Shape Contours

Reconnaissance des Formes

Iacob Sergiu

4th February 2020

Contents

1 Introduction **2**

1.1 Context . . . 2

1.2 Motivation . . . 2

1.3 Goals . . . 2

2 Experiments **3**

2.1 Technical acknowledgements . . . 3

2.2 Loading data . . . 3

2.3 Fourier Descriptors . . . 4

2.3.1 Outer contour points . . . 4

2.3.2 Cancelling Fourier descriptors . . . 6

2.4 Simplifying the shape contour . . . 7

3 Conclusion **8**

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

Shapes are everywhere and of many kinds. Bigger, smaller, rotated, symmetric or not and so on. But what about their contours? How would we know how exactly to draw such a shape? This is the motivation behind trying to code and decode a *shape's contour*, so one may easily reconstruct such a shape or, often times very important, *simplify* them.

1.3 GOALS

Suppose we have multiple shapes $S_1, \ldots, S_n$, each shape S_i described by the *outer points* of its contour. We need to come up with ways to code these points and also simplify them, that is to reduce the number of points. That would allow us to reduce the memory needed to serialise these shapes, but at the cost of precision. Therefore, if we can somehow *adjust* the trade-off between simplicity and precision, we would have a rather versatile method to work with shape contours that could fit our needs, based on the context we find ourselves in.

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

In the first phase, we'll be working with files containing pre-defined contour points. These are just pairs (x, y) representing the coordinates of the outer points of a shape's contour. To simplify things, we'll transform the coordinates to complex numbers: Wikipedia 2020a.

```r
# Lit un contour dans un fichier texte
rdfChargeFichierContour <- function (nom) {
    contour <- read.table (nom)
    complex (real = contour$V1, imaginary = contour$V2)
}
```

Listing 2.1
Loading outer contour points

Images used 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 2.2
Loading images

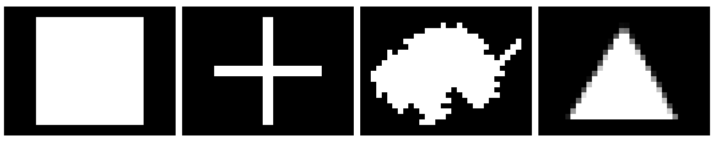


Figure 2.1
Shape images

2.3 FOURIER DESCRIPTORS

2.3.1 OUTER CONTOUR POINTS

Let's take a look at how one of our initial shape looks like. The red contour contains all of our points. The blue one is drawn between each 4th point, and the green one is drawn between each 8th point.

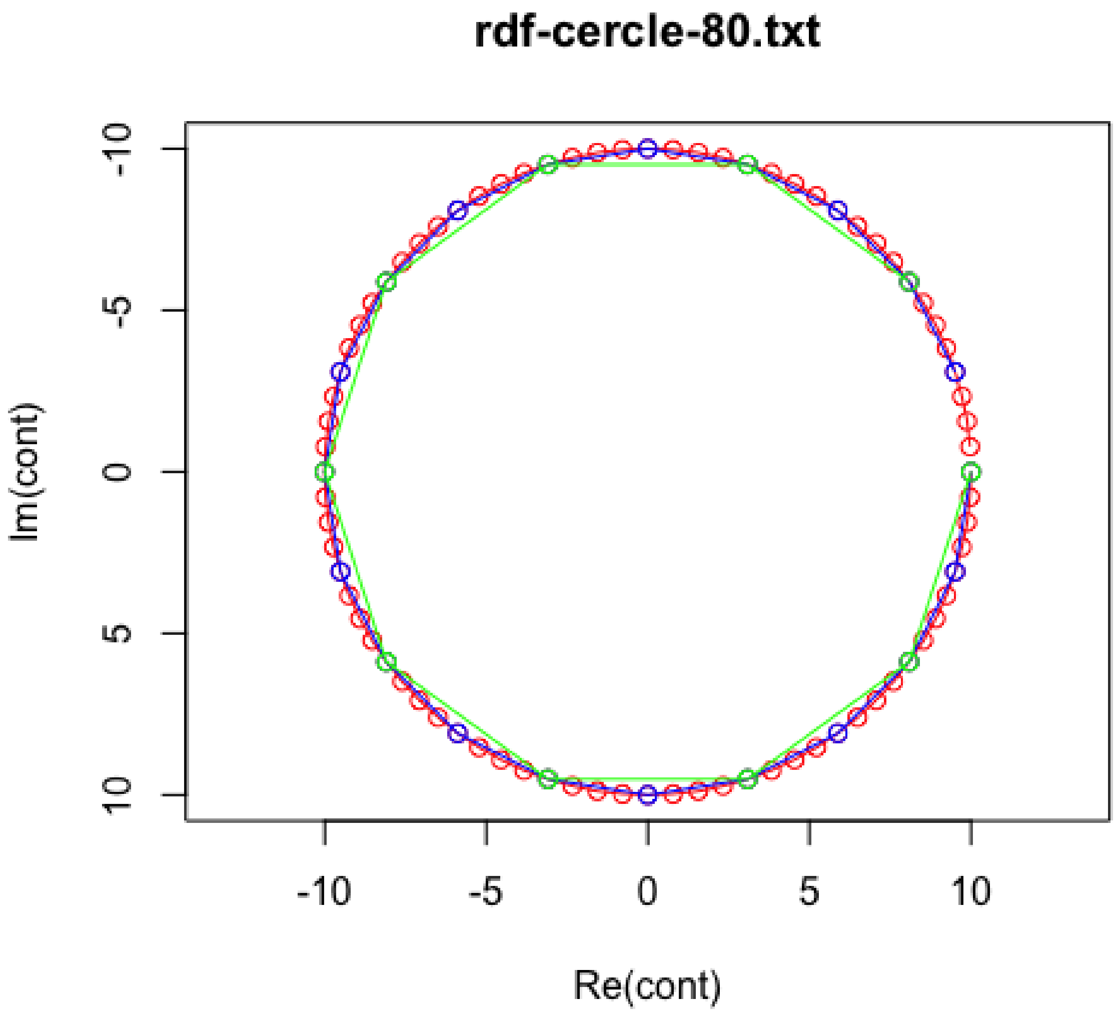


FIGURE 2.2
Circle contour

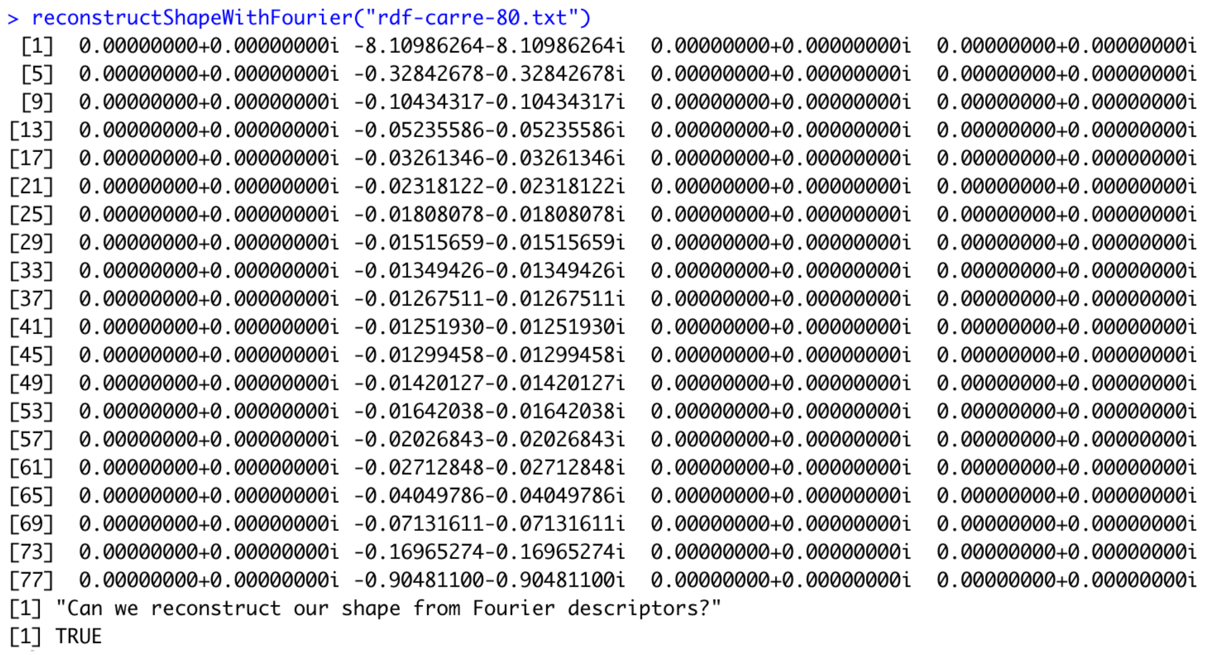
If we apply the *Fast Fourier Transform* on our contour points, we'll get our *Fourier descriptors*[1]. We can use these to describe our contour and we're even able to reconstruct our contour starting from fourier descriptors.

```
reconstructShapeWithFourier <- function(nom){
    cont <- rdfChargeFichierContour(nom)
    fourier <- fft(cont)/length(cont)
    inversed <- fft(fourier, inverse=TRUE)
    print ("Can we reconstruct our shape from Fourier descriptors?")
    print (all.equal(inversed, cont))
}
```

LISTING 2.3
Calculating Fourier descriptors

[1]The Fourier descriptors were normalised by dividing them to the length of the total contour points

Here's how the Fourier descriptors look like calculated on the contour of the square:



```
> reconstructShapeWithFourier("rdf-carre-80.txt")
 [1]  0.00000000+0.00000000i -8.10986264-8.10986264i  0.00000000+0.00000000i  0.00000000+0.00000000i
 [5]  0.00000000+0.00000000i -0.32842678-0.32842678i  0.00000000+0.00000000i  0.00000000+0.00000000i
 [9]  0.00000000+0.00000000i -0.10434317-0.10434317i  0.00000000+0.00000000i  0.00000000+0.00000000i
[13]  0.00000000+0.00000000i -0.05235586-0.05235586i  0.00000000+0.00000000i  0.00000000+0.00000000i
[17]  0.00000000+0.00000000i -0.03261346-0.03261346i  0.00000000+0.00000000i  0.00000000+0.00000000i
[21]  0.00000000+0.00000000i -0.02318122-0.02318122i  0.00000000+0.00000000i  0.00000000+0.00000000i
[25]  0.00000000+0.00000000i -0.01808078-0.01808078i  0.00000000+0.00000000i  0.00000000+0.00000000i
[29]  0.00000000+0.00000000i -0.01515659-0.01515659i  0.00000000+0.00000000i  0.00000000+0.00000000i
[33]  0.00000000+0.00000000i -0.01349426-0.01349426i  0.00000000+0.00000000i  0.00000000+0.00000000i
[37]  0.00000000+0.00000000i -0.01267511-0.01267511i  0.00000000+0.00000000i  0.00000000+0.00000000i
[41]  0.00000000+0.00000000i -0.01251930-0.01251930i  0.00000000+0.00000000i  0.00000000+0.00000000i
[45]  0.00000000+0.00000000i -0.01299458-0.01299458i  0.00000000+0.00000000i  0.00000000+0.00000000i
[49]  0.00000000+0.00000000i -0.01420127-0.01420127i  0.00000000+0.00000000i  0.00000000+0.00000000i
[53]  0.00000000+0.00000000i -0.01642038-0.01642038i  0.00000000+0.00000000i  0.00000000+0.00000000i
[57]  0.00000000+0.00000000i -0.02026843-0.02026843i  0.00000000+0.00000000i  0.00000000+0.00000000i
[61]  0.00000000+0.00000000i -0.02712848-0.02712848i  0.00000000+0.00000000i  0.00000000+0.00000000i
[65]  0.00000000+0.00000000i -0.04049786-0.04049786i  0.00000000+0.00000000i  0.00000000+0.00000000i
[69]  0.00000000+0.00000000i -0.07131611-0.07131611i  0.00000000+0.00000000i  0.00000000+0.00000000i
[73]  0.00000000+0.00000000i -0.16965274-0.16965274i  0.00000000+0.00000000i  0.00000000+0.00000000i
[77]  0.00000000+0.00000000i -0.90481100-0.90481100i  0.00000000+0.00000000i  0.00000000+0.00000000i
[1] "Can we reconstruct our shape from Fourier descriptors?"
[1] TRUE
```

FIGURE 2.3
Reconstructing contour from Fourier descriptors

As we can see, we can safely reconstruct our initial contour thanks to the *Inverse Fast Fourier Transform*: Wikipedia 2020b, FFT. An interesting observation is that the first Fourier descriptor, Z_0, represents the center of the shape. If we modify this complex number (for example $Z_0 = Z_0 + 0.5 + 3i$) we'll actually do a *translation*.

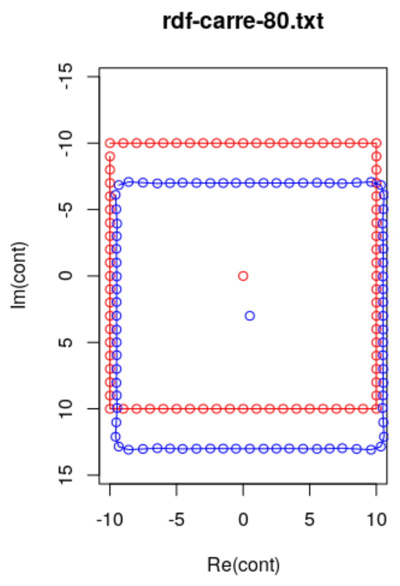


FIGURE 2.4
Modifying Z_0

2.3.2 CANCELLING FOURIER DESCRIPTORS

We can simplify our contour by “cancelling” some of the Fourier descriptors. We can do so by defining a *ratio* of descriptors that we’d like to keep. When doing so, it is important to know that we should start cancelling the descriptors from the middle of the descriptor list: Cabestaing 2020, RDF Course.

```
rdfAnnuleDescFourier <- function (desc, ratio){
    if (ratio == 1.0)
        desc

    nb_delete <- length(desc) - length(desc) * ratio
    middle <- length(desc)/2
    start <- middle - nb_delete/2
    end <- middle + nb_delete/2
    desc[start:end] <- 0
    desc
}
```

LISTING 2.4
Simplifying contour

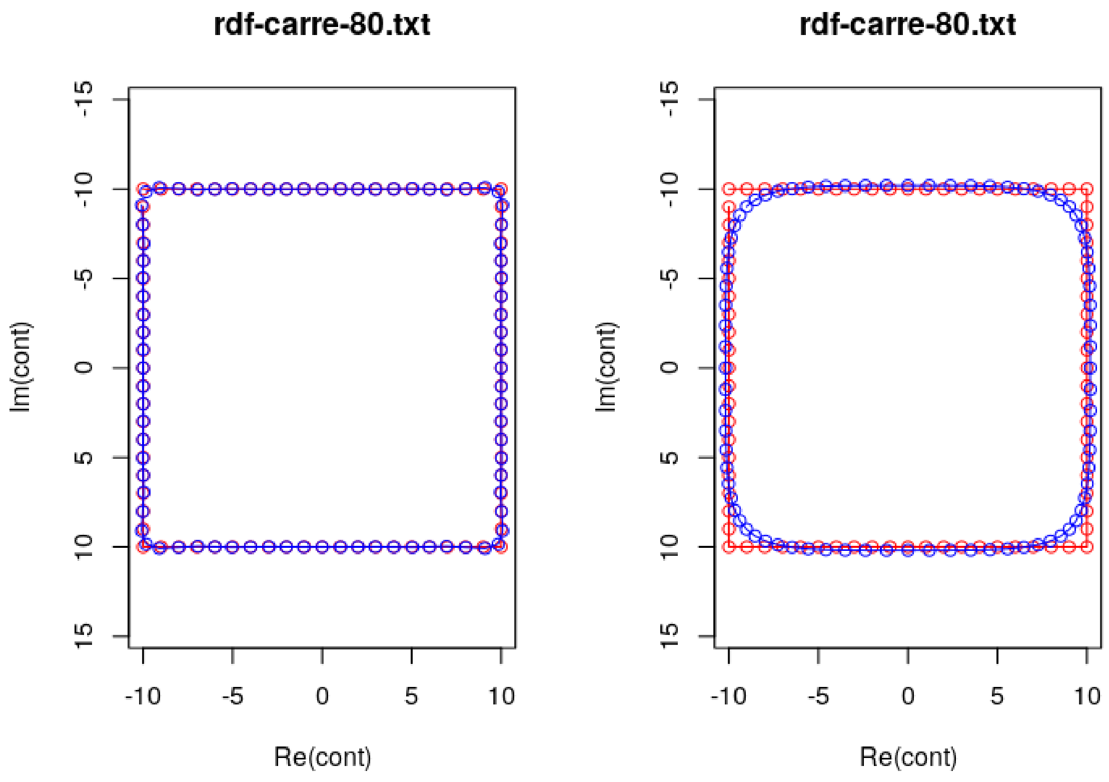


FIGURE 2.5
Simplified contours, ratio from left to right: 0.5, 0.075

2.4 Simplifying the shape contour

Chapter 3

Conclusion

We have managed to use the *moments of a shape* in order to define *shape attributes*. Having these, we now know information such as the *surface* of a shape, the *barycenter* of the shape, what its *main axis of inertia* is and how much “inertia” exists on each of these axis, thanks to the *eigen values* calculated from I, the *inertia matrix*. Being able to make these invariant to the translation, rotation and scale of a shape is also a big advantage and should serve us well in the future.

Bibliography

Wikipedia (2020a). *Complex number*. Wikimedia Foundation. URL: https://en.wikipedia.org/wiki/Complex_number (visited on 4th Feb. 2020).

– (2020b). *Fast Fourier Transform*. Wikimedia Foundation. URL: https://en.wikipedia.org/wiki/Fast_Fourier_transform (visited on 4th Feb. 2020).

Cabestaing, François (2020). *Reconnaissance des formes*. Université de Lille. URL: http://master-ivi.univ-lille1.fr/Cours/RdF (visited on 4th Feb. 2020).