4. Geometrical features of binary objects

4.1. Introduction

This lab work presents some important geometric properties of binary images and the algorithms used for computing them. The properties described are the area, the center of mass, the elongation axis, the perimeter, the thinness ratio, the aspect ratio and the projections of the binary image.

4.2. Theoretical considerations

After applying segmentation and labeling algorithms, we obtain a new image where each object can be referred separately.

An object 'i' in the image is described by the function:

$$I_i(r,c) = \begin{cases} 1, & \text{if } I(r,c) \in \text{object labeled 'i'} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{where } r \in [0...Height-1] \text{ and } c \in [0...Width-1]$$

The geometric properties of the objects can be classified into two categories:

- position and orientation properties: the center of mass, the area, the perimeter, the elongation axis
- shape properties: aspect ratio, thinness ratio, Euler's number, the projections, the Feret diameters of the objects

4.2.1. Area

$$A_i = \sum_{r=0}^{H-1}\sum_{c=0}^{W-1} I_i(r,c)$$

The area A_i is measured in pixels and it indicates the relative size of the object.

4.2.2. The center of mass

$$\overline{r_i} = \frac{1}{A_i}\sum_{r=0}^{H-1}\sum_{c=0}^{W-1} rI_i(r,c)$$

$$\overline{c_i} = \frac{1}{A_i}\sum_{r=0}^{H-1}\sum_{c=0}^{W-1} cI_i(r,c)$$

The equations above correspond to the row and column where the center of mass is located. This attribute helps us locate the object in a bi-dimensional image.

4.2.3. The axis of elongation (the axis of least second order moment)

$$\tan(2\varphi_i) = \frac{2\sum_{r=0}^{H-1}\sum_{c=0}^{W-1}(r-\overline{r_i})(c-\overline{c_i})I_i(r,c)}{\sum_{r=0}^{H-1}\sum_{c=0}^{W-1}(c-\overline{c_i})^2 I_i(r,c) - \sum_{r=0}^{H-1}\sum_{c=0}^{W-1}(r-\overline{r_i})^2 I_i(r,c)}$$

If both the nominator and the denominator of the above equation are equal to zero, then the object has a circular symmetry, and any line that passes through the center of mass is a symmetry axis.

For finding the direction of the line (the angle) one must apply the arctangent function. The arctangent is defined on the interval (-∞, +∞) and it takes values in the interval (-π/2, π/2). The evaluation of the arctangent becomes unstable when the denominator of the fraction tends to zero.

The signs of the numerator and of the denominator are important for determining the right quadrant in which the result lays. The arctangent function does not make the difference between directions that are opposed. For this reason, the usage of the function "atan2" is suggested. The "atan2" function has as arguments the numerator and the denominator of such fraction, and it returns a result in the interval (-π, π).

The axis of elongation gives information about how the object is positioned in the field of view, more exactly, its orientation. The axis corresponds to the direction in which the object (seen as a plane surface of constant width) can rotate most easily (has a minimum kinetic moment).

After the φ_i angle is found, the correctness of the resulted value can be validated by drawing the axis of elongation. The axis of elongation will correspond to the line that pass through the center of mass and determines the φ_i angle with Ox axis.

4.2.4. The perimeter

The perimeter of the object helps us determine the position of the object in space and it also gives information about the shape of the object. The perimeter can be computed by counting the number of pixels on the contour (pixels of value 1 and having at least one neighbor pixel of value 0).

A first approach to contour detection is the scanning of the image, line by line and counting the number of pixels in the object that satisfy the condition mentioned above. A main disadvantage of this method is that we cannot distinguish the exterior contour from the interior contours (if they exist they are generated by the holes in the object). As the pixels of digital images represent distributions on a rectangular raster, the length of curves and oblique lines in the image cannot be correctly estimated by counting the pixels. A first correction is given by the multiplication by π/4 of the perimeter that resulted in the previous algorithm. There are other methods for length correction. These methods take into account the type of neighborhood used (4 neighbors, 8 neighbors etc.).

Another method for detecting the contour of an object involves the usage of an existing algorithm for edge detection, the thinning of the edges until they become 1 pixel thick and in the end the counting of the resulted edge pixels.

Methods of type "chain-codes" represent complex methods for contour detection and offer a high accuracy.

4.2.5. The thinness ratio (circularity)

$$T = 4\pi\left(\frac{A}{P^2}\right)$$

The function above has the maximum value equal to 1, and for this value we obtain a circle. The thinness ratio is used for determining how "round" an object is. If the value of T is close to 1, the object tends to be round.

The value of the thinness ratio also offers information on how regular an object is. The objects that have a regular contour have a greater value of T than the objects of irregular contours. The value 1/T is called irregularity factor of the object (or compactness factor).

4.2.6. The aspect ratio

This property is found by scanning the image and keeping the minimum and maximum values of the lines and columns that form the rectangle circumscribed to the object.

$$R = \frac{c_{\max} - c_{\min} + 1}{r_{\max} - r_{\min} + 1}$$

4.2.7. The projections of the binary object

The projections give information about the shape of the object. The horizontal projection equals the sum of pixels computed on each line of the image, and the vertical projection is given by the sum of the pixels on the columns.

$$h_i(r) = \sum_{c=0}^{W-1} I_i(r,c) \qquad v_i(c) = \sum_{r=0}^{H-1} I_i(r,c)$$

The projections are used in applications of text recognition in which the interest object can be normalized.

4.3. Implementation details

In order to distinguish between the various objects present in an image, we will suppose that each one of them is painted using a different color. These colors may be the result of a previous labeling step, or may be generated manually (see Fig. 4.1).

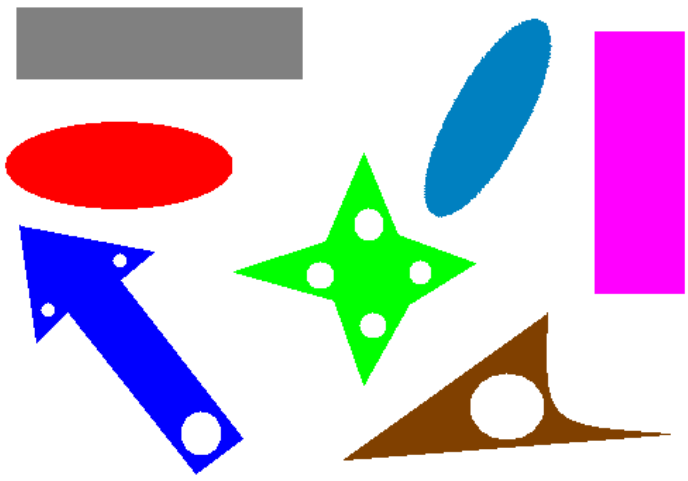

Fig. 4.1 Example of a labeled image on which the described algorithms could be tested

There are various approaches for implementing the geometrical properties extractors:

4.3.1. Compute the geometrical features for all objects in an image at once

For each object, the compound pixels are selected based on the object unique label (color) and the corresponding geometrical features are computed. This procedure is applied to each object from the input labeled image.

4.3.2. Compute the geometrical features for a specific object selected with the mouse

The user should position the mouse pointer over a pixel belonging to the desired object and *click* on it. In response to this action, the geometrical features of the desired object should be computed and displayed in the standard output.

In order to add an event *handler*, we will use the `setMouseCallback` function from OpenCV which has the role to set a *handler* for the mouse in a specific window.

```
void setMouseCallback(const string& winname, MouseCallback onMouse, void* userdata=0)
```

winname – window title,
onMouse – *callback* function name that is called when a mouse event occurs on the **winname** window
userdata – optional parameter that may be passed to the *callback* function.

The computation of the desired features will be implemented in **onMouse** function.

```
void onMouse (int event, int x, int y, int flags, void* param)
```

event – is the mouse event and can take the following values:
- EVENT_MOUSEMOVE
- EVENT_LBUTTONDOWN
- EVENT_RBUTTONDOWN
- EVENT_MBUTTONDOWN
- EVENT_LBUTTONUP
- EVENT_RBUTTONUP
- EVENT_MBUTTONUP
- EVENT_LBUTTONDBLCLK
- EVENT_RBUTTONDBLCLK
- EVENT_MBUTTONDBLCLK

x, y – are the x and y coordinates where the event occurred,
flags – specific condition whenever a mouse event occurs,
param – corresponds to the userdata pointer passed through **setMouseCallback** function.

In *OpenCVApplication* framework, an example of event handler is presented in the **testMouseClick()** function.

In order to draw the elongation axis, use the **line** function from OpenCV to draw the line:

```
void line( Mat img,  Point pStart, Point pEnd, Scalar color, int thickness )
```

img – image where the line segment is drawn
pStart, **pEnd** – the two points that define the line segment
color – line color
thickness – line thickness

4.4. **Practical work**

1. For a specific object in a labeled image selected by a mouse *click*, compute the object's area, center of mass, axis of elongation, perimeter, thinness ratio and aspect ratio.
    a. Display the results in the standard output
    b. In a separate image (source image clone):
        - Draw the contour points of the selected object
        - Display the center of mass of the selected object
        - Display the axis of elongation of the selected object by using the *line* function from OpenCV.
    c. Compute and display the projections of the selected object in a separate image (source image clone).
2. Create a new processing function which takes as input a labeled image and keeps in the output image only the objects that:
    a. have their *area* < *TH_area*
    b. have a specific orientation *phi*, where *phi_LOW* < *phi* < *phi_HIGH*

    where *TH_area*, *phi_LOW*, *phi_HIGH* are given by the user.
3. **Save your work. Use the same application in the next laboratories. At the end of the image processing laboratory you should present your own application with the implemented algorithms!!!**

4.5. Bibliography

[1] Umbaugh Scot E, *Computer Vision and Image Processing*, Prentice Hall, NJ, 1998, ISBN 0-13-264599-8.