# Fast Color-Based Object Recognition Independent of Position and Orientation

Martijn van de Giessen and Jürgen Schmidhuber

IDSIA, Galleria 2, 6928 Manno (Lugano), Switzerland &
TU Munich, Boltzmannstr. 3, 85748 Garching, München, Germany

**Abstract.** Small mobile robots typically have little on-board processing power for time-consuming vision algorithms. Here we show how they can quickly extract very dense yet highly useful information from color images. A single pass through all pixels of an image serves to segment it into color-dependent regions and to compactly represent it by a short list of the average hues, saturations and color intensities of its regions; all other information is discarded. Experiments with two image databases show that in 90 % of all cases the remaining information is sufficient for a simple weighted voting algorithm to recognize objects shown in query images, independently of position and orientation and partial occlusions.

## 1 Introduction

Small, fast, vision-based mobile robots must process many images per second to react in time. It does not matter so much if some object sometimes is not properly recognized the first instant it is seen, provided the robot's sequential vision system (SVS) can use subsequent camera shots to incrementally increase its confidence about whether or not the object is present in its visual field. In principle such an SVS for dealing with uncertainty and noise can be implemented by Bayesian sequential decision makers or learned by recurrent neural networks [1,2,3].

The image pre-processor should be able to quickly produce a compact yet informative description of the current image, to be fed into the SVS. So we are interested in fast algorithms that often (but not necessarily always) produce image descriptions containing all the information necessary for decent object recognition. Of course, the more reliable the pre-processor, the less burden on the SVS.

Many previous approaches to object recognition are computationally too demanding for the limited on-board computers of mobile robots, or permit only small changes in object position and orientation, and few if any occlusions. Here we propose a simple, fast, and rather reliable method based solely on the number of image regions with similar color.

In what follows we will describe two methods, a fast one for image processing and coding (Section 2), and another one for demonstrating that the image codes convey sufficient information for object recognition, given a database of images

W. Duch et al. (Eds.): ICANN 2005, LNCS 3696, pp. 469–474, 2005.
© Springer-Verlag Berlin Heidelberg 2005

(Section 3). The latter searches for a match in the database by judging similarity between the query image and database images based on weighted votes. Performance in terms of recognition rate and speed is evaluated in Section 4.

## 2 Processing Images

### 2.1 HSV Images

We represent images in HSV color space, which provides a good distinction between three properties of a color: hue, saturation and illumination (value). These properties are related to each other in Figure 1. HSV codes provide simple ways of filtering out non-reliable color information conveyed by areas with low illumination or low saturation (e.g., almost grey areas). The shaded parts in figure 1 contain such non-reliable colors.

### 2.2 Region Extraction

Since every step in the recognition process should be fast, we use a very simple region extraction algorithm loosely inspired by the intensity-based method proposed by Tuytelaars and Van Gool [4]. From the top left to the bottom right of an image, every pixel $j$ is compared to the regions of its upper and left neighbor, if these neighboring pixels exist (at borders the pixel $j$ is only compared to the regions of its upper or left neighbor, depending on which pixel exists). We ask whether the differences between the hue ($h_j$), saturation ($s_j$) and value ($v_j$) of pixel $j$ and the average hue ($h_i$), saturation ($s_i$) and value ($v_i$) of region $i$ are smaller than the thresholds $t_h$, $t_s$ and $t_v$, respectively:

$$|h_j - h_i| < t_h, |s_j - s_i| < t_s, |v_j - v_i| < t_v \tag{1}$$

Note that the hue describes a circle as in figure 1. The pixel $j$ is added to the most similar region or, if inequality 1 does not hold for both regions, a new region is formed. When the pixel $j$ is added to a region, the other region adjacent to $j$ is merged with the region containing $j$ if $|P_j - P_k| < t_p$ holds for hue, saturation and value. $P_j$ and $P_k$ again stand for the average hue, saturation and value of the region containing pixel $j$ and the region adjacent to pixel $j$ respectively. Comparing to the average characteristics of the neighboring region instead of to the neighboring pixel has the advantage that more coherent regions are generated. This is due to the fact that the average values change more slowly when a region grows, so pixels in fast changing gradients are not added to a region.

After all the regions have been extracted, regions with a very small area (e.g. less than 50 pixels) and regions with their average saturation and value in the shaded regions of figure 1 are discarded. One reason why small regions are abandoned is that there is a high chance that they will not appear on a picture of the same object on a different scale or viewed from a different angle. Another reason is that smaller regions are more sensitive to the distortions caused by

pixels on the edge of that region. We save the average hue, saturation and value of every region in a database, since this information is completely position and orientation independent.

## 3 Querying Images

### 3.1 Initial Selection

The following object recognition procedure is not mandatory for mobile robots that just need to compactly encode images and feed them into an SVS-based controller, without having to match them against a database. But it serves to demonstrate that the image codes retain essential information about the depicted objects. It may be possible to further speed up the straight-forward procedure below, e.g., by rearranging the database in hierarchical form.

To search for images of objects similar to the one in a query image, we first extract the latter's regions as above. Every region in the query image is compared to all regions of all images in the database. To avoid wasting time on computing complex distance measures between very different regions, we first discard those that are clearly dissimilar to the queried region. This is done in a way similar to the one of Nene and Nayar [5]: All regions are considered as points in a 3D space with hue, saturation and intensity (value) on the axes. A box is computed with the query region in the center and the sides perpendicular to the three axes. Database regions outside this box are discarded. Thus groups of similar regions are formed for every region in the query image.

### 3.2 Weighted Voting

All database regions $i$ within each group get a weighted vote $w_{ij}$. This vote is determined by the distance

$$d_{ij} = 5^{-\frac{1}{2}}((v_i - v_j)^2 + (s_i \cos h_i - s_j \cos h_j)^2 + (s_i \sin h_i - s_j \sin h_j)^2)^{\frac{1}{2}} \quad (2)$$

in color between a region $i$ in the database and a region $j$ in the query image. We compensate for the distinctiveness of a query region and the complexity of a database image. The distance between the average hue, saturation and value of two regions is computed as in [6]. The distinctiveness of a region is determined on the basis of how many regions in the database survive after the initial selection in section 3.1. The larger the number of regions in the box, the less distinctive the region. To give more weight to more distinctive regions, we divide by the number of remaining regions, $n_{similar}$. It is also useful to compensate for 'complex' images with many regions, which have a greater chance of featuring a region close to a query region. That is why we divide by the number of regions in the image containing this region, $n_{rpi}$. The total vote per region becomes

$$w_{ij} = \frac{1 - d_{ij}}{n_{similar} \cdot n_{rpi}} \quad (3)$$

The total vote for an object is given by the sum of the weighted votes for all regions of that object.

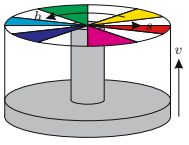


**Fig. 1.** The HSV color space in cylindrical form. Gray areas contain non-reliable color information.

**Fig. 2.** The color information (hue) in low quality JPEGs is unreliable (left), compared to this information in a high quality image of the same object (right).

## 4 Experimental Results

### 4.1 ZuBuD Buildings Database

We tested both recognition rate and speed on a databases containing real-world images, resampled to $320 \times 240$ pixels, that were not made under special simplifying conditions. Our first choice was the ZuBuD database [7] containing 1005 pictures of 201 buildings in Zürich, taken outside and under varying weather conditions. The database comes with 115 query images (low-quality jpegs, resolution $320 \times 240$ pixels). For every queried image, we list the top five matches produced by our algorithm. Despite the low quality of color information in the queries, 82 images are recognized correctly, 29 images are within the top five, and 4 outside the top five. The color information in the query images is of a very low quality, as can be seen clearly in Figure 2, where the hue from a query image and a database image from the same building are shown next to each other. This lack of reliable color information prevented a correct recognition. The query image in the top row of Figure 3 exemplifies our algorithm's insensitivity to obstructions, such as trees.

To test performance on high-quality images (easily producible on small mobile robots), we built a new database containing 4 images of each of the 201 objects, using the 5th image of every object as a query image. The improved image quality led to substantially improved recognition rate: 183 objects correctly recognized, 13 in the top 5 matches, and only 5 out of 201 outside the top 5. The second row of figure 3 shows our algorithm's insensitivity to orientation and position.

The reasons for the five failures and some of the non-perfect results in the top five seem to be twofold. One reason is that these database and/or query images mainly have regions with very low saturation. The second is that some of these query images contain only a rather small part of the object. The last row of figure 3 shows one of the misses due to low saturation. For a learning robot with an SVS based on adaptive recurrent neural networks [3] these misses will not pose a big problem, since they will be identified as noise by the learning algorithm.

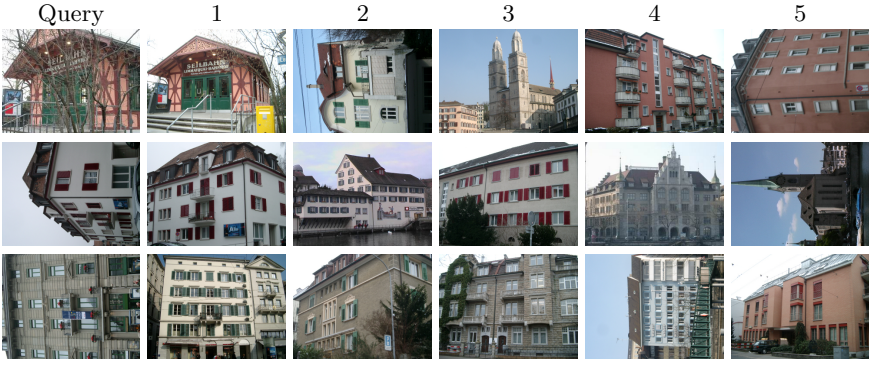


**Fig. 3.** Three examples of query images (left) and the top five similar objects according to our recognition algorithm. The top row shows the robustness against occlusions, the middle row shows the independence of position and orientation and the third row shows a miss, because of the low saturation of the object in the query image.

### 4.2 Coil-100 Object Database

Our method is *not* specialized on the ZuBuD database. It is designed to be widely applicable. We hardly tuned any parameters; one just has to select the tresholds of Section 3.1 as small as possible to speed up the recognition process. To illustrate the method's generality, we also applied it to the Coil-100 database [8], which contains 100 objects photographed from various angles. We placed images of objects taken under angles 0, 100, 215, 270 and 325 degrees in the database and used the 25 degree views as query images. Results: 84 recognized objects, 12 in the top 5, 4 outside the top 5. Again the non-recognized images mainly contain areas in various shades of gray, discarded for their poor color information.

### 4.3 Performance in Terms of Speed

We report results for a standard Pentium 4 2.8 GHz machine. The two time consuming parts of the recognition process are region extraction and the search for similar regions in the database. The latter is not mandatory for robots that just need to compress the relevant information for an SVS. The speed of the former depends on image complexity, but even complex images like those in the ZuBuD database were processed within 0.19s.

Database search speed linearly depends on the number of regions in the database and on the number of regions found in the query image. The database for the second experiment contains 93682 regions. On average 177 regions are extracted from an image in the ZuBuD database. An average search takes roughly 0.7s, including region extraction in the query image. Simple objects speed up the process: Recognizing one of the 100 objects in the Coil-100 database takes roughly 0.18s, including query image processing.

## 5 Conclusions

We propose a simple, fast, rather reliable algorithm for image coding and recognition. It uses a simple color-based region extractor and an object matcher based on weighted voting. The former works in HSV color space, discarding regions with unreliable color information, keeping only reliable, position and orientation independent color data to encode objects. Image similarity is measured by votes whose weights depend on the similarity between the regions of a query image and database images; we compensate for the number of similar regions and regions per database image. On the ZuBuD and Coil-100 databases we obtain satisfactory recognition rates and speeds.

It should be noted, however, that this work is quite preliminary; more in depth studies are necessary to compare our algorithm to previous proposals in the literature. This is the subject of ongoing work.

## Acknowledgements

We would like to thank Hao Shao for supplying code used by his alternative color-based image recognition algorithm [9]. It gave us a fine starting point for the development of the present algorithm.

## References

1. Williams, R.J., Zipser, D.: Gradient-based learning algorithms for recurrent networks and their computational complexity. In: Back-propagation: Theory, Architectures and Applications. Hillsdale, NJ: Erlbaum (1994)
2. Pearlmutter, B.A.: Gradient calculations for dynamic recurrent neural networks: A survey. IEEE Transactions on Neural Networks **6** (1995) 1212–1228
3. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9** (1997) 1735–1780
4. Tuytelaars, T., Van Gool, L.: Wide baseline stereo matching based on local, affinely invariant regions. In: British Machine Vision Conference. (2000)
5. Nene, S.A., Nayar, S.K.: A simple algorithm for nearest neighbor search in high dimensions. IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997)
6. Smith, J.R., Chang, S.F.: Visualseek: A fully automated content-based image query system. In: ACM Multimedia. (1996) 87–98
7. Shao, H., Svoboda, T., Van Gool, L.: ZuBuD — Zürich buildings database for image based recognition. Technical Report 260, Computer Vision Laboratory, Swiss Federal Institute of Technology (2003) Database downloadable from http://www.vision.ee.ethz.ch/showroom/.
8. Nene, S., Nayar, S., Murase, H.: Columbia object image library: Coil-100. Technical Report CUCS-006-96, Department of Computer Science, Columbia University (1996)
9. Shao, H., Svoboda, T., Tuytelaars, T., Van Gool, L.: Hpat indexing for fast object/scene recognition based on local appearance. In Lew, M., Huang, T., Sebe, N., Zhou, X.S., eds.: Computer lecture notes on image and video retrieval. LNCS 2728, Springers (2003) 71–80