A Feature Construction and Classification Approach to Pixel Annotation

Arno Knobbe KNOBBE@LIACS.NL
LIACS, Leiden University, the Netherlands

Pieter Hoogestijn PIETER@HOOGESTIJN.NL
Everest, 's Hertogenbosch, the Netherlands

Durk Kingma DURK@ADVANZA.NL
Advanza, Utrecht, the Netherlands

Abstract

In this paper, we present a Machine Learning approach to an important problem in Computer Vision: the pixel annotation task. This task is concerned with assigning one of a small set of predefined classes to every pixel in an image. The classes can be any concept defined by the user, but in our case typically will represent particular surface-types. A logical example that we will be considering is the classification of road-surface. We will demonstrate how likely areas of tarmac can be identified using a trained classifier, and demonstrate how this low-level information may be used for navigation purposes. An important characteristic of the selected approach is the importance of feature construction. We define a large collection of features based on different aspects of a pixel and its environment. These include concepts such as colour, edges, textures and so on. Additionally, we will include a range of features derived from the spatial information available from stereo images. The information about surface location and orientation obtained through stereo vision may hold important clues about the pixel class that cannot be obtained from single images.

Appearing in *Proceedings of the 20th Machine Learning conference of Belgium and The Netherlands.*

1. Introduction

Despite many recent advances in the field of Computer Vision, it is still very hard for computers to truly understand what is happening in a given image. Quite a number of steps are necessary before the detailed low-level information can be turned into high-level semantic descriptions of the scene at hand. Most methods start by applying filters to the original image, for example for edge detection. These local features can then be combined into more intermediate-level objects such as lines and corners. Further steps may finally lead to the recognition of objects and their spatial relationships, although this general problem is still only solved for very restricted settings. In this paper, we aid this complex process, by presenting a solution for one of the necessary steps, namely the recognition of specific surface-types. This is done by training a learning algorithm on a database of previously annotated images.

Clearly, being able to assign types of material or surface to certain areas of the image is a very useful step in this process. If a certain procedure is able to find with a reasonable level of reliability which parts of an image represent, say, fur, then the task of recognizing (the location of) dogs is already partly solved. Next to being a good starting point for object recognition and image understanding, the ability to find areas of a given type can also be used for lower-level reasoning about possible actions in the observed scene. One of the tasks we consider in this paper is that of recognizing road surface. Clearly, an efficient form of path planning can be obtained from the knowledge of where exactly the road is heading, without exactly understanding the three-dimensional details of the scene and the location of possible obstacles. Minor errors in the assignment of

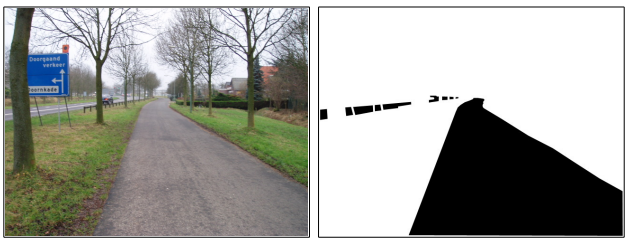

Figure 1. The original image and the binary annotation related to the concept 'road surface'.

road surface will be ignored because for path planning, only the overall identification of safe areas is relevant.

In this paper, we take a machine learning approach, by treating the surface identification problem as a classification problem. It is assumed that a (relatively small) number of images is annotated by hand, and a classifier is trained on the (large set of) examples obtained from these images. In the basic setting, the classification problem is assumed to be binary: a particular region either belongs to a certain class or it does not. Hence the annotation of a training image consists of a two-coloured image of the same proportions (see Figure 1). The binary setting can be easily upgraded to a multi-class problem by inducing multiple classifiers, one for each class, and then combining predictions.

The classification problem at hand is challenging for a number of reasons. First of all, it is unlikely that basic features such as RGB-values of individual pixels will be very discriminative. For example, the colour of tarmac is likely to be similar that of concrete, or even the sky on some days. Therefor, a considerable amount of feature construction will be necessary, in order to include local properties such as texture and gradient information. Section 3 lists the range of features that we consider. The problem is also challenging simply because of the complexity of the outside world. There may be many variations in the actual image, depending on weather and lighting conditions. Clearly, there is a need for robust features that are somewhat insensitive to such variations. For example, we require features that are not very sensitive to shadows on the road from, say, trees along the road. The *colour saturation* [1] (an inverse measure for the 'greyness') is such a lighting-invariant feature. Finally, the positive examples in the training data may form non-convex, or even noncontiguous regions in the input space, and classifiers need to account for such complexities. In the road surface task, for example, both the road markings and the actual tarmac need to be classified positively, even though these examples clearly lie in separate areas of the RGB space. In our second task, the identification of traffic signs, a range of different, bright colours may be encountered, each of which should be classified positively.

One special property of our approach (to the best of our knowledge unique) is the inclusion of spatial information in the list of features. In the simplest case, this involves the x and y-coordinate of the pixel, such that the classification procedure can estimate the likelihood of certain objects appearing in certain segments of the image (e.g. roads never appear above the horizon). More importantly, we are including 3D information obtained from stereo images. The Computer Vision package we are using, Harmonii [2], determines the three-dimensional location of each pixel in world-coordinates, using a disparity algorithm that determines the distance between corresponding pixels in the two images. This 3D information, and derived features such as the direction of the surface normal and the flatness of the surface, may help improve the predictive accuracy considerably. The down-side of including stereo information is the non-negligible computational cost of stereo vision. We will compare the results with and without 3D information, and will demonstrate that in some cases, the classifier-based approach may actually be an efficient replacement for the expensive stereo algorithm.

2. Problem Definition

The problem scope of this paper is the *classification of pixels* within a two-dimensional image, also referred to in literature as *pixel annotation* [3]. It is important to keep in mind that this is distinct from the related problem of image annotation [4], which is about classification of a whole image and automatic finding of the right image labels. The task of a pixel classifier, on the other hand, is to find the correct labelling of a pixel, given a finite set of possible classes. Since pixel classification is giving a separate class to each pixel, this is both intrinsically harder then image annotation, but also potentially more powerful since its result is more detailed in semantics.

In the field of Machine Learning, supervised learning is concerned with classification of unseen data given some training data with class labels. The training data is pairs of input vectors and class labels (this as opposed to unsupervised/semi-supervised learning, which is concerned with learning from (partially) unlabeled data). The job of the classifier is to generate a model using the available training data, and use this model to predict class labels for unseen, unlabeled data.

We are concerned with two separate classification tasks. The first task, *Tarmac*, is to distinguish road from non-road: a typical task in a robotic planning setting, where route planning is only acceptable over road surface. Note that the target concept not only includes actual tarmac, but also road markings, etc. Our second task, *Sign*, is to correctly pick out the pixels belonging to traffic signs. A challenge here is the many different colours that may appear on signs. Fig. 1 shows an example of the kind of signs we intend.

For each task, training data was generated by manually crafting monochrome overlays, black being the positive class (e.g. road), white being the negative class (e.g. non-road). This effectively created annotated training images. For both tasks, a moderate number of images were taken in an automotive setting with different types of tarmac and lighting conditions. A deterministic sliding window mechanism was used to generate the individual training samples, resulting in about 11k training points per image.

One important question is: what input vector should we use? An obvious answer is to simply extract the pixel values (RGB or HSV values) of the window surrounding the pixel. For example, we take a window of 5 by 5 pixels surrounding the pixel we want to classify, and use their RGB values as input vector, resulting in $5 \cdot 5 \cdot 3 = 75$ values as input vector and use these to learn and predict the class of the central pixel. In theory, if the window size is sufficiently large, the information provided should be enough. However, most classification algorithms in literature require features that are individually discriminative. In most applications, individual pixel values may not be very informative, so classification algorithms can fail to generate good models on such data.

To solve this problem, we add a pre-processing phase that constructs a large number of features. Each feature takes as input some information from a window surrounding the pixel, and outputs a scalar value calculated from this window. As these features are calculated as combination of pixel values, at least some features should have good discriminative power. Our complete list of features will be described in the next section.

An interesting research question is the value of depth (3D) information. The software we used, Harmonii [2], provides a robust stereo vision engine that generates a point cloud from two images, effectively providing 3D information for pixels in our images. Several features use this spatial information, and we will report the improvement in classification accuracy in the results section.

3. Feature Construction

As input for the classifier, a number of features were extracted from the images. Besides using the features already provided by the Harmonii software, we also used several combinations of these features.

In the subsections below, we explain all the features used in our experiments. The features are divided into a number of feature groups, which in our experiment could be toggled on and off for usage in building the classification model. In total, 137 features were defined to capture the properties of a particular pixel and its neighbourhood.

3.1. Colour features

The colour of the pixel and its neighbourhood is the most obvious information available to the classifier. The following 52 colour features were defined:

3.1.1. RGB and HSV, Grey.

The basic RGB-values were taken, as well as the derived HSV-values (*Hue*, *Saturation* and *Value* [1]). Additionally, we took *Grey*, which is the mean of the R, G and B components.

3.1.2. Colour intensities of red, green, blue.

Apart from the absolute colour levels, we also included the relative colour intensity of the colours red, green and blue:

$$R_i = \frac{R}{G+B}, \quad G_i = \frac{G}{R+B} \quad \text{and} \quad B_i = \frac{B}{R+G}$$

where R_i, G_i and B_i are the intensity feature values.

3.1.3. Mean μ and variance σ^2 of RGB, HSV, Grey.

A $N \times N$ window around the central pixel was taken, and for each of the seven RGB/HSV/Grey components, the mean and variance were calculated. Note that the mean is essentially analogous to the result of a homogenous convolutional blur operation, and is known to be effective against Gaussian noise since the deviations are normally distributed, and are relative to the original value.

3.1.4. Colour Histograms.

Histograms are used to count the quantity of pixel values from within a certain value range, for each of the seven elements (R, G, B, H, S, V and Grey). To keep the amount of features within acceptable bounds, four bins were used, corresponding to the intervals. This

results in 28 features.

3.2. Edges

The next set of features is related to the existence of directed changes in intensity located near the pixel, so-called *edges*.

3.2.1. Sobel edge detection.

The Sobel filter performs a measure on the approximate spatial gradient of the grey intensity at the centre pixel. The following convolution kernels are computed:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A$$

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

where A is the original image (in grey). The responses are combined using

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

The 2 features used were the response of the central pixel, and the mean response of the surrounding window.

3.2.2. Gabor filters.

The Gabor filter [5], like Sobel edge detection, is an approximate measure of spatial gradient at the centre pixel, but can be tuned for response to a wider variety of directions and frequencies. The convolutional kernel is computed by a Gaussian multiplied by a harmonic function. By distinguishing some predefined orientations and frequencies, a bank of kernels was obtained. We use the following formula

$$g(x, y; \lambda, \theta, \omega, \sigma, \gamma) = \exp(-\frac{{x'}^2 + \gamma^2 {y'}^2}{2\sigma^2}) \cos(2\pi \frac{x'}{\lambda} + \omega)$$

$$x' = x \cos\theta + y \sin\theta$$

$$y' = -x \sin\theta + y \cos\theta$$

and defined 20 Gabor filters by using wavelengths $\lambda \in 3.5, 4.2, 5.0, 6.0, 7.4$ and rotations $\theta \in 0, 0.25\pi, 0.5\pi, 0.75\pi$. Further, $\omega = 0.5$, $\sigma = 0.35\lambda$ and $\gamma = 0$. These values have been derived by Hubel and Wiesel [6] as compatible with biological vision, and have been found to be good values in practical applications [7].

The computational complexity of computing the response for larger frequencies is substantial. A practical remedy, that reduces computations by an order of magnitude, is the following:

1. shrink the whole image by ratio α;
2. convolute the image by an equally shrunken kernel;
3. enlarge the convoluted image by ratio α.

After the convoluted images were calculated, the following features were defined for each pixel: maximum response over surrounding window, average response over surrounding window and the rotation of maximum response.

3.3. Texture

For many applications of pixel annotation, specifically for example for the Tarmac task, information related to the texture may be relevant. One way of recognizing textures is by means of a library of know textures. However, in most of our applications, we can safely assume a certain level of randomness, which forces us to use features that describe more general properties of the local texture. The following features were used.

3.3.1. Discrete Cosine Transformation.

One way to describe the texture of a surface is by using the *Discrete Cosine Transformation* (DCT). This transformation describes a function (or a signal) by the sum of cosines with different frequenties and amplitudes.

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}(n + \frac{1}{2})k\right] \quad k = 0, \ldots, N-1$$

where N is number of pixels in the transformation input vector. According to Chiang and Knoblock [8], using a 2-dimentional DCT is an effective feature in classifying texture characteristics. The DCT function we implemented is equivalent to the transformation used by Chiang and Knoblock.

$$X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos\left[\frac{\pi}{N_1}(n_1 + \frac{1}{2})k_1\right] \cdot \cos\left[\frac{\pi}{N_2}(n_2 + \frac{1}{2})k_2\right]$$

In this case $N_1 \times N_2$ describes the size of a 2-dimentional input vector around the pixel, for which the DCT is calculated. k_1 and k_2 are constructed in the same way as in the one-dimensional case, described earlier in this subsection.

Figure 2. The image from Fig.1. after applying the Tamura features Coarseness (left), Contrast (middle) and Kurtosis (right).

3.3.2. Tamura.

As described by Howarth and Rüger, Tamura *et al.* tried to construct a collection of texture features that correspond to the human visual perception [9]. Howarth and Rüger compared the six features constructed by Tamura, by psychological measurements.

Coarseness is defined as the most powerful feature, introduced by Tamura et al. Images contain textures of several scales. The coarseness feature aims to identify the largest scale of the texture. This is done by taking the average over all *grey*-values for a number of neighbourhoods, and calculating the difference between two pairs of non-overlapping neighbourhoods, each on opposite sides of the pixel to classify. The average pixel value in a neighbourhood is calculated using the following equation:

$$A_k(x,y) = \sum_{i=x-2^{k-1}}^{x+2^{k-1}} \sum_{j=y-2^{k-1}}^{y+2^{k-1}} I(i,j)/2^{2k}$$

where $I(i,j)$ is the *grey*-value of the image at coordinate (i,j) and k is the size of the neighbourhood. All neighbourhoods are of size $2^k \times 2^k$.

The difference between two sets of neighbourhoods is calculated in the following way:

$$E_{k,h} = |A_k(x-2^{k-1},y) - A_k(x+2^{k-1},y)|$$
$$E_{k,v} = |A_k(x,y-2^{k-1}) - A_k(x,y+2^{k-1})|$$

Here, the k value that maximizes the maximum value of $E_{k,h}$ and $E_{k,h}$ is used as input for the model.

Besides the *Coarseness* feature, we added the Tamura *Contrast* feature, as described by Howarth and Rüger [9]. This Contrast feature aims to capture the dynamic range of grey levels in an image, together with the distribution of black and white. For calculating the dynamic range of grey levels, the standard deviation is used, and the *kurtosis* α_4 is used to calculate the distribution of black and white. Combining both parts gives us the following *Contrast* measure:

$$F_{contrast} = \sigma/(\alpha_4)^n$$

In this formula the kurtosis α_4 is calculated by dividing the fourth moment around the mean *grey*-value by its variance squared. As our final texture feature we use this definition of *Kurtosis*:

$$\alpha_4 = \frac{(I(i,j)-\mu)^4}{\sigma^4}$$

Here, μ is the mean *grey*-value of the image and σ is the standard deviation.

The effect of the three Tamura features on the test image (Fig 1) is demonstrated in Fig. 2. As can be seen, especially the Kurtosis feature makes sense for the recognition of road surface.

3.4. 3D Information

As explained in the introduction, we use the Computer Vision package Harmonii for extracting 3D information from stereo images. For each pixel in the image, the 3-dimensional world coordinates are calculated, as well as a scalar reliability measure. The reliability measure is used to reduce the feature's sensitivity to noise, e.g. by taking a weighted mean instead of the true mean.

3.4.1. Height and Depth.

For the centre pixel, the height and depth were taken as features. From the surrounding window, the mean and variance were calculated for height and depth values. These features were calculated for the standard window size, and twice the standard window size, resulting in eight features per centre pixel.

3.4.2. Fitted Vertical Plane.

Least-squares linear regression was used to fit a vertical plane to the window, using

$$\hat{\beta} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

where $\bar{x}$ and $\bar{y}$ are the average x and y coordinates, and $\hat{\beta}$ is the optimal slope. The MSE(*mean squared error*) of the model is calculated as the variance from the model. Exposed to a window pointed at a vertical surface, such as a wall or a traffic sign, this method should find the correct slope, and variance from this slope should be low, relatively to non-vertical slopes.

4. Experiments

As mentioned, the Harmonii system was used to compute the different features from the stereo images and produce the necessary data files. As our classifier, we

have selected the popular decision tree algorithm C4.5, as implemented by the J48 component of the Weka [10] learning environment built by the University of Waikato (New Zealand). Because Weka classifiers are programmed in Java, the resulting classifier could be easily integrated into Harmonii. Although C4.5 is a fairly standard algorithm, with some known limitations, it is useful in our setting because of its ease of interpretability. We like to stress that, depending on the specific pixel annotation task at hand, other classifiers may be a better choice. In our implementation, there are no limitations for using alternative classifiers from the Weka package.

4.1. Input

For each task, a number of (stereo) images was selected that are typical for the task at hand. Each stereo image consists of a left and right image, as well as a binary annotation image, as demonstrated in Fig. 1. The features related to the 3-dimensional information used both stereo images. For the intensity and colour-related features, only the left image was used. Each image was of resolution 640×480, which in theory produces $307,200$ data points per image. For reasons of efficiency, only every 5th pixel was computed and exported to file. Note that although not all pixels were computed, the neighbourhood and texture information still was computed on the highest possible resolution. Together with boundary pixels being ignored this resulted in $122 \times 90 = 10,980$ data points per image. Using Harmonii, all features were computed and exported to the *ARFF*-data files required by WEKA. There were 7 images available for the Tarmac task, and 5 for the Sign task, which resulted in two datasets of sizes $76,860$ and $54,900$. In the Tarmac dataset, 34.97% was assigned positive, and in the Sign dataset 3.02%.

As cross-validation approach, we have opted for a strategy that is somewhat different from the usual approach. Rather than dividing the dataset in n random subsets of equal size, we divide according to the source image. This means that a classifier is trained on the data resulting from $n-1$ images and then tested on the data from the remaining image. This process is then repeated for each image, and results are averaged. As such, we do not just test the normal generalisation capabilities of the classifier, but also the generalisation from one situation to the next. Note that there may be many changes between images depending on the weather and lighting conditions. We require our classifiers to deal with such variations.

Table 1. Results (in % and standard deviations between brackets) on the two different task.

	without 3D	with 3D
Tarmac	92.0 (± 5.3)	92.3 (± 6.0)
Sign	99.5 (± 0.147)	99.5 (± 0.147)

4.2. Results

The results of building and cross-validating the classifier on the two task are shown in Table 1. An immediate conclusion from this table is that both tasks show good performance. The Tarmac task shows a considerable increase, from a baseline of 65.03% to 92.3% (with all features included). The Sign tasks shows a near perfect 99.5% on average, but we have to take into account a baseline of 96.98% (signs are relatively small and do not appear that often). Still the increase is considerable, given that highly skewed target distributions are notoriously hard in machine learning.

As can be seen from the table, the effect of adding 3D features has at most a marginal effect. In the Sign task, no clear effect could be observed. It should be noted that this effect is not so much due to the unimportance or unreliability of the 3D features, but rather to redundancy in the large set of features produced. In the case of Tarmac, most pixels below the horizon actually belong to the ground. This means that both the 2D information (where in the image?) as well as the 3D information (where in the outside world?) is equally informative. Only where obstacles appear below the horizon, but are not directly on the ground, the 3D features may be more predictive. In fact, in the Tarmac task, the 3D features do appear in decision trees, just not always as the first split. For the Sign task, the classifiers are quite simple, and only use the saturation of the colour, and the 2D information. We would like to stress that for other tasks, the 3D features may become more important than any of the other features, for example when the distance from the observer is relevant. This is information that in general can only be obtained using stereo vision.

The colour and 2D information appear to be important in both selected tasks. In the Tarmac task, the 2D information was most relevant, followed by colour and 3D information. Finally, texture information was least important in this task. As mentioned, colour was most relevant for the Sign task, followed by 2D information. In all cases, almost equally performing classifiers can be obtained by removing some of the features used, hinting to the level of redundancy amongst the features. On the whole, the edge features did not appear

Figure 3. The results of applying a classifier to one of the images in the Tarmac task.

to be very informative.

By informal inspection of the classified images, a sense can be obtained of the types of mistakes the classifiers tend to make. On the whole, false positives are more frequent than false negatives. This means that, for example, areas outside the actual road may sometimes be labelled 'road', but on the road itself, very few mistakes are made. Often, the miss-classifications appear in areas where the annotation is arguable anyhow, such as pavements that are made of material very similar to tarmac. Surprisingly, the classifiers seem to have little trouble with complications such as road markings or shadows on the road, nor did the classifier struggle with the multiple colours appearing on the signs.

4.3. Path Planning in Autonomous Vehicles

As a simple demonstration of how pixel annotation can aid larger tasks based on Computer Vision, we show how the tarmac-identification classifier can be employed in an automotive setting. In Fig. 3, we see a scene with a road and a number of obstacles, as observed from an autonomous vehicle (left picture). Additionally, we see where a trained classifier thinks areas of road surface are (right picture). Purely based on 3D information, the navigation system (included in Harmonii) would identify the relevant obstacles, in this case three trees, and suggest any path that does not lead to collisions. In this case however, following the road is more urgent than avoiding obstacles. A straightforward solution to this is to simply look up the 3D information for each non-tarmac pixel in the classified image, and treat these as possible obstacles also.

As a more challenging setting, assume we have only a single camera, and thus no depth information. On first sight, it seems that we have insufficient information to follow the road, as we do have reliable classification, but no way to translate this into 3D information. However, by adding a simple assumption, we can obtain this information. If the vehicle is situated on (or near) the road, and can identify this road, then we

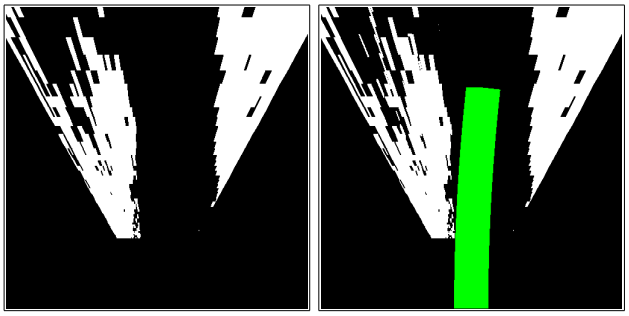

Figure 4. The results of applying the inverse perspective tranformation to the classified image, and planning a trajectory for a mobile robot.

can safely assume that the road is in a horizontal plane at some measurable distance below the camera. This means that with a simple transformation of the 2D image, we can map the identified tarmac on the horizontal plane. This operation is known as the *inverse perspective* transformation [11], and works as follows:

$$x = \frac{g \cdot x'}{y'}$$
$$y = g$$
$$z = \frac{c \cdot g}{y'}$$

where x, y and z are the induced 3D coordinates, x' and y' are the coordinates in the image (origin in the middle of the image on the horizon). The camera is assumed to be positioned horizontally. c is a constant related to the field of view and resolution of the camera. g is the measured height from the ground in meters.

The effect of this transformation is displayed in Fig. 4 (left). Note that the inverse perspective basically skews and stretches the original image. In Fig. 4 on the right the effect of simply navigating on the classified and transformed tarmac image is demonstrated. Clearly, the system suggests a path in 3D that is based on information from a single camera.

5. Conclusion

We have presented a method for pixel annotation: classifying regions in images into predefined discrete classes, such as 'road' and 'non-road'. Our method relies heavily on a collection of features extracted from the original image(s). These features capture different properties of the pixel itself, but also of the context it appears in (built up by the neighbouring pixels). These features can be divided into a number of groups: colour, edge, texture and location features. For the last group, we have 2D information concerning the location of the pixel in the image, as well as

3D information about the likely location of the pixel in the outside world. This 3D information was obtained by using stereo images, and computing the 3D information in the Harmonii Computer Vision system [2].

Experiments show that, even with a straightforward classifier, reliable results can be obtained for realistic pixel annotation tasks. We have tested two tasks in an automotive setting and obtained good results in both. This is not trivial, as the classifier has to deal with a lot of complexities of real-life situations. These include natural variations in the situation itself, but also in the lighting and weather conditions. By choosing a number of images that show these variations, robust classifiers could be obtained. It turns out that most of the mistakes made by the classifiers are logical: the actual annotation is ambiguous, or humans would also have a hard time making correct predictions without domain knowledge or using context information.

As was demonstrated in Section 4.3, the pixel annotation module can be a useful component in larger systems, such as autonomous vehicles. The task of navigating along a road can be facilitated by simply recognising the road-surface, and suggesting a path that respects this surface. Note that even without having actual 3D information, for example because only a single camera is available, some spatial reasoning can be done using the classifier's output (see Fig. 4). It turns out that the set of features used is overcomplete, and if certain information is lacking (e.g. spatial) other features may reliably take their place.

References

[1] Alvy Ray Smith. Color gamut transform pairs. *Computer Graphics*, 12 (3), 1978.

[2] Harmonii. `http://www.kiminkii.com/harmonii.html`.

[3] R. Marée, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, pages 34-40. IEEE, June 2005.*

[4] G. Carneiro, A. Chan, P. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No. 3, March 2007.*

[5] P. Kruizinga and N. Petkov. Non-linear operator for oriented texture. *IEEE Trans. on Image Processing*, 8(10):1395–1407, 1999.

[6] D. Hubel and T. Wiesel. Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *J Neurophysiol*, 28:229–89, 1965.

[7] Stanley Bileschi and Lior Wolf. *A Unified System For Object Detection, Texture Recognition, and Context Analysis Baed on the Standard Model Feature Set.* The Center for Biological and Computational Learning, Massachusetts Institute of Technology, 2002.

[8] Yao-Yi Chiang and Craig A. Knoblock. Classification of line and character pixels on raster maps using discrete cosine transformation coefficients and support vector machine. *ICPR - Proceedings of the 18th International Conference on Pattern Recognition*, 2:1034 – 1037, 2006.

[9] P. Howarth and S. Rüger. Robust texture features for still-image retrieval. *Vision, Image and Signal Processing, IEE Proceedings*, 152:868 – 874, 2005.

[10] Weka. `http://www.cs.waikato.ac.nz/~ml/weka/`.

[11] T. Schouten and E. van den Broek. Inverse perspective transformation for video surveillance. *Computational Imaging VI : proceedings of SPIE, vol. 6814. SPIE, the International Society for Optical Engineering*, 2008.