Université de Lille

SHAPE ATTRIBUTES

RECONNAISSANCE DES FORMES

Iacob Sergiu

27th 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, $S_1, \dots, S_n$, 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. **??**

2.2 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.

```r
    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

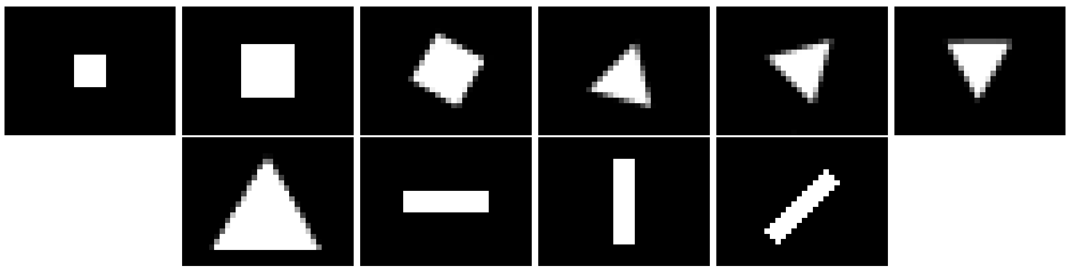


Figure 2.1
Shape images

2.3 MOMENTS OF A SHAPE

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 *barycenter* 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))
  }
```

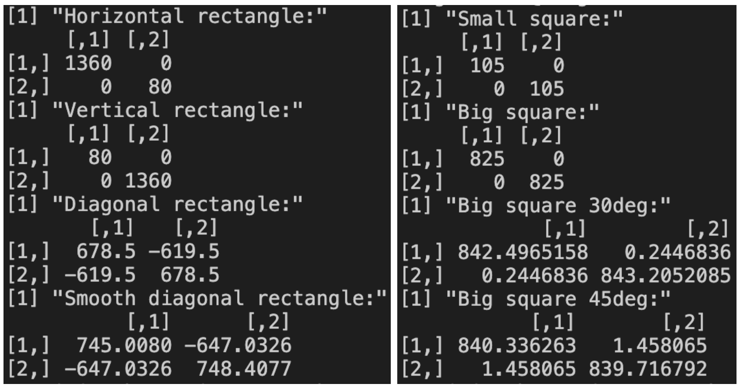
LISTING 2.3
Calculating centered moments

2.4 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: $\mu_{20} > \mu_{02}$. That's because there are more pixels distributed over the x axis. For the vertical rectangle, it's the opposite. 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 values differ, 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. Another interesting observation is that for the shapes that aren't rotated, $\mu_{11} = 0$. All of these already give us valuable information as to how the shape looks, so they can definitely be used as shape attributes. However, we can still improve these.

2.5 SHAPE'S INERTIA

2.5.1 NORMALISED MOMENT CENTERS

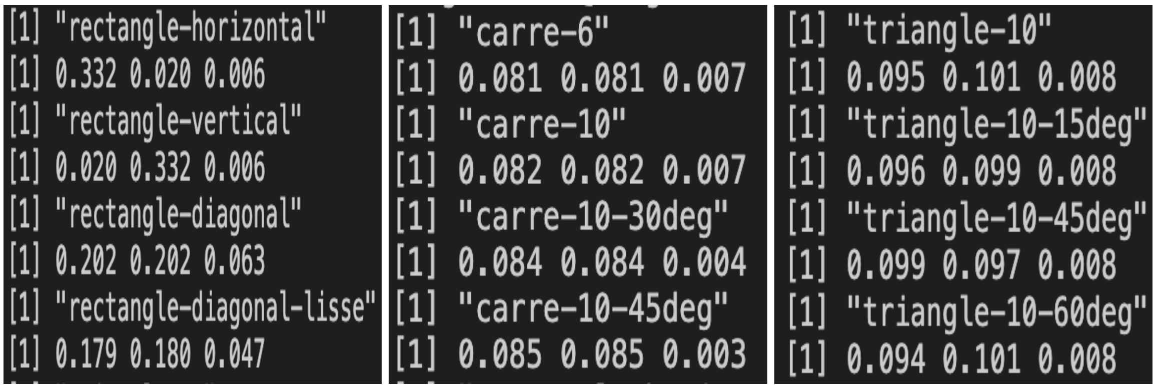
For our next trick, we can make μ_{ij} invariant to the scale of the image. We define the *normalized centered moment* as the following:

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{1+\frac{i+j}{2}}}$$

```
rdfMomentCentreNormalise <- function (img, p, q){
    upq = rdfMomentCentre(img, p, q)
    u00 = rdfMomentCentre(img, 0, 0)
    normalised = upq / (u00 ** (1 + (p + q)/2))
    normalised
}
```

LISTING 2.4
Calculating normalised centered moments

Taking a look at the values[1], we can notice that η_{22}, for example, becomes invariant to the size of the image (for example for the triangles). η_{20} and η_{02} still tell us how pixels are distributed over the x and y axis, but they are now not influenced by the shape's size.



```
[1] "rectangle-horizontal"
[1] 0.332 0.020 0.006
[1] "rectangle-vertical"
[1] 0.020 0.332 0.006
[1] "rectangle-diagonal"
[1] 0.202 0.202 0.063
[1] "rectangle-diagonal-lisse"
[1] 0.179 0.180 0.047
```

```
[1] "carre-6"
[1] 0.081 0.081 0.007
[1] "carre-10"
[1] 0.082 0.082 0.007
[1] "carre-10-30deg"
[1] 0.084 0.084 0.004
[1] "carre-10-45deg"
[1] 0.085 0.085 0.003
```

```
[1] "triangle-10"
[1] 0.095 0.101 0.008
[1] "triangle-10-15deg"
[1] 0.096 0.099 0.008
[1] "triangle-10-45deg"
[1] 0.099 0.097 0.008
[1] "triangle-10-60deg"
[1] 0.094 0.101 0.008
```

FIGURE 2.3
Normalised moment centers

2.5.2 MAIN MOMENTS OF INERTIA

Using the *normalised centered moments*, rather than just *centered moments*, we can calculate:

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

Afterwards, we can calculate the *eigen values* of the matrix I. The values obtained, I_1 and I_2 are the *main moments of inertia*:

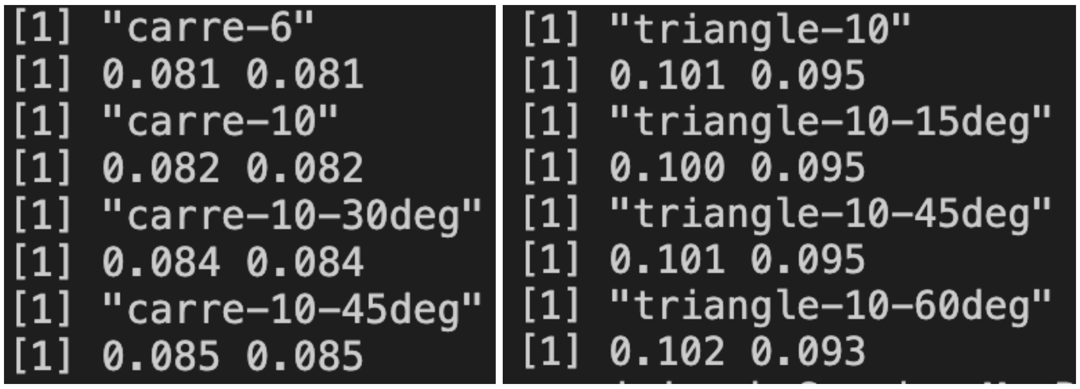
$$I' = \begin{pmatrix} I_1 & 0 \\ 0 & I_2 \end{pmatrix}$$

```r
mainInertionAxis <- function(im, normalise=TRUE){
    if (normalise == TRUE)
        f <- rdfMomentCentreNormalise
    else
        f <- rdfMomentCentre
    u20 <- f(im, 2, 0)
    u11 <- f(im, 1, 1)
    u02 <- f(im, 0, 2)
    I <- matrix(c(u20, u11, u11, u02), nrow=2, ncol=2)
    ev <- eigen(I)
    tenseur <- diag(ev$values)
}
```

LISTING 2.5
Calculating the main moments of inertia

[1]The values were rounded to 3 decimals

Let's have a look at how I_1 and I_2 look for the shapes above. We can see that these values[2] tell us how much "inertia" there is on each of the shape's axis. However, we have no information as to how shapes are rotated. For that, we can look at the *eigen vectors* of the I matrix.



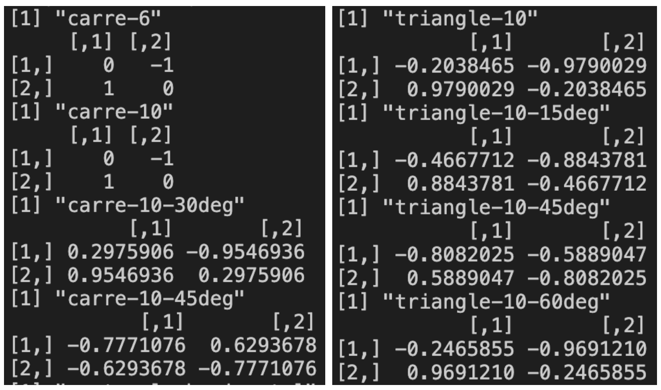
```
[1] "carre-6"
[1] 0.081 0.081
[1] "carre-10"
[1] 0.082 0.082
[1] "carre-10-30deg"
[1] 0.084 0.084
[1] "carre-10-45deg"
[1] 0.085 0.085
```

```
[1] "triangle-10"
[1] 0.101 0.095
[1] "triangle-10-15deg"
[1] 0.100 0.095
[1] "triangle-10-45deg"
[1] 0.101 0.095
[1] "triangle-10-60deg"
[1] 0.102 0.093
```

FIGURE 2.4
Main inertia moments

2.5.3 MAIN AXIS OF INERTIA

The *eigen vectors* of the I matrix can tell us how the shapes are oriented. We can clearly see that the 2 squares that are not rotated have the same *eigen vectors*. The triangles, however, all have different vectors because they are all rotated in a different way, but their *main moments of inertia* are the same.



```
[1] "carre-6"
     [,1] [,2]
[1,]    0   -1
[2,]    1    0
[1] "carre-10"
     [,1] [,2]
[1,]    0   -1
[2,]    1    0
[1] "carre-10-30deg"
          [,1]       [,2]
[1,] 0.2975906 -0.9546936
[2,] 0.9546936  0.2975906
[1] "carre-10-45deg"
           [,1]       [,2]
[1,] -0.7771076  0.6293678
[2,] -0.6293678 -0.7771076
```

```
[1] "triangle-10"
           [,1]       [,2]
[1,] -0.2038465 -0.9790029
[2,]  0.9790029 -0.2038465
[1] "triangle-10-15deg"
           [,1]       [,2]
[1,] -0.4667712 -0.8843781
[2,]  0.8843781 -0.4667712
[1] "triangle-10-45deg"
           [,1]       [,2]
[1,] -0.8082025 -0.5889047
[2,]  0.5889047 -0.8082025
[1] "triangle-10-60deg"
           [,1]       [,2]
[1,] -0.2465855 -0.9691210
[2,]  0.9691210 -0.2465855
```

FIGURE 2.5
Main inertia axis

Having made these observation, we can conclude that these attributes are fairly important to describe a shape. They can tell us how their "weight" is distributed over their *main axis of inertia*.

[2]The values were rounded to 3 decimals

2.6 HU INVARIANTS

Using the *normalised center moments*, we can define the *Hu invariants*. These are invariants with respect to translation, scale, and rotation. Please refer to NANNANANANA for more information.

Analysing the invariants, we can see that...

CHAPTER 3

CONCLUSION