Cross-Frame Association of Through-Wall Handheld-Radar-Based Detections

by

Michael Hiebert

B.S. Computer Science and Engineering
Massachusetts Institute of Technology, 2021

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

Author .....
Department of Electrical Engineering and Computer Science
May 20, 2021

Certified by .....
Dr. Raoul Ouedraogo
Assistant Group Leader, MIT Lincoln Laboratory Group 45
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Cross-Frame Association of Through-Wall Handheld-Radar-Based Detections

by

Michael Hiebert

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Recent work with radar-based detection systems has demonstrated its efficacy in identifying [10], classifying [6] [8] [2] humans and animals, and even recognizing gestures [5] in low-light environments and through walls, cases where conventional vision-based systems fail. Most previous research has involved onsite (edge) gathering of data and offsite (non-edge) processing to produce detections, i.e. the experimental platforms have not been productionized nor tested live in applicable environments. Further, many of the proposed architectures rely on the specific motion paths of subjects to identify them. MIT Lincoln Laboratory's (MITLL) Group 45 has designed a prototype portable radar system capable of producing similar radar data to that collected in the aforementioned research and then identifying individuals in-frame, solely based on vital-signs and regardless of motion. I propose a computational architecture which can incorporate some of the previous advances with tracking in computer vision to detect and identify individuals while operating on the edge under the compute and power constraints of the handheld radar system in which it will be embedded.

Thesis Supervisor: Dr. Raoul Ouedraogo
Title: Assistant Group Leader, MIT Lincoln Laboratory Group 45

Acknowledgments

I would like to thank Raoul Ouedraogo and the rest of MIT LL's through-wall detection radar team–especially Andres Sisneros, Marc Vaillant, Eric Phelps, and Jay White Bear–for this incredible opportunity and all of the continued assistance they provided me for the duration of my tenure as a military fellow. It has been invaluable and appreciated always.

To my family–Mom (*especially* Mom, for whom this thesis is dedicated), Dad, and Patrick–your unwavering support and consistent guidance has helped me through every challenge I have faced over these last few years. Thank you for everything.

To my friends–Ian, Bella(s), Brandon, Rosie–thank you for every great day I have had at this school. It has not always been easy, but somehow we have managed to have each other's back at every step and have fun the whole way through. I will never forget the memories and you.

And finally, an enormous thank you to everyone at the Paul Revere Battalion, MIT's Army ROTC program. Deciding to commit and join the program here remains one of the best decisions I have made to date.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In security and disaster-relief applications, situational awareness–particularly the ability to locate, detect, and identify other individuals–is paramount. Extensive research has been conducted into computer-vision (CV) based detection and classification platforms capable of providing such awareness by recognizing humans, animals, and objects in video and images. However, even with higher quality cameras and models, the performance of such platforms quickly falls off in environments where ideal conditions cannot be satisfied, e.g. due to low-light, dust, occlusion, etc. - characteristics commonplace in, say, disaster-relief scenarios where such platforms would be most useful.

In particular, when conducting security and rescue operations in urban environments, it would be highly advantageous to be able to detect individuals across solid barriers to provide a headcount without first needing to dig through the rubble or check the inside of urban structures. As such, radar based alternatives for detection, capable of overcoming such barriers, have garnered much interest in recent years. Such systems have been shown to be successful in tasks such as detecting humans [8], gesture detection [5], posture estimation [12], transport classification [3], and even facial recognition [7].

1.1 Radar System

MIT LL has developed a handheld system (the user interface of which can be seen in Figure 1-1) capable of producing signal-based detections of stationary/moving individuals through concrete barriers, as well as the ranges from the radar at which these detections occur. Its novelty stems largely from its application: much of the prior work thus far assumes ideal conditions and thus operates with relatively noiseless signals. Group 45's system leverages novel denoising methods to provide binary classification of signals in realistic environments and at the edge.

At a high level, the radar system works by passing data from process to process through a series of messages. This message-passing system allows multiple types of computation to be conducted in parallel and in real-time. A visual overview can be seen in Figure 1-2.

Raw pulses from the radar are emitted and reflected off of objects in the scene it is facing. These pulses are received and denoised by our signal processing process, and then filtered to sproduce clean breathing and walking moving target indication (MTI) signals.

This processed signal is then fed into our detection process, which seeks to map our signal to a list of detections (if there are any targets in the scene). Some conventional methods of producing these detections involved finding the max peak of our signal above a certain threshold (for single-target detection), finding multiple peaks (for multi-target detection), and using 1-D Constant False Alarm Rate (1D CFAR) detection. At the time of this thesis' writing, there was machine learning (ML) infrastructure in place which could reliably produce a binary classification of whether or not there was at least one person in frame, but a reliable ML-based object detection system which could identify targets alongside their range from the radar was in development. Detections were fed into the tracker, which would associate detections to a unique track ID based on characteristics about their position and movement. Also at the time of this thesis' writing, the tracker consisted of a Kalman filter and some straightforward extensions of the filter to track targets solely based on posi-

tional information (henceforward referred to as the "Vogl tracker"). The results of this tracker are shown in Chapter 4.

1.2 Problem Formulation

We decided that the system, while robust in its current state, would benefit immensely from improvement to its detection and target-tracking processes. More specifically, detections could be more reliable and detailed (i.e. more specific in their classification of targets), and the tracker could leverage more than just positional information to inform and associate long-term tracks. Our research aims to augment the existing radar system with a number of features which will aid in improvement across the board, but specifically focuses on concrete improvement of the tracking system.

The problem we set forth to solve, then, was to expand upon existing infrastructure to create a detection and tracking pipeline which was 1) competitively performant 2) quantitatively evaluated and 3) easily iterated upon. For this, we refactored our data pipeline, implemented evaluation and metric-producing systems, reworked how detections are handled, and created a new tracker which could incorporate more information for more informed results.

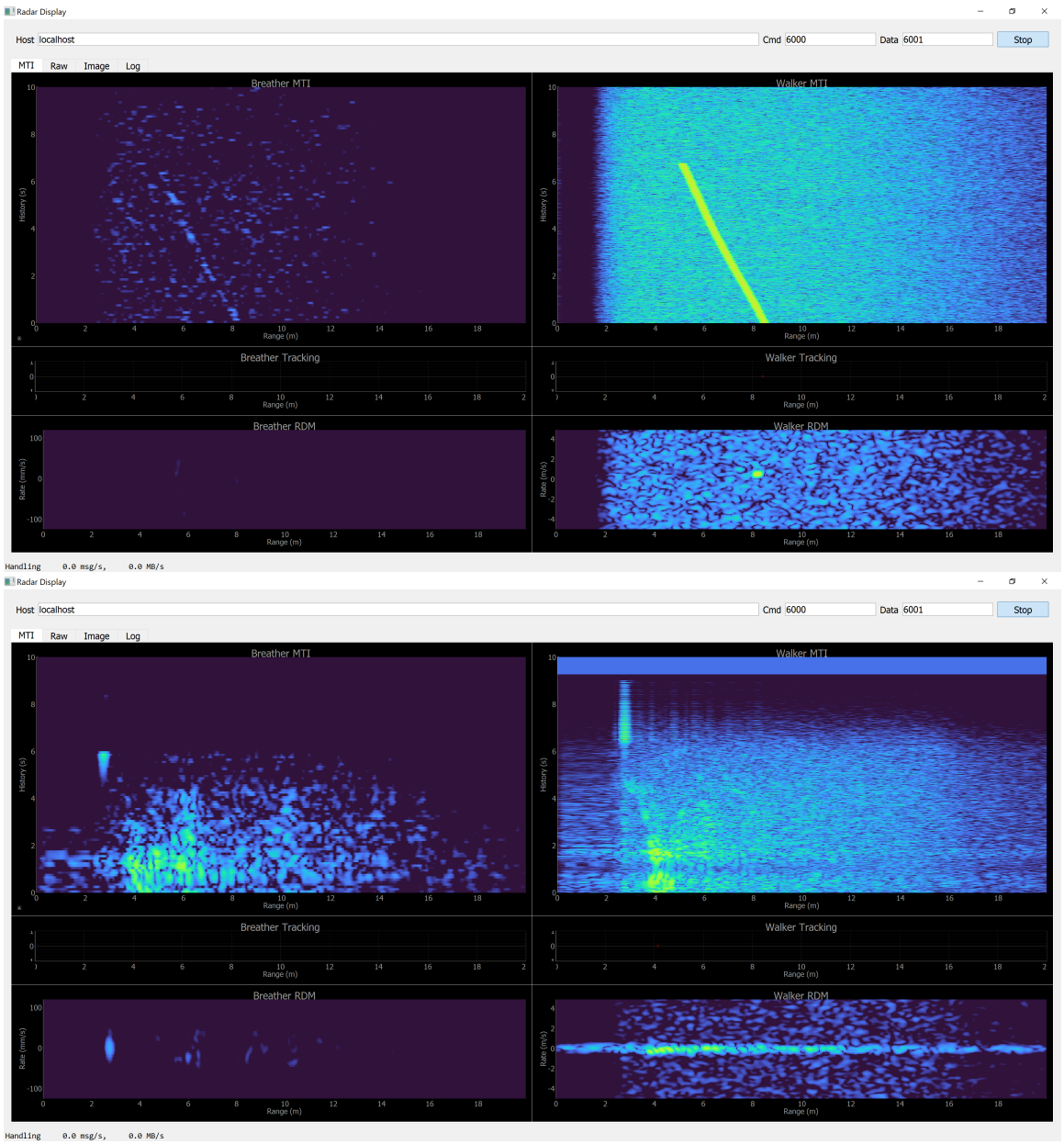

Figure 1-1: Example screengrabs of the Radar's current GUI. Relatively clean synthetic data of an individual walking away from the radar is shown on the top, in contrast with the noisier real data of an individual moving and sitting at a roughly fixed distance on the bottom.

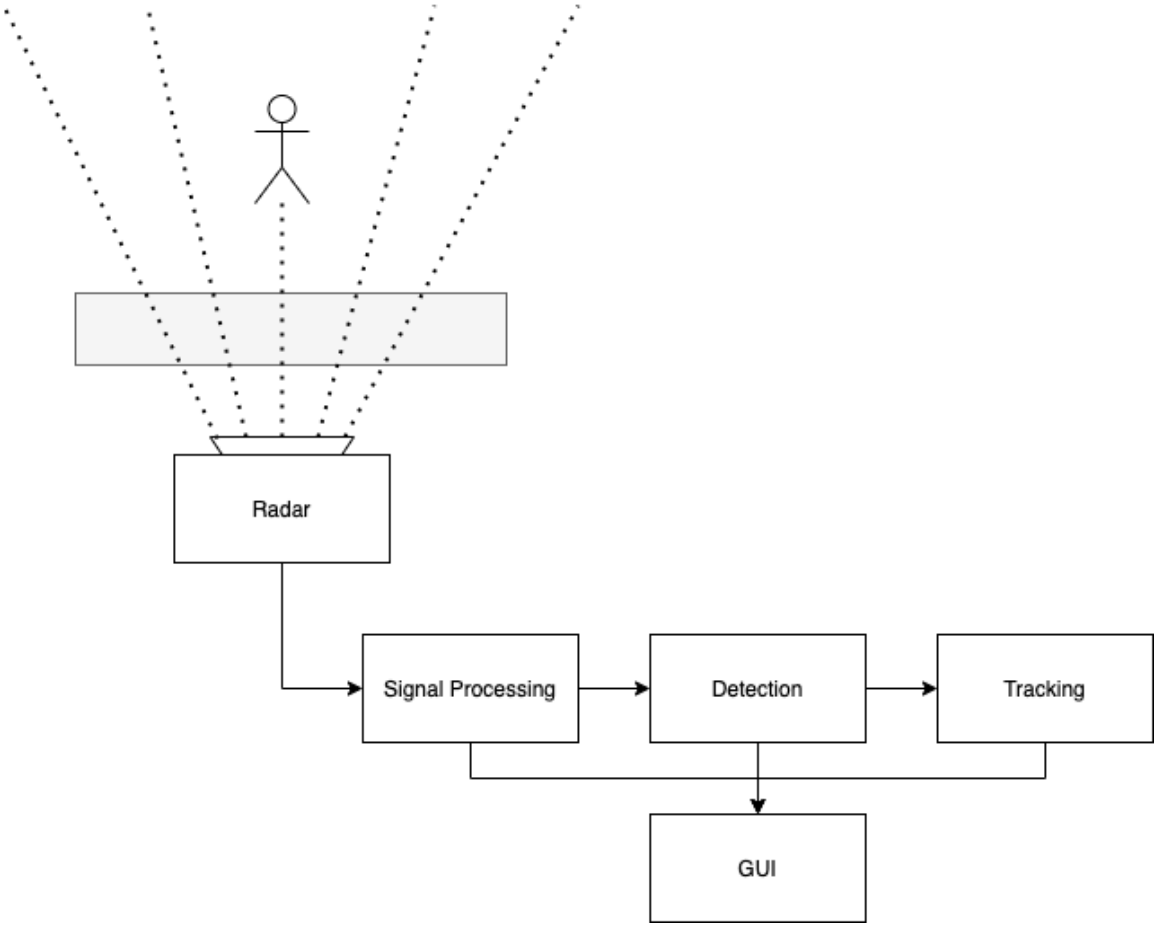

Figure 1-2: High-level overview of radar architecture

Chapter 2

Related Work

There has been much recent work exploring the ability of radar based systems to perform classification and detection tasks where standard computer vision systems fail. Radar devices transmit electromagnetic signal and receive the reflections of these signals off of objects the waves collide with. These reflections contain information about the motions of the objects with which they collide due to the Doppler effect. In the case of identifying human presence and motion, for example, we can combine the Doppler signals of all moving parts of the human body to generate a unique micro-Doppler (MD) signature. These signatures can then be passed as input to more complex classification pipelines to detect and classify humans and animals.

Otero [8] was able to create a binary classifier which could identify humans through walls and in low light conditions with a low power/cost commercial-off-the-shelf (COTS) X-band microwave motion sensing module by analyzing the unique doppler signatures of various moving components of the body when walking (i.e. the cadence of the swing of the arms or legs). Through this unique doppler signature, he was also able to demonstrate the differences and classification potential in discerning human from animal presence. In this work, targets were required to walk to or from the radar at fixed speed.

Garreau et al. were able to employ a similar system to identify 13 distinct individuals and classify their gender with near perfect ($\geq 90\%$) accuracy. Their architecture, designed to be highly computationally efficient, combined several simple

pre-processing steps with k-means clustering to classify individuals and predict gender by comparing euclidean distance to ground truth clusters. Similar to Otero, this work required all subjects to walk on a treadmill facing the radar with varying speed.

Much of the earlier work using gait analysis to detect humans required subjects to walk in specific, predetermined motion paths during data collection. Vandersmissen et al. [10] removed this restriction in their study, permitting subjects to walk freely in any direction within a room, using Deep Convolutional Neural Networks (DCNNs) to identify 5 separate individuals based on their MD signatures.

Additionally noting that much of the previous work relied on manual feature engineering to achieve relatively high levels of accuracy, Le, Phung et al. [6] implemented a deep autoencoder to learn these features autonomously and run classification based on these hidden encodings.

Several other studies have demonstrated the efficacy of such platforms to classify humans even without macro-level translational motion. An MITLL study from 2012, Peabody et al. [9], proposed a vehicle-mounted radar sensor which was demonstrated to detect stationary humans even through 20cm of concrete.

MIT's own CSAIL (Zhao et al [12]) explored the advantages of multimodal training in producing a model which could successfully estimate pose even when the detected individual travelled behind walls or was otherwise occluded from the frame.

MIT LL has synthesized much of this work and has been focused on its application in producing a handheld (roughly laptop sized) sensor capable of acting as a portable through-wall sensor for human and animal detection. The novelty of such a system rests not just on its synthesis of existing work but also its real-world implementation into a system expected to function in environments far more dynamic than the static testing environments of nearly all previous studies in this area. Currently, it relies on vital signs to run binary classification on signal, determining presence and range of humans in frame. In order to successfully classify and/or associate these detections through time, more information may be required.

Building off of the SORT Algorithm [1], Wojke et al. [11] incorporated some kind of encoded appearance information to augment the positional prediction to compen-

sate for lost tracks after occlusion. With this intrinsic, non-positional information included in the tracker's predictions, they were able to see a marked decrease in identity-swaps across large scale object detection and tracking experiments.

Chapter 3

Methodology

3.1 Data Collection

A significant portion of this effort was the generation and collection of new data. Particularly data with specific enough labels that algorithmic performance could be evaluated in a quantitative sense and without strict human verification.

Data collection for this specific endeavor began at the end of 2020 using the handheld radar device at Lincoln Laboratory, and existing radar data encompassing a wide variety of different contexts and scenarios was provided, dating back several years.

3.1.1 Manual Collection

We devised a series of scenario partitions that would encompass key use cases of the radar system. These partitions can be located in Table 3.1.

Data collected during this project was either accompanied by a text file describing the subject in frame (for scenarios involving subjects remaining still), or orally annotated with an audio file synchronized to the start of radar data collection (for scenarios involving more complex movement).

Other radar data was centralized from a number of different sources spanning several years, geographical locations, subjects, and scenarios. An initial challenge

Poses	Actions	Wall Material	Actors
Sitting:	Breathing:	Wood	Male
Standing	Not Breathing	Concrete	Female
Lying	Walking	Door	Single Target
Crouching	Running	Plaster	Multi-Target
	Still	Metal	
	Gesturing		
	Talking		
	Exercising		

Table 3.1: Scenario partitions we sought to collect in our data

arose with how this data was labelled: some data was described with an accompanying text file, though start/end times of certain actions were not exact, and some data was only described with a title. These inconsistencies in the labelling of this data made their use somewhat cumbersome: they were useful for qualitative assessment of the radar platform in a binary sense (either the target is or is not mostly there, and doing what they are predicted to be doing), but less so for quantitative analysis.

How the collected raw data and the miscellaneous existing raw data were converted into a central, consistent, and usable format is described in chapter 3.1.3.

3.1.2 Synthetic Generation

While nothing quite compares to real world data, the signals we were processing could also be reasonably simulated by synthetically generating the pulse data we would expect to receive from the radar itself.

To this end, we created a tool which could take high level features, such as *"One target walking from 3m to 10m starting at 5s and then ending at 15s, another target starting at 20m and walking to 7m starting at 10s and ending at 25s"* and convert them into pulse data which could then be ingested and processed by other pieces of the radar's detection pipeline. These "scenes" encompassed both an abstract understanding of the figures and actions going on in-frame, as well as the ground-truth signals and labels that accompanied them. A visual depiction of the additions to existing infrastructure is shown in Figure 3-1.

In our implementation, scenes were composed of even more fundamental building

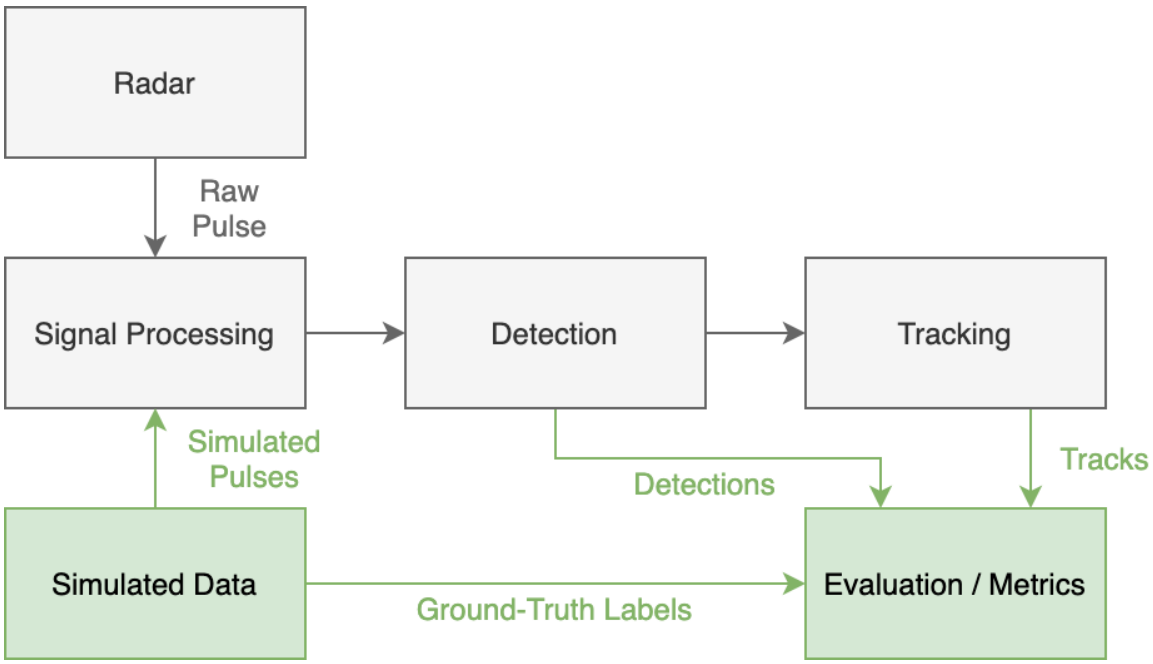

Figure 3-1: The additional components added to the radar processing pipeline to facilitate training and evaluation of algorithmic performance

blocks: targets. Our targets, described in more depth in Table 3.2, each represented specific motion patterns which could be combined together to form complex scenarios in relatively few lines of code.

3.1.3 Labeling for Object Detection

A helpful feature of these scenes were that they produced pulse-by-pulse labels which could then be used to assess the predicted detections of the existing (and proposed) radar infrastructure. This property allowed them to not just describe–and then produce data for–artificial scenarios, but the trove of *existing* data as well. The labels and titles associated with the previously collected samples could be used to inform the creation of a scene which mimicked the samples themselves, and could output labels which very nearly resembled ground truth. Exact pulse alignment proved to be a nearly impossible challenge without extensive manual effort for each sample, but label/prediction discrepancies existing at the edges of their overlap were written off under the assumption that the relatively overwhelming overlap would prove such

Target Type	Description
Noise	The generation of random noise in our output signal on each pulse
Clutter	A sub-type of Noise where the per-pulse signal is fixed, instead of random
Stationary	A non-moving target that still reflects pulses, e.g. inanimate objects, perfectly still people
Linear	A target moving from point A to B linearly, e.g. a person walking in its most trivial case
Sinusoidal	A target moving according to a sinusoidal motion pattern about a fixed center, e.g. a person breathing regularly and consistently
LinearSinusoidal	A target moving linearly from point A to B while their exact position oscillates about their moving center according to a fixed center, e.g. a more true-to-form representation of a human walking motion
CompoundMotion	A target whose motion is some compounded combination of any of the target motion patterns above, e.g. an irregular breathing rhythm
MultiMotion	A target whose motion is some sequence of any of the target motion patterns above, e.g. a person who walks, stops, then starts walking again

Table 3.2: A description of the various targets of our data generation system, implemented as part of the detection and tracking system creation.

discrepancies negligible in faithfully evaluating performance.

3.2 Metrics and Evaluation

With a pipeline in place to accrue data samples and generate their associated ground truth, the next step was to develop infrastructure to quantitatively evaluate our system's performance.

3.2.1 Detection Evaluation

The two most important metrics that we focused on from an object detection perspective were precision (if the radar predicts that there is a person in frame, how likely is that to be true?) and recall (if a person is in frame, how likely is our radar

to detect them?).

We calculate these metrics by organizing our labels and predictions into two sets $\mathcal{D}^L$ and $\mathcal{D}^P$. We consider $D_t^L \in \mathcal{D}^L$ and $D_t^P \in \mathcal{D}^P$ to be the set of all ground truth and predicted detections, respectively, at timestep t. We then compute our true positives, false positives, and false negatives and further compute precision, recall, and fscore values with Algorithm 1.

3.2.2 Tracking Evaluation

Tracking performance, while very dependent on the efficacy of the detection system it is built over, requires a different perspective. In our tracking evaluation, our ground truth tracks keep a record of which predicted tracks overlap temporally and positionally throughout the course of the datastream. A ground truth track t_g has knowledge of its intended length and start/end times, as well as a set of predicted subtracks $P = \{t_p\}$. At evaluation time, each track calculates the percentage of uptime, $u_a = \frac{\Sigma_P |t_p|}{|t_g|}$, as well as computing a continuity score for the predicted tracks. Our continuity score c is computed with:

$$c = \frac{\sum_{i=1} \gamma^{i-1} |t_p^{(i)}|}{|t_g|} \tag{3.1}$$

where γ is some decay factor and P is presorted in descending order by track length before computing continuity.

3.3 Target Detection

Detection and tracking are two sides of the same coin. Without a solid and reliable detection architecture running under the hood, no tracker will ever produce useful tracks. As such, work was done to tweak existing detection platforms as well as support the development of more robust detection methods.

Algorithm 1: Our detection evaluation algorithm. Interpolation is described in Section 3.4.3

Result: Returns a precision, recall, and f1 score value for these predictions

```
// Initialize evaluation parameters
```
$\epsilon_i \leftarrow$ ... `// Number of frames to skip up front`

$\sigma \leftarrow$ ... `// Count every other` σ `frames`

$\epsilon_f \leftarrow$ ... `// Frame overlap threshold`

$\epsilon_c \leftarrow$ ... `// Confidence threshold`

$\epsilon_r \leftarrow$ ... `// Range overlap threshold`

`// Initialize our data`

$\mathcal{D}^L \leftarrow$ ... `// Label data`

$\mathcal{D}^P \leftarrow$ ... `// Predicted data`

if $\epsilon_f > 1$ **then**
  $\mathcal{D}^P \leftarrow \text{INTERPOLATE}(\mathcal{D}^P, \epsilon_f)$;
end

$p_t \leftarrow$ `// True positives`

$p_f \leftarrow$ `// False positives`

$n_f \leftarrow$ `// False negatives`

for $D_i^L \in \mathcal{D}^L$ **do**
  if $i < \epsilon_i$ *OR* $i \ mod \ \sigma \neq 0$ **then**
    CONTINUE
  end
  if $D_i^P \notin \mathcal{D}^P$ **then**
    $n_f \leftarrow n_f + |D_i^L|$;
    CONTINUE;
  end
  for $d_L \in D_i^L$ **do**
    $p_t \leftarrow p_t + \mathbb{1}(\exists d_P \in D_i^P.\ d_P == d_L)$;
    $n_f \leftarrow n_f + \mathbb{1}(\nexists d_P \in D_i^P.\ d_P == d_L)$;
  end
  for $d_P \in D_i^P$ **do**
    $p_f \leftarrow p_f + \mathbb{1}(d_P \notin D_i^L)$;
  end
end

precision $\leftarrow p_t/(p_t + p_f)$;

recall $\leftarrow p_t/(p_t + n_f)$;

fscore $\leftarrow 2 * ((\text{precision} * \text{recall})/(\text{precision} + \text{recall}))$

3.3.1 Clustering

Due to the reflection of our pulses off of targets and our environment, correct detections of moving or breathing targets would often be accompanied by several weaker detections mirrored (more-or-less) on either side. This phenomenon would result in levels of predicted detections roughly five times higher than what is truthfully there, dramatically reducing the evaluated precision of the system.

To remedy this, we modified our detection process to cluster detections immediately following their prediction and before being passed to our tracking process and subsequently delivered to the end user via the attached GUI. Detections–acting as nodes in our graph–would be clustered according to a raw, thresholded distance. Nodes within this distance would be labelled as one detection, combined into a single all-encompassing detection predicted to be at the average range of all involved nodes.

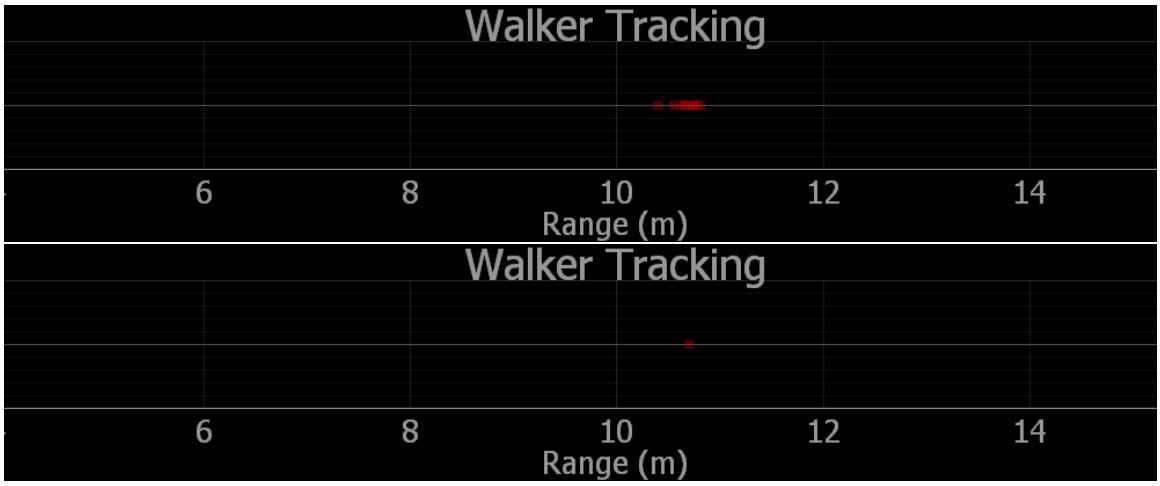

Figure 3-2: The difference between no clustering (top) and clustering (bottom) when applied immediately after detection and before tracking.

3.4 Tracking: RadarSORT

Our contributions to the tracking architecture were largely inspired by the Simple Online and Realtime Tracking (SORT) algorithm [1] and its successor, DeepSORT [11]. The premise of the latter's novelty essentially being the addition of some appearance-based similarity metric to associate tracks.

In situations where our targets were "occluded" (to use the equivalent computer vision term) when two or more targets were superimposed, tracks were lost as the detections all fused into one. This would often result in a brand new track ID being assigned after the individuals involved passed or started moving again. To mitigate this, we proposed a separate extension of SORT, based largely on the same premise and adapted for use in this radar detection domain.

The high level concept of RadarSORT is sketched out in Figure 3-3.

3.4.1 State Estimation

We define our tracking scenario on the relatively simple three-dimensional space $(x, \dot{x}, \ddot{x})$ representing a target's position x, their velocity $\dot{x}$, and their acceleration $\ddot{x}$. We simply use the detected positions x as direct observations of our object's state.

On each update at timestep t, we propagate all of our live tracks forward in time to retrieve their predicted Kalman states at t. Tracks keep track of their time since update, or t_{update}. If a track is associated at a given timestep, it is considered "active." If it was not associated but its t_{update} is under a provided threshold, it is considered "stale", but still live. Unlike many other computer vision tracking systems, we *do not* ever fully drop tracks, and stale tracks can be re-associated with detections if they are found to be an optimal match.

3.4.2 Linear Assignment

We define a composition of two similarity metrics to generate a cost matrix and then solve the corresponding linear assignment problem. As in [11], the first of these similarity metrics describes the relative position of detections as they pertain to predicted positions of our tracks, and the second similarity metric describes properties intrinsic to both the detection and our track, something representative of the target, agnostic of position: gait.

The first of these metrics we represent as the Mahalanobis distance between detections and predicted Kalman states:

$$d_{pos}(y_p, x_p) = \gamma^* + (x_p - y_p)^T S^{-1}(x_p - y_p) \tag{3.2}$$

where x_p represents the position of some detection x, y_p represents the predicted position of some track y, and S is the corresponding covariance matrix of y. γ^* represents some temporal discount factor defined as:

$$\gamma^* = \gamma |t_y - t_x| \tag{3.3}$$

with γ being some predefined temporal discount, t_y being the last update time of track y, and t_x being the time of detection for x.

Alternatively, Euclidean distance was also used to produce a distance-based cost:

$$d'_{pos}(y_p, x_p) = \gamma^* + |y_p - x_p| \tag{3.4}$$

For any given track-detection pairing i, j, we create an indicator variable

$$b^{i,j}_{pos} = \mathbb{1}(d_{pos}(i, j) \leq \tau_{pos}) \tag{3.5}$$

where τ_{pos} is a predefined threshold value.

Now for the second of these metrics. For any detection x_j, we also retrieve its local neighborhood in the processed signal n_j. For any given track y_i, we also store a history of up to the previous 100 neighborhoods $N_h^{(i)}$. When comparing the similarity between track i and detection j, we define our second similarity metric as follows:

$$d_{sim}(x_j, y_i) = \texttt{min}\{1 - \hat{n_i}\hat{n_j}^T \mid n_i \in N_h^{(i)}\} \tag{3.6}$$

or, in other words, the minimum cosine distance between the neighborhood of our detection and the neighborhood history of our track. In this case, our encoder function is just a normalizing function for the input vectors. In theory, this could be any function $E : \mathbb{R}^{|n|} \to \mathbb{R}^N$. More generally, our similarity metric would be:

$$d_{sim}(x_j, y_i) = \texttt{min}\{1 - \hat{E(n_i)}\hat{E(n_j)}^T \mid n_i \in N_h^{(i)}\} \tag{3.7}$$

Once again, we define an indicator for this similarity metric, as well:

$$b_{sim}^{i,j} = \mathbb{1}(d_{sim}(i,j) \leq \tau_{sim}) \tag{3.8}$$

Our cost matrix is formed as a weighted sum of both of these distance metrics. We can define the elements of our cost matrix C as:

$$c_{i,j} = \lambda d_{pos} + (1-\lambda) d_{sim} \tag{3.9}$$

Increasing λ places more weight on the position-based similarity criterion, and decreasing it places more weight on the intrinsic similarity-based criterion. Unless otherwise stated, we use a λ-value of 0.75 in our experiments.

3.4.3 Detection Interpolation

Even in successful relatively successful cases, our detection system often had several-pulse gaps in multi-pulse tracks of a target. While virtually unrecognizable to the human eye, such gaps posed issues in how machines evaluated the accuracy of our predicted tracks or how summaries of scenes were generated after the fact. We were able to retroactively interpolate detections based on high-confidence measurements within our tracks.

For some track i with position $p_t^{(i)}$ at timestep t, we define some arbitrary interpolation bound ϵ such that if t has high-confidence measurements at timestep t_1 and t_2, $t_2 - t_1 < \epsilon$, and $t_2 > t_1$, we can interpolate positions with the following equation:

$$p_t^{(i)} = p_{t_1}^{(i)} + (t - t_1)(p_{t_2}^{(i)} - p_{t_1}^{(i)}) \tag{3.10}$$

This is particularly useful for smoothing out predictions in a way that can be evaluated consistently with expected prediction formats.

3.5 MetaTracking

Even the most robust trackers are prone to identity switches or lost tracks if the underlying detection platform breaks down. Especially in this domain, where detection architectures in austere environments are not thoroughly established, such errors are rampant.

In realtime, it is impossible to discern the ground truth underlying such events. But in retrospect, when all the tracks are laid out, a human can reasonably infer which tracks actually belong to a single target, though this is more difficult for a machine. Such an exercise is useful for self-correction and scene summarizing on frequencies just below realtime.

To tackle this particular problem, we propose MetaTracking, or the formation of new tracks based not off of detections, but whole tracks. MetaTracking turns our problem into a Meta-Clustering problem, where we must connect entire clusters (tracks) of detections with one another to yield the most optimal result.

3.5.1 Potency

At the onset of any uncertain clustering, there are a number of different states that the graph (and each of its nodes) could be in. Until it has been reasonably checked, our clustering could evolve into any one of a number of different ground-truth clusterings. We have taken this concept and called it graph potency, after a similar property possessed by stem cells.

Potency is calculated by determining how many other existing clusters a specific cluster could be, and graph potency is just the cluster-wise sum of this value.

Take some Meta Cluster $\mathcal{M}^{(i)}$ which keeps track of $\mathcal{S}^{(i)}$, the set of other Meta Clusters that belong to the same ground truth cluster as itself, $\mathcal{C}^{(i)}$, the set of other Meta Clusters that *could* belong to the same ground truth cluster as itself, and $\mathcal{D}^{(i)}$, the set of other Meta Clusters that *do not* belong to the same ground truth cluster as itself. Mathematically, we define the potency $\pi^{G}_{(t)}$ of graph G at any given timestep t as:

$$\pi_{(t)}^{G} = \Sigma_i |\mathcal{C}_{(t)}^{(i)}| \tag{3.11}$$

for all $\mathcal{M}^{(i)} \in G$.

3.5.2 Meta-Clustering Algorithm

We first take all of our tracks and segment them into their components by breaking full motion tracks down into a disjoint set of relatively continuous points. In an example case where a track is lost for some amount of time and then regained, resulting in a gap neighbored on either side by an otherwise uninterrupted track, we could form two segments from the full track.

Then, we take each segment and create a new one-segment track out of it. We further assign each new track to a corresponding meta-track. These meta-tracks can be thought of as abstract clusters, monitoring which other clusters it is, could be, and could not be related to.

Next, we iterate through all possible combinations of these segments until the potency of our track alignment is 0. At each timestep, we randomly suggest a pairing of two tracks that could be related to one another, and compare them by some objective function d_{seg}. If d_{seg} is under some acceptable threshold, we consider the pairing to be good. Otherwise, we consider it to be bad.

In a good pairing, we merge the information of both meta-tracks into one, meaning that all of the information about what tracks a specific track could or could not be related to is transferred to its new paired track, and vice versa. In a bad pairing, we simply update the "could not be" sets of both meta-tracks and their connected tracks.

Upon completion of our loop, we prune out tracks that fall below a certain size threshold and then return a list of our newly merged tracks. A more succinct explanation of this process can be found in Algorithm 2.

Algorithm 2: Our MetaTracking algorithm

Result: Segment and merge existing tracks into smoother meta-tracks

$\mathcal{T} \leftarrow$ tracks $t_1, ...t_N$ `// Initialize tracks`

$\mathcal{S} \leftarrow$ SEGMENT($\mathcal{T}$) `// Segment tracks`

$\mathcal{T} \leftarrow \{T(s_i) \mid s_i \in \mathcal{S}\}$ `// Create tracks from new segments`

$\mathcal{M} \leftarrow \{M_i(T_i) \mid T_i \in \mathcal{T}\}$ `// Create meta-tracks`

$\pi \leftarrow \Sigma_i |\mathcal{C}^{(i)}|$ `// Set potency`

while $\pi > 0$ **do**

  $M_i, M_j \leftarrow$ SUGGEST_PAIRING();

  if $d_{seg}(M_i, M_j) \leq \epsilon$ **then**

    MARK_GOOD_PAIRING(M_i, M_j);

  else

    MARK_BAD_PAIRING(M_i, M_j);

  end

  $\pi \leftarrow \Sigma_i |\mathcal{C}^{(i)}|$ `// Set potency`

end

PRUNE_TRACKS();

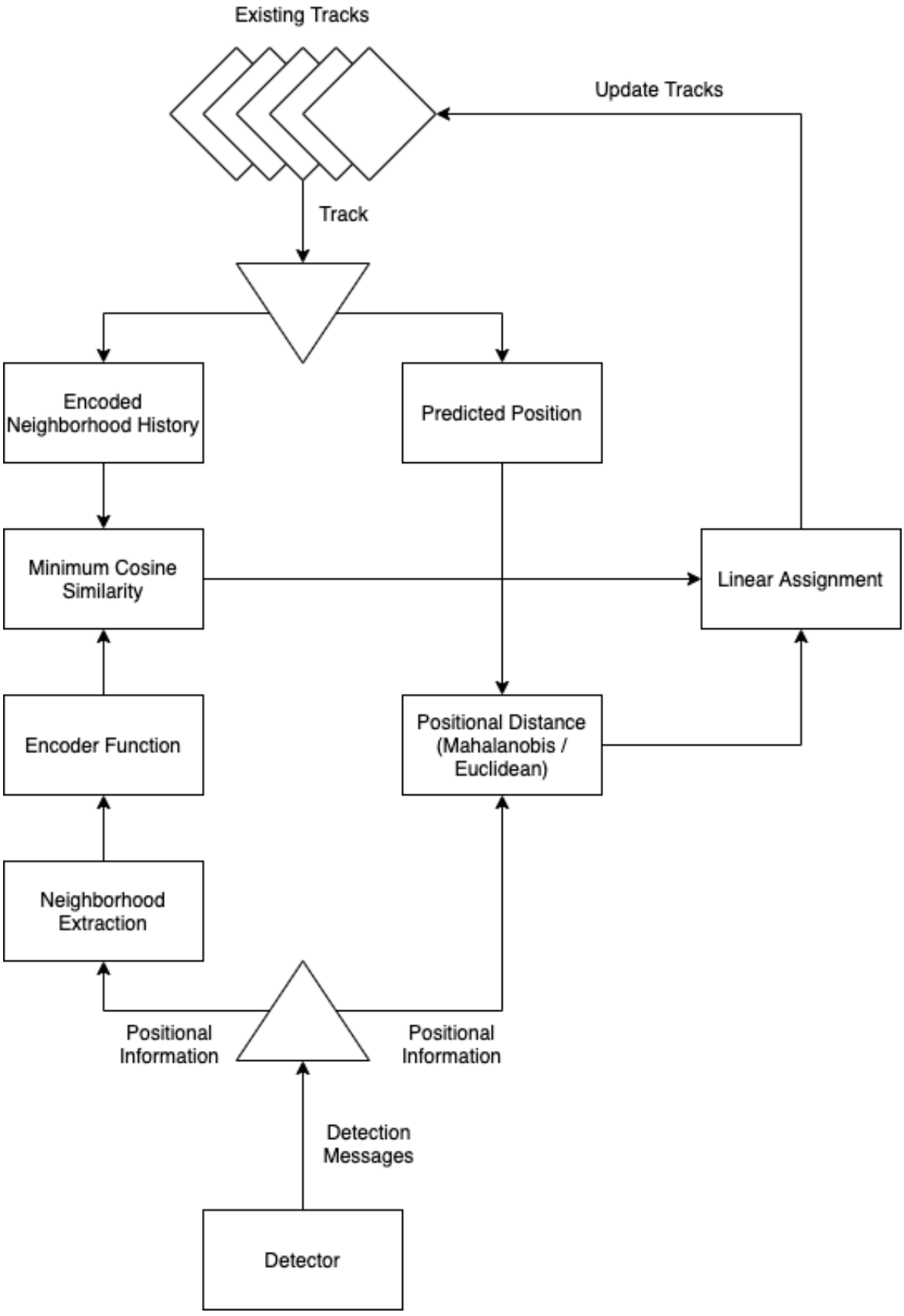

Figure 3-3: RadarSORT's internal architecture.

Chapter 4

Results

We compared our proposed detection and tracking system with some existing options as a baseline on both real and synthetically generated scenes. We describe an interesting subset of these experiments here to highlight various improvements and drawbacks of RadarSORT with alternative infrastructure.

Currently, these results are visualized in realtime in the GUI shown in Figure 1-1. For analysis, we plot our tracks over time to evaluate relative performance of our algorithm.

4.1 Synthetic Experiments

4.1.1 Experiment 1: Single Target Mutli-Motion Path

In this experiment, a synthetic target walks from 5m to 15m, stands still while breathing, and then returns to the original 5m position. The challenge in this scenario is the inconsistency in the target's movement throughout. For a tracker to be successful, it must adjust over time to the changing motion function that our target exhibits. Figure 4-1 shows RadarSORT's performance while Figure 4-2 shows the Vogl Tracker's performance.

The results of this first experiment show clear trends that we see persist throughout all experiments: Vogl is generally a lot tighter of a track positionally, but is

highly prone to dropping a target and forcing an identity switch. RadarSORT, on the other hand, is quite good at approximating a target's motion path, reducing identity switches, and even reassociating lost tracks as the target exhibits familiar motion patterns through similar regions of the scene (so the neighborhood's encoded similarity contributes to the reassociation).

In this experiment, we see the Vogl tracker very nearly mimic the ground truth, but across roughly 20 separate tracks. In other words: rather high f1 score and rather low continuity. RadarSORT, on the other hand, has an alright f1 score (it still approximates the motion to a significant degree), while keeping with a far more continuous track.

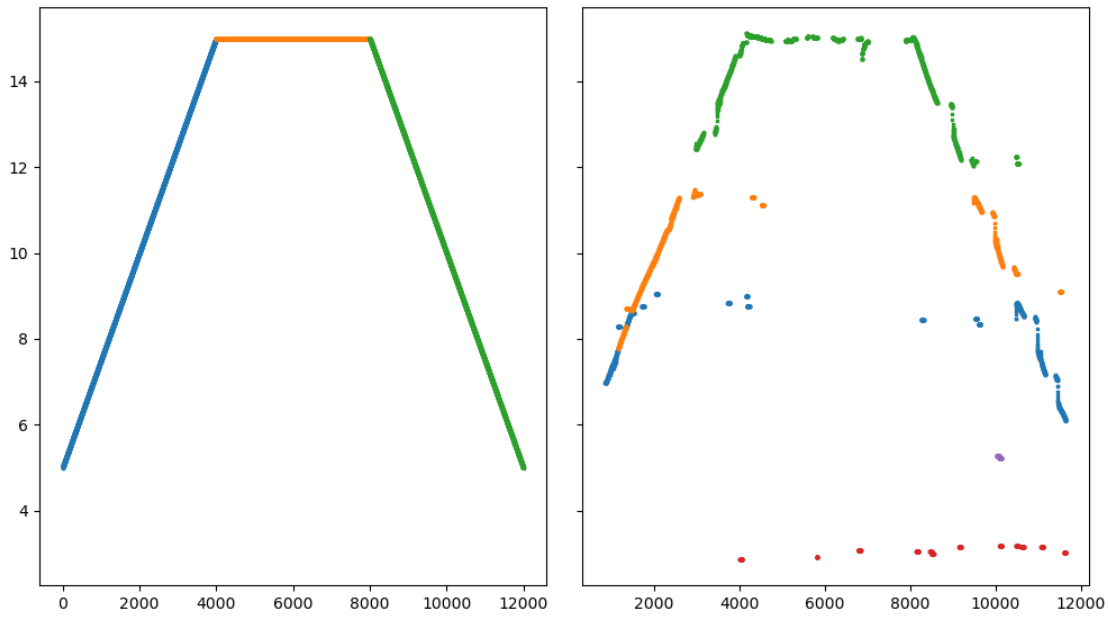

Figure 4-1: All tracks visualized of the single target's motion path. On the left is the ground truth, with the three components of motion highlighted in different colors. On the right is RadarSORT's predicted path, with colors representing track IDs. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

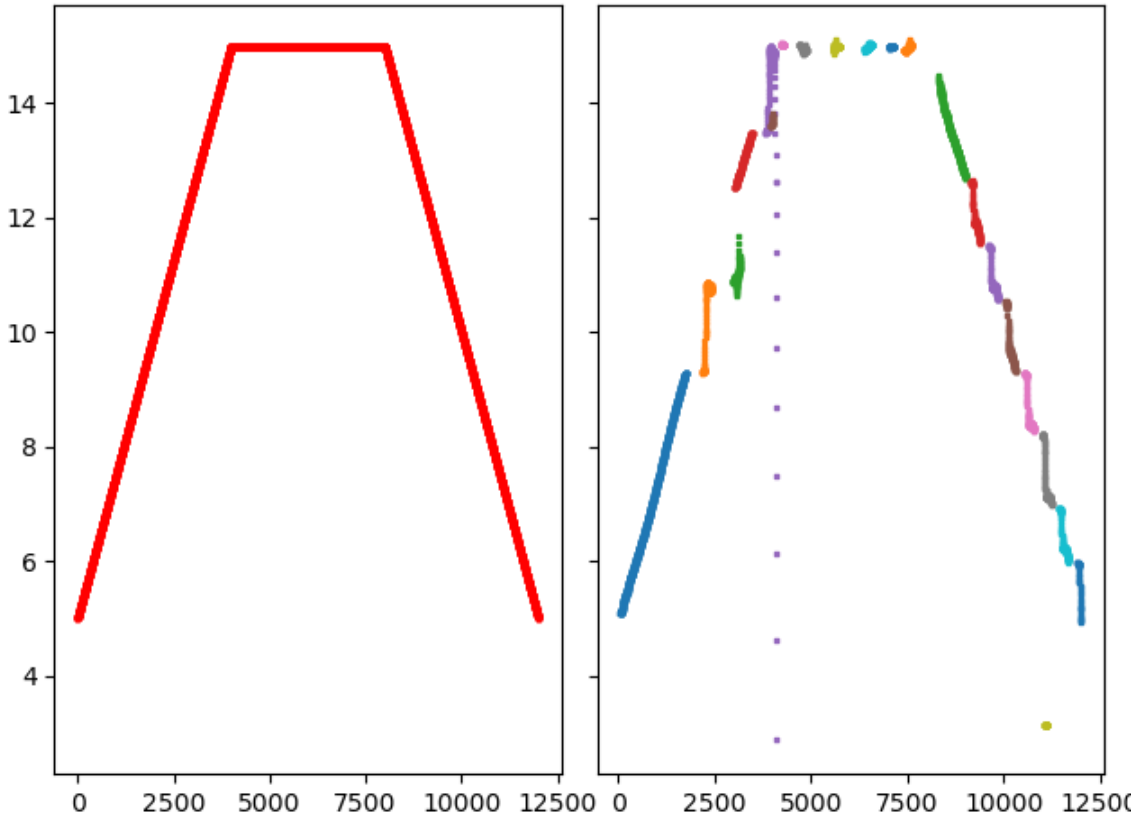

Figure 4-2: All tracks visualized of the single target's motion path. On the left is the ground truth, with the whole of the motion in red. On the right is the predicted path using the Vogl tracker, with colors representing track IDs. NOTE: Due to matplotlib's own color rotation, the colors on the right of the motion plot do *not* correspond to the same track as their respective colored track on the left. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

4.1.2 Experiment 2: Two Targets Crossing

In this experiment, two synthetic targets walk from opposite extremes of the radar (far and near, respectively) and walk to the opposite extreme (near and far, respectively). They cross at a common midpoint. The true challenge here is that the tracker must infer which motion path corresponds to which subsequent direction at the cross.

The visualized results of the SORT-based tracker can be found in Figure 4-3 and the visualized results of the Vogl tracker can be found in Figure 4-4.

Once again, we see the Vogl tracker produce many identity switches. Overall, the

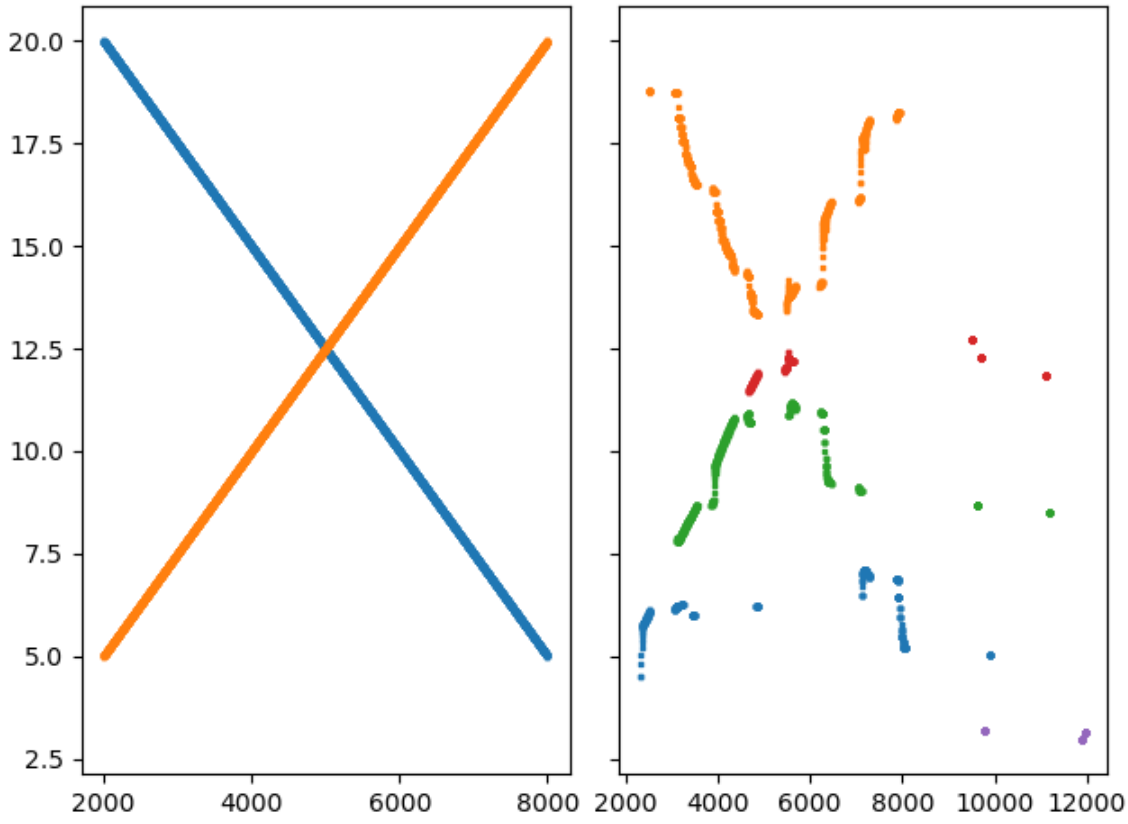

Figure 4-3: All tracks visualized of the two-target motion paths. On the left is the ground truth, with each ground truth track as a distinct color. On the right are the predicted tracks of RadarSORT. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

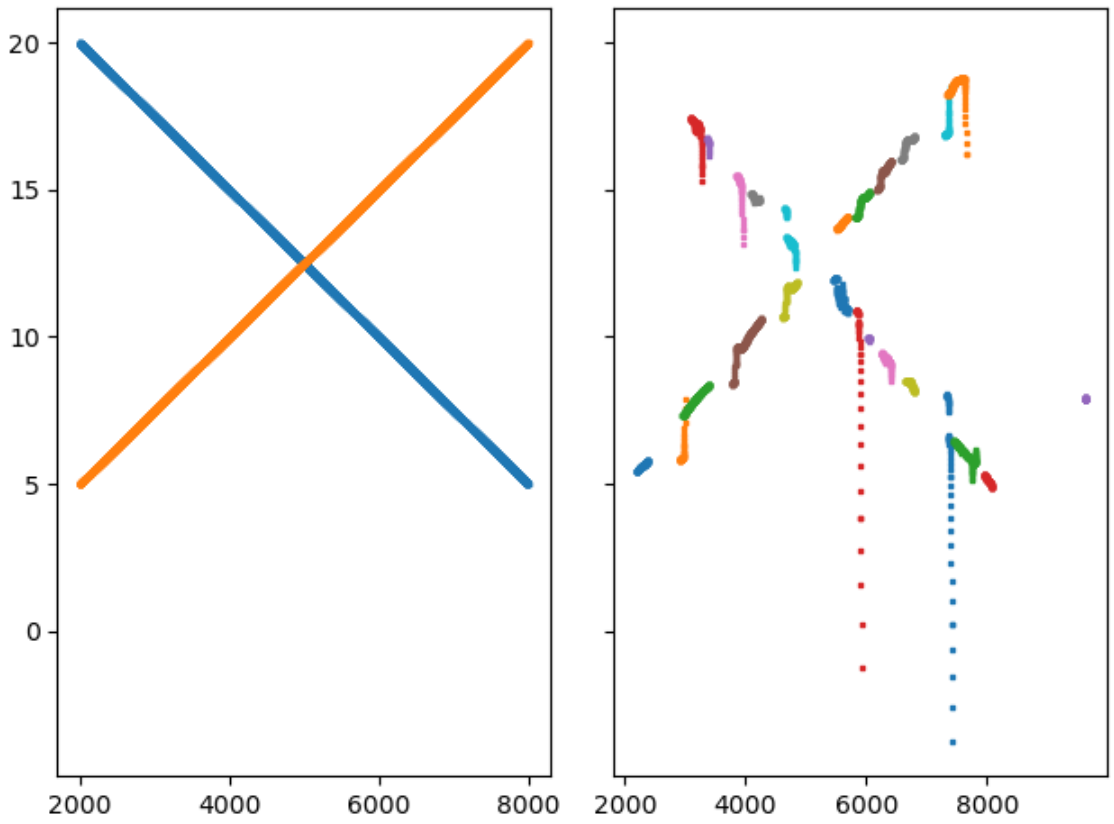

Figure 4-4: All tracks visualized of the two-target motion paths. On the left is the ground truth, with each ground truth track as a distinct color. On the right are the predicted tracks of the Vogl tracker. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

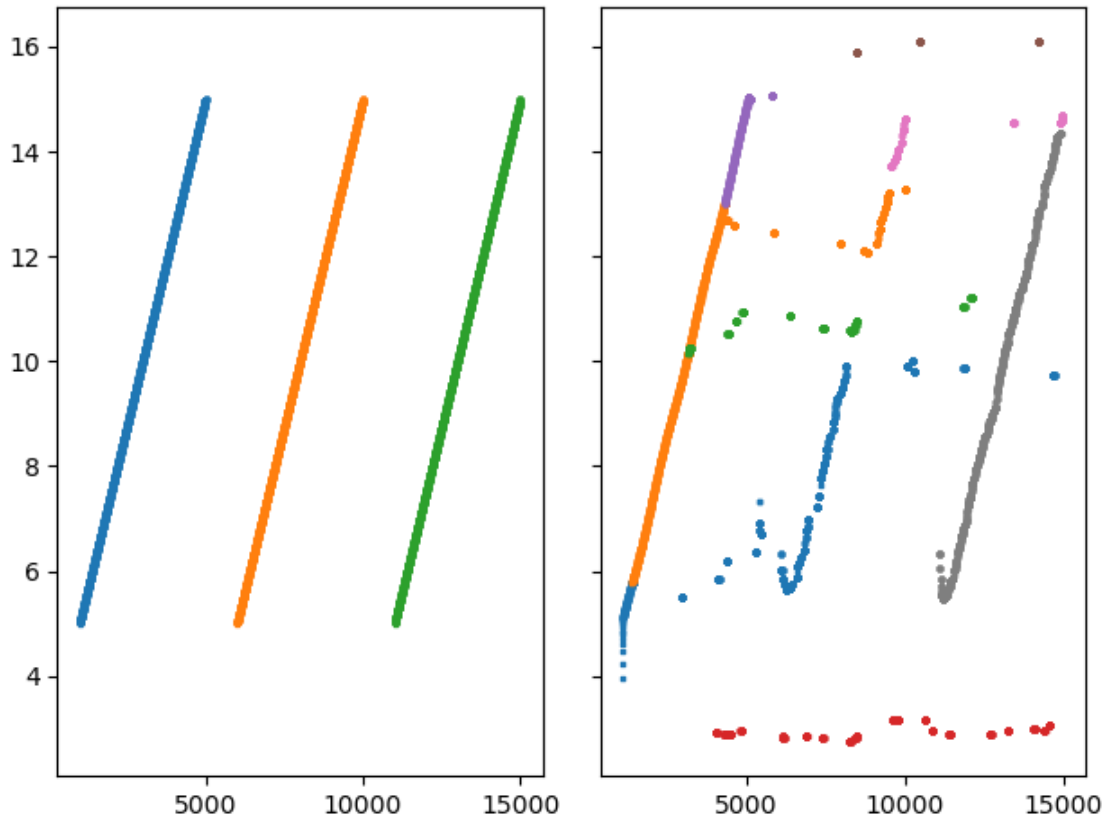

Figure 4-5: All tracks visualized of the three-target motion paths. On the left is the ground truth, with each ground truth track as a distinct color. On the right are the predicted tracks of RadarSORT. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

positions of both tracks are roughly equivalent, although both seemingly incorrect. The SORT tracker predicted tracks corresponding to a scenario where the targets turned around, instead of kept on going. It is unclear what "decision" was made by the Vogl tracker, as there are too many identity switches.

4.1.3 Experiment 3: Three Targets Walking in Sequence

In this experiment, three targets walk from 5m to 15m in separate and distinct time segments, with 5s of no movement in between each path.

The visualized results can be seen in Figures 4-5 and 4-6 for the SORT and Vogl trackers, respectively.

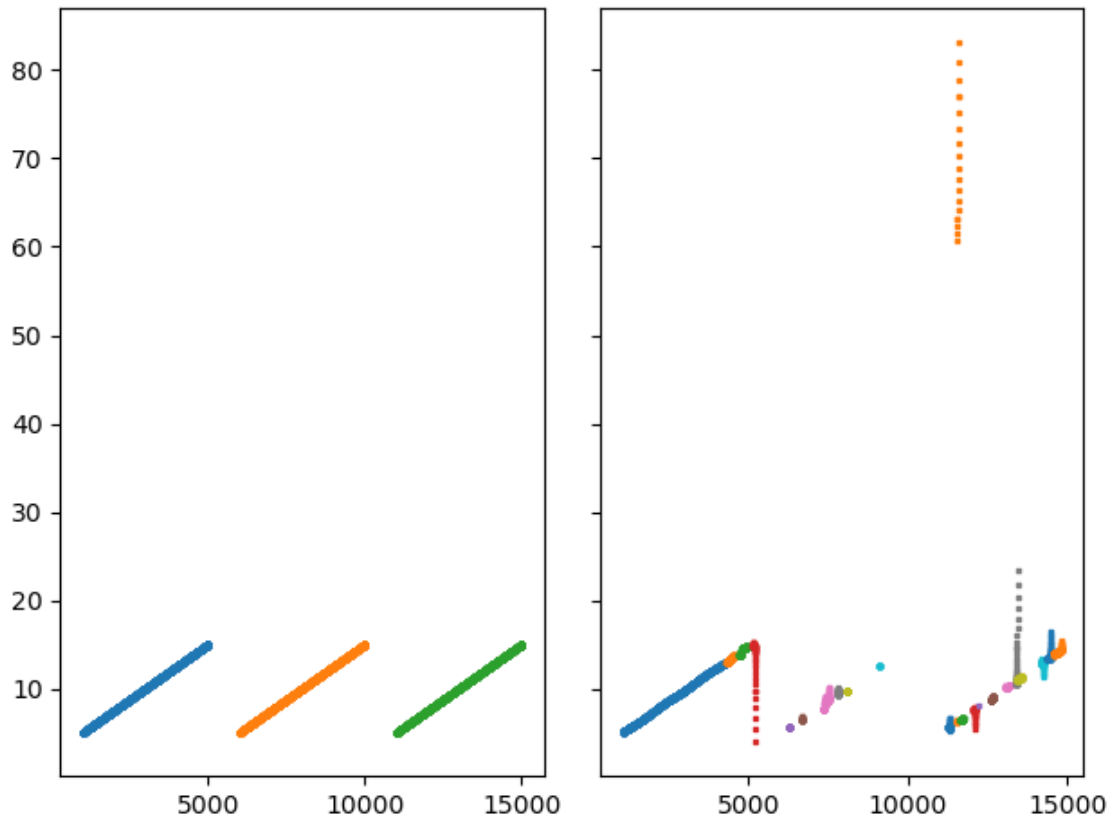

Figure 4-6: All tracks visualized of the three-target motion paths. On the left is the ground truth, with each ground truth track as a distinct color. On the right are the predicted tracks of the Vogl tracker. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

RadarSORT was able to form three distinct motion paths that are more-or-less whole in their tracks. RadarSORT also seems to have cross-associated much of the paths from both the 1st and 2nd ground-truth target, though the third is entirely distinct and whole. The Vogl tracker tracked the first target exceptionally well, and then performance fell off. It missed the middle target almost entirely, and the latter two targets were rife with identity switches. There was also an erroneous track very far away from any ground-truth track towards the end of the scene's duration.

4.2 Real Experiments

4.2.1 Experiment 4: Walking and Holding Breath

In this experiment, a single target walks roughly 4m away from the radar, pauses, holds their breath, and then comes back.

The visualized results can be seen in Figures 4-7 and 4-8 for the SORT and Vogl trackers, respectively.

RadarSORT is able to encompass the entirety of the ground-truth track in just one predicted track. It does not handle multipathing well, which is the phenomenon where a target appears to exist in intervals at distances farther than its true range due to the excess reflections of pulses on the scene. While it still includes them as tracks, it considers them to be separate tracks.

The Vogl tracker actually tracks the one target, and just the one target, rather tightly. It switches identity many times, however, and appears to fall off randomly.

4.2.2 Experiment 5: Standing Still

In this experiment, a single target stands roughly 4m away from the radar for the duration of the trial.

The visualized results can be seen in Figures 4-9 and 4-10 for the SORT and Vogl trackers, respectively.

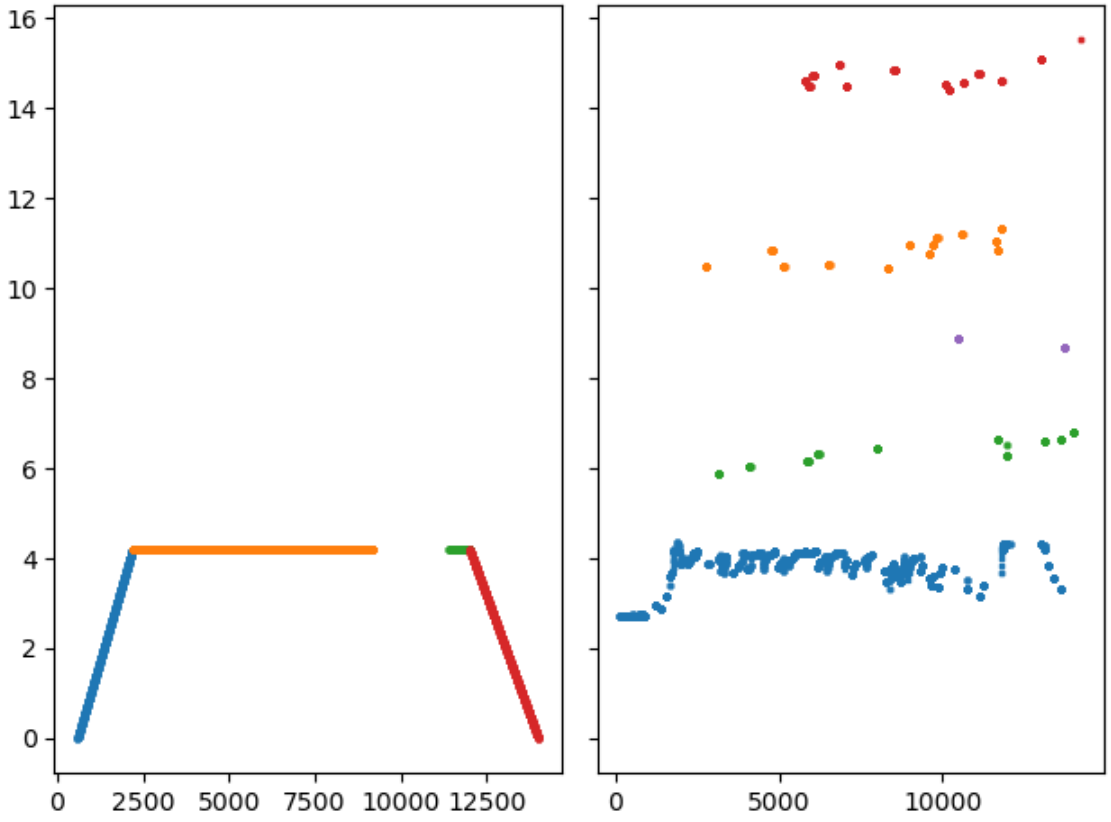

Figure 4-7: All tracks visualized of the one-target motion path. On the left is the ground truth, with each component of the single ground-truth track as a distinct color. On the right are the predicted tracks of the RadarSORT tracker. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

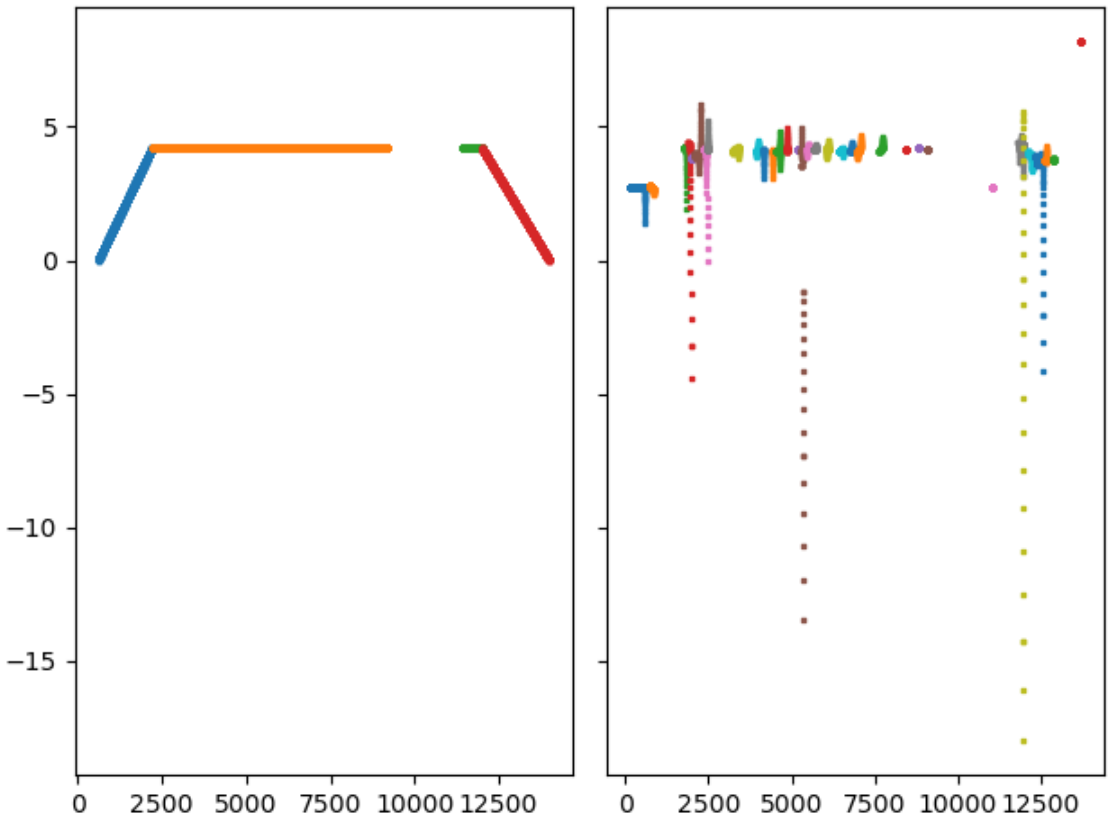

Figure 4-8: All tracks visualized of the one-target motion path. On the left is the ground truth, with each component of the single ground-truth track as a distinct color. On the right are the predicted tracks of the Vogl tracker. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

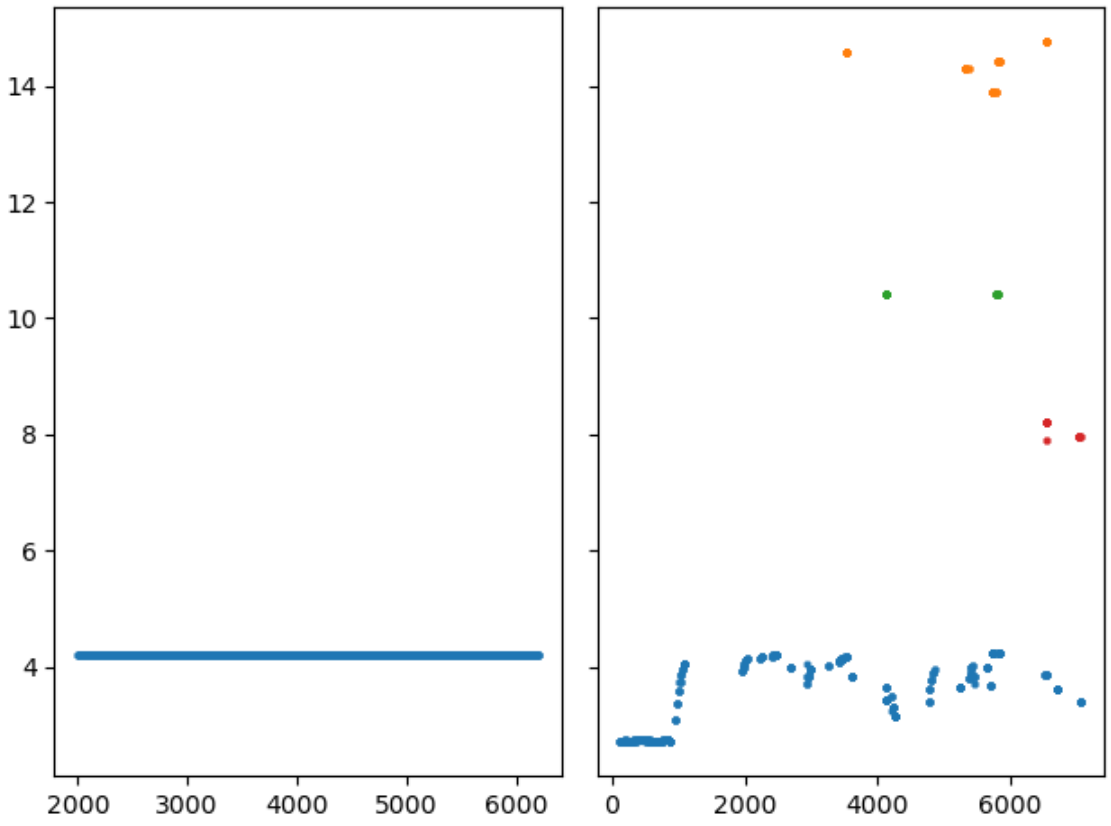

Figure 4-9: All tracks visualized of the single target's stationary path. On the left is the ground truth. On the right are the predicted tracks of the RadarSORT tracker. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

RadarSORT does a poor job of tightly tracking the stationary target, though it does approximate its position and associate a single track for it.

The Vogl tracker is unable to hold a solid track on the target for any stretch of the scene.

4.2.3 Experiment 6: Walking and Holding Breath II

In this Experiment, as in Section 4.2.1, a target walks away from the radar to roughly 4.5m and then remains stationary, holding their breath for a brief period. While stationary, the target gestures and crouches.

The visualized results can be seen in Figures 4-11 and 4-12 for the SORT and Vogl

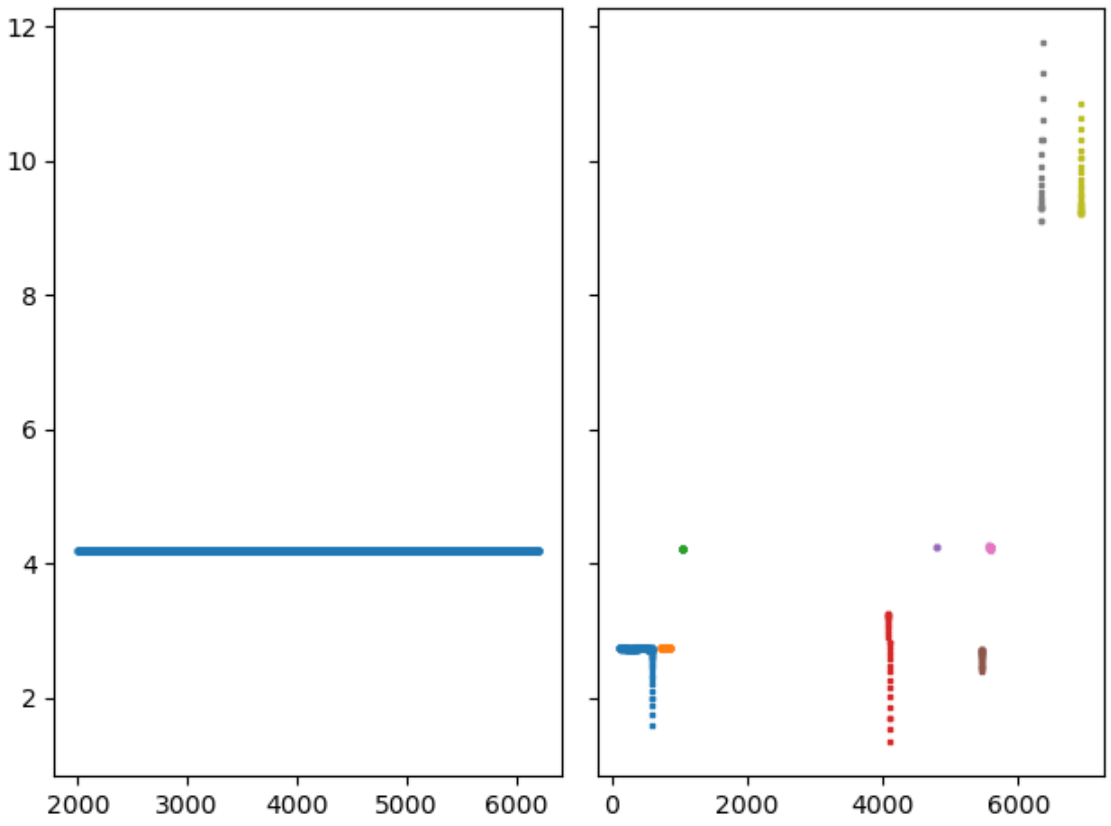

Figure 4-10: All tracks visualized of the single target's stationary path. On the left is the ground truth. On the right are the predicted tracks of the Vogl tracker. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

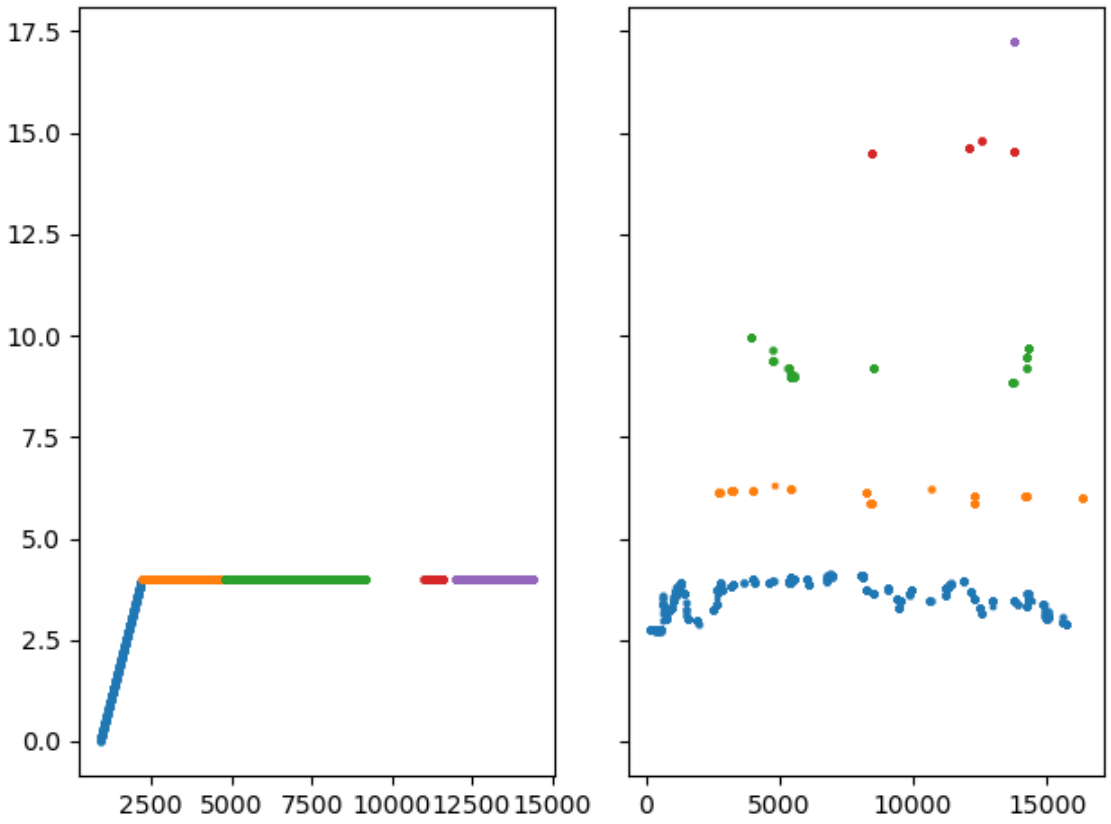

Figure 4-11: All tracks visualized of the one-target motion path. On the left is the ground truth, with each component of the single ground-truth track as a distinct color. On the right are the predicted tracks of the RadarSORT tracker. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

trackers, respectively.

RadarSORT is able to approximate the track somewhat well, suffering the same multipath issues as before but compartmentalizing tracks well.

The Vogl tracker struggled much more with holding a consistent track with the additional motion. As can be seen in the Figure, there are many gaps and identity switches throughout the duration of the scene.

4.2.4 Experiment 7: Sitting Still

In this last experiment, we have one target simply sitting at roughly 4.5m, breathing normally.

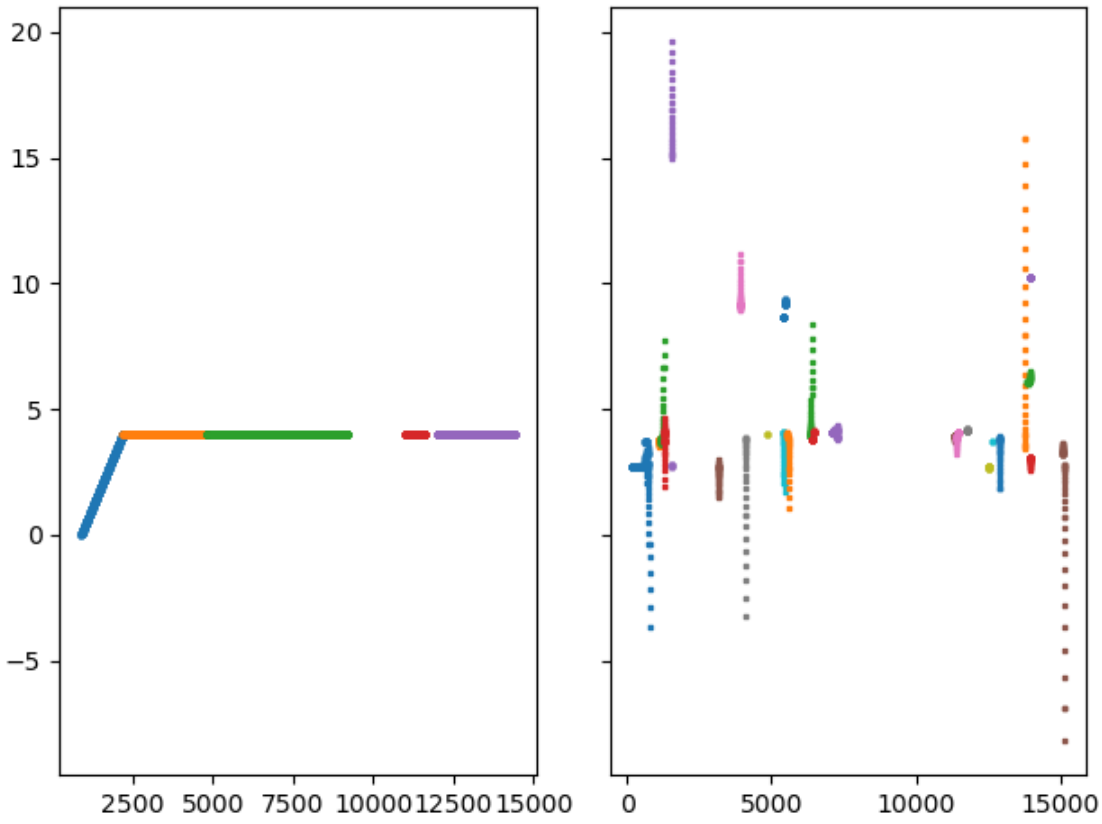

Figure 4-12: All tracks visualized of the one-target motion path. On the left is the ground truth, with each component of the single ground-truth track as a distinct color. On the right are the predicted tracks of the Vogl tracker. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

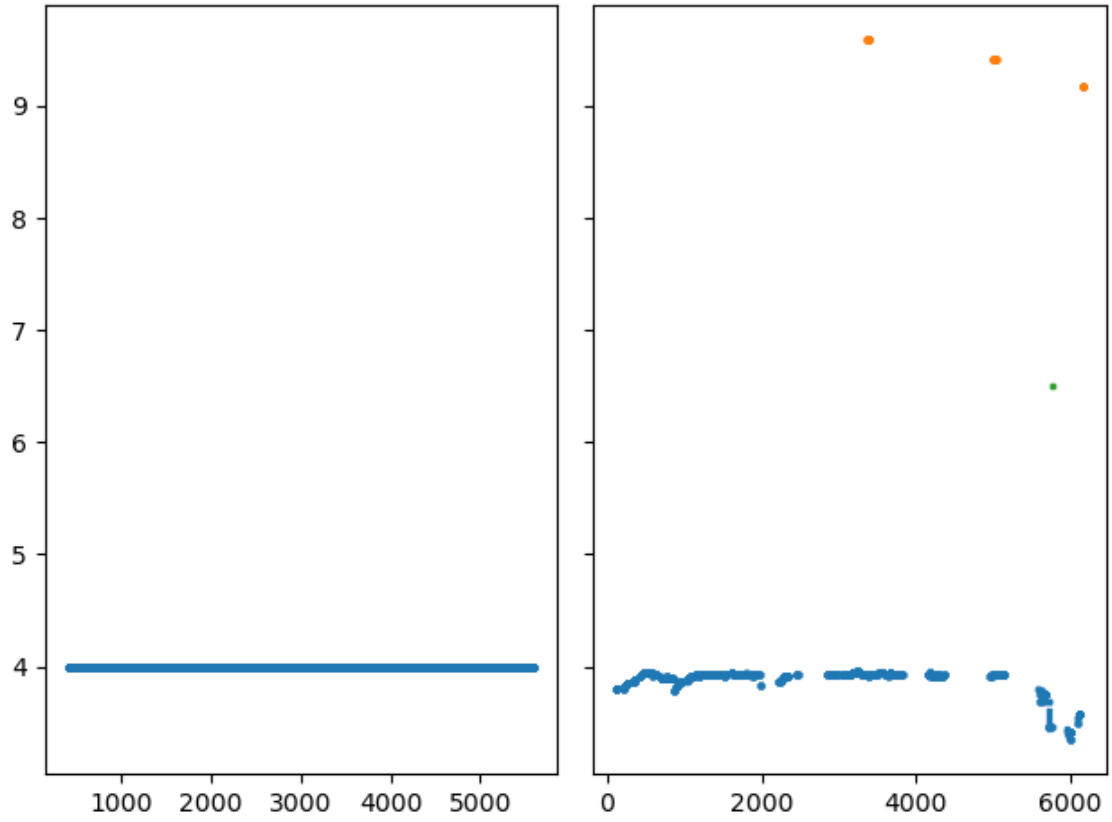

Figure 4-13: All tracks visualized of the single target's stationary path. On the left is the ground truth. On the right are the predicted tracks of the RadarSORT tracker. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

The visualized results can be seen in Figures 4-13 and 4-14 for the SORT and Vogl trackers, respectively.

RadarSORT is able to approximate the track somewhat well, suffering the same multipath issues as before but compartmentalizing tracks well.

As is consistent with the trends we have seen thus far, RadarSORT appears to track the stationary target better, with zero identity swaps and relatively few gaps.

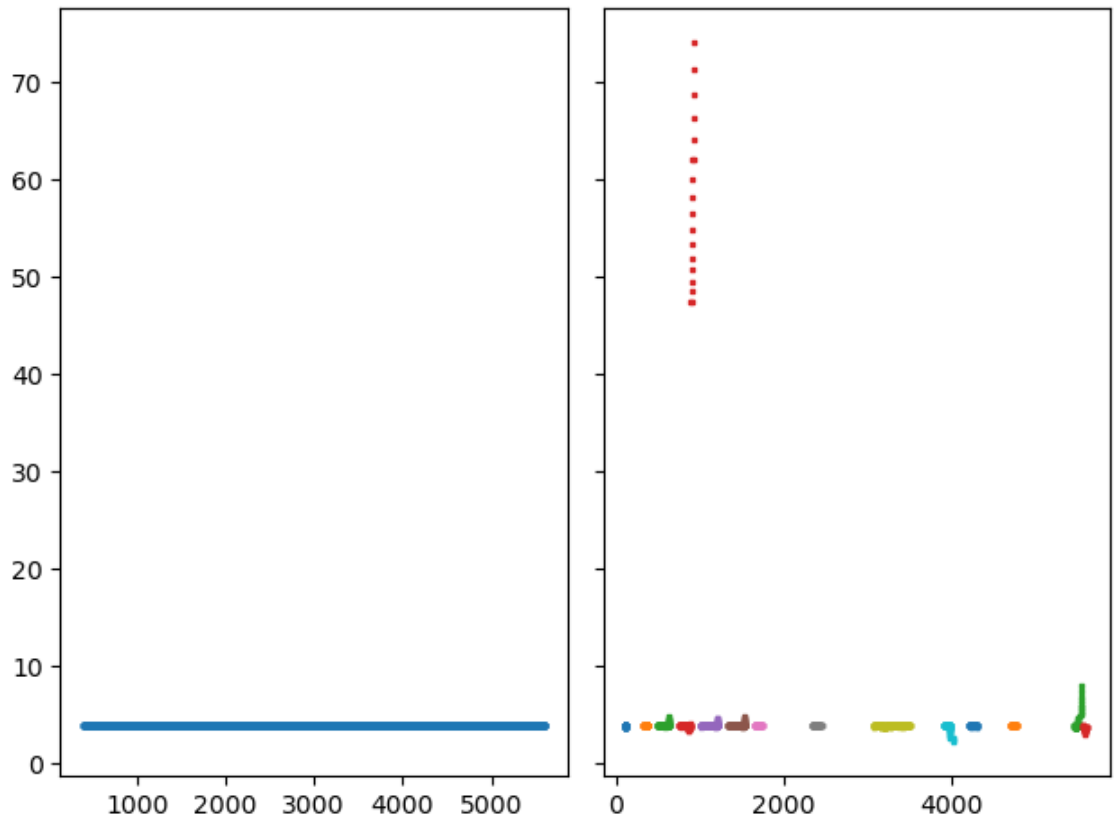

Figure 4-14: All tracks visualized of the single target's stationary path. On the left is the ground truth. On the right are the predicted tracks of the Vogl tracker. The x-axis represents the pulse number of a particular detection, and the y-axis represents range from the radar in meters.

Chapter 5

Conclusion

5.1 Limitations

As is shown in results, RadarSORT makes improvements on the existing detection system's tracker, though it does not solve them entirely. Some critical limitations identified thus far:

1. Sometimes-shoddy positional tracking. While RadarSORT is empirically able to approximate the position of a tracked target over the course of their track, the existing Vogl tracker was always able to produce tighter tracks, even if the tracks themselves were short-lived.

2. No identity-switching when there *should* be. While it is nice in some circumstances that our system is able to re-associate tracks it has not seen in a while over some kind of similarity metric, this is not always the correct decision.

3. Identity-switching when there *shouldn't* be. Certain ground-truth tracks are broken up into multiple predicted sub-tracks. As an opposite, but equally incorrect, mistake to the previous limitation, this results in our system predicting more individuals are in a scene than is actually true.

The impact of these limitations can be mitigated with the utilization and improvement of the meta-tracking procedure to double-check tracks with short delay, as

well as by improving the internal prediction system of our tracker. We lay out some suggestions for how to approach these improvements in Section 5.3

5.2 Ethics

As development continues on this endeavor to create a robust and versatile hand-held radar detection platform, it is vital that the very real experiences of its future users remain central to the design process throughout. [4] sets out a paradigm for approaching such problems with three key criteria: functionality, transparency, and potential. The first, functionality, demands that the system either works according to the designer's intent, or that when it does not, it fails in a manner that is acceptably ethical. The second, transparency, demands that *how* a system arrives at an answer is well-defined and clearly conveyed. The third, potential, demands that, even when assumed to be perfectly functioning (which is never), a system amplifies the efforts of the ethical over those of the unethical.

Without delving into the vast and nuanced philosophy driving such decisions, we will briefly address ways that future work can address these three criteria and reinforce a commitment to ethical development.

5.2.1 Functionality

Simply put, the system must work. The surest way to assess that the system *does* work is to quantitatively verify that it does across large-scale and rigorous testing, with concrete data and results behind it. Much of the work in this paper actively contributes to this endeavor, setting the stage for consistent evaluation of our system moving forward.

Further, if and when the system fails, the system must be deliberate in how it chooses to communicate uncertainty over detections or the lack thereof. As described in Section 3.2.1, precision and recall are the two overarching measures of a detection system's performance. It is the opinion of this author that recall is the far more important of the two in matters of uncertainty. In other words, it is better for our

system to predict there *is* someone, say, under the rubble following a natural disaster, than to *not* predict it over some low certainty value. In development, efforts should be made to raise both values as close to 100% as possible, but if there is ever a need to choose one over the other, recall should be optimized.

5.2.2 Transparency

Our system must convey information about its predictions in a manner that is non-distracting in stressful situations, but highly informative if the user chooses to see more detail. Information from every phase of the detection/tracking pipeline must be made available to the user on demand, and confidence scores should be clearly visible alongside detections.

5.2.3 Potential

At the end of the day, this system is an intelligence augmentation for its users. It makes no autonomous decisions, it simply ingests and processes information before providing its results to some user. As far as potential goes, it is incapable of harm in a vacuum, and so long as it strives to satisfy the previous two constraints to a reasonable level, it has a capacity for good far, far greater than any capacity for harm.

5.3 Future Work

5.3.1 VoglSORT

While the actual tracking of RadarSORT was generally more performant than the existing Vogl Tracker, we can extract the positional state prediction of the Vogl tracker for use with RadarSORT's pipeline. This should give the best of both worlds, providing tight and continuous predicted tracks.

5.3.2 Live-Clustering

Just as the proposed MetaTracking algorithm was a clustering problem, we can boil down tracking to essentially just that, as well. It would be interesting to see the labelling of new detections as a live-clustering problem, with something like the STREAM algorithm applied to decide track IDs.

5.3.3 Realtime Meta-Tracking

A performant (both accuracy- and speed-wise) meta-tracking algorithm would help to establish more accurate summaries of what exactly is happening in a scene in near-realtime. Such functionality, if verifiably more accurate, is undoubtedly useful.

Appendix A

Tables

In this appendix, we provide a comprehensive breakdown of various metrics across selected experiments.

Table A.1: Result Table

Experiment	Tracker	Precision	Recall	f1 Score	Avg. Continuity	Avg. Uptime
S3-256 Trial 1	Vogl	88.089%	50.341%	64.069%	17.078%	26.071%
S3-256 Trial 1	RadarSORT	70.662%	15.443%	25.347%	45.143%	45.143%
S3-256 Trial 2	Vogl	81.752%	5.089%	9.581%	1.65%	1.667%
S3-256 Trial 2	RadarSORT	72.104%	10.524%	18.367%	24.333%	24.333%
S3-256 Trial 3	Vogl	98.593%	70.271%	82.056%	24.879%	29.120%
S3-256 Trial 3	RadarSORT	97.481%	23.054%	37.289%	36.560%	36.560%
S3-256 Trial 4	Vogl	45.094%	14.334%	21.753%	6.281%	6.761%
S3-256 Trial 4	RadarSORT	54.72%	6.510%	11.634%	17.681%	17.681%
S3-256 Trial 8	Vogl	96.312%	78.628%	86.676%	28.224%	38.500%
S3-256 Trial 8	RadarSORT	97.185%	50.900%	66.809%	67.192%	67.192%

Bibliography

[1] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *CoRR*, abs/1602.00763, 2016.

[2] G. Garreau, C. M. Andreou, A. G. Andreou, J. Georgiou, S. Dura-Bernal, T. Wennekers, and S. Denham. Gait-based person and gender recognition using micro-doppler signatures. In *2011 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 444–447, 2011.

[3] G. Garreau, N. Nicolaou, C. Andreou, C. D'Urbal, G. Stuarts, and J. Georgiou. Computationally efficient classification of human transport mode using micro-doppler signatures. In *2011 45th Annual Conference on Information Sciences and Systems*, pages 1–4, 2011.

[4] Michael D Hiebert. The human element: The ethics of ai in defense. *ethics*, Jul 2020.

[5] Y. Kim and B. Toomajian. Hand gesture recognition using micro-doppler signatures with convolutional neural network. *IEEE Access*, 4:7125–7130, 2016.

[6] H. T. Le, S. L. Phung, and A. Bouzerdoum. Human gait recognition with micro-doppler radar and deep autoencoder. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3347–3352, 2018.

[7] Zhenwei Miao, Wei Ji, Yong Xu, and Jun Yang. A novel ultrasonic sensing based human face recognition. pages 1873 – 1876, 12 2008.

[8] Michael Otero. Application of a continuous wave radar for human gait recognition. *Proceedings of SPIE - The International Society for Optical Engineering*, 5809:538–548, 05 2005.

[9] John E. Peabody, G. Charvat, J. Goodwin, and M. Tobías. Through-wall imaging radar. 2012.

[10] B. Vandersmissen, N. Knudde, A. Jalalvand, I. Couckuyt, A. Bourdoux, W. De Neve, and T. Dhaene. Indoor person identification using a low-power fmcw radar. *IEEE Transactions on Geoscience and Remote Sensing*, 56(7):3941–3952, 2018.

[11] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. *CoRR*, abs/1703.07402, 2017.

[12] M. Zhao, T. Li, M. A. Alsheikh, Y. Tian, H. Zhao, A. Torralba, and D. Katabi. Through-wall human pose estimation using radio signals. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7356–7365, 2018.