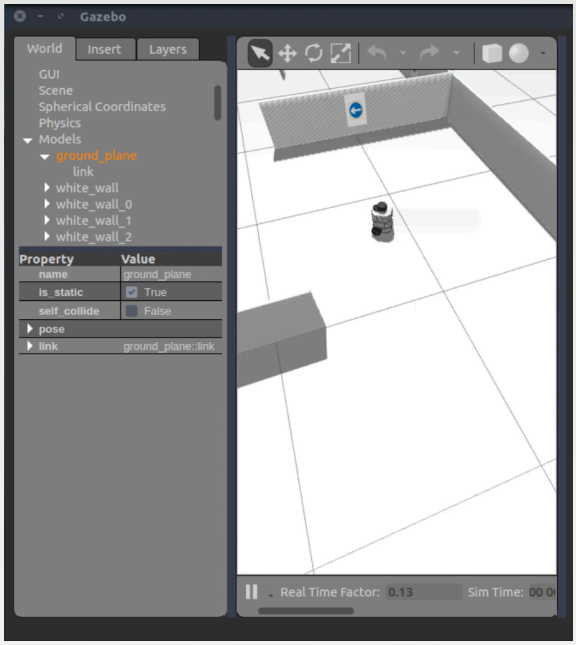
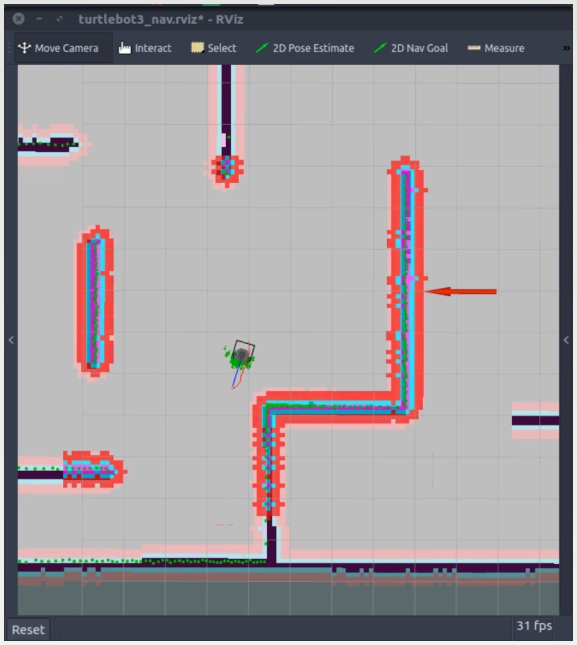
Project Portfolio

Nicolas Shu

Automated Semi-Guided Maze Path Planning with Turtlebots

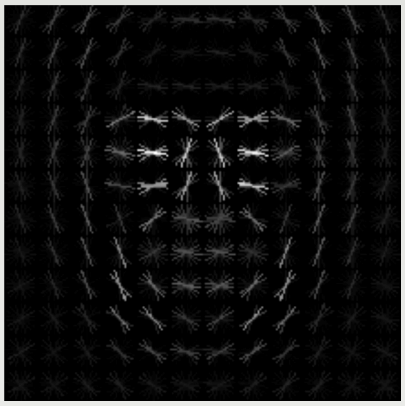


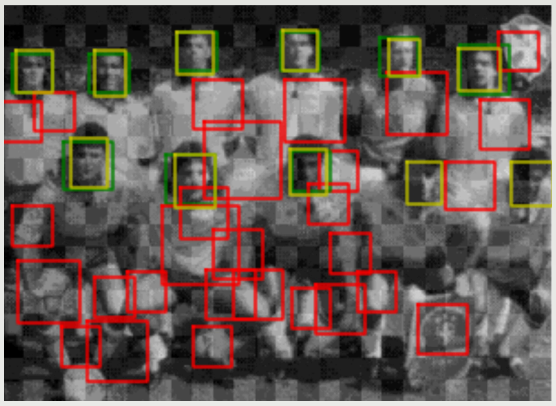


The project is focused on implementing a real-time algorithm on a Turtlebot, which is a robot built on a Raspberry Pi with ROS, to navigate through a real-life maze, where traffic figures are placed around the maze which indicate the direction to the goal. As the Turtlebot possesses a RaspiCam, extensive image processing was performed on the incoming images to then use the processed data as an input vector to a k-Nearest Neighbors algorithm to identify the suggested direction. Once the robot had identified the direction, the robot would use a PID controller to traverse to the direction, making sure to contour walls, and continuously loop through the algorithm until it found the goal. The project was first implemented on Gazebo, and once it worked, the hyperparameters were reoptimized for the real-life maze since the simulated world was scaled differently.

Platform	**Languages/Libraries**
Gazebo	Python
ROS	OpenCV

Face Detection with Dalal-Triggs Algorithm





Identifying faces on an image has become a very important tool, especially in digital photography. Although Viola-Jones were the initial algorithms used in digital cameras to identify faces, this project was implemented with the methodology described by Dalal-Triggs, by utilizing the concept of histogram of gradients (HoG) to create features to be passed through a support vector machine classifier. The classifier was trained on positive and negative samples, and then mined for hard negatives, as described by Dalal-Triggs, by training it again on false-positive examples to obtain a more robust classifier. The dataset was further augmented by adding variational disturbances to increase both positive as well as negative samples (e.g. face flipping every face double the positive dataset). A sliding window was passed through the HoG feature space in parallel to passing through the image and then converted to the HoG space, to then be passed through the support vector machine classifier. To avoid an enormous set of detections, adaptive non-maximal suppression was implemented to prevent "overcounting", and to allow focused detection on a face.

Languages/Libraries

Python

OpenCV

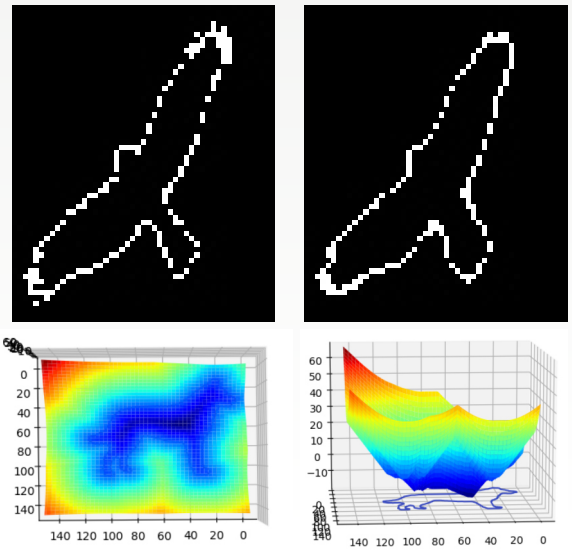
Scipy

Scene Recognition with Bag of Words

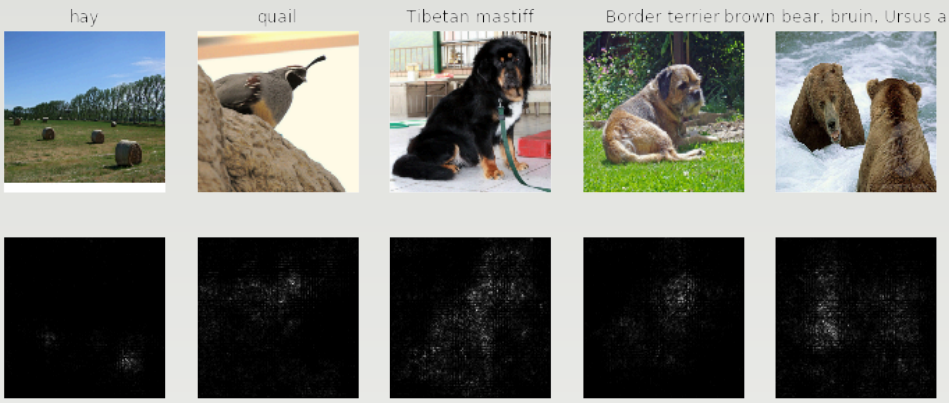
Around the world, there are different interpretations on decorations of one's home or how a street may look, or the appearance of houses in the suburbs. This project aims to perform a scene recognition by using the concept of "bag-of-words" borrowed from Natural Language Processing in order to create features and to perform a multiclass classification. This was achieved by obtaining a large number of SIFT descriptors for the images, and to cluster the via an unsupervised learning algorithm such as k-means, and each cluster then becomes a "visual word" and the compendium of words become a "vocabulary". When training (and testing), each image yield SIFT descriptors, but instead of storing them all, they are distributed along the visual words and a histogram is created for such image describing how many SIFT descriptors are in each visual word, and the histogram is used as the feature for the classifier. The classifier implemented was a multiclass support vector machine using the one-vs-all fashion.

Using Level Sets to Develop Active Shape Models of images

Identifying contours to an object in an image is an important tool in computer vision that can be expanded into multiple directions, such as using it as priors for image segmentation. This project sought out to have a raw implementation of active shape models by using level sets created by principal component analysis in mathematics. By using a training set to create an efficient set of principal components allow the components to work in congruency to create entire level sets based on signed distance functions that the cross-sections are evolved to become contours of a specific type of object in an image. On the images on the right, the top left eagle is the true contour and the top right eagle is the predicted contour using level sets. An example of level sets for dogs is shown on the bottom with two different perspectives of the surface depiction.



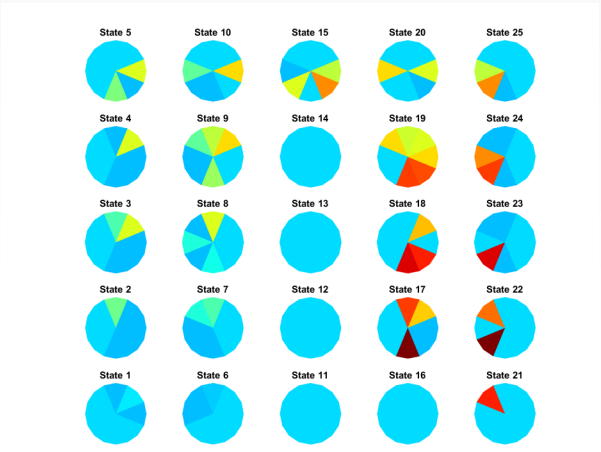
Saliency Maps of Weights in Neural Networks

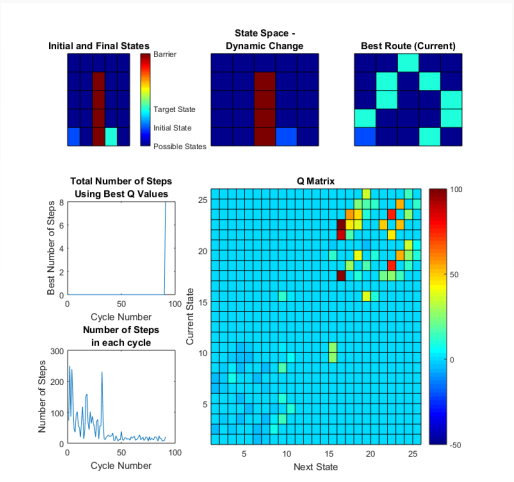


This small project was a reproduction of the work from Simonyan et al. in PyTorch where it is the observation of the saliency maps of how would weights focus on which sections of an image based on what the network had learned. This is achieved by looking at the partial derivative of the scoring function to a specific class with respect to the image itself, which yields the images shown above.

Q-Learning Algorithm for Path Planning

By creating a very basic, low dimensional map with a boundary, a reinforcement Q-learning algorithm was implemented, which follow major concepts used in dynamic programming to identify the optimal path based on past experiences to find a goal on the environment. An additional version of the project was also implemented by adding a probability distribution of where the goal could be located such that the optimal path would be influenced by the previous experiences.



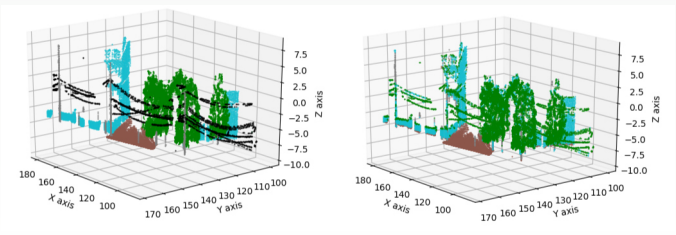


Adversarial Search on a Queen Isolation Game with Minimax and αβ-Pruning

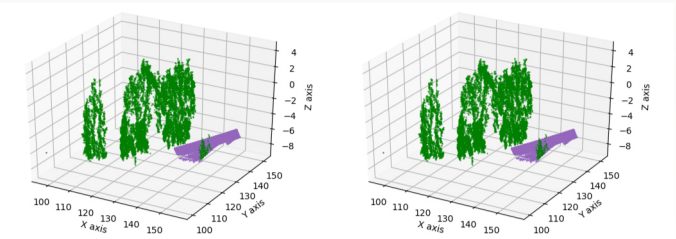
The project involved implementing a modified version of the Queen Isolation game with the features that a player could kick another player off the board. This adversarial search involved the implementation of the minimax algorithm, as well as the alpha-beta Pruning version of the minimax algorithm. Because the game followed a time constraint of a few seconds for each round, both algorithms were wrapped to perform iterative deepening in order to find the best possible move, given an objective scoring function, to play within a time limit.

Online Learning Classification of LIDAR Observations of an Environment

Given recordings from a drone flying outdoors on an environment and collecting LIDAR data, this project involved comparing the performance of three different classification algorithms to be implemented (raw) in an online learning fashion in order to classify the different sections of the environment. The algorithms used on the online learning problem were perceptron, support vector machine classifier, and Bayesian Linear regression. As they can be seen on the following picture, the environment classes were cable wires (black), poles (gray), a building wall (cyan), trees (green), and a terrain on the ground (brown).



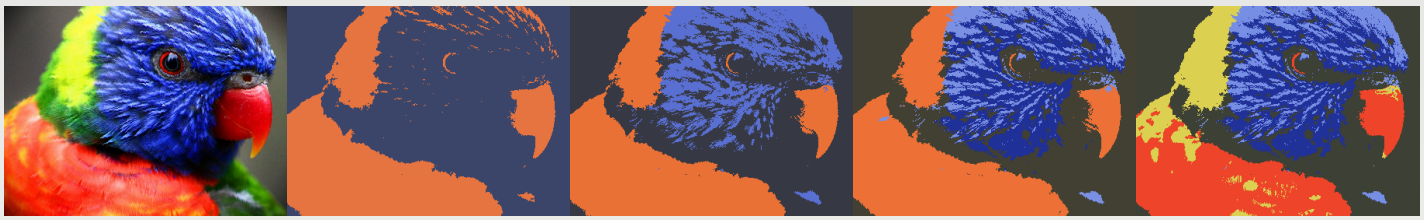
The ground truth (left) and the predictions from a gradient descent algorithm (right)



The ground truth (left) and the predictions from a Bayesian Linear Regression (right)

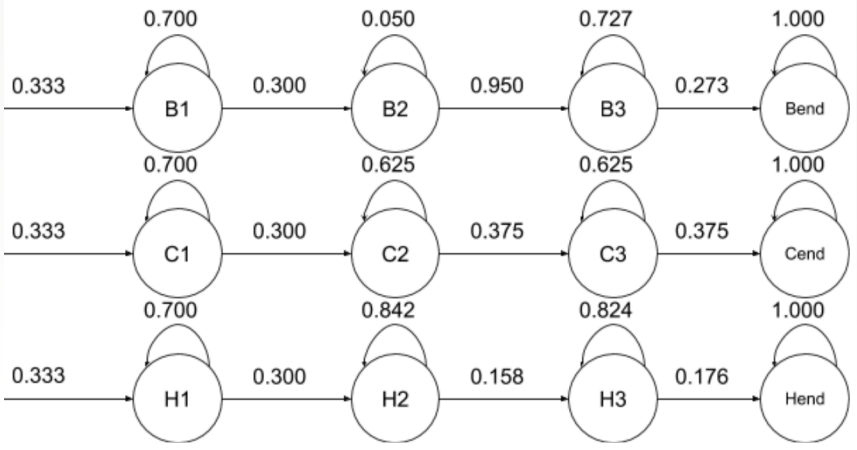
Image Segmentation with Unsupervised Learning Algorithms

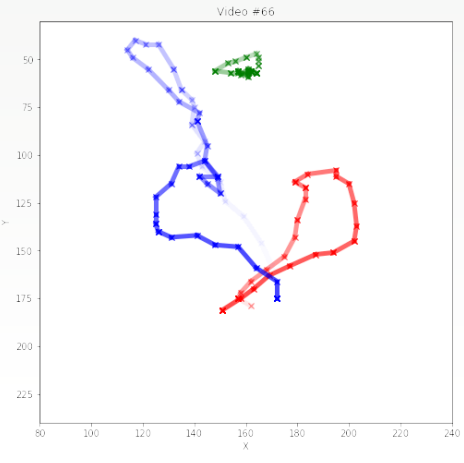
Often times, datasets do not have a ground truth label, hence unsupervised learning algorithms allow clustering to occur on such types of data. This project involved the raw implementation of k-means algorithms and Gaussian Mixture Models in vectorized forms to beat time limits to perform image segmentations over the colors of photographs. In other words, the algorithms were required to be fast enough to segment images before a timer ran out by using vectorization over the code. The original photograph is shown on the left, which is followed by segmentations sorted by number of components (or centroids), which are 2, 3, 4, and 5.



American Sign Language Interpretation with Hidden Markov Models and DTW

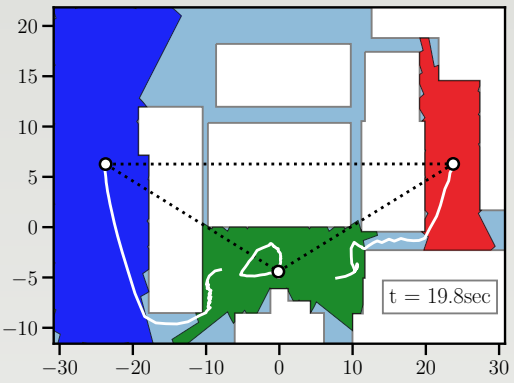
As temporal data are very interesting and challenging to work with, dynamic time warping algorithms were used on a 2D projection of a single hand as well as the 2D projection of two hands when doing ASL signs to be translated to words. This involved a raw implementation of hidden Markov models and a raw implementation of the Viterbi-Trellis algorithm to allow the propagation over the HMMs.





Robust Maximum Coverage Networked Control for Sensor Networks

Sensor placement in an environment is important for maximum coverage. However, typical algorithms used for maximum coverage are incapable of performing maximum coverage in non-simply connected environments without good initial conditions. An interactive framework was built in Python in order to run experiments for networked agents. It is capable of creating tessellations in non-simply connected environments, and it allows for continuous measurements, allowing for quantitative determination of covered area. A robust networked control was created to allow for randomized initializations, and to provide exploratory controls for each individual agent. This entire project was made from scratch

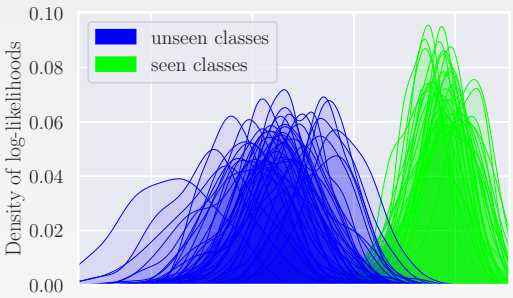


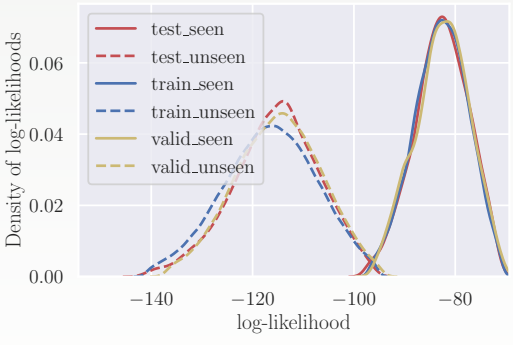
Languages/Libraries

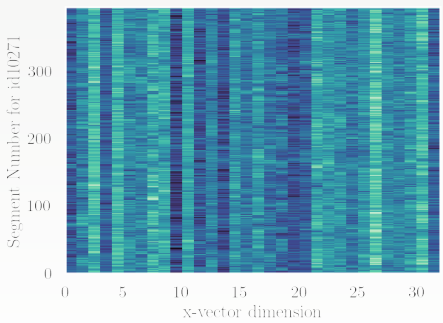
Python

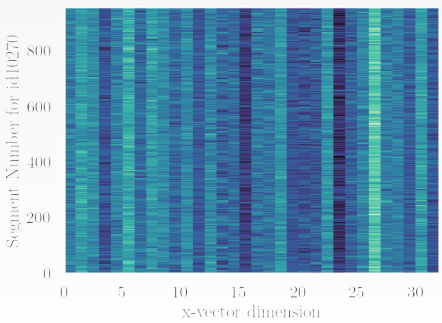
Detection of Out-of-Class Samples in Speaker Identification

When designing a speaker identification system, it is important for a real-time system to be able to detect whether an individual has been heard before or whether it belongs to a brand new speaker. This project integrated probability theory, few-shot learning, and speaker identification systems to develop a system capable of detecting whether a voice belonged to the set of registered speakers or not, reaching F1 scores up to 0.92.







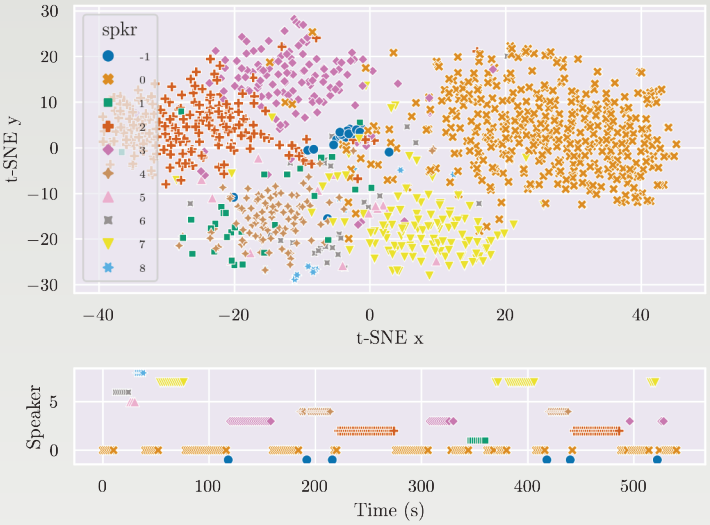


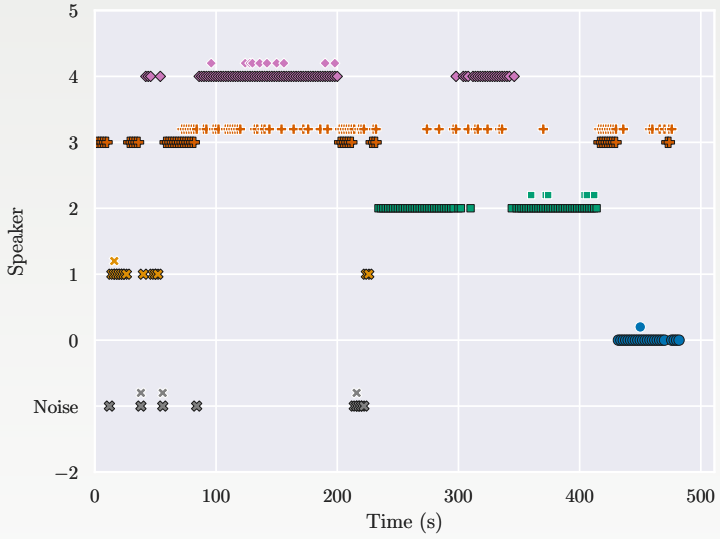
Languages/Libraries

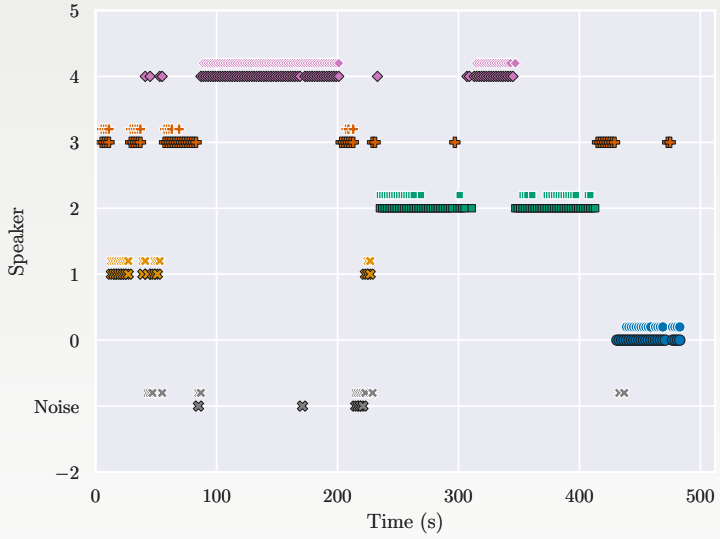
Tensorflow

Few-Shot and Zero-Shot Speaker Identification

Most speaker identification systems require enrollment of voices in order to identify those individuals by their voices. However, this requires a large amount of speech, ranging between 30mins to hours of speech, making it difficult to obtain. This project coupled prototypical networks with probability theory and dynamic adaptation to develop a system capable of learning an individual's voice within 2.5 seconds, operate in real-time, and re-identify and adapt them via zero-shot learning.





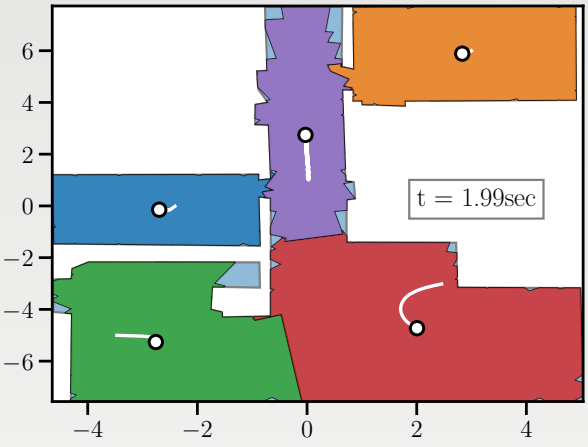


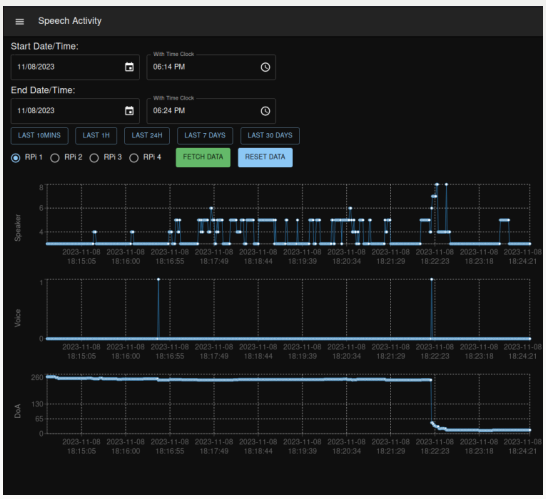
Languages/Libraries

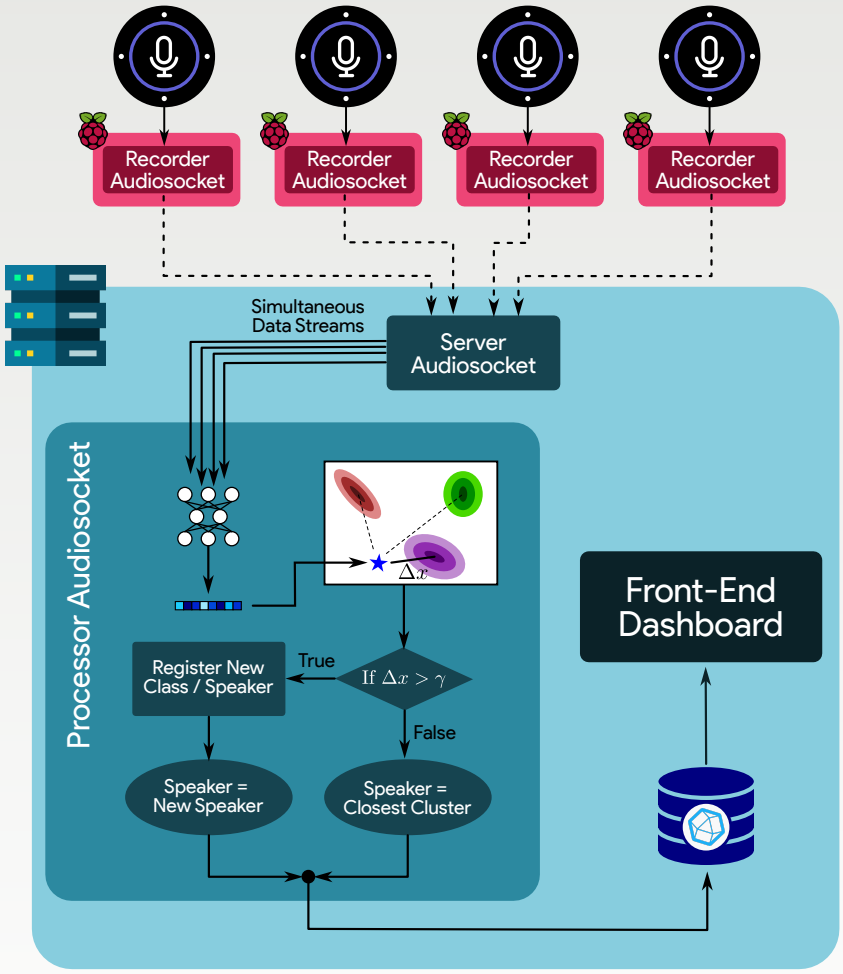
Tensorflow

Audiosockets and Dashboard for Real-Time Speaker Identification

In this project, I created a Python package capable of socket programming allowing multiple nodes (e.g. Raspberry Pis) to communicate with a central server in real-time to do real-time signal processing for speaker identification under very low latencies. The project would send real-time data to the server, which would log it onto an InfluxDB database after it had been processed, and show the results on a front-end dashboard. This was mounted inside a home, which was mapped via a LiDAR sensor, and the microphones were placed at optimal locations in the home.







Languages/Libraries

Python
Javascript
ReactJS
Tensorflow
Networked Control