaved structure

A MIMO radar system consists of multiple transmitters and receivers, creating a large virtual array which improves spatial resolution, required for accurate target detection. Additionally, MIMO radar enables beamforming applications, a highly efficient technique to perform angle estimation required to detect and localize closely spaced body parts such as different joints. However, using MIMO radar increases the system's complexity. When all transmitters of a MIMO radar transmit over the same subcarriers where each signal takes a different propagation path, phase and timing mismatches will occur at the receiver, distorting the orthogonality of the OFDM symbols. This phenomenon is called Inter Carrier Interference (ICI) and is prevented by using an interleaved structure. The subcarriers are spectrally interleaved as described in [4] so that the efficient bandwidth is retained. By dividing the N subcarriers over N_{Ch} channels which are indexed by user index $u = 0, 1, ..., N_{Ch} - 1$, each channel will be assigned $N_u = \frac{N}{N_{Ch}}$ subcarriers. The total subcarriers per channel N_u are assigned with the equation $n_u = u + iN_{Ch}$, where n_u is an element of N_u where $i = 0, 1, ..., \frac{N}{N_{Ch}} - 1$. The interleaving is visualized in Figure 2

2.3 Antenna geometry

2.3.1 Virtual array

For a MIMO radar system containing P transmitters (Tx) and Q receivers (Rx), each transmitter's signal is seen by each receiver, increasing the effective number of signal paths to $P \times Q$ [4]. Additionally, this results in a larger effective antenna array —

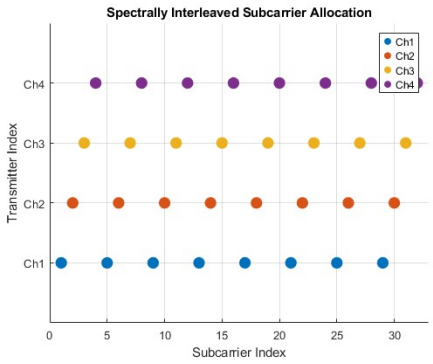


Figure 2: Visualization of subcarrier allocation for interleaved structure

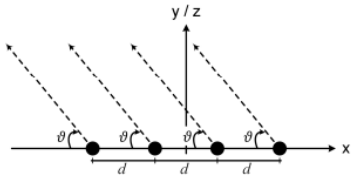


Figure 3: Visualization of phase shift incurred based on DOA [4]

referred to as a virtual array further on—which increases spatial resolution. The resulting $P \times Q$ signal paths makes it possible to implement beamforming which is one of the main motivations for using a MIMO radar system. Beamforming enables direction-of-arrival (DOA) estimation which allows distinguishing between targets that are at similar distances from the radar. The accuracy of the DOA estimation is further improved by the increase in spatial resolution.

2.3.2 Monostatic uniform linear array (ULA)

A ULA antenna consists of multiple antenna elements, lying on the same axis spaced at distance d as shown in Figure 3. The simplicity of this geometry allows straightforward application of phase shifts across the antenna elements and simplifies DOA estimation. Because all the elements lie on the same axis, using a ULA can only perform azimuth angle estimation which makes it impossible to perform 3D DOA estimation. However, using a ULA is acceptable for proper range and Doppler estimation. In this thesis it is assumed that the distance between the Tx and Rx elements can be neglected because it is much smaller than the distance between the radar and the human. Because the Tx and Rx are assumed to be co-located and to have identical physical geometries, the radar system is considered to be monostatic [4].

2.3.3 Monostatic Uniform Planar Array (UPA)

To perform 3D DOA recognition, the ULA model must be extended to a more complex model. A UPA array can be visualized as multiple ULA antenna arrays stacked on top of each other, creating a 2D array which is visualized in Figure 4. Similar to the ULA, this configuration is still easy to model, which still allows simple DOA estimation.

2.4 System overview

Modeling Assumption on Antenna Array Geometry In this thesis, for the range and Doppler processing a ULA is used. The antenna configuration does not affect these estimations, other than that the accuracy improves when using more Tx and Rx. However, because a UPA is required for DOA estimation, a UPA setup is used for the theoretical model.

2.4.1 Scenario

The system consists of a monostatic UPA, containing 16×16 Tx elements and 4×4 Rx elements with $d = \frac{\lambda}{2} = \frac{c_0}{2f_c}$, where d is the element spacing and f_c is the carrier frequency. The spacing d is chosen to avoid grating lobes [4]. A carrier frequency of $f_c = 24GHz$ is used as elaborated in Section 5, resulting in

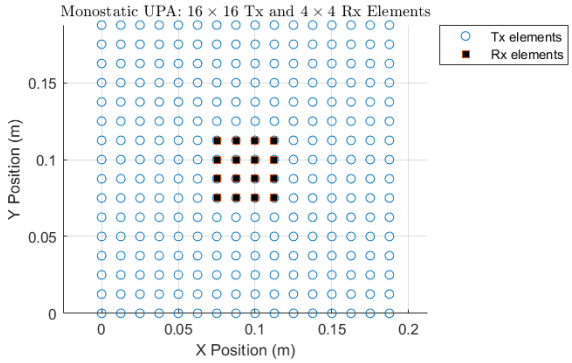
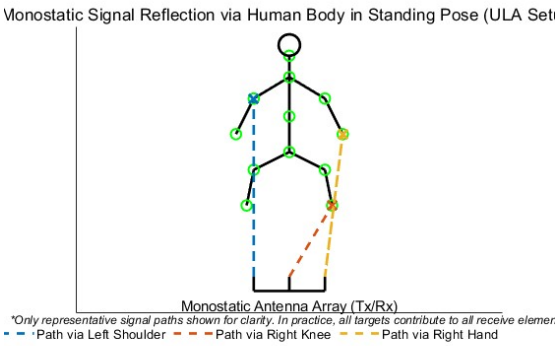


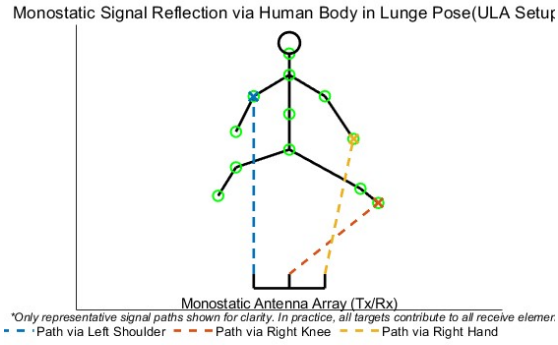
Figure 4: Monostatic UPA antenna array with $d = \frac{\lambda}{2} = 0.0125m$ spacing between elements.

an element spacing of $d = \frac{3 \cdot 10^8}{2 \cdot 24 \cdot 10^9} = 0.0125m$. This antenna setup is shown in Figure 4.

As illustrated in Figure 5a and 5b, different human poses result in different signal paths. This results in the radar receiving different signals depending on the pose of the human, enabling pose recognition by evaluating the received data. These Figures represent a ULA antenna array, however the visualization of the signal paths remains identical for a UPA antenna array. The amount of points used in estimations in visualized in Section 5. By testing with several poses and observing different outputs, it is possible to perform pose recognition for these tested poses without relying on machine learning. However, this visualization does lack in showing the effect of dynamic poses such as jumping and waving. In such cases, Doppler estimation becomes essential for pose recognition.



(a) Monostatic signal reflection via human body in standing pose with a uniform linear array (ULA).



(b) Monostatic signal reflection via human body in lunging pose with a uniform linear array (ULA)

Figure 5

2.5 Channel

In this section, real-world transmission effects will be modeled and accounted for. This is done step by step to construct a full signal model.

2.5.1 Propagation delay phase shift

Because the signal travels towards and away from the target, a propagation delay phase shift is induced, visualized in Figure 6. Because the MIMO radar system is monostatic,

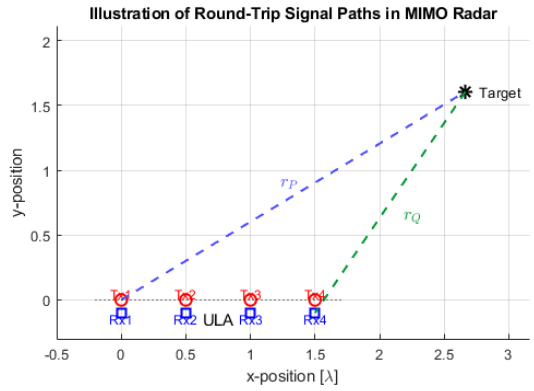


Figure 6: Propagation delay visualization

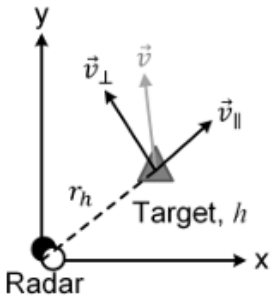


Figure 7: Detectable Doppler shift visualization [4]

$r_p = r_q = r_h$ making the total round-trip distance $r_p + r_q \approx 2r_h$. The round-trip time can be expressed as $\tau_h = \frac{2r_h}{c_0}$, where c_0 is the speed of light. Accounting for the subcarrier spacing Δf, the incurred propagation delay phase shift is equal to

$$e^{-j2\pi f_n \tau_h} \tag{3}$$

However, due to the interleaved structure, each transmitter transmits on subcarriers spaced by a factor N_{ch}. This scales the phase shift by a factor $u+iN_{ch}$, providing the following adapted phase shift

$$\phi_\tau = e^{-2\pi j f_n \tau_h (u+iN_{ch})} \tag{4}$$

This can be separated into two terms where one term depends on the user index u and the other term depends on the subcarrier index of the channel iN_{ch}.

$$\phi_\tau = e^{-j2\pi f_n \tau_h u} \cdot e^{-j2\pi f_n \tau_h i N_{ch}} = \phi_{\tau,u} \cdot \phi_{\tau,i} \tag{5}$$

Because $\phi_{\tau,u}$ is a constant phase shift over all subcarriers in the channel, it will not affect range estimation [4]. This makes it a valid assumption to neglect the effect of $\phi_{\tau,u}$, meaning $\phi_\tau = \phi_{\tau,i}$. Expressing ϕ_τ in vector form enables range estimation, which is required to determine the humans position relative to the radar. Being able to estimate the range of the targets is crucial for pose recognition, as different poses will contain different range profiles.

$$\vec{k}_{r_h} = \begin{bmatrix} 0 & e^{-j2\pi N_{Ch} \Delta f \tau_h} & \dots & e^{-j2\pi \left(\frac{N}{N_{Ch}}-1\right) N_{Ch} \Delta f \tau_h} \end{bmatrix} \tag{6}$$

2.5.2 Doppler phase shift

To perform accurate pose recognition, the radar system must be able to detect velocities of targets. When targets are in motion, a Doppler phase shift is introduced which can be detected by the radar. This allows velocity estimation of the parallel velocity component $\vec{v_\parallel}$ as shown in Figure 7, since the perpendicular component can not be observed by the radar. Since the signal travels towards and away from the target, the radar detects a Doppler shift corresponding to $v_{est} = 2 \cdot \vec{v_\parallel}$ This Doppler shift is presented in the work of Sit (2017) [4]

$$\phi_D(t) = e^{j2\pi (f_n + f_c) \frac{2v_{est}}{c0} t} \tag{7}$$

Separating this into two terms results in

$$\phi_D(t) = e^{j2\pi f_n \frac{2v_{est}}{c_0} t} \cdot e^{j2\pi f_c \frac{2v_{est}}{c_0} t} = \phi_{D,f_n}(t) \cdot \phi_{D,f_c}(t) \tag{8}$$

Because in this thesis the carrier frequency is in the order of GHz resulting in $f_n <<<$

f_c, ϕ_{D,f_n} can be neglected. So the total incurred Doppler shift is

$$\phi_D(t) = \phi_{D,f_c}(t) = e^{j2\pi f_{D,h} t} \tag{9}$$

where $f_{D,h} = \frac{f_c \cdot 2v_{est}}{c_0}$ The problem with this equation however, is that it assumes continuous time. To correctly apply this phase shift on each OFDM symbol, t is made discrete. The symbol time T_s can be expressed as $T_s = \frac{T}{N+CP}$ [4], so $T = T_S(N + CP)$. Using this discrete expression of t, $\phi_D(t)$ can be written in vector form

$$\vec{k}_{\phi_D} = \begin{bmatrix} 0 & e^{j2\pi f_{D,h} T} & \dots & e^{j2\pi f_{D,h}(M-1)T} \end{bmatrix} \tag{10}$$

2.5.3 Free space path loss (FSPL)

When the OFDM signal travels through space, it suffers from FSPL, reducing the amplitude of the signal when arriving at the receiver. This loss can be computed with Friis transmission equation

$$P_R = \frac{P_T G_T G_R \lambda_c^2}{(4\pi r_h)^2} \tag{11}$$

Where P_R and P_T is the signal power at the receiver and transmitter, and G_T and G_R are the antenna gains of the transmitter and the receiver. To model two-way path loss (TWPL), the power of the signal at the target is

$$P_{target} = \frac{P_T G_T \lambda_c^2}{(4\pi r_h)^2} \tag{12}$$

And the power at the receiver after reflection is

$$P_R = P_{target} \frac{\sigma}{(4\pi r_h)^2} G_R \tag{13}$$

Where σ is the radar cross section (RCS) and depends on the reflectivity of the target. Because in section 4 all the targets are modeled with identical sensors a value of $\sigma = 1$ is assumed. By substituting equation 12 in 13, and taking the square root of this expression

P_{Tx}	Transmit power
G	Total transmit and receive antenna gain
f_c	Centre frequency
B	Signal bandwidth
$k_B T$	Boltzmann's constant times receiver noise temperature
NF	Total receiver chain noise figure (including digital acquisition)
d	Target distance
σ_{RCS}	Target radar cross section

Figure 8: SNR parameters [1]

to find attenuation factor α_h the equation becomes

$$\alpha_h = \sqrt{\frac{\lambda_c^2}{(4\pi)^3 r_h^4} G_T G_R} \tag{14}$$

2.5.4 White noise

SNR The signal to noise ratio (SNR) is a metric that determines systems efficiency, depending on factors shown in Figure 8 [1]. These factors are important to consider when creating a model for the OFDM radar system. However, full hardware and system modeling of the antennas is outside the scope of this thesis. This is why a fixed SNR of $30dB$ is set for this research.

Modeled noise To make the theoretical model more realistic zero-mean complex Gaussian noise is added. The value of this noise is dependent on the noise power, expressed as

$$P_{noise} = \frac{P_{signal}}{10^{\frac{SNR_{dB}}{10}}} \tag{15}$$

The Gaussian noise is then modeled as

$$z(t) = \sqrt{P_{noise}} \cdot w(t) \tag{16}$$

Where $w(t)$ is considered random Gaussian noise, with zero mean and unit variance.

2.5.5 Full channel model

The full channel $\boldsymbol{H}_{qp}(n_u, m)$ where qp represent the propagation path from p to q,

where p is the receiver element so that $p = 0, 1..., P-1$ and q is the transmitter element so that $q = 0, 1, ..., Q-1$. The channel accounts for all targets, including the range- and Doppler phase shift and attenuation factor can be written in an equation as

$$\mathbf{H}_{qp}(n_u, m) = \sum_{k=0}^{K-1} \alpha_h \cdot e^{-j2\pi f_n \tau_h i N_{ch}} \cdot e^{j2\pi f_{D,h} m T} \tag{17}$$

Where K is the total amount of targets and k represents the individual targets.

2.6 Beamforming

2.6.1 Antenna geometry phase shift

Because of the UPA geometry of the MIMO radar system, a phase shift depending on the location of the Tx and Rx elements is incurred. An array factor is modeled where a

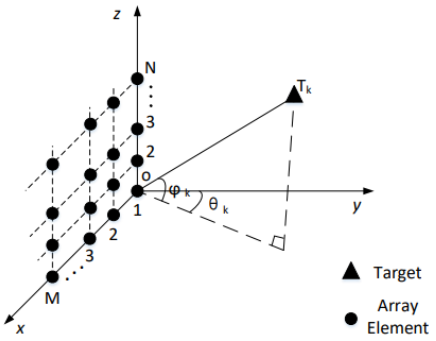


Figure 9: UPA Geometry visualizing both azimuth angle ϕ and elevation angle θ [5]

phase shift is incurred dependent on the location of each element, visualized in Figure 9. This array factor can be expressed as

$$AF(\theta, \phi) = \sum_{n_x=0}^{N_x-1} \sum_{n_y=0}^{N_y-1} e^{jk_c d(n_x \sin(\phi)\cos(\theta) + n_y \sin(\phi)\sin(\theta))} \tag{18}$$

Where N_x is the total amount of horizontal Tx elements, N_y the total amount of vertical Tx elements, M_x is the total amount of horizontal Rx elements, M_y the total amount of vertical Rx elements. Furthermore, n_y, n_x, m_y and m_x represent the individual elements and $k_e = \frac{2\pi}{\lambda_c}$ is the wavenumber. The difference in phase per transmitter and receiver elements enables implementation of beamforming. Equation 18 can also be written as two separate steering vectors for all the transmitters and receivers, assuming $P = N_x \times N_y$ and $Q = M_x \times M_y$. The size of these vectors are$\vec{a}_{\text{Tx}}(\phi, \theta) \in \mathbb{C}^{P\times 1}$ and $\vec{a}_{\text{Rx}}(\phi, \theta) \in \mathbb{C}^{Q\times 1}$. The steering vector of the transmitter can be written as

$$\vec{a}_{\text{Tx}}(\theta_h, \phi_h) = \begin{bmatrix} 1 \\ e^{jk_e d(1\cdot\sin(\phi_h)\cos(\theta_h))} \\ e^{jk_e d(2\cdot\sin(\phi_h)\cos(\theta_h))} \\ \vdots \\ e^{jk_e d((N_x-1)\cdot\sin(\phi_h)\cos(\theta_h)+(N_y-1)\cdot\sin(\phi_h)\sin(\theta_h))} \end{bmatrix} \tag{19}$$

Similarly, the steering vector of the receiver can be written as

$$\vec{a}_{\text{Rx}}(\theta_h, \phi_h) = \begin{bmatrix} 1 \\ e^{jk_e d(1\cdot\sin(\phi_h)\cos(\theta_h))} \\ e^{jk_e d(2\cdot\sin(\phi_h)\cos(\theta_h))} \\ \vdots \\ e^{jk_e d((M_x-1)\cdot\sin(\phi_h)\cos(\theta_h)+(M_y-1)\cdot\sin(\phi_h)\sin(\theta_h))} \end{bmatrix} \tag{20}$$

2.6.2 MRT beamforming for 3D DOA estimation

To steer the signal towards the targets, maximum ratio transmission (MRT) beamforming is implemented. This method applies beamforming at the transmitter and the receiver, and scans over a range of angles enabling visualization of the radiation pattern, allowing DOA estimation. First, the transmit and receive steering vectors will be applied to the channel.

$$\mathbf{H}_h(n_u, m) = \sum_{k=0}^{K-1} \alpha_h \cdot e^{-j2\pi f_n \tau_h} \cdot e^{j2\pi f_{D,h} m T} \cdot \vec{a}_{\text{Rx}}(\phi, \theta) \cdot \vec{a}_{\text{Tx}}^{H}(\phi, \theta) \tag{21}$$

These steering vectors $\vec{a}_{\text{Rx}}(\phi, \theta)$ and $\vec{a}_{\text{Tx}}(\phi, \theta)$ create a phase shift dependent on the angle of

the target in the channel, which can be used for 2D DOA estimation. Introducing conjugate transpose steering vectors $\vec{w}_{\mathrm{Rx}}(\phi_s, \theta_s) = \vec{a}^H_{\mathrm{Rx}}(\phi, \theta)$ and $\vec{w}_{\mathrm{Tx}}(\phi_s, \theta_s) = \vec{a}^H_{\mathrm{Tx}}(\phi, \theta)$, enables plotting a channel response for different values of scanning angles ϕ_s and θ_s. This results in a radiation pattern described by

$$P(\phi_s, \theta_s) = |\vec{w}_{\mathrm{Rx}}(\phi_s, \theta_s) \cdot \mathbf{H}_h(n_u, m) \cdot \vec{w}_{\mathrm{Tx}}(\phi_s, \theta_s)|^2 \tag{22}$$

By sweeping over the angles $-90 < \phi_s < 90$ and $0 < \theta_s < 90$, the radiation pattern will show peaks for combinations where $(\phi_s, \theta_s) \approx (\phi, \theta)$.

2.7 Full signal model

To be able to calculate the range and velocity of each target, the full signal model is constructed in this section. Because the DOA estimation does not have effect on the range or Doppler, the effects of the beamsteering vectors will be neglected in the full signal model. This makes it a valid assumption to use equation 17 for the full signal model [4]. Coherently summing the data of all transmitters at each receiver gives

$$\begin{aligned} \mathbf{Y}_q(n, m) = &\sum_{p=0}^{P-1} \mathbf{X}_p(n_u, m) \cdot \mathbf{H}_{qp}(n_u, m) \\ &+ \mathbf{Z}(n, m) \end{aligned} \tag{23}$$

Where $\mathbf{Y}_q(n, m)$ is the data obtained by receiver q, $\mathbf{X}_p(n_u, m)$ is the transmitted data by transmitter p, $\mathbf{Z}(n, m)$ is the added white noise and $\mathbf{H}_{qp}(n_u, m)$ is the channel without any DOA effects.

2.8 Obtaining the channel effects

Based on the assumptions in [4], the following quotient matrix —a quotient of the input and output symbols-at antenna index p, q at subcarrier n_u, symbol m— can be realized

$$\boldsymbol{D}_{qp}(n_u, m) = \sum_{k=0}^{K-1} \alpha_h \left(\vec{k}_{r_h} \circledast \vec{k}_{D_h} \right) + \boldsymbol{Z}(n_u, m) \tag{24}$$

Because the Doppler shift vector is dependent on phase shift along the direction of the symbols and the range phase vector is dependent on the user index, they are orthogonal to each other described by the kronecker product $\circledast$. By taking IFFT's across the axis of the subcarriers and FFT's across the axis of the symbols of $\boldsymbol{D}_{qp}$, it is possible to estimate the range and the velocity of targets [4].

2.9 CFAR

To identify targets and distinguish them from background noise, a reliable thresholding method must be implemented. For this research, Constant False Alarm Rate (CFAR) detection was implemented. This method of detection will scan over each range-Doppler bin —except the ones near to the edge— on the RD-map by surrounding the tested range-Doppler bin, also called Cell Under Test (CUT) with guard- and training cells as shown in Figure 10. The guard cells are spaced around the CUT and are neglected by the scan, because an actual target that has high energy could leak energy in near frequency bins. After this, the average power of the CUT and all the training cells is computed and compared. If the average power of the CUT appears to be a certain threshold higher than the average power of the training cells, the CUT is marked as a target. This method becomes less effective around the edges of the RD-map, because there will be less available training and guard cells. However, this is not a significant issue due to none of the targets being able to appear around the edge.

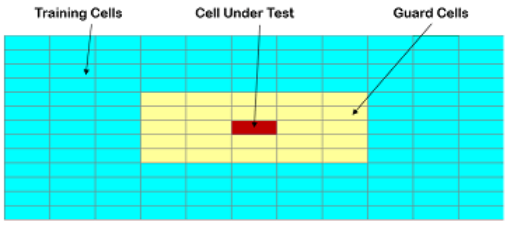


Figure 10: Representation of CFAR on a range-Doppler map

2.10 Range and Doppler estimation

2.10.1 Range

The range can be estimated by taking the IFFT across the axis of the frequency subcarriers of $\boldsymbol{D}_{qp}$. This results in each target k showing a peak at one specific subcarrier representing a range. The subcarrier where this peak occurs denoted by $\hat{k}_k$ where $\hat{k}_k$ is in the set $= \{0, 1, ..., \frac{N}{N_{Ch}} - 1\}$. This can be written in an equation as [4]

$$\hat{k}_k = N\Delta f \tau_h = \frac{2N\Delta f}{c_0} r_h, \tag{25}$$

Expressing this in r_h results in the range of the corresponding target

$$r_h = \frac{c_0}{2N\Delta f} \cdot \hat{k}_k \tag{26}$$

This equation shows that range is dependent on round trip time τ_h, the total amount of subcarriers N, subcarrier spacing Δf and the index $\hat{k}_h$,

Range resolution Because $\hat{k}_k$ is an integer, only multiples of $\frac{c_0}{2N\Delta f}$ can be detected as valid ranges. This means the range resolution is equal to

$$\Delta r_h = \frac{c_0}{2N\Delta f} = \frac{c_0}{2BW} \tag{27}$$

Where BW is the total occupied bandwidth of the system. The required value of Δr_h will be discussed in the results.

2.10.2 Doppler estimation

The Doppler can be estimated by taking a Fast Fourier Transform (FFT) over the symbol axis of $\boldsymbol{D}_{qp}$, where each target k will result in a peak at one specific OFDM symbol m. The subcarrier where this peak occurs denoted by $\hat{l}_k$ where $\hat{l}_k$ is in the set $\{0, 1, ..., M-1\}$. This can be written in equation as [4]

$$\hat{l_k} = M f_D T = \frac{2MTf_c}{c_0} v_{est} \tag{28}$$

Expressing this in v_{est} results in the velocity of the corresponding target

$$v_{est} = \frac{c_0}{2MTf_c} \hat{l_k} = \frac{1}{2MT\lambda_c} \hat{l_k} \tag{29}$$

The estimated velocity depends on the total amount of OFDM symbols M, carrier frequency f_c and symbol duration T. Similar to the range estimation, each object h will give one individual peak at $\hat{l}_k$ along the axis of the OFDM-symbols.

Doppler resolution Similar to the range resolution, only multiples of $\Delta f_D = \frac{1}{MT}$ can be detected as valid Doppler shifts. The required value of Δf_D will be discussed in the results.

3. Implementation

MATLAB was used to create the model of the OFDM beamformed radar. The used code can be found in the section 9. MATLAB has a lot of convenient toolboxes for signal processing and radar modeling and can make computations with large amount of data, which is ideal for this application. All the written code is given in Appendix A. The main code is divided into multiple sections, this pipeline contains the implementation of the range and Doppler estimation. In this pipeline, several functions are used. These functions are

the QAM-mod function, the OFDM function and the OFDM-channel function. In an additional script beamforming is implemented, making use of the function beamsteering vector. The beamforming and the range- and Doppler estimation code are discussed separately. In section 9.11, a block diagram of the full implementation system architecture is shown. It should be noted that this is an idealized representation of the block diagram, because the beamforming has been implemented separately and is not functional in the complete pipeline. This block diagram will be referred to further on in this thesis, giving more detailed descriptions for some blocks where possible.

3.1 Range- and Doppler estimation code

In this subsection, the implementation of the range- and Doppler estimation. The code contains multiple sections, each discussed separately. A block diagram is given to visualize what each part of the code does.

3.1.1 Main code section 1-3: Parameters and signal generation

In these sections parameters are defined and the signal to transmit will be generated. This is done by using the QAM-mod function, which creates 16-QAM symbols ready for transmission. These subcarriers also get assigned to the right channels and transmitters to match the interleaved structure and the amount of Tx. Dividing the channels over the Tx is done by using a round-robin structure. These ready to transmit signals get passed to the next section where the channel effects during transmission will be modeled.

3.1.2 Main code section 4: Defining targets and channel

For testing purposes, multiple test targets are created in this section. This is implemented by assigning each target a specific realistic Doppler shift and a certain distance it is away from the radar. For each transmitter which contains different channels, the channel effects are applied to each target. The channel is a separate function called OFDM-Channel.

OFDM-Channel In the channel, real world effects are modeled as described in section 2.5. Because the transmitted signal is in the time domain, the Doppler phase shift is applied first. After this the signal is briefly converted back to the frequency domain where the range phase shift is applied on the data-carrying subcarriers, after which it gets converted to the time domain again. Due to a time constraint in implementation the phase shift due to the array geometry is not applied here, because DOA estimation is done separately. Finally, the FSPL and noise with a fixed randomness is applied to the signal to account for the environment. The signal with the channel effects applied is then passed onto the next section.

3.1.3 Main code section 5:9 Receiving and processing the data

The transmitted data that passed through the channel is received and processed in these sections. Firstly, because there are no Rx/Tx specific effects, the data of all the transmitters will be summed at the receiver to improve signal strength. A change in implementation for receiving the data was made. Firstly, this was done by simple division as explained in section 2.8. However, this sometimes resulted in errors because division by 0 would frequently occur. This is why matched filtering has been implemented instead. This

is implemented in MATLAB by convolving the transmitted signal and the received signal, with MATLAB's conv2 function. However, in the event of spectral leakage where the energy of a frequency bin leaks into closely spaced frequency bins, regular thresholding based on power after applying matched filtering is not sufficient. Regular thresholding will require frequent adjusting and will often lead to false target detections. To ensure accurate detection of targets, CFAR detection and range filtering is implemented.

MATLAB implementation of CFAR
In the implementation an algorithm is created to perform CFAR. MATLAB's CFAR function was not integrating well with the pipeline of the already written code, which is why this own algorithm was designed. This algorithm properly creates a 'block' of a CUT, surrounded by guard cells and a limited amount of training cells. An extra step is taken by not using CFAR around the edges where it becomes less reliable, to further prevent any false detections.

3.1.4 Main code section 10-12: Performing calculations and plotting the data

Multiple plots and graphs have been utilized to visualize the data. By performing an FFT-shift, it ensures the Doppler bins will appear around the center so that negative- and positive Doppler shifts can be visualized. Additionally, if this FFT-shift is not applied the Doppler bins will appear around the edges which will reduce the efficiency of CFAR. Additionally, the range-Doppler map is visualized in dB for a more accurate representation. These plots will be shown and analyzed in section 5.

3.2 3D Beamforming code

This part of the code is listed in 9.7 and implements 3D beamforming. In section 1 of the code the parameters are defined, such as the amount of uniform planar array (UPA) antenna elements. The distance Rb is the distance of the 3D targets to the UPA. This and the 3D position of a list of targets is used in section 2 to compute the matching azimuth (ϕ) and elevation angles (θ) the UPA will see. For simplicity, a simple arbitrary channel is set up, on which the azimuth- and elevation beamsteering vectors are applied with the function steeringvector2. After this, a full 3D angle scan is performed — so for $-90 < \phi < 90$ and $0 < \theta < 90$, which applies the beamforming on the pairs of angle that show the strongest response. This is not the most efficient technique, however, the simplicity of the channel does not make this implementation too process heavy. After this, a simple linear color map is created to visualize these positions. After this, linear thresholding is used to keep the strongest contributions of the beamformer output, this value will be adjusted until the output appears sufficient. This will often occur in multiple points cluttered together, which is why an algorithm is designed which only keeps the center point of a cluster of points. This is then used to create a set of points, which are then manually connected to plot the reconstructed stickman.

3.3 Visualization tools

A few smaller sections of code have been used to visualize some important specifications. Because the implementation is not complex, they are briefly explained in this section. The code in appendix 9.8 plots a 2D stickman in (ϕ, θ) angle space, identical to the implementation in the beamforming code. The code in appendix 9.9 extends this to a 3D stickman figure, by assigning a depth y to each (x, z)

Figure 11: Camera setup for the measurements

point used in the creation of the 2D stickman.

4. Human Model Measurements

In this section, the performed measurements are elaborated. Due to time constraints, the signal model has not been tested with most of the actual measurements. However, an explanation of how this would have been implemented is provided to show the relevance of these measurements.

4.1 Camera groundtruth benchmark

To create a more realistic scenario, measurements with cameras were performed. These cameras were positioned as shown in Figure 11 The measurements were carried out with help of two assistants. They provided the equipment and the measurement software and setup. The measurements were executed by the assistants wearing sensors on multiple different limbs and joints and taking on different poses. These were both stationary and dynamic poses.

The cameras were able to record data of the sensors when they were facing towards them. Due to these measurements being planned externally, the camera setup was non-ideal because when sensors were facing away from the camera, the sensor disappeared. This disappearing of sensors happened frequently, especially when there was rotation in some of the limbs. It still delivered meaningful data that was sufficient to use for the research, but it made evaluating the measurements more time-consuming.
The data acquired by the sensors gave a real-time human model to use the model on.

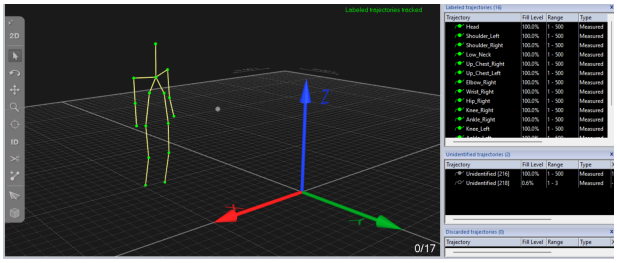


Figure 12: Qualisys software measurement environment

4.2 Implementation plan

In Figure 12 the Qualisys software environment is shown. The green dots represent the sensors, which are detected by the cameras. By performing a measurement with somebody wearing the sensors, these detected sensor can be connected and a stickman human model can be created. This can then be saved as a model, and constantly reapplied when performing measurements with different poses. Occasionally the cameras lost track of the sensors or anomalies showed up on the cameras. These are represented by the grey dots shown in figure 13.

This requires the generated data to be carefully reviewed and evaluated, before being ready for use. By converting the measurements to a MATLAB script, the described signal model can be tested on a realistic human model. This brings several benefits, especially to improve the accuracy of the an-

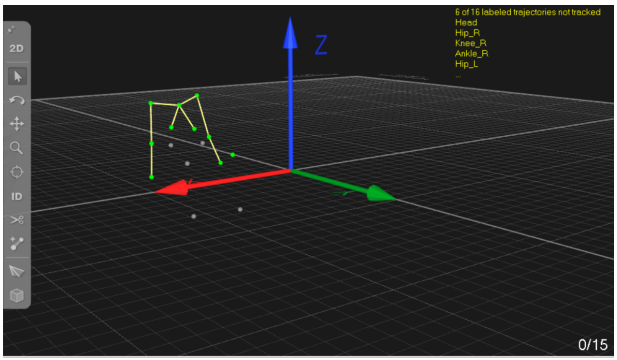


Figure 13: Visualization of sensors not being captured by the cameras

alyzed poses. This helped significantly by specifying the required range resolution, by being able to visualize the distance between limbs. Due to the described time constraints, the recorded data has not been used for implementation and validation of results.

5. Results

5.1 DOA, Range and Doppler Estimation Theoretical Verification

In this section. This will first be done by defining the system parameters, after which the verification will be presented.

5.1.1 System Parameter Design and Selection

In this section, the parameters used for the results are defined. The required parameters with brief explanations are defined in a table in appendix 9.12, where the required value is the minimum system requirement. The parameters that were actually used are shown in the Chosen Value column.

The minimum required range resolution was set to $\leq 20\,\text{cm}$. However, the modeled system allowed using $N = 8192$ subcarriers without causing any performance issues. This is why N is chosen to be significantly higher than the minimum requirement. Additionally, $M = 2222$ symbols were chosen to ensure that after applying the frequency shift there still existed a Doppler bin at $0\,\text{Hz}$. If this number is odd, it is impossible to determine if targets are stationary.

5.1.2 Antenna Specifications for Range- and Doppler Estimation

For the range- and Doppler estimation a ULA antenna array is used with $P = 16$ Tx and $Q = 4$ Rx. Using a UPA antenna array was not feasible for the range- and Doppler estimation, because this would require the use of too many channels. This inconsistency could not be resolved in time and is further elaborated in the discussion. The amount of channels is set to the amount of transmitters to ensure each Tx is not transmitting any overlapping data, so $N_{ch} = 16$. Because the focus of the research was more on the signal processing than the actual physical implementation of antennas, there has not been any research conducted about the physical antenna model and other specifications. For the 3D DOA estimation, a UPA antenna array is used with $P = 16 \times 16$ Tx and $Q = 4 \times 4$ Rx. The decision to use these values is motivated in section 5.1.7.

5.1.3 Range estimation

In Figure 14 and 15, the range-Doppler map and the distance velocity relation is visualized of 7 different targets —parameters specified in Figure 14—. The 7 targets can clearly be distinguished, which is expected because the distance between the targets is larger than the range resolution.

5.1.4 Doppler estimation

In Figure 16a and 16b, 7 targets with identical distances but varying Doppler shifts have

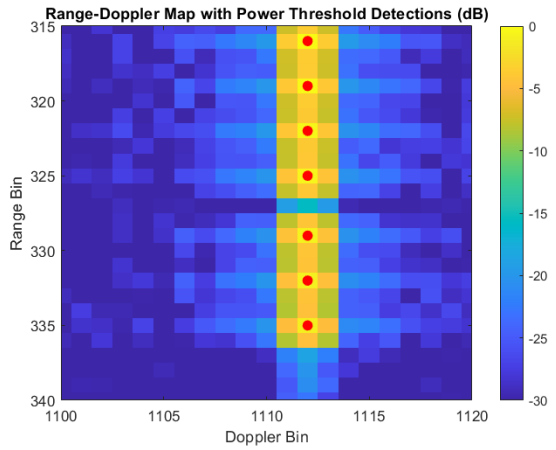


Figure 14: Range-Doppler map visualizing stationary targets with ranges of 48-51m (increments of 0.5m)

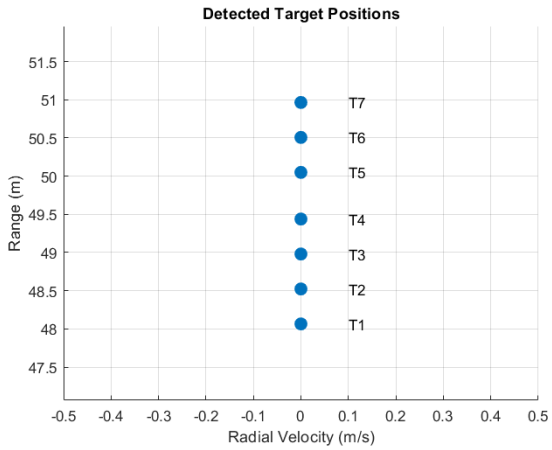


Figure 15: Ranges of targets visualized in Figure 14

been plotted to clearly visualize the Doppler detection. 7 distinguishable Doppler bins show up, all representing different targets. In Figure 16b, these Doppler bins have been converted to actual velocities, making it possible to detect multiple velocities between $-3.5 - 3.5m/s$. This makes it possible for the radar system to detect a limited amount of limb motions, mainly being able to distinguish between faster and slower motions.

5.1.5 Range and Doppler estimation

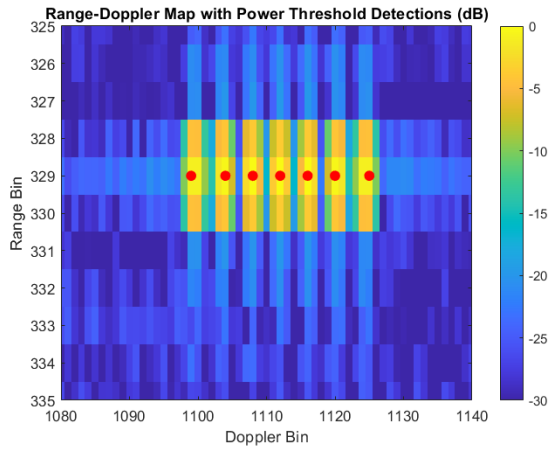
The results also have been visualized when multiple targets have different ranges and velocities. This is shown in Figures 17a and 17b, confirms that the range-Doppler estimation is performing correctly. In Figure 18 the actual measured Doppler shift, range and velocity is shown.

5.1.6 Comparison CFAR and regular power thresholding

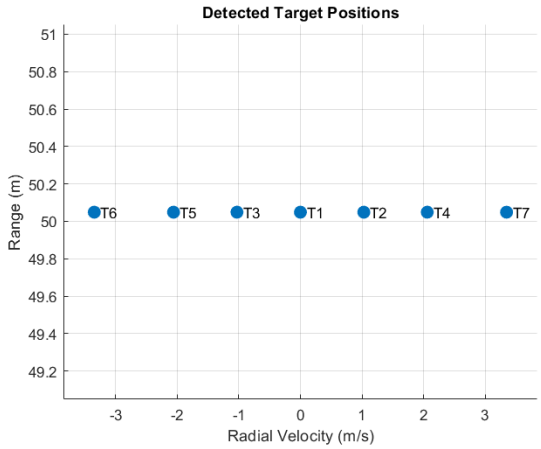
As an additional verification, CFAR was compared with regular power thresholding implemented as shown in 9.5 This lead to the surprising result shown in Figures 17a and 19that even at a really low threshold value such as 0.05, the regular power thresholding performed a lot better than the CFAR with a lower threshold value (10). The CFAR showed large amounts of cluttering and the range and Doppler estimation became extremely inaccurate. The reason this occurred is elaborated in section 6. Because of this inaccuracy, the previous range-Doppler maps have been plotted by using regular power thresholding and CFAR has been discarded.

5.1.7 3D DOA estimation

For accurate DOA estimation, only implementing 2D beamforming is not enough, as explained in 2.3.3. Therefore the 2D beamforming estimation was omitted later in the research and only the verification of 3D beamforming remained relevant. To verify if the 3D DOA angle estimation is functioning correctly, 3 azimuth angles $\phi_1 = -30°, \phi_2 = 10°, \phi_3 = 50°$ were plotted with a fixed elevation angle $\theta_f = 30°$, shown in Figure 20a. Furthermore, 3 elevation angles $\theta_1 = 20, \theta_2 = 40, \theta_3 = 55$ were plotted with a fixed azimuth angle $\phi_f = 20$, shown in Figure 20b Both of these plots show clear peaks at the expected

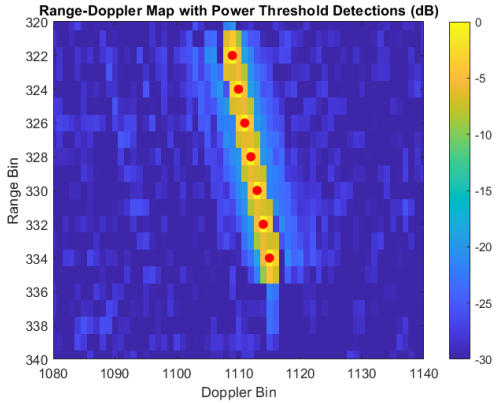


(a) Doppler map of targets with Doppler shift -600 - 600 Hz (increments of 200 Hz)

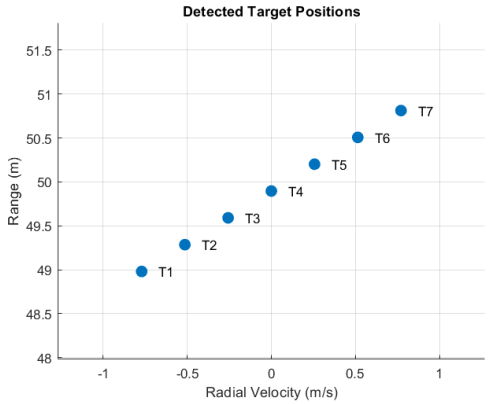


(b) Velocities of targets visualized in Figure 16a

Figure 16



(a) range-Doppler map of targets with Doppler shift -150 - 150 Hz (increments of 50 Hz) and distances of 49-50.8m (increments of 0.3m)



(b) Plot of the velocities and ranges of targets

Figure 17

angles, implying sufficient functionality of the 3D DOA estimation. To create a more meaningful result, the Half Power Beam Width (HPBW) can be calculated. However, this is not included in the research due to time constraints.

Angular resolution One of the most important specifications of the 3D beamforming is the angular resolution in both azimuth and elevation. This is required to properly determine positions of targets in section 5.2. There is not a trivial formula to compute the angular resolution of both elevation- and azimuth angles, which is why these resolutions were calculated by testing. By plotting a power response of the 3D beamforming of a singular target positioned at $(\phi, \theta) = (0, 50)$ the width and height in degrees of this response can be calculated. This is visualized in Figure 21 and shows that the angular resolution in both azimuth and elevation is around $\Delta\phi \approx \Delta\theta \approx 8°$ These tests were also carried out by using $P = 8 \times 8$. However, the angular resolution with this setup was too low which resulted in the peaks of both plots being significantly

```
→ Target 1: Range = 48.98 m | f_D = -144.01 Hz | v = -0.77 m/s
→ Target 2: Range = 49.29 m | f_D = -96.01 Hz | v = -0.51 m/s
→ Target 3: Range = 49.59 m | f_D = -48.00 Hz | v = -0.26 m/s
→ Target 4: Range = 49.90 m | f_D = 0.00 Hz | v = 0.00 m/s
→ Target 5: Range = 50.20 m | f_D = 48.00 Hz | v = 0.26 m/s
→ Target 6: Range = 50.51 m | f_D = 96.01 Hz | v = 0.51 m/s
→ Target 7: Range = 50.81 m | f_D = 144.01 Hz | v = 0.77 m/s
```

Figure 18: Actual values of range, Doppler and velocity of the targets

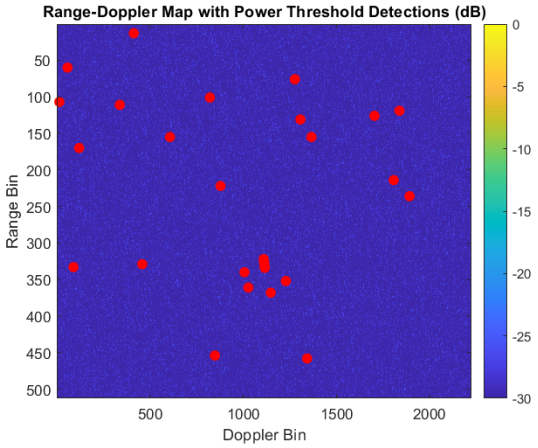


Figure 19: Low CFAR thresholding

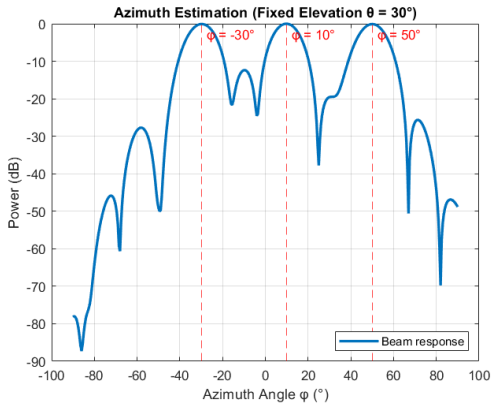
less distinct, which is why $P = 16 \times 16$ was used for the Tx UPA array.

5.2 Recognizing a Squat Position

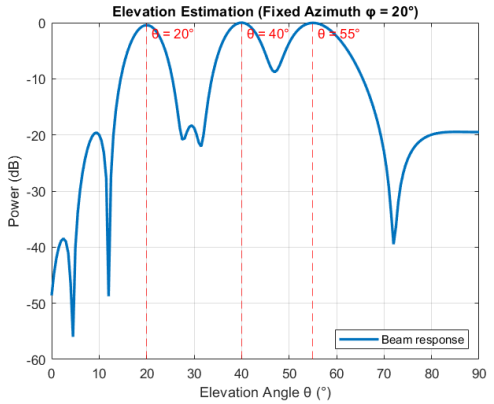
This subsection of the results contains a process where the individual data of the DOA estimation and range- and Doppler is integrated to recognize a position.

5.2.1 Pose Recognition Scenario

In this section, a scenario is created of a human in a squatting position, which will be estimated by performing 3D DOA estimation and range- and Doppler estimation. The chosen scenario to be modeled is a human in a squatting position, because it introduces vertical displacement of body parts such as the knee, head and lower torso. Additionally, because the lower torso is positioned further back this also creates an additional displacement which can be captured by the simple



(a) Plot of DOA estimation of $\phi_1 = -30°, \phi_2 = 10°, \phi_3 = 50°$ with fixed elevation angle $\theta_f = 30°$



(b) Plot of DOA estimation of $\theta_1 = 20, \theta_2 = 40, \theta_3 = 55$ with fixed azimuth angle $\phi_f = 20$

Figure 20

model. Before recognizing the squatting position, first some basic assumptions and conditions will be noted. A simple human model is constructed which is implemented by creating a stickman model. This stickman model consists out of several 2D points, which are then converted to an angular representation. Afterwards, these 2D points are modeled as 3D points to receive a 3D model as shown in Figures 22a and 22b. However, this implementation is not realistic for a real world scenario. The estimation of the location of these joints is not possible in this way because the whole body would be seen as a cluster con-

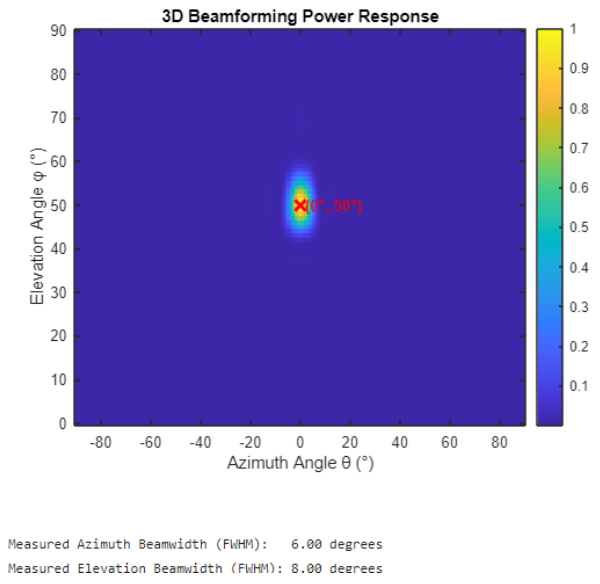
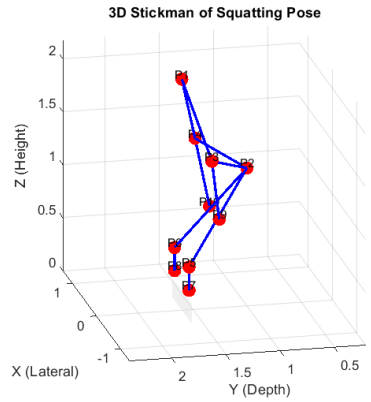
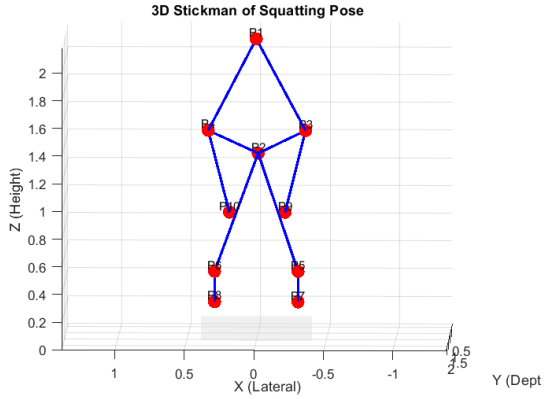


Figure 21: Estimation of elevation- and azimuth angular resolution



(a) Simple 3D stickman visualization (side view)



(b) Simple 3D stickman visualization (front view)

Figure 22: Simple 3D stickman visualizations

taining millions of these points. This is why for this scenario and application, a human must wear sensors at the locations of these modeled points for pose recognition to be performed. Additionally, it is important to note that due to an inaccuracy with coding, the notation of the azimuth and elevation angle has been swapped in the upcoming sections. This means that for the results the elevation angle is ϕ and the azimuth angle is θ

5.2.2 Pose Recognition Specifications

The chosen antenna setup is a UPA with $Tx = 16 \times 16$ and $Rx = 4 \times 4$. Choosing the 2D points to create the 3D model came with limitations, because the angular resolution is limited. Targets could not be too far away from the radar because at further distances their angular separation decreases. By performing multiple tests, only distances of around $1m$ ended up showing promising possibilities to perform recognition. Taking the 1m distance from the radar into consideration, allowed creating the stickman visualized in Figures 22a and 22b. The coordinates of these points (x,y,z) are given in Table 1. By trying multiple sets of coordinates, these coordinates showed promising results to show the possibilities of pose estimation with the designed system. Because of the limitations created by the limited angular resolution, the model is not the most realistic and the human is positioned on top of a small box.

The specific points are passed through the pipeline described in 3.2. Firstly, this will create a power map of the scanned angles by implementing the beamforming. This is visualized in Figure 23. The crosses mark the targets for visualization purposes.

Joint	X (m)	Y (m)	Z (m)
Head	0.00	1.60	2.18
Torso (Pelvis)	0.00	1.00	1.30
L Shoulder	-0.35	1.40	1.50
R Shoulder	0.35	1.40	1.50
L Knee	-0.30	1.60	0.50
R Knee	0.30	1.60	0.50
L Foot	-0.30	1.60	0.28
R Foot	0.30	1.60	0.28
L Hand	-0.20	1.30	0.90
R Hand	0.20	1.30	0.90

Table 1: Joint coordinates for the 3D stickman squatting pose. The Z-axis represents height.

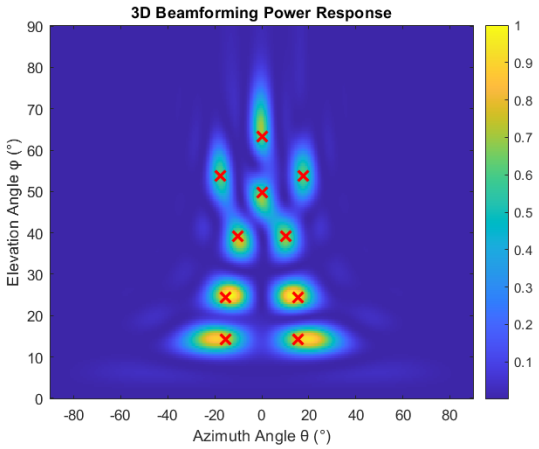


Figure 23: Visualization of the 3D beamforming power response

Regular power thresholding is applied to only keep the spots that have significantly higher power. This thresholding is not the most ideal since at really low elevation angles, the azimuth separation appears smaller from the radars perspective, reducing the precision. This explains the less concentrated peaks in Figures 23 and 24 at lower elevation angles.

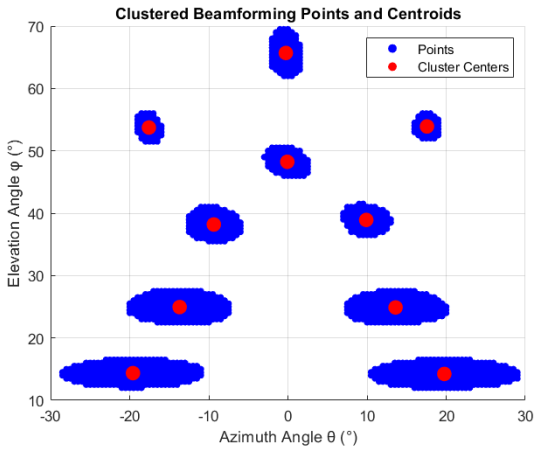


Figure 25: Visualization of the cluttered signals with centroids

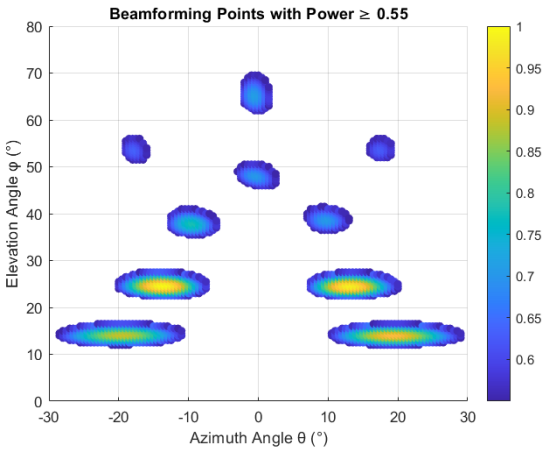


Figure 24: Power map after thresholding

In Figure 25, the data acquired by thresholding is processed. By applying these centroids, a more accurate target representation which can be used for reconstructing the stickman is acquired. This reconstruction is visualized in Figure 26

5.2.3 Range and Doppler estimation

In a pipeline where DOA estimation and range- and Doppler estimation are integrated, the DOA estimation is already used for target detection and range and Doppler shift is only computed after these targets have been detected. To reproduce this analogy, the range

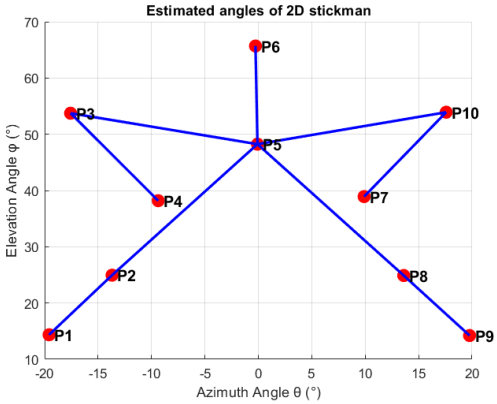
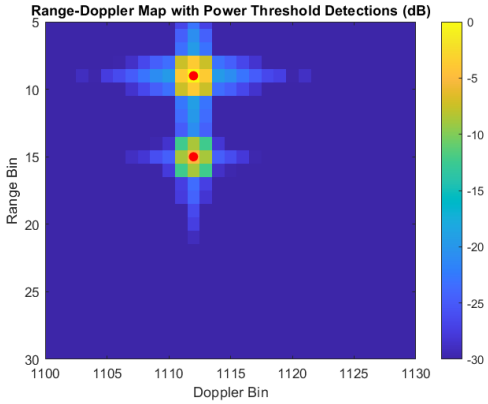
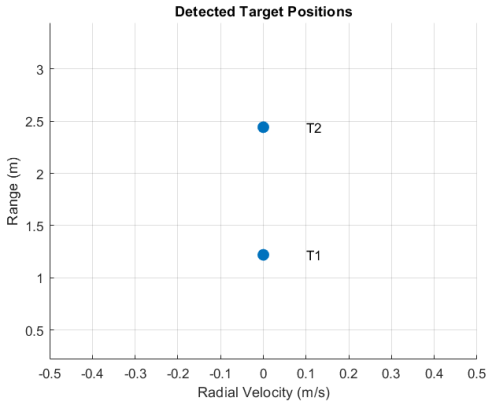


Figure 26: Reconstructed 2D stickman model



(a) Range-Doppler map of stationary lower torso and knee



(b) Distance and velocity of stationary lower torso and knee

Figure 27: Analysis of stationary lower torso and knee.

and Doppler of a select amount of points detected with the DOA estimation will be computed. 2 points will be evaluated, being the lower torso, and the right knee. Two variations of this will be considered, one where the pose is completely stationary and one where the lower torso is moving.

Stationary pose The two targets were able to be distinguished successfully as shown in Figures 27a and 27b. When more than 2 targets were included —e.g. the knee— only two targets showed up. This is mainly due to the fact that some separate targets have very similar distances from the radar, which can not be distinguished unless the range resolution is high. However, this is not an issue, as accurate DOA estimation makes it possible to calculate the range and Doppler of each individual target. This means pose recognition is still feasible

For extra verification the amount of subcarriers were set to $N = 16384$, effectively halving the range resolution. This finer range resolution lead to the ranges of all 3 targets being visible, as shown in Figure 28

Moving torso For this test, a set velocity is applied to the torso to represent movement in the squat position. In Figures 29 and 30, this is clearly visualized and it can be seen that the target representing the lower torso has a specific velocity. This also confirms the Doppler estimation is functioning correctly, making it possible to also perform pose estimation of mobile poses.

5.2.4 Angular resolution

The explored method of detecting a squat position delivered useful results in contexts of pose recognition. However, the effective range where the DOA estimation was accu-

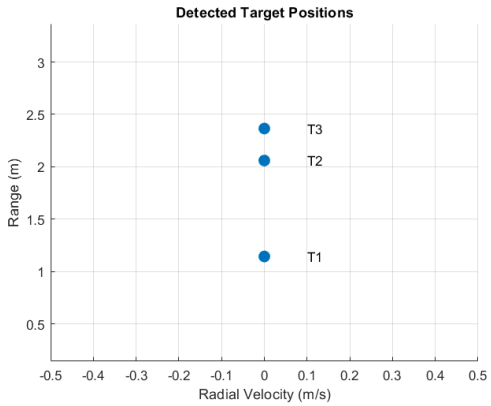


Figure 28: Distance and velocity of stationary lower torso, head and knee

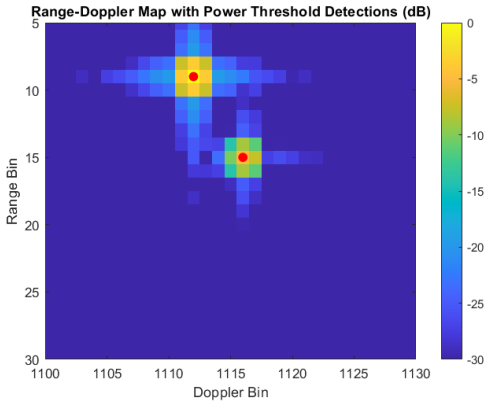


Figure 29: Range-Doppler map of moving lower torso and stationary knee

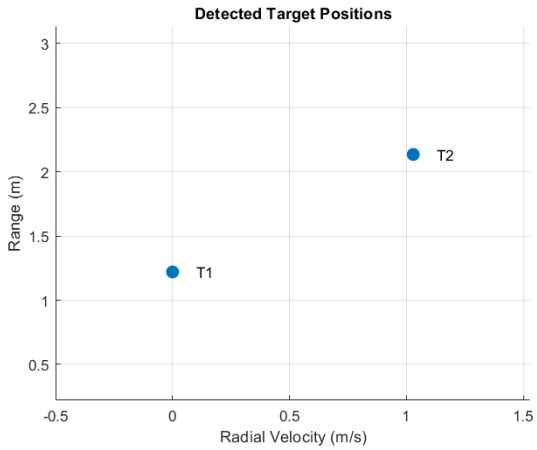


Figure 30: Distance and velocity of moving lower torso and stationary knee

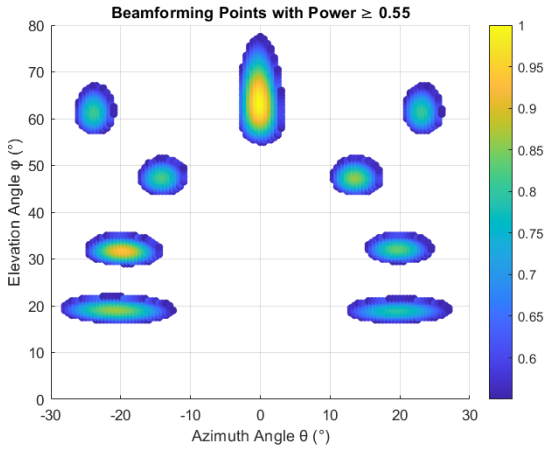


Figure 31: Response of beamforming for Tx size 16x16 when distance to radar = 0.8m

rate was only in between $0.9 - 1.3m$. If the distance of the human was closer or further away than this, results became inaccurate as visualized in Figures 32 and 31. In Figure 32 the lower torso fused together with the head and in Figure 31 the feet fused together.

To explore why this occurred, the amount of Tx was changed to 32×32. Even though this value is unrealistic for real life application, it increases the aperture size which results in an increase in range resolution. Here, the angle estimation remained accurate in a range of $0.7 - 1.7m$ as shown in Figures 33a and 33b.

6. Discussion

The main complication in this thesis was the integration of the DOA estimation with the range and Doppler estimation. The complexity of integrating these estimations was not anticipated which lead to prioritizing the implementation instead of confirming that the method of integration was supported by underlying theory. In hindsight, this ended up being time-consuming, which limited the scope of the research. For complications in integration stages in future research, it is rec-

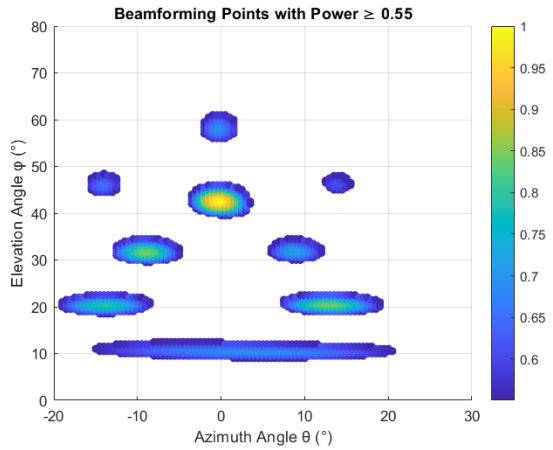
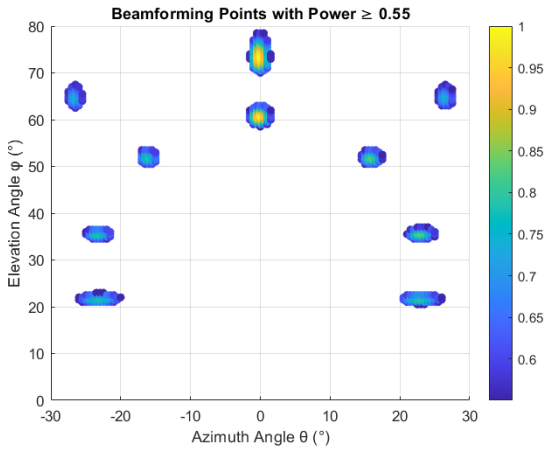
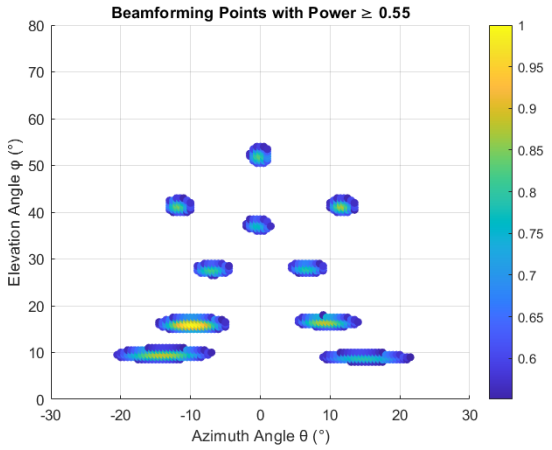


Figure 32: Response of beamforming for Tx size 16x16 when distance to radar = 1.4m



(a) Response of beamforming for Tx size 32x32 when distance to radar = 0.7m



(b) Response of beamforming for Tx size 32x32 when distance to radar = 1.7m

Figure 33

ommended to revisit the required theory instead of fixating on resolving the implementation issues.

One of the main issues encountered when coding in Matlab, was an excessive amount of memory usage in specific sections. This resulted in large processing times and 'out of memory' errors on multiple occasions. This was mainly due to code design that dit not prioritize efficient memory usage. N and M were reduced to combat this issue. However, more efficient approaches could have been taken, e.g. clearing unused variables during iterations or reducing the amount of consecutive for loops.

For further research, it is recommended to perform tests with other modulation schemes such as 64-QAM to gain better understanding of the effect of the modulation technique on the overal system performance. The decision to use 16-QAM was mostly based on other research and basic knowledge as it is a modulation scheme that is not too intensive and contains an acceptable 4 bits per modulation symbol. For research purposes however, even with time constraints a light amount of research could have lead to better understanding of modulation schemes in the researched application.

The interleaved structure was implemented as a precaution against ICI. However, this motivation would have benefited more from additional research instead of implementing it as an precaution. The interleaved structure made the processing pipeline more complex, which also led to complications with the beamforming implementation. Another general point of discussion related to this is the excessive amount of implemen-

tation without complete understanding of the theoretical background. This approach led to integration issues, which could have been prevented by doing more thorough background research before implementation. For complications in integration stages in future research, it is recommended to revisit the required theory instead of merely trying to fix the implementation.

Furthermore, a wrong assumption was made for the chosen amount of channels N_{Ch}. This should have been equal to $P \times Q = 16 \times 4$ for the range- and Doppler estimation, however the wrong assumption $N_{Ch} = P$ was made which is why only 16 channels were used. This misconception did not have any effect on the range- and DOA estimation, however, it should still be noted that this modeling error was made.

Additionally, the motivation of using an interleaved structure would have benefited from a comparison with a non-interleaved structure. This also would have clearly visualized the effects of potential ICI. If the effect of the ICI appeared to be minimal, a simpler processing chain without an interleaved structure could be utilized.

Implementing CFAR appeared to improve results, however, this masked the effect of another error in the MATLAB code. The main motivation to find another detection method such as CFAR, was that often a lot of false targets showed up on the range-Doppler map. However, these false detections appeared because for some specific target distances, a target was not assigned to an actual range bin and were not related to the method of detection. This was eventually resolved by always mapping the propagation delay to a closely spaced range bin. As shown in the results, simple thresholding based on power also ended up being a sufficient approach.

The research produced promising results for using a MIMO radar system with OFDM for pose recognition purposes. However, the system only functioned under quite strict conditions which makes the system really sensitive and impractical for actual use.

The acquired angular resolution by using beamforming ended up being quite poor, severely limiting the effective range of the radar as shown in the results. In future work, finer DOA estimation techniques such as MUSIC could be explored to improve this and increase the effective range the pose estimation model functions on.

Because the radar only was functional for really near distances, the 3D DOA estimation became inaccurate at low and high elevation angles. By increasing the aperture size, this problem appeared to be less signficant. However, this increased aperture size — a 32x32 Tx grid — is extremely unrealistic and can never be realized in real life scenarios. Requiring large aperture sizes to obtain a better angular resolution is another drawback of beamforming, creating another reason to research DOA methods such as MUSIC.

Additionally, the angle scanning for 3D beamforming was not optimized. It had to scan over a full $180° \times 90°$, with a step size of $0.5°$ which did not lead to too large complexity because the DOA estimation was implemented separately. However, when integrated with the range- and Doppler estimation this could lead to large computational loads. A more sophisticated method such as fan- or pencil beamforming scanning could have been used. These methods detect a space containing multiple angles at the same time, reducing the amount of required computations. This should be considered for further research

Furthermore, creating the human model brought several complications with it. The amount of points and the location of these points for the human model had to be chosen so that they were still distinguishable for DOA estimation. This approach had to be taken to show that pose estimation would be possible with the system even tough this was only under specific conditions. Creating a more accurate human model was not possible because of this which made the researched method of pose recognition quite unreliable. To be able to perform realistic and reliable pose recognition, earlier listed measures should be taken to enable more precise human modeling.

7. Conclusion

Research about OFDM MIMO radar in the context of pose recognition has been carried out and delivered results on the use of DOA and range-Doppler estimation in the context of pose recognition. Even though the desired outcome of full integration of DOA estimation and range- and Doppler estimation has not been achieved, performing a thorough amount of theoretical background research and implementation led to both of these estimations functioning well on their own as the results proved. Multiple points of improvement such as using more advanced signal-processing techniques to obtain better angular resolution have been listed in the discussion and provided valuable insights for future research. Several techniques such as CFAR and MRT beamforming have been explored and their relevance to this thesis has been addressed. For further research, the human measurements can be utilized to test the reliability of the implementation in real life applications. Furthermore, the thesis also highlighted how a MIMO OFDM radar system was able to perform pose recognition under certain constraints. The implementation stage also revealed multiple challenges, mainly between the computational load and the system specifications, requiring trade-offs between processing time and resolution. By simulating in MATLAB, several results proved accurate DOA and range- and Doppler estimations. By performing several tests, an attempt at pose estimation was conducted and the DOA estimation of a squatting position provided accurate and promising results. This made it possible to estimate the range and velocity of a few data points used for the DOA estimation, showing the possibilities of pose recognition with this system. Even though it was possible to detect a squatting pose and valuable results were produced, these results remained limited by angular resolution, range- and Doppler resolution and by using a coarse human model. In future work the focus should lie on improving the system by conducting tests on the human measurements, exploring more efficient signal processing techniques, integrating the estimations and fine tuning specifications to produce a fully integrated functioning system for pose recognition.

8. AI Statement

ChatGPT+ has been used as a tool to assist with writing and coding. It should be highlighted that this was used as an assisting tool and everything created by chatGPT+ was evaluated critically. Non of the research is written by chatGPT+ and it was only used for creating ideas / creating ideas for polishing up some sentences.

9. Acknowledgements

I would like to thank Nguyen Dao and Mensah Obeng Afrane for their effort and assistance with the measurements. Additionally I would like to thank Yang Miao and again Nguyen Dao for their supervision and motivation, without them this research would not have been made possible.

References

[1] Martin Braun, Christian Sturm, and Friedrich K. Jondral. *On the single-target accuracy of OFDM radar algorithms*. 2011.

[2] Ashwin Nanjappa, K K Biswas, and Subhashis Banerjee. Deep learning based human pose estimation: A survey. *arXiv preprint arXiv:2012.13392*, 2020.

[3] Baban Rindhe and Santosh Narayankhedkar. Effects of cyclic prefix on ofdm system. pages 420–424, 02 2010.

[4] Y.L. Sit. *MIMO OFDM Radar-Communication System with Mutual Interference Cancellation*. KIT Scientific Publishing, Karlsruhe, 2017.

[5] Chuang Wang, Jianmin Hu, Qunying Zhang, and Xinhao Yuan. An efficient 2d doa estimation algorithm based on omp for rectangular array. *Electronics*, 12(7), 2023.

[6] Mingmin Zhao, Ye Tian, and Hang Zhao. Rf-pose3d: Body pose estimation using wifi. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 267–281. ACM, 2018.

[7] Mingmin Zhao, Hang Zhao, Tianhong Wei, Shuyang Zhang, Yan Geng, Antonio Torralba, and Dina Katabi. Through-wall human mesh recovery using radio signals. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10113–10122, 2019.

Appendix

9.1 Main code

```
clear all
%Section 1: Parameters
N = 2048*2^2; %Amount of subcarriers
CP = 1/8*N; %Cyclic prefix length
Nch = 16; %Amount of channels
Nu = N / Nch; %Amount of subcarriers
    per channel
M = 2048; %Amount of OFDM symbols
M_o = 16; %Modulation order
SNR_dB = 30; %SNR: Still needs a
    precise calculation
vrel_h = 1; %Relative velocity of the
    target, not used a.t.m.
delta_f = 120e3; %Subcarrier spacing
c0 = 3e8; %Light speed
f_c = 28e9; %Center frequency
lambda_c = c0 / f_c; %Center frequency
    wavelength
p = 8; %Amount of Tx
q = 4; %Amount of Rx


params.N = N;
params.Nch = Nch;
params.M = M;
params.M_o = M_o;
params.CP = CP;
params.delta_f = delta_f;
params.vrel_h = vrel_h;
params.SNR_dB = SNR_dB;
params.f_c = f_c;


%Section 2: Pre-allocating antennas and
     signal space
bits_per_user = Nu * log2(M_o);
```

```
channel_to_tx_map = mod(0:Nch-1, p) +
    1;

A_p = cell(p, 1);
for tx = 1:p
    A_p{tx} = zeros(N, M);
end



%Section 3: Generating interleaved QAM
    symbols to improve ICI
for m = 1:M
    qamSymbols = cell(Nch, 1);
    for ch = 1:Nch
        qamParams.bits = bits_per_user;
        qamParams.M_o = M_o;
        qamParams.N = Nu;
        qamSymbols{ch} = QAM_mod(
    qamParams);
    end

    for ch = 0:Nch-1
        tx = channel_to_tx_map(ch+1);
        for i = 0:(Nu - 1)
            subcarrier_index = i * Nch
    + ch;
            A_p{tx}(subcarrier_index +
    1, m) = qamSymbols{ch + 1}(i + 1);
        end
    end
end


% === Section 4: Target and Channel
    Setup (3D)  NEW
K = 1;  % Number of targets

% 3D positions of each target [x, y, z]
     in meters
R_vec =  [90, 0, 0];%Example of tested
    values


f_D_vec = 0;         % Doppler shifts [
    Hz]
```

```
alpha_vec = ones(1,K);
           % Reflection coefficients
%
% theta_deg = 45;
% theta_rad = deg2rad(theta_deg);  %
    Will use later for steerin

OFDM_tx_CP = cell(p, 1);  % Output
    after channel (for each Tx)

% Antenna array geometry
% d_ant = lambda_c / 2;
% tx_pos = (0:p-1) * d_ant;       % Tx
    ULA along x-axis
% rx_pos = (0:q-1) * d_ant;       % Rx
    ULA along x-axis
% tx_pos_3D = [tx_pos(:), zeros(p,1),
    zeros(p,1)];  % 3D coords: [x, y, z]
% rx_pos_3D = [rx_pos(:), zeros(q,1),
    zeros(q,1)];

% Simulate per-Tx
for tx = 1:p
    [OFDM_tx_CP_raw] = OFDM(params, A_p
    {tx});
    total_rx = zeros(size(
    OFDM_tx_CP_raw));
    params.tx = tx;
    for k = 1:K
        pos_k = R_vec(k, :)
     % 3D position
        R_k = norm(pos_k)
    % Actual range
        params.R = R_k;
      % Range for delay calc
        params.f_D = f_D_vec(k);
      % Doppler

        % Angle computation for
    steering
        % Vector from origin (array
    center) to target
%         unit_vec_k = pos_k / R_k;
       % Unit direction vector

        % Call channel with geometry-
    aware input
```

```
        OFDM_rx_k = OFDM_Channel2(
    params, OFDM_tx_CP_raw);
%                                       ,
    tx_pos_3D, rx_pos_3D, unit_vec_k);

        total_rx = total_rx + alpha_vec
    (k) * OFDM_rx_k;
    end

    OFDM_tx_CP{tx} = total_rx;
end



%Section 5 Receiving the signal
received_signal = cell(q, 1);
for rx = 1:q
    received_signal{rx} = zeros(N+CP, M
    );
    for tx = 1:p
        received_signal{rx} =
    received_signal{rx} + OFDM_tx_CP{tx
    };
    end
end


%Section 6 Stripping signal of CP and
    converting it back to f-domain
Y = cell(q, 1);
for rx = 1:q
    received_signal_noCP =
    received_signal{rx}(CP+1:end, :);
    Y{rx} = fft(received_signal_noCP, N
    , 1);
end

%Section 7 Receiving desired data
D_qp = cell(Nch, q);
for ch = 1:Nch
    tx = channel_to_tx_map(ch);  % Get
    which Tx sent on this channel

    for rx = 1:q
        D_qp{ch,rx} = zeros(N, M);  %
    Preallocate

        for m = 1:M
            for i = 0:(Nu - 1)
                n_u = i * Nch + (ch -
    1);  % Actual subcarrier index

             %matched filter by
    multiplying with conjugate
                tx_sym = A_p{tx}(n_u+1,
     m);  % Transmitted symbol
                rx_sym = Y{rx}(n_u+1, m
    );    % Received symbol (FFT output)

                D_qp{ch,rx}(n_u+1, m) =
     rx_sym * conj(tx_sym);
                % Correlation, high
    value if Tx and Rx symbol match
            end
        end
    end
end


%Section 8: Combined Range-Doppler
    Processing
R_sum = zeros(Nu, M);
range_win = kaiser(Nu,5);
for ch = 1:Nch
    used_indices = (0:(Nu - 1)) * Nch +
     (ch - 1);
    R_combined = zeros(Nu, M);
    for rx = 1:q
        R_temp = zeros(Nu, M);
        for m = 1:M
            d_used = D_qp{ch,rx}(
    used_indices + 1, m);
            d_used_win = d_used .*
    range_win;
            R_temp(:, m) = ifft(
    d_used_win);
        end
        R_combined = R_combined +
    R_temp;
    end
    win = kaiser(M,5).';  % Apply along
     time axis
    R_combined = R_combined .* win;  %
    Element-wise multiplication
```

```
    R_doppler = fftshift(fft(R_combined
    , [], 2), 2);
    R_sum = R_sum + R_doppler;
end
R_sum = abs(R_sum);
R_sum_dB = 10*log10(R_sum / max(R_sum
    (:)) + 1e-10);

%Section 9: CFAR-Based Detection

% Normalize power map
R_power = R_sum.^2;
R_power = R_power / max(R_power(:));

% CFAR parameters
guard_cells = 2;
training_cells = 8;
alpha = 10  ;  % CFAR scaling factor

% Total CFAR window size
total_cells = guard_cells +
    training_cells;
win_size = 2 * total_cells + 1;

% Create CFAR mask: 1 for training
    cells, 0 for guard + CUT
cfar_mask = ones(win_size);
cut_pos = total_cells + 1;
cfar_mask(cut_pos-guard_cells:cut_pos+
    guard_cells, ...
          cut_pos-guard_cells:cut_pos+
    guard_cells) = 0;

% Normalize the mask
num_training_cells = sum(cfar_mask(:));
cfar_kernel = cfar_mask /
    num_training_cells;

% Estimate noise level via 2D
    convolution
noise_est = conv2(R_power, cfar_kernel,
     'same');

% Threshold calculation
threshold = alpha * noise_est;
```

```
% Suppress edges (where CFAR is
    unreliable)
valid_mask = zeros(size(R_power));
valid_mask(cut_pos:end-cut_pos+1,
    cut_pos:end-cut_pos+1) = 1;

% Apply CFAR detection
detection_mask = (R_power > threshold)
    & valid_mask;

%Keep only local maxima (remove blobs/
    duplicates)
fprintf('\n==== R_power DIAGNOSTICS
    ====\n');
disp(['Class: ', class(R_power)]);
disp(['Size: ', mat2str(size(R_power))
    ]);
fprintf('Min: %.4e | Max: %.4e\n', min(
    R_power(:)), max(R_power(:)));
fprintf('Any NaNs? %d\n', any(isnan(
    R_power(:))));
fprintf('Any Infs? %d\n', any(isinf(
    R_power(:))));
fprintf('Any complex values? %d\n', ~
    isreal(R_power(:)));
local_max = imregionalmax(R_power);
detection_mask = detection_mask &
    local_max;

% Extract indices of detected targets
[range_idx, doppler_idx] = find(
    detection_mask);
peak_vals = R_sum(sub2ind(size(R_power)
    , range_idx, doppler_idx));

disp("Detected peaks (range bin,
    doppler bin, value):");
disp([range_idx, doppler_idx, peak_vals
    ]);

% Limit number of reported peaks
num_peaks = min(length(peak_vals), 30);
[~, top_idx] = maxk(peak_vals,
    num_peaks);
range_idx = range_idx(top_idx);
doppler_idx = doppler_idx(top_idx);
```

```
% %Section 10: Convert indices to
    physical units
range_res = c0 / (2 * N * delta_f);
T_sym = (N + CP) / (N * delta_f)
T_test = T_sym*(CP+N)
f_d_res = 1 / (M * T_sym);
r_vals = (range_idx - 1) * range_res;
v_vals = (doppler_idx - (M / 2 + 1)) *
    f_d_res * (lambda_c / 2); %Present
    to maybe be implemented with doppler
     filtering.



%Section 11: Calculating the range and
    velocity

fprintf("Range resolution: %.3f m |
    Doppler resolution: %.2f Hz\n",
    range_res, f_d_res);

for t = 1:length(range_idx)
    k_hat = range_idx(t);
    d_hat = doppler_idx(t);

    r_est = range_res * (k_hat - 1);
    d_bin_offset = d_hat - (M / 2 + 1);
    f_D_est = f_d_res * d_bin_offset;
    v_est = (f_D_est * lambda_c) / 2;

    fprintf("\t Target %d: Range = %.2f
     m | f_D = %.2f Hz | v = %.2f m/s\n
    ", ...
        t, r_est, f_D_est, v_est);
end

figure;
imagesc(R_sum_dB);
xlabel('Doppler Bin'); ylabel('Range
    Bin');
title('Range-Doppler Map with Power
    Threshold Detections (dB)');
colorbar; caxis([-30 0]);
hold on;
scatter(doppler_idx, range_idx, 50, 'r
    ', 'filled');
ylim([5, 30]);
xlim([1100, 1130]);



% Define axes
doppler_axis = ((1:M) - (M/2 + 1)) *
    f_d_res;
velocity_axis = doppler_axis * (
    lambda_c / 2);
range_axis = (0:Nu-1) * range_res;

% Compute physical values for detected
    peaks
r_vals = (range_idx - 1) * range_res;
v_vals = (doppler_idx - (M / 2 + 1)) *
    f_d_res * (lambda_c / 2);

% Plot detected targets
figure;
scatter(v_vals, r_vals, 80, 'filled');
xlabel('Radial Velocity (m/s)');
ylabel('Range (m)');
title('Detected Target Positions');
grid on;

% Autoscale with padding
xlim([min(v_vals)-0.5, max(v_vals)
    +0.5]);
ylim([min(r_vals)-1, max(r_vals)+1]);

for i = 1:length(r_vals)
    text(v_vals(i) + 0.1, r_vals(i),
    sprintf('T%d', i));
end

figure;
imagesc(10*log10(R_sum)); colorbar;
xlabel('Doppler Bin'); ylabel('Range
    Bin');
title('Range-Doppler Power Map');
T_sym = (N + CP) / (N * delta_f);
f_d_max = 1 / (2 * M * T_sym);
fprintf("Max unambiguous Doppler: %.2f
    Hz\n", f_d_max);
```

9.2 OFDM

```
    function [OFDM_symbols_CP] = OFDM(
    params, pre_interleaved_QAM)
%Parameters
CP = params.CP; %Cyclic prefix length
N = params.N; %Amount of subcarriers


X = pre_interleaved_QAM;
OFDM_symbols = ifft(X,N,1); %Taking
    IFFT of each column

    % Normalize for power per active
    subcarrier
    active_bins = sum(any(X ~= 0, 2));
    scale = sqrt(N / active_bins);  %
    Boost to preserve energy
    OFDM_symbols = scale * OFDM_symbols
    ;

OFDM_symbols_CP = [OFDM_symbols(end-CP
    +1:end, :); OFDM_symbols]; %Taking
    last CP subcarrier symbols,
    concatinating them in front of the
    OFDM symbol.
end
```

9.3 steering-vector

```
    function a = steering_vector(N, d,
    lambda, theta_deg)
    theta_rad = deg2rad(theta_deg);
    n = 0:N-1;  % element indices
    a = exp(1j * 2 * pi * d / lambda *
    n * sin(theta_rad)).';  % column
    vector
end
```

9.4 TestBeamforming

```
    % Parameters
p = 16;           % Number of Tx
    antennas
q = 8;           % Number of Rx
    antennas
f_c = 28e9;      % Carrier frequency
c0 = 3e8;        % Speed of light
lambda_c = c0 / f_c;
d_ant = lambda_c / 2;  % Half-
    wavelength spacing

% Define multiple targets
theta_targets = [30, 46];           %
    AoAs in degrees
rcs_targets = [1, 1];           %
    Reflection coefficients
K = length(theta_targets);
     % Number of targets

% Construct the total channel matrix
H = zeros(q, p);
for k = 1:K
    theta_k = theta_targets(k);
    rcs_k = rcs_targets(k);

    a_tx_k = steering_vector(p, d_ant,
    lambda_c, theta_k);  % p x 1
    a_rx_k = steering_vector(q, d_ant,
    lambda_c, theta_k);  % q x 1

    H = H + rcs_k * (a_rx_k * a_tx_k');
      % Accumulate target contributions
end

% Beamforming angle sweep
angles = -90:0.5:90;
beam_response = zeros(size(angles));

for i = 1:length(angles)
    theta = angles(i);
    a_tx = steering_vector(p, d_ant,
    lambda_c, theta);
    a_rx = steering_vector(q, d_ant,
    lambda_c, theta);

    R_theta = a_rx' * H * a_tx;
    beam_response(i) = abs(R_theta)^2;
end

% Normalize and plot
beam_response = beam_response / max(
    beam_response);
```

```
figure;
plot(angles, beam_response, 'LineWidth
    ', 2);
xlabel('Angle (degrees)');
ylabel('|Response|^2');
title('Beamforming Response vs. Angle')
    ;
grid on;

% Mark true target angles
hold on;
for k = 1:K
    xline(theta_targets(k), '--r',
    sprintf('Target %d: %d', k,
    theta_targets(k)), ...
        'LabelOrientation', 'horizontal
    ');
end
```

9.5 Regular power thresholding

```
    %Section 9 (Alternative): Regular
    Power Thresholding

% Normalize power map
R_power = R_sum.^2;
R_power = R_power / max(R_power(:));

% Set a fixed threshold (adjust as
    needed)
power_thresh = 0.05;  % Try 0.30.5 for
    tuning

% Apply simple threshold
detection_mask = R_power > power_thresh
    ;

% Keep only local maxima
local_max = imregionalmax(R_power);
detection_mask = detection_mask &
    local_max;

% Extract indices of detected targets
[range_idx, doppler_idx] = find(
    detection_mask);
peak_vals = R_sum(sub2ind(size(R_power)
    , range_idx, doppler_idx));

disp("Detected peaks (range bin,
    doppler bin, value):");
disp([range_idx, doppler_idx, peak_vals
    ]);

% Limit number of reported peaks
num_peaks = min(length(peak_vals), 30);
[~, top_idx] = maxk(peak_vals,
    num_peaks);
range_idx = range_idx(top_idx);
doppler_idx = doppler_idx(top_idx);
```

9.6 OFDM-Channel

```
        function [rx_OFDM_symbols_CP] =
    OFDM_Channel2(params, OFDM_tx_CP)
%Parameters
M = params.M;
CP = params.CP;
N = params.N;
f_c = params.f_c;
delta_f = params.delta_f;
SNR_dB = params.SNR_dB;
R = params.R;
f_D = params.f_D;
% Constants
c0 = 3e8;
f_s = N * delta_f;

% --- Doppler shift in time domain ---
total_samples = (N + CP) * M;
t = (0:total_samples - 1).' / f_s;
doppler_phase_shift = exp(1j * 2 * pi *
    f_D * t);
% Apply Doppler effect
tx_vector = reshape(OFDM_tx_CP, [], 1);
rx_vector = tx_vector .*
    doppler_phase_shift;
rx_OFDM_symbols_CP = reshape(rx_vector,
    N + CP, M);

% --- Range shift in frequency domain
    ---
% Remove CP to get symbols
```

```
rx_OFDM_noCP = rx_OFDM_symbols_CP(CP+1:
    end, :);

% FFT to convert to frequency domain
Y = fft(rx_OFDM_noCP, N, 1);

%Propagation phase shift
n = 0:N-1;
f_n = n * delta_f;
T_s = 1 / (N * delta_f);
tau = 2 * R / c0;
tau_quantized = round(tau / T_s) * T_s;
range_phase_shift = exp(-1j * 2 * pi *
    f_n * tau_quantized).';
Y = Y .* range_phase_shift;

% Back to time domain
rx_OFDM_noCP = ifft(Y, N, 1);
rx_OFDM_symbols_CP = [rx_OFDM_noCP(end-
    CP+1:end, :); rx_OFDM_noCP];



lambda = c0 / f_c;
G_tx = 1;  % linear gain (0 dBi)
G_rx = 1;  % linear gain (0 dBi)

% One-way distance (squared below for
    round-trip)
% Radar: Round-trip loss  (R^4), so
    Friis becomes:
amplitude_loss = (G_tx * G_rx * lambda
    ^2) / ((4 * pi * R)^2);
rx_OFDM_symbols_CP = rx_OFDM_symbols_CP
    * amplitude_loss;



% --- Add AWGN ---
rng(42);
signal_power = mean(abs(
    rx_OFDM_symbols_CP(:)).^2);
SNR_linear = 10^(SNR_dB / 10);
noise_power = signal_power / SNR_linear
    ;
noise = sqrt(noise_power / 2) * (randn(
    size(rx_OFDM_symbols_CP)) + 1j *
    randn(size(rx_OFDM_symbols_CP)));

rx_OFDM_symbols_CP = rx_OFDM_symbols_CP
    + noise;
end
```

9.7 2D Beamforming

```
    clc; clear; close all;
%Beamforming: Sec1
%% Parameters
f_c = 28e9;                 % Carrier
    frequency (Hz)
c0 = 3e8;                   % Speed of
    light (m/s)
lambda_c = c0 / f_c;        % Wavelength
d_ant = lambda_c / 2;       % Half-
    wavelength antenna spacing

% Transmit UPA: 16x16 = 256 elements
Nx_tx = 16; Ny_tx = 16; p = Nx_tx *
    Ny_tx;

% Receive UPA: 4x4 = 16 elements
Nx_rx = 4; Ny_rx = 4; q = Nx_rx * Ny_rx
    ;

Rb = 1.1;  % Distance to stickman


%Beamforming: Sec2
x_vals = [ ...
    0,     0,    -0.35,  0.35,  -0.3,
    0.3,  -0.3,  0.3,  -0.2,  0.2];  %
    lateral [m]

z_vals = [ ...
    2.08,  1.20,  1.40,   1.40,   0.40,
       0.40,  0.18,   0.18,   0.80,
    0.80]+0.1; % height [m]

K = length(x_vals);

% Convert stickman points to azimuth
    and elevation angles (degrees)
theta_targets = atan2d(x_vals, Rb);  %
    Azimuth angle
```

```
phi_targets = atan2d(z_vals, Rb);    %
    Elevation angle

% Assign rcs
rcs_targets = ones(1, K);

% Construct channel matrix H
H = zeros(q, p);
for k = 1:K
    a_tx_k = steering_vector2(Nx_tx,
    Ny_tx, d_ant, lambda_c,
    theta_targets(k), phi_targets(k));
    % (p x 1)
    a_rx_k = steering_vector2(Nx_rx,
    Ny_rx, d_ant, lambda_c,
    theta_targets(k), phi_targets(k));
    % (q x 1)
    H = H + rcs_targets(k) * (a_rx_k *
    a_tx_k');
end

%Full 2D Beamforming Sweep
angles_theta = -90:0.5:90;   % Azimuth
    sweep (columns)
angles_phi   = 0:0.5:90;     %
    Elevation sweep (rows)

R_2D = zeros(length(angles_phi), length
    (angles_theta));  % (phi x theta)

for ti = 1:length(angles_theta)
    theta = angles_theta(ti);
    for pi = 1:length(angles_phi)
        phi = angles_phi(pi);
        a_tx = steering_vector2(Nx_tx,
    Ny_tx, d_ant, lambda_c, theta, phi);
      % (p x 1)
        a_rx = steering_vector2(Nx_rx,
    Ny_rx, d_ant, lambda_c, theta, phi);
      % (q x 1)
        R = a_rx' * H * a_tx;
        R_2D(pi, ti) = abs(R)^2;
    end
end

R_2D = R_2D / max(R_2D(:));  %
    Normalize


%2D Beamforming Power Heatmap
figure;
imagesc(angles_theta, angles_phi, R_2D)
    ;
xlabel('Azimuth Angle  ()');
ylabel('Elevation Angle  ()');
title('2D Beamforming Power Response');
colorbar;
set(gca, 'YDir', 'normal');

% Mark true target positions
hold on;
for k = 1:K
    plot(theta_targets(k), phi_targets(
    k), 'rx', 'MarkerSize', 10, '
    LineWidth', 2);
    text(theta_targets(k)+2,
    phi_targets(k), ...
        sprintf('P%d (%.1f, %.1f)', k,
    theta_targets(k), phi_targets(k)),
    ...
        'Color', 'r', 'FontWeight', '
    bold');
end

%Beamwidth est
[max_val, max_idx] = max(R_2D(:));
[row_peak, col_peak] = ind2sub(size(
    R_2D), max_idx);

% Azimuth cut (fixed elevation)
az_slice = R_2D(row_peak, :);
peak_power = max(az_slice);
mask_theta = az_slice >= 0.5 *
    peak_power;
beamwidth_theta = max(angles_theta(
    mask_theta)) - min(angles_theta(
    mask_theta));

% Elevation cut (fixed azimuth)
el_slice = R_2D(:, col_peak);
peak_power_el = max(el_slice);
mask_phi = el_slice >= 0.5 *
    peak_power_el;
beamwidth_phi = max(angles_phi(mask_phi
    )) - min(angles_phi(mask_phi));
```

```
% print beamwidths
fprintf('\nMeasured Azimuth Beamwidth (
    FWHM):   %.2f degrees\n',
    beamwidth_theta);
fprintf('Measured Elevation Beamwidth (
    FWHM): %.2f degrees\n',
    beamwidth_phi);

%% 2D Azimuth and Elevation Cuts of
    Beam Pattern
% Azimuth slice plot (fixed elevation)
figure;
plot(angles_theta, az_slice, 'LineWidth
    ', 2);
xlabel('Azimuth Angle  ()');
ylabel('Normalized Power');
title('Azimuth Beam Pattern Slice (
    Fixed Elevation)');
grid on;
ylim([0 1.1]);

% Elevation slice plot (fix azimuth)
figure;
plot(angles_phi, el_slice, 'LineWidth',
     2);
xlabel('Elevation Angle  ()');
ylabel('Normalized Power');
title('Elevation Beam Pattern Slice (
    Fixed Azimuth)');
grid on;
ylim([0 1.1]);

%Beamforming: Sec3
threshold = 0.3;  % normalize power
    threshold (0 to 1)



figure;
hold on;
grid on;
xlabel('Azimuth Angle  ()');
ylabel('Elevation Angle  ()');
title(sprintf('Targets with Beamforming
     Response >= %.2f', threshold));

% Loop over targets
plotted_any = false;
for k = 1:length(theta_targets)
    % Find closest angle indices in
    grid
    [~, idx_theta] = min(abs(
    angles_theta - theta_targets(k)));
    [~, idx_phi]   = min(abs(angles_phi
     - phi_targets(k)));

    % Check if power response passes
    threshold
    if R_2D(idx_phi, idx_theta) >=
    threshold
        plot(theta_targets(k),
    phi_targets(k), 'ro', 'MarkerSize',
    10, 'LineWidth', 2);
        text(theta_targets(k)+1,
    phi_targets(k), sprintf('P%d', k), '
    Color', 'r', 'FontWeight', 'bold');
        plotted_any = true;
    end
end

if ~plotted_any
    warning('No targets exceed the
    threshold of %.2f.', threshold);
end

%Beamforming: Sec 4
%Only keep points above threshold
threshold = 0.55;  % normalize power
    threshold (0 to 1)
%find indices of points above threshold
[row_idx, col_idx] = find(R_2D >=
    threshold);

% Convert indices to angles
theta_vals = angles_theta(col_idx);
phi_vals = angles_phi(row_idx);
power_vals = R_2D(sub2ind(size(R_2D),
    row_idx, col_idx));

% --- Plot ---
figure;
scatter(theta_vals, phi_vals, 30,
    power_vals, 'filled'); % color by
    power
```

```
colorbar;
xlabel('Azimuth Angle  ()');
ylabel('Elevation Angle  ()');
title(sprintf('Beamforming Points with
    Power  %.2f', threshold));
ylim([0, 80]);
grid on;


%Beamforming: Sec 5
eps_clustering = 3;  % cluster max
    distance in degrees

%Find points above threshold
[row_idx, col_idx] = find(R_2D >=
    threshold);

% Convert indices to angles
theta_vals = angles_theta(col_idx);
phi_vals = angles_phi(row_idx);

% To use cluster, combine angles
points = [theta_vals(:), phi_vals(:)];

% Cluster points using DBSCAN
cluster_labels = dbscan(points,
    eps_clustering, 2);

% Only use unique clusters
clusters = unique(cluster_labels);
clusters(clusters == -1) = [];

%Compute cluster by taking mean of
    points in cluster
centroids = zeros(length(clusters), 2);
for i = 1:length(clusters)
    cluster_points = points(
    cluster_labels == clusters(i), :);
    centroids(i, :) = mean(
    cluster_points, 1);
end

% Plot original points and cluster
    centers
figure; hold on;
scatter(points(:,1), points(:,2), 20, '
    b', 'filled');    % all points above
    threshold
scatter(centroids(:,1), centroids(:,2),
    100, 'r', 'filled'); % cluster
   centroids
xlabel('Azimuth Angle  ()');
ylabel('Elevation Angle  ()');
title('Clustered Beamforming Points and
    Centroids');
legend('Points', 'Cluster Centers');
grid on;
hold off;



%Beamforming: Sec 6
figure; hold on; grid on;
scatter(centroids(:,1), centroids(:,2),
    100, 'r', 'filled');
xlabel('Azimuth Angle  ()');
ylabel('Elevation Angle  ()');
title('Stickman Points');

for k = 1:size(centroids,1)
    text(centroids(k,1) + 0.5,
    centroids(k,2), sprintf('P%d', k),
    ...
        'FontWeight', 'bold', 'FontSize
    ', 12, 'Color', 'k');
end
connections = [
    1 2;   % L Foot to L Knee
    2 5;   % L Knee to Torso
    3 5;   % L Shoulder to Torso
    5 6;   % Torso to Head
    5 10;  % Torso to R Shoulder
    10 7;  % R Shoulder to R Hand
    5 9;   % Torso to R Knee
    8 9;   % R Knee to R Foot
    3 4;   % L Shoulder to L Hand
];



% Plot with lines
figure; hold on; grid on;
scatter(centroids(:,1), centroids(:,2),
    100, 'r', 'filled');
```

```
xlabel('Azimuth Angle  ()');
ylabel('Elevation Angle  ()');
title('Estimated angles of 2D stickman
    ');

% Draw connections
for i = 1:size(connections,1)
    p1 = connections(i,1);
    p2 = connections(i,2);
    plot([centroids(p1,1), centroids(p2
    ,1)], [centroids(p1,2), centroids(p2
    ,2)], 'b-', 'LineWidth', 2);
end

% Label
for k = 1:size(centroids,1)
    text(centroids(k,1) + 0.5,
    centroids(k,2), sprintf('P%d', k),
    ...
        'FontWeight', 'bold', 'FontSize
    ', 12, 'Color', 'k');
end

hold off;
```

9.8 2D Stickman

```
    %% Adjustable Range to Radar
Rb = 1;  % (meters)

%% x: lateral, z: vetical
% Points: head, waist, shoulders, hands
    , knees, feet
x_vals = [ ...
    0,     0,    -0.35,  0.35,  -0.3,
    0.3,  -0.3,  0.3,  -0.2,  0.2];  %
    lateral [m]

z_vals = [ ...
    2.08,  1.20,  1.40,   1.40,   0.40,
       0.40,  0.18,   0.18,   0.80,
    0.80]+0.1; % height [m]


K = length(x_vals);  %Plot K points
```

```
%% Convert to azimuth and Elevation
    Angles
theta_deg = atan2d(x_vals, Rb);      %
    Azimuth: -90 to +90
phi_deg   = atand(z_vals ./ Rb);     %
    Elevation: 0 to 90
% Round to remove noise
theta_rounded = round(theta_deg, 2);
phi_rounded   = round(phi_deg, 2);

% Extract unique sorted values
theta_unique = sort(unique(
    theta_rounded));
phi_unique   = sort(unique(phi_rounded)
    );

% Compute differences of theta and phi
    pairs
theta_diffs = diff(theta_unique);
phi_diffs   = diff(phi_unique);

% Get minimum nonzero spacing
min_az_spacing = min(theta_diffs);
min_el_spacing = min(phi_diffs);

% Display result
fprintf('\n--- Verified Angular
    Spacings ---\n');
fprintf('Azimuth angles used:   %s\n',
    mat2str(theta_unique));
fprintf('Elevation angles used: %s\n',
    mat2str(phi_unique));
fprintf('Minimum Azimuth Spacing:   %.2
    f\n', min_az_spacing);
fprintf('Minimum Elevation Spacing: %.2
    f\n', min_el_spacing);

%% Print Angles for Each Point
fprintf('--- Stickman Point Angles at R
     = %.2f m ---\n', Rb);
for k = 1:K
    fprintf('P%-2d: Azimuth = %+6.2f,
    Elevation = %6.2f\n', ...
        k, theta_deg(k), phi_deg(k));
end
```

```
%% Plot in Azimuth-Elevation Angle
    Space
figure;
plot(theta_deg, phi_deg, 'bo', '
    MarkerSize', 10, 'LineWidth', 2);
grid on;
xlabel('Azimuth Angle  ()');
ylabel('Elevation Angle  ()');
title(sprintf('Stickman Point Angles at
     R = %.1f m', Rb));
xlim([-30 30]);
ylim([0 90]);

% Label each point as P1, P2, ..., P9
hold on;
for k = 1:K
    text(theta_deg(k) + 1, phi_deg(k),
    sprintf('P%d', k), ...
        'FontSize', 10, 'Color', 'b', '
    FontWeight', 'bold');
end




for k = 1:K
    text(theta_deg(k) + 1, phi_deg(k),
    sprintf('P%d', k), ...
        'FontSize', 10, 'Color', 'b', '
    FontWeight', 'bold');
end

%Create connections
connections = [...
    1 2;  % Head to chest
    2 3;  % Chest to left shoulder
    2 4;  % Chest to right shoulder
    2 5;  % Chest to left knee
    2 6;  % Chest to right knee
    7 5;  % Left knee to left hand
    8 6;
    3 9;
    4 10]; % Right knee to right hand

% Draw lines connecting points
for i = 1:size(connections, 1)
    p1 = connections(i, 1);
    p2 = connections(i, 2);
    plot([theta_deg(p1), theta_deg(p2)
    ], [phi_deg(p1), phi_deg(p2)], ...
        'k-', 'LineWidth', 2);
end
```

9.9 3D Stickman

```
    % Joint coordinates
x = [0, 0, -0.35, 0.35, -0.3, 0.3,
    -0.3, 0.3, -0.2, 0.2];  % lateral
z = [2.08, 1.20, 1.40, 1.40, 0.40,
    0.40, 0.18, 0.18, 0.80, 0.80] + 0.1;
     % height
y = [1.6, 1.0, 1.4, 1.4, 1.6, 1.6, 1.6,
     1.6, 1.3, 1.3];  % depth



% Connections between joints
connections = [
    1 3;   % Head to left shoulder
    1 4;   % Head to right shoulder
    3 9;   % Left shoulder to left hand
    4 10;  % Right shoulder to right
    hand
    3 2;   % Left shoulder to hip
    4 2;   % Right shoulder to hip
    2 5;   % Hip to left knee
    2 6;   % Hip to right knee
    5 7;   % Left knee to left foot
    6 8;   % Right knee to right foot
];


figure;
hold on; grid on; axis equal;
xlabel('X (Lateral)'); ylabel('Y (Depth
    )'); zlabel('Z (Height)');
title('3D Stickman of Squatting Pose');
view([-30, 20]);

% Plot platform (to account for 0.28m
    height difference)
fill3([-0.4, 0.4, 0.4, -0.4], [1.6,
    1.6, 1.6, 1.6], [0, 0, 0.18, 0.18],
    [0.85 0.85 0.85], 'FaceAlpha', 0.4,
    'EdgeColor', 'none');
```

```
% Plot joints
scatter3(x, y, z, 100, 'r', 'filled');

% Plot joint connections
for i = 1:size(connections,1)
    idx1 = connections(i,1);
    idx2 = connections(i,2);
    plot3([x(idx1), x(idx2)], [y(idx1),
     y(idx2)], [z(idx1), z(idx2)], 'b',
    'LineWidth', 2);
end
```

9.10 steeringvector2

```
    function a = steering_vector2(Nx,
    Ny, d, lambda, theta_deg, phi_deg)
    theta = deg2rad(theta_deg);
    phi   = deg2rad(phi_deg);

    nx = 0:Nx-1;
    ny = 0:Ny-1;
    [nx_grid, ny_grid] = meshgrid(nx,
    ny);

    phase = 2 * pi * d / lambda * ...
        (nx_grid .* sin(phi) .* cos(
    theta) + ny_grid .* sin(phi) .* sin(
    theta));

    a = exp(1j * phase);
    a = reshape(a, [], 1);  % column
    vector
end
```

9.11 System Block Diagram

9.12 Simulation Parameters

Table 2: Parameter requirements and selected simulation values

Parameter	**Formula / Description**	**Required Value / Target**	**Chosen Value**
N (Number of subcarriers)	$N = \frac{BW}{\Delta f}$	≥ 6250	8192
Δf (Subcarrier spacing)	Set by FR2 standard (5G NR)	120 kHz	120 kHz
BW (Bandwidth)	$N \cdot \Delta f$	≥ 750 MHz	983.04 MHz
Δr (Range resolution)	$\frac{c_0}{2 \cdot BW}$	≤ 20 cm	15.26 cm
CP (Cyclic prefix length)	$\frac{1}{8}N$	Literature-based (typical)	1024
T (OFDM symbol duration)	$\frac{N+CP}{N \cdot \Delta f}$	–	9.39 μs
M (OFDM symbols)	$\frac{c_0}{2Tf_c v_{\min}}$	≥ 2221	2222
Δf_D (Doppler resolution)	$\frac{1}{MT}$	≤ 48.01 Hz	48.01 Hz
f_c (Carrier frequency)	Based on 5G FR2 band	$24.25 - 52.6$ GHz	28 GHz

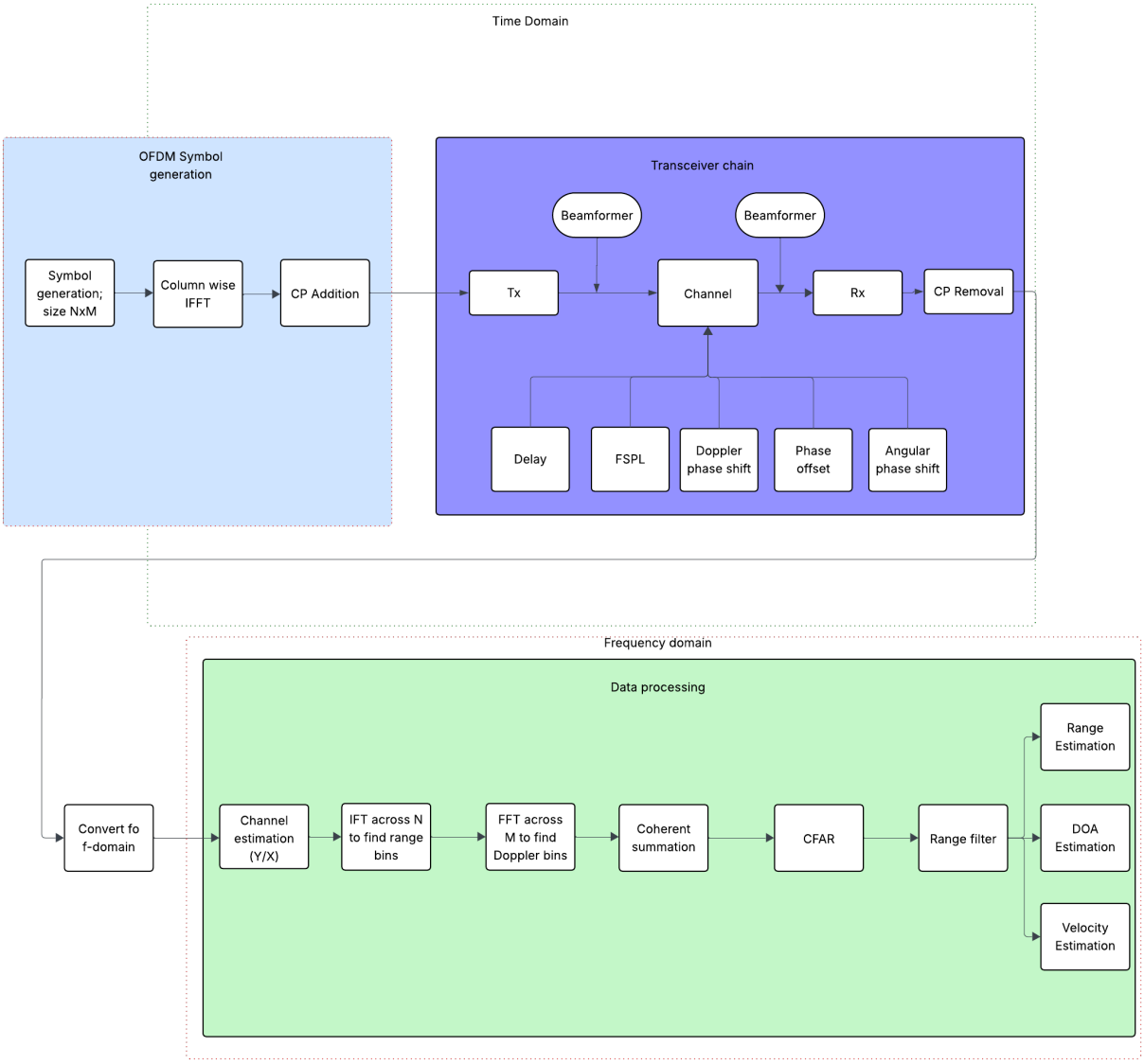


Figure 34: Full System Architecture (Idealized Implementation)