



NICOLAS SHU


PROJECT PORTFOLIO

@ nicolas.s.shu@gmail.com

 linkedin.com/in/nicolasshu

 nicolasshu.com

 github.com/nicolasshu

 (718) 612-4856

Projects

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

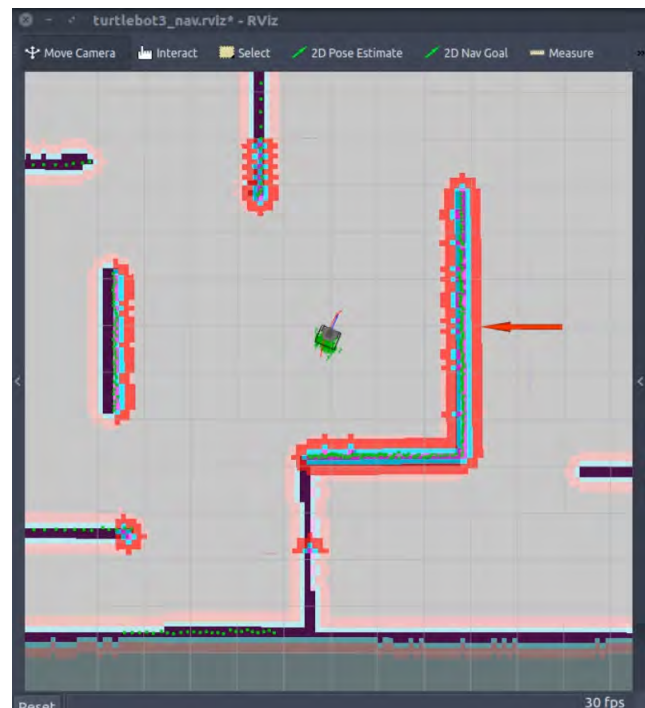
Gazebo

ROS

Languages and Libraries

Python

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 and Libraries

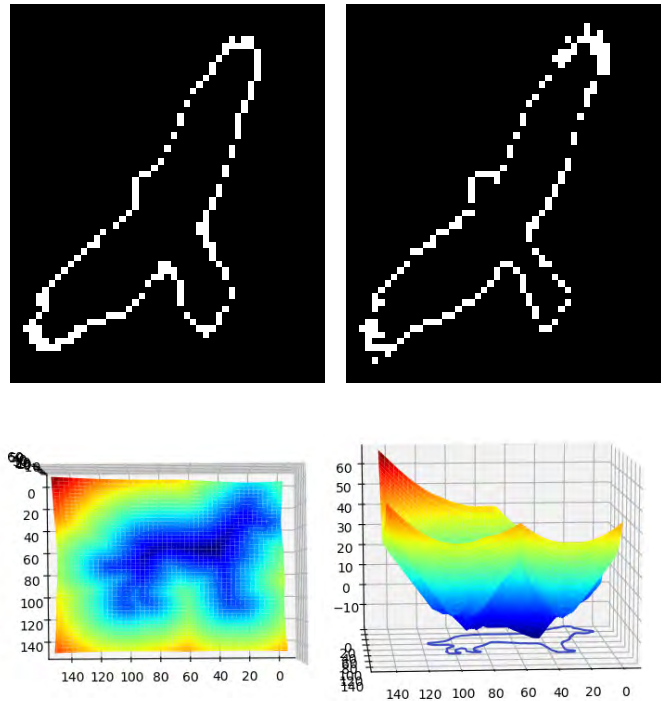
Python

OpenCV

Scipy

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.



Languages and Libraries

Python

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 multi-class classification. This was achieved by obtaining a large number of SIFT descriptors for the images, and to cluster them 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 yields 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.

Languages and Libraries

Python

OpenCV

Scikit-Learn

Scipy

SALIENCY MAPS OF WEIGHTS IN NEURAL NETWORKS



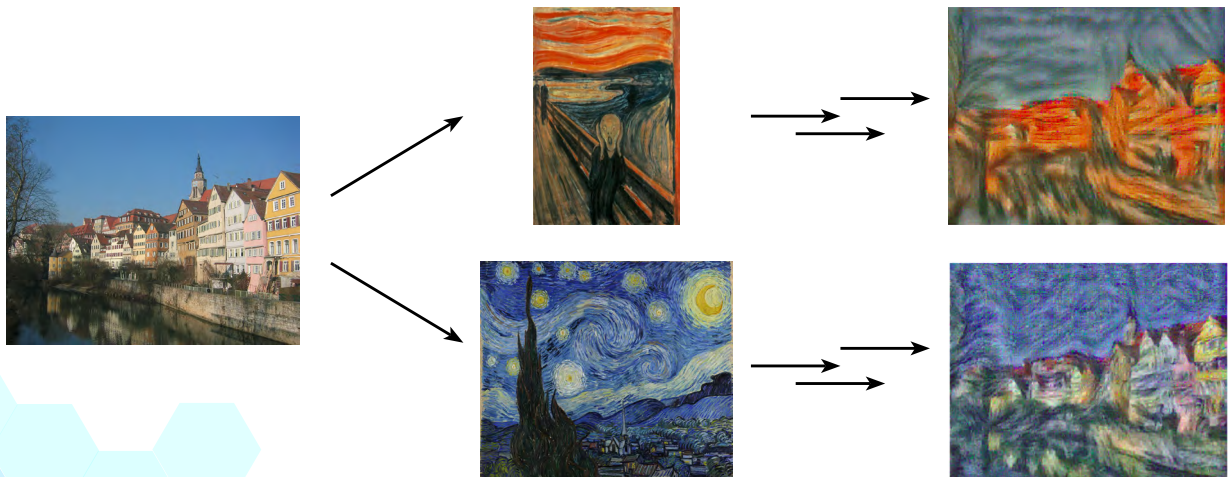
This small project was a reproduction of the work from Simonyan et al. where it is the observation of the saliency maps of how would weights focus on which sections of an image based on what the network had learned. This is achieved by looking at the partial derivative of the scoring function to a specific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

Languages and Libraries

Python

PyTorch

IMAGE STYLE TRANSFERRING USING CONVOLUTIONAL NEURAL NETWORKS



This small project replicated the works from Gatys et al. to obtain painting styles from famous artists to be transferred to actual images by utilizing convolutional neural networks.

Languages and Libraries

Python

PyTorch

RECURRENT NEURAL NETWORKS AND LONG SHORT-TERM MEMORY NETWORKS FOR IMAGE DESCRIPTION

a young man with cap skating on a slope <END>
GT:<START> a young man with cap skating on a slope <END>



a man standing on the side of a road with bags of luggage <END>
GT:<START> a man standing on the side of a road with bags of luggage <END>



a bed with some chairs and a wooden tv <END>
GT:<START> a bed is very close to a window with dark <UNK> <END>



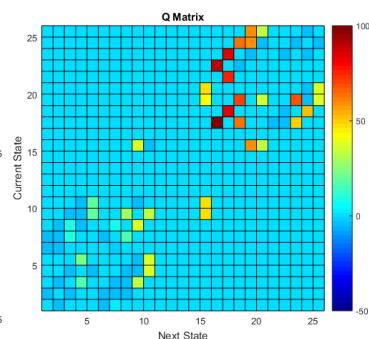
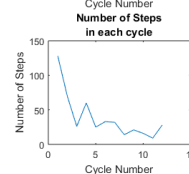
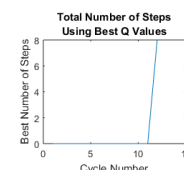
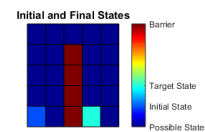
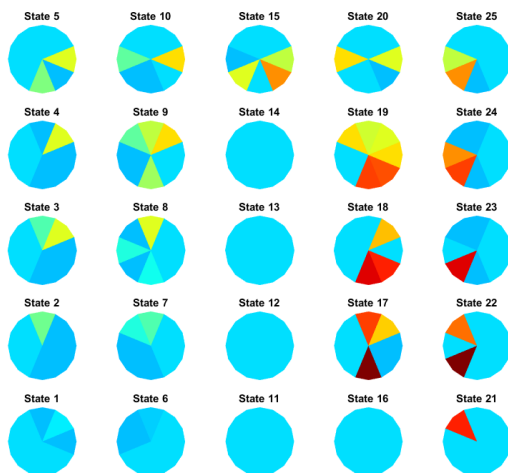
Identifying contours to an object in an image is an important tool in computer vision that can be expanded into multiple directions, such as using it as priors for image segmentation. This project sought out to have a raw implementation of active shape models by using level sets created by principal 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.

Languages and Libraries

Python

PyTorch

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.

Languages and Libraries

MATLAB

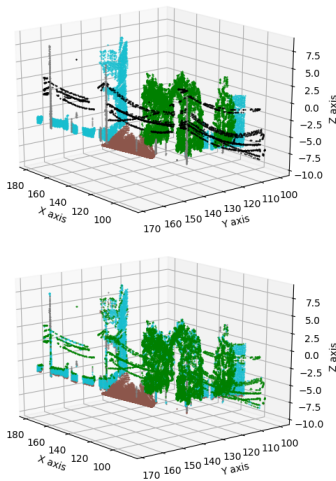
ADVERSARIAL SEARCH ON A QUEEN ISOLATION GAME WITH MINIMAX AND $\alpha\beta$ -PRUNING

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.

Languages and Libraries

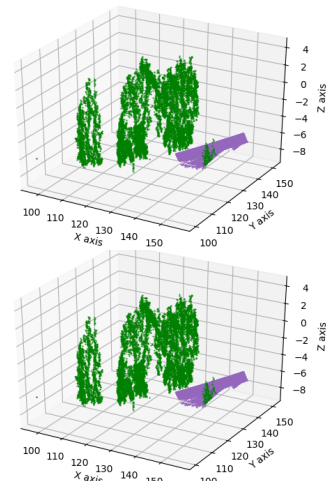
Python

ONLINE LEARNING CLASSIFICATION OF LIDAR OBSERVATIONS OF AN ENVIRONMENT



The ground truth (top) and the predictions from a gradient descent algorithm (bottom)

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 (top) and the predictions from a Bayesian Linear Regression (bottom)

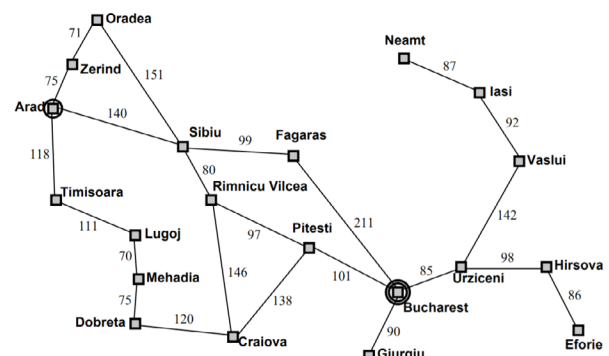
Languages and Libraries

Python

Scipy

PATH PLANNING VIA BREADTH FIRST SEARCH, UNIFORM COST SEARCH, A*, BIDIRECTIONAL UNIFORM COST SEARCH AND BIDIRECTIONAL A*

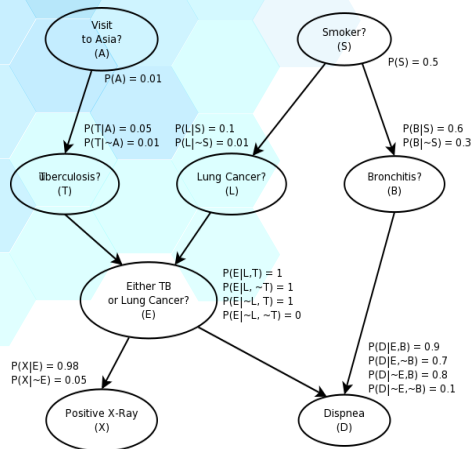
This project involved the implementation of various path planning algorithms to solve optimal paths to take in order to go from a starting point to a goal point. The algorithms used were Uniform Cost Search (UCS), Breadth First search, A*, Bidirectional UCS, and Bidirectional A*. The map used were that of Romania (as shown on the left).



Languages and Libraries

Python

ANSWERING PROBABILISTIC PROMPTS VIA PROBABILISTIC ALGORITHMS



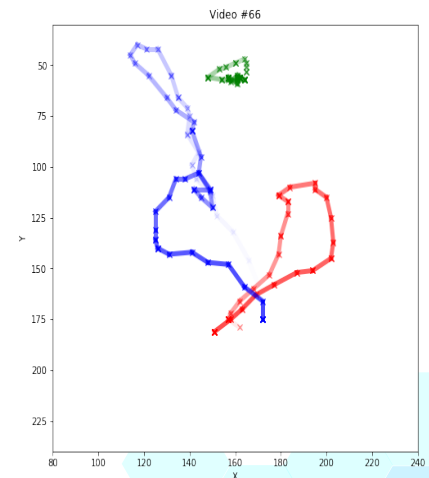
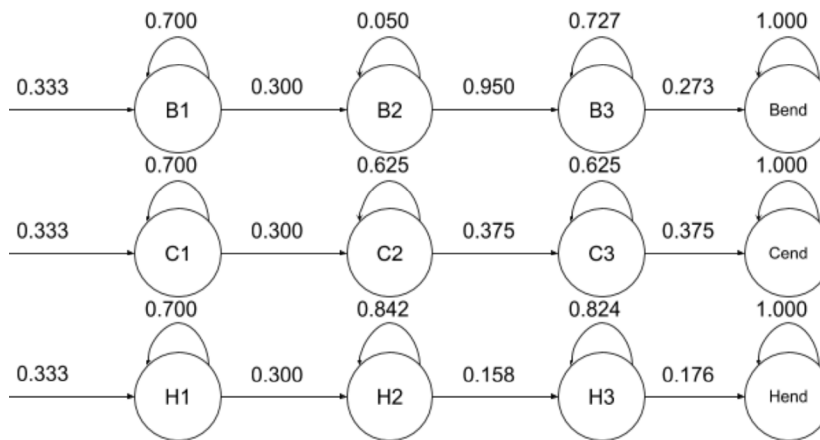
Many problems in the real world contain multiple parts with multiple options. As an example, a nuclear power plant may be prone to failure, which could be indicated by temperatures, however the temperature sensor could be working or not, and the alarm ring could be working or not. If one was to receive an observation, it is important to identify marginal probabilities, such as the probability that the temperature is in fact elevated or not. However, such probabilities are not easily computed, as the analytical solution computation would exponentially increase as the number of components increases. Bayesian Networks are good solutions to accurately compute marginal probabilities given certain observations for the system (e.g. an alarm ringing). Another alternative is to use sampling methods, such as Gibbs sampling and Metropolis-Hastings sampling in a Monte-Carlo fashion to obtain accurate approximations of the probabilities of the system.

Languages and Libraries

Python

Gmpy (Lib for Prob. Graphical Models)

AMERICAN SIGN LANGUAGE INTERPRETATION WITH DYNAMIC TIME WARPING ALGS.

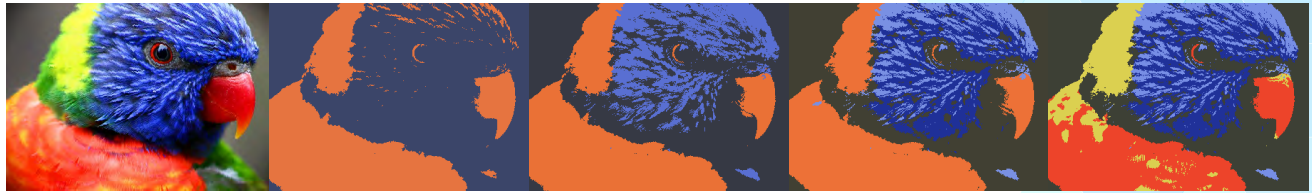


As temporal data are very interesting and challenging to work with, dynamic time warping algorithms were used on a 2D projection of a single hand as well as the 2D projection of two hands when doing ASL signs to be translated to words. This involved a raw implementation of hidden Markov models and a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

Languages and Libraries

Python

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.

Languages and Libraries

Python

SHOPAHOLIC: AN ANDROID APP THAT SEARCHES PRODUCTS BY LOCATION NEAR YOU

Many times, one wishes to purchase an item in an emergency, and it becomes crucial for the user to go to a location where the item is in stock, and at the time, there wasn't an Android application capable of performing such task. To perform the task successfully, it was required to implement a system which could identify item stocks in real-time over the information that companies publish on their websites. This was a team project which focused on developing an Android app that combined the APIs from Google Maps, Semantics3 and Yelp in order to identify the availability of commercial products by Euclidean distance from the phone's location. When the user entered the product's name on the application, the system used Semantics3 to identify the price and the availability of the product over a high number of companies. The results would then be passed to the Yelp API to locate the companies' addresses, which, when one option (one company versus another) would be selected, it would be sent to the Google Maps API to provide directions to the destination. I was responsible for all of the sections, including the user interface, except for the operation and implementation of the Google Maps API.

Platform

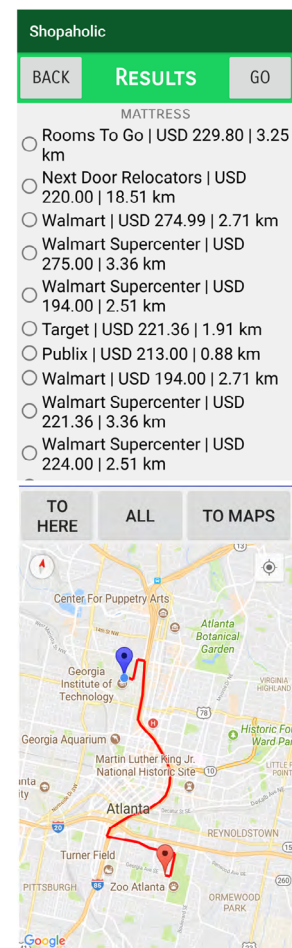
Android Studio

Languages and Libraries

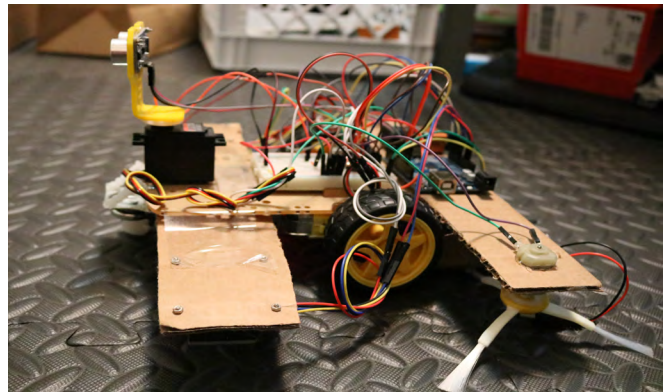
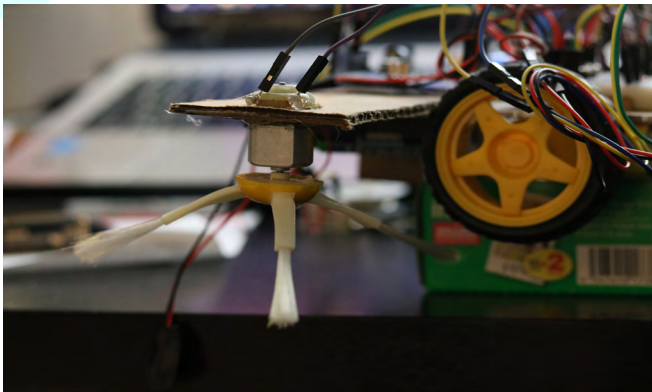
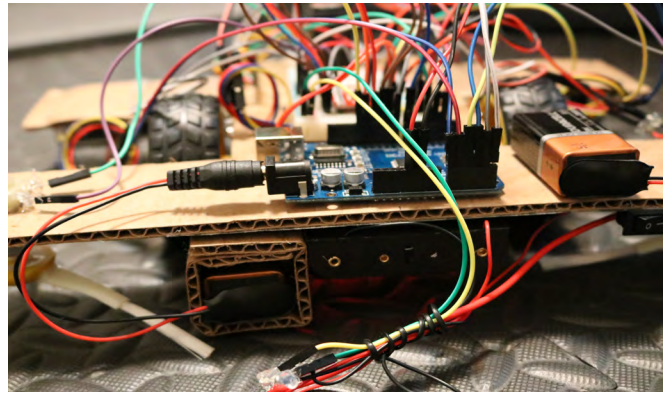
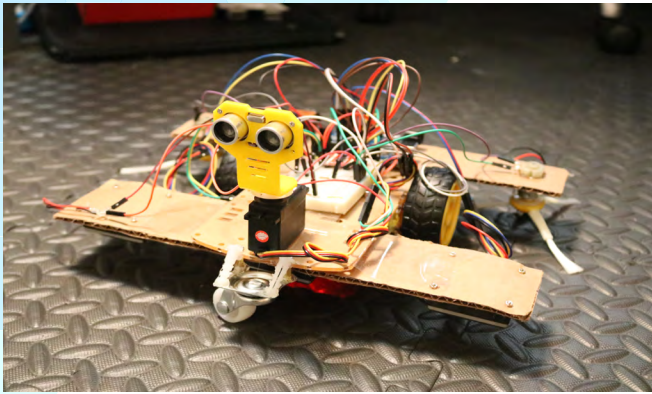
Java

Yelp API

Semantics3 API



AUTOMATED SWEEPING ROBOT ON AN ARDUINO



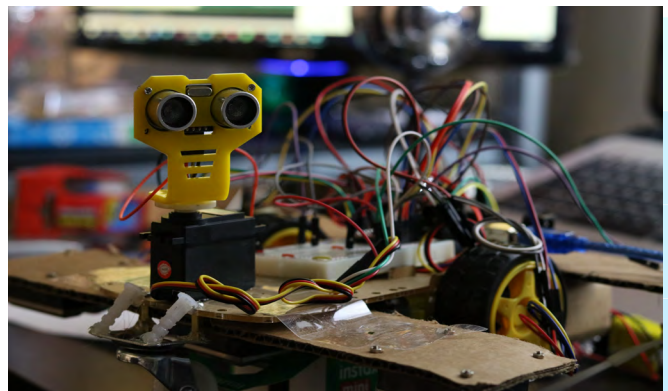
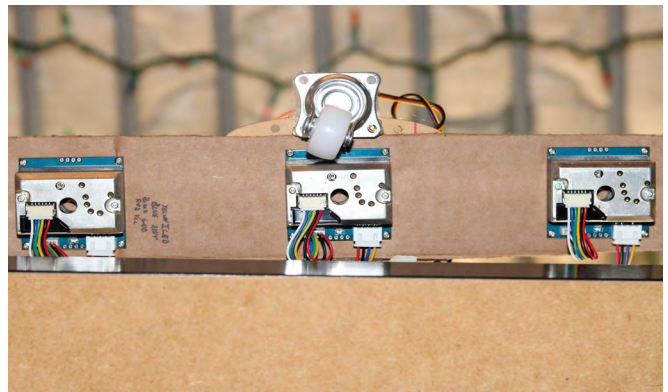
An Arduino was used as a microcontroller to simulate a sweeping robot, inspired by iRobot's Roombas. The project used an Arduino that was attached to multiple parts. DC motors were used for rotating sweepers and for the wheels to move forward, backward, or to rotate. A set of ultrasonic sensors were used on top of a periodic rotating servo motor to identify walls and to avoid crashing against walls, where the echo time would be converted to distance within the microcontroller. There were three dust sensors used in the front bottom of the robot to bias the direction of which the robot would take in order to climb up the gradient of dust. Finally the whole system was powered by three 5V batteries.

Platform

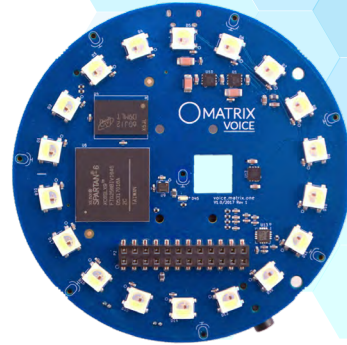
Arduino

Languages

C



VOICE-ACTIVATED SPEECH RECOGNITION ON A RASPBERRY PI



This project involved a technologist approach to engineering by using previously build APIs to build a system that performs a voice transcription over a speech once it was vocally activated by a particular individual with a specific key phrase. The system was built on a Raspberry Pi with a Voice Matrix which contains 8 microphones.

Languages and Libraries

Python

Google Text-to-Speech API

Snowboy




@ nicolas.s.shu@gmail.com

 linkedin.com/in/nicolasshu

 nicolasshu.com

 github.com/nicolasshu

 (718) 612-4856

