Automatic Pixel Classification for Indoor Navigation

Michael Lazos

Advisor: Professor Pedro Felzenszwalb

Abstract

In this thesis we present the design and C++ implementation of a program for automatically identifying navigable terrain in an indoor environment efficiently. This system functions by first breaking up an input image into square tiles. Texture and HSV features are then computed for each of the tiles, and a non-linear support vector machine then classifies each of the tiles as either navigable or non-navigable. A smoothing process is then applied to this labeling to determine the most-likely partition of navigable and non-navigable terrain. This system is implemented efficiently enough to run in real-time for a low-power robotic platform to use for indoor navigation.

Background and Significance

This thesis builds upon the previous work of Hoiem, Efros and Hebert in "Recovering Surface Layout from an image." In that paper, it is demonstrated that the 3D layout of an image can be inferred by learning the appearance and orientation of various image characteristics. A system is proposed which is able to classify pixels as belonging to a specific geometric class, i.e. pixels representing ground, sky, left-facing planes, or right-facing planes. To generate this classification

this system first performs multiple segmentations of an image, with a differing desired number of segments. Each of these segments is then coalesced by similarity and classified individually using boosted decision trees with various color, location, texture, and perspective features to perform the classification. The system performs quite well, but there are still some questions that are unanswered. Some features that may improve the accuracy of the classification, such as gradient orientations were not used, while some other features were included which may be contributing little discriminatory power. It is important to understand why specific features are effective at this task in order to build simpler, more effective classification systems.

As proposed in Hoiem et al. these classification systems could be applied to the problem of vision-based navigation. The most obvious application that has become more prominent recently is vision systems in autonomous cars. For this application, gaining semantic information from a single image can be invaluable. Such a system was used in 2005 to complete the DARPA Grand Challenge (see Dahlkamp). That algorithm worked by using a laser range finder, paired with a long-range camera. The range finder was used to identify flat navigable terrain very close to the vehicle. A Gaussian mixture model was then continually updated online and used to score pixels outside the identified drivable area in the camera's image. From this a 2-D cost map of the surrounding terrain is generated and an optimal path through the map was chosen. This can allow a driving system to make better decisions about which paths to take in order to arrive at a specific destination.

Similar classifiers can also be used for small robots to navigate indoors. With the rise of household robotics, there will most likely be many different types of

robots moving through our homes in the future. Instead of the traditional ultrasonic range finders, a simplified classifier could be used provide the navigation information necessary for advanced tasks. This application is closest to the main problem addressed in this thesis.

The problem of identifying navigable terrain indoors is possible using the system described in Hoiem et al. However, the major drawback of this system is the run time. This system doesn't run quickly enough to be used on a low-power robotic platform. To address this drawback, we first simplify the problem to labeling pixels as either navigable or non-navigable. As result of this, a simpler and more efficient binary classifier can now be applied to discern navigable terrain. To improve runtime further and to simplify the algorithm, we use a simple tiling of an image as a substitute for the multiple segmentation and coalescing strategy. Our algorithm then computes a small number of very discriminative features on each tile. The classification is then performed with a non-linear SVM from the LIBSVM C++ package.

Constraining the navigation problem to an indoor environment brings with it some benefits that cannot be ignored. Indoor environments are easier to discriminate with texture, since floor is often a different texture or color than wall for instance. With the addition of carpeting and other floor textures, the problem becomes greatly simplified, since many carpet textures can be learned quickly by simple histogram of oriented gradients (HOG) features (see Dalal). Furthermore, indoor environments are quite repetitive in their structure. Most rooms have dull-colored non-reflective walls on the side, carpet or tile on the floor, and a ceiling that

can be similar to the walls, but most likely differs from the composition of the floor. This structure is ripe for a learning system with simple features meant to exploit this structure.

Although there are numerous benefits to performing classification on the characteristics of an indoor environment, there are some unique issues that arise with the problem of classifying indoor images. One issue with indoor flooring that is often overlooked is the amount of reflectance that tiled floor can have. This can allow for images where light sources can appear to originate on the floor, which can be confusing for any classification system looking for navigable terrain. This problem does not occur as often in an outdoor environment, since most outdoor environments have surfaces that diffuse light in all directions as opposed to uniformly reflecting it in a single direction.

Another problem which can arise indoors is a large a presence of people. Since individuals obviously wear different clothing patterns, it is difficult to allow a classifier to generalize in the presence of so many patterns that are profoundly different from learned patterns. This could lead to a higher false positive rate in practice that would be detrimental to any platform using this system because it would now mistakenly label a portion of a person's body as navigable terrain. Ideally, a more conservative classifier should be selected that will safely label unknown patterns as non-navigable.

Additionally in indoor environments, especially in engineering buildings, there can be a lack of color variety between the tiled floor and walls. This can make it difficult to devise features that can discriminate between a very bright beige tile

and white walls and ceilings. A possible solution to this is to use combinations of features in the description vector, or perhaps incorporate context surrounding a tile to determine whether nearby tiles are navigable or non-navigable, because this could provide the extra information needed for tiles that are near the margin between the two classes.

A final problem that can occur in indoor environments is change due to outdoor debris bought in by individuals. On pavement or grass outdoors walking on a piece pavement many times doesn't change its appearance. When learning a model for an indoor environment however, the properties of the floor can change drastically from dirt, salt, or water brought in by individuals; this requires a model invariant to changes in reflectance or small changes in texture.

Feature Selection

To determine which features were to be used, a dataset of about 100 images from the target environment of Barus and Holley building at Brown University was used for training and testing. The proposed model was trained on 50 of the images and tested on the other 50. To get an initial estimate of the discriminative potential of these features, a linear SVM was trained using 39 of the original 82 features from Hoiem computed on each of the tiles from the training images. Table 1 below shows each feature and their corresponding weight value.

Feature	Weight
Mean Absolute Oriented Filter Response	-0.0178
	-0.0184
	-0.0188
	-0.0177
	-0.0148
	-0.0126
	-0.0118
	-0.0127
	-0.0151
	-0.0177
	-0.0187
	-0.0185
Edginess – Mean of all OFRs	-0.0162
Index of largest OFR	-0.0017
Dominance of Largest Response	-0.0007
Red Mean	0.8115
Green Mean	0.1369
Blue Mean	0.0937
Hue Mean	0.8176
Saturation Mean	0.4305
Value Mean	0.4987
Y Location Mean	0.2595
X Location Mean	-0.0464
Hue Histogram – 5 Bins	-0.1809
	0.4946
	-0.3247
	-0.0912
	0.1022
Hue Histogram Entropy	-1.6317
Saturation Histogram – 3 Bins	-0.1149
	0.42
	-0.3052
Saturation Histogram Entropy	-2.3618
10^{th} Percentile Y	-6.6848
90^{th} Percentile Y	0.3855
10^{th} Percentile X	0.2886
90^{th} Percentile X	-0.3069
Edginess Center Y	0.3489
Edginess Center X	0.4118

Table 1 – Feature Weights from Linear SVM

Not surprisingly, location is a very important feature, since intuitively the navigable terrain is toward the bottom of an image, especially in an indoor environment like Barus and Holley. An intriguing result is that Hue-Saturation-Value means carried more weight than the corresponding Red-Green-Blue means. To then determine more finely which features will be used, different combinations of the 42 features above were tested to determine the final selection of features. The results are shown below.

Feature Set	Accuracy
Including all features (except location)	0.7715
Excluding dominance of largest response	0.7715
Excluding DLFR, largest FR	0.7549
Excluding DLFR, largest FR, edginess	0.7549
Excluding all filter features and edginess	0.7221
Excluding all filter features, edginess, hue histogram	0.6766
Excluding all filter features, edginess, edge centers	0.7221
Excluding all filter features, hue, saturation, value means	0.6309
Including hue, saturation histograms	0.6288
Including hue, saturation histograms, hsv means	0.7221

Table 2 – Feature Set Accuracy Results

In addition to the features proposed from Hoiem et al., other features that we thought would be effective for this problem were evaluated in combination with the best features that were determined in the procedure outlined above. These additional features included texture features such as gray-scale variance, hue gradient orientation histograms, as well as standard gray-scale gradient orientation histograms.

Features

Location

The most intuitive of the features chosen is location. As is obvious to any outside of observer, ground, and hence navigable terrain tends to be toward the bottom of an image. For the purposes of this system, location is normalized by position in the image, and weighted less than the other features with a value of 1/1000. If the feature is weighted more highly, The SVM tends to overuse it and the classifier simply partitions the image horizontally, classifying any tile below the partition as navigable, and anything above it s non-navigable. This is not very useful, since there are areas of the ground that are non-navigable. This highlighted a weakness in the gathered dataset, since there were actually many images down hallways, such as the one shown below, which actually can be classified by a horizontal partition. To compensate for this weakness the dataset was expanded with more images that were taken into the corner of walls or rooms, and with objects in the center of the image shown in the subsequent image.

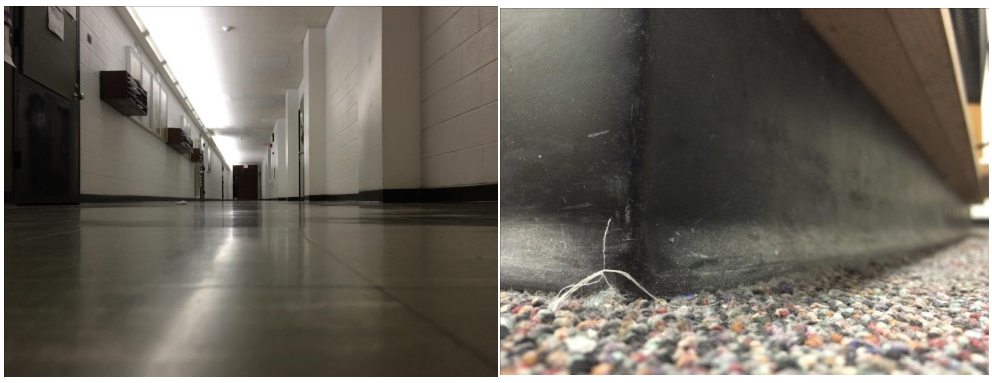


Figure 1 – Original dataset image (left) could be separated by horizontal partition, while new dataset image (right) must be learned with features other than location.

Hue, Saturation, Value Means

Intuitively color would seem to be a useful feature, since the floor is often a different color than the ceiling and the walls of the building. This is true much of the time, but on some floors, reflection from light sources can drown out the color of the floor and prevent accurate classification of that section of the image. Using an HSV color model slightly addresses this problem, since value corresponds to the brightness of a pixel. This, in conjunction with location, can often discern the bright reflective tiles from the bright lights on the ceiling; this is shown below.

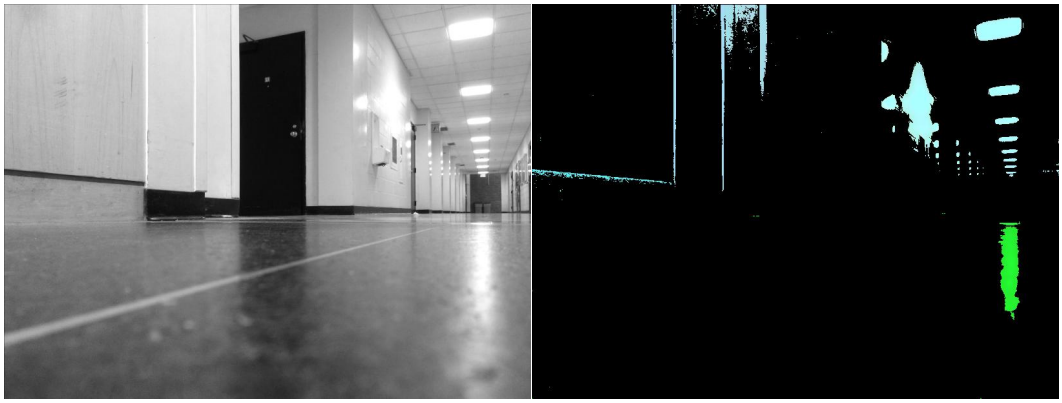


Figure 2 – Brightness image (left) with the same image with dim pixels filtered out (right) and bright ground tiles highlighted (green)

Hue was quite good at discriminating between the walls and the floor, since in most cases in the building where the dataset was collected, especially in the images with carpet, the floor had a different hue value than the walls. This is shown in the image below.

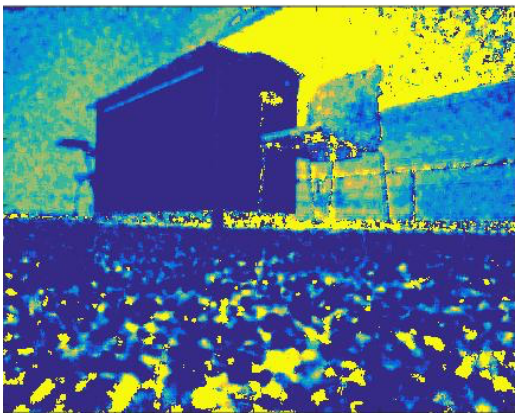


Figure 3 – Hue image demonstrating difference between carpet and wall

Saturation contributed in cases where hue was very similar throughout the image. For instance, in the below image, the brushed wood wall has a similar hue to the floor, but since the wall is obviously a different color than the floor, they must have different saturation levels. For comparison, the original image, the hue image and the saturation image are shown together.

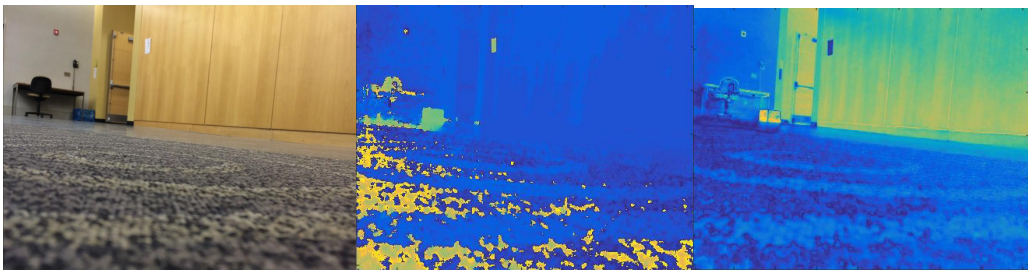


Figure 4 – RGB image (left) Hue image (middle) and Saturation image (right)

From these images, it is clear that hue and saturation together present a powerful combination for identifying navigable terrain.

Hue and Saturation Distributions

In addition to the means, the distribution of hue and saturation were also used to discriminate between navigable terrain and rest of the image. This feature allows the system to measure patterns in carpet to some degree, since carpets often have a color pattern associated with them, while plain characteristics such walls and ceilings often have a single color, so their distributions tend to center around a single value. To approximate the distributions, a hue histogram with 5 bins was constructed. An example demonstrating the effectiveness of these features is shown below in figure 5.

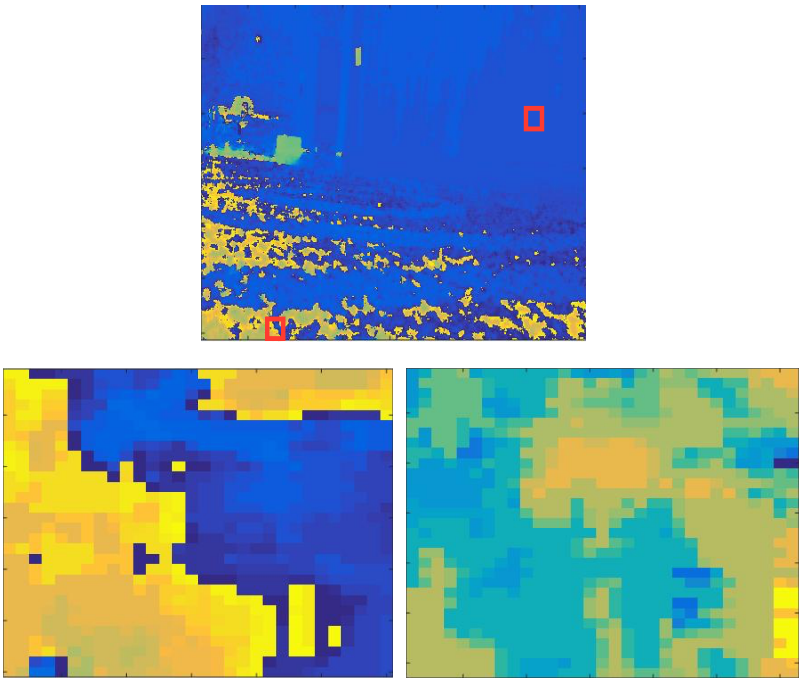


Figure 5 – Hue image (top) with corresponding navigable tile (left) and non-navigable tile (right)

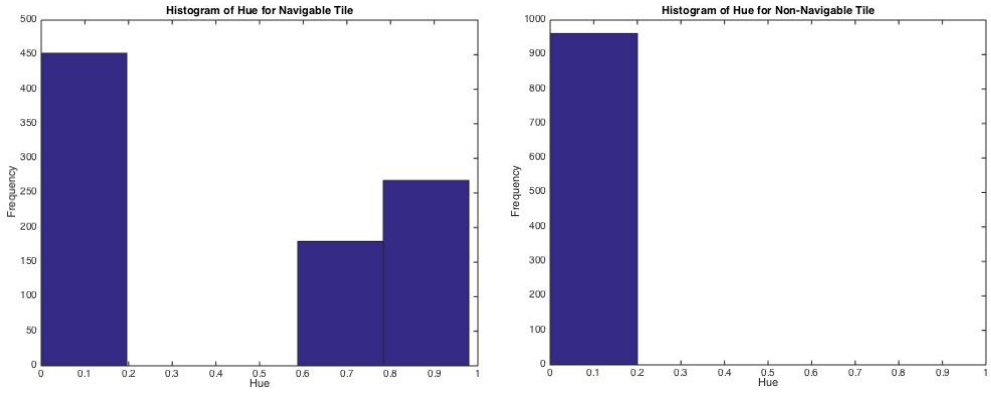


Figure 6 – Corresponding tile hue histograms

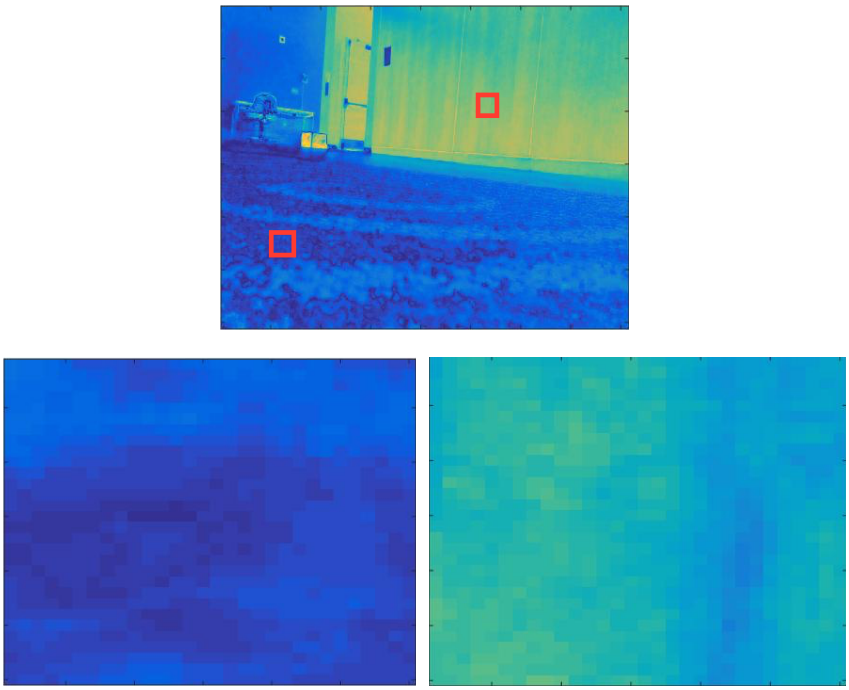


Figure 7 – Saturation image (top) with corresponding navigable tile (left) and non-navigable tile (right)

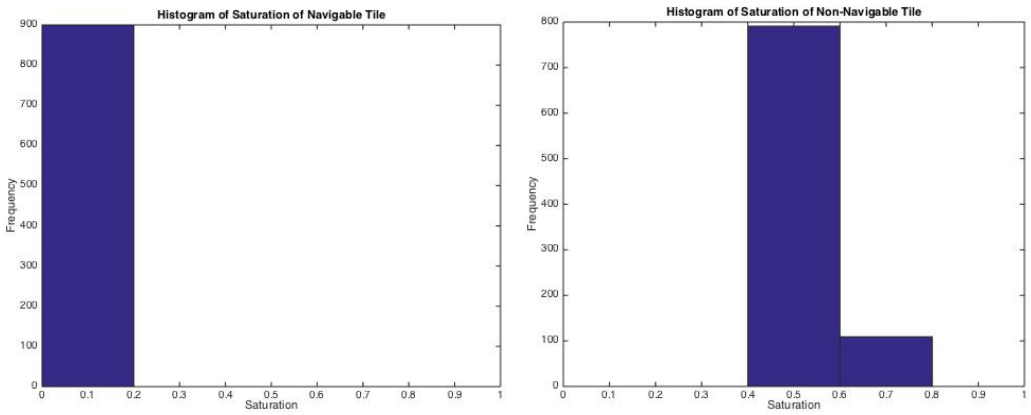


Figure 8 – Corresponding tile saturation histograms

To aid in learning different textures, the entropy of the histograms is also used. Using the entropy allows the classifier to learn different textures because distributions that are more uniform have a higher entropy value from distributions that center around specific values.

Gray-Scale Variance

The variance of the brightness of the pixels in a tile can also provide valuable texture information. This feature is simply the sample variance of the pixel brightness within the tile. What makes the variance useful is that it can discern tiles that are uniform such as walls, ceilings, or metal objects, from more textured tiles like floor and carpet, both of which indicate navigable terrain. An example image is shown below. The original image is shown on the left, while the image of the variance of each tile is shown on the right. Higher variance is indicated by a lighter blue, and in a single case yellow. This image is a great example showcasing the use

of this feature since the carpet is very discernable from the walls in the variance image.

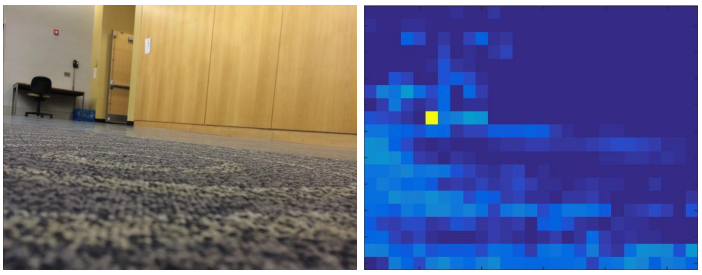


Figure 9 – Original image (left) Tile variance image (right)

Histogram of Oriented Gradients (HOG)

Another useful feature for detecting texture is a histogram of gradient orientations (see Dalal). This idea of this feature is compelling because textures in images are at their core sharp gradients in some repeated pattern. The use of this feature overlaps with some of the previous features in that it is able to detect texture differences between materials that traditionally compose navigable terrain such as tile flooring and carpet, from surfaces that are non-navigable, such as walls and ceilings. This makes sense on an intuitive level, since the gradient orientation for carpet would vary considerably over a tile due to its rough texture. The gradient of the wall however, would be more likely to be uniform and have fewer changes in gradient orientation. This is partially confirmed by the orientation image below. In this image, the orientation varies from -180 degrees (blue) to +180 degrees (green). The pattern in the rug can be easily discerned from the patterns on the wall. A more unexpected result is that the wall does appear quite rough in the image, although

upon closer examination, the wall's pattern differs from the carpet's pattern in a meaningful way, which is shown in the resulting histograms.

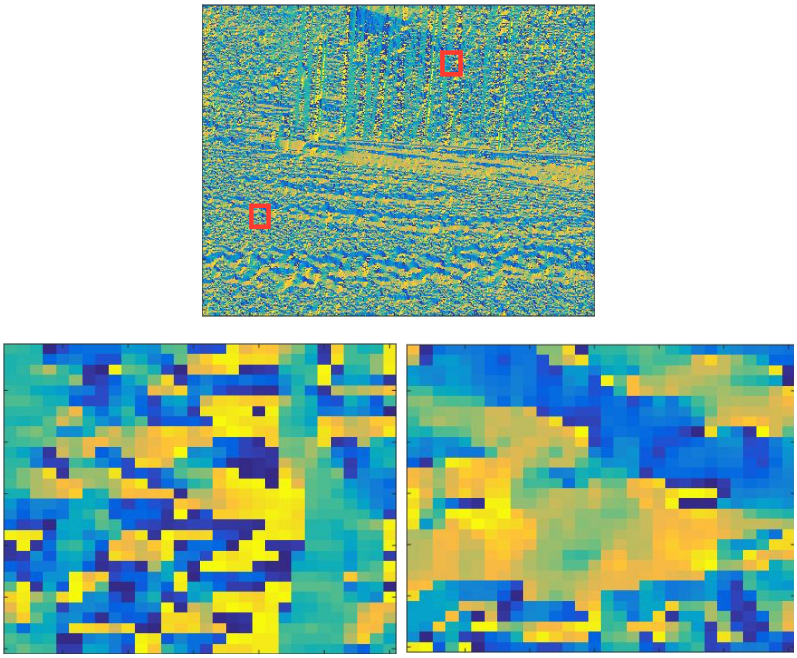


Figure 10 – Gradient image (top) with corresponding navigable tile (left) and non-navigable tile (right)

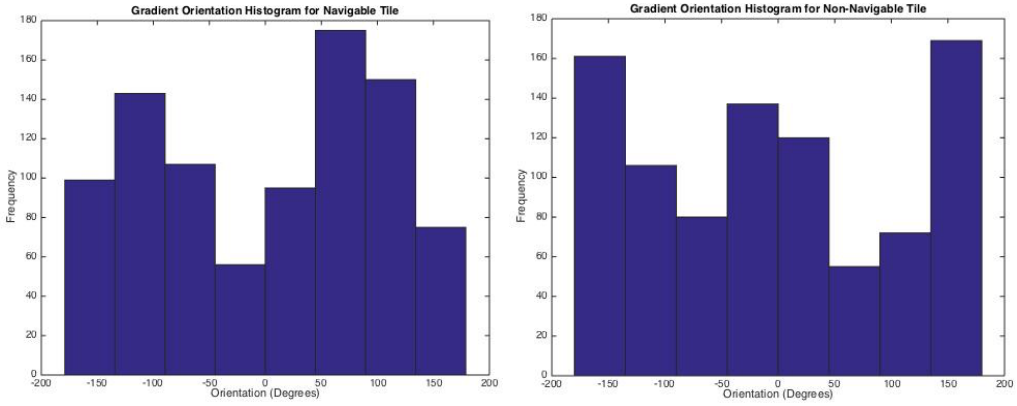


Figure 11 – Corresponding gradient orientation histogram

An additional caveat in the histogram features used in the final system is that the votes for each orientation are weighted by the gradient magnitude. That is, instead of each orientation contributing a value of 1 to its respective bin in the histogram, it contributes a value proportional to the gradient magnitude. This

allows the histogram to discriminate between even more textures, due to the added information content of the gradient magnitudes.

Parameter/Kernel Selection

To determine which kernel should be used for the SVM in this classification problem, an initial slack variable (cost value) of 3 was used (this was the default value in previous experiments), and then a linear SVM and a non-linear SVM with three different kernel functions were tested. The accuracy results are shown below:

Kernel	Accuracy
Gaussian	86%
3rd Degree Poly	85%
5th Degree Poly	89%
3rd Degree Poly (C=100)	93%

Table 2 – Kernel Accuracy Results

The first was the radial basis function (RBF) kernel. This initially looked to be the most promising, since it had performed well in earlier experiments with the system described in Hoiem. Also since this function essentially places a Gaussian at sample points, it can be interpreted as a similarity measure to the training points. This is a compelling idea because the navigable terrain being classified in training images should be somewhat similar to the navigable terrain that is encountered in the testing images. However, due to the similarity between some wall tiles and reflective beige floor tiles, it makes intuitive sense that other kernels that allow features to interact in different combinations may perform better. This conclusion

is validated by the performance of the other two kernels, the 3rd and 5th degree polynomials which both performed more effectively, with the 3rd degree polynomial performing best overall. The cost parameter was tested with values of 0, 50, and 100, with each kernel. The accuracy of the Gaussian did not change significantly, while the 5th degree polynomial SVM did not finish training in the allotted time of 3 days (determined by the time needed to iterate experiments) if the cost parameter was greater than 10.

Selection of Tile Size

The main source of efficiency in this algorithm is the simplification of the problem by using local neighborhoods of the image, in the form of a square tiling. This tiling is performed over the whole image, and features are computed in each of these tiles independently. This allows an image to be processed with only a single pass through the image, and also opens the opportunity for parallel processing, although this capability isn't used in the current implementation since the target robotic platform uses a single-core processor. To determine the optimal tile size, an implementation of the system described in Hoiem was used with superpixel regions consisting of the tiling used by our algorithm on a general dataset provided by the authors. In the first iteration of the experiment, a coarse range of tile sizes was chosen. The result of this experiment is shown in figure 12.

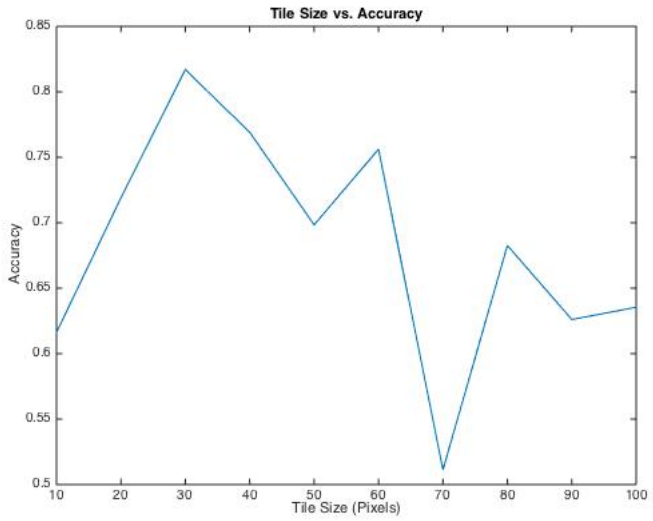


Figure 12 – Initial coarse-grained tile size experimental results

After a clear peak was found at 30x30, a finer range from 20x20 to 35x35 was tested, to see if 30x30 was indeed the highest value in that neighborhood of values. The result of this experiment is shown in Figure 2.

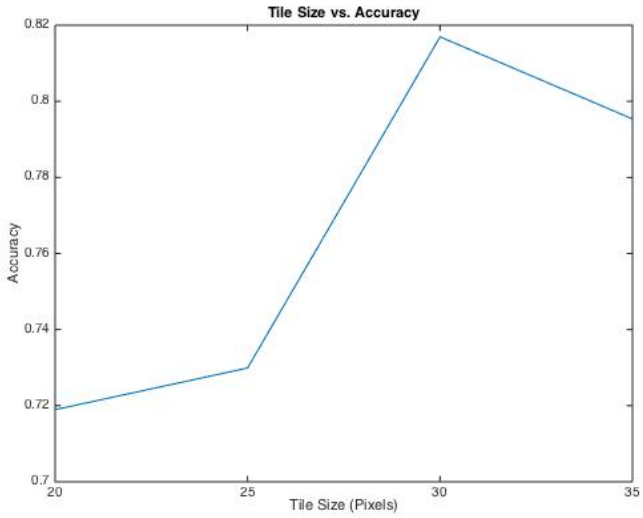


Figure 13 – Fine-grained tile size experimental results

The outcome of the experiment shows a clear peak at 30x30. This confirms the intuition that there is an optimal neighborhood that provides a context where the features chosen can more accurately discriminate between regions.

After the image is tiled, the features described in the previous section are then calculated. These feature vectors are then classified by the SVM.

Column Smoothing

After the initial classification, each column of tiles is smoothed to determine a clean partition (i.e. the navigability horizon) between the navigable portions of the column and non-navigable portion. This can be understood by observing that in most images of indoor environments, there is some horizon between navigable and non-navigable terrain, this is marked below in red:

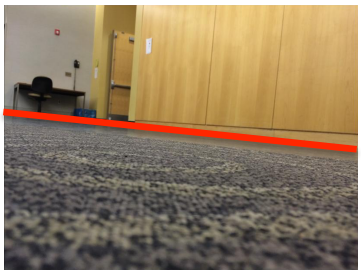


Figure 14 – Image with horizon (red)

To find this horizon the likelihood of each possible horizon is determined for each column, and the most likely horizon is selected. The first step in this process is estimating the probability of the SVM score given the class of a tile. In this context,

the score is simply the distance from the tile's feature vector to the linear separator of the SVM. To estimate these distributions, ten images were taken from the test data, their tile scores were calculated, and then the ground-truth labels were used to create two ten-bin histogram of the number of scores that fell in bins of size .5 on the interval [-5,5]. If a tile is navigable, its score is tallied in the navigability histogram, and if a tile is non-navigable, its score is tallied in the non-navigability histogram. If a score falls above or below this interval, its label is counted toward the highest or lowest bin respectively. The probability of a score conditioned on class is then estimated by normalizing each histogram. This measure is then used to calculate the probability of each horizon which is defined as the product of the probability that each tile above that horizon is non-navigable and the probability that each tile below the given horizon is navigable. Each of those probabilities is calculated in turn as the product of the probabilities of the score of each tile above the proposed horizon conditioned on that tile being non-navigable with the probabilities of the score of each tile below the horizon conditioned on those tiles being navigable. This is illustrated in figure 15 and Eq. 1 below.

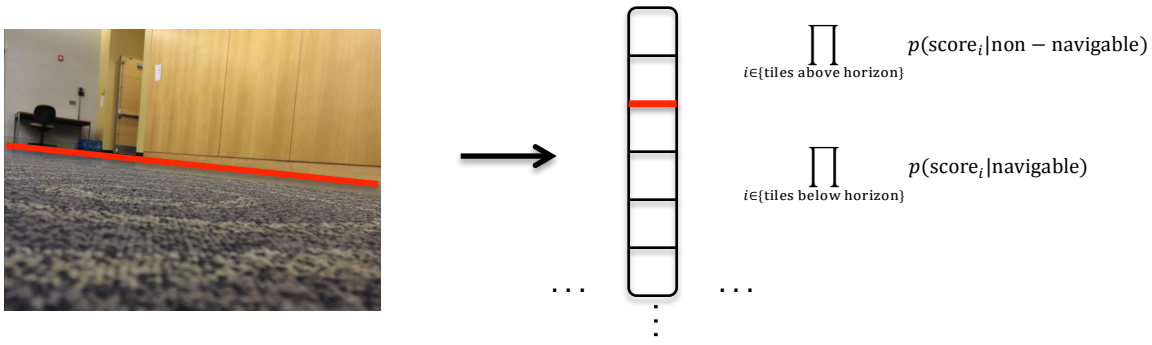


Figure 15 – Image with horizon (left) with sample column horizon calculation (right)

$$p_{horizon} = \prod_{i \in \{\text{tiles above horizon}\}} p(\text{score}_i | \text{non} - \text{navigable}) * \prod_{j \in \{\text{tiles below horizon}\}} p(\text{score}_j | \text{navigable})$$

Equation 1 – Probability of a horizon

This can be calculated in linear time by building a table of the probabilities of each sub-column, accessing the table to compute the probability of each horizon, and then performing a final iteration to determine the most likely horizon.

Hidden Markov Model Row Smoothing

After the initial stage of column smoothing, a stage of row smoothing across these new columns is performed in order to make the horizon more regular. This smoothing is performed using a Hidden Markov Model. In the HMM, the true value of each column's horizon is treated as a latent variable with 27 possible states where each state maps to a horizon level while the observables are the optimal horizon determined in the previous stage of the algorithm. The transition matrix chosen assumes a Gaussian distribution on the state transitions, i.e. a given column's is most likely to transition to a nearby horizon, with a probability of switching to another horizon equal to the probability of a discrete normal distribution centered at the current horizon value. This is shown in Eq. 2 and figure 16 below.

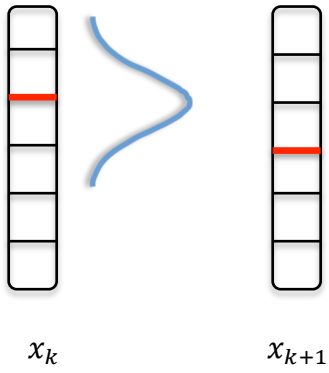
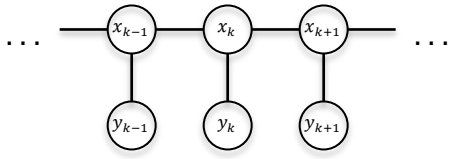


Figure 16 – Illustration of the relationship between subsequent columns denoted by discrete Gaussian transition probabilities

$$p(x_{k+1}|x_k) \approx \frac{e^{\frac{-(x_{k+1}-x_k)^2}{2*.9^2}}}{.9\sqrt{2\pi} * Z_k} \text{ where } Z_k \text{ is for normalization}$$

Equation 2 – Conditional probability relationship between subsequent columns

Additionally, it is assumed each latent variable x_k (the true horizon) is corrupted by Gaussian noise, so that the observed column state y_k (the calculated optimal horizon) is a sample from a Gaussian distribution whose mean is the value of the latent state, and whose variance is .9 (determined by experiment). Once the observables are calculated by accumulating all positive examples in a column, the most likely sequence of latent states can be calculated using dynamic programming (see Forney). The graph structure of the model is shown below



$$y_k \sim N(x_k, 4.15)$$

Figure 17 – Structure of HMM (top) with relationship between observables and latent variables (bottom)

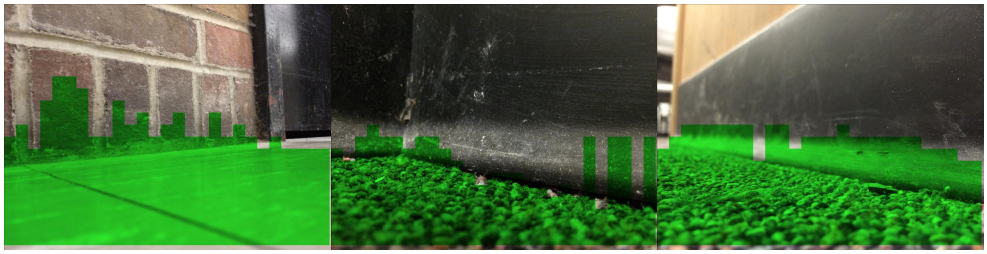
The variance of the noise was determined by sweeping values from approximately 3.0 to 5, in intervals of .05. 4.15 provided optimal performance. In future work, with more available data, this parameter could be estimated from the labeled ground horizons.

Evaluating Algorithm Performance

To train and test the final version of the algorithm a dataset of 177 images was collected and labeled for navigable terrain. These images consist of corridors, walls, corners, furniture, classrooms and the lobby from about 3mm above the floor to properly model the perspective of the robot that will be running the algorithm. For labeling, binary images were created from an annotating program that displayed the navigable terrain as white, with the rest of the image black. These ground truth images were then tiled into 30x30 neighborhoods using the same methods as the algorithm. Then, tiles that are greater than or equal to 90% navigable are classified as navigable, and tiles that are less than 90% navigable are classified as non-navigable. Finally, to evaluate the performance of the algorithm, half of the dataset was used as a training set, while the other half was used as a test set, with the results being compared to the ground truth binary images.

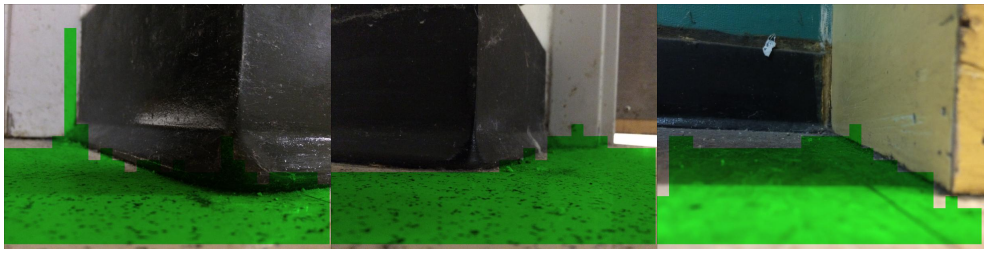
Discussion

Overall, the performance of the algorithm was very good on this dataset; it was about 95% accurate on the 87 test images. A sampling of 18 classified images is shown below.

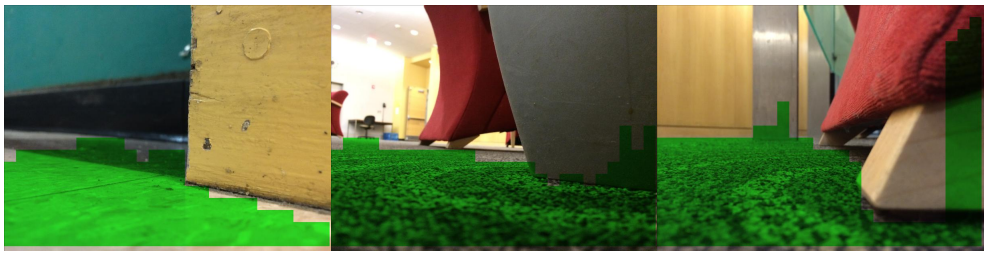


(a) (b) (c)

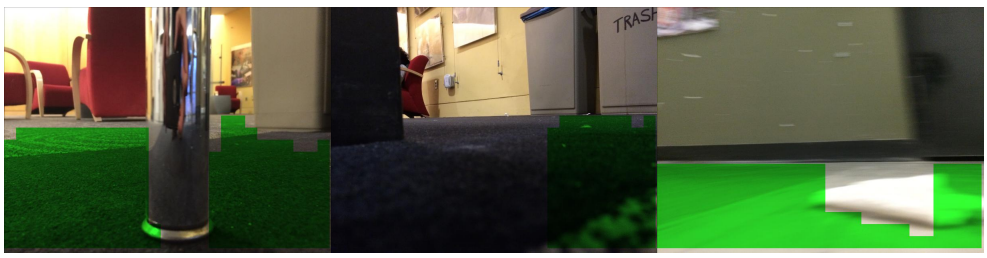
(d) (e) (f)



(g) (h) (i)



(k) (l) (m)



(n) (o) (p)

(q) (r) (s)

Figure 18 – Image Results

These cases show that there are examples where texture may not be enough, or that more training data is needed. For instance in image (a) where the brick wall is partially classified as navigable, and in image (c) the wall's black edging with floor is occasionally very dusty, giving it a light tint which can be misclassified as lighter flooring. However there are images where the classifier performs perfectly such as (f) and (h). Despite some more complex shapes such as the corners of walls and some furniture pieces, the classifier is still able to identify navigable terrain. The blurry image (p) is a great example of how the texture features are able to discern plain white at a low point in the image from beige tile.

As for the run-time, the algorithm can process images at ~2fps which is fast enough for real-time navigation, and significantly faster than the minutes it would take to process an image to determine the full surface layout. A breakdown of algorithm time for each phase of the algorithm is shown below:

Stage of Program	Run Time (ms)
Feature Computations	121 ms
SVM Classification	330 ms
Post-Processing	< 1 ms

Table 3 – Run time of program stages

These results confirm that it is possible to classify navigable terrain in an indoor environment very accurately, in real-time. Future work on this classifier would be to improve the efficiency of the SVM classifier, test how well it generalizes to other indoor environments, and perhaps test it on outdoor environments as well, although this will likely require some amount of new training data because of the different textures present in such varied environments.

Acknowledgements

I wish to thank Professor Pedro Felzenszwalb for his advice and guidance throughout this project as well as Professor Chad Jenkins for his discussion and suggestions for this thesis. Finally, I would like to thank my family for their encouragement and enthusiasm for my pursuit of this project.

References

Dahlkamp, H., A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. "Self-supervised Monocular Road Detection in Desert Terrain." *Proceedings of Robotics: Science and Systems* (2006): n. pag. Web.

Dalal, N., and B. Triggs. "Histogram of Oriented Gradients for Human Detection." *Computer Vision and Pattern Recognition* 1 (2005): 886-93. Print.

Forney, G.d. "The Viterbi Algorithm." *Proceedings of the IEEE* 61.3 (1973): 268-78. Web.

Hoiem, Derek, Alexei E. Efros, and Martial Hebert. "Recovering Surface Layout from an Image." *International Journal of Computer Vision* 75.1 (2007): 151-72. *Robotics Institute: Recovering Surface Layout from an Image*. International Journal of Computer Vision. Web. 14 Apr. 2015.