Chapter 8

Extracting the Component's Contours for Calculating Number of Objects

On successful completion of this course, students will be able to:

- Explain how to extract component's contours.
- Develop a program to count an object in image.

Introduction

Image processing for obtaining object‘s information in an image is important part in digital era. Important images generally contain representation of specific objects. In order to perform a content-based analysis of an image, it is necessary to extract meaningful features from the collection of pixels and contours that constitute the image. Contours are fundamental image elements that define an image's content. In this paper, we propose a method for calculating number of objects using computer vision based on contours and shape descriptor of image.

Introduction of Contours

Images generally contain representation of objects. One of the goals of image analysis is to identify and extract those objects. In object detection/recognition applications, the first step is to produce a binary image showing where certain objects of interest could be located. The next step is to then extract the objects which are contain in this collection of 1s and 0s. More specifically, we will extract the connected components, that is, shapes made of a set of connected pixels in a binary image. In this paper, we propose a method for calculating objects in an image using contour and shape descriptor. The implementation of this method is such as to get information about traffic density, how many cars in a street.

The contours are extracted by a simple algorithm that consists of systematically scanning the image until a component is hit. From this starting point of the component, its contour is followed, marking the pixels on its border. When the contour is completed, the scanning resumes at the last position until a new component is found. The identified connected components can then be individually analyzed. Implementation of image segmentation for extracting foreground object can be use GraphCut algorithm based on mathematical morphology [3]. GrabCut is computationally more expensive than watershed, but it generally produces a more accurate result. It is the best algorithm to use when one wants to extract a foreground object in a still image. So for this research, we propose simple mechanism for calculating the objects in an image, by processing the contour and the shape descriptor of an image using connected component.

Shape descriptors are important tools in content-based image retrieval systems, which allow searching and browsing images in a database with respect to the shape information. The shape description methods can be divided into three main categories; contour based, image based and skeleton based descriptors [5]. A Connected component often corresponds to the image of some object in a pictured scene. To identify this object, or to compare it with other image elements, it can be useful to perform some measurements on the component in order to extract some of its characteristics. Many OpenCV functions are available when it comes to shape descriptor and offers a simple function which extracts the contours of the connected components of an image using cv::findContours function.

```
Cv::findCounteours (image,
CV_RETR_EXTERNAL, //retrieve the external contours
CV_CHAIN_APROX_NONE); // all pixels of each contours.
```

For example, if some prior knowledge is available about the expected size of objects of interest, it becomes possible to eliminate some of the components. Let’s then use a minimum and a maximum value for the perimeter of the components by iterating over the vector of contours and eliminating the eliminating the invalid components. The implementation for finding contours shown in the program below:

```
// Eliminate too short or too long contours
intcmin=100;int cmax=1000 //min and max contour length
std::vector<std::vector<cv::Point>>::
const_iteratoritc=contours.begin();
while (itc !=contours.end()) {
if (itc->size()<cmin ||itc->size() >cmax)
      itc=contour//Eliminate.erase(itc);
else
      ++itc;
}
```

Counting Objects

We using singe images for the experiment, and convert it first to binary, then we extract the contour, shape descriptor, counting and displaying number of objects. The algorithm for extract the contours shown below:

Algorithm 8.1. `Extracting the contour and counting the object:`

```
Begin
 Counter=0
   reading image and convert to binary image
   Extract contours
   Output the vector of contours
   Counting the objects
   Couter+=1
   Displaying the number of object
End
```

The diagram block of our method shown in figure 8.1:

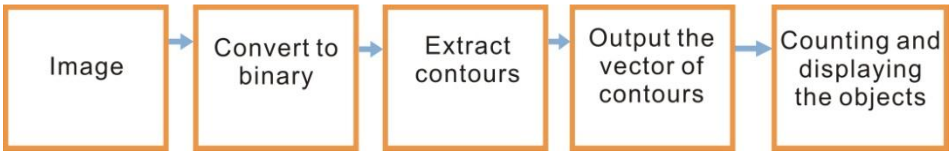
Image
Convert to binary
Extract contours
Output the vector of contours
Counting and displaying the objects

Figure 8.1 *Diagram block of the system for calculating number of objects.*

We test an example image with 4 animal and size 400x600 pixel as shown in figure 8.2:

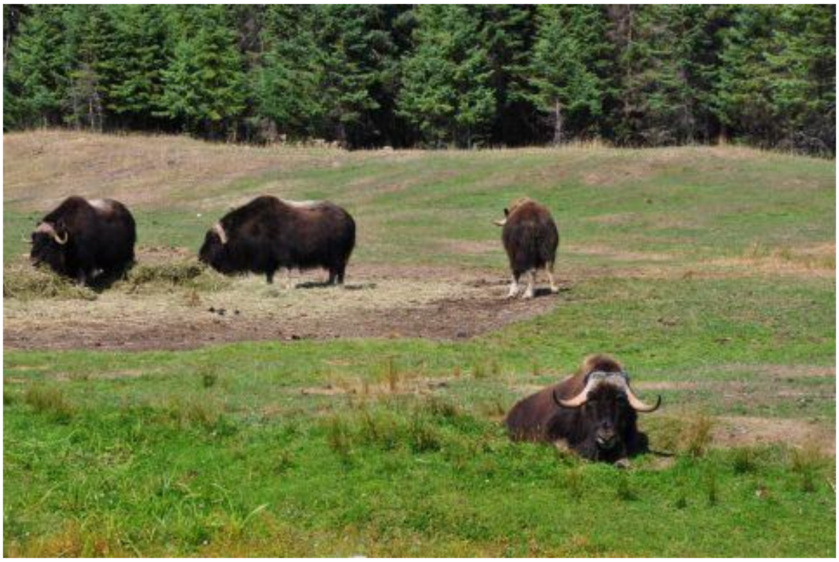

Figure 8.2 *Testing image.*

Then the result of binary image from fig. 8.2 shown in figure 8.3:

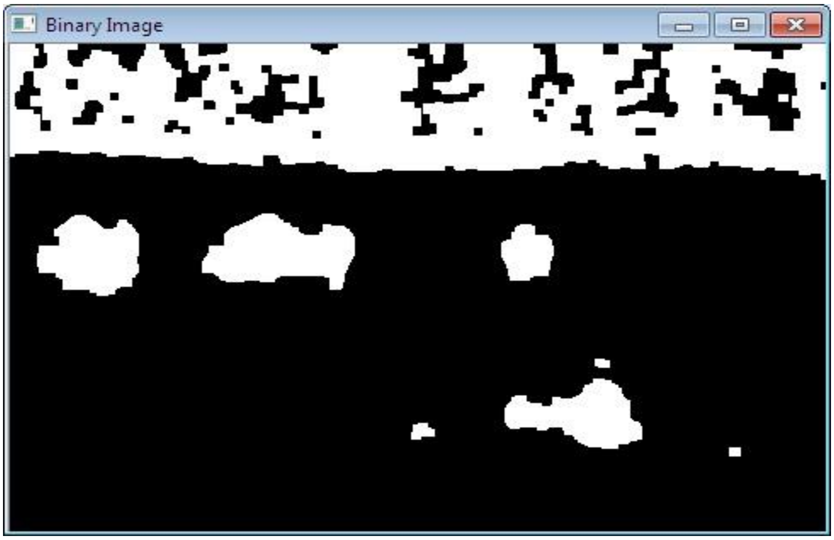

Figure 8.3 *Binary image.*

After that, we find contours of image as shown in figure 8.4:

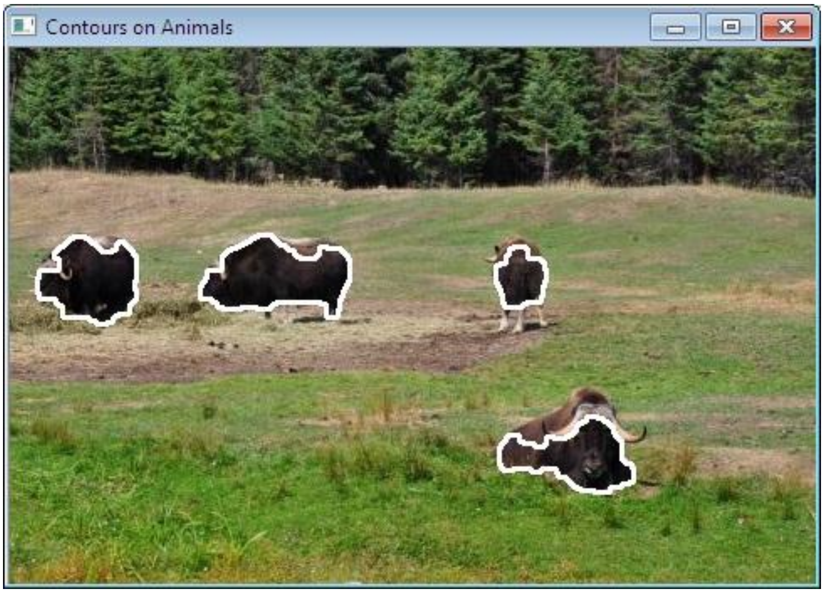

Figure 8.4 *Contour of image obtained.*

After that, we got total number of object with its shape descriptors as shown in figure 8.5, using function:

```
CvPoint pt1;
pt1.x=100;pt1.y=60;
cvInitFont( &font, CV_FONT_HERSHEY_COMPLEX, 0.5, 0.5, 0.0, 1, CV_AA );
cvPutText( img, "TOTAL :", varCount, &font , CV_RGB(0,0,255) );
```

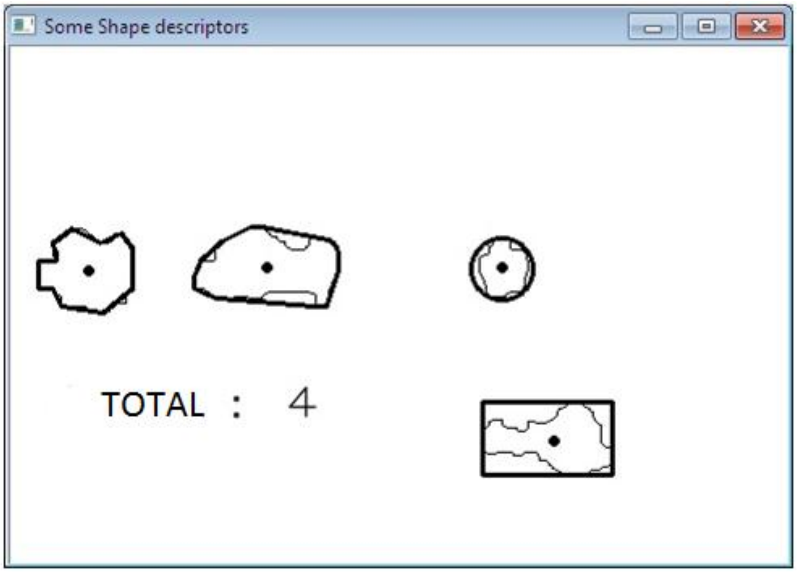

Figure 8.5 *Number of objects obtained with processing time less than 1 second.*

References

[1] Robert Laganière, OpenCV 2 Computer Vision Application Programming Cookbook, Apress Publisher, 2011.

[2] R.C. Gonzalez, Digital Image Processing (3rd ed.), Addison Wesley, 2007.

[3] Rother, A Blake, GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts, ACM Transaction on Graphics, vol. 23 no. 3, 2004.

[4] R. Szeliski, Computer Vision, Algorithms and Applications, Springer Publisher, 2011.

[5] Latecki, L. J., Lakamper, R., Shape Similarity Measure Based on Correspondence of Visual Parts.IEEE Transaction. PAMI, 22, 10, pp.1185-1190, 2000.