Icon Retrieval System

Implementing a Shape Descriptor feature in LIRE

Manuel Scurti
Politecnico di Torino
manuel.scurti.dev@gmail.com

Jeanne Bosc Bierne
ISAE-SUPAERO
jeanne.boscbierne@homtail.fr

March 19, 2019

***K*eywords** CBIR · Image Processing · Information Retrieval

1 Introduction

Designing a graphical UX requires the knowledge of matching combinations of colors, shapes and textures in order to create a nice looking interface for the user experience. Very often happens that designers spend most of the time getting inspired by templates or example UXs to create their own. In this work, we will try to help designers in the process of searching icons with similiar style of the input icon. This is potentially useful to combine icons from different icon sets. As a solution, we propose a content based image retrieval for icons, that can analyze the input icon style that the designer is searching for and retrieve all icons with similiar characteristics. To address this problem we will implement an extension of LIRe - Lucene Image Retrieval.

2 LIRe Implementation

2.1 Why LIRe?

LIRE is a Java library that provides a simple way to retrieve images based on color and texture characteristics. The main reason to use this library is that is already oriented to retrieve images, thus it is supporting a lot of state-of-the-art feature extraction techniques. Furthermore, it is also open source, so that it can be easily extended to our needs.

2.1.1 Goal

Our goal is to create a feature extractor able to classify, with the highest possible accuracy, the shape of the outermost contour contained in the icon, in three classes: rectangle, circle, undefined. Let's see some examples:

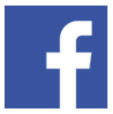
Figure 1: Square shape

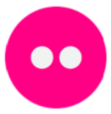
Figure 2: Circle shape

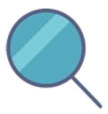
Figure 3: Other shapes

Most of the icons available out there for designers belongs to these 3 categories. By enabling the LIRE indexer to rank icons by its shape, we expect to deliver better results for the icon style research process. Other features used by the indexer together with the ShapeDetector are:

- Color and Edge Directivity Descriptor (CEDD)
- Fuzzy Color and Texture Histogram (FCTH)

These two features were chosen because they perform good, combined with our feature extractor, for this kind of task, because they're both focusing on textures and colors at different granularity.

2.2 Implementing a shape descriptor feature in LIRe

2.3 LIRe Architecture

LIRe works by creating a Lucene index of a set of images given as input. Once the index is created, it gives the possibility to search through it by using already implemented algorithms[1]. All the available features extractors are implementations of a main interface called LireFeature. In order to create a custom feature extractor we just need to extend one of the two interfaces. In our case, ShapeDetector is an extension of the GlobalFeature interface.

2.4 IT Design

The architecture of the icon retrieval system is showed in the figure below. When indexing, LIRe calls extract() method

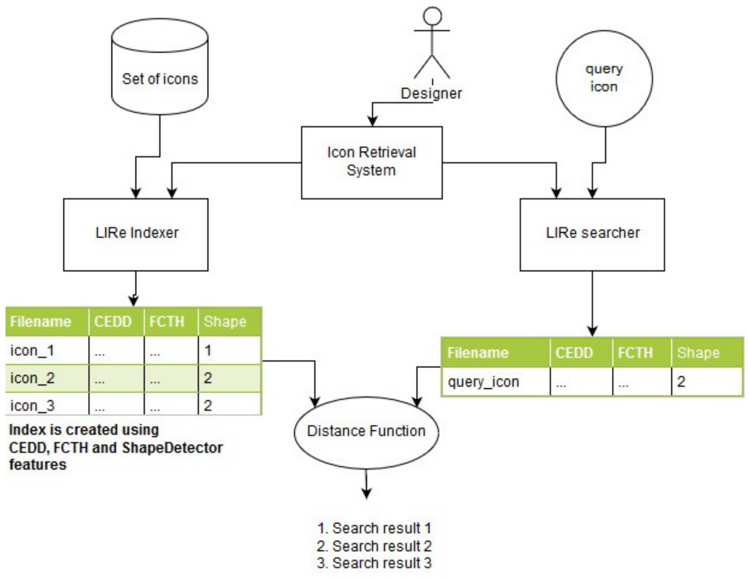

Figure 4: IT design

inside our class, for each image, and save the result into the Lucene index file. When searching, LIRe calls getDistance() method inside our class to compute the metric distance to compare two search results and rank them. Our feature vector, i.e. the result of the extract() method, contains only one value: 0 if it's an undefined shape, 1 if it's a rectangle, 2 if it's a circle. The getDistance() is a discrete function that will give back 0 if two objects have the same shape, 100 otherwise.

2.4.1 Feature Extraction Process

The ShapeDetector custom feature works by using low level feature extractions APIs provided by the OpenCV library[2]. The workflow of the process follows these steps:

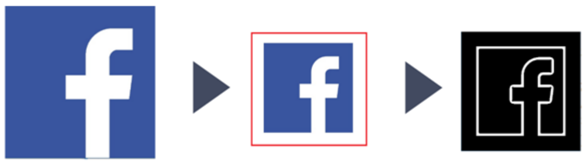

Figure 5: Shape Detector Process

- A preprocessing phase is applied to the original icon (as showed in the figure above). This includes (in order):
  - Rescaling and border extension
  - Gray scale transformation
  - Gaussian Blur filtering

- Thresholding
- Rectangles or circles are searched into the icon
  - Circles are searched by Hough Circles technique [3]
  - Rectangles are searched by analysing contours and vertices[4][5]
  - The biggest area between the circle and rectangle that is also similar, in terms of size, to the area of the icon surface is classified

2.5 Performance

To test the accuracy of the shape classifier we used a dataset consisting of 80 icons, downloaded from FlatIcon[6], of which:

- 25 circle icons
- 30 squared icons
- 25 undefined shape icons

Table 1: ShapeDetector Performance

Accuracy	Recall	Precision
58%	0.58	1.0

2.5.1 Results and future improvements

Our strong assumption in our work is that if two icons have different shape then they have different style. This can sound strange to you, but if two icons are not both rectangle, or both circles, or both irregulars then it is most likely the case in which the styles are different and then it is not what the designer is looking for. This strong assumption is the main reason for us to create a new feature in LIRE that strongly sets this constraint, even if there are already some shape descriptor implemented. Even though there is still plenty of work that can be done here, we think that this a good starting point to future improvements. Some possible improvements are: a better performing shape classifier, adding more features extractors (e.g. thickness descriptor), speeding up by parallelizing search and indexing processes and so on.

References

[1] LIRe Documentation - `http://www.semanticmetadata.net/wiki/`.

[2] OpenCV Documentation - `https://docs.opencv.org`.

[3] Circle Detection with OpenCV - `https://docs.opencv.org/trunk/d4/d70/tutorial_hough_circle.html`.

[4] Generic Shape Detection with OpenCV - `http://answers.opencv.org/question/176614/detecting-shapes-using-opencv-with-java/`.

[5] Examples of OpenCV using Java - `https://www.programcreek.com/java-api-examples/?class=org.opencv.imgproc.Imgproc&method=approxPolyDP`.

[6] FlatIcon - `https://www.flaticon.com/packs`.