Université de Lille

SHAPE ATTRIBUTES

RECONNAISSANCE DES FORMES

Iacob Sergiu

26th January 2020

CHAPTER 1

INTRODUCTION

1.1 CONTEXT

This report is the result of a university assignment. It aims to prove that the student understood the motivation, goals and means of studying pattern recognition. Therefore, this is a way of summarizing a series of observations and experiments done on images containing shapes.

1.2 MOTIVATION

We want to be able to extract shapes from images and differentiate between them. Using various **shape attributes**, we should be able to conclude which shapes are similar, which are different, which of them are the same but just rotated at different angles, what their main axis of inertia is and so on. Being able to recognise these attributes is a small but important step towards us to identifying and categorising shapes from images. From there on, many posibilities exist: image indexing, searching for certain shapes (e.g. “images with squares”) etc.

1.3 GOALS

Given multiple shapes, $S_1, \ldots, S_n$, each shape S_i being represented by its image pixels, find proper **shape indexes** that would allow us to classify these shapes and conclude on a shape’s features (e.g. main axis of inertia for S_i). Therefore, we are interested in finding means (that is **shape attributes**, **shape invariants**) to uniquely identify them. The **shape invariants** would help us say that S_i and $S_j, i \neq j$ are the same, even if S_j is rotated at a certain angle, for example.

CHAPTER 2

EXPERIMENTS

2.1 TECHNICAL ACKNOWLEDGEMENTS

The following code presented was written using *R*. The images from which the shapes were extracted were used as gray images. 2.1

2.2 LOADING DATA

Images are loaded then converted to grayscale. Right now, we are not interested in their color, so they'll only be black and white.

```
rdfReadGreyImage <- function (nom) {
    image <- readImage (paste('images/', nom, sep=''))
    if (length (dim (image)) == 2) {
        image
    } else {
        channel (image, 'red')
    }
}
```

LISTING 2.1
Loading images in R

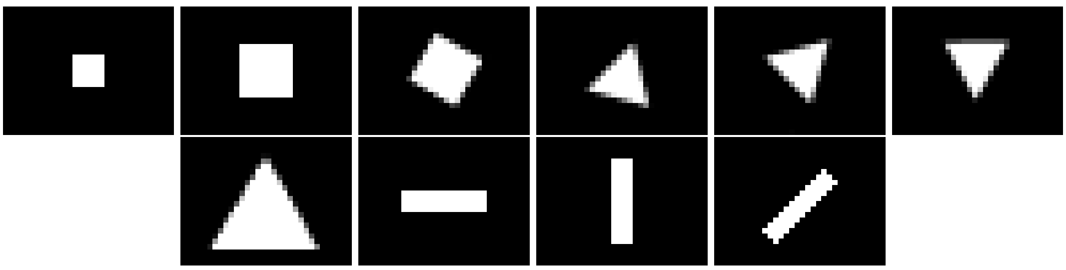

FIGURE 2.1
Shape images

2.3 MOMENTS OF A SHAPE

2.3.1 PREREQUISITES

We define the *moment of a shape* as the following:

$$M_{ij} = \sum_x \sum_y x^i y^j A(x, y)$$

where $A(x, y)$ is the value of the pixel (x, y) of the shape's image. One can easily observe that M_{00} is equal to the number of pixels that are not black. We call that the *surface* of the shape.

```
rdfMoment <- function (im, p, q) {
    x <- (1 : (dim (im)[1])) ^ p
    y <- (1 : (dim (im)[2])) ^ q
    as.numeric (rbind (x) %*% im %*% cbind (y))
    }
```

LISTING 2.2
Calculating the moment of a shape

The *barycentre* is defined as being the following:

$$(\bar{x}, \bar{y}) = (\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}})$$

Having these, we can now make our M_{ij} invariant to the translation of the shape. We can define the *centered moment* as being:

$$\mu_{ij} = \sum_x \sum_y (x - \bar{x})^i (y - \bar{y})^j I(x, y)$$

```
rdfMomentCentre <- function (im, p, q) {
    # Barycentre
    s <- rdfSurface (im)
    cx <- rdfMoment (im, 1, 0) / s
    cy <- rdfMoment (im, 0, 1) / s
    # Initialiser les vecteurs x et y
    x <- (1 : (dim (im)[1]) - cx) ^ p
    y <- (1 : (dim (im)[2]) - cy) ^ q
    # Calcul du moment centre
    as.numeric (rbind (x) %*% im %*% cbind (y))
  }
```

LISTING 2.3
Calculating centered moments

2.3.2 INERTIA MATRIX

Inspiring ourselves from physics, we may use these moments to calculate the inertia matrix:

$$I = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}$$

We'll be using I later on to tell which is the main axis of inertia for different shapes. For now, let's take a look at its values for the images with rectangles and squares.

```
[1] "Horizontal rectangle:"
     [,1] [,2]
[1,] 1360    0
[2,]    0   80
[1] "Vertical rectangle:"
     [,1] [,2]
[1,]   80    0
[2,]    0 1360
[1] "Diagonal rectangle:"
       [,1]   [,2]
[1,]  678.5 -619.5
[2,] -619.5  678.5
[1] "Smooth diagonal rectangle:"
          [,1]      [,2]
[1,]  745.0080 -647.0326
[2,] -647.0326  748.4077
```

```
[1] "Small square:"
     [,1] [,2]
[1,]  105    0
[2,]    0  105
[1] "Big square:"
     [,1] [,2]
[1,]  825    0
[2,]    0  825
[1] "Big square 30deg:"
            [,1]        [,2]
[1,] 842.4965158   0.2446836
[2,]   0.2446836 843.2052085
[1] "Big square 45deg:"
           [,1]       [,2]
[1,] 840.336263   1.458065
[2,]   1.458065 839.716792
```

FIGURE 2.2
Inertia matrices for rectangles and squares

Upon calculating the values for the rectangles, we can observe that their matrix of inertia is different for each shape. For the horizontal rectangle, it's intuitive that the following equation should happen: $\mu_{20} > \mu_{02}$, and for the vertical rectangle it should be the other way aroung. The inertia matrix tells us just that. For the diagonal rectangle, there's an interesting observation: $\mu_{20} = \mu_{02}$. That's because the rectangle is rotated at 45 degrees, so the pixels are uniformely distributed horizontally and vertically. For the two diagonal rectangles, the surface differs, but it's interesting to notice that the proportion $\frac{\mu_{20}}{\mu_{02}}$ is about the same. That would mean that the rectangles are similarly oriented, which is true.

The $\frac{\mu_{20}}{\mu_{02}}$ proportion is also kept in the case of the the squares which aren't rotated. For the other ones, the values

Chapter 3

Results

CHAPTER 4

DISCUSSION

CHAPTER 5

CONCLUSION