Image Classification

Vanden Berghen Frank.
fvandenb@iridia.ulb.ac.be

9-7-2003

1 Introduction

To classify an object in a image, we must first extract some features out of the image. We use these features inside a classification tool like C4.5 (decision tree) to obtain the final class. I have made a small package to work with gray-level images (gray color coded on 8 bits). This package has been highly speed-optimized. For example, it uses some "tricks" of demo-coders to allow fast resize of images without any floating point operations nor multiplications. Here is a small description of the capabilities of the package. It's currently used on-line in flat glass defect detection and classification (a factory which produces glass for your windows for example).

2 Noise cancellation

two parameters used: s1,s2.

To be able to filter correctly the image, we must know the value of the background gray level= BGL. This is an iterative algorithm. At each pass, the value of the BGL is refined. The BGL is calculated using the following algorithm (s1 is a parameter to give):

$$\begin{array}{rcll} BGL(x,y) & = & s1 & *(\text{average gray level on line y} \quad)+ \\ & & (1-s1) & *(\text{average gray level on column x} \quad) \end{array} \tag{1}$$

When we calculate the two above average we only use the points that belongs to the object we want to isolate. A point C does NOT belong to the object if $(BGL - s2) < C < (BGL + s2)$. The parameter s2 must be given. The average are thus calculated on set of points changing at each passes.

One last pass is done to re-center all the gray level to 128. For each point of the image, we execute:

$$\text{(new gray level)} = \text{(old gray level)} - BGL + 128 \tag{2}$$

3 Cropping

three parameters used: s3,s4,s5.

Let's consider an image with 3 white peaks. Two of these peaks belongs to the object to classify, the last one (far away from the 2 others) is noise. We want to crop the image so that only the object to classify

remains.

The algorithms begins by searching the point P_1 which has the Gray Level the further away from 128 (128 is now the color of the background: see previous step: noise cancellation). This point P_1 will usually belongs to one of the two peaks evoked before. From this point we will make a "special flood fill" (see end of the section to have a more precise description of the flood fill). The program draws a rectangular box around the zone which has been colored by the "flood fill". Everything outside this box is cropped.
Ideally, the "flood fill" should color the two right peaks. Unfortunately, there is very often, between these 2 peaks, a zone of the image at 128. This zone prevents the "flood fill" to expand up to the second peaks. To overcome this difficulty, the "flood fill" is executed on a reduced copy of the original image (horizontal scaling:s4, vertical scaling: s5). Based on this colored copy, we crop the original image.
FLOOD FILL: coloring algorithm.
The algorithm start from a given pixel. Then, It colors all the adjacent pixel. The coloring stops when we find a gray level C such that $128 - a < C <= 128 + a$ with $a = abs(b - 128) * s3$ and b=gray level of point P_1.

REMARK: This algorithms works with white or black objects. No problem.

4 Cropping 2

one parameter used: s6.

If the image to classify is still to big, we must crop a little bit more. We can create a new "cuted" image which contains $s6\%$ ($s6$ should be 90 or 95) of the luminosity of the original image. The luminosity of an image is defined by the sum of the gray level of all the points in the image.
The "cutting" algorithm is the following:

1. find the center of gravity of the image (the weight of each point C is *weight*=$abs(c - 128)$).
2. put a box of dimension (1,1) around the center of gravity.
3. extends this box in the direction which increases the most the weight of the box.
4. go to point 3 while the luminosity of the box is lower than $s6\%$ of the global luminosity.

5 Fast features extraction.

four parameters used: s7,s8,s9,s10.

With the toolbox you can easily extract these features (all these in less the 10 ms on a P3 600 MHz):

- surface
- height
- width

- aspect ratio
- (number of bright pixel)/(number of dark pixel)
- lowest gray level
- highest gray level
- number of peaks in the x direction.
  For each column of the image, we calculate the average of the gray level. This average follows a curve $C1$. To be less sensitive to the noise, we count the number of peaks of the curve $C2$ with $C2(x) = floor(C1(x)/s7)$.
- number of peaks in the y direction.
  For each column of the image, we calculate the average of the gray level. This average follows a curve $C1$. To be less sensitive to the noise, we count the number of peaks of the curve $C2$ with $C2(y) = floor(C1(y)/s8)$.
- number of radial peaks.
  For each column of the image, we calculate the average of the gray level. This average follows a curve $C1$. To be less sensitive to the noise, we count the number of peaks of the curve $C2$ with $C2(r) = floor(C1(r)/s9)$.
- number of angular peaks.
  For each column of the image, we calculate the average of the gray level. This average follows a curve $C1$. To be less sensitive to the noise, we count the number of peaks of the curve $C2$ with $C2(\theta) = floor(C1(\theta)/s10)$.

6 Slow feature extraction: image correlations.

The correlation of an image $I1$ of size $(X1, Y1)$ with an image $I2$ of size $(X2, Y2)$ is an image IC of size $(X1, Y1)$ (non-commutativity).

$$
\begin{aligned}
IC(x,y) &= Corr_{(I1,I2)}(x,y) && (3)\\
&= I1 * (I2)' && (4)\\
&= \sum_{i=1}^{i=X2} \sum_{j=1}^{j=Y2} I1(i+x, j+y)\ I2(i,j) && (5)\\
& && (6)
\end{aligned}
$$

with $x = 1, ..., X1$ et $y = 1, ..., Y1$.
$(I2)$' : the image $I2$ after a rotation of 180.
* : convolution operator.
$I1(a,b) = 0 \ \ \forall(a > X1)$ and $\forall(b > Y1)$

The equation 5 is way to slow. We will use an other technique. For the calculation of the correlation between two images of size (128,128), the other technique is more than 100 times faster.

Let's define:

$$\mathcal{I}1 = \mathcal{F}(I1) \quad (7)$$
$$\mathcal{I}2 = \mathcal{F}(I2) \quad (8)$$
$$\mathcal{I}C = \mathcal{F}(IC) \quad (9)$$
$$(10)$$

with $\mathcal{F}(I)$ the Fourier transform of the image I.

We have:

$$IC = I1 * (I2)' \quad (11)$$
$$\mathcal{I}C(i,j) = \mathcal{I}1(i,j)\ (\mathcal{I}2(i,j))^* \quad (12)$$

with x^*= transpose of x.

La convolution operator is a simple multiplication in the Fourier domain. For each point, we calculate only a single multiplication.

Now, we have $\mathcal{I}C$. we want IC. We still have to calculate an inverse Fourier Transform:

$$IC = \mathcal{F}^{-1}(\mathcal{I}C) \quad (13)$$

The Fourier transforms are calculated using an algorithm called FFT: "Fast Fourier Transform". This algorithm imposes that all the dimension of the images are exactly a power of two: 2,4,8,16,32,64,128,256,512,... We will pad the images to have an allowed size (with zero's).
The correlation of two images is available only for images where the gray level is coded on a double ("ImageD" class). There is a simple function which creates "ImageD" images from normal images where the gray level is coded in integer on 8 bits (normal "Image" class).

It's interesting the calculate the correlation of the images that you want to classify with "prototype" images to see "how close they look like". This feature is very slow to calculate but it usually contains lots of useful information. The result of the correlation calculation is an image. From this image, we will only use (as a feature to inject inside a classifier) the greatest value of the color of the pixels of IC.