8

Accurate Color Classification and Segmentation for Mobile Robots

Raziel Alvarez, Erik Millán, Alejandro Aceves-López and Ricardo Swain-Oropeza
Tecnológico de Monterrey, Campus Estado de México
Mexico

1. Introduction

Visual perception systems are fundamental for robotic systems, as they represent an affordable interface to obtain information on different objects in the environment for a robot, and because they emulate the most commonly used sense in humans for world perception.

Many techniques can be used to identify an object within an image. Some of these techniques are color object identification, shape detection and pattern matching. Each one of these techniques has different advantages; however, color based techniques are usually preferred in real-time systems, as they require less computing power than other approaches.

Color object identification is composed by two phases: image segmentation, and object identification. The goal of the first phase is to identify all regions of the image that belong to the same object of interest. These regions are analyzed by the second phase in order to extract features of interest from these objects like geometry and relative distances and to infer the presence of a specific object.

Color image segmentation relies highly in the identification of a set of colors. Hence, color classification, which consists on identifying a pixel as a member of a color class, is essential for this process. In this chapter a technique for color image classification and its application for color segmentation will be explained in detail.

This chapter will start by presenting a set of general concepts on image processing, which will simplify the understanding of the rest of the chapter. Then, in Section 3, some of the existing approaches used for color image classification, as well as some of their advantages and drawbacks, will be described. Section 4 will describe an efficient technique for accurate color classification of images using implicit surfaces. In Section 5, it will be explained a color segmentation technique based on custom tolerance of color classification. Finally some applications will be presented in Section 6, and conclusions and future work will be discussed in Section 7.

2. Background on Image Processing

2.1 Images and pixels

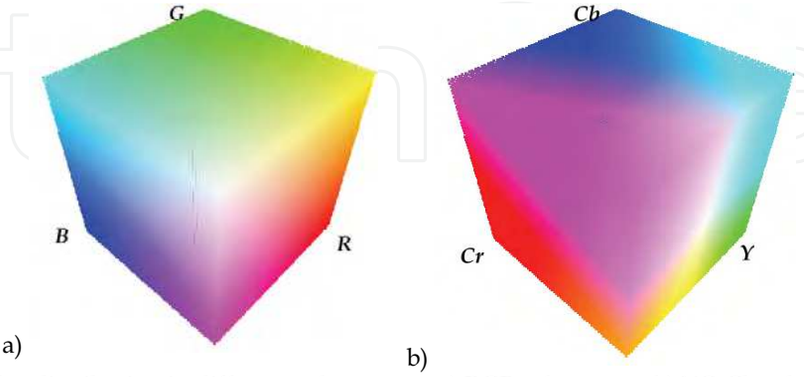
An image is the graphic representation of something, usually from a 3D world, in a 2D space. That image can be defined as a function, $f(x, y)$, where x and y are spatial coordinates, and the value of f denotes the intensity in such coordinates. In particular, image processing is concerned with digital images, which contain a discrete number of x, y locations and of f values. Therefore, a

Source: Mobile Robots: Perception & Navigation, Book edited by: Sascha Kolski, ISBN 3-86611-283-1, pp. 704, February 2007, Plv/ARS, Germany

digital image is composed of a finite set of elements, each with a particular position and an associated value or set of values. Each one of these elements is referred as picture element, or *pixel*. Hence, a digital image can be considered as a two-dimensional array of pixels.
The number of values used to describe a pixel depends on how much information is used to encode the color for such elements of the image. In the case of grayscale images, a single value is used to describe the intensity of the pixel. In the case of color images, three values are usually required, f_1, f_2, f_3, indicating the intensity of different color components (i. e. primary colors) that combined will produce the color perceived by our eyes. These components will depend on the used color space.

2.2 Color spaces

A *color space*, also known as *color signal*, defines a set of attributes that uniquely identify a color. Color spaces are then important, as they set the distribution that colors present for different objects, which is fundamental in color classification and color segmentation. In addition, each color space provides with features that may be better suited for specific problems, such as varying lighting conditions or noisy environments.
Color spaces can be classified in linear and non-linear (Forsyth & Ponce, 2002). Linear color spaces are based on a set of primary colors, and describe colors in terms of a weighed sum of the intensities of these primary colors.
For instance, the RGB color space used in computer monitors describes colors in terms of three primary colors: red, green and blue. A linear combination of these components will produce all the colors that can be shown in such screens.
Another linear color space is YUV, or YCrCb. In this color space, the Y component express the brightness of the pixel, while the U and V components define their chromaticity, in terms of the amounts of blue and red, respectively. This color space is common in video cameras. Color distribution in both RGB and YCrCb color spaces is shown in Figure 1.
In contrast, non-linear color spaces can include more properties of color spaces that may help humans or different image processing techniques to better describe colors. Properties used in these spaces include tone, saturation, and intensity or brightness. *Tone* can be defined as the property that describes the way a color changes from other colors, *Saturation* describes how a color is faded by a white light, and *Intensity* or *Brightness* specifies how bright the pixel is, no matter the color. An example of a non-linear color space is HSL, which describe colors in terms of Hue (denoting tone), Saturation and Lightness (describing Intensity).



a) b)

Fig. 1. Color distribution in different color spaces. a) RGB color space. b) YCrCb color space.

2.3 Color image segmentation

Color image segmentation can be defined as the process of decomposing an image into regions with certain meaning according to the contents and the application for that specific image (Watt & Policarpo, 1998). Here, a region is a set of connected pixels sharing a set of attributes.

Classic segmentation techniques can be divided in global approaches, region-based approaches, and edge-based approaches.

Global, or threshold, approaches rely in the knowledge of pixel attributes. Thus, it is required to provide with a set of attributes that bind the classes that must be identified within an image. In the case of color images, this approach uses the color space as a 3D domain over which a set of groups or clusters will be identified for each color class. This technique is simple and efficient, but depends heavily on a good threshold definition.

Region-based approach consists in dividing an image in a set of regions that present similar properties. Techniques using this approach usually start by growing a region from an initial pixel, or *seed*, and expanding this region according to a set of homogeneity criteria. This approach presents two general problems. First, an initial seed should be picked, and this requires an additional process. And second, it is usually hard to define and parameterize the homogeneity criteria. An incorrectly defined homogeneity criterion may lead to *flooding* problems, where regions grow over the visible boundaries of the region, or to prematurely stop the growth process.

The edge-based approach uses edge detection to find a closed boundary that defines what lies inside and outside a region. The hypothesis in which this approach relies is that pixels in the boundary between two regions should be considerably different, regarding properties such as color or intensity. However, problems are produced in blurry areas of an image, where colors are not very contrasting. In addition, a problem with this approach is that failures are common when detecting closed boundaries, as borders are usually discontinuous.

3. Different approaches for color classification

A set of pixels forming a specific color image correspond to a specific set (or cloud) of points in the color space. Color classification needs to pre-define the geometry of sub-space (class), in which all contained points share the same property. Simpler the geometry of subspace is, easier the classification of pixels is but with a high risk of misclassified pixels.

There are many existing techniques to define, create and calibrate different color classes used to segment an image. Most of them try to fulfill two basic assumptions. First, resulting color classes should precisely define objects in the environment, having good generalization properties for conditions not considered in the class definition, but avoiding excessive generalization that may produce false positives. Second, color classes should be reliable enough to identify objects under variable light conditions. Definition of a color class is a complex task that usually involves adjusting a volume to a cloud of samples obtained from pixels belonging to objects of such color class. This presents a clustering problem, which may be solved either by using traditional or simplified methods. Specifically, by using simplified methods, less processing power from a mobile robot is need, allowing its use for online classification and calibration. The use of thresholds for color classification is a process that involves partitioning a color space. As presented by Castleman (Castleman, 1996), different objects on an image belong to individual groups or *clusters* of points in a histogram defined on the color space. These groups of points represent the samples from which color classes will be constructed and defined.

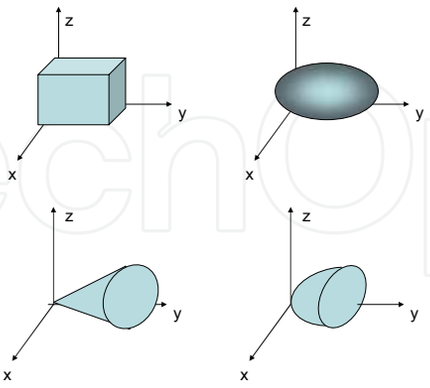


Fig. 2. Top left: Parallelepiped subspace, Top right: Ellipsoid subspace, Bottom left: Conic subspace, Bottom right: Paraboloid subspace

Color class construction process starts by taking color samples from objects in the environment. This sampling process consists on extracting the components of the color signal for each pixel in the objects, using images captured by the camera of the mobile robot for which the color classification is intended. These images are obtained trying to include most of the possible external conditions that the robot may find. Classic choices of sub-spaces geometries are shown in Figure 2.

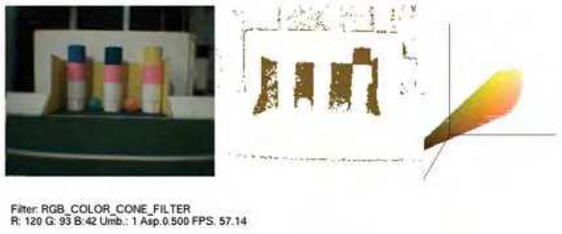
One of the simplest techniques to define a color subspace is by setting a pair of thresholds for each component of a color signal. Knowing that linear color space is Cartesian, the subspace defined by these six thresholds (two for each coordinate) will produce a parallelepiped. The implementation of this approach produces a fast classification of pixels in the image (Bruce et al., 2000). The main advantage of this method lies on its simplicity and speed to define color subspaces, which is important if an online recalibration is expected. The most important drawback is that this volume adjusts poorly to the cloud of samples, leading to a poor classification of pixels and reduced generalization capabilities. This is due to a high risk of overlapping when many colors classes are needed, causing misclassification on pixels. A sample classification produced by this type of threshold is shown in Figure 3.a. As a result, a different subspace is need to better fit points of each color. Quadric surfaces are evaluated as a better option. These surfaces adapt more precisely to the apparent geometry of clouds of samples, for the particular color spaces used in this research. While a cloud of points in a color space has some spatial characteristics, the same has other spatial characteristic when other color space is considered. Different color spaces may benefit from other subspaces representations, as the shape of a color class may change according to the attributes analyzed for a color. In order to exemplify quadric subspaces, both RGB and YUV color spaces were used in the color image of Figure 3. Cones are a logical choice in RGB color space. If a vector is traced passing through the origin, it will intersect a set of values with the same chrominance, but with a different brightness. By using this vector to create a cone, the tolerance of the contained range will include a set of similar tones of the classified color. The radius of the cone will define the maximum threshold that the color class will allow. A sample conic threshold is shown in Figure 3.b.

However, when samples are close to the origin of the color space —when colors become very dark—, samples are more susceptible of being affected by image noise. A similar quadric

subspace that avoids the classification of samples near the origin is a paraboloid. While its shape largely resembles a cone, providing similar color classification capabilities, a variation in the base of the paraboloid denotes a criterion to discard dark samples subject to noise. The focus of the paraboloid will be calculated according to the mean value of the cloud of samples, and may be modified to include or discard dark samples. Paraboloid thresholds may be seen in Figure 3.c.

a) Parallelepiped color class

b) Cone color class



c) Paraboloid color class

d) Ellipsoid color class

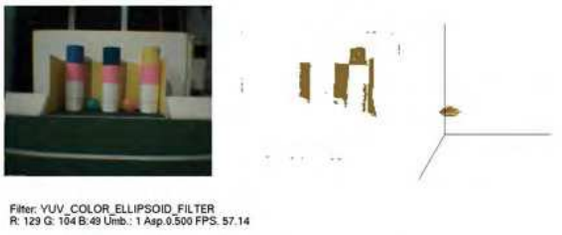


Fig. 3. Use of different bounding subspaces as color classes for classification of yellow in different color spaces. From left to right: Original image, Pixels from the original image classified as yellow, 3D representation of the color subspace. a) YUV Parallelepiped color class. b) RGB Cone color class. c) RGB Paraboloid color class. d) YUV Ellipsoid color class.

Again, an issue that both of these approaches may find is their difficulties in classification of highlights or bright areas. By creating an ellipsoidal primitive, color regions are bounded more precisely, discarding highlights that may also be caused by noise. The use of ellipsoids adapt stylishly to different color space, such as RGB and YUV, as they provide a more precise approximation to the final color class. This is shown in Figure 3.d.

4. Color Image Classification through Implicit Surfaces

Though techniques based on quadric subspaces produced better results than the use of simple parallelepiped subspaces, the resulting classification still haves some areas of improvement.

First, the bounding quality of the surfaces to the cloud of samples depends highly on the distribution of the classified color in the selected color space. For instance, cones and paraboloids are more accurate describing color distribution in the RGB color space than in the YUV color space. However, ellipsoids produce better results in both color spaces.

Second, even when the color space is suited for a quadric subspace, its may not adjust as tightly as desired to a color class, which may lead to some classification problems. This is because shape of clouds of samples is affected by light incidence and reflection.

Hence, a better technique of classification was proposed in (Alvarez et al., 2004) to overcome some of these drawbacks. This new approach is based on a technique used for 3D object reconstruction (Lim et al., 1995), and the use of implicit surfaces as the threshold that bounds and defines color classes.

4.1 Implicit Surfaces

Formally, implicit surfaces are 2D geometric shapes existing in the 3D space, and defined according to a particular mathematical expression (Bloomenthal, 1997). Usually, this implies a function *f* that defines a surface. When *f* is equal to a certain threshold for a given point, this point lies on the surface, while when a point is below this threshold, it is contained by this surface.

An implicit surface is usually characterized by a skeleton and a blending function. The skeleton is a set of primitive elements—such as points and lines— that define individual implicit functions which will be the base for the final surface. The blending function defines the way in which these primitives will be combined to produce the final surface.

Implicit surfaces were selected for color classification as they easily allow evaluating when a sample lies inside or outside the volume defined by such surface, providing a natural interface for color classification. Besides, they present the property of being able to blend with other surfaces in order to produce a single one. In this way, a set of relatively sparse samples may produce a continuous surface containing all the intermediate regions to produce a single color class. Similarly, if groups of samples within a color class are very distant from each other, the surface will split, but the resulting clusters will still belong to the same class.

In this way, it is possible to adjust a surface to a cloud of samples if we can properly distribute a set of primitives, of correct dimensions, that work as the skeleton of our surface;

similarly to the work from Lim et al. (Lim et al., 1995) to reconstruct a 3D object from a set of points from its surface.

4.2 Construction Algorithm

This algorithm starts from a set of images from which a user selects a set of samples for a given color class. Then, a number of spherical primitives are uniformly distributed along the cloud of samples, to later apply the *k*-means algorithm to adjust their position. Once distributed, the radius of each primitive is estimated according to the standard deviation of samples related to each primitive. Finally, these primitives are blend to produce the final surface, or color class. The resulting color class will then be translated into a look-up table, in order to produce an efficient online color classification. Another option is to use the resulting surface function as the classifier, defining a threshold for this function as a tolerance criterion to add or exclude samples.

4.3 Generation of Primitives

Once a set of color samples has been selected, a set of primitives should be distributed among this cloud. An initial naïve approach could be the use of Delaunay tetrahedralization (Langetepe & Zachmann, 2006), which produces a set of tetrahedrons that connects every sample in the cloud. Adding a spherical primitive in the center of each tetrahedron would then produce a relatively close approximation of the sampled data. However, the resulting number of primitives would be extremely large, and discarding small tetrahedrons would produce a poor approximation of the surface.

Instead (Lim et al., 1995) propose to start the surface reconstruction by minimizing a cost function, expressed as a sum of squared terms. This cost function represents the error between the implicit surface, including the desired features of this surface, and the points from which it is constructed. Some functions may include the distance between the surface and the points, the curvature, the normal vectors of the surface, or the radius of the primitives. However, minimization of such function is neither simple nor fast, and this method is suited for samples from the surface rather than within the surface. A possible solution would include extracting the samples on the surface, but discarding samples within the surface would eliminate potentially valuable information for the process. Instead, (Alvarez et al., 2004) propose a different approach using the *k*-means algorithm.

The *k*-means algorithm, as defined by Bishop (Bishop, 1996), divide a set of n point samples ($x_1, x_2, \ldots x_n$) in a set of k disjoint, non-hierarchic sets ($Q_1 \ldots Q_k$), through the minimization of a distance criterion d. Usually, Euclidian distance metric is used, which produces spherical clusters that, in this case, suit well as primitives for an implicit function. The *k*-means algorithm is a minimization problem of a cost function:

$$J = \frac{1}{2}\sum_{j=1}^{n} d^2\left(x_j, c_{g(x_j)}\right) \tag{1}$$

where

$c_{g(x_j)}$ is the center of the cluster $g(x_j)$

$g(x_j)$ is the closer cluster to point x_j

The total cloud of samples is seen as a large cluster that will be approximated by a number of small clusters, defined by k primitives. The k-means algorithm distributes the primitives in the volume used by the cloud samples, iteratively adjusting its position to provide each cluster with a subset of the samples.
The proposed k-means algorithm is summarized as follow.

Step 1. An arbitrary set of clusters (primitives) is distributed randomly along the cloud of samples.
Step 2. Calculate the distance metrics from each sample (x_1, x_2, … x_n) to the center of each primitive (c_1, … c_k).
Step 3. Relate each sample to the closest primitive.
Step 4. Recalculate the center of primitives as the average of all of the samples related with such primitive.
Step 5. Repeat iteratively steps 2-4 until the change in the position of all centers lies below a certain threshold ε.

This process guarantees that each sample will belong to a group or primitive, and that the distribution will converge to a local minimum of (1).
Once the primitives have been distributed using the k-means algorithm, the radius for each primitive is estimated as follow. For each group, the standard deviation from the center of the primitive to all samples is calculated. The radius of the primitive is then set as a multiple of this standard deviation according to the confidence interval expected.

4.4 Construction of the color class

Once the spherical primitives are defined, centers and radius are known, they will be combined into a implicit surface, as proposed in (Lim et al., 1995).
The implicit function for each spherical primitive i is defined as:

$$f_i(P) = \frac{d^2\left(P - c_i\right)}{r_i^2} \tag{2}$$

where

P is an arbitrary point in the color space.

This function has the following properties:

$f_i(P) < \Gamma$ for points P inside the surface
$f_i(P) > \Gamma$ for points P outside the surface
$f_i(P) = \Gamma$ for points P on the surface

Here, Γ is the threshold, which may be understood as a scale parameter for the sphere. For example, if Γ is set to 1 the original scale of the spherical primitive will be considered.
Primitives are then joined into a unique implicit function, using a blending function to construct the final implicit surface. Here, a function presented by Ricci (Ricci, 1973) is used to blend the primitives:

$$f = \left\{ \sum_{i=0}^{k} \frac{1}{f_i^{\rho}} \right\}^{-\frac{1}{\rho}} \text{ for } k \text{ primitives} \tag{3}$$

where

ρ is the blending degree

When ρ tends to infinity, this function converges to the absolute minimum. Graphically, this parameter controls how tight will the surface fit the primitives that compose it; when ρ tends to infinity, the function converges to the original set of primitives.
From the implicit function f , it is possible to precompute the color class and store it into a look-up table, which will permit a fast mechanism to perform online classification. However, the implicit surface function could also be used as the classifier, defining a threshold as required. Doing so will allow this approach to be more dynamic, and to adjust during the execution of online classification.
Moreover, the implicit function f can be seen as a possibility function, which provides more information about the classified sample other than its class, but also the degree of similarity between the sample and the color class, resembling a Gaussian mixture. Using other criteria —like Mahalanobis distance—would add different characteristics to this approach, which might be useful for certain application areas.
An example result for the yellow color class using this approach is shown in Figure 4.

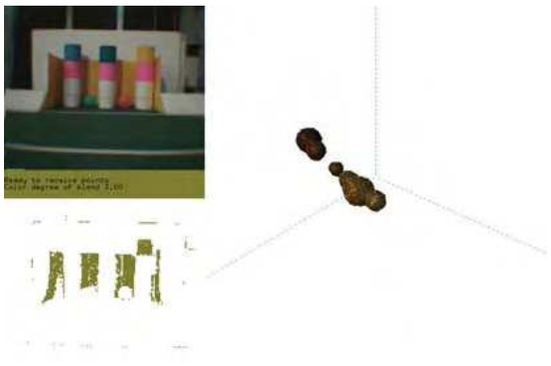


Fig. 4. Color classification using the implicit surfaces technique. Shape on the left is the resulting implicit surface for the yellow color class.

One advantage of the implicit surfaces technique is the simplicity to modify the scale parameter Γ for spherical primitives, and the blending degree ρ, in order to modify the amount of samples identified as part of a color class. By modifying these thresholds, a set of self-contained color classes is obtained. Classes with different thresholds may be used for different segmentation stages on the same process.

5. Multilevel Classification for Image Segmentation

In general, segmentation techniques consume a lot of computer power by processing every pixel on an image, increasing the possibilities of recognizing external noise as possible objects of interest.
To avoid processing the entire image, Seed Region Growth algorithms (or *SRG*, for short) use only a set of pixels or seeds which are very likely to be part of an object of interest (von Hundelshausen & Rojas, 2003; Wasik & Saffiotti, 2002). Then, these seeds are used to grow regions of interest based on a homogeneity criterion, such as contrast between neighboring pixels. However, these techniques require a good homogeneity criterion to avoid flooding problems.
A good way to analyze a small set of pixels with high probabilities of being part of an object of interest is to use information on the camera position and orientation to only evaluate

pixels located in areas of the image where objects can be present, and discard areas where no object can be located. In particular, the use of scanlines, which are perpendicular to the horizon, discards objects above a certain height and provide different resolutions for image scanning, according to the expected distance to objects (Jüngel, 2004). However, this method alone may discard information that might be found obtaining complete regions.
The presented approach combines the advantages of scanlines and SRG algorithms by using a multilevel color classification (Alvarez et al., 2005). This classification takes advantage of the custom threshold of the implicit surfaces presented in section 4.
First, a set of scanlines will be used to extract a set of color seeds. Pixels on scanlines will be identified using a small threshold color class to reduce the number of identified pixels. Then, extracted seeds will be used by a region growth algorithm to identify regions of interest. The SRG algorithm will use a color class with a bigger threshold to better characterize entire regions.

5.1 Seed Extraction through Scanlines

According to the approach from Jüngel (Jüngel, 2004), a set of vertical scanlines will be used to locate objects of interest within an image. Density of these lines will be different according to their distance to the camera. When a line is closer to the horizon, pixels on it will probably represent objects farther from the camera. As the line moves downward away from the horizon, pixels in the scanline will probably belong to objects closer to the camera, so a smaller line density should be sufficient to locate these objects.
In order to obtain the horizon for the camera on each picture, the kinematic model of the camera should be known. This is highly dependent on the nature of the robot system and cannot be generalized, but by obtaining this kinematic model, a lot of undesired data can be discarded from the processing pipeline.
Once the horizon is known, a scan pattern is projected on the image. The scan pattern consists on a set of lines, perpendicular to the horizon, that start above the horizon and go down the interest area. Distance between scanlines depends on the projected size of objects on the screen. As the projected size of objects grows as they get closer to the camera, scanlines should become sparser as they get below the horizon. An intertwined pattern of short and long scanlines will deal with this varying scanline density. An example of this pattern is shown in Figure 5.a. Pixels found in these scanlines will be classified according to a color class with a low threshold, as seen in Figure 5.b.

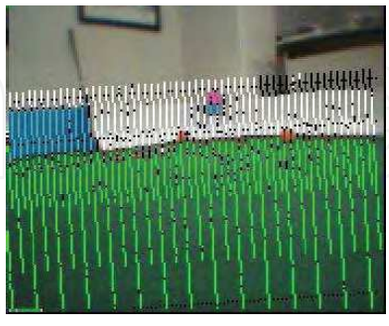
a)  b) 

Fig. 5. a) Scan pattern projected on a custom image. Scanline density is higher as pixels get closer to the horizon (dotted line). b) Seeds extracted from evaluating pixels on scanlines using a low-threshold color class.

The scanline algorithm avoids processing the entire image, reducing the number of processed pixels. In addition, this routine discards pixels above a certain height, eliminating possible sources of noise.

5.2 Seed Region Growth

Seed Region Growth is a region based technique used to perform image segmentation (Wasik & Saffiotti, 2002). A seed is selected from this initial set of color seeds. In our approach, the initial set of color seeds is obtained from the scanline seed extraction.

The selected seed is assigned to a new region, and this region is grown on a neighborhood. Usually, a 4-pixel neighborhood is selected, although an 8-pixel neighborhood can sometimes be used. When a neighboring pixel already belongs to the current region, it is ignored. If not, this pixel is evaluated through the homogeneity criterion to find if it belongs to the current region. If it does, this new pixel is added as a new seed for this region. Growth continues until there are no new seeds for the current region. Once a region is complete, a new seed is selected from the initial seed set, and new regions are created until no more initial seeds are found.

The homogeneity criterion, as mentioned before, is that the new pixel belongs to the same a high-threshold color class than the initial seed. This avoids color flooding in areas with similar contrast.

6. Applications and Results

The present technique was applied in the 4-legged league of the Robocup competition (www.robocup.org). The basic idea behind this league is to make four autonomous legged mobile robots to play soccer. This league uses a standard robot, Sony's AIBO ERS-7 robot, to guarantee a common hardware platform for each team, prohibiting the addition of additional hardware equipment for this robot. The soccer field is color tagged, providing specific colored and sized objects to allow the robot to identify the object type and location, and then to infer its position on the field.

Rules of this league state that the robot should play autonomously on the soccer field, so every decision must be taken based on an efficient evaluation of the environment based on the information from the built-in camera. The robot provides a maximum camera resolution of 208x160pixels, producing 25 frames per second. This image should be processed by an internal MIPS R7000 CPU at 576 Mhz, which should also deal with the control of all the motors of the robot and the general strategy of the soccer game. The vision algorithm should be very efficient in order to deal with as many images as possible.

An application to acquire samples from environment objects and build the color classes was programmed using the proposed algorithm. A screenshot from this application is shown in Figure 6. This application allows the selection of pixels from images to create clouds of samples for each desired color. The efficiency of the classification approach allows that, interactively, after selecting new samples from the image, the color class is updated, producing the implicit surface on the right area of the application, and classifying pixels on the image in the lower left area of the application. However, in order to produce a more efficient online classification, the application produces a look-up table

that will reduce the online color classification step to a single query in a 3D array for each pixel.

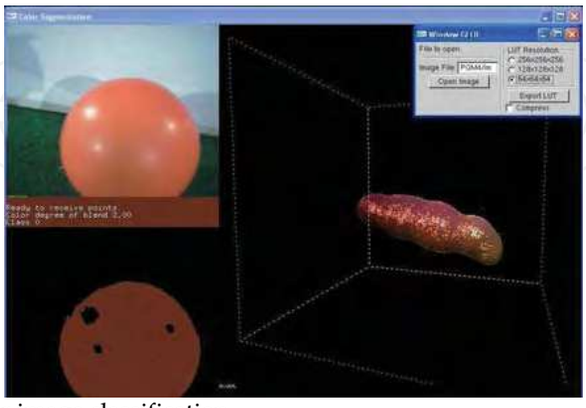


Fig. 6. Tool for color image classification.

The produced primitives and the resulting implicit surfaces fit tightly the samples used in the segmentation process, producing a precise representation of the color class. The blending degree can be also modified interactively.
Figure 7 provides an example on how changes on this parameter affect the result of the color class and of the classification itself. Smaller blending degree produces robust color identification, while larger blending degree produces more precise results. This attribute, together with the threshold for the primitive radius, is useful to resolve overlapping between nearby color classes, and provides better control on the class tolerance.

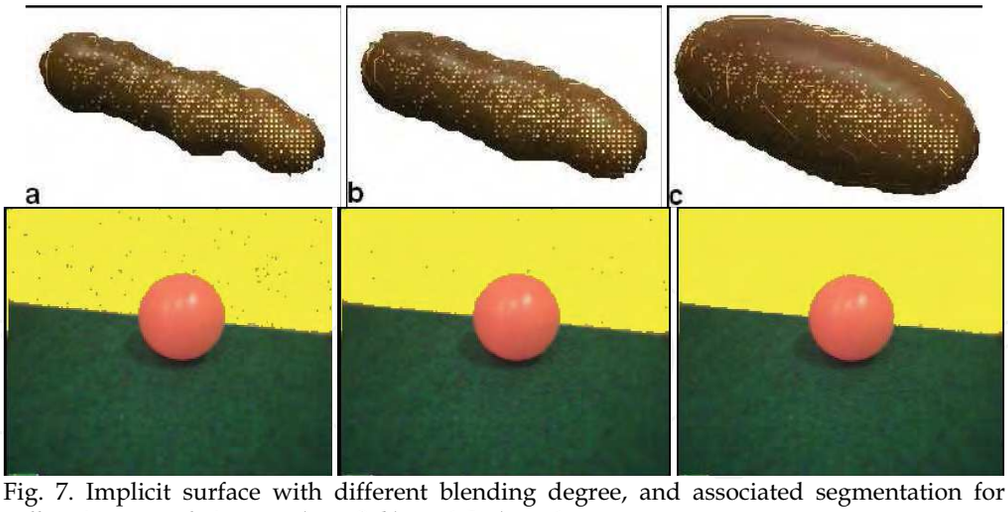


Fig. 7. Implicit surface with different blending degree, and associated segmentation for yellow in a sample image. a) $\rho = 2$, b) $\rho = 1.5$, c) $\rho = 1$.

Color classification was also tested under different lighting conditions comparing the results with the parallelepiped threshold technique. Images processed with the implicit surface approach are better identifying colors, even under extreme illumination changes. This test is shown in Figure 8. In addition, this figure also shows the color subspace produced by the

color class. The parallelepiped threshold approach produces some misclassification problems that are visible in the lower images from Figure 8.
In addition to a higher tolerance to illumination changes, the proposed approach could be extended to dynamically adapt the color subspace, by using a region growth approach as the one presented here in the segmentation algorithm.
The multilevel classification technique improved the efficiency of previous techniques while producing a better quality image segmentation, and higher tolerance to illumination changes. The evaluation of this algorithm was simulated on a Pentium IV at 1.6 Ghz desktop computer, using real information from the robot.
The seed extraction process used a set of color classes with a 0.5 threshold value. As the entire regions for the field lines and the field itself are not required, this step is used to identify lines and borders of the field, and these colors are discarded from the region growth step. The first step of this simulation takes an average of 16 ms.

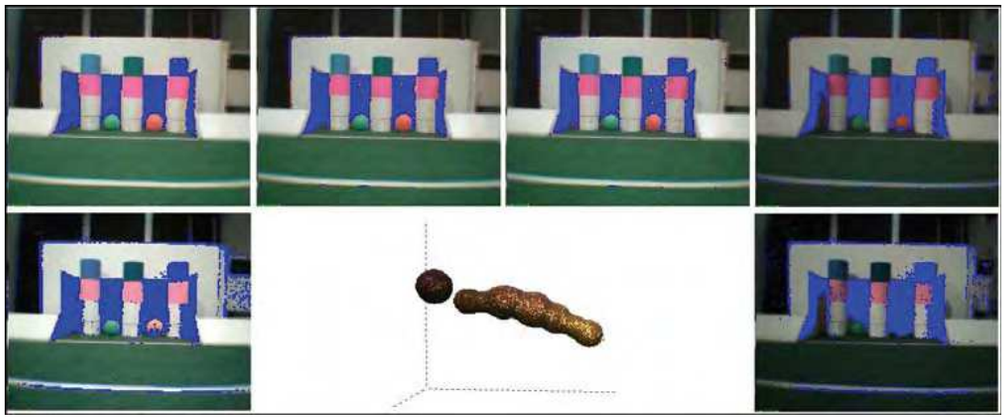


Fig. 8. Tolerance to changing lighting conditions. The yellow color is being classified and replaced by blue. Upper row: Color classification using an implicit surface threshold under different lighting conditions. Lower row, middle: Color subspace used for the images on the upper row. Lower row, left and right: Color classification using parallelepiped thresholds.

Once the seeds are extracted, the region growth stage is executed, using a set of color classes with a 1.0 threshold value. This step takes an average of 24 ms.
Some results of both steps of the segmentation are seen in Figure 9.

7. Conclusions and Future Work

The presented color classification technique shows a good approximation for color subspaces without the need of transforming the color signal of pixels. The produced implicit surface binds tightly the cloud of color samples, reducing possible overlapping problems with other color classes. The use of a look-up table was an efficient method that allows the classification of a single pixel with a single lookup on a 3D array.
In addition, evaluation of this algorithm under varying lighting conditions showed a better color classification than that produced by other color classification methods. The main reason behind this benefit is the tight approximation of the color class to the cloud of

samples; as overlapping problems are reduced, a larger number of samples with different color intensities and tones may be selected to produce the color class.

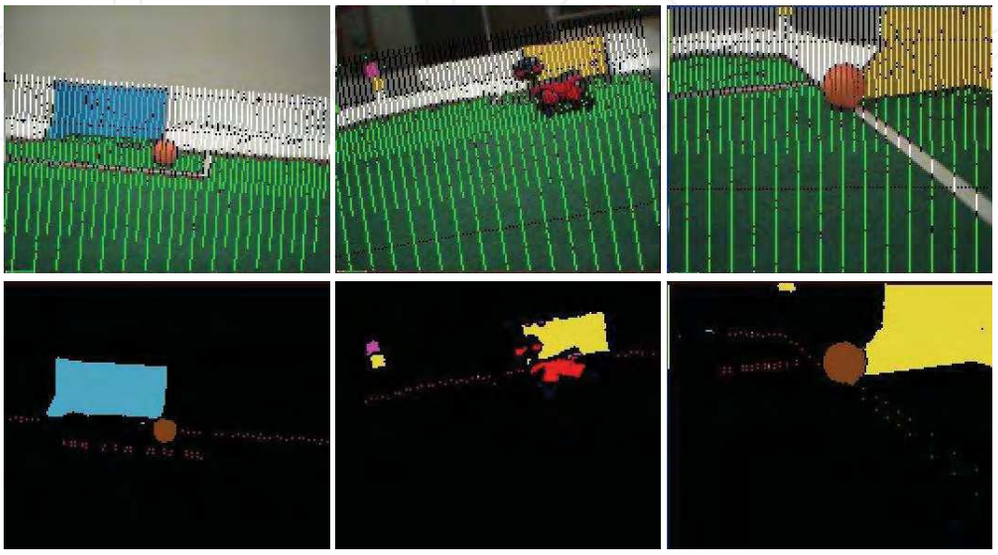


Fig. 9. Top row: Results of scanlines seed extraction stage. Bottom row: Results of region growth stage.

Implicit surface was shown as a well suited threshold surface for color classification, as it is simple to identify whether a color lies inside or outside the color class by just evaluating the value of the implicit function. In addition, once the parameters for the primitives of the implicit surface have been estimated, the tolerance to noise and to changing lighting condition can be easily customized by changing both the blending degree and the primitive threshold for the desired color class, requiring no further execution of the entire classification process.

While the entire classification process has not yet been implemented to be executed by a mobile robot, its offline classification process runs at interactive times. This may allow the later implementation of this technique on a mobile robot to dynamically modify color classes using some similarity criteria.

The presented color image segmentation technique combines the advantages of different techniques reducing, at the same time, some of their drawbacks. The use of scanlines reduces the number of pixels evaluated by the classification and segmentation process, and helps to discard external noise due to objects outside the areas of interest of the image.

Using a high-threshold color class as homogeneity criterion for the region growth algorithm provides with a greater control on the growth process, avoiding flooding problems on low-contrast boundaries.

Besides, the use of region growth completes the information extracted from scanlines. Pixels found for a region are the same that would be obtained by processing the entire image, but scanlines avoid processing pixels with no useful information, improving the efficiency of the segmentation phase.

In addition, scanlines should only find one pixel from an object to identify it, as the SRG step will identify the rest of the pixel. This reduces the required number of scanlines to obtain a good image segmentation.

Many possible improvements can be considered. First, the *k*-means algorithm requires an initial number of primitives. It would be desirable that the number of primitives would also adapt to the nature of the color class. For this purpose, different initialization algorithms may be incorporated to produce better results.

Another possible improvement lies on the nature of the primitives. Spherical clusters are well suited as an initial approach, as Euclidean distance is a simple metric to evaluate distance to a cluster. However, different distance metrics may provide with ellipsoidal or other-shaped clusters, which may produce as a result the use of less primitives and possibly a tighter adaptation of the color class to the samples.

In addition, the online adjustment of the implicit surface threshold may be used by an online dynamic color classification technique that will adapt to varying lighting conditions. The required samples for this segmentation may be obtained from the region growth algorithm. When new pixels are included into a region, the dynamic classification algorithm would add these pixels to the cloud of samples for a given color class. Then, after receiving a set of pixels, the color classification process would be executed and the color class updated, producing a new classification suitable for current illumination conditions. This mechanism would also require an additional step to discard invalid samples and to validate new samples, either by the frequency of a sample, its last appearance on a frame, or its distance to the previous color class. This technique would produce a reliable, non-supervised color segmentation technique that could adapt constantly to different conditions from the environment.

8. References

Alvarez, R.; Millán, E.; Swain-Oropeza, R. & Aceves-López, A. (2004). Color image classification through fitting of implicit surfaces, *9th Ibero-American Conf. on Artificial Intelligence (IBERAMIA*), Lecture Notes in Computer Science, Vol. 3315 (January 2004), pp. 677 – 686, ISSN 0302-9743.

Alvarez, R.; Millán, E. & Swain-Oropeza, R. (2005). Multilevel Seed Region Growth Segmentation. *MICAI 2005: Advances in Artificial Intelligence, 4th Mexican International Conference on Artificial Intelligence*, Lecture Notes in Artificial Intelligence, Alexander F. Gelbukh, Alvaro de Albornoz, Hugo Terashima-Marín, Ed., pp. 359 – 368, Springer, ISBN 3540298967, Berlin / Heidelberg.

Bishop, C. M. (1996) *Neural networks for pattern recognition*, Oxford University Press, ISBN: 0198538642, United States

Bloomenthal, J. & Wyvill, B. (1997) *Introduction to Implicit Surfaces*, Morgan Kaufmann Publishers Inc, ISBN: 155860233X.

Bruce, J.; Balch, T. & Veloso, M. M. (2000). Fast and inexpensive color image segmentation for interactive robots, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, Vol. 3, (October 2000), pp. 2061–2066, ISSN 0932-8092.

Castleman, K. (1996). *Digital image processing*, Prentice Hall, ISBN: 0132114674

Forsyth, D. & Ponce, J. (2002). *Computer Vision: A Modern Approach*, Prentice Hall, ISBN: 0130851981, United States

Jüngel, M. (2004). Using layered color precision for a self-calibrating vision system. *RoboCup 2004: Robot Soccer World Cup VIII*, Lecture Notes in Artificial Intelligence, Daniele Nardi, Martin Riedmiller, Claude Sammut, José Santos-Victor, Ed., pp. 209-220, Springer, ISBN 3540250468, Berlin / Heidelberg.

Langetepe, E. & Zachmann, G. (2006). *Geometric data structures for computer graphics*, A. K. Peters, Ltd., ISBN: 1568812353, United States.

Lim, C. T.; Turkiyyah, G. M.; Ganter, M. A. & Storti, D. W. (1995). Implicit reconstruction of solids from cloud point sets, *Proceedings of the third ACM symposium on Solid modeling and applications*, pp. 393–402, ISBN: 0-89791-672-7, Salt Lake City, Utah, United States, 1995, ACM Press.

Ricci, A. (1973) A constructive geometry for computer graphics, *The Computer Journal,* Vol. 16, No. 2, (May 1973), pp. 157–160, ISSN: 0010-4620.

Von Hundelshausen, F. & Rojas, R. (2003). Tracking regions. *RoboCup 2003: Robot Soccer World Cup VII*, Lecture Notes in Computer Science, Daniel Polani, Brett Browning, Andrea Bonarini, Kazuo Yoshida, Ed., pp 250-261, Springer, ISBN: 3540224432, Berlin / Heidelberg.

Wasik, Z. & Saffiotti, A. (2002). Robust color segmentation for the Robocup domain. *International Conference on Pattern Recognition*, pp. 651-654, ISBN: 076951695X, Quebec City, Canada.

Watt, A., & Policarpo, F. (1998). *The computer image*, Addison Wesley, ISBN: 0201422980

Mobile Robots: Perception & Navigation

Edited by Sascha Kolski

ISBN 3-86611-283-1

Hard cover, 704 pages

Publisher Pro Literatur Verlag, Germany / ARS, Austria

Published online 01, February, 2007

Published in print edition February, 2007

Today robots navigate autonomously in office environments as well as outdoors. They show their ability to beside mechanical and electronic barriers in building mobile platforms, perceiving the environment and deciding on how to act in a given situation are crucial problems. In this book we focused on these two areas of mobile robotics, Perception and Navigation. This book gives a wide overview over different navigation techniques describing both navigation techniques dealing with local and control aspects of navigation as well es those handling global navigation aspects of a single robot and even for a group of robots.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Raziel Alvarez, Erik Millan, Alejandro Aceves-Lopez and Ricardo Swain-Oropeza (2007). Accurate Color Classification and Segmentation for Mobile Robots, Mobile Robots: Perception & Navigation, Sascha Kolski (Ed.), ISBN: 3-86611-283-1, InTech, Available from: http://www.intechopen.com/books/mobile_robots_perception_navigation/accurate_color_classification_and_segmentation_for_mobile_robots

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen