Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Food Engineering 66 (2005) 353–360

JOURNAL OF FOOD ENGINEERING

www.elsevier.com/locate/jfoodeng

Segmentation of colour food images using a robust algorithm

Domingo Mery [a,*], Franco Pedreschi [b]

[a] *Departamento de Ciencia de la Computación, Pontificia Universidad Católica de Chile, Av. Vicuña Mackenna 4860(183), Santiago de Chile, Chile*

[b] *Departamento de Ciencia y Tecnología de Alimentos, Facultad Tecnologíca, Universidad de Santiago de Chile (USAH), Av. Ecuador 3769, Santiago de Chile, Chile*

Received 14 October 2003; accepted 4 April 2004

Abstract

In this paper, a robust algorithm to segmenting food image from a background is presented using colour images. The proposed method has three steps: (i) computation of a high contrast grey value image from an optimal linear combination of the RGB colour components; (ii) estimation of a global threshold using a statistical approach; and (iii) morphological operation in order to fill the possible holes presented in the segmented binary image. Although the suggested threshold separates the food image from the background very well, the user can modify it in order to achieve better results. The algorithm was implemented in Matlab and tested on 45 images taken in very different conditions. The segmentation performance was assessed by computing the area A_z under the receiver operation characteristic (ROC) curve. The achieved performance was $A_z = 0.9982$.

Keywords: Image analysis; Image processing; Segmentation; Colour images

1. Introduction

Computer vision is a novel technology for acquiring and analyzing an image of a real scene by computers and other devices in order to obtain information or to control processes. The core technique in computer vision is always related to image analysis/processing, which can lead to segmentation, quantification and classification of images and objects of interest within images. Computer vision has proven successful for online measurement of several food products with applications ranging from routine inspection to the complex vision guided robotic control (Gunasekaram, 1996). Brosnan and Sum (2004) present a review indicating the application of computer vision in certain foods such as bakery products, meat and fish, vegetables, fruits, grains, prepared consumer foods and in food container inspection. As shown in Fig. 1, the steps involved in image analysis are (Gonzalez & Wintz, 1991):

- Image formation, in which an image of the food under test is taken and stored in the computer.
- Image pre-processing, where the quality of the digital image is improved in order to enhance the details.
- Image segmentation, in which the food image is found and isolated from the background of the scene.
- Measurement, where some significant features of the food image are quantified.
- Interpretation, where the extracted features are interpreted using some knowledge about the analysed object.

The segmentation process partitions the digital image into disjoint (non-overlapping) regions (Castleman, 1996). Segmentation is an essential step in computer vision and automatic pattern recognition processes based on image analysis of foods as subsequent extracted data are highly dependent on the accuracy of this operation. In general, the automated segmentation is one of the most difficult tasks in the image analysis (Gonzalez & Wintz, 1991), because a false segmentation will cause degradation of the measurement process and therefore the interpretation may fail. Food image segmentation is still an unsolved problem because of its

[*] Corresponding author. Fax: +56-2-354-4444.
E-mail address: dmery@ing.puc.cl (D. Mery).
URL: www.ing.puc.cl/~dmery.

0260-8774/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jfoodeng.2004.04.001

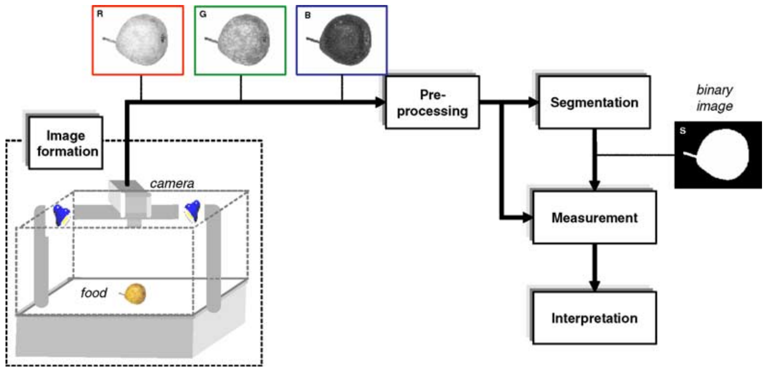


Fig. 1. Schematic representation for food image analysis.

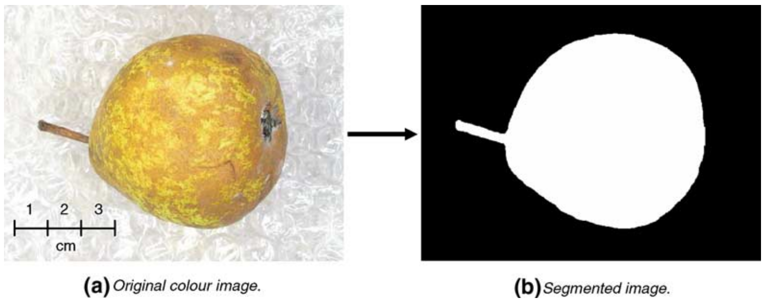


Fig. 2. Segmentation process for a pear image.

complex and underconstrained attributes. Generally, in the food image segmentation, there are only two regions: the foreground (food image itself) and the background. Thus, the result is a *binary image*, where the pixels may take only one of the values '1' or '0'. These values correspond to the foreground or background respectively. Fig. 2 shows an example, where the input is a colour image of a pear (with background) and the output is an image with only two colours: white for the pear and black for the background. Recently, Sun and Du (2004) developed an algorithm for segmenting complex images of many types of foods including pizza, apple, pork and potato. In this approach, the food image itself is segmented into different sub-regions, e.g. a pizza is partitioned into cheese, ham, tomato sauce, etc.

A robust algorithm was developed for segmenting food images from their backgrounds using colour images. The proposed method has three steps: (i) computation of a high contrast grey value image from an optimal linear combination of the RGB colour components; (ii) estimation of a global threshold using a statistical approach; and (iii) a morphological operation in order to fill the possible holes presented in the segmented binary image.

2. Materials and methods

2.1. Materials

The following foods were analysed: mango, almond, Nestlé cereal cluster, corn flakes, cookies, mandarin, wheat, potato chip, raisin, pear, nectarine, plum, pepino, red apple, green apple, pear, avocado, banana, orange, tomato, passion fruit (granadilla) and peanut.

2.2. Image acquisition

Images were captured using an image acquisition system for a digital colour camera similar to that developed by Papadakis, Abdul-Malek, Kamdem, and Yam (2000), namely:

(a) Samples were illuminated by using four parallel fluorescent lamps (length of 60 cm) with a colour temperature of 6500 K (Philips, Natural Daylight, 18W) and a colour rendering index (*Ra*) near to 95%. The four lamps were situated 3.5 cm above the sample and at angle of 45° of the food sample plane. This illumination system gave a uniform light intensity over the food plane.

(b) A Digital Colour Camera (DCC) Canon, Power Shot G3—Japan—was located vertically at a distance of 22.5 cm from the sample. The angle between the camera lens axis and the lighting sources was around 45°. Sample illuminators and the DCC were inside a wood box whose internal walls were painted black to avoid the light and reflection from the room. The white balance of the camera was set using a standardized grey colour chart of Kodak.
(c) Images were captured with the mentioned DCC at its maximum resolution (2272×1704 pixels) and connected to the USB port of a PC. Canon Remote Capture Software (version 2.6.0.15) was used for acquiring the images directly in the computer in TIFF format without compression.

2.3. Segmentation

A robust algorithm for segmenting food colour images from the background was developed using Matlab code. This algorithm has three steps: (i) computation of a high contrast grey value image from an optimal linear combination of the RGB colour components; (ii) estimation of a global threshold using a statistical approach; and (iii) a morphological operation in order to fill the possible holes presented in the segmented binary image.

3. Results and discussion

3.1. Computation of a high contrast monochrome image

After the colour image acquisition, an RGB image is obtained. The RGB image is stored in three matrices, called **R**, **G** and **B** respectively, which contain the intensity values of the red, green and blue components of the image. The corresponding intensity values for a (x,y) pixel are denoted in this paper as $R(x,y)$, $G(x,y)$ and $B(x,y)$, for $x=1,\dots,M$ and $y=1,\dots,N$, where M and N are respectively the size of the files and columns of the digital image.

There are several colour space transformations (see for example Hunt, 1991). They attempt to obtain a better representation of the colour. Many of these transformations have the linear form:

$$I(x,y) = k_rR(x,y) + k_gG(x,y) + k_bB(x,y) \tag{1}$$

where (k_r, k_g, k_b) are the weights that ponder the RGB components, and $I(x,y)$ is the transformed grey value of the (x,y) pixel. For instance, the chrominance value Q in the YIQ space is computed with $k_r = 0.212$, $k_g = -0.523$ and $k_b = 0.311$ (Gonzalez & Wintz, 1991), and a grey value image is obtained with $k_r = 0.2989$, $k_g = 0.5870$ and $k_b = 0.1140$, where the hue and saturation information is eliminated while retaining the luminance (MathWorks, 2003).

In our approach, we use a normalised monochrome image computed as:

$$J(x,y) = \frac{I(x,y) - I_{\min}}{I_{\max} - I_{\min}} \tag{2}$$

where $I_{\min}$ and $I_{\max}$ are the minimum and maximum values of **I**. Thus, the normalisation ensures that the grey values of **J** are between 0 and 1.

An appropriate representation of the image should have a high contrast, i.e., a low homogeneity. Since an image with low homogeneity will have a high variance, we seek the *best* combination (k_r, k_g, k_b) in (1) that maximises the variance of **J**:

$$\sigma_J^2(k_r, k_g, k_b) \rightarrow \max \tag{3}$$

Since only the ratios $k_r{:}k_g{:}k_b$ are significant for the normalisation, we can set $k_b = 1$ without loss of generality. The optimal high contrast image can be found using an exhaustive method that evaluates (3) for several combinations (k_r, k_g) (with $k_b = 1$) and takes the combination that maximises σ_J^2. In the exhaustive search we can use the following values for $k_r, k_g = k_0, k_0 + \Delta k, \dots, k_1$, with $k_0 = -1$, $k_1 = 1$ and $\Delta k = 0.1$. However, a better solution can be obtained using a numerical gradient method. In this case, we start with an initial guess $(k_r, k_g)_0$ and update this value using the iteration

$$(k_r, k_g)_{i+1} = (k_r, k_g)_i + (\Delta_r, \Delta_g)_i \tag{4}$$

where $(\Delta_r, \Delta_g)_i$ is computed using the gradient of (3) evaluated at $(k_r, k_g)_i$. The iteration is interrupted, once no considerable modification of $(k_r, k_g)_{i+1}$ is achieved upon adding $(\Delta_r, \Delta_g)_i$. This multidimensional unconstrained non-linear maximisation is included in the toolbox for optimisation of Matlab (MathWorks, 2000) (see for example function `fminsearch` that can be used to minimise $-\sigma_J^2$). Section 3.4 shows a Matlab program, called `rgb2hcm`, that computes the high contrast monochrome image from an RGB image. Several examples are illustrated in Fig. 3, where the colour components (**R**, **G**, **B**) of Fig. 2a are transformed into a new optimal high contrast monochrome image **J**. In addition, we show a typical greyscale image **I**′ converted from the RGB image using $k_r = 0.2989$, $k_g = 0.5870$ and $k_b = 0.1140$ (in some cases **−R**, **−G**, **−B**, **−I**′ are shown in order to preserve the dark background). In these examples, the Matlab command `imshow(X, [ ])` was employed to display image `X` using the whole scale, i.e., the minimum value in `X` is displayed as black, and the maximum value as white. The greyscale image **I**′ was computed using the Matlab command `rgb2gray`. In all these examples, we observe the ability of our transformation to obtain a high contrast monochrome image. Since the high variability of

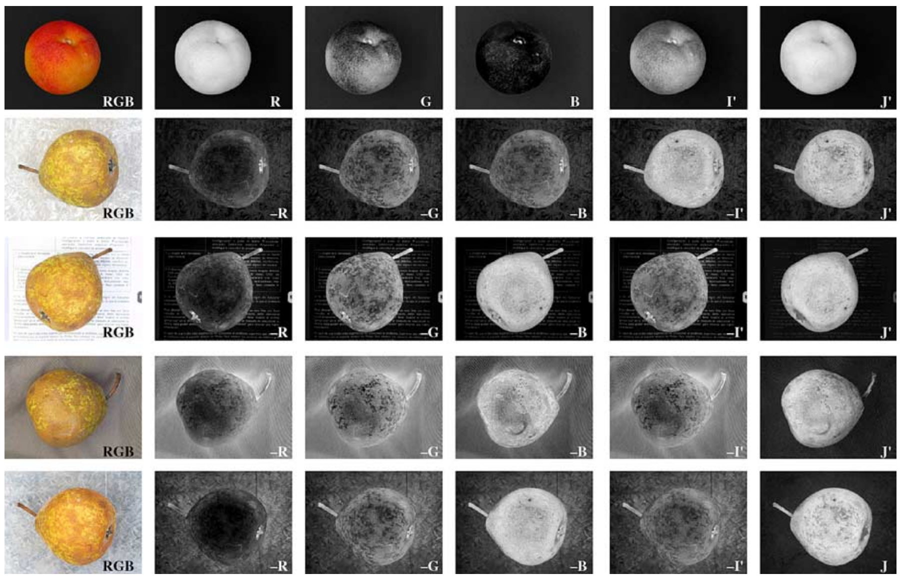


Fig. 3. Comparison between high contrast image **J** and **R**, **G**, **B** colour components and greyscale image **I**′ in RGB images with different backgrounds.

the background is attenuated, the foreground can be clearly identified.

3.2. Global threshold estimation

The obtained image **J** has a bimodal histogram as shown in Fig. 4, where the left distribution corresponds to the background and the right to the food image. In this high contrast image, a first separation between foreground and background can be performed estimating a global threshold t. Thus, we define a binary image

$$K(x,y) = \begin{cases} 1 & \text{if } J(x,y) > t \\ 0 & \text{else} \end{cases} \tag{5}$$

where '1' means foreground and '0' background, that define two classes of pixels in the image. The problem is to determine a best threshold t that separates the two modes of the histogram from each other. A good separation of the classes is obtained by ensuring (i) a small variation of the grey values in each class, and (ii) a large variation of the grey values in the image (Haralick & Shapiro, 1992). The first criterion is obtained by minimising a weighted sum of the within-class variances (called *intraclass* variance $\sigma^2_{\mathrm{W}}(t)$):

$$\sigma^2_{\mathrm{W}}(t) = p_{\mathrm{b}}(t)\sigma^2_{\mathrm{b}}(t) + p_{\mathrm{f}}(t)\sigma^2_{\mathrm{f}}(t) \tag{6}$$

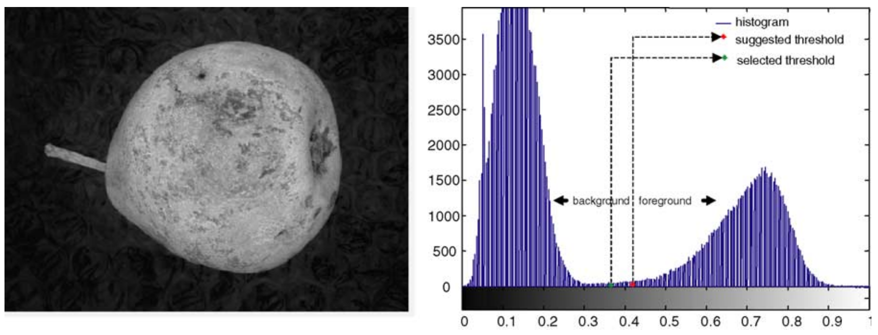
where the indices 'b' and 'f' denote respectively background and foreground classes, and p and σ^2 are respectively the probability and the variance for the indicated class. These values can be computed from the histogram.

The second criterion is obtained by maximising the between-class variance (called *interclass* variance $\sigma^2_{\mathrm{B}}(t)$):

$$\sigma^2_{\mathrm{B}}(t) = p_{\mathrm{b}}(\mu_{\mathrm{b}}(t) - \mu)^2 + p_{\mathrm{f}}(\mu_{\mathrm{f}}(t) - \mu)^2 \tag{7}$$

where μ_{b}, μ_{f} and μ indicate the mean value of the background, foreground and the whole image respectively.

The best threshold t can be estimated by a sequential search through all possible values of t that minimises



(a) *High contrast monochrome image.* **(b)** *Histogram.*

Fig. 4. High contrast image and corresponding histogram.

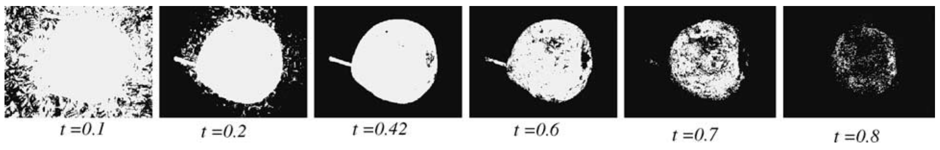


Fig. 5. Separation between foreground and background of Fig. 4a for different thresholds. The value $t = 0.42$ was suggested by the outlined algorithm (see histogram in Fig. 4b).

$\sigma^2_W(t)$ (or maximises $\sigma^2_B(t)$). Both criteria however lead to the same result because the sum $\sigma^2_W + \sigma^2_B$ is a constant and corresponds to the variance of the whole image (Haralick & Shapiro, 1992). Matlab computes the global image threshold by minimising the intraclass variance $\sigma^2_W(t)$. The threshold can be obtained with the function `graythresh` (MathWorks, 2003) (see Section 3.4 for details). An example is shown in Fig. 5.

3.3. Morphological operation

We observe in Fig. 5 that the segmentation suffers from inaccuracy because there are many dark (bright) regions belonging to the foreground (background) that are below (above) the chosen threshold and therefore misclassified. For this reason, an addition morphological processing must be achieved.

The morphological operation is performed in three steps as shown in Fig. 6: (i) remove small objects, (ii) close the binary image and (iii) fill the holes.

In the first step, we remove from binary image **K** obtained in previous section all connected regions that have fewer than n pixels (see image **A** in Fig. 6).This operation is necessary to eliminate those isolated pixels of the background that have a grey value greater than the selected threshold. This situation may occur when there are shiny surfaces in the background that produce specular reflections. Empirically we set $n = NM/100$, where $N \times M$ is the number of pixels of the image.

The second step *closes* the image, i.e., the image is *dilated* and then *eroded*. The dilation is the process that incorporates into the foreground the background pixels that touch it. On the other hand, erosion is the process that eliminates all the boundary pixels of the foreground. The closing process (dilation followed by erosion) fills small holes and thins holes in the foreground, connecting nearby regions, and smoothing the boundaries of the foreground without changing the area significantly (Castleman, 1996) (see image **C** in Fig. 6). This operation is very useful in foods that have spots in the boundary (see for example the pear in Fig. 6).

Finally, the last operation fills the holes in the closed image (see image **R** in Fig. 6). We use this operation to incorporate into the foreground all pixels '0' that are inside of the region (see for example the mandarin and the plum in Fig. 6). The whole algorithm is summarised in Fig. 7.

3.4. Matlab programs

In this section we describe briefly the Matlab programs implemented. The main program is called `SegFood` (see Fig. 8). We can use this program with the following instructions:

```
I = imread(file_name);
[R,E,J] = SegFood(I,p);
```

In this example, the image saved in *file_name* is stored in matrix `I`. The program `SegFood` segments image `I` in `R`. An example is shown in Fig. 2. In addition, the edges of this binary image are given in `E`, and the high contrast monochrome image is stored in `J`. The user can give a value for `p`. If no value is given, the program assumes `p` $= -0.05$.

`SegFood` calls three functions: `rgb2hcm`, `graythresh` and `MorphoFood`. The first one, as shown in Fig. 8, computes the high contrast monochrome

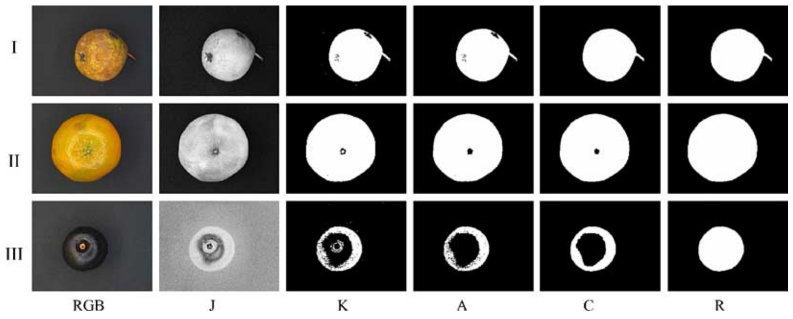


Fig. 6. Morphological operations: **RGB**: colour image, **J**: high contrast monochrome image, **K**: binary image after thresholding, **A**: after removing of small objects, **C**: after closing process, and **R**: after filling holes. (I) pear, (II) mandarin and (III) plum.

```
0. Read image
$\mathbf{X} = \text{read}(\textit{colour image})$
$\mathbf{R} = \text{red\_component}(\mathbf{X})$
$\mathbf{G} = \text{green\_component}(\mathbf{X})$
$\mathbf{B} = \text{blue\_component}(\mathbf{X})$

1. High contrast monochrome image
$(\hat{k}_r, \hat{k}_g) = \arg\max_{k_r,k_g}(\sigma_J^2)$
    $where$
    $\rightarrow \sigma_J^2 = \text{variance}((\mathbf{I}(k_r,k_g) - I_{\min})/(I_{\max} - I_{\min}))$
    $\rightarrow \mathbf{I}(k_r,k_g) = k_r\mathbf{R} + k_g\mathbf{G} + \mathbf{B}$
$\hat{\mathbf{I}} = \hat{k}_r\mathbf{R} + \hat{k}_g\mathbf{G} + \mathbf{B}$
$\mathbf{J} = (\hat{\mathbf{I}} - \hat{I}_{\min})/(\hat{I}_{\max} - \hat{I}_{\min})$

2. Global threshold estimation
$\hat{t} = \arg\min_t(\sigma_W^2)$
    $where$
    $\rightarrow \sigma_W^2 = p_b(t)\sigma_b^2(t) + p_f(t)\sigma_f^2(t)$
    $\rightarrow \sigma_f^2(t) = \text{variance}(\mathbf{J} > t)$
    $\rightarrow \sigma_b^2(t) = \text{variance}(\mathbf{J} \leq t)$
    $\rightarrow p_f(t) = \text{probability}(\mathbf{J} > t)$
    $\rightarrow p_b(t) = \text{probability}(\mathbf{J} \leq t)$
$\mathbf{K} = (\mathbf{J} > \hat{t})$

3. Morphological operation
$\mathbf{A} = \text{remove\_small\_objects}(\mathbf{K})$
$\mathbf{C} = \text{close}(\mathbf{A})$
$\mathbf{R} = \text{fill\_holles}(\mathbf{C})$
```

Fig. 7. Algorithm.

```
% File SegFood.m
function [R,E,J] = SegFood(I,p)
  J = rgb2hcm(double(I)/256);
  t = graythresh(J);
  if ( exist('p'))
    p = -0.05;
  end
  [R,E] = MorphoFood(J,t+p);

% File rgb2hcm.m
function J = rgb2hcm(RGB);
  if (size(RGB,3)==1)
    I = RGB;
  else
    RGB64 = imresize(RGB,[64 64]);
    k = fminsearch('StdMonochrome',[1 1],[ ],RGB64);
    I = k(1)*RGB(:,:,1) + k(2)*RGB(:,:,2) + RGB(:,:,3);
  end
  J = I - min(I(:));
  J = J/max(J(:));
  n = fix(size(J,1)/4);
  if (mean2(J(1:n,1:n)) > 0.3)
    J = 1 - J;
  end

% File StdMonochrome.m
function s = StdMonochrome(k,RGB)
  I = k(1)*RGB(:,:,1) + k(2)*RGB(:,:,2) + RGB(:,:,3);
  s = -std2(I)/(max(I(:))-min(I(:)));

% File MorphoFood.m
function [R,E] = MorphoFood(J,t);
  A = bwareaopen(J>t,fix(length(J(:))/100));
  C = imclose(A,strel('disk',7));
  R = bwfill(C,'holes',8);
  E = bwperim(R,4);
```

Fig. 8. MATLAB code (see algorithm in Fig. 7).

image by minimising the variance of a normalised image (see explanation in Section 3.1 and examples in Fig. 3). The variance is computed by function `StdMonochrome` shown in Fig. 8. The second function belongs to Matlab Image Processing Toolbox, and calculates the threshold of a monochrome image according to Otsu's methods (see explanation in Section 3.2 and examples in Fig. 5). Finally, the third function computes the morphological operations as explained in Section 3.3. Examples are given in Fig. 6. The code is presented in Fig. 8.

3.5. Performance analysis

The performance of our method is very sensitive to the variation of the threshold t. For this reason, we define a new threshold as $t_n = t + p$ (see Fig. 4b where $p = -0.05$, in this case t and t_n are suggested and selected threshold respectively). Parameter p can be given by the user (if no parameter p is given, the software assumes the default value $p = -0.05$). If $p > 0$ (or $p < 0$) the software will increase (or decrease) the area belonging to the background.

In order to assess the segmentation performance, the receiver operation characteristic (ROC) (Egan, 1975) curve is analysed (see Fig. 9), which is a plot of the 'sensitivity' (S_n) against the '1-specificity' (1-S_p) defined as:

$$S_n = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad 1\text{-}S_p = \frac{\text{FP}}{\text{TN} + \text{FP}}, \tag{8}$$

where

- TP is the number of true positives (pixels of the foreground correctly classified);
- TN is the number of true negatives (pixels of the background correctly classified);
- FP is the number of false positives or false alarms (pixels of the background classified as foreground); and
- FN is the number of false negatives (pixels of the foreground classified as background).

Ideally, $S_n = 1$ and 1-$S_p = 0$, i.e., all pixels belonging to the food are classified as foreground without flagging false alarms. The ROC curve permits assessment of the detection performance at various operating points of p, e.g. $p = -0.3, \ldots, 0.3$. The area under the ROC curve (A_z) is normally used as a measure of performance because it indicates how reliably the detection can be

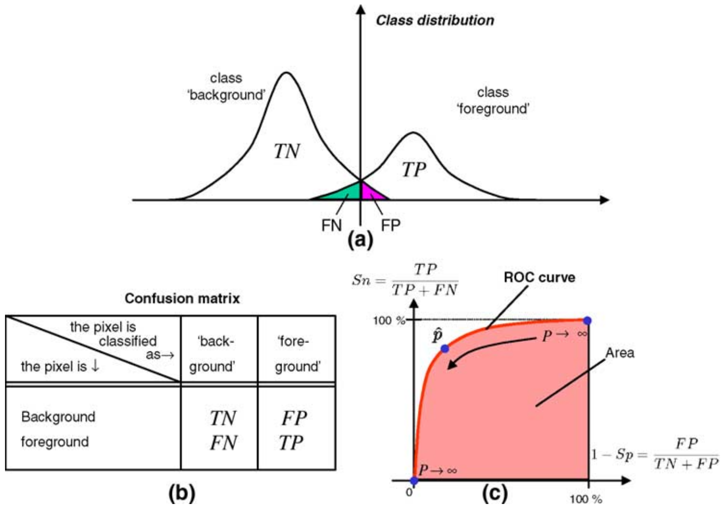


Fig. 9. Analysis ROC: (a) class distribution, (b) confusion matrix, (c) ROC curve.

Table 1
Performance analysis on 45 images

Description	Samples	A_z	$\hat{p}$	$\widehat{S}_n$	1-$\widehat{S}_p$	N	M	d
Almond	(2)	0.9983	−0.2500	0.9960	0.0003	426	568	6.2 cm
Avocado	(1)	0.9853	−0.1250	0.9599	0.0000	426	568	10.1 cm
Banana	(1)	0.9954	−0.3000	0.9909	0.0007	426	568	16.3 cm
Chip	(7)	0.9989	−0.2607	0.9976	0.0008	390	520	6.2 cm
Cluster	(1)	0.9986	−0.3000	0.9974	0.0005	426	568	6.2 cm
Cookie	(2)	0.9966	−0.2875	0.9959	0.0045	426	568	6.2 cm
Corn flake	(2)	0.9998	−0.1250	0.9994	0.0011	426	568	6.2 cm
Green apple	(1)	0.9981	−0.2250	0.9918	0.0002	426	568	10.1 cm
Mandarin	(3)	0.9998	−0.1583	0.9990	0.0004	426	568	6.2 cm
mango	(1)	0.9989	−0.3000	0.9978	0.0006	426	568	10.1 cm
Puffed wheat	(2)	0.9999	−0.2750	0.9997	0.0005	426	568	6.2 cm
Nectarine	(1)	1.0000	−0.2500	0.9995	0.0011	426	568	6.2 cm
Orange	(1)	0.9979	−0.3000	0.9958	0.0004	426	568	10.1 cm
Passion fruit	(1)	0.9951	−0.3000	0.9903	0.0006	426	568	10.1 cm
Pear	(12)	0.9990	−0.1729	0.9947	0.0020	426	568	10.1 cm
Peanut	(1)	0.9994	−0.1000	0.9936	0.0002	480	640	2.0 mm
Pepino	(1)	0.9998	−0.2750	0.9988	0.0015	426	568	10.1 cm
Plum	(2)	0.9988	−0.0500	0.9967	0.0039	426	568	10.1 cm
Raisin	(1)	0.9999	−0.2500	0.9985	0.0008	426	568	6.2 cm
Red apple	(1)	1.0000	−0.1750	0.9989	0.0010	426	568	10.1 cm
Tomato	(1)	0.9992	−0.3000	0.9985	0.0005	426	568	10.1 cm
Mean	–	0.9985	−0.2122	0.9956	0.0013	–	–	–
Max	–	1.0000	0.1000	0.9999	0.0087	–	–	–
Min	–	0.9853	−0.3000	0.9599	0.0000	–	–	–

A_z: area under the ROC curve, $\hat{p}$: threshold deviation, $\widehat{S}_n$: sensitivity, 1-$\widehat{S}_p$: 1-specificity, $N \times M$: image size in pixels, d: image width.

performed. A value of $A_z = 1$ gives perfect classification, whereas $A_z = 0.5$ corresponds to random guessing. The best value for p, denoted by $\hat{p}$, is chosen as the point on the ROC curve nearest to the ideal point (top left corner, i.e., $S_n = 1$ and 1-$S_p = 0$). The coordinates of the nearest point are denoted by $\widehat{S}_n$ and 1-$\widehat{S}_p$.

In our experiments, 45 colour images of foods were analysed. For each image, an *ideal detection* was achieved using visual interpretation. Our methodology was to create an ideal binary image ('1' is foreground and '0' is background) according to the visual information with the software Microsoft Paint using the biggest scale (zoom = 800%). The results obtained with our algorithm were then compared with the ideal binary image. Thus, the values for TP, TN, FP and FN were tabulated. The results obtained in each food image are summarised in Table 1, in which the number of food samples, the areas A_z and the optimal values $\hat{p}$, $\widehat{S}_n$ and 1-$\widehat{S}_p$ are given. In order to reduce the table, an average is presented when more than one sample was evaluated.

We observe that the mean sensitivity of all images was 0.9956, i.e., on average 99.56% of all pixels of the foods were correctly detected. The mean quote of false alarms (pixels of the background detected as foreground) was 0.13%. In addition, Table 1 shows the dimensions in pixels of each image and the corresponding image width captured by the camera.

Analysing all images together with the same p for each image, we obtain $A_z = 0.9982$. The best performance is achieved at $\hat{p} = -0.05$. In this case, $\widehat{S}_n = 0.9831$ and $1\text{-}\widehat{S}_p = 0.0085$. For this reason, we chose the default value of p as -0.05.

4. Conclusions

In this paper, a robust approach to segmenting food images is proposed. The approach has three steps: (i) computation of a high contrast grey value image from an optimal linear combination of the RGB colour components; (ii) estimation of a global threshold using a statistical approach; and (iii) a morphological operation in order to fill the possible holes presented in the segmented binary image. After testing the implemented algorithm in Matlab on 45 images, the assessed segmentation performance computed from the area under the Receiver Operation Characteristic (ROC) curve was $A_z = 0.9982$.

Acknowledgements

Authors acknowledge financial support from FONDECYT—Chile Project No. 1030411.

References

Brosnan, T., & Sun, D.-W. (2004). Improving quality inspection of food products by computer vision—a review. *Journal of Food Engineering, 61*(1), 3–16.

Castleman, K. (1996). *Digital image processing*. Englewood Cliffs, New Jersey: Prentice-Hall.

Egan, J. (1975). *Signal detection theory and ROC analysis*. New York: Academic Press.

Gonzalez, R., & Wintz, O. (1991). *Digital image processing* (3rd ed.). Massachusetts: Addison-Wesley Publishing Co.

Gunasekaram, S. (1996). Computer vision technology for food quality assurance. *Trends in Food Science and Technology, 7*(8), 245–246.

Haralick, R., & Shapiro, L. (1992). *Computer and robot vision*. New York: Addison-Wesley Publishing Co.

Hunt, R. (1991). *Measuring colour* (2nd ed.). New York: Ellis Horwood.

MathWorks (2000). it Optimization toolbox for use with MATLAB: User's Guide. The MathWorks Inc., September.

MathWorks (2003). *Image processing toolbox for use with MATLAB: User's Guide*. The MathWorks Inc., January.

Papadakis, S., Abdul-Malek, S., Kamdem, R., & Yam, K. (2000). A versatile and inexpensive technique for measuring color of foods. *Food Technology, 54*(12), 48–51.

Sun, D.-W., & Du, C.-J. (2004). Segmentation of complex food images by stick growing and merging algorithm. *Journal of Food Engineering, 61*(1), 17–266.