Gaussian Mixture Models and Markov Random Fields

Implementing Probabilistic Methods for Optimizing Floor Area Recognition Software

Project and Paper by Jacob Laufer and Weipeng Wang

GLOSSARY

Term	**Definition**
B	
Bandwidth Parameter	The parameter which controls the degree of smoothness for a kernel density function.
Boundless Distributions	A probability distribution in which a random variable has a positive probability of occurring at any location along the real number line.
C	
Cartesian Coordinate System	A two-dimensional coordinate system which defines points by pair of real numbers which correspond to the distances from the origin in perpendicular directions.
Communication	An equivalence relationship which indicates that two states are reachable from one another in a finite number of steps.
Cue	Information about data which upon interpretation aids in determining an unknown property about the data.
Cyclical	A property for graphical models in which the random variables can change back to their initial states after an initial state change.
D	
Data Clustering	A method of data organization in which groups of datapoints have similar properties.
Dataset	A collection of datapoints which are all the same datatype.
Datatype	A category of data that a computer can manipulate in the same manner.
Distance Formula	A formula which determines the proximity between two points in Euclidean space.

E

Energy Function	An expression of the total energy distribution for a system in any given state.
Entries	Any single value or index in a matrix.
Entropy	The degree of likelihood that a random variable will change from its current state.
Euclidean Space	A space of finite dimensions where the number of coordinates it takes to define a point is equal to the number of dimensions in the space.
Event	The occurrence of a probabilistic thing of interest.

F

Feature Vector	A vector defining the properties of a test image region that is fed into a support vector machine.
Feature Plane	A two-dimensional projection which realizes non-floor regions.
Final Mask	The binary data that defines the floor region in an image after making the minimum cut.
Floor plane	The area of a two-dimensional indoor image which corresponds to the floor in the three-dimensional projection.
Flow Network	A graph with edges that transmits a flow magnitude to its connecting nodes.

G

Generic Mask	The binary data which defines the floor region in an image after comparing each entry of a total score matrix to the score threshold.
Global Markov Property	The property which classifies a graphical model as a Markov Random Field.

H

Hard Partitioning	The process of assigning datapoints deterministically to the nearest cluster in Euclidean space.
Histogram Density Estimation	A probability density estimation which discretizes the range of possible random variable values into groups and counts the number of times samples of the data fall into each group.

Hyperplane	A subspace within Euclidean space which is one dimension less than that of the Euclidean space.
I	
Image Graph	A graphical representation of an image.
Iterative Approach	A method of feeding the output of an algorithm back to the input a specific number of times.
K	
Kernel Event	An event with a non-negative distribution which integrates to one.
M	
MAP Estimation	A parameter estimation method which determines the mode of the posterior distribution.
Minimum Energy Cut	A path which upon removal has a minimal effect on the energy function.
Mixture Models	A probabilistic model which combines subpopulations of known distributions into an overall population.
Multivariate GMM	A mixture model which combines multiple Gaussian subpopulation.
N	
Neighborhood Approach	A graphical approach for determining probability distributions which groups together events that occur in proximity to one another.
Neighboring Nodes	A pair of nodes in a graph which share an edge connection.
Non-Parametric Statistics	A branch of statistics which deals with defining probability distributions without using parameters.
O	
Overfitting	A graphical error which occurs when a function is fit to a data sample and the function does not represent the distribution of the population.
P	
Penalty Function	A component of an energy function which is inversely proportional to the entropy.

Penalty Value	The numerical value at a graphical node which equals the magnitude of penalty function at that node.
Population	The total set of observations from a dataset.
Posterior Distribution	A distribution which defines the probability of possible future observations.
Probability Density Map	An array which defines the probability of events occurring on a map.
Probability Score Matrix	A probability density map which represents the probabilities of a Gaussian Mixture Model over a two-dimensional test image.

R

Real Three-Dimensional Space	The three-dimension space that we live in.
RGB Color Value	The red, green and blue color concentrations which mix to create a color.

S

Scalar Distance	The magnitude of the difference between numerical random variable properties.
Separating Subset	A graphical nodal subset which upon removal disables communication between two nodes.
Subpopulations	A subset of a larger population.
Super Pixel	A grouping of pixels which exhibit similar properties.
Supervised Learning	A data mining tactic which considers user labeling of data to create a function.
Supremum	The least number of a variable which maximizes a given condition.
Support Vector Machine	A model which uses supervised learning to analyze data.

T

Test Image	An image which is an input of an algorithm.
Test Image Graph	A graphical representation of a test image.
Threshold Value	A value which separates numerical datapoints into groups depending on if each datapoints exceeds the value or not.

U

Undirectional Graphical Model	A graphical model where random variables may communicate with one another.
Unsupervised Clustering	A machine learning algorithm which clusters data without user input.

X

XY Coordinates	The dual horizontal and vertical coordinate pair that defines points in Cartesian coordinates.

INTRODUCTION

Pattern recognition is a fast-growing field with numerous applications across many industries. In the medical field, pattern recognition is the cornerstone of computer-aided diagnostic (CAD) systems while applications include face and speech recognition as well as image detection. The focus of this paper is to put forth the methods of creating floor detection image processing software which aims to take indoor images from a camera on a moving robot and distinguish the maneuverable floor area from non-maneuverable areas. The outputs of the software are readable by a basic computer processor such as a Raspberry Pi. In conjunction with the image processing software, depth estimation and LTL path planning yields a program which enables the navigation of autonomous robots in indoor environments.

An overview of the process which this paper aims to detail is as follows. The image processing software takes an image as an input and prepares it for analysis. The software analyzes the image for machine learning, position density and geometric cues to create a generic floor region. The generic floor region, along with color and positional values of the image's pixels, forms a Gaussian Mixture Model. A probability score matrix arises from evaluating the value of the Gaussian Mixture Model at each pixel in the image. A result of treating the test image pixels as random variables in a Markov Random Field is a penalty function which defines the gradient in adjacent pixel properties. Using the penalty function and the probability score matrix, a minimum cut algorithm separates the floor area in the image from the non-floor area. Upon repeatably forming a Gaussian Mixture Model, determining the penalty function, creating the energy function and employing the cutting algorithm, results converge to optimal values. A pictorial representation of the process flow is shown in the figure below.

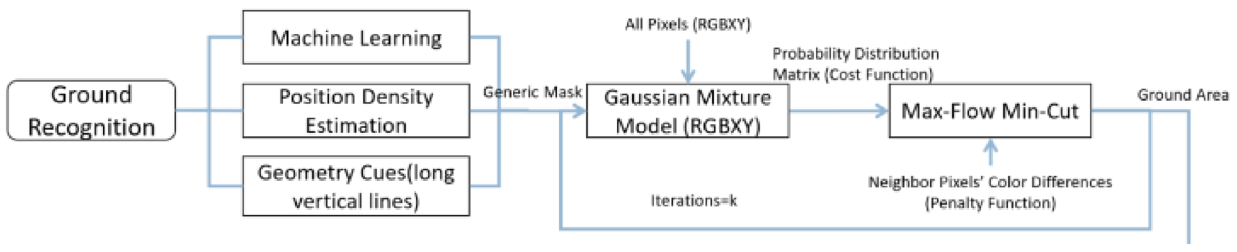
Ground Recognition
Machine Learning
Position Density Estimation
Geometry Cues(long vertical lines)
Generic Mask
All Pixels (RGBXY)
Gaussian Mixture Model (RGBXY)
Probability Distribution Matrix (Cost Function)
Max-Flow Min-Cut
Ground Area
Neighbor Pixels' Color Differences (Penalty Function)
Iterations=k

Figure 1: *A representation of the process flow for the floor detection image processing software*

In addition to providing insight into the workings of the floor detection software, this paper aims to transmit general knowledge about the use of probabilistic processes in image detection software. The creators of this software intend to provide information that will bring value to those in, or having interest in, image processing and random processes. The *Glossary* provides definitions, some of which are unique to this paper and some of which are ubiquitous in machine learning and pattern recognition, that appear in this paper. It is not the purpose of this paper to provide ground breaking research but rather to educate and put forwards a clear process to help others create vision software of their own. Python's scikit-learn allows simple implementation of many of the concepts this paper covers. For the source code, visit the following link: (https://github.com/Robertwwp/Ground-Recognition/blob/master/Ground_main).

GAUSSIAN MIXTURE MODEL

Mixture models are a composition of independently distributed **subpopulations**. A Gaussian Mixture Model (GMM) is a **population** consisting of the combination of a finite set of Gaussian distributions. A graphical representation of a **multivariate GMM** will not necessarily appear as a traditional Gaussian distribution. Assuming the peaks of the distributions lie sufficiently far apart, the graphical representation of a GMM will contain several peaks equal to the number of independent subpopulations. The equation below will compute the probability density of the GMM distribution.

$$F(x) = \sum_{i=1}^{n} w_i P_i$$

Equation 1: Where $F(x)$ is the probability density function of a mixture model, P_i is the distribution of any given subpopulation and w_i is the weight of the given distribution such that $\sum_{i=1}^{n} w_i = 1$.

A realization of a mixture distribution is deterministically obtainable from a set of subpopulations, however, obtaining the parameters of subpopulations from a mixture distribution requires statistical inferences. Partitioning a multivariate mixture distribution into individual subpopulations presents challenges since the independent subpopulations may have overlapping datapoints. When a multivariate mixture model contains subpopulations with **boundless distributions**, such as Gaussian distributions, datapoints from separate subpopulations will inevitably overlap, making it difficult to deterministically compute the parameters of the subpopulations. A statistical **iterative approach** provides a reasonable method of computing parameters, such as means and covariances, of subpopulations, which fully define the subpopulation distributions.

Unsupervised clustering is a common method of partitioning and determining parameters of subpopulations given a mixture model. K-means clustering is a common **data clustering** method. This method does not use a statistical approach but rather **hard partitions** datapoints to clusters. By the k-means method, the cluster which is closest in proximity to a given datapoint in **Euclidean space** is the cluster in which that datapoint will reside. Although k-means is a valid method for determining spherical Gaussian subpopulations from multivariant GMMs, it is not an effective method of partitioning a GMM that contains subpopulations with varying variances or a non-diagonal covariance matrixes. In contrast, the expectation maximum (EM) clustering method uses probabilistic values to partition datapoints and enables parameter computation for all types of Gaussian distributions.

The EM clustering method uses an iterative approach to estimate parameters of subpopulations from univariate GMMs as they converge asymptotically. Parameters from k-means clustering provides reasonable initial EM clustering values. Another parameter initialization method for EM clustering is to adopt Gaussian distributions with centroids about the model's origin. This is a less accurate initializing method and may require more iterations before convergence in comparison to initializing by k-means clustering. By employing the **distance formula**, the EM clustering method computes the probability of each datapoint belonging to each initialized distribution. The probabilistic results of the set of datapoints is the initial posterior distribution. The maximum a posterior (MAP) method then estimates centroidal and covariance parameters of the subpopulation using the initial **posterior distribution**. The equation below displays the **MAP estimation** for any given parameter.

$$\hat{\theta}_{MAP}(X) = \underset{\theta}{\arg max} f(\theta|X)$$

Equation 2: Where $\hat{\theta}_{MAP}(X)$ is the distribution of the parameter of interest and $f(\theta|X)$ is the posterior distribution of the parameter given a **dataset** X.

A more accurate representations of the subpopulations arise from the outputs of the MAP estimation along with a new posterior distribution. The MAP estimate can then enhance the accuracy of the subpopulation distributions once again. Upon iterative the MAP estimation and posterior distribution calculation steps, the parameters of the subpopulation distributions converge asymptotically. The maximum likelihood estimation (MLE) offers an alternate approach for choosing optimal parameters for subpopulation distributions in mixture models. The MLE is convenient since it calculates parameters of subpopulations directly from the dataset and therefore does not require an initialization step. Although convenient, one major drawback of the MLE is it tends to **overfit** distributions.

SCORING

A compilation of information from training data composes a dataset. To assign a probabilistic value to an input of a pattern recognition algorithm, information about the input and the dataset generates a score corresponding to the likelihood that the input is of a certain category. The dataset and input must be of the same **datatype** to effectively determine the score. In the case of image processing, the combination of information from a dataset and a **test image** forms a scoring matrix containing **entries** corresponding to scores at different regions of the image. The only allowable datatype for inputs to the floor detection software are images of indoor environments containing both floor and non-floor regions.

Learning with supervision trains the dataset to recognize properties of floor regions in test images. Dividing training images into patches of pixels which exhibit similar properties, or **super pixels**, is an effective way of preparing data from images for a dataset or for scoring [1]. Individually scoring and normalizing values of super pixels properties then weighting and summing the values provides a score corresponding to the probability that the super pixel is part of the **floor plane.** The super pixel properties come from analyzing machine learning, position kernel density and geometric edge detection **cues**. This summation is shown below.

$$S_T = w_M S_M + w_P S_P + w_G S_G$$

Equation 3: Where the total score, S_T, is equal to the sum of the three normalized scores, $S_M,\ S_P\ and\ S_G$, with weights $w_M, w_P\ and\ w_G$ from machine learning, position kernel density and geometric edge detection cues.

The dataset dictates the score due to machine learning. Properties of super pixels in training images and test images append to create **feature vectors** at each super pixel. Machine learning for floor detection compiles data in the form of a multi-dimensional feature vectors containing information on the color, brightness, hue, texture, position and shape of the super pixels from the image. A **support vector machine** classifies the feature vector data into one of two classes. In its most basic form a support vector machine finds the **hyperplane** which maximizes the margin between the two classes. The machine learning score of any give datapoint is proportional to the minimum distance from the hyperplane to the datapoint. In a **cartesian coordinate system**, the datapoint score from the support vector machine is the perpendicular distance between the datapoint and a line which separates the data sample into two classes. Datapoints on opposing sides of the hyperplane have scores of opposing signs so the score function increases.

The location of a super pixel in an **image graph** may affect the probability density of an **event** occurring. The kernel density estimation (KDE) method uses a **neighborhood approach** to construct a **probability density map** corresponding to the likelihood of an event occurring at different locations on the image graph. Discretizing images from the dataset and test image into equal allows for **kernel events**. The KDE method provides a **non-parametric** approach to create a smooth density function over an entire image graph. Distributing probability throughout regions surrounding event locations creates the smoothness of the density function. The peak of the distribution occurs at the event location and the probability decreases due to the event as the distance from the location to the event increases. The equation that determines the KDE distribution due to the combination of kernel events on an image graph is shown below.

$$\rho_{K(y)} = \sum_{i=1}^{N} K(\frac{y - x_i}{h})$$

Equation 4: Where $\rho_{K(y)}$ is the probability of detecting an event at a location, *y*, using the kernel density function, *K*; h is the bandwidth parameter and x_i is the location where an event occurs.

A **bandwidth parameter** defines the size of the discretized regions and the degree of smoothness of the KDE distribution. The area of a single discretization of a KDE image graph is like a bin from a **histogram density estimation**, however, the probability due to events sum differently. While a histogram sums probability in bins by uniformly distributing probability across the bin's range, a kernel event spreads probability across numerous discretization. A reasonable bandwidth parameter is obtainable by testing different values to find one that is sufficiently smooth without sacrificing accuracy. An optimal bandwidth parameter exists and is obtainable by minimizing the mean integrated square error and is left for future work.

As standard for all kernel events, the sum of the area under a Gaussian distributed kernel event is equal to one. Since Gaussian distributions are continuous and without bounds, and the test image restricts the bounds of the kernel density function, some probability is lost to area outside the bounds of the image. The probability loss will most severely impact the probability distribution of kernel events which occur along the sides of the image graph. This probability loss will not affect the final cutting algorithm nor the results of the software. The kernel density function for the Gaussian distribution [2][3] is shown below.

$$K(x;h) \propto \exp(-\frac{x^2}{2h^2})$$

Equation 5: Where $K(x;h)$ is the kernel density function with bandwidth parameter h at a location x.

Unlike the scores from machine learning and positional kernel density, the geometric edge detection score does not depend on the dataset. It is fair to assume that objects on the ground in the **real three-dimensional space** consist of edges which lie perpendicular to the ground. When the real three-dimensional space projects to a test image, the perpendicular lines project to vertical lines. The Canny Edge Detection method [4] detects and projects edges from a two-dimensional test image to a corresponding graph of equivalent size as noiseless white lines in black space. Canny Edge detection does not consider connectivity of edges and therefore will typically detect long edges with discontinuities.

A Probabilistic Hough Transform [5] creates long continuous edge lines from line segments by employing a voting algorithm. This transform graphs the horizontal distance to points on the Canny Edge detection graph and the angle of the edge relative to horizontal to a **feature plane**. Collinear edge segments are detectable on the feature plane and these segments transform back to the image plane as continuous. Averaging the y-coordinate midpoints of the long lines and composing a horizontal line through the average y-value creates a threshold line to filter lines whose midpoints lie above the threshold. Additionally, filtering out non-vertical lines due to the assumption that all object edges transform to vertical lines on an image leaves a set of lines with high probabilities of being object edge lines and containing bottom endpoints which are part of the floor plane. Connecting the bottom endpoints splits the test image into sections, the lower which has a high probability of being a part of the floor plane. The percentage of a given super pixel that resides on the floor plane is the normalization of the geometric edge score.

A constant **threshold value** evaluates the score of every pixel in a test image to create a **generic mask**. Experimenting with different threshold values over a wide array of training data will yield a reasonable value for detecting a parameter correctly. The floor detecting mask assigns binary values for floor and non-floor regions in the form of a matrix corresponding to pixels in the test image. The algorithm for creating the generic mask is shown below.

```
1. Set a threshold value, T.
2. For an i x j matrix, D ⟶ If $D_{ij} > T$ set $D_{ij} = 1$, else $D_{ij} = 0$ to form the generic mask.
```

Algorithm 1: Where i and j correspond to the number of rows and columns in a test image.

THE GMM MASK

From the generic mask, the **RGB color values** and **XY coordinates** of each pixel in a test image fully defines the properties necessary to construct a GMM which represents a topographical probabilistic map. The EM method assumes **RGBXY values** are Gaussian distributed i.i.d. random variables and evaluates the five random variables from the GMM at each pixel to create a **probability score matrix**. When generating probability score entries, analyzing pixels provides more accurate results then analyzing super pixels at a slight sacrifice for processing time. The pixel level approach ensures that the crucial floor and non-floor boundary regions have distinct probabilities. The probability matrix score depends only on the RGBXY pixel values which are relatively light computationally in comparison to the computationally intensive cue scoring.

To determine the GMM mask, a threshold value is set to determine binary mask entries. This is the same method for creating the generic mask. Entries in the probability score matrix which score above the threshold value are set to the binary floor plane value while those below the threshold value are set to the binary feature plane value. Through testing different threshold values, one emerges which creates an accurate GMM mask over a wide array of images. Optimizing the threshold values to yield optimal results is left for future work.

MARKOV RANDOM FIELDS

A Markov Random Field (MRF) is a set of random variables with a joint probability distribution in the form of an **undirectional graphical model**. As an undirectional graphical model, an MRF, $G = (N, E)$, represents random variables with *N* nodes and *E* connectivity edges. Given the **global Markov property**, two nodes in an MRF are conditionally independent of one another if a **separating subset** exists. Considering two **neighboring nodes** in an MRF, the random variables have a positive correlation and **communicate** with one another. The communication between neighboring nodes in an MRF allows for the random variables to be **cyclical** unlike in directional Bayesian networks where the random variables are acyclical. The communicative edge between two neighboring nodes is the **entropy** of the random variable. The edges are also comparable to an energy requirement which upon surpassing, changes states.

An MRF classification is appropriate for a graphical model that satisfies the global Markov property. The Markov property for Discrete Time Markov Chains declares that the future states of a Markov chain are strictly dependent on the present state of the chain and independent of prior states. Similarly, an MRF has a distinct number of dependencies which the global Markov property specifies. A random variable displays the global Markov property if, given three disjoint subsets of nodes A B and C, any node in subset A, is independent of any node in subset B, given a separating subset C. Upon removal of the separation subset, no path exists from node A to node B. Below is a mathematical expression of the global Markov property.

$$X_A \perp\!\!\!\perp X_B | X_C$$

Expression 1: Where X_i is a random variable of subset *i*.

Indexing test images and considering properties of each index creates MRFs and is a proven method of feature detection in image processing and pattern recognition [6]. Indexing an image by pixel creates the **supremum** number of indexes that maximizes accuracy. A **test image graph** comes from considering each pixel as a random variable and expressing a **penalty function** corresponding to the gradient between neighboring pixel properties. In floor detection, the properties which determines the penalty function are the RGB pixel color values. Given the difference in color value magnitude between two neighboring pixels, x and y, the penalty from x to y is equivalent to that from y to x. This equivalence statement proves that a test image graph is an undirectional graphical model. Furthermore, the test image graph is an MRF because the graph satisfies the global Markov property. To prove this, assign all random variables in test image graph to one of three subsets. Subset A contains any individual pixel of the test image, subset C contains all adjacent pixels to subset A and subset B contains every pixel in the image that is neither a part of subset A or subset C. The Global Markov property holds true since subset B is a separating subset of subsets A and C. Since a separating subset exists for any pixel in the test image graph, this graph is an MRF. The figure below further illustrates this argument.

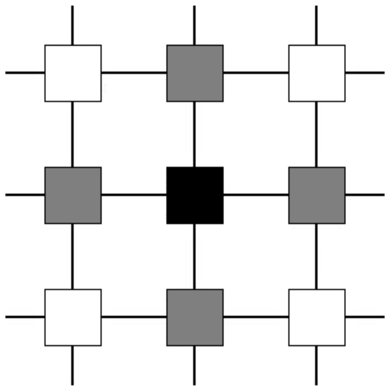

Figure 2: A test image graph where the grey nodes comprise a separating subset between the black and white nodes.

The **scalar distance** matrix assigns numerical values which represent changes in properties between adjacent pixels in a test image graph. The scalar distance is the only dependent variable of the penalty function. Although XY coordinates are often an important component of the scalar distance, when considering equally discretized test image graphs, considering them does not yield optimal results. The reason is that pixels in the test image graph contain no space between one another and are of equal size so the scalar distance due to the XY components are uniform. Negating XY coordinates is beneficial since they only dilute the RGB color values which adds gradient information to the scalar distance. The scalar distance equation is given below.

$$Z = \sqrt{\sum \Delta {X_i}^2}$$

Equation 6: Where *Z* is the scalar distance and the ΔX_i's are the random variables gradients between adjacent pixels.

The scalar distance directly provides a penalty function; however, manipulation of the scalar distance matrix entries are necessary to optimize results. In ground recognition software, taking the inverse of the product of the scalar distance and the exponential of the scalar distance provides reasonable results for the penalty function [7]. The product term serves to increases the variance in the penalty function which in turn increases the weight of the scalar distance term in the results. Inverting the product term creates an inverse relationship between the scalar distance and penalty magnitudes. This relationship causes high scalar distances to correspond to low **penalty values** which matches. This relationship matches the entropy of the probability score. The calculation of the penalty function from the scalar distance is shown below.

$$V = \frac{k}{Z * e^{2Z}}$$

Equation 7: Where *V* is a penalty function, *Z* is the scalar distance and *k* is a scaling factor.

Maximum Flow Minimum Cut

A maximum flow minimum cut theorem [8] makes a deterministic cut based on the **energy function** of an image. Incorporating a source and a sink into the test image graph MRF creates a **flow network**. An analogy of a statistical flow network exists to fluid flow example. Assuming the model of the flow network is at equilibrium, the flow rate out of the source will equal the flow rate into the sink. The purpose of the maximum flow minimum cut algorithm is to cut a path through the flow network that will maximize the flow network imbalance. Precisely, the cut will maximize the sum of the magnitude of outflow from the source to the graph and the magnitude of the inflow from the graph to the sink.

Both the source and sink connect to each pixel in the test image graph. Pixels which correspond to high probability score matrix entries will have a strong connection with the source and pixels corresponding to low probability score matrix entries will have a strong connection to the sink. Pixels with strong connections to either the source or the sink will influence the direction of the cut more so than pixels which have moderate connections to both the source and the sink. Pixels with strong connections will be more likely to reside on one section of the cut than the other if the penalty value does not dominate the probability score at that location. Namely, pixels with strong connections to the source are probable residents of the section that is maximizing source outflow while pixels with strong connections to the sink are probable residents of the section that is maximizing sink inflow.

The penalty function adds another dimension to the flow network by considering flow between neighboring nodes in the test image graph MRF. The penalty value of a pixel is the summation of the total penalty between that pixel and its adjacent pixels. Evaluating the probability score matrix entry and penalty value at each pixel yields the energy function. Manipulation of the penalty function and probability score distributions from a test image to create distributions with similar variances is left for future work. The energy function calculation is shown below.

$$E_{ij} = V_{ij} + P_{ij}$$

Equation 8: Where E_{ij} is the energy function at each pixel, V_{ij} is the sum of the penalty value at each pixel and P_{ij} is the probability score entry at each pixel.

Effectively estimating the **minimum energy cut** which optimizes the accuracy of the **final mask** is an iterative process. The cutting algorithm locates low energy areas within the flow network and in these areas tries different cuts. From the set of trial cuts, the one with the minimum energy requirement is chosen deterministically. The final mask from the cut is set to the generic mask and fed back into the scoring algorithm to create a new probability score matrix and penalty function which combine to form another energy function for cutting. Upon repeating the process, the results will begin converging as the number of iterations increase. The number of iterations the cutting algorithm performs will depend on balancing the accuracy requirements and the time it takes for an iteration.

Results

For the environments which the floor detection software analyzes iterating twice over the cutting algorithm provides the best results. The benefits do not outweigh the increase in processing time when iterating more than twice. In fact, in some cases, increasing the number iterations decreases the accuracy of the results due to over smoothing of the ground area. The figure below shows the image results.

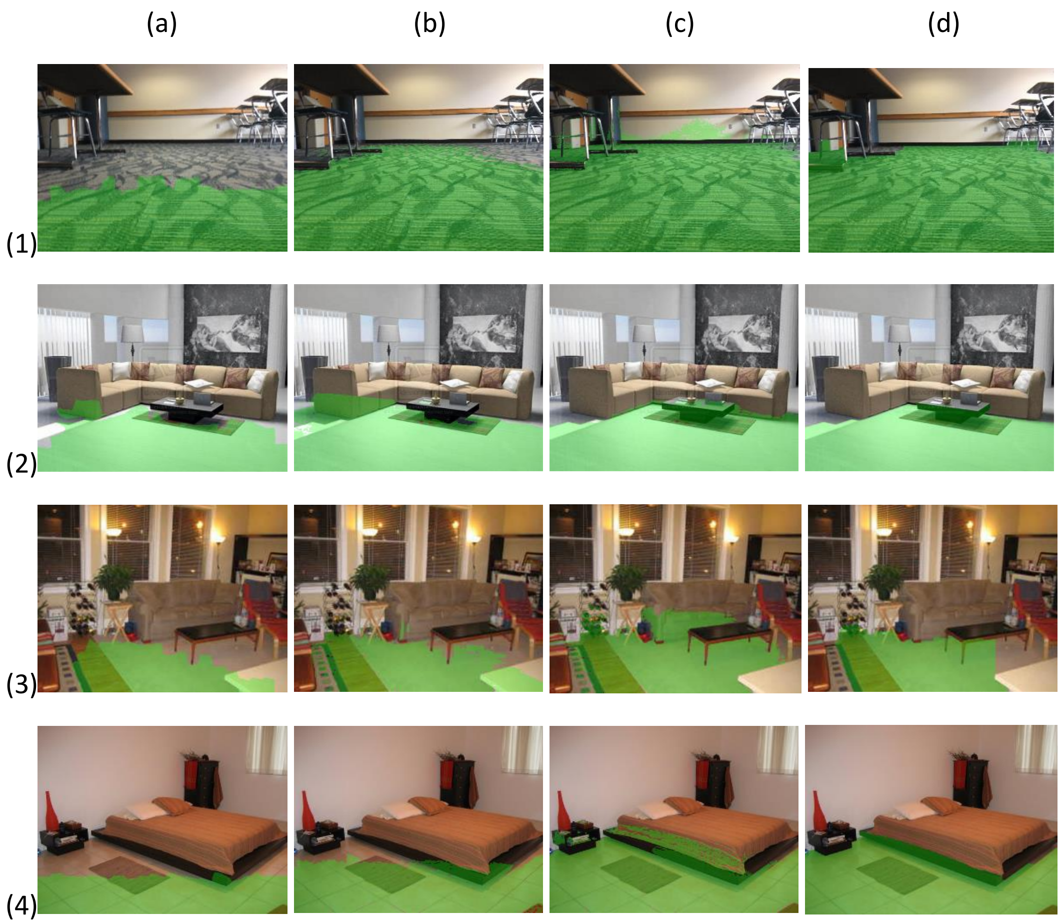

Figure 3: Where (a) is the first generic mask; (b) is the first GMM mask; (c) is the second GMM mask using inputs from the first maximum flow minimum cut iteration and (d) is the result after the second maximum flow minimum cut.

The table below summarizes the processing times necessary to achieve the results in the above figures. These processing times are unique to an Intel i7-6650U processor with 2.20GHz CPU.

	Image (1)	Image (2)	Image (3)	Image (4)
Generic Mask	15.58s	15.14s	15.49s	15.85s
Cutting (twice)	18.69s	16.31s	10.89s	14.04s
Total	34.27s	31.45s	26.38s	29.89s

Table 1: The processing times associated with the four results.

Conclusions

The methods put forwards in this paper provide accurate results for floor detection software. Considering two types of probabilistic models, GMMs and MRFs, and combining the results creates an energy function for deterministic cutting. By choosing a minimum energy cut from the energy function and iterating over the process a second time allows for accurate computer segmentation of two-dimensional test images representative of three-dimensional space. Extrapolating the abilities of this software to pattern recognition field, computers can perceive features in three-dimensions much like living organisms with eye sight and depth perception.

There are several areas for possible future work. Some of these areas are listed below

- Manipulating the penalty function and probability score distributions from a test image to create distributions with similar variances.
- Optimizing the bandwidth parameter for the kernel density distribution by minimizing the mean integrated square error.
- Determining optimal threshold values for creating the generic and GMM masks.
- Modifying the test image dataset to include images taken from the viewpoint of the robot car.
- Incorporating reinforcement learning without supervision by feeding results back into the training dataset.
- Increasing the efficiency of the program which will decrease the software run time.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süsstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274-2282, Nov. 2012.

[2] Rosenblatt, Murray. “Remarks on Some Nonparametric Estimates of a Density Function.” The Annals of Mathematical Statistics, vol. 27, no. 3, 1956, pp. 832–837.

[3] Parzen, Emanuel. “On Estimation of a Probability Density Function and Mode.” Ann. Math. Statist., vol. 33, no. 3, 1962, pp. 1065–1076.

[4] Canny, John. “A Computational Approach to Edge Detection.” Pattern Analysis and Machine Intelligence, IEEE Transactions On, PAMI-8, no. 6, 1986, pp. 679–698.

[5] Hough, P.V.C. Method and means for recognizing complex patterns, U.S. Patent 3,069,654, Dec. 18, 1962

[6] Guo, Chunzhao. “Robust Detection and Tracking in Challenging Scenarios Based on Markov Random Fields with Unsupervised Learning.” IEEE Transactions on, Vol. 13, No. 3, Sept. 2012.

[7] Aggarwal, Sanchit. “Estimating Floor Regions in Cluttered Indoor Scenes from First Person Camera View.” CVIT, International Institute of Information Technology, Hyderabad, India, Aug. 2014.

[8] G. B. Dantzig and D. R. Fulkerson, “On the Max-Flow MinCut Theorem of Networks,” in “Linear Inequalities,” Ann. Math. Studies, no. 38, Princeton, New Jersey, 1956.