Indoor False Detection Mitigation with Object Classifcation for People Detection, Tracking Applications

60GHz Indoor False Detection Mitigation Lab Overview

October 2018

Purpose

- This slideshow provides motivation and overview for the 60GHz indoor classification lab.
- After reviewing this slideshow the reader should have a basic understanding of the following:
  - The motivation and context for overlaying object classification on top of indoor people detection and tracking functionality in 60GHz radar
  - A high level understanding of the processing chain for people detection and how a classification algorithm has been integrated to this chain to help discriminate between humans and certain kinds of indoor moving clutter such as fans, blinds, and curtains.
  - Some of the design steps that would be common for approaching the application of object classification in the context of 60GHz FMCW radar.

TI mmWave in Building Automation

- Building Automation is an area with many sensing applications
- TI's mmWave sensors have advantageous properties for enabling and enhancing many building automation applications as illustrated on the next slide.
- In particular, TI mmWave sensors are well suited to many applications where it is important to accurately detect the presence of people and/or track their locations and movements due to the rich information in range, angle, and doppler.
- In many cases it is also important, when detecting and tracking people, to also discriminate between humans moving through a scene, and other movement generated by non-human ground clutter. For instance, reducing false alarms in a security application.
- Again, TI mmWave sensors provide rich information in range, angle, and doppler which can be processed to distinguish between humans and non-human generated movements.
- The following slide illustrates some building automation applications and outlines some of the advantages and challenges to the use of mmWave sensors for those applications.

Sector

EE

TI mmWave in Building Automation

Building Automation
Building Security Systems
Motion Detectors

Building Automation
Building Security Systems
People Counting

Building Automation
Building Security Systems
Automated Doors & Gates

Building Automation
Video Surveillance
IP Network Camera

GOAL: Robust, small form-factor detection and sensing of people near buildings, cameras, and doors

Advantages

- Robust to false detection/movements with integrated processing
- Radar information can give position and velocity – easy background subtraction, movement classification
- Robust to environment – lighting, temperature, moisture
- No camera or lens for privacy-conscience applications
- Rich data set can be used for classification/object identification

Challenges

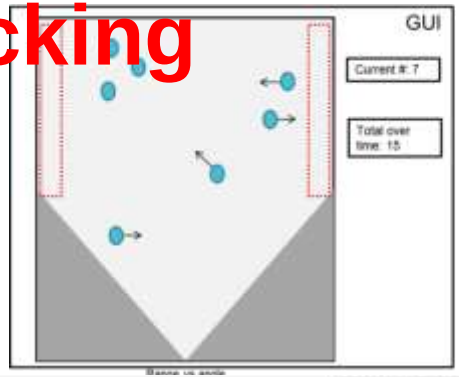
- **Angular resolution** of radar is lower, complex scenes can be challenging
- **Power consumption** for wireless, battery-powered sensors

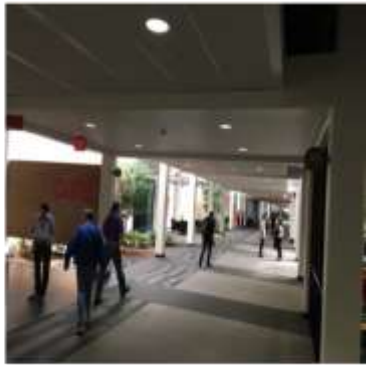
People Detection and Tracking

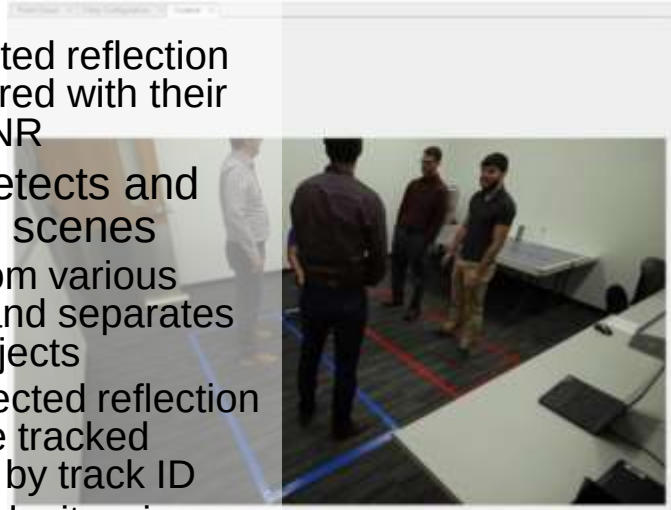
- In previously released TI Labs and TI Designs, a 77GHz mmWave sensor has been used to detect, localize, track, and count people in an indoor environment demonstrating the practical effectiveness of mmWave sensing in building automation.
- Leveraging the scalability and compatibility of TI's mmWave product portfolio, this capability has been ported to the 60GHz band for unlicensed use around the world.
- This capability is the baseline for demonstrating object classification and discriminating between humans and non-human moving ground clutter in indoor environments.
- The following slide illustrates the functionality of people detection, tracking and counting in indoor environments.

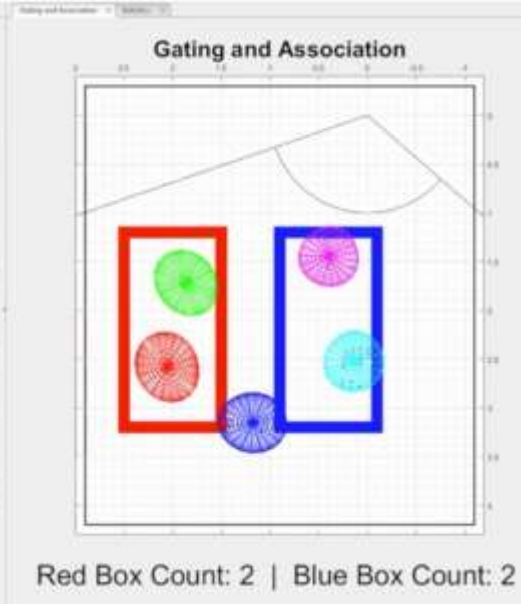
People Detection and Tracking

- 60GHz People Detection System
  - A Range/Angle Detection Algorithm with Doppler Extraction
    - The chirp design is tailored to indoor ranges, and removal of zero doppler objects with rich point cloud and sensitivity to slow movement.
    - Frame by frame, all the detected reflection points in the scene are delivered with their range, angle, doppler, and SNR
  - A group tracking algorithm detects and separates people in complex scenes
    - Takes the point reflections from various moving objects in the room, and separates them into different tracked objects
    - From frame to frame, the detected reflection points are associated with the tracked objects, each point is labeled by track ID
    - Tracking provides location, velocity, size estimates and predictions for each object.







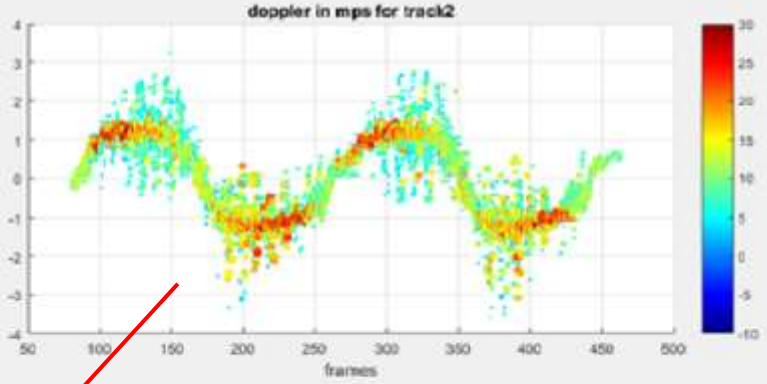


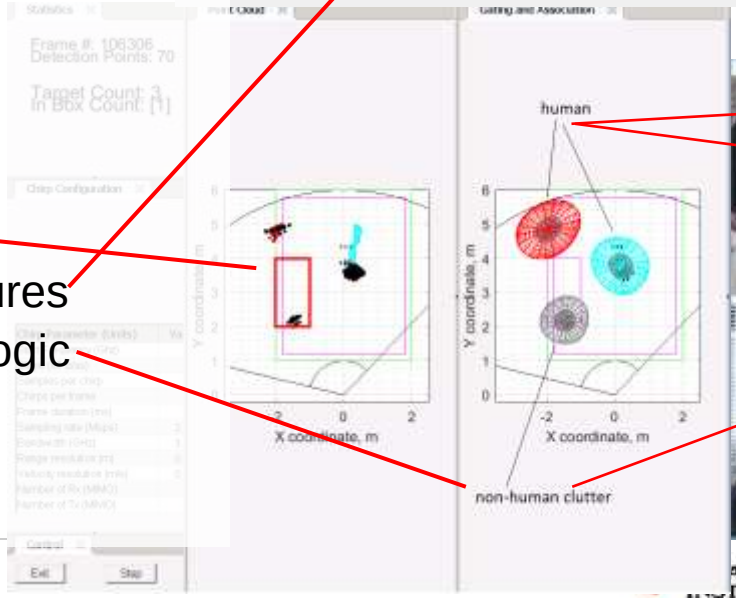
People Detection and Tracking with Classification

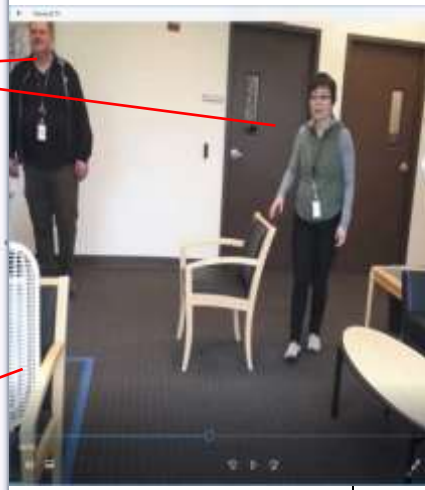
- The baseline people detection and tracking processing chain provides a frame by frame point cloud stream that includes:
  - Removal of the purely static reflections with zero doppler
  - All points labelled with a track ID of the object to which they are associated, or an indicator that they are associated with no object.
  - The detection attributes of each point such as the range, doppler, angle, and SNR.
  - The list of active tracks and each track's attributes such as the location, and velocity of the object and other learned attributes.
- In general this chain will attempt to detect and track, and therefore report any moving object in the scene.
- Since the detection layer is sensitive to movement, and the tracker is tuned to detect clusters of reported reflection points as objects and attempt to track them as a human we have the following challenges:
  - Can we use the information provided to separately decide for each tracked object, if it is a human or some other moving ground clutter such as a fan, or a wind blown curtain or blind?
- This provides many essential elements for the development of an object classification system. In particular:
  - The frame by frame detected points stream could potentially provide statistical information useful for distinguishing between humans and moving ground clutter.
  - The tracker function of associating points from frame to frame with separate tracks provides both separation of different objects in the scene
  - It also provides a persistent time sequence over many frames so that statistical measures longer than one frame can be taken.
  - Furthermore, it provides a history of track attributes, such as position and velocity history.
- Classification of objects as human vs. moving ground clutter can be attempted by designing an algorithm that takes as input, the point cloud stream and tracker outputs, and subsequently extracts statistical features for each tracked object to make a decision if it is a human or a non-human moving clutter.
- The next three slides illustrate this approach to the classification problem as a post processing on the outputs of the detection layer and group tracker.

Indoor People Tracking-Counting:
False Alarm Mitigation, Behavior Classification

- Detect, Track, Count People in Complex Indoor Scenes
  - Eliminate Motion Based False Alarms
- Key Technical Challenges
  - Discriminate between people and moving fans, curtains, blinds.
  - People vs. small animals
- A Classification Problem
  - Detect and Track Objects
  - Extract Key Radar Signal Features
  - Apply Classification Decision Logic
- Roadmap Items
  - Classify People Behaviors

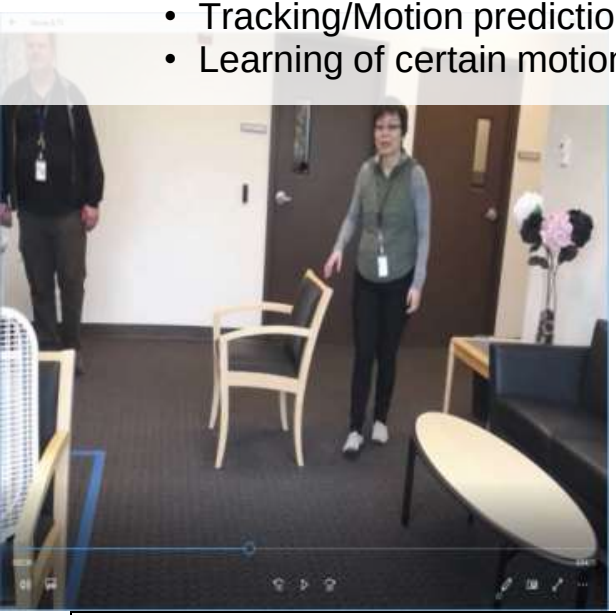


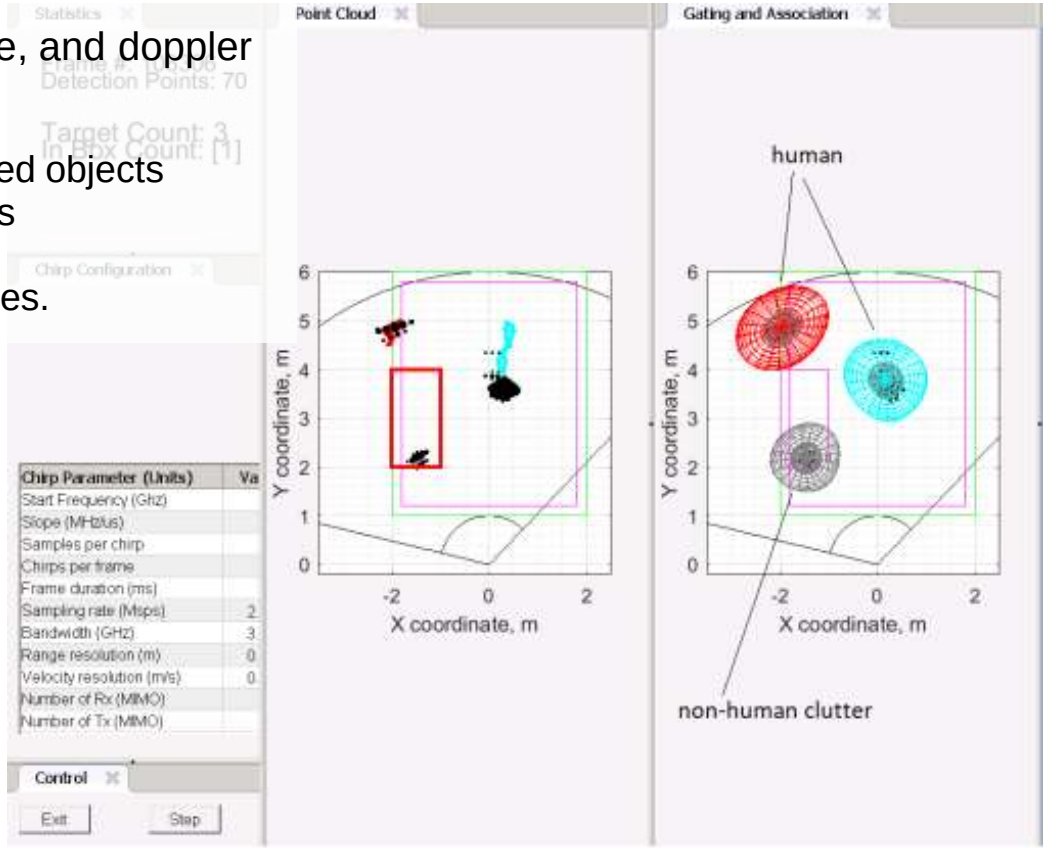




Indoor People Tracking-Counting With False Alarm Mitigation:
False Alarm Mitigation Classification

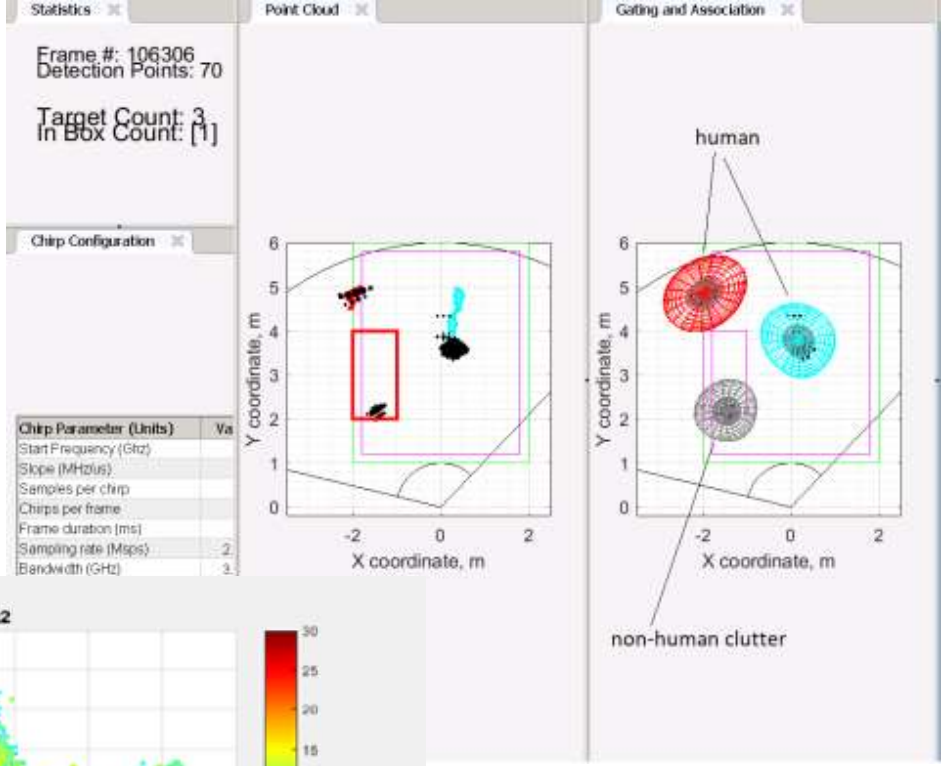
- Zoom in on Point Cloud generation:
  - Detection points with range, azimuth angle, and doppler
- Group Tracker: Frame by Frame:
  - Association of the points to existing tracked objects
  - Decision logic for detection of new objects
  - Tracking/Motion prediction on the objects
  - Learning of certain motion related attributes.

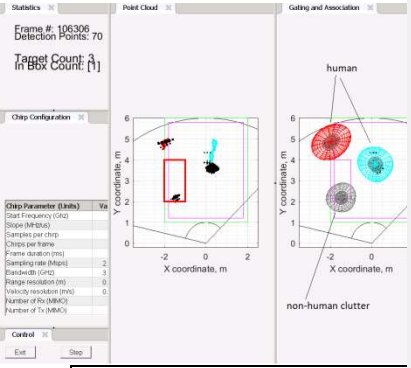


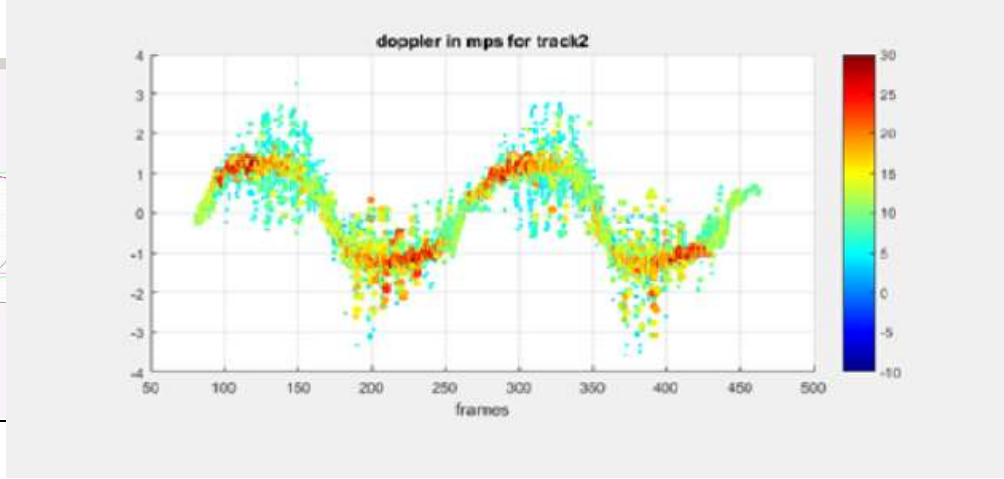


Indoor People Tracking-Counting:
False Alarm Mitigation Classification

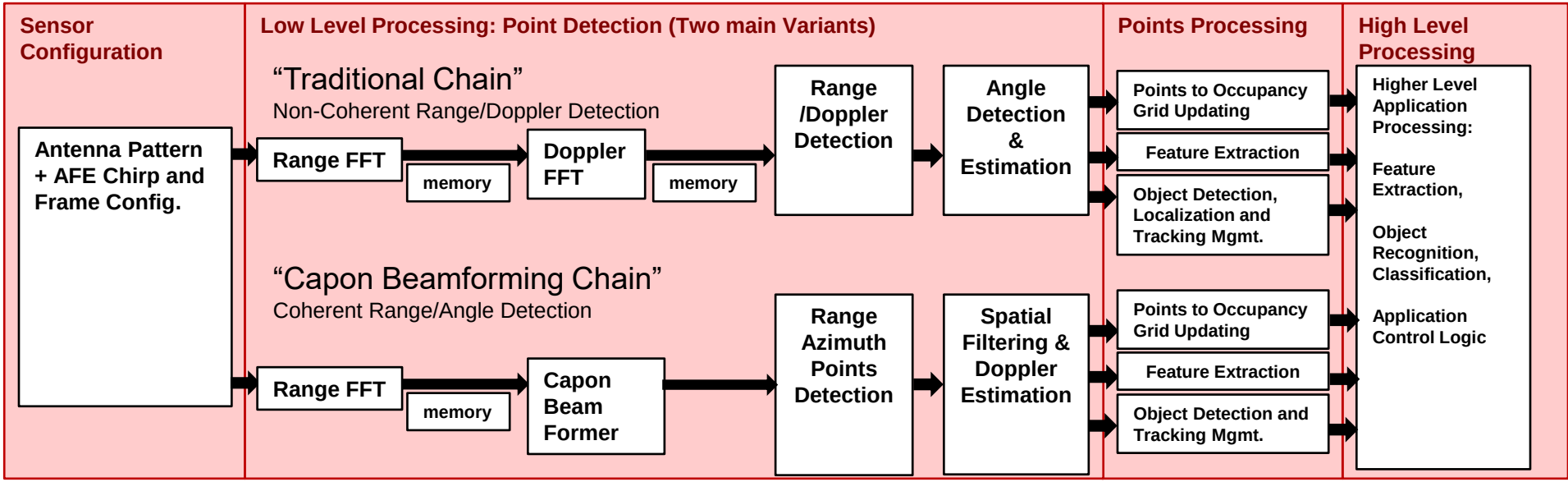
- Tracker Output, Feature Extraction:
  - A Multi-Frame Window of Localized Point Cloud, and Track Attributes is used for feature extraction
- Classification Decisions
  - made based on feature observation, and some object management state logic.





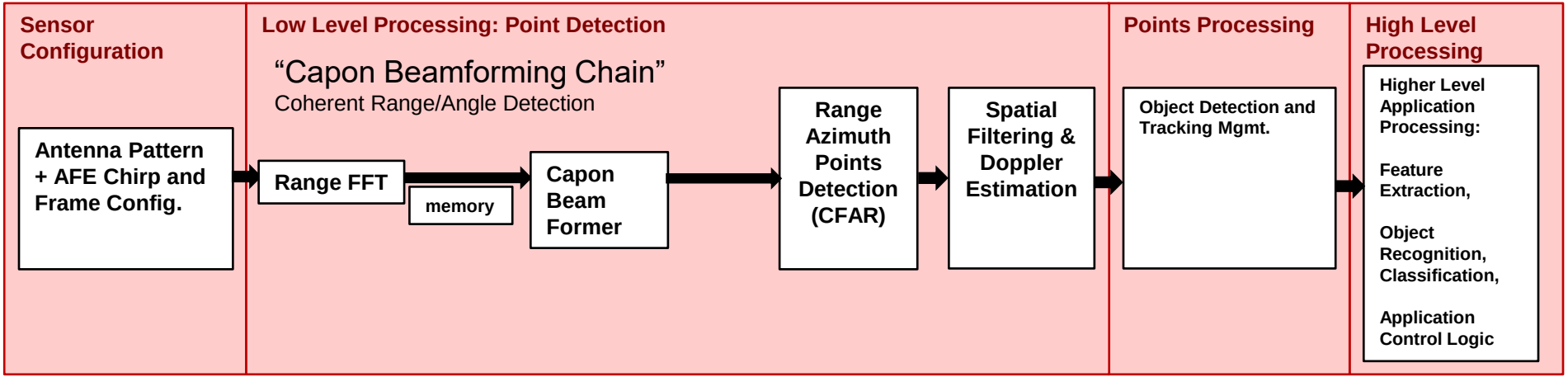


General View of Processing Chain Elements



- We have Two main Low Level Point Detection chain variants: Traditional Chain, and the Capon Beamformer.
  - We have used "Traditional Chain" mostly in Outdoor environments and where object speeds are generally higher.
  - We have found the "Capon Beamformer" well suited to indoor environments and where object speeds are lower. This is the chain used for this lab.
- Points Processing can be Occupancy Grid and/or Object Detection, Localization, and Tracking
- Higher Feature Extraction can happen at any level potentially.
  - Features are any statistics that could be useful for object recognition, classification.
- Higher level processing would include Object Recognition, Classification, and Application control

Processing Chain Elements for Indoor People Detection, Tracking, Classification

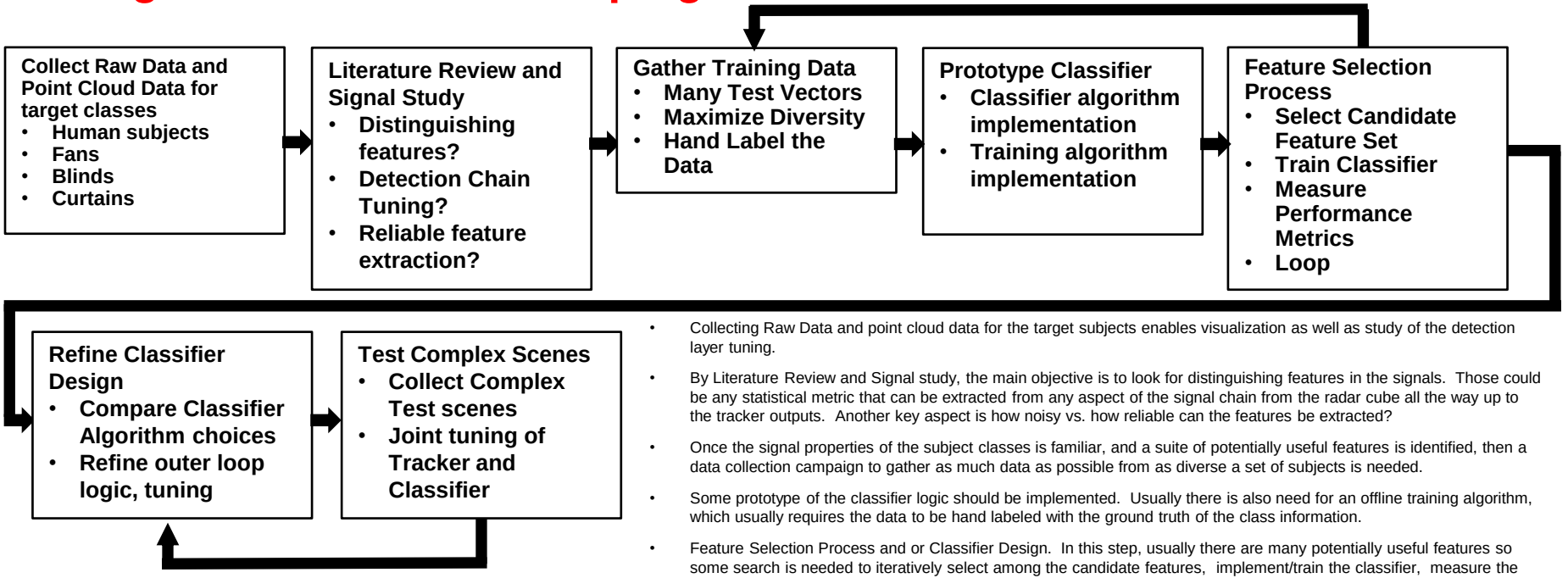


- The chain described above is a high level view of the functional split of the software specifically for this lab.
- The chirp configuration parameters dictate the analog front end configuration, the frame configuration, and the Range FFT size.
- The capon beamformer works on the per frame antenna data to estimate the virtual antenna covariance matrices for each range bin. This behavior is dictated by the antenna configuration, and the chirp and frame configurations.
- The range azimuth heat map is then formed and a range-azimuth peak detection algorithm is applied. It is here that parameters specified by `cfarCfg` and `doaCfg` affect the range-azimuth detection behavior of the chain.
- After peaks are detected in the range and azimuth heat map, then, for each detected point, further processing is done to extract the doppler spectrum, which is then searched for doppler spectrum peaks, further refining the list of detected points. `doaCfg` also contains parameters that configure this behavior.
- After the doppler search/estimation is done, then, for the given radar frame, there is a list of detected points. Each point has a range, angle of arrival, and a doppler value. This is the so called "point cloud", which is used by the group tracker for object detection and tracking. The configuration of the group tracker is specified in other documents, but the behavior of the initial object detection, and the subseqent tracking, and deallocation of the tracks, can be controlled by those parameters specified by the `trackingCfg` command.
- For each allocated track, the object classification then relies on a feature extraction operation, and then processing of the features to decide on the object class. The behavior of the feature extraction and classification logic can be configured by the parameters specified in the `classifierCfg` command

Designing the Classifier

- The next section will give an overview of the design steps involved in the development of the classifier

Design Process for Developing the Classifier



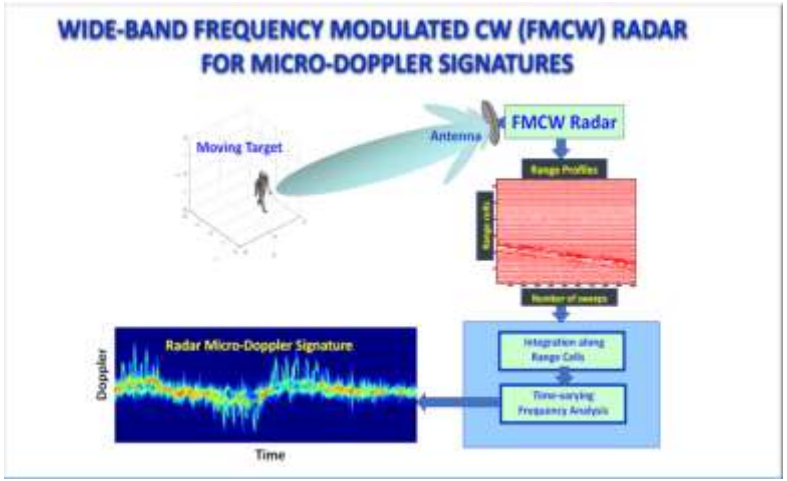
- Collecting Raw Data and point cloud data for the target subjects enables visualization as well as study of the detection layer tuning.
- By Literature Review and Signal study, the main objective is to look for distinguishing features in the signals. Those could be any statistical metric that can be extracted from any aspect of the signal chain from the radar cube all the way up to the tracker outputs. Another key aspect is how noisy vs. how reliable can the features be extracted?
- Once the signal properties of the subject classes is familiar, and a suite of potentially useful features is identified, then a data collection campaign to gather as much data as possible from as diverse a set of subjects is needed.
- Some prototype of the classifier logic should be implemented. Usually there is also need for an offline training algorithm, which usually requires the data to be hand labeled with the ground truth of the class information.
- Feature Selection Process and or Classifier Design. In this step, usually there are many potentially useful features so some search is needed to iteratively select among the candidate features, implement/train the classifier, measure the performance, usually on simple test vectors, and then repeat. This continues until a set of features is identified for which good performance on simple test scenes is good.
- Refining the classifier involves potentially comparing differnent classifier algorithms for the selected features. Also there can be some joint design refinements of the classifier interaction with the detection settings, and the tracker settings. The need for some outer loop logic for smoothing, filtering, or other special case handling may become apparent.
- Complex scene testing is the most challenging aspect as now the system should be subjected to scenes with multiple targets of different class types in the same scene. Again joint tuning of the algorithms at different layers may be needed. It may be the case that deficiencies in the detection layer or the tracker layer are the root cause of classification errors so joint tuning may be necessary. It may be necessary to collect many complext scenes, or to revisit and refine the classifier design. It may be necessary to revisit the training set size and the quality of the algorithm training as well.

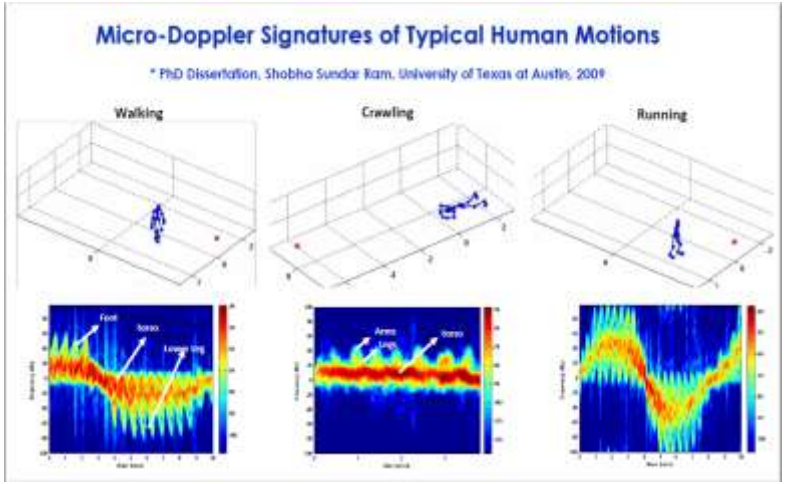
Indoor Classification Initial Design Goals

- Demonstrate classification of Humans vs. moving ground clutter
  - Humans vs. Fans, Blinds, Curtains
- Attempt to use Point Cloud based feature extraction
  - Feature extraction of lower level signals (radar cube data) is more memory and computationally intense
  - Can we extract similar features from the attributes of the point cloud.
- Using classic “hand crafted features” plus classical classification
  - Do not directly address “deep learning” or auto-learning of features.
- Algorithm structure will center on an inner algorithm that assesses short term feature characteristics, and continuously reports a decision, with an outer loop logic that handles longer term filtering of the outputs and special case handling.
- Complete the entire practical system including the point cloud, object detection and localization, feature extraction, and classification on target.

Literature Search and Raw Signal Study:
Micro-Doppler Classification is often sited as promising

- Typical research lacks all the essential elements needed to achieve a full, reliable system for object classification.
  - The features are limited ONLY to micro-doppler, whereas current FMCW has Doppler, Range, AND Angle.
  - Most of these systems do not address proper localization, cropping of the signal, assume one cooperating subject.
  - Doppler spectrograph based processing is memory and mips intensive.
- Example Below Extracted from Tutorial from Dr. Victor Chen and Prof. Gang Li (RadarCon 17)
  - Mainly looking at Doppler signatures to distinguish targets
    - Human gait/gesture analysis
    - Rotor blade (for helicopter or drone

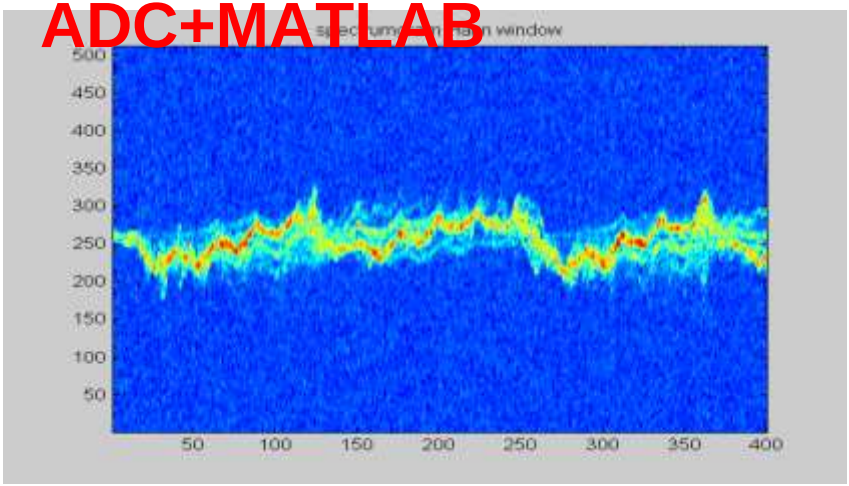


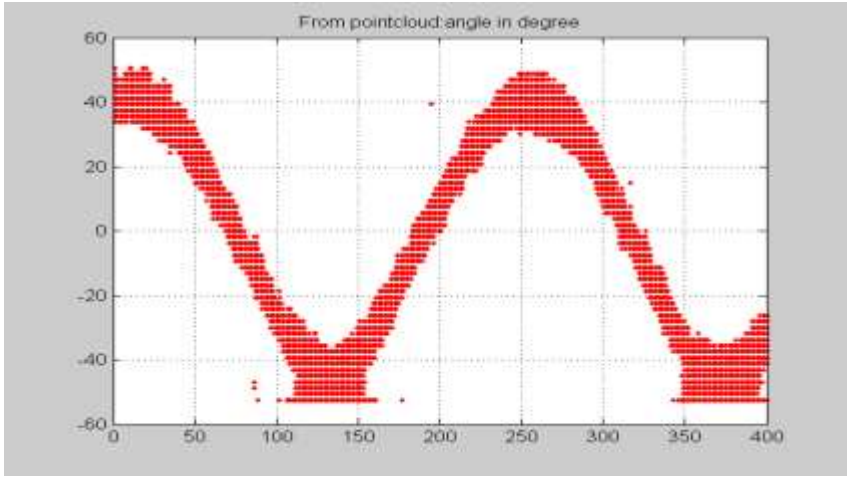


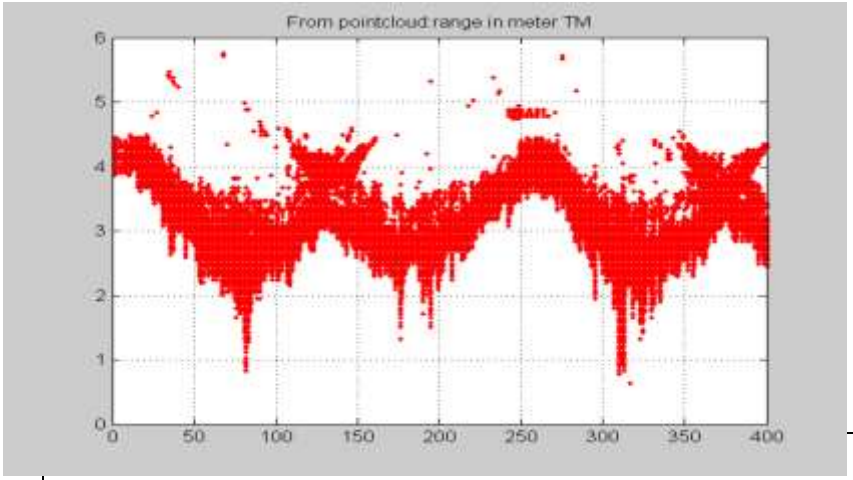
Signal/Feature Study

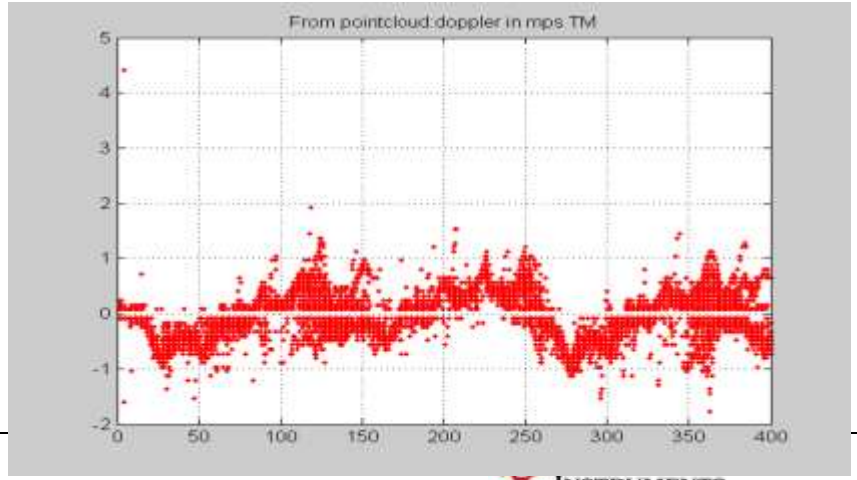
- The next few slides, as well as several in the backup, illustrate the characteristics of the raw signal doppler spectrograph as well as the point cloud based doppler spectrum information collected from several subjects as part of the initial signal study.
- Visual inspection alludes to the possibility of distinguishing features.
- The open question was exactly what statistics to extract that would be reliable for separating the classes?

People (Scene1, ~tangentially walking): single target, ADC+MATLAB

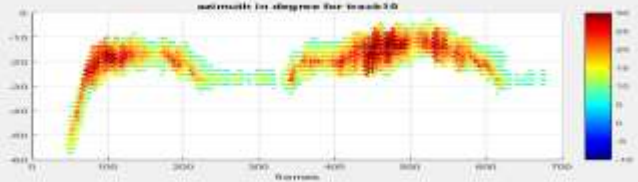


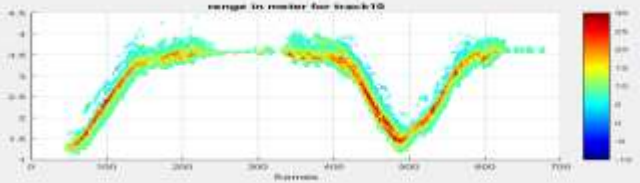


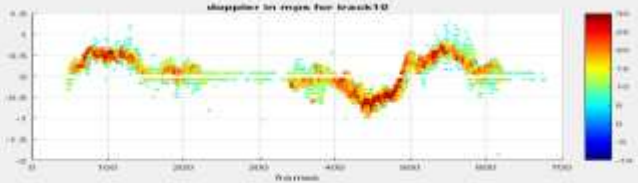


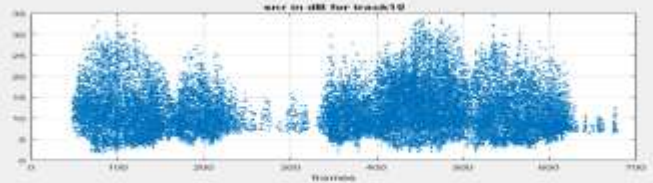


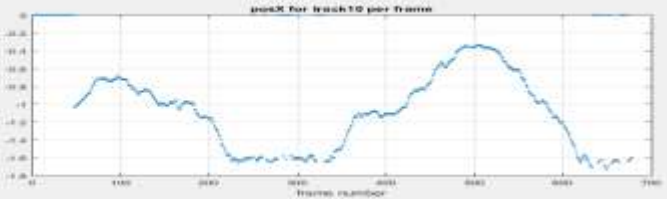
Single person slowly walking and sit down sideway

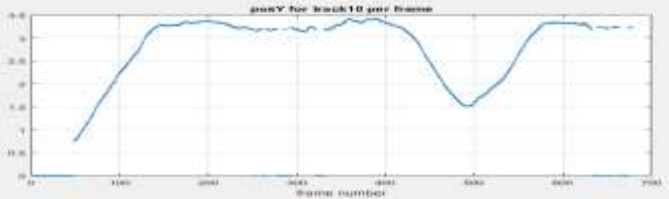


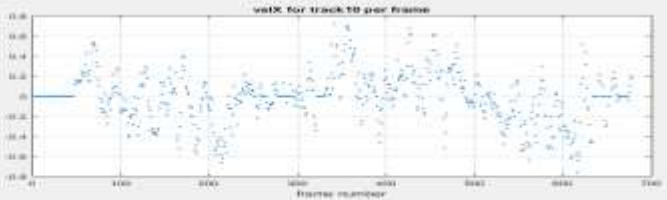


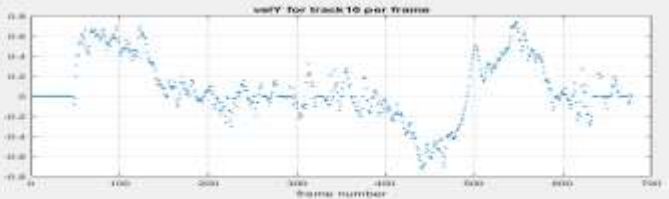




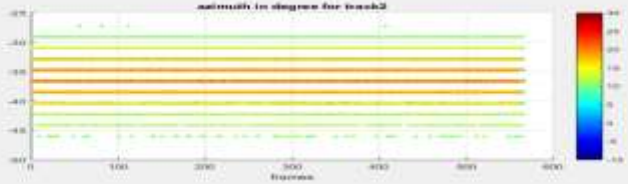


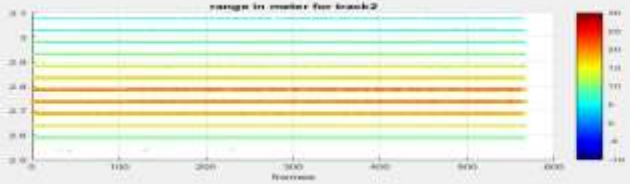


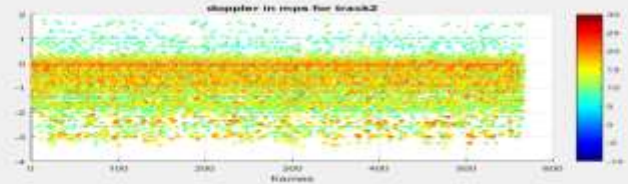


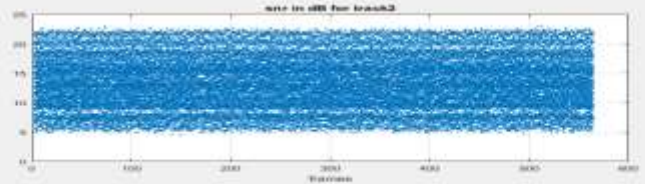


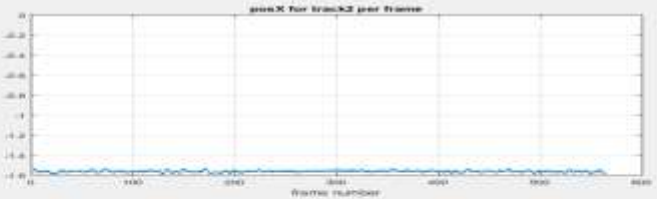
Single metal table fan fixed facing sensor, speed 1

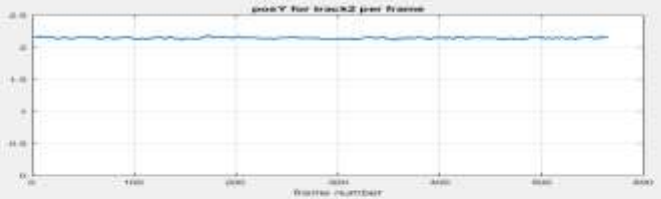


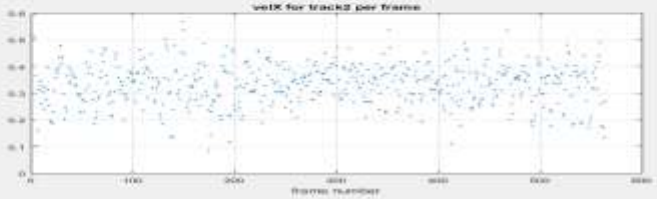


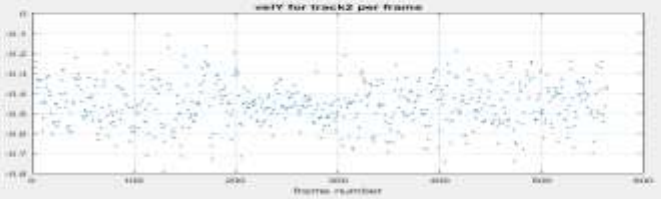




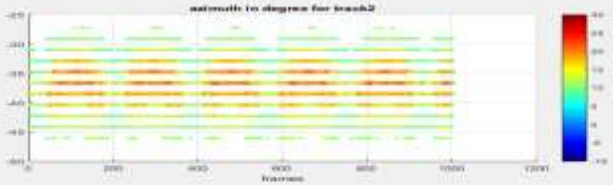


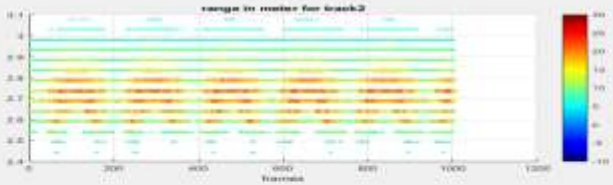


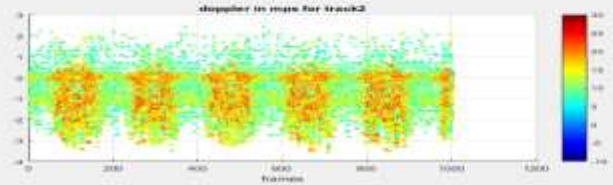


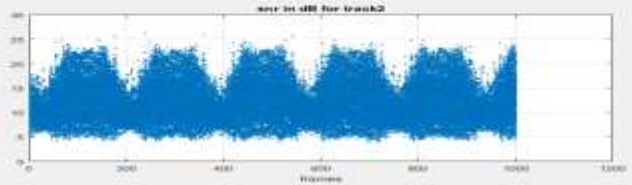


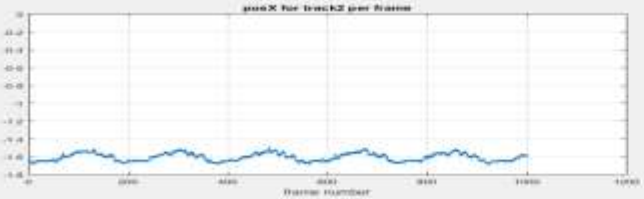
Single metal table fan oscillating, speed 3

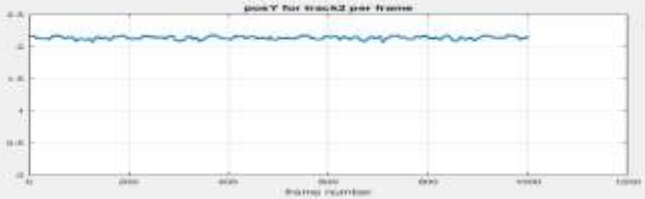


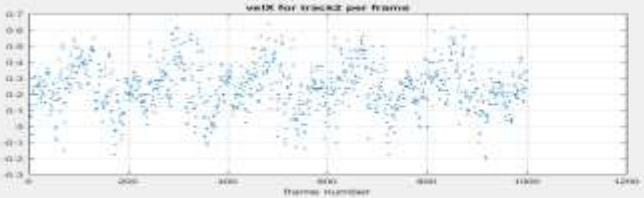


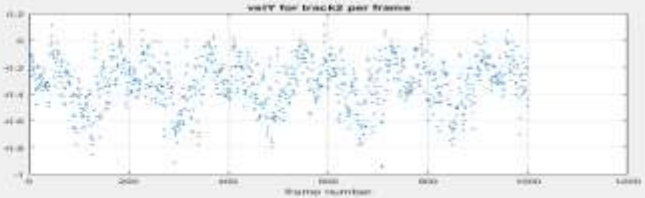












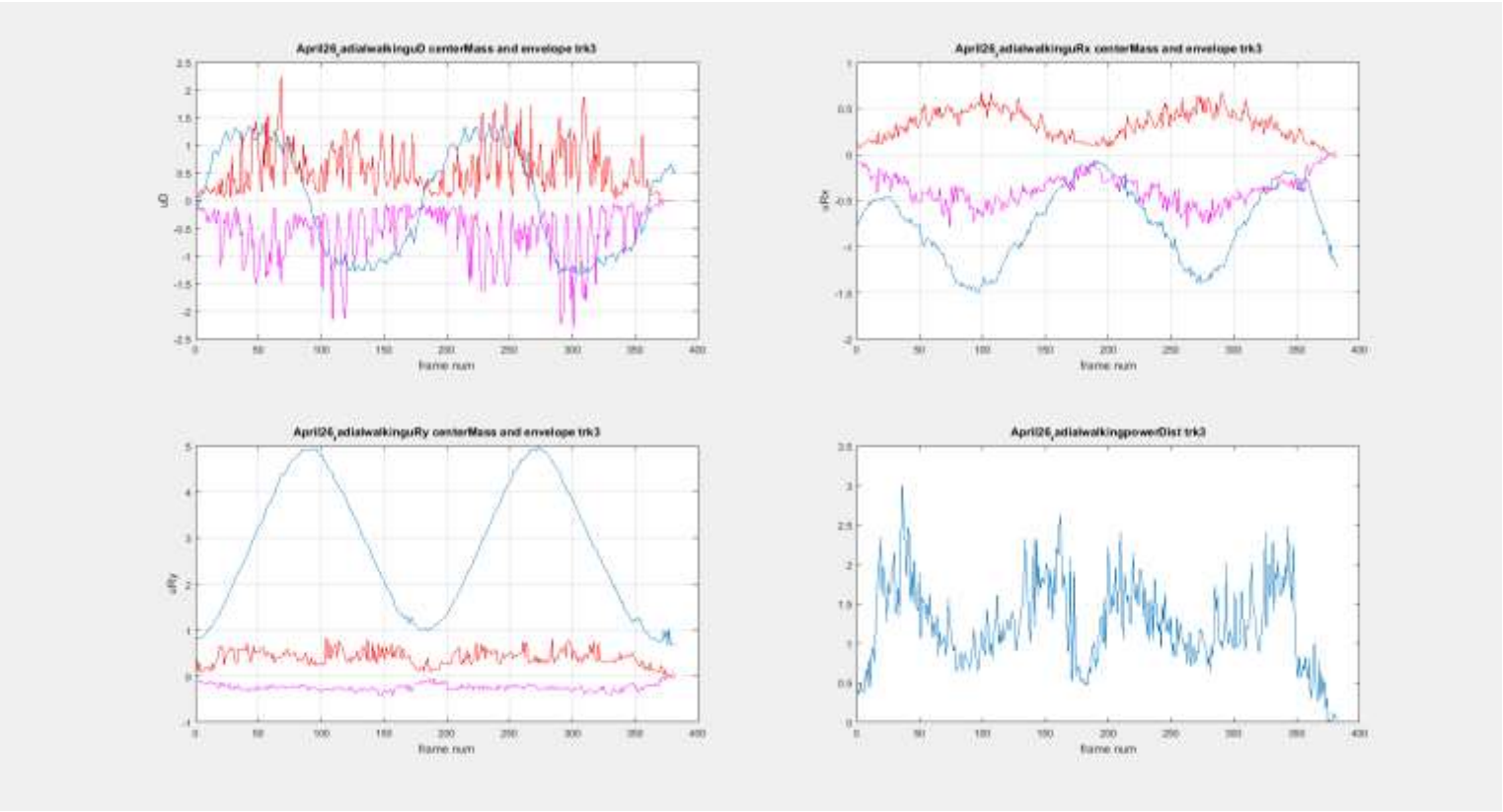
Extracted Features from the signal

6 uD Features from often cited: Kim-Ling features
Modified to be extracted from point cloud
Extended similar ideas to range and angle as well

- (1) Torso Doppler frequency
  - I implemented as center of mass for the cloud doppler per frame
- (5) normalized standard deviation of the doppler signal strength
  - I implemented as the std(linear SNR)/mean(linear SNR)
- The other 4 features need envelop estimation first
  - TI implementation: High-frequency envelope is the max(doppler - torsoFreq) per frame
  - TI implementation: Low-frequency envelope is the min(doppler - torsoFreq) per frame
  - All 4 features have dependencies on processing block size (num frames to look at)
  - (2) Doppler BW is: max(High-frequency envelope) – min (Low-frequency envelope)
  - (3) Doppler offset is: mean(High-frequency envelope) – mean (Low-frequency envelope)
  - (4) Torso BW is: minx(High-frequency envelope) – max (Low-frequency envelope)
  - (6) Limbo motion period: I tried to do FFT on the 2 envelopes, but the peaks are not so obvious

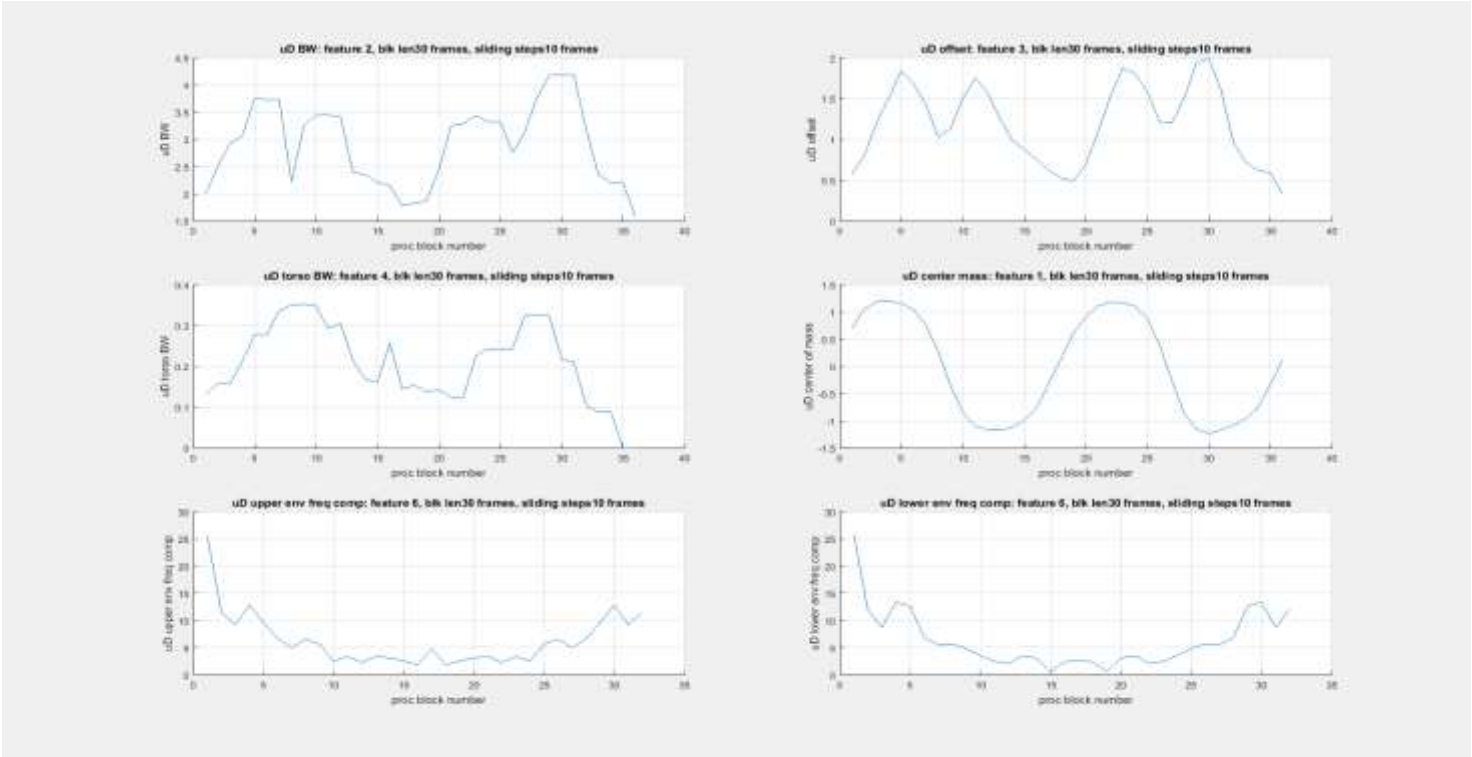
Center of mass, Up_env, low_env and power dist

Example Feature extraction from human test



uD feature 1~4 and 6

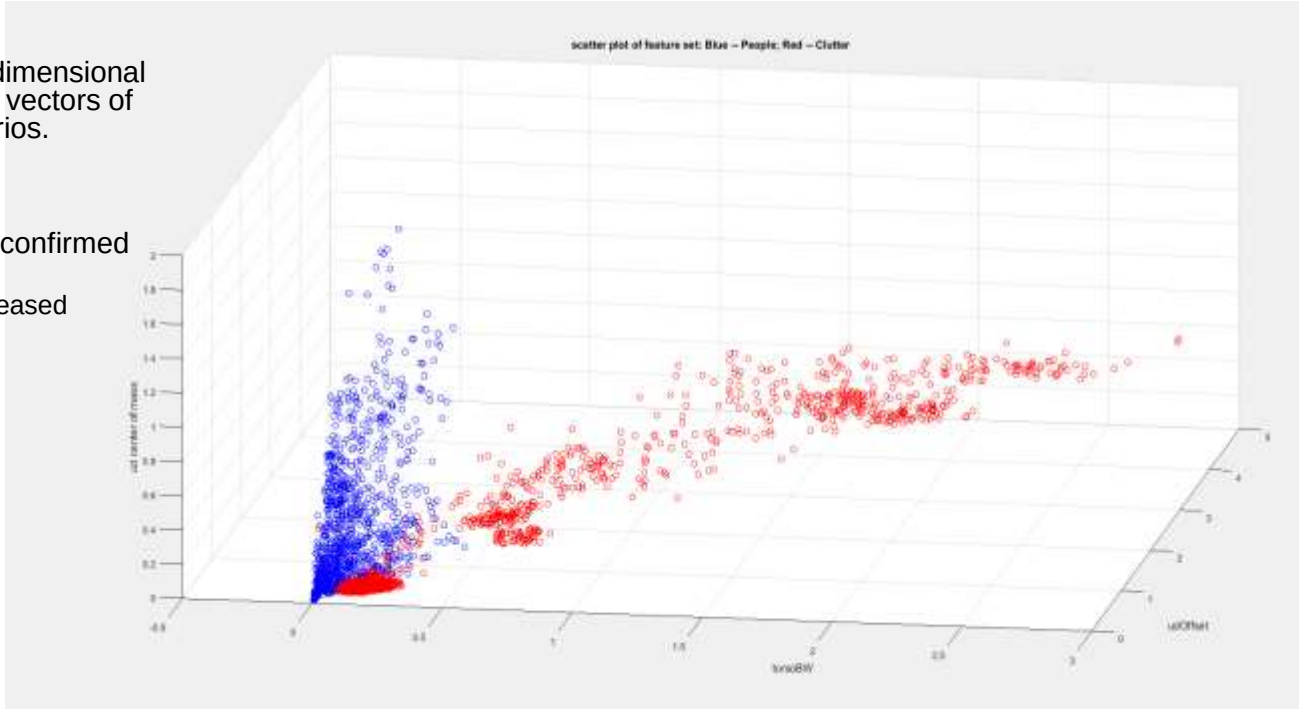
Example Feature extraction, human subject



Feature Selection:

Initial try of separating people and clutter

- By inspection, three features were recognized as showing promise for discriminating humans vs. fans and blinds
  - Micro Doppler Offset
  - Micro Doppler Torso BW
  - Micro Doppler Center of Mass Avg
- Here we see 3000 points plotted in 3 dimensional feature space extracted from ~ 30 test vectors of humans, fans, blinds in various scenarios.
  - Blue is human
  - Red is non-human
- The utility of these three features was confirmed through feature selection search.
  - No additional candidate feature increased classifier performance significantly.



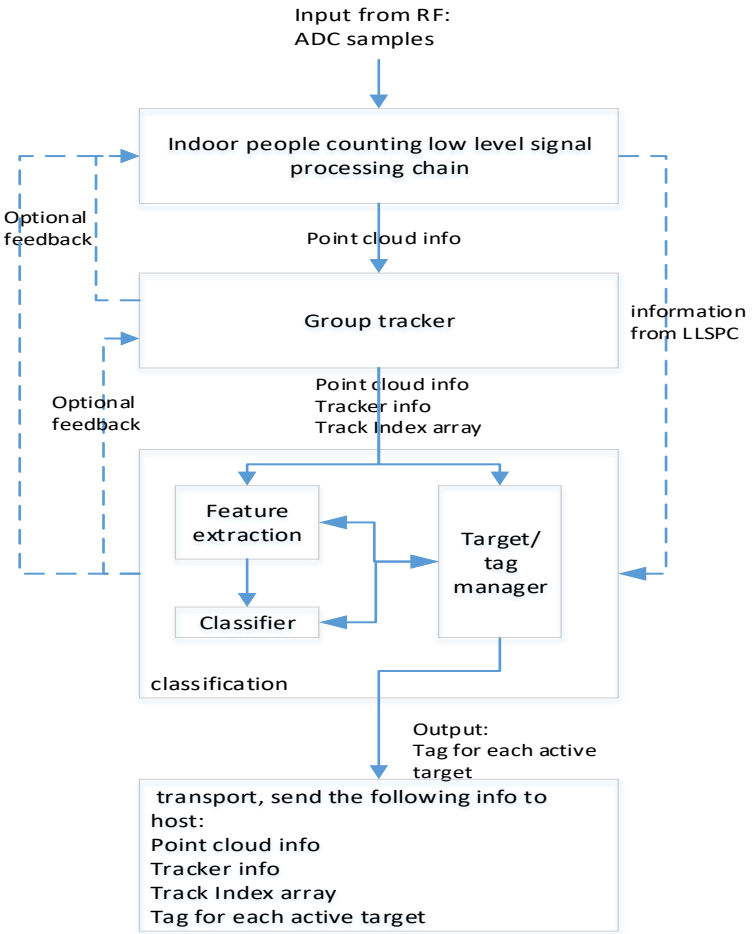
Classifier Selection

- The initial study of feature extraction for classifier design was completed, and the following features were identified as providing good separation this this problem, and for the current data set:
  - Micro Doppler Offset
  - Micro Doppler Torso BW
  - Micro Doppler Center of Mass Avg
- The features are extracted from 30 frame window with a 5 frame offset
  - A 50msec frame rate, a feature vector is generated on a 1.5 sec window every 0.5 second.
- A “snapshot” classifier is designed to look just at this 1.5sec feature vector and declare a decision every 0.5 sec.
- The training set was augmented to 9000 test vectors to help improve out of training set performance.
- K nearest neighbor(kNN) classifiers were designed and evaluated on the collected training set.
  - Using ~10% of the data set as the codebook, achieved approx 5% error rate.
  - In some sense we viewed the kNN as the simplest and quickest to design classifier to get a good measure of the performance of these features for the classification problem.
- A variant of K Means Clustering was also evaluated
  - With a codebook of 256 codewords on the same training set, achieves approx 3.5% error rate in training set.
  - We also observed codebook sizes as low as 64 codewords achieve approximately 5% error rate in training set.
- kNN was observed to be more stable outside the training set for our relatively small training set, compared to K Means Clustering so we have recommended using the kNN on the target with the largest practical codebook size (up to 600 points). It is speculated that given a much larger training set, K Means Clustering might be the preferred algorithm, but this would require further study.

Classifier Architecture

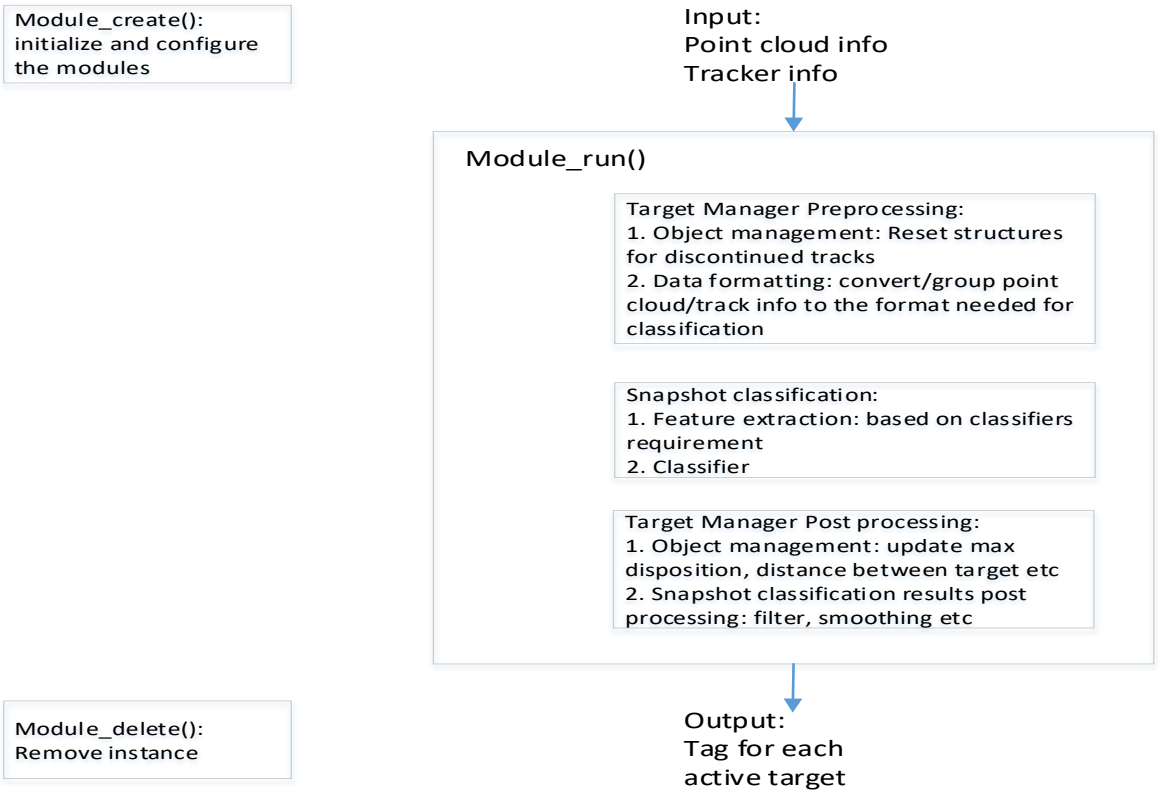
Block diagram

- For each active track, feature extraction for the 1.5 second sliding window happens partially every frame. Every 0.5 seconds, the kNN based “snapshot” classifier uses the extracted feature vector to make a short term decision on the class of the object.
- The target manager subsystem performs the outer loop processing and manages the final decision of the classifier module.
  - The configuration description document explains how the behavior of the outer loop processing can be affected by adjusting configuration parameters.
- As this block diagram suggests, in a more general design, it is possible that other Low Level Signal Processing information could be input to the Classifier. It is also possible that the output of the Classifier could be used as feedback to other sub-systems to refine the processing chain knowing the class of objects that are being detected and tracked.
- We currently do not implement any feedback of the classifier output to the other layers.



Classifier Architecture

Classification module structure



Factors Affecting Classifier Performance

- The following two slides discuss factors affecting classifier performance and other classifier design logistics.

Other design and data collection considerations

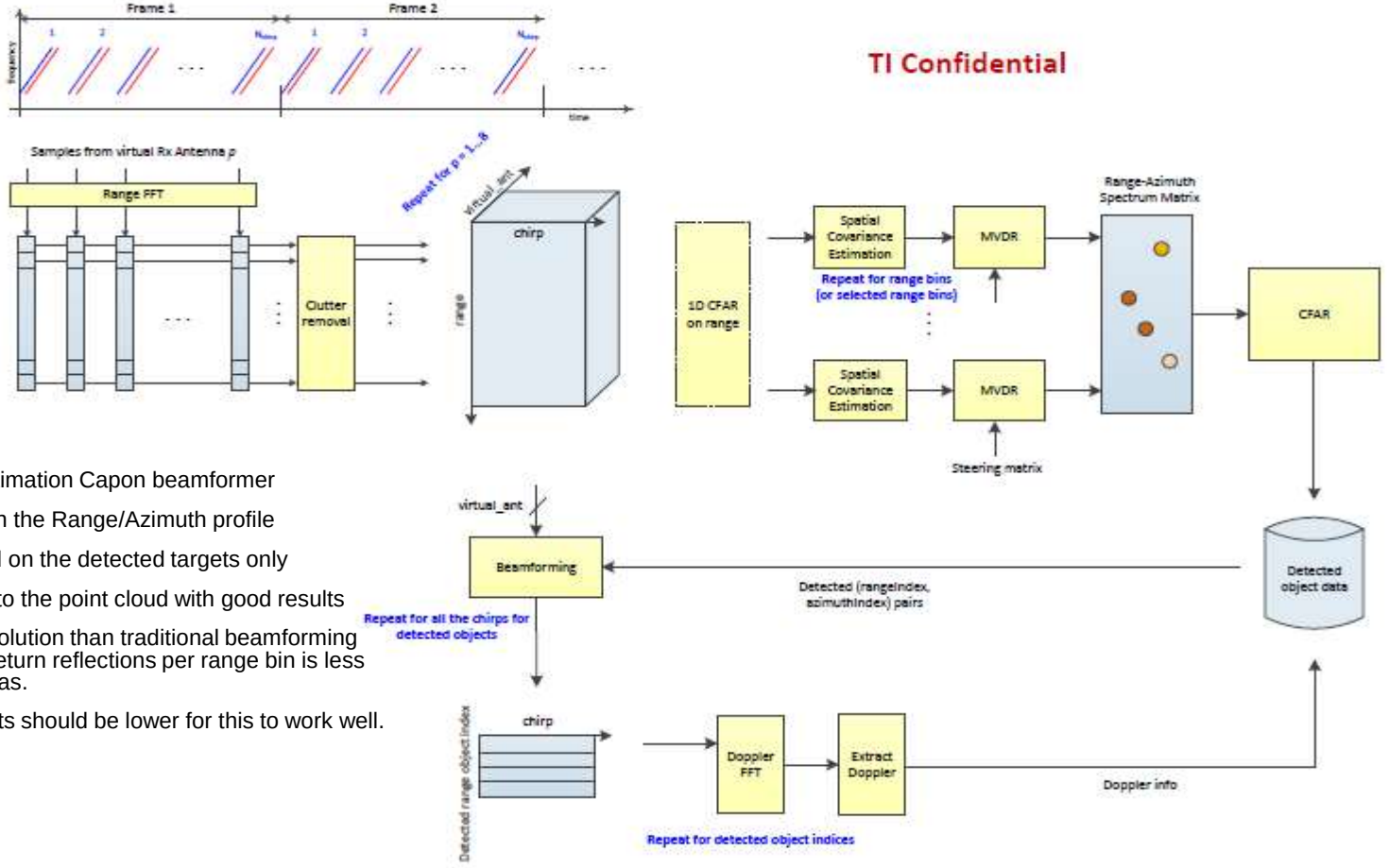
- We know in general, some of the following principles about the nature of classification problems and data collection
- That the more data, and the more varied data that we can collect and use in the classification training set, the potentially more robust we can make a classification system
- Also, the more data, and the more varied the data, the more effectively we can measure the system performance, and potentially predict the performance delta we could expect in new environments.
- The types of variations in the data that are most important, are primarily driven by the kind of problem that is being posed, and can also be somewhat driven by the kind of algorithm being applied.
- All the classification solutions we will be considering involve "supervised learning". Supervised, in this context means, the training data collected is scored, usually manually by a human, to indicate the correct answers, or the ground truth, and this information is also stored in the test suite in machine readable form.
  - **The manual scoring of the data is usually one of big practical limiting factors in gathering large amounts of data.**
- In supervised learning, the training data is processed offline to train the algorithm on a training set which is as representative as possible of all the variations in inputs that the system may encounter. The algorithm then effectively learns from this data subject to several limitations to be discussed later.
- An "in training set" performance can be observed, which is, after training, the fully trained system can be run on the training set itself and performance metrics can be reported.
- Usually, a portion of the data is reserved for "out of training set" performance evaluation. i.e. a set of data that was not used in the training, is applied to the trained classifier and "out of training set" performance metrics are reported.
- There is almost always a drop in performance for out of training set vs. in training set. These can come from several reasons. Furthermore, there is almost always some variation in performance when the classification system is then "deployed" in environments that are different than the ones used to collect the original training and test sets. Again, for several reasons which can be similar.
- Reasons for variations in the performance between the "in training set", the "out of training set" and "deployed" can be multiple but a few of them include:
  - The original training set is not varied enough to properly capture all the characteristics of the desired object classes.
  - There is some bias in the way the data set is constructed that causes the algorithm to learn a bias that isn't representative of the true environment.
  - The algorithm is overtrained or over tuned on a specific training set and, again, learns biases, or other characteristics in the data, that appear to separate the classes but are in fact specific to the training set or not generalizable.
  - New objects and new backgrounds are introduced in the deployed environment that were never seen before.
  - The input signals to the system change in some way that alters the probability distributions of the signals and/or the features extracted from the signals.
    - For example a radome is used in the deployment but not used in the original data collection.
    - The general noise level in the training data is different than in the deployed environment(particularly if there is more noise in the deployed environment),
    - the front end configuration is different,
    - there is a change in any subsequent processing of up to the extraction of the features that are the input to the classifier. i.e. point cloud extraction is a different chain or with different thresholds, or the frame sizes are changed, etc.
  - The algorithm selected, itself has inherent characteristics towards robustness, or towards overtraining etc.
- **Getting enough data do proper algorithm design and evaluation is often the key limiting factor in being able to hit performance targets, and bound expected performance variation when deployed.**
- **The second most key limiting factor once getting enough data is addressed, then is algorithm design iterations to understand the inherent robustness vs. overtraining characteristics of the different algorithm options, and to discover where the corner cases are that are driving the performance issues.**

Other design and data collection considerations

- Another set of general considerations involve the overall system performance, and in particular, the performance of the localization function.
  - Typically an object classifier, or object recognizer, etc, assumes some form of pre-processing to present the appropriately cropped, parsed, or localized input data bounded in space and time, to the algorithm that represents one object or one instance for a decision. Sometimes this is hard coded, for instance in some gesture systems, it is assumed the gesture will be performed in a specific place relative to the sensor, so the spatial localization is assumed, then there may be a preprocessing to determine when a gesture is present, to crop the gesture in time, and then present it to the classifier.
  - To use an object classifier on a general scene for generic scene interpretation applications where there are multiple objects potentially, moving in space, and appearing at different times in the scene, the localization has to be automatic, and in this case we use the group tracker to perform several key aspects of this, first, to do the initial object presence detection. i.e. the tracker detects the presence of objects in the scene by a generic object detection logic and assigns a track and track ID. Secondly, for each track, for each frame, the group tracker associates detected points with existing tracks and tags them, and then maintains a tracker state. The continuous, frame by frame stream of the tracker outputs, the tagged points, and the tracker state information, are provided as the spatially localized input to the classifier. The tracker also decides when the objects have left the scene or are no longer being tracked and deallocates the tracks.
  - The performance of the overall system and in particular the output of the classifier, is then also a function of the performance of the localization layer which also has embedded in it, an object detector, and learning mechanisms used for association and tracking the state of the objects.
  - The tracker will likely be subject to the same kinds of “in set”, “out of set” and “deployed” performance variations for training/tuning.
  - The data collection should then also be designed to help train/tune the tracker, and evaluate the performance of the tracker.
  - The Tracker training/tuning can then be viewed either as a pre-requisite, or at least a joint design/evaluation with the classifier.
    - Also note, the level of interaction between the tracker and the classifier could vary depending on the design choices and architecture.

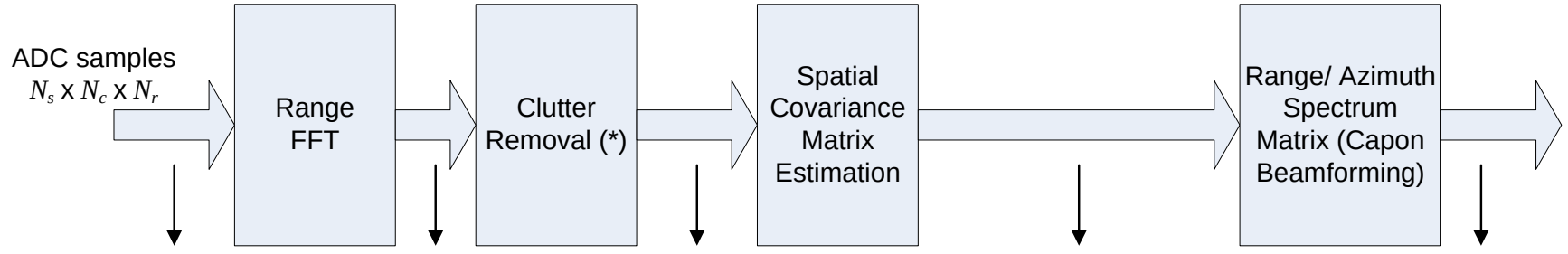
Appendix

Indoor People Detection and Localization Chain Based on Capon Beamforming aka MVDR



- Covariance based AoA estimation Capon beamformer
- Detection: 2D CA-CFAR on the Range/Azimuth profile
- Doppler estimate extracted on the detected targets only
- Group Tracking is applied to the point cloud with good results
- We see higher angular resolution than traditional beamforming as long as the number of return reflections per range bin is less than the number of antennas.
- Maximum velocity of objects should be lower for this to work well.

Processing Chain (details)



N_s x N_c x N_r

N_{range_fft} x N_c x N_r

N_{range_fft} x N_c x N_r

N_{range_fft} x N_a

N_a - Number of azimuth bins

N_s - Number of samples per chirp
N_c – Number of chirps
N_r – Number of receive antennas

$$\overline{X}_{nr} = \frac{1}{N_c}\sum_{c=1}^{N_c} X_{ncr}$$

$$X_{ncr} \leftarrow X_{ncr} - \overline{X}_{nr}$$

$$\mathbf{R}_{xx,n} \approx \frac{1}{N_c}\sum_{c=1}^{N_c} \mathbf{X}_{nc}\mathbf{X}_{nc}^H, \mathbf{X}_{nc} = [X_{nc1},...,X_{ncN_r}]^T$$

Diagonal loading: $$\mathbf{R}_{xx,n} \leftarrow \mathbf{R}_{xx,n} + \alpha \frac{tr(\mathbf{R}_{xx,n})}{N_r}\mathbf{I}_{N_r}$$

$$P_{na} = \frac{1}{\mathbf{a}_a^H \mathbf{R}_{xx,n}^{-1}\mathbf{a}_a}, \mathbf{a}_a = [1, e^{j\mu_a},..., e^{j(N_r-1)\mu_a}]^T$$

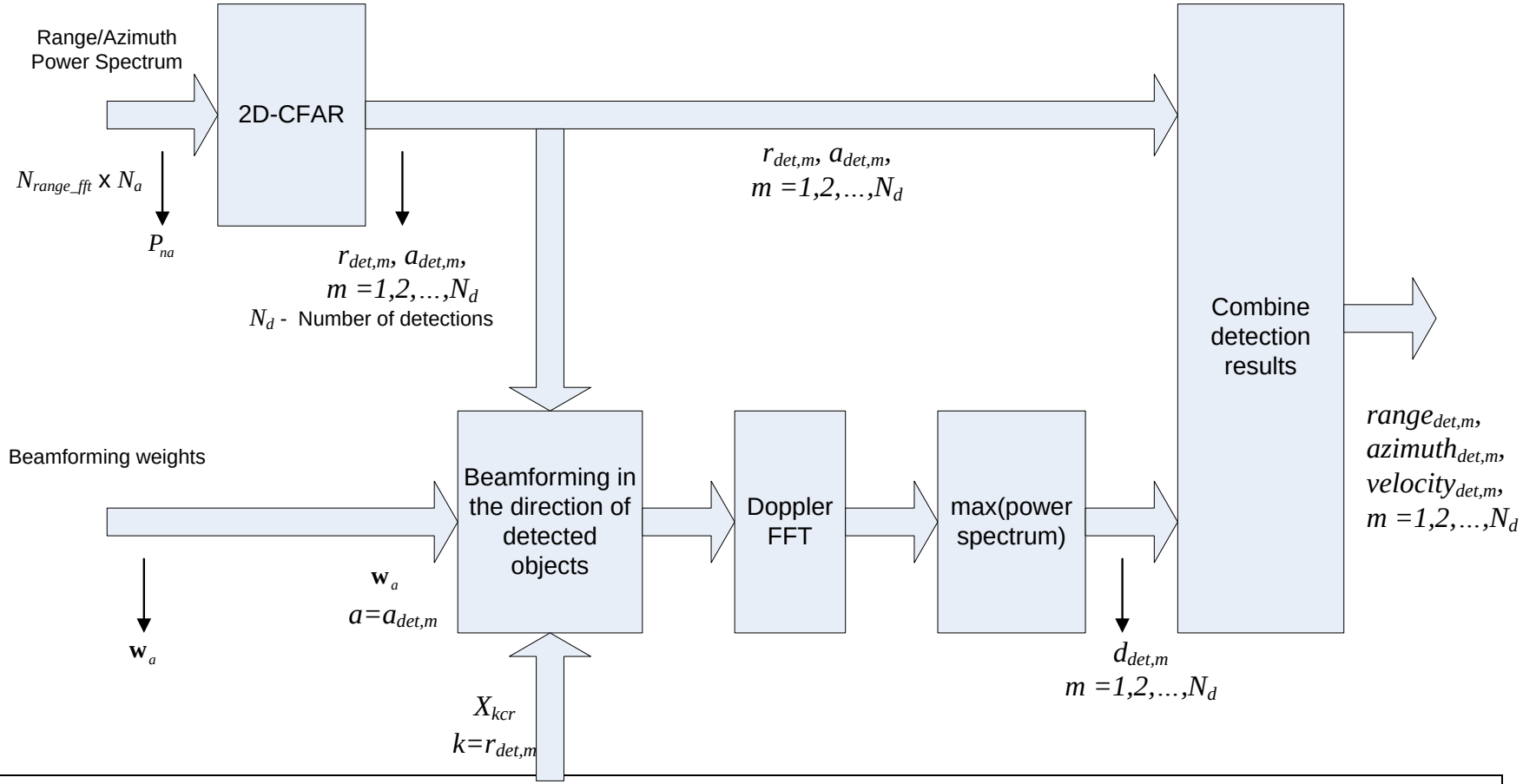
$$\mu_a = 2\pi\frac{d}{\lambda}\sin(\theta_a)$$

Beamforming weights:

$$\mathbf{w}_a = \frac{\mathbf{R}_{xx,n}^{-1}\mathbf{a}_a}{\mathbf{a}_a^H \mathbf{R}_{xx,n}^{-1}\mathbf{a}_a}$$

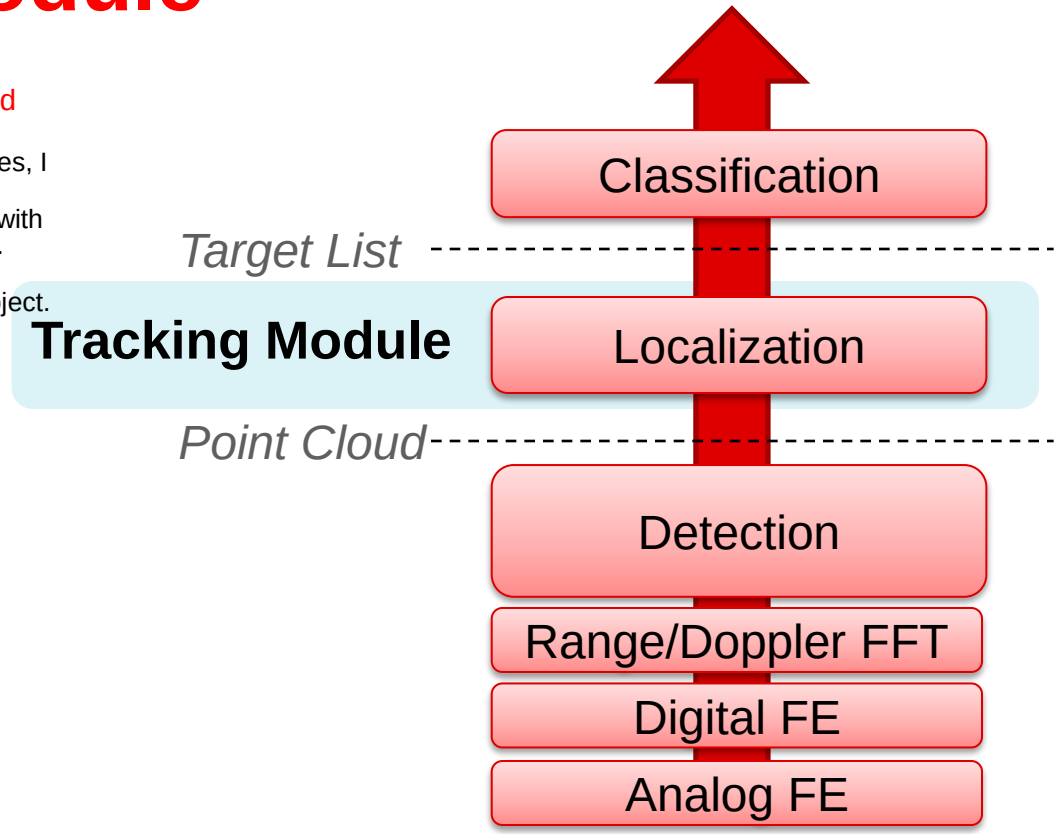
(*) See clutter removal info in SDK Doxygen documentation

Processing Chain (details, cont.)

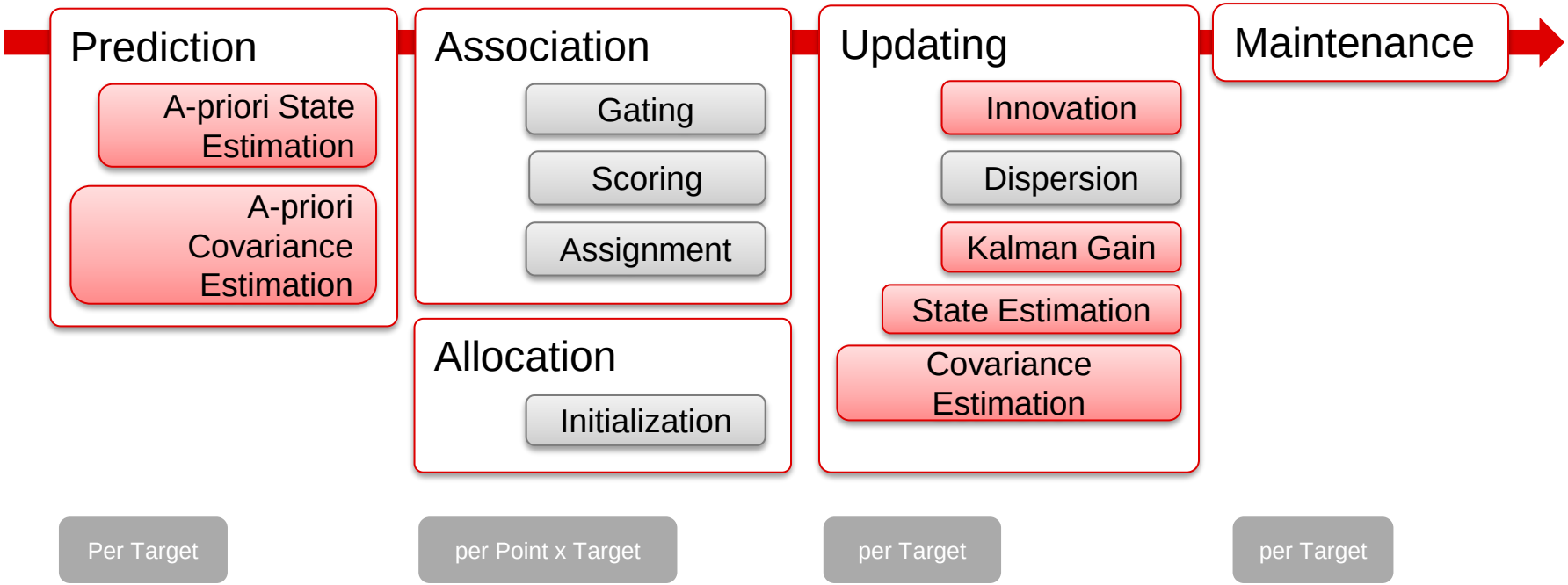


Group Tracking Module

- Implements multi-point Object Detection, Localization, and Tracking
  - I consistently observe group of points with certain properties, I detect an object to track.
  - From frame to frame, the group tracker associates points with existing, tracked objects, OR declares them unassociated.
  - Then for the allocated tracks, the tracker is observing/estimating a set of states/properties for each object.
  - Uses Detection Layer Data
  - Is a State-full Layer
- Input: Point Cloud
  - ~100s-1000 of measurements (detection points)/frame
  - Each point has range/angle/radial velocity with associated variances
- Output: Target List
  - ~10s of targets/frame
  - Each target properties may include
    - Cartesian position
    - Cartesian velocity
    - Additional Features
      - » Number of points/reflections
      - » Dispersion/Sizes/Density
      - » Timing/History
      - » Velocity spread



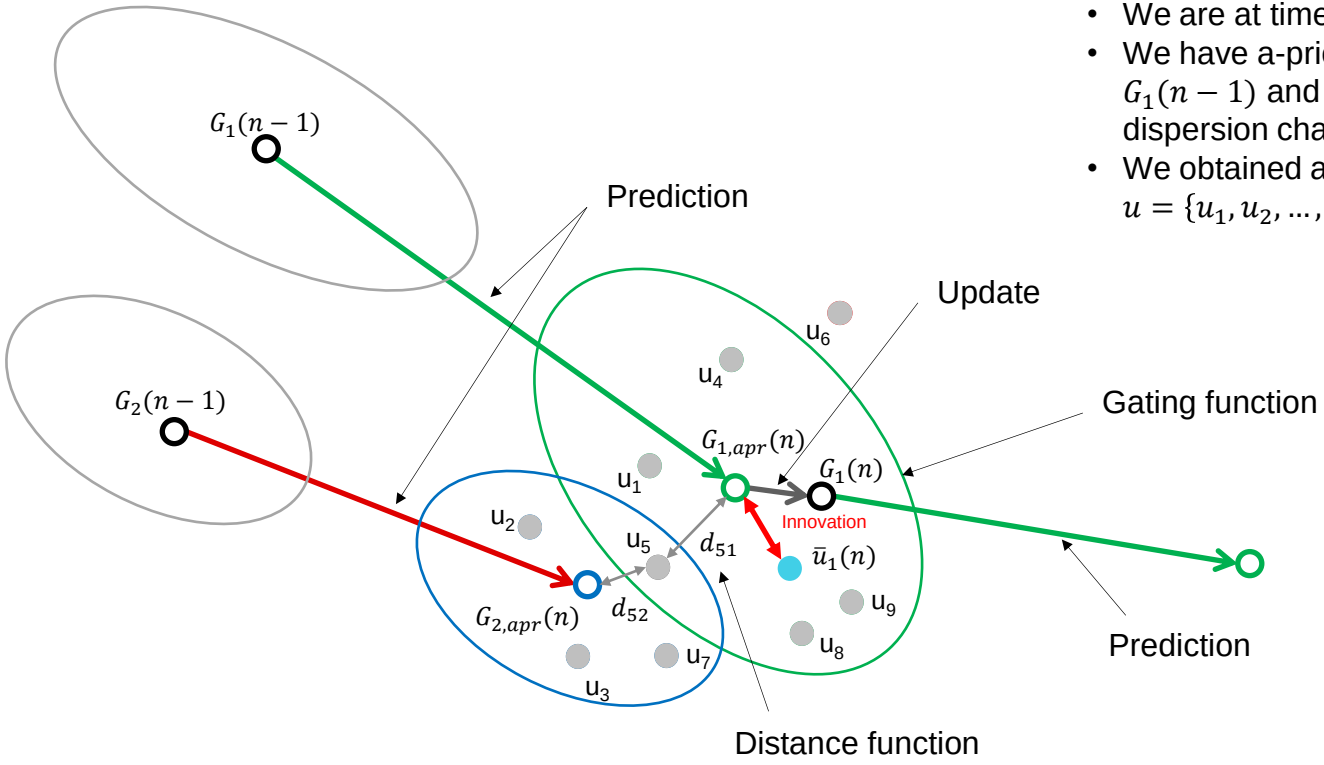
Tracking Algorithm Blocks



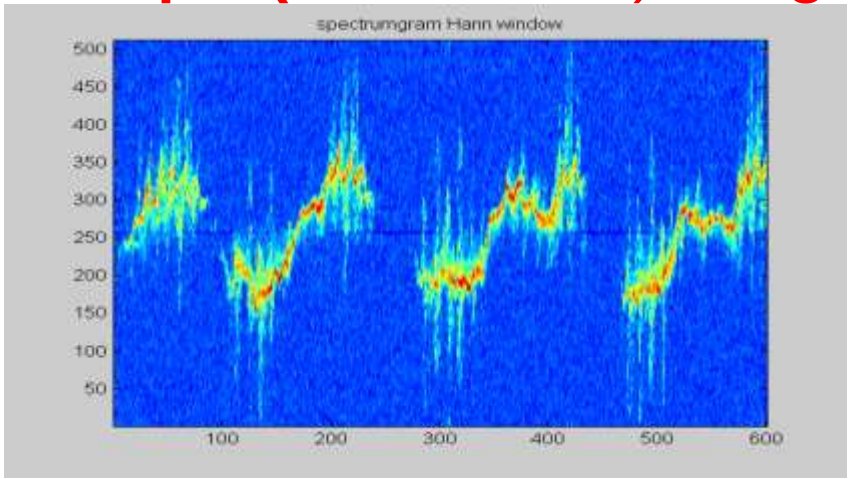
Group Tracking Illustration

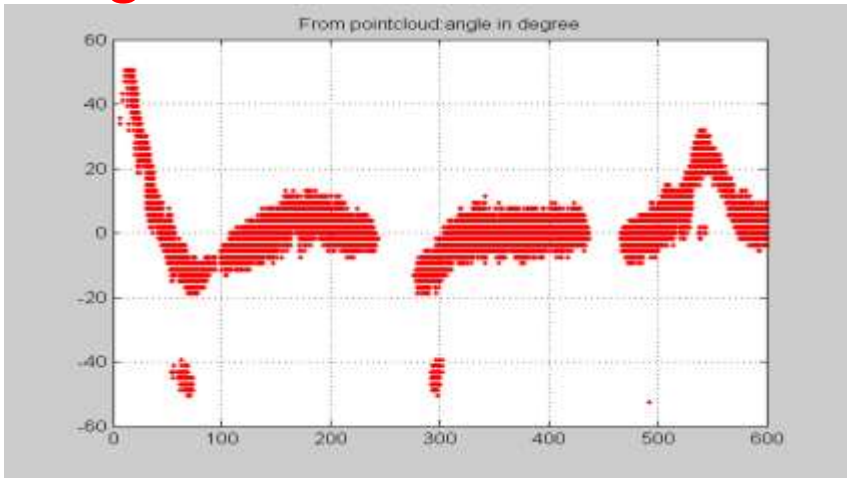
For this example we assume

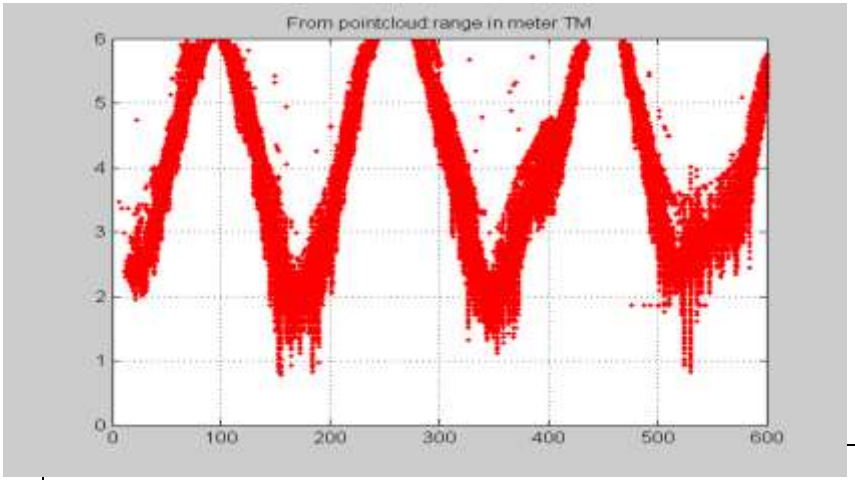
- We are at time n
- We have a-priori knowledge of two targets $G_1(n-1)$ and $G_2(n-1)$, their states and dispersion characteristics
- We obtained a point cloud with measurements $u = \{u_1, u_2, \dots, u_9\}$

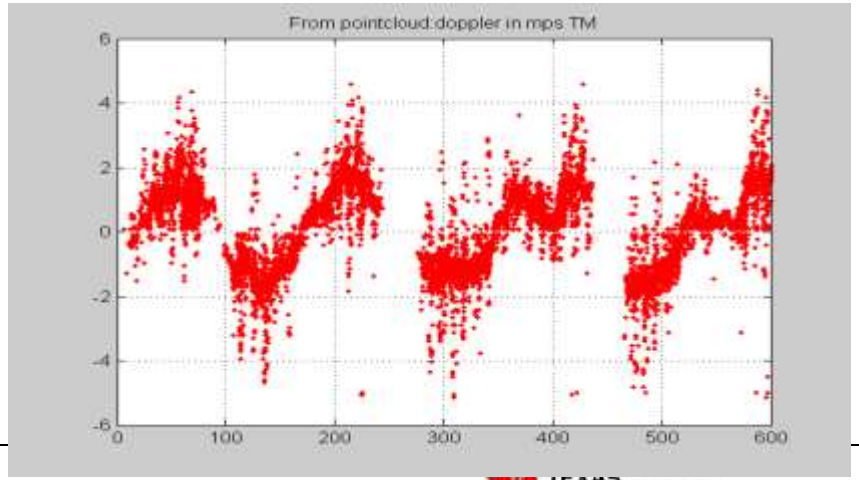


People (Scene_6old): single target, ADC+MATLAB

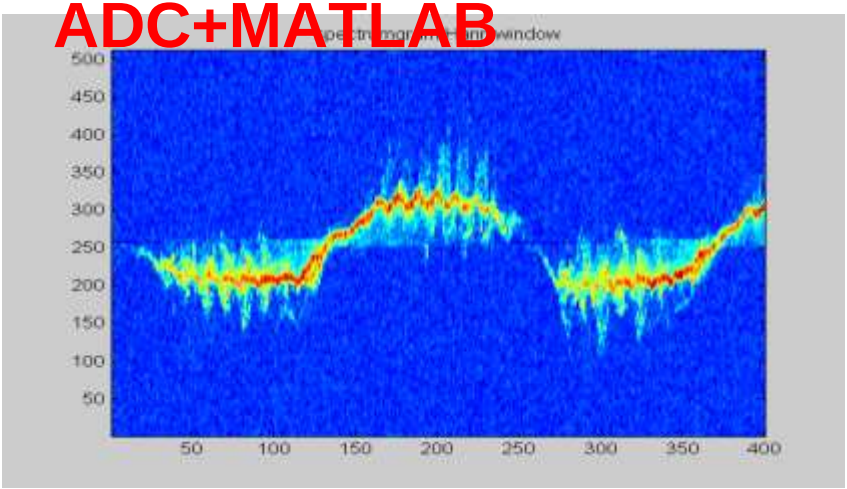


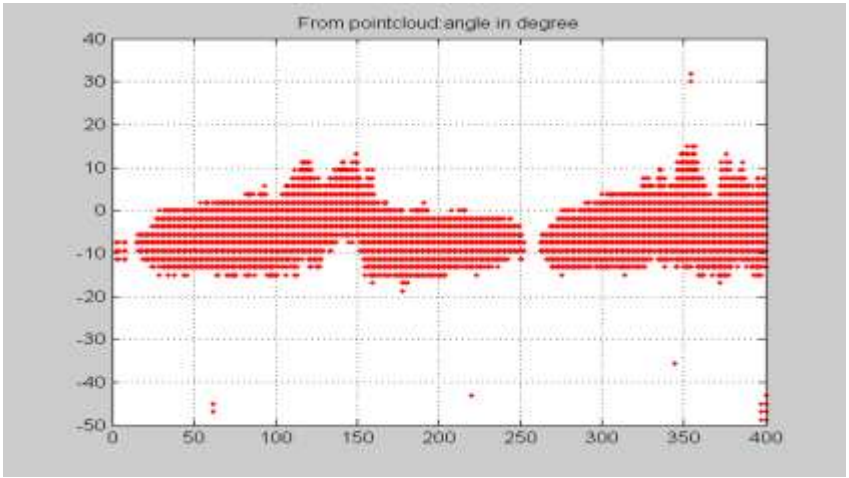


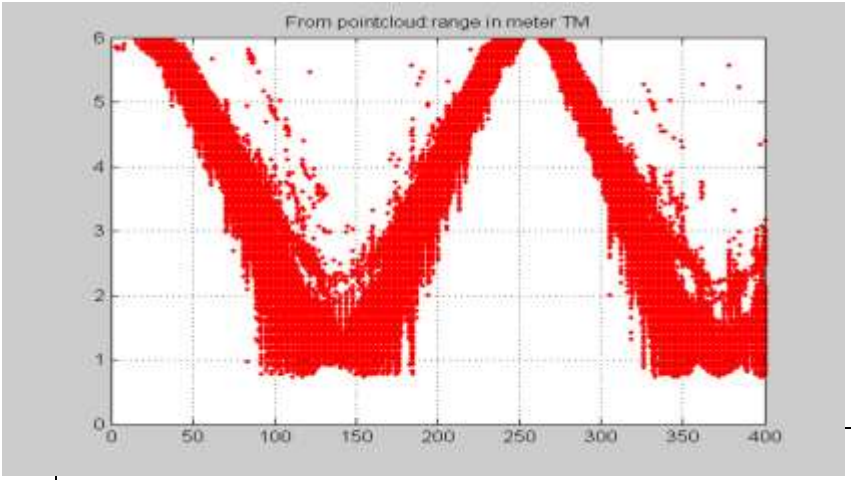


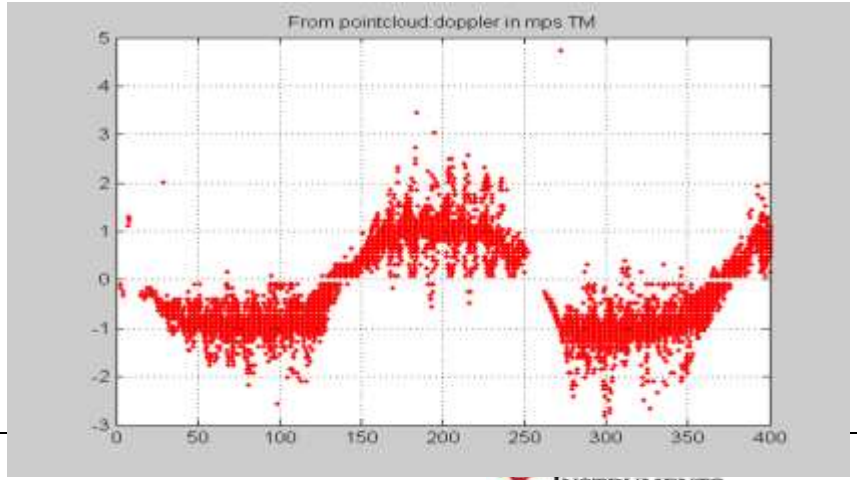


People (Scene1, ~radially walking): single target, ADC+MATLAB

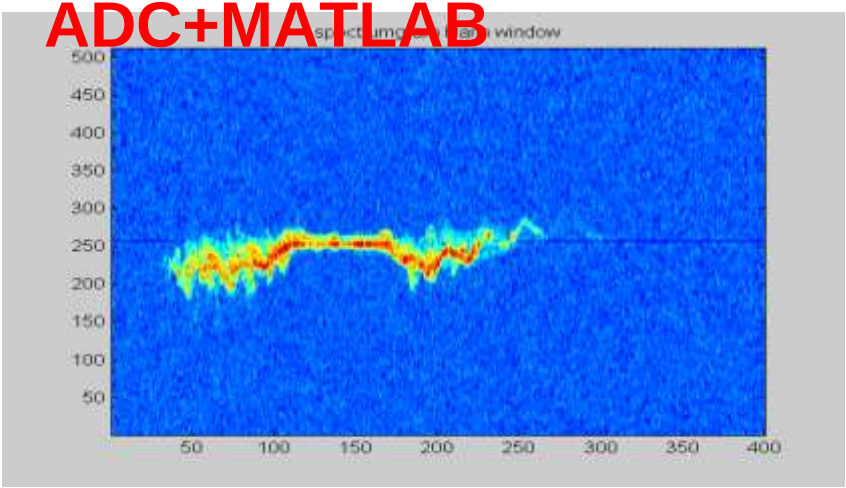


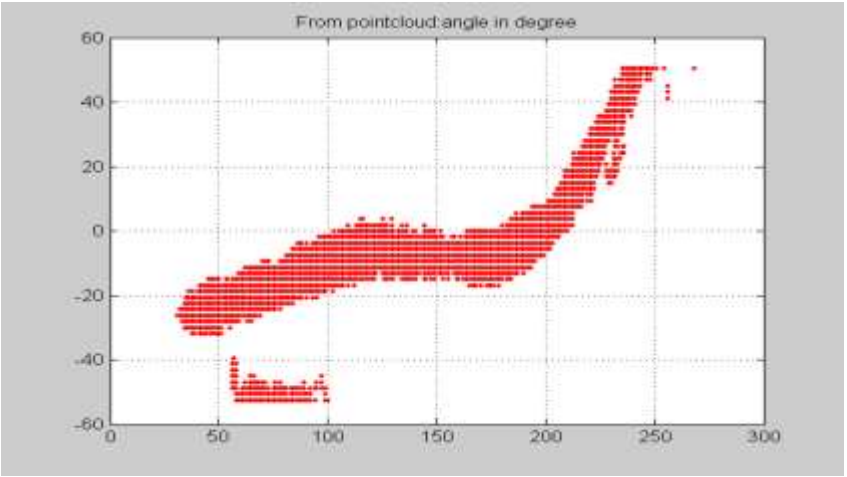


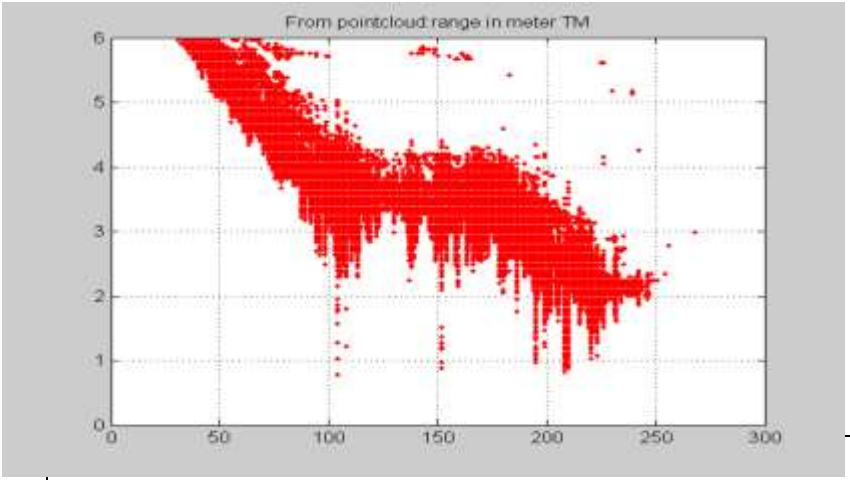


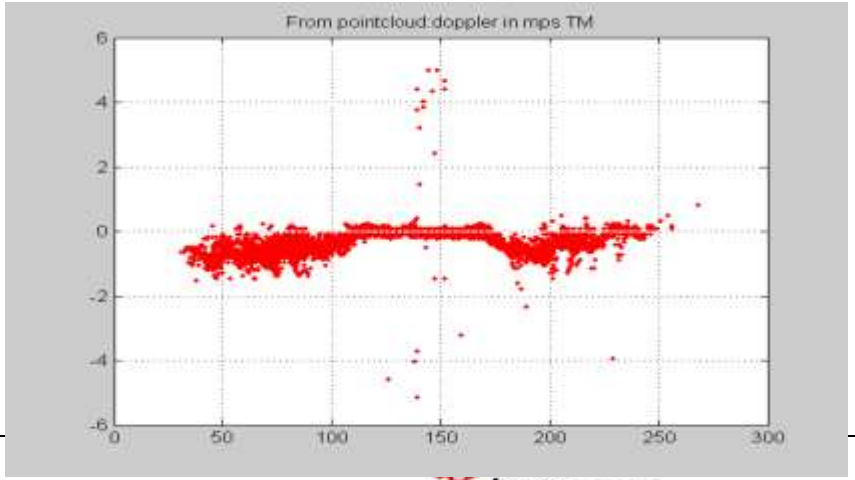


People (Scene1, ~diagonally walking): single target, ADC+MATLAB



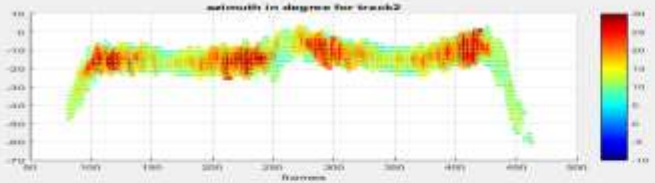


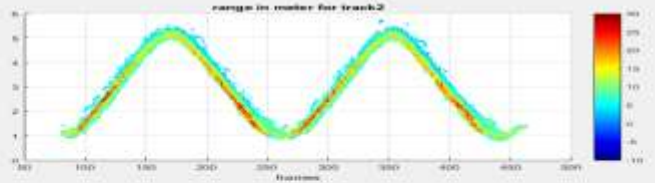


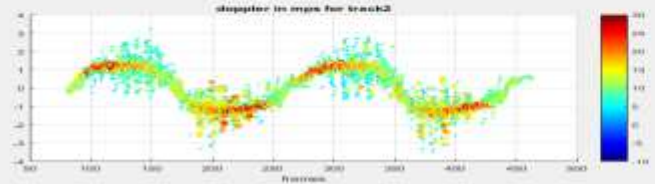


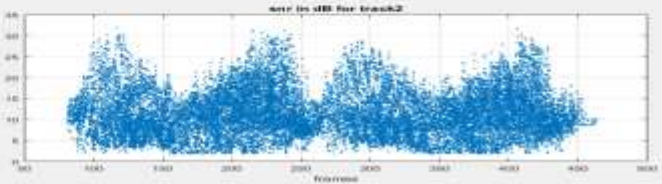
INSTRUMENTS

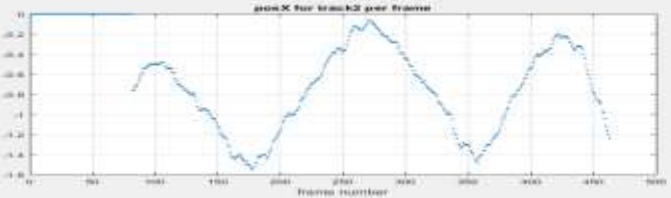
Single person walking radially

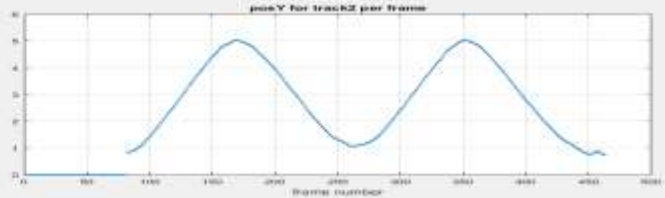


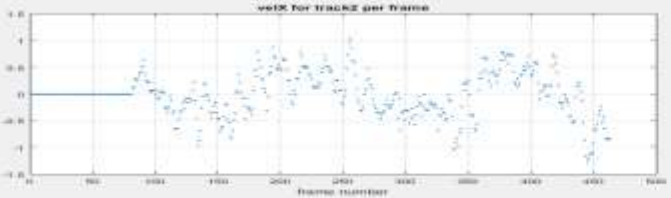


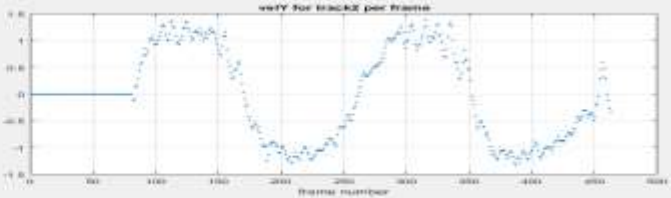




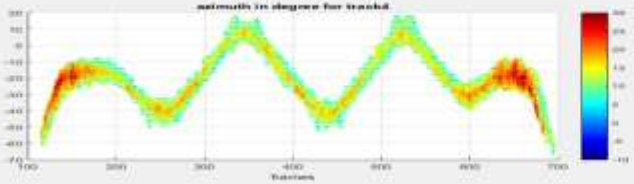


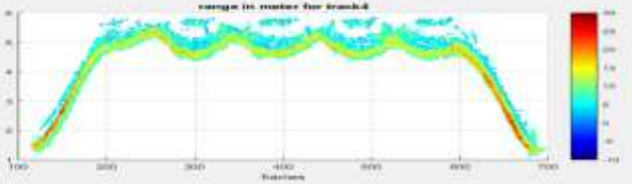


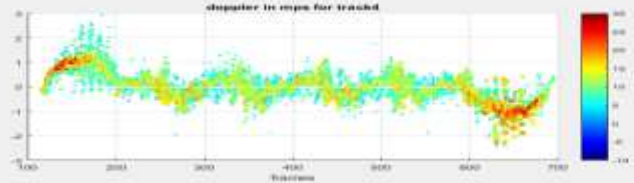


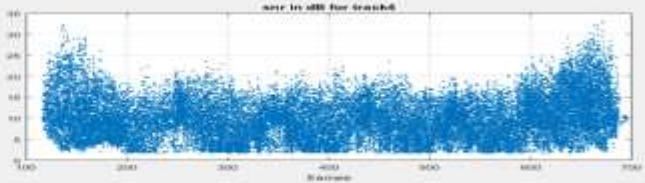


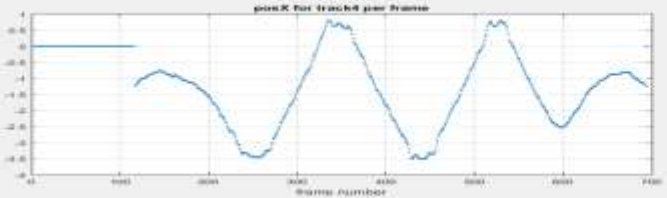
Single person walking tangentially

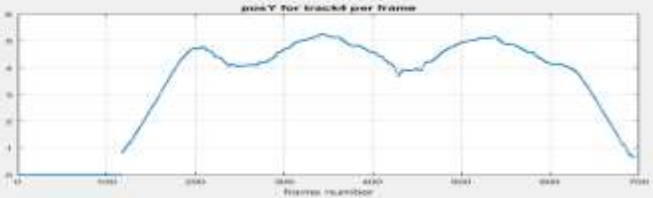


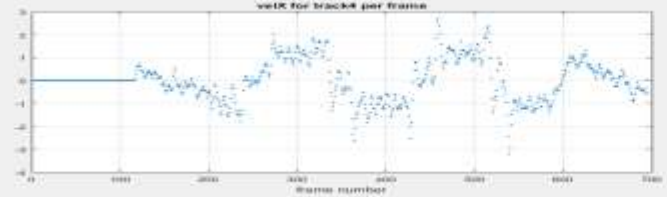


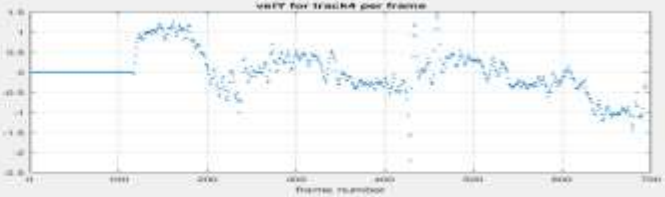






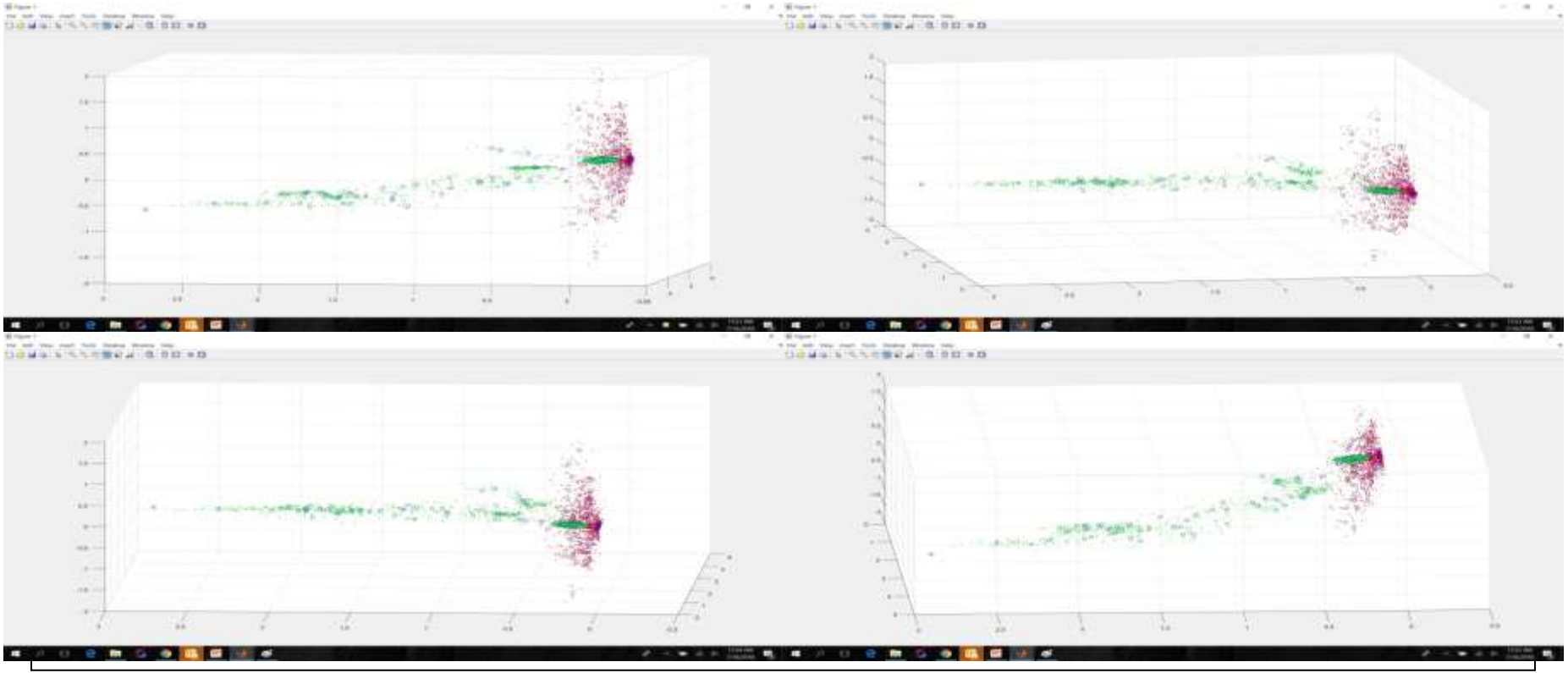






K means 256 codebook design

- The plots show the training set of ~3000 data points, with Red representing human class, and green representing the union of all the non-human classes (fan, curtain, blind).
- The blue circles are the codewords that result from the K means training over the entire training set.
- Each codeword is assigned a class, based on the training set observed probabilities.
- Input feature vector vector is encoded by minimum distance search over the codebook, and the class of the codeword is the snapshot classifier oututput.
- The output of the snapshot classifier is then post processed/filtered.



Demo and Algorithm Systems Development

Development Flow Example