2008 IEEE/RSJ International Conference on Intelligent Robots and Systems
Acropolis Convention Center
Nice, France, Sept, 22-26, 2008

Fast Color/Texture Segmentation For Outdoor Robots

Morten Rufus Blas, Motilal Agrawal, Aravind Sundaresan, Kurt Konolige

***Abstract*— We present a fast integrated approach for online segmentation of images for outdoor robots. A compact color and texture descriptor has been developed to describe local color and texture variations in an image. This descriptor is then used in a two-stage fast clustering framework using K-means to perform online segmentation of natural images. We present results of applying our descriptor for segmenting a synthetic image and compare it against other state-of-the-art descriptors. We also apply our segmentation algorithm to the task of detecting natural paths in outdoor images. The whole system has been demonstrated to work online alongside localization, 3D obstacle detection, and planning.**

I. INTRODUCTION

Autonomous navigation for outdoor, unstructured environments is an important research problem in robotics with numerous applications. The ability to recognize navigable terrain and avoid obstacles is a critical component for autonomous navigation. Current state-of-the-art systems employ a range sensor such as stereo cameras or LADAR to reason about the geometry of the world and identify geometrical obstacles. However, geometrical reasoning alone is unlikely to result in intelligent behavior of the robot. For example, it is hard to distinguish between tall grass and a short wall based on geometry alone. In addition, learning is a very important component of an intelligent system. If the robot has traversed over tall grass earlier, it can learn that as traversable terrain and mark it as such if it sees it again. It is clear that appearance-based terrain recognition plays an important role in such intelligent behaviors and segmentation is the first step toward recognition.

Appearance-based segmentation is a classical problem in computer vision. For robotics, the specific challenge is to be able to do reliable segmentation of outdoor scenes in an efficient manner so that it can be used online. In our experience, color alone is not a reliable feature. For example, in Figure 1(a) it is hard to distinguish between the bushes and the darker grass on the ground based on color. However, the texture of the grass and the bushes is very different.

In this paper, we present an online segmentation algorithm that combines color with texture information to group similar regions. Our algorithm has several novel features.

This material is based upon work supported by the United States Air Force under Contract No. FA8650-04-C-7136. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force

Morten Rufus Blas is with the department of Electrical Engineering at DTU in Lyngby, Denmark. mrb@elektro.dtu.dk

Motilal Agrawal, Aravind Sundaresan, Kurt Konolige are with the Artificial Intelligence Center at SRI International in Menlo Park, CA, USA. {agrawal, aravind, konolige}@ai.sri.com

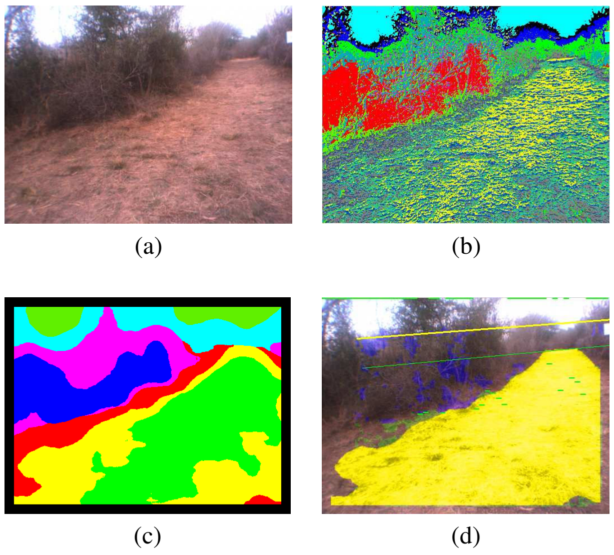

Fig. 1. Various steps of our segmentation algorithm on a typical outdoor image. (a) The image from one of the stereo cameras. (b) Each pixel assigned to a texton. (c) Each histogram of textons gets assigned to a histogram profile. (d) A path is recognized (in yellow)

- Compact texture/color descriptors. It is important to have compact representations of the information necessary to distinguish textures. Here we carefully choose a small local neighborhood vector that incorporates the important aspects of texture and color.
- A two-stage unsupervised online learning process. For each image, we cluster neighborhood vectors to find a small set of basis vectors (textons [1]) that characterize scene textures (Figure 1(b)). Then, we cluster histograms of textons over larger areas to find more coherent regions with the same mixture of textons (Figure 1(c)).

Note that the problem we are interested in here is online, unsupervised segmentation, not classification based on a library. One application is finding paths in off-road terrain, where the path appearance may be unlike anything seen previously (Figure 1(d)). We have successfully demonstrated online path detection in a complete outdoor navigational system that uses stereo-vision as its primary sensor.

II. ALGORITHM OVERVIEW AND RELATED WORK

A. Texture Representation

Approaches to texture representation include co-occurrence probabilities [2], Markov modeling [3], [4], [5], multichannel filtering [6], [7], [8], [9], Local Binary Patterns (LBP) [10], and texton-based approaches [9], [11], [1]. The more recent approaches use either a filter bank or a small neighborhood as a feature descriptor for each pixel – for example, LBP's are formed by subtracting the intensity of the center in a small local neighborhood and then binarizing the intensity variation in the neighborhood.

In a seminal paper, Leung and Malik [1] showed that many textures could be represented and re-created using a small number of basis vectors extracted from the local descriptors; they called the basis vectors *textons*. While Leung and Malik used a filter bank, later Varma and Zisserman [12] showed that small local texture neighborhoods may be better than using large filter banks. In addition, a small local neighborhood vector can be much faster to compute than multichannel filtering such as Gabor filters over large neighborhoods.

Many schemes exist for combining local texture with color information [9], [11], [13]. The sheer number of variations makes it hard to decide what is a good representation of both color and texture for segmentation. In this paper, we describe a segmentation algorithm that uses a compact descriptor for representing color and texture. Our descriptor fits into the class of local texture neighborhoods and in that sense is similar to LBP's. For each local neighborhood (a 3x3 or 5x5 region centered at a pixel), the descriptor is composed of the color information of the center and the relative change in intensity in the neighborhood. This is computed by subtracting out the intensity of the center from the intensities in the neighborhood. Unlike LBP, we do not binarize the center subtracted intensity values, thereby retaining the actual gradient values. In contrast with other local neighborhood descriptors, ours is more compact since we do not store the color variation in the neighborhood. For a typical 3x3 window size, for example, our descriptor is an 11-dimensional vector, whereas storing the raw RGB values will result in a 27-dimensional vector. A compact descriptor becomes crucial for the clustering step to be fast and real time.

B. Segmentation

The raw descriptors must be grouped to segment the image; a number of clustering algorithms exist for this task. The K-means algorithm and its many variations is a standard way of doing this [14], [15]. Graph-cut-based approaches [9] generally result in better and sharper boundaries but are computationally more demanding. Self-Organizing Maps [16] yield clusters such that neighborhood relations between the clusters are preserved, allowing one to better visualize the input space. Another approach are level-sets [13] which handles boundaries by first finding homogeneous areas in the image and then propagates these areas to unlabelled parts of the image.

In our algorithm we use two-stage, unsupervised clustering to find smooth similar regions based on the descriptors. The choice of clustering framework was largely dictated by the need for it to be fast and efficient. In our method, we use the K-means algorithm to perform clustering – K-means has the best trade-off between good results and speed.

For the first clustering step, our descriptors are computed at each pixel and are then clustered using K-means to find the basis vectors or textons. Each pixel then gets assigned to the closest texton. This is shown in Figure 1(b). As can be seen, a segmentation based on simple pixel classification is very noisy. To capture statistics of larger areas, we compute a histogram of these textons over a window, and cluster the histograms again using K-means to find similar regions in the image. Histograms of textons are computed efficiently using integral images [17]. Regions that are close together are then merged to give the final segmentation. Figure 1(c) shows the final segmentation results.

Not only is our descriptor compact and faster to compute but it captures all the local texture variations, resulting in better segmentations. We present experimental results of comparing these texture descriptors to segment a synthetic image. While there has been previous work [12] on comparing the different types of texture descriptors for the task of texture classification, to our knowledge no comparisons have been done earlier for the task of texture segmentation.

C. Path Finding

Finally, for an application, we use the segmentation algorithm to recognize natural paths in outdoor images. The image is segmented, and then we look for regions that share the geometric attributes of a path. Because there are only a small number of regions, various combinations of regions that could possibly be a path can all be checked. This path detection algorithm runs online on the robot in real time and we present results of our path detection algorithm on several types of outdoor paths. Other work has previously been done in road detection where we mention the work by Fernandez and Price [18] who used region growing in HSI color values to find the road borders. Others include Dahlkamp et al. [19], who used self-supervised learning on color images to extend roads found by LADAR. In the area of stereo-vision Soquet et al.[20] used a stereo-based color segmentation algorithm to determine road segments. Texture has also been used as seen in Zhang and Nagel [21], who explore anisotropic texture features of roads for segmentation.

While all the individual components of our segmentation algorithm are known, we have judiciously selected each step of the processing pipeline so that we are able to run our algorithm online on the robot in real time. It is the integration of these fast techniques, coupled with our compact texture descriptor and a two-stage online learning process, that characterizes our work.

The rest of the paper is organized as follows. Section III describes our segmentation algorithm in detail, and the results of our segmentation algorithm for a synthetic image are presented and compared with other texture descriptors in Section IV. Our path recognition algorithm is discussed in Section V and results of this algorithm are discussed in Section VI. Section VII concludes the paper and discusses ongoing and future work.

III. SEGMENTATION ALGORITHM

The first step of our segmentation algorithm is to learn a set of textons (basis descriptor vectors) for the image. A local descriptor (3x3 or 5x5 window) is computed at each pixel location, and the ensemble of descriptors is clustered to find a small set of textons. Each pixel location then gets assigned to one of these textons by comparing its descriptor using Euclidean distance.

A. Textons

Our descriptor is composed of color and texture information for a 3x3 or 5x5 pixel neighborhood. The image is first transformed to the CIE*LAB colorspace using an efficient lookup table to do the RGB to LAB conversion. Colors in LAB are more perceptually linear than in the RGB space, thereby resulting in better clusters. This gives the brightness information L and the color channels a, b. The texture information is taken as the surrounding pixel intensities minus the center intensity. Each pixel location p_i in the image can then be represented using the descriptor:

$$p_i = \begin{bmatrix} W_1 * L_c \\ W_2 * a_c \\ W_2 * b_c \\ W_3 * (L_1 - L_c) \\ \vdots \\ W_3 * (L_8 - L_c) \end{bmatrix} \qquad (1)$$

Here (L_c, a_c, b_c) is the color of the center pixel, and $L_1, L_2, ..., L_8$ are the intensities of the surrounding pixels. The set of weights $\{W_1 = 0.5, W_2 = 1, W_3 = 0.5\}$ is used to balance how much to rely on color, texture, and brightness for the clustering. These were set as to weigh chrominance higher than luminance. Also, since texture takes up many of the descriptor rows it must be downweighted so the color still has an impact on the clustering. The assumption here is that in a local neighborhood the color does not vary much, so including the color channels for all 3x3 pixels does not provide additional information. The L_c, a_c, b_c components could also be computed as an average of the 3x3 neighborhood but this was not done to speed up computation. See top of Figure 2 for an overview. The 5x5 version of the descriptor is similar but uses a larger neighborhood size resulting in a 27-dimensional descriptor.

B. Clustering to Textons

The K-means algorithm seeks to minimize

$$J = \sum_{i=1}^{n} \min_j |p_i - c_j|^2 , \qquad (2)$$

where p_i is each descriptor in the image and c_j are the textons; basically, it finds a set of basis descriptors such that

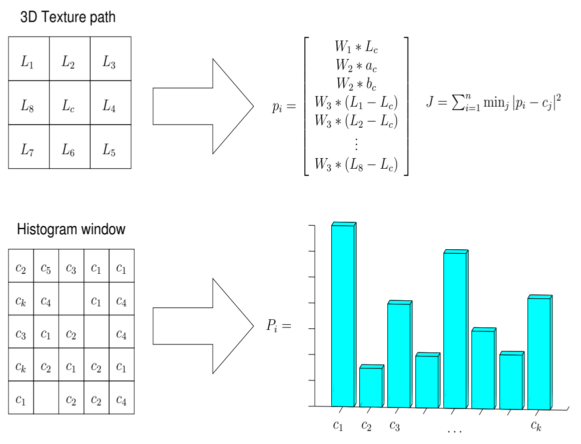
3D Texture path
L_1 L_2 L_3
L_8 L_c L_4
L_7 L_6 L_5
$p_i = [W_1 * L_c, W_2 * a_c, W_2 * b_c, W_3 * (L_1 - L_c), W_3 * (L_2 - L_c), \vdots, W_3 * (L_8 - L_c)]$
$J = \sum_{i=1}^{n} \min_j |p_i - c_j|^2$
Histogram window
c_2 c_5 c_3 c_1 c_1
c_k c_4 c_1 c_4
c_3 c_1 c_2 c_4
c_k c_2 c_1 c_2 c_1
c_1 c_2 c_2 c_4
$P_i =$
c_1 c_2 c_3 ... c_k

Fig. 2. The segmentation algorithm works in two steps. First textons are learned from the image. Then histograms are constructed from textons and clustered. The missing values in the histogram window represent outliers.

the Euclidean distance between them and all descriptors is minimized. $j = 1, .., k$ is the number of textons we desire to learn. For our outdoor robotic sequences $k = 16$ gave a good trade-off between accuracy and efficiency. In the K-means iterations, reclassification attempts are not made for points that lie less than half the mean absolute distance away from their currently classified center (similar to [22]). This considerably speeds up the implementation without a significant loss in precision.

C. Histogram Clustering

Once the 16 textons for a given image have been established, each pixel is classified as belonging to one of these using Euclidean distance. A simple threshold identifies outliers. Integral images [17] are then constructed for each of the 16 textons. An entry in the integral image at location x is simply the sum of the count of each of the 16 textons in the rectangle formed by the point x and the origin. With the integral image calculated, it takes only four additions to calculate the total number of each texton over any upright, rectangular area, independent of its size. This is then used to extract a histogram profile for a window neighborhood across the image. Experimentally, a 32x32 window gives the best results for our image size. This is similar to what was observed in [13] where the window was chosen to 1-2% of the total image size.

K-means is then run on the histograms to extract a set of histogram profiles, using Euclidean distance as the norm. Boundary conditions between areas of different texture are not explicitly treated and thus may receive their own cluster representing ”mixed terrain”. The choice of histogram clusters ($k = 8$) was set so as to slightly over-segment the image. Texton outliers are not included in the histogram clustering. See bottom of Figure 2 for an overview.

Finally, the Earth Movers Distance [23] was used to merge similar clusters if the threshold was below 100. EMD is a good distance measure for histograms, but too computation-

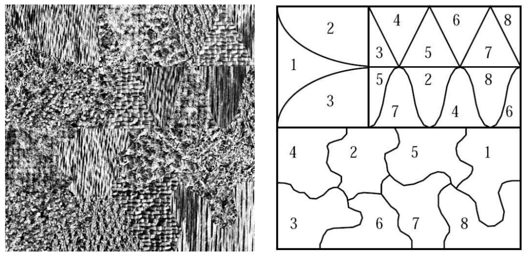

Fig. 3. Synthetic texture mosaic used (provided by USC via its website). The left image is the texture mosaic. The right image shows which texture regions belong to which texture.

ally expensive to be used directly in the K-means clustering. The EMD ground distance matrix for the histograms was set to the Euclidean distance between each basis texton. Given two textons $c_{t,i}$ and $c_{t,j}$ the Euclidean distance between them can be written as $d_{t,ij}$:

$$d_{t,ij} = \|c_{t,i} - c_{t,j}\|^2 \tag{3}$$

Given that we have 16 basis textons this gives a 16x16 distance matrix $\mathbf{D}$ for comparing two histograms (generalized for an m-by-n matrix):

$$\mathbf{D} = \begin{bmatrix} 0 & d_{t,01} & d_{t,02} & \cdots & d_{t,0n} \\ d_{t,10} & 0 & d_{t,12} & \cdots & d_{t,1n} \\ d_{t,20} & d_{t,21} & 0 & \cdots & d_{t,2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{t,m0} & d_{t,m1} & d_{t,m2} & \cdots & 0 \end{bmatrix} \tag{4}$$

The EMD then attempts to solve the transportation problem of

$$\text{WORK}(P, Q, \mathbf{F}) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{t,ij} f_{ij} \tag{5}$$

subject to a number of constraints on f_{ij} as described in [23]. P and Q are the two compared histograms and $\mathbf{F}$ is the flow that minimizes the above cost. If the flow is very small the clusters are similar and are merged based on a threshold.

Last, the image is reclassified using the computed histogram profiles. Histograms that are not close to the computed histogram profiles are thresholded as outliers.

IV. SEGMENTATION RESULTS

It is important that the textons contain the information necessary to accurately discriminate between different textures. We compare our texture descriptor to various other state-of-the-art descriptors by applying them to the task of segmenting a synthetic image into different textures.

The University of Southern California (USC) hosts the Brodatz texture database and also provides texture mosaics that are a number of Brodatz textures stitched together in a jigsaw-type pattern. *texmos3* was selected as the texture mosaic for benchmarking our texton descriptors. Figure 3 shows this mosaic along with the ground truth segmentation. This mosaic has eight textures and does not contain color, which tests the descriptors' ability to discriminate textures. Four basic descriptors are tested: a 48-dimensional descriptor composed of the responses from the Leung-Malik filter bank (LM,32); a 75-dimensional descriptor of 5x5 raw RGB (RGB,5x5,32) values as used in [11] (which in effect is 25-dimensional on grayscale images); the LBP in a 3x3 neighborhood (LBP,3x3,32); and two versions of our descriptor – the 3x3 neighborhood (11-dimensional with the L,a,b color components set to zero), (SRI,3x3,32) and a 5x5 neighborhood (SRI,5x5,64) with the descriptor components still being the intensities minus the center intensity. For the test, the LM filter bank is the only one where the descriptors are not learned on the image itself. For all other descriptors, 32 textons are learned from the image itself. Our 5x5 version used 64 textons illustrating our best possible result. The lack of color information meant that more textons were needed to discriminate the textures. The second stage of clustering is then applied to give the segmentation results. It is important to note that the underlying segmentation algorithm is the same (as described in section III) for each of these descriptors.

Each descriptor is then scored using two scores – the detection rate and the confusion rate. The detection rate gives a measure of how much of a given texture it managed to classify correctly. The confusion rate gives a measure of how many correct versus false detections to expect. A good segmentation will have a high detection rate and a low confusion rate. For the detection rate we look only at texture regions that are entirely inside our 32x32 histogram window (so only one texture is present inside the window). This is done by eroding the borders of each texture region with a flat 16 pixel radius circle structuring element. This gives us a maximum on the number of possible correct inliers $D_{\max,t}$ for a given texture. The cluster that takes up the most area of a given texture is chosen as the cluster that belongs to that texture. For a specific texture t, the number of correct detections is called $D_{c,t}$, and the number of false detections is $D_{f,t}$. The two scores for a specific texture are then calculated as

$$\text{Confusion Rate} = 100 \times \frac{D_{f,t}}{D_{c,t} + D_{f,t}} \tag{6}$$

$$\text{Detection Rate} = 100 \times \frac{D_{c,t}}{D_{\max,t}} \tag{7}$$

The total confusion and detection rates are simply the sums over all the eight textures.

$$\text{Total Confusion Rate} = 100 \times \frac{\sum_{t=1}^{8} D_{f,t}}{\sum_{t=1}^{8} (D_{c,t} + D_{f,t})} \tag{8}$$

$$\text{Total Detection Rate} = 100 \times \frac{\sum_{t=1}^{8} D_{c,t}}{\sum_{t=1}^{8} D_{\max,t}} \tag{9}$$

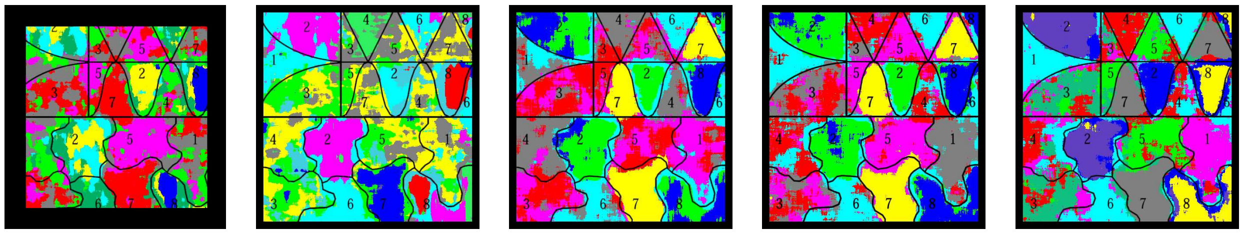

Fig. 4. Results for the synthetic texture segmentation. Each color represents a different histogram cluster. An overlay shows which regions should have homogeneous colors. (a) LM Filter, 32, (b) RGB 5x5, 32, (c) LBP 3x3,32, (d) SRI 3x3,32, (e) SRI 5x5,64.

%	LM,32	RGB	LBP	SRI,3x3	SRI,5x5
Total Conf.	50	56	46	38	34
Total Det.	40	53	68	79	68

TABLE I
TOTAL RATES

The actual segmentations obtained for each descriptor can be seen in Figure 4. Figures 5 and 6 show the two scores for each of the eight textures present in the mosiac. The total confusion and detection rates are shown in Table I. The LM filter bank performs the worst, as it has higher confusion and lower detection rates than all the other descriptors. This fits with the observations in [12]. The raw intensity value descriptor also performs poorly. LBP has problems discriminating between textures 2 and 8 but is otherwise clearly better than the raw intensities and LM filter bank. Our descriptors do a much better job at discriminating between textures 2 and 8, which indicates that the intensity gradients are necessary to do this and that it is not enough to rely just on the gradient direction. All the methods find it hard to discriminate between textures 3 and 4 except the LBP, which aids it greatly in the total scores. The results for our descriptor are on average better than the other methods on this dataset. Interestingly, for our descriptors the 3x3 version actually gets a better total detection rate than the 5x5 version at the cost of a higher total confusion score.

The results presented here are typical for other mosaics in the synthetic dataset and have been omitted beacuse of space constraints.

V. APPLICATION: PATH RECOGNITION

This work has been carried out in conjunction with a larger research project entitled Learning Applied to Ground Robotics (LAGR). The project deals with outdoor navigation in unstructured environments using stereo vision. The goal is to navigate a robot autonomously to a GPS waypoint through unknown terrain at a speed of roughly $1m/s$. Many of the environments tested include small paths in the form of dirt/asphalt roads as well as more natural paths such as beaten-down tracks through vegetation. Many of the paths do not have a clear signature in the 3D output of the stereo-vision sensor. We have used our segmentation algorithm to

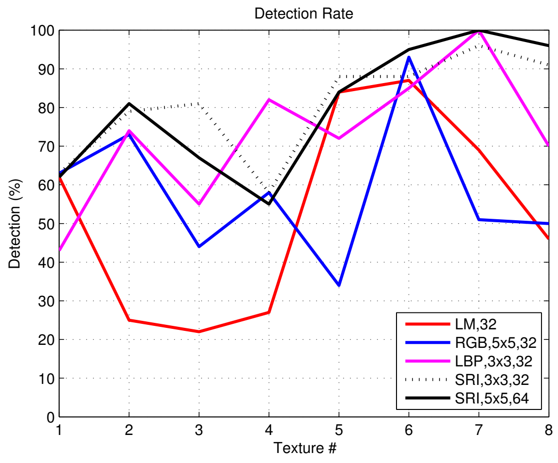

Fig. 5. Detection rate of the individual textures for the five tested feature descriptors on the artificial dataset.

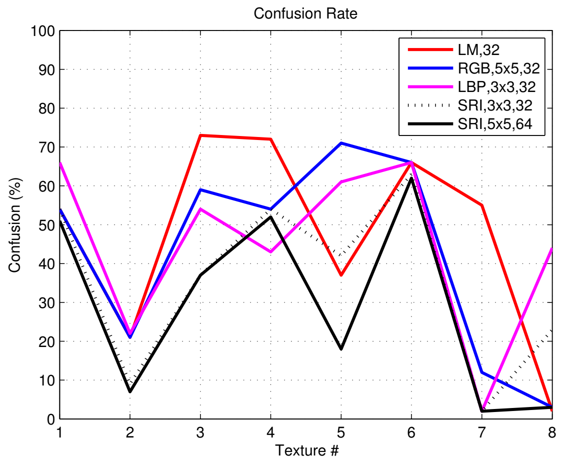

Fig. 6. Confusion rate of the individual textures for the five tested feature descriptors on the artificial dataset.

recognize these paths, using geometrical constraints from the stereo sensors (flatness, width) to find segmented regions that could be paths.

Figure 7 illustrates a sample image (a), the texture-based segmentation (b), the disparity map computed from stereo (c), and the ground plane inliers (d). The ground plane is computed from stereo information. The objective is to determine if any of the segments in the image (b) constitute a path. We project the segmented image onto a 2D grid on the ground plane (Figure 8 (a)) to obtain the segmentation map (Figure 8 (b)), which is a bird's-eye view of the textures placed on the ground plane. In order to determine if a set of segments constitutes a path, we first obtain the corresponding path map (Figure 8 (c)) and compute path statistics on the path map as described in Section V-B. The statistics such as the width profile help us determine if the selected segments actually constitute a path. In Section V-A, we describe how different segments or textures are combined to detect paths.

A. Path Detection Using a Segmented Image

We sample the segmented image on a 2D grid on the ground plane to obtain the "segmentation map". The grid points are illustrated in Figure 8(a) and the corresponding $N_i \times N_j$ "segmentation map", $T_{i,j}$, is illustrated in Figure 8(b). We note that the path can be composed of a single segment or a combination of segments. For a given combination of segments, $\mathcal{S}$, the width profile can be computed directly from the segmentation map as

$$w_i^S = \sum_{j=1}^{N_j} \sum_{k \in \mathcal{S}} \delta(T_{i,j} - k). \quad (10)$$

The mean and deviation (15)-(16) serve as a simple means to identify segments or combinations of segments that could constitute a path. A set of segments is detected as a path if its mean width, deviation, and length lie within certain predetermined thresholds. In the LAGR experiments, for example, we considered paths whose width was in the range 0.5 m to 2 m, with deviation less than 0.15 m and length greater than 4 m and the thresholds were set accordingly. The simple width profile can be computed quickly and is also linear in the number of segments, i.e.,

$$w_i^S(\mathcal{S}_1 + \mathcal{S}_2) = w_i^S(\mathcal{S}_1) + w_i^S(\mathcal{S}_2). \quad (11)$$

We see from (11) that it is easy to compute the width profile of a path comprising multiple segments using the width profile of the component segments. The width profile computed in this manner can be used to identify single and compound segments that constitute a path for different combinations of segments. Once we obtain a list of candidate segments we can check for both row-wise and column-wise spatial coherence (next subsection). For a compound segment path consisting of the set of segments, $\mathcal{S}$, the path map is assigned as

$$p_{i,j} = \begin{cases} 0 & T_{i,j} = 0 \\ 1, & T_{i,j} \in \mathcal{S} \\ -1, & \text{otherwise.} \end{cases} \quad (12)$$

Figure 8(c) illustrates the path map obtained by combining segments labeled red and yellow. Figure 8(d) illustrates the width at each pixel ($W_{i,j}$) as well as the path center that was computed by fitting a quadratic curve.

B. Computing the Path Profile

We describe how we compute the width profile and other statistics of a $N_i \times N_j$ 2D path map such as the one in Figure 8 (c). The path map is a 2D grid on the ground plane whose grid points are labeled as "path", "not path", or "unknown" with values as follows.

$$\text{Pixel } p_{i,j} \text{ is labeled as} \begin{cases} \text{path,} & \text{if } p_{i,j} = 1 \\ \text{not path,} & \text{if } p_{i,j} = -1 \\ \text{unknown,} & \text{if } p_{i,j} = 0 \end{cases} \quad (13)$$

The basic idea is to determine the existence of a consistent path by computing its width profile, i.e, its width in each row. We first determine the simple width profile (w^S), and then consider the spatial coherency in each row (w^R), and finally across the columns (w^C). We compute the simple width profile, w_i^S, which is the width of the path in the i^{th} row, as the number of pixels that are labeled as path in each row.

$$w_i^S = \sum_{j=1}^{N_j} \delta(p_{i,j} - 1) \quad (14)$$

The mean width and the deviation can be computed for a width profile as

$$\mu = \frac{1}{N_j} \sum_i w_i \quad (15)$$

$$d = \frac{1}{N_j} \sum_i |w_i - \mu| \quad (16)$$

While w^S is a simple means of testing if the width is consistent, it fails to take into account if the path pixels are spatially adjacent. We therefore compute a "running average" of the path width centered at each pixel using a window of length $2L + 1$, which is computed as

$$W_{i,j} = {}^{+}\sum_{k=j-L}^{j} p_{i,k} + {}^{-}\sum_{k=j+L}^{j+1} p_{i,k} \quad (17)$$

where

$${}^{+}\sum_{k=0}^{j} x_k = \max\left(0, {}^{+}\sum_{k=0}^{j-1} x_k + x_j\right), \text{ and} \quad (18)$$

$${}^{-}\sum_{k=L}^{j} x_k = \max\left(0, {}^{-}\sum_{k=L}^{j+1} x_k + x_j\right). \quad (19)$$

The two terms in (17) describe the widths on the left and right of the pixel (i, j). These can be computed recursively (18-19) and are constrained to be nonnegative. The new width profile, w_i^R, of row i is computed as

$$w_i^R = \max_j W_{i,j} \quad (20)$$

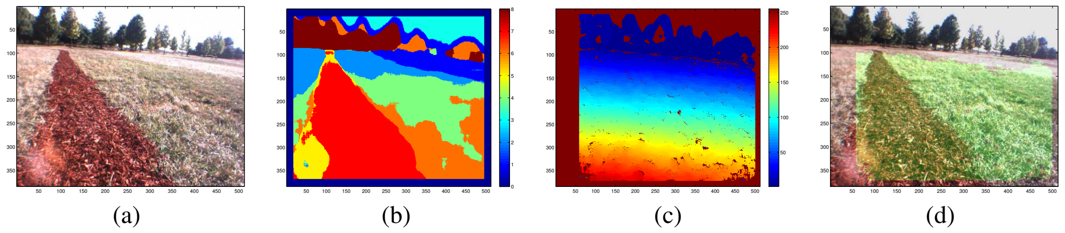

Fig. 7. Images illustrating the information used in path detection. (a) The image from one of the stereo cameras. (b) Assigned texture labels. (c) Disparity values of the pixels; red is closer, blue is farther away. (d) The inliers in the ground plane in green overlay, computed from (c).

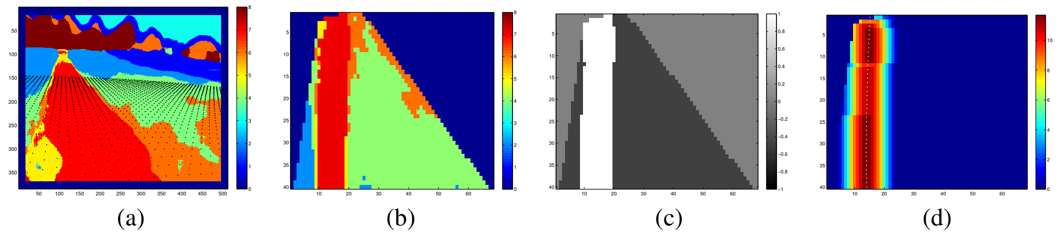

Fig. 8. Computing the path map and path statistics. (a) Ground plane uniform grid projected onto the image. (b) The texture values at the grid points (segmentation map), showing an overhead geometrical view of the textures on the ground plane. Note the main path texture (red) is now clearly a path. (c) A path map obtained by combining the red and yellow textures. The hypothesized path is in white, textures that are "not path" are in black, and unknown areas are in gray. (d) The width of the hypothesized path at each pixel. The center of the path is marked by a white dot.

and takes into account spatial coherency in each row. Thus, if a row has a certain number of path pixels, its width is highest when all of them are adjacent.

We next check the spatial consistency of the path across rows. We note that $W_{i,j}$ in each row obtains the maximum value at the center of the path and we obtain the column corresponding to the maxima of $W_{i,j}$ for each row $i : j_i^{\max} = \arg_j \max W_{i,j}$. We then fit a quadratic curve to the set of points $(i, j_i^{\max})$ using RANSAC to obtain the path center in each row. The column-wise spatially coherent width profile, w_i^C is computed as the path width at the fitted path center

$$w_i^C = W_{i,j_i}, \qquad (21)$$

where j_i is the fitted path center at row i.

VI. PATH RECOGNITION RESULTS

As part of the LAGR program, an independent testing group ran monthly blind tests of the perception and control software. The nine competing teams in the LAGR program were compared to a baseline system; each team was scored based on the time taken by its robot to reach the goal. Figure 7(a) is one of the tests. Here, a path can be identified as a dirt section among the grass.

In real time tests, we run the segmentation algorithm at slightly slower than a 1 Hz rate. The perception system passes information about paths to the planner, along with other information about obstacles and freespace. Figure 9 shows the trail amongst the bushes detected as a path. This information is then passed onto the planner. Figure 10 shows the planner operating on the information returned by the perception algorithms. The path segmentation contributes the yellow center section, which is preferred by the planner. Using the path helps the robot to stay away from the bushes surrounding the path (where the robot wheels might get stuck). Also, since path costs are lower, the robot avoided squeezing through the open space between the bushes. This behavior is similar to that of a person, who would prefer the easy path rather than more dense terrain among the bushes.

Because of the strong geometric tests, no false positives of recognized paths have been experienced in any of the LAGR tests.

Timings for our segmentation algorithm on a 512 by 384 color image are shown in Table II. The computational platform is a 2 GHz Pentium-M machine. For our K-means algorithm, the maximum number of iterations was fixed at 100. In practice, K-means converges much before that and is stopped when it has converged. It takes about 1 s to perform the two-stage online learning process, and about 150 ms to classify a 512 by 384 image using our 3x3 descriptor. The LM filters have not been implemented to work with color and so are excluded from the timings.

	LM,32	**RGB**	**LBP**	**SRI,3x3**	**SRI,5x5**
Time (s)	*N/A*	5.14	2.59	1.11	3.01

TABLE II
SEGMENTATION TIMES

VII. CONCLUSIONS

We have presented a segmentation algorithm suitable for robotic applications in outdoor environments. Our segmentation algorithm uses a compact feature descriptor in a two-

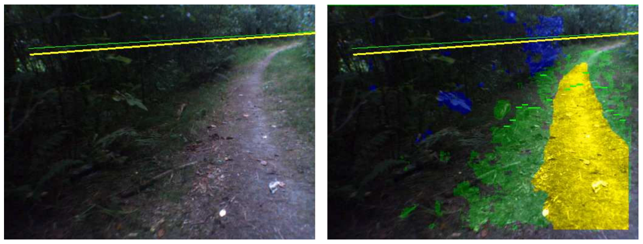

Fig. 9. Example of path classification. The costmap in Fig. 10 was created by driving along this path. The yellow color indicates the detected path. Green is ground plane and blue are obstacles. The two horizontal lines are estimates for the location of the horizon.

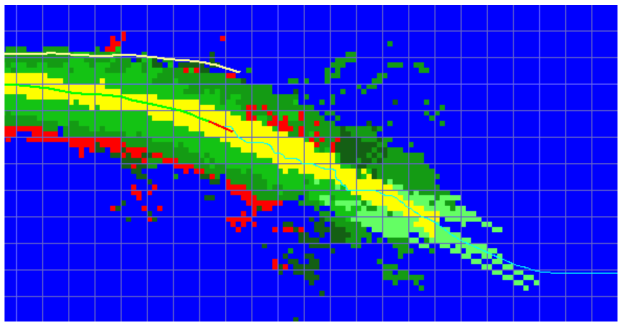

Fig. 10. Example of a costmap built by the LAGR robot and subsequently used by the planner. Blue is unknown terrain. Obstacles are shown in red. Ground plane is shown in various shades of green (with brighter colors indicating lower cost). The yellow region is the detected path. The robot position is marked with a red line. The cyan line indicates the planned trajectory. The green line indicates where the robot has driven. The superimposed grid squares have a length of 1 m.

stage K-means-based clustering algorithm. We have shown that our descriptor does a better job at texture segmentation than other commonly used texture descriptors. We have also applied our segmentation algorithm for recognizing natural paths in outdoor environments in real time. The approach has been demonstrated online for following natural paths on an outdoor robot. Although false positives have not been experienced in the LAGR tests, they could potentially occur if the segments happen to resemble a path geometrically. Another failure mode could occur if the path is segmented into too many regions. We also need to look into the maximum angle and distance relative to the robot at which the path can be detected.

The segmentation algorithm forms a basis for performing appearance-based terrain classification. We are currently looking into building a database of commonly seen terrain types such as tall grass, sandy soil, gravel, and mulch; textons can be learned offline for each class, and then online each segment can be classified into one of these terrain types. Such terrain classification can also be performed entirely online, wherein the robot learns from its own experience. For each terrain type that the robot has been on, the robot can learn its color, texture, and navigability characteristics. Subsequently, it can predict the navigability characteristics of an unknown terrain by recognizing its color and texture. Indeed, such behaviors can make the robot appear 'intelligent'.

References

[1] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *IJCV*, vol. 43 (1), 2001.

[2] R. Haralick, K. Shanmugan, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man. Cybern.*, vol. SMC-3 6, pp. 610–621, 1973.

[3] C. Kervrann and F. Heitz, "A markov random field model-based approach to unsupervised texture segmentation using local and global spatial statistics," *IEEE Trans. Image Process*, vol. 4 6, pp. 856–862, 1995.

[4] M. Bello, "A combined markov random field and wave-packet transform-based approach for image segmentation," *IEEE Trans. Image Process*, vol. 3 6, pp. 834–846, 1994.

[5] B. Manjunath and R. Chellappa, "Unsupervised texture segmentation using markov random field models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13 5, pp. 478–482, 1991.

[6] A. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *Pattern Recognition*, vol. 24 12, pp. 1167–1186, 1991.

[7] M. Unser, "Texture classification and segmentation using wavelet frames," *IEEE Trans. Image Process.*, vol. 4 11, pp. 1549–1560, 1995.

[8] A. Bovik, M. Clark, and W. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12 1, pp. 55–73, 1990.

[9] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, 2004.

[10] C. Chen and P. Wang, *Handbook of Pattern Recognition and Computer Vision, 3rd Edition*. World Scientific, 2005.

[11] A. Angelova, L. Matthies, D. Helmick, and P. Perona, "Slip prediction using visual information," *RSS*, 2006.

[12] M. Varma and A. Zisserman, "Texture classification: Are filter banks necessary?" *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 691–696, 2003.

[13] S. Liapis, E. Sifakis, and G. Tziritas, "Color and/or texture segmentation using deterministic relaxation and fast marching algorithms," *Journal of Visual Communication and Image Representation*, vol. 15, pp. 1–26, 2004.

[14] A. Jain and R. Dubes, *Algorithms for Clustering Data*, E. Cliffs, Ed. Prentice-Hall, 1988.

[15] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. Wiley, 1973.

[16] J. Martin-Herrero, M. Ferreiro-Arman, and J. Alba-Castro, "Grading textured surfaces with automated soft clustering in a supervised som," *Proceedings International Conference on Image Analysis and Recognition (ICIAR)*, pp. 323–330, 2004.

[17] P. Viola and M. Jones, "Robust real-time face detection," in *ICCV01*, 2001.

[18] D. Fernandez and A. Price, "Visual detection and tracking of poorly structured dirt roads," in *12th International Conference on Advanced Robotics, 2005*, 18-20 July 2005, pp. 553–560.

[19] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski, "Self-supervised monocular road detection in desert terrain," in *Robotics: Science and Systems*, 2006.

[20] N. Soquet, D. Aubert, and N. Hautiere, "Road segmentation supervised by an extended v-disparity algorithm for autonomous navigation," in *Proc. IEEE Intelligent Vehicles Symposium*, 13-15 June 2007, pp. 160–165.

[21] J. Zhang and H.-H. Nagel, "Texture-based segmentation of road images," in *Proceedings of the Intelligent Vehicles '94 Symposium*, October 1994, pp. 260–265.

[22] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24(7), pp. 881–892, 2002.

[23] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society, 1998, p. 59.