International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:12, 2009

An efficient obstacle detection algorithm using colour and texture

Chau Nguyen Viet, and Ian Marshall

Abstract—This paper presents a new classification algorithm using colour and texture for obstacle detection. Colour information is computationally cheap to learn and process. However in many cases, colour alone does not provide enough information for classification. Texture information can improve classification performance but usually comes at an expensive cost. Our algorithm uses both colour and texture features but texture is only needed when colour is unreliable. During the training stage, texture features are learned specifically to improve the performance of a colour classifier. The algorithm learns a set of simple texture features and only the most effective features are used in the classification stage. Therefore our algorithm has a very good classification rate while is still fast enough to run on a limited computer platform. The proposed algorithm was tested with a challenging outdoor image set. Test result shows the algorithm achieves a much better trade-off between classification performance and efficiency than a typical colour classifier.

Keywords—Colour, texture, classification, obstacle detection.

I. Introduction

Obstacle detection is an essential task of an autonomous navigation system. For example, an autonomous mobile robot needs to detect and locate any presence of obstacles in the surrounding environment to operate safely. For this purpose, visual sensing devices are often used. This work concerns a vision system which has a single camera i.e. monocular vision system. This visual system has a tasks of detecting obstacles and recognising safe traversable areas in images taken from a camera mounted on-board a mobile robot. Such system often consists of two parts: a supervised learning algorithm and a classifier. The learning algorithm learns the model of traversable paths from a training image set, in which patches of pixels of paths and obstacles are labelled by human. The classifier then compares a patch of pixels in new images to the learned model and decides if the patch is a path or an obstacle. Our problem is a two classes classification problem [1], a pixel patch can either be path or non-path. In this context, obstacle detection and path recognition are identical task.

There are many visual features that can be used to model the appearance of a traversable path. Colour is a popular choice of feature for path recognition [2]–[5]. The main reason for it's popularity is that colour feature is a local property hence computationally cheap and easy to learn. However a classifier uses colour only has several shortcomings. First, a real world path's colour appearance is almost always non homogeneous. Second, a path and obstacles on it may have similar colours. Most seriously, colour of a surface changes dramatically when the illumination condition changes, i.e. the colour constancy problem [3]. The relationship between colour appearance and illumination is non-linear and very difficult to model. Other visual features such as contrast, texture, geometrical structures, shapes have been used in object recognition, image classification, and image retrieval system [6]–[8]. But because of their computationally expensive cost and application specific tuning, these features have not been adopted widely in the mobile robotics domain. Only recently, there are a number of attempts to use texture in addition to colour for path recognition in an on-board vision system [4], [9], [10]. Unlike colour, texture feature the arrangement between pixels and is at least in theory invariant to illumination. However in the mentioned works, the benefit from adding texture features was either not reported or unsubstantial.

C. Nguyen Viet, and I. Marshall are with the Computing Department, Lancaster University, UK. Email: c.nguyenviet@lancaster.ac.uk

In this paper, new method is proposed to combine colour and texture for the task of obstacle detection. During training, instead of learning texture for a separate classifier, the texture features are leaned specifically to enhance the performance of a colour classifier. Instead of learning texture models from all training samples, the method learns texture of the image samples that the colour classifier fails to recognise. The learning algorithm only selects the best texture features for classification from a subset of training samples. Both colour and texture features are modeled using the histogram method. In classification stage, the colour feature is used first. The texture features are consulted only when the colour classifier can not produce a decision with high confidence hence our algorithm is very efficient. In most cases, the proposed algorithm uses colour feature only so the average run-time to classify a sample is very fast. All textural features are simple enough for real time running on an on-board system.

The next section describes how colour and texture features are learned independently for classification. Section III explains how texture features are learned specifically to improve a colour classifier and the combined features classifier. Evaluation test result on a set of real out-door images is reported in section IV. Section V summarises the paper.

II. Learning colour and texture features using histograms

A. Colour feature

Colour is a popular choice of feature for image segmentation and object recognition. Colour information is readily available as input from a colour camera so no extra processing is required. Colour of a pixel is represented as a vector in the Red , Green, Blue (RGB) colour space or Hue, Saturation, Value (HSV). Real world surfaces often have more than a single

(a) A typical example of the path (b) An image sample taken in the shade

(c) A path with a different appearance (d) Path and obstacles with similar appearance

Fig. 1. Examples of the image set

colour e.g. in Fig 1 the path has parts that are either white, pink, gray, or black. To learn the colour appearance of a real world surface, the histogram method is often used. A colour histogram contains a fixed number of bins and divides the colour space into regular regions. Each bin of a histogram of a pixel patch is the count of pixels in the patch whose colour falls in the region defined by that bin. A normalised histogram H approximates the distribution of colours in an image patch P, each bin in H is calculated by

$$H_i = n_i/N \tag{1}$$

where n_i is the number of pixels whose colour falls into the region defined by bin i and N is the total number of pixels in P. When histogram is used for classification, during the training stage, a set of model histograms of a class is learned. In classification stage, a sample histogram is compared with the model histograms. The sample's class is assigned with the best matched model's class. For histogram matching, there are different ways of measuring the distance or similarity between two histograms: general distance measurements such as Euclid, and Manhattan distance, or statistical test such as the Chi square test. The histogram intersection method was used which is computationally efficient. The intersection method calculates the sum of overlapped areas between two histograms s and m:

$$M(s,m) = \sum_i \min(s_i, m_i) \tag{2}$$

in which s and m are model and sample histogram, s_i is the value of bin i in s. No complex operation but an adding and comparison is needed. The greater and closer to one the intersection value is the more similar the two histograms are.

For obstacle detection in an autonomous navigation application, the training algorithm often can only learn the model of known traversable paths. Because the presence and type of obstacles are unknown prior to operation, them can not be modelled in the training stage. During classification stage, image samples from an on-board camera are compared to the learned path's models. A sample is classified as path if it is similar enough to a path's model. In this case, if $M(s,m)$ is greater than a constant threshold then the sample patch is labelled as path. The threshold value is crucial and often determined by the experimental method. If it is set too high, the classifier will label a lot of traversable samples as obstacles. If it is set too low, obstacles will be misclassified as path. The curves in Fig 3 demonstrates the effect of varying this threshold. As the threshold decreases, both the path detection rate and obstacle error rate increases.

B. *Texture features*

Although there is no formal definition of texture, the texture of a visual surface can be thought of as spatial distribution of intensity/colour variations. There are several approaches to capture texture in digital images: statistical methods [11], filter banks [12], [13], local pixel distribution [14]. A few comparative studies on the differences between these approaches have been conducted [14]–[16]. Most of these studies concluded that the performance of a texture approach is very application dependent. There is no clear best texture capturing method and non of the modern approaches outperforms the simple co-occurrence matrix developed three decades ago [11]. However, a conclusion can be made that the performance of a particular approach depends on the complexity of its implementation. For instance, in a texton filter based approach to texture classification [17], the performance depends on the number of textons is used. In the case of co-occurrence matrix, the performance depends on the number of gray levels, displacement sets are used.

In this study, texture is used to improve the performance of a colour based classifier but with a consideration of computational speed and memory requirement. A good trade-off between performance improvement and added complexity is desired. For this reason, two texture learning methods were chosen: the Local Binary Pattern (LBP) method [14] and a statistical method using 8 histograms of the intensity difference between two pixels at 2, 4, 6, and 8 pixels apart at horizontal and vertical directions. These two methods are significantly less complex than other texture methods.

The LBP contains local intensity pattern of a pixel. The LBP of pixel n with intensity x is defined as:

$$lbp(n) = \sum_{i=1..8} 2^{bi(x-x_i)}$$

$$bi(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases}$$

where $x_{1..8}$ are the intensity values of the 8 pixels neighbouring n. A LBP has a range from 1 to 512, each value represents an unique local pattern of a pixel. To accommodate this feature, the local contrast was used which is the total sum of differences between a pixel and its intermediate neighbours. To capture texture information at larger scales, the DIFFX, DIFFY histograms described in [14] were used. The variation in intensity of pixels at four different intervals and two directions were captured. The histogram method described previously is again used to model texture features. A histogram of 512 bins is used for the LBP. The LBP is a categorical variable so there can not be any bin grouping. The local contrast and each of the DFFX, DIFFY histograms has 16 bins. The matching and classification method for texture histograms is identical to colour histograms. Each histogram model has an associated threshold. Because more than one histogram is needed to describe an image patch, when two image patches are matched, a mechanism is needed to combine each histogram matching value. For simplicity, the average sum of each histogram matching value is used.

III. Combining colour and texture features for classification

A. Partitioning training set using colour histograms

From initial tests of the colour based classifier, it was found that the average performance was relatively good. The colour information is rich enough in most cases. Classification errors arise when the sample images were taken under bad illumination condition i.e. under or over exposure in Fig 1(b) , when there were obstacles with very similar colour appearance to the path (even to a human eye) in Fig 1(d). It was noted that extracting texture features is several magnitude more complex than extracting colour feature. From these observations, algorithm was built that uses colour as the primary feature and only consider texture feature in special cases where it is difficult to use colour to discriminate. Instead of having one threshold for each model colour histogram, two thresholds are learned. One high threshold is used to determine if a sample is matched with a model, one low threshold used to determine if it is not. If the intersection value falls in between these two thresholds, the texture feature is used. The samples that fall in this regions are the difficult cases for the colour classifier.

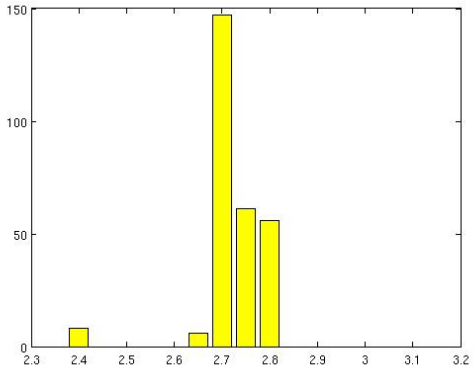
The colour model learning procedure works as follow. A set of colour histograms of a path is extracted from the training images. This set usually has a large number of histograms and a smaller set of representative histograms is needed. To find this set, a clustering algorithm is applied to the training set and histograms belong to the same clusters were merged into one model histogram. The popular K-mean clustering [1] was adopted. K-mean clustering is a simple iterative algorithm that minimises the sum of distances between data points in a cluster according to a distance measurement. In K-mean clustering, the user has to specify the desired number of clusters k. Equation 2 is used as the distance measurement. Because the intersection value $M(s,m)$ measures the similarity between two histograms, to calculate the distance d : $d(s,m) = 1 - M(s,m)$. After the clustering has finished, histograms in each cluster are merged by:

$$H_m(x) = \sum_{i=1..n} H_i(x)/n$$

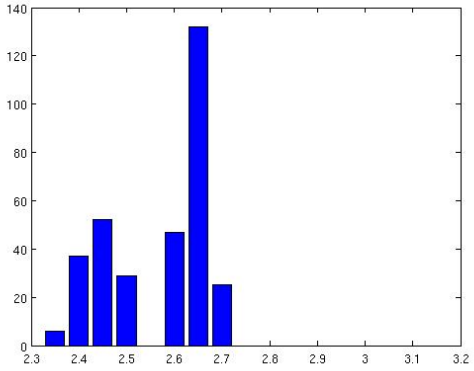
where $H_{1..n}$ are the histograms in the cluster and $H_i(x)$ is the value of bin x in H_i.

Next, the non-path histograms were classified into k groups using the k representative colour histograms. The training set was divided into k subsets, each contains both path and non-path samples. Within a subset, all samples have similar colour appearance since they are all matched to one colour histogram. Each of the k representative colour histograms is associated with two thresholds high and low. The thresholds value are controlled globally by setting the percentage of path and non-path histograms in a training subset would match the representative colour histogram. For example, the high threshold can be set so a model colour histogram matches 95% of the path histograms belonging to the same cluster with the model, and the low threshold so that model will match all path histograms but also match a larger number of non-path histograms.

Fig 2 demonstrates how a colour model histogram separates the data. Fig 2(a) is a histogram of intersection values (homogeneity measurements) from path data points in a cluster to the model histogram defining the cluster. The data points were taken from real images. Comparing this histogram to the histogram in Fig 2(b), we can see that the majority of path data points have higher homogeneity measurement than non-path data points as expected. All points with homogeneity greater than 2.7 are path. There are some path points that have homogeneity value as low as 2.4, which is lower than most of non-path points. These points should be treated as outliers. The overlapping area between two histograms is relatively small, however the biggest column in each histogram is next to each other. This means that if one threshold is used to separate the two classes, two problems occur. One, 100% correct classification is impossible. The best threshold to use in this case is 2.65 which will give a overall classification rate of about 85%. Second, the performance of the classifier is very sensitive to the threshold value. When using the best



(a) Histogram of homogeneities from path points



(b) Histogram of homogeneities from non-path points

Fig. 2. Histogram of distances to a path model

threshold learned from the training data, even if it is very close to the best threshold for the testing data, the performance on training and unknown testing data can be very different. A good solution is to use two thresholds to separates the data points into three parts: path, non-path and uncertain. A good choice of these two thresholds in this example is 2.6 and 2.75, all points below 2.6 are non-path , all points above 2.75 are paths. Texture features is used to classify uncertain points in between these thresholds.

B. Texture features selection and learning

In the proposed algorithm, each training subset has a model features from one texture method. During training stage both the LBP with contrast histograms and eight absolute difference histograms are learned. However texture histograms from only one method is saved and used in the classification stage. All texture features were not used for efficiency reason. The LBP with contrast and absolute difference histograms method both have similar computation and memory requirement with the former method slightly more demanding. The learning algorithm selects the texture method with the better classification error rate. First, it searches for the minimum threshold value that would recognise a fixed percentage of the path samples. The error rates of the two texture methods with the found thresholds are compared and the texture method with lower error rate is selected. Obviously one could use two thresholds for texture features to divide the data set the same way as colour and use more features to classify the uncertain area. From experiences, using both methods for classification with an additional mechanism to resolve potential conflicts does not improve classification rate of while doubling up memory usage and computation load. By training more features on a small data set , the chance of the classifier becomes over-fitting to the training set decreases.

C. Classification using colour and texture

During classification stage, the algorithm maps a pixel patch of unknown class label to either path or non-path. A patch is labelled as path if it's colour histogram is matched with any of the k colour histograms saved from the learning stage using the high threshold. If non of the model histograms matches with the sample, the algorithm iterates through the k models again. This time the intersection value between the sample and the model colour histogram is compared with the low threshold. If a matched model is found, the sample's texture histogram is compared with the model's texture histograms. A pseudo-code of the algorithm is listed below

for $i = 1$ to k **do**
  if $M(s_c, m_{c,i}) > high\_threshold_i$ **then**
    s is path
    return
  end if
end for
for $i = 1$ to k **do**
  if $M(s_c, m_{c,i}) > low\_threshold_i$ AND $M(s_t, mt, i) > texture\_threshold_i$ **then**
    s is path
    return
  end if
end for
s is non path.

s_c s_t are the sample's colour and texture histogram, $m_{c,i}$ and $m_{c,t}$ are colour and texture histogram of the ith model.

IV. Experimental Test And Evaluation

A set of outdoor images were collected from a campus path at Lancaster University, Fig 1. shows some of the samples. The pictures were taken at different times and illumination conditions. Many potential obstacles were tried , also borders between the path and another type of terrains e.g. grass. Even though those pictures were taken from the same surface type, there are enough variations to make the model learning and classification task challenging. For example, Fig 1(a) is an image taken in normal bright condition, while Fig 1(b) is a sample taken in a shade and the path looks very different, Fig 1(d) is an instance where the path and obstacles have very similar visual appearances. One hundred images were labelled manually to path and obstacle regions. The image set was divided into training and testing set. The training set has 20 images and testing set has 80 images. The reason why the

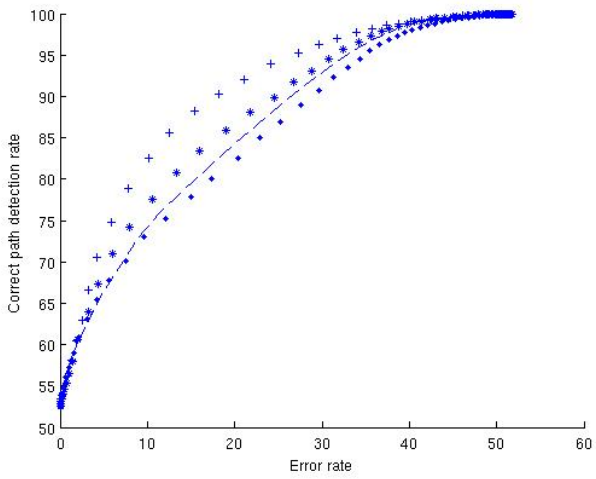


Fig. 3. ROC curves of colour classifiers. '.' k = 5 , '-' k = 10 , '*' k = 20 , '+' k = 100

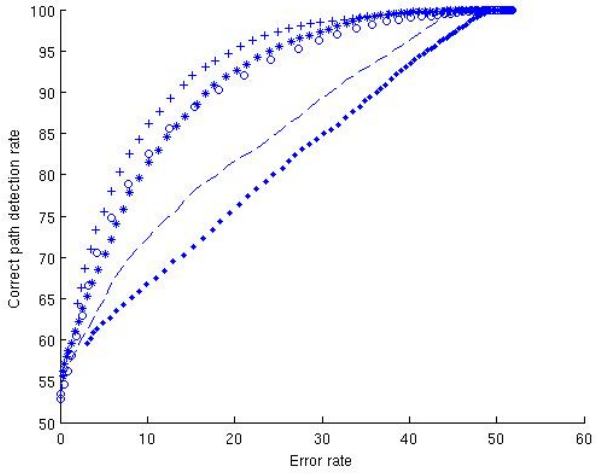


Fig. 4. ROC curves of colour and texture combined classifiers. '.' k = 5 , '-' k = 10 , '*' k = 20 , '+' k = 100 , 'o' colour only k=100

training set has much less samples than the testing is because in a robot application, the number of training images is often limited since collecting and labelling images takes time. The ability to generalise is essential. The algorithm should be tested with a small set of training samples.

A. Experiment results and discussion

The classification algorithm using colour only were tested first. Each image was divided into non overlapping window of 20x20 pixels. Several different values of k, number of models, were tried. Fig 3 shows the ROC (Relative Operating Characteristic) curves of colour classifier with $k = 5, 10, 20, 100$. For a two classes classification problem, the ROC curve describes the performance better than an average error curve. The figure shows that using a large number of model histograms k yields better performance. To detect $90\%r$ of the path patches, a classifier using five models wrongly classifies almost 30% of non-path patches, using 20 models the error rate is around 20% and when using 100 models the error rate is only 15%. At this level of path recognition, the error rate and the number of models used have an almost linear relationship.

TABLE I
RUN-TIME OF CLASSIFIERS IN SECONDS

	Avg	Max	Std
C only $k = 100$	0.16	0.21	0. 05
C and T $k = 20$	0.07	0.19	0.09

The ROC curves of the classifier using both colour and texture is showed in Fig 4. The same set of k values was used as the colour classifier. Similar to the colour only method, the performance of this classifier depends on the number of models used. However the rate of change in performance against the number of models is different. When $k = 5$ the performance of the combined features classifier is actually slightly worse than colour only classifier. One plausible explanation is that with a small number of k, each training subset contains largely dissimilar texture patches. As our algorithm uses one texture model for each training subset, clustered by the colour histograms, the texture classifier becomes unreliable and can make the combined classifier worse. This explanation is backed up with the significant improvement of performance when using bigger k. With $k = 20$ the colour and texture classifiers ROC curve is almost identical with the ROC curve of colour only classifier with $k = 100$ and is much better than colour only with the same k. Increasing k from 20 to 100 further improves the classifier but with a smaller gain in performance than from 10 to 20.

The run-time of colour classifier with 100 models and colour and texture classifier with 10 models was tested on a Pentium III PC. Table I contains the classifiers' run-time statistics of processing a 320x240 image over a set of 80 images. In this configuration, the classifier using both colour and texture features has better run-time on average and in the worst case scenario than classifier using colour only. Another test was done for the combined features classifier on a 200Mhz Gumstix mini computer [18]. On this platform the average time to process an image is 0.22 seconds or 5Hz. Implementing the colour classifier with 100 learned models on the Gumstix was not possible due to memory restriction.

B. Algorithm sensitivity over training set

One factor that often affect the performance of a classification algorithm is the quality of the training set. A learning algorithm can only model the variations that are presented in the training set. Ideally, the training set should be a representative subset of the real population. In practice this is very difficult to achieve and one can only try to use the biggest available training set. However even with a large training set, it is not known if it is a good representation of the real population. Because of the difficulty in selecting a good training set, one important aspect of a classification algorithm is how sensitive it is to different training sets. A good supervised algorithm should learn all variations in the

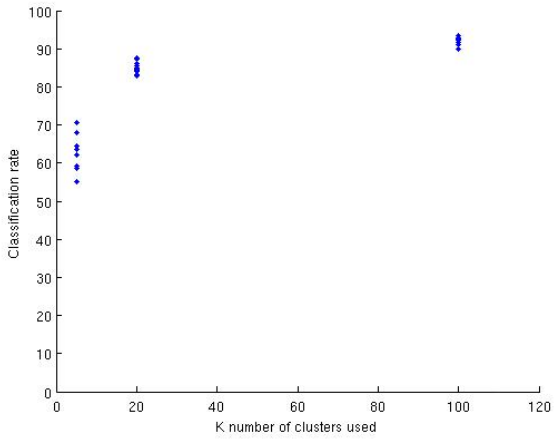


Fig. 5. Classification performances with different training sets. K = 5,20 and 100

training set without over-fitting. Given a set of labelled data, the algorithm should be test with different pairs of training and testing sets. Ten training sets was randomly selected and tested the classification with $k = 5, 20, 100$. The performances are plotted in Fig 5. The average performance of the classifier with $k = 20$ is significantly higher than $k = 5$, while $k = 100$ is only slightly better than $k = 20$. The variation in performance between different training sets is smaller when k is bigger. The range of performances with $k = 20$, the recommended value for k, is less than 7%, with the worst case still better than 80%. This is an indication that the algorithm is resilient to changes in the training sets.

V. Conclusion

An efficient classification algorithm for obstacle detection that combines colour and texture features were presented. The algorithm achieves a good trade off between performance and efficiency. A feature learning method was described. The learning method selected from a set of texture features the best features that can improve a colour classifiers performance. By using the complex texture features only when the basic colour classifier fails, our algorithm does not waste any computational resource for extracting texture in most cases. It was concluded that by utilising both colour and texture features, the algorithm achieves a good classification rate while is still simple enough to run on a mini computer platform suitable for outdoor robots. There are few cases that the algorithm fails to classify, but these cases are difficult even for human. Additional sensor such as sonars or bump sensors can be used to resolve these cases.

References

[1] P. E. Richard O.Duda and D. G.Stork, *Pattern Classification*. Inter-Science, 2001.
[2] I. Ulrich and I. R. NourbakhshLo, "Appearance-based obstacle detection with monocular color vision," in *AAAI/IAAI*, 2000, pp. 866–871.
[3] M. Sridharan and P. Stone, "Color learning and illumination invariance on mobile robots: A survey," *Robotics and Autonomous Systems*, vol. 57, no. 6-7, pp. 629 – 644, 2009.
[4] L. Lorigo, R. Brooks, and W. Grimsou, "Visually-guided obstacle avoidance in unstructured environments," in *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 1, Grenoble, France, Sep. 1997, pp. 373–379.
[5] T. S. team, "Stanley : The robot that won the darpa grand challenge," Stanford University, Tech. Rep., 2005.
[6] S. Belongie, C. Carson, H. Greenspan, and J. Malik, "Color- and texture-based image segmentation using em and its application to content-based image retrieval," *iccv*, vol. 00, p. 675, 1998.
[7] Y.-C. Cheng and S.-Y. Chen, "Image classification using color, texture and regions," *Image and Vision Computing*, vol. 21, no. 9, pp. 759–776, Sep. 2003.
[8] D. Cremers, M. Rousson, and R. Deriche, "A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape," *International Journal of Computer Vision*, vol. 72, no. 2, pp. 195–215, Apr. 2007.
[9] M. Shneier, T. Chang, T. Hong, W. Shackleford, R. Bostelman, and J. Albus, "Learning traversability models for autonomous mobile vehicles," *Autonomous Robots*, vol. 24, no. 1, pp. 69–86, Jan. 2008.
[10] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *ICML '05: Proceedings of the 22nd international conference on Machine learning*. New York, NY, USA: ACM Press, 2005, pp. 593–600.
[11] R. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, May 1979.
[12] T. P. Weldon, W. E. Higgins, and D. F. Dunn, "Efficient gabor filter design for texture segmentation," *Pattern Recognition*, vol. 29, no. 12, pp. 2005–2015, Dec. 1996.
[13] M. Unser, "Texture classification and segmentation using wavelet frames," *Image Processing, IEEE Transactions on*, vol. 4, no. 11, pp. 1549–1560, Nov 1995.
[14] T. Ojala, M. Pietikinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, Jan. 1996.
[15] S. Grigorescu, N. Petkov, and P. Kruizinga, "Comparison of texture features based on gabor filters," *Image Processing, IEEE Transactions on*, vol. 11, no. 10, pp. 1160–1167, Oct 2002.
[16] T. Randen and J. Husoy, "Filtering for texture classification: a comparative study," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 4, pp. 291–310, Apr 1999.
[17] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *Int. J. Comput. Vision*, vol. 43, no. 1, pp. 29–44, 2001.
[18] "http://www.gumstix.org."