ICON RETRIEVAL SYSTEM

MANUEL SCURTI
JEANNE BOSC-BIERNE

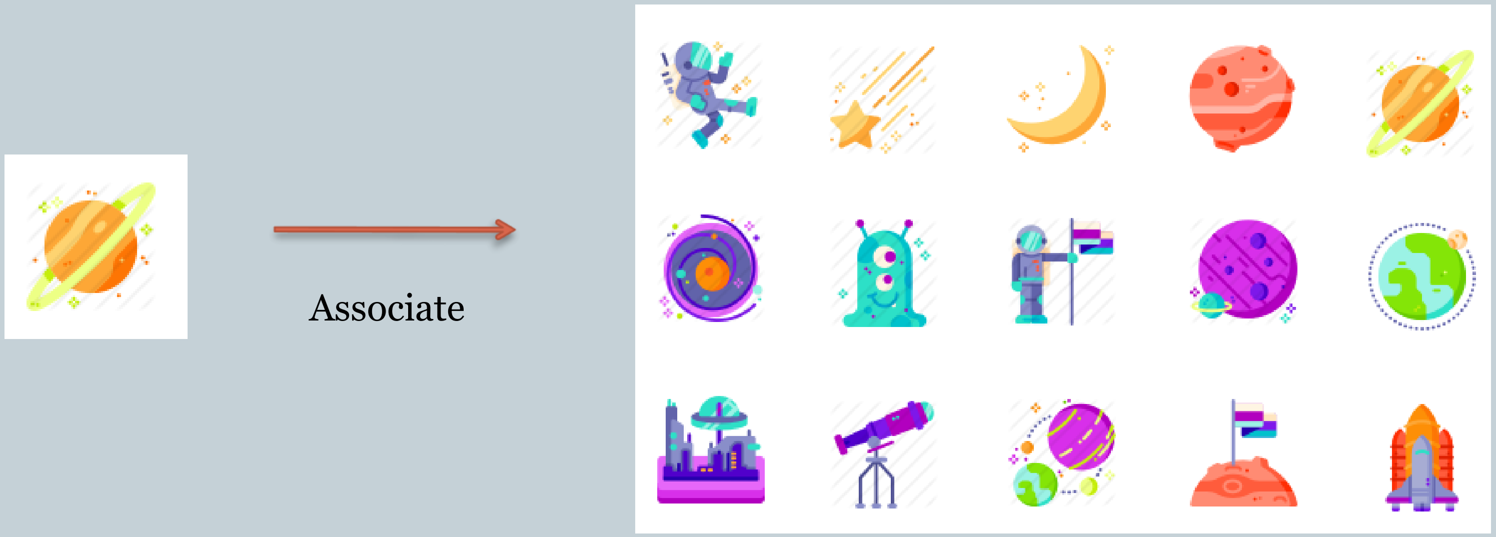
19 MARCH 2019

Plan

- Introduction
  - Icons characteristics
- LIRe implementation
  - Why LIRe?
  - Implementing a shape descriptor feature in LIRe
    - Goal
    - Architecture
    - Process
  - Performances
    - Results and future improvement

Motivation

- Designers : get inspired by icon's style and want to access to all the similary icons from this same style from a dataset

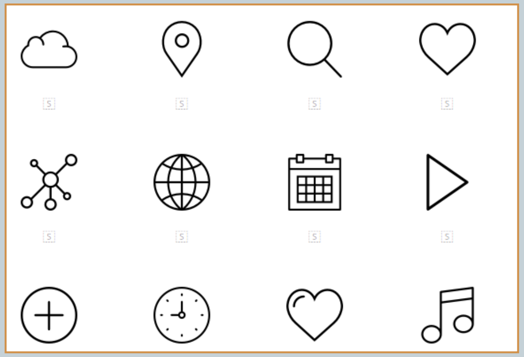


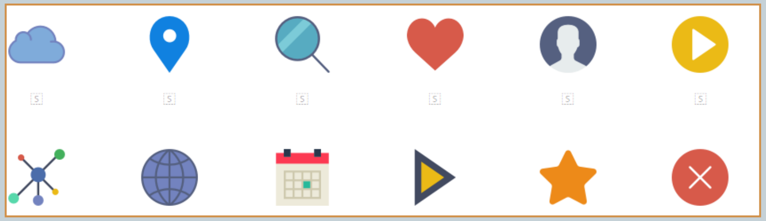
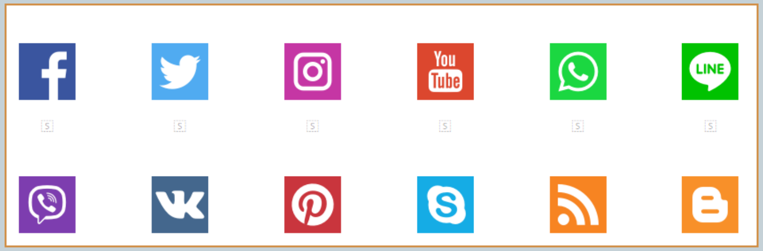
Efficency of icons

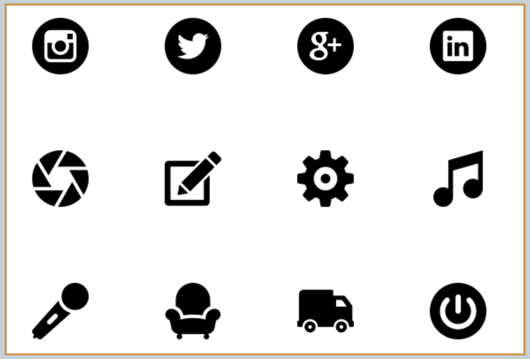
- Icons issues:
  - Easy to identify
  - Really fast
  - References by cathegories of icons of a same style

- How to recognize a *style* of icons?
- Which criteria?
- How to implement it on LIRe?

Variety of icons





Characterization of an icon style

- Colors
  - Number of colors
    - Black/white
    - Grey
    - Colored
      - 1-2 colors
      - Large numer of color
  - Background
    - Transparency?

- Lines
  - Thickness
  - Angles radius

- Shape
  - Size
  - Shape of outline
  - Resolution
  - Degree of detail

Why LIRe?

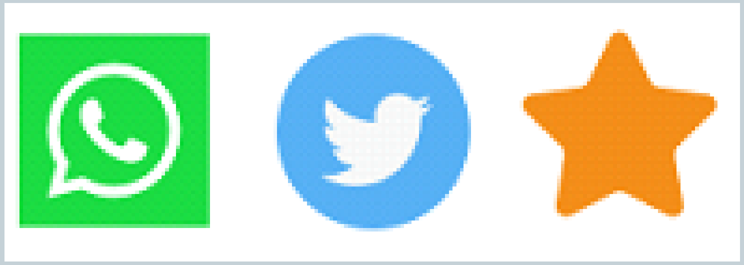
- LIRE is a Java library that provides a simple way to retrieve images **based on color and texture** characteristics.

- The main reason to use this library is that is already oriented to retrieve images.

- Furthermore, it is also **open source**, so that it can be easily extended to our needs

Goal

- Implementing a **shape descriptor feature** in LIRe
  - To create a feature extractor able to classify the shape of the outermost
- Contour contained in the icon, in three classes:



Rectangle Circle Unidefined

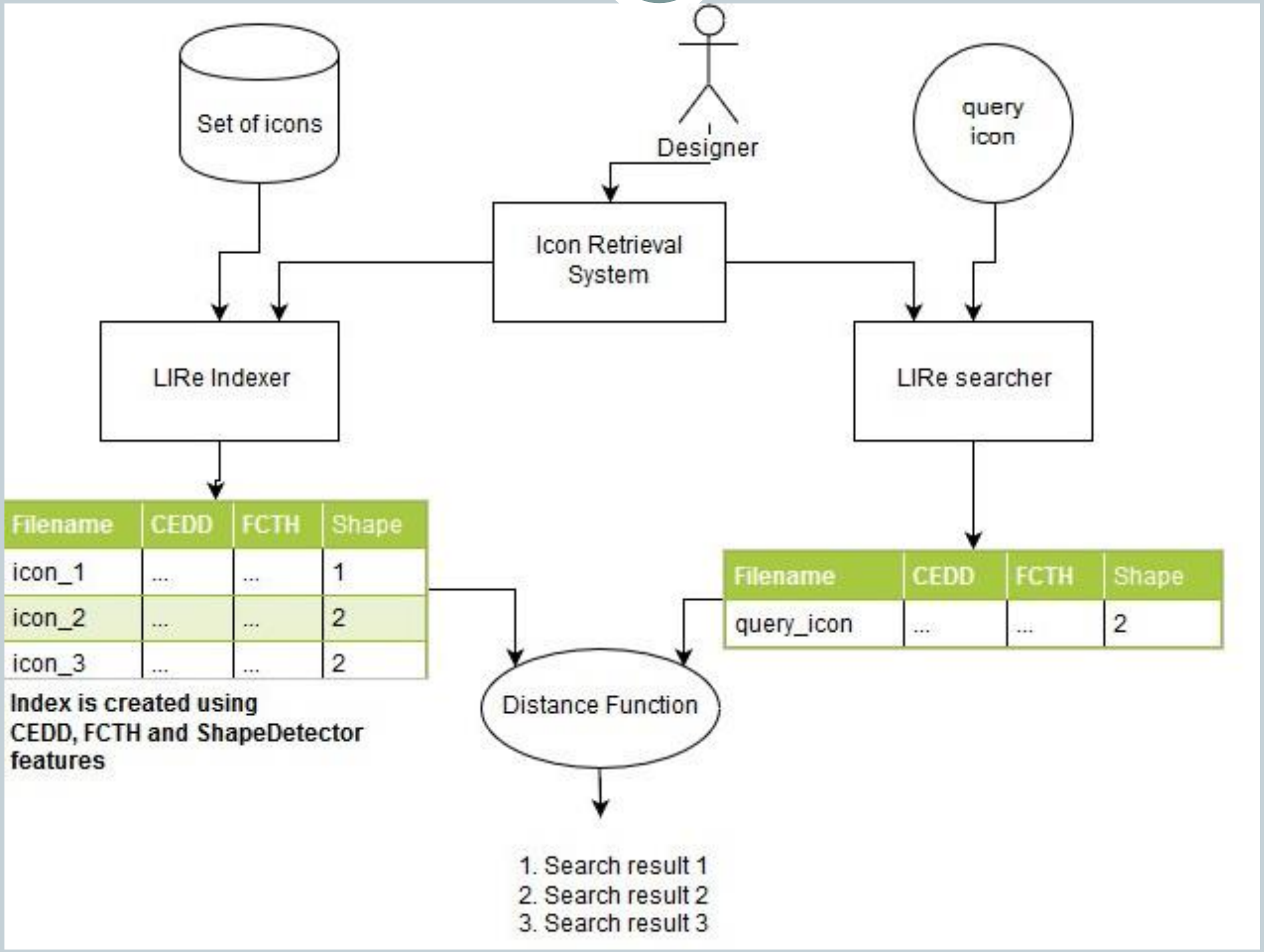
LIRe Architecture

- 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.

- 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.

IT Design



Filename	CEDD	FCTH	Shape
icon_1	...	...	1
icon_2	...	...	2
icon_3	...	...	2

Index is created using CEDD, FCTH and ShapeDetector features

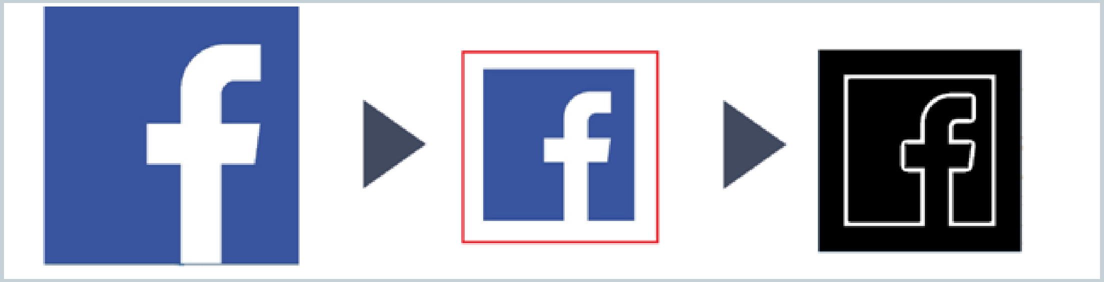
Filename	CEDD	FCTH	Shape
query_icon	...	...	2

1. Search result 1
2. Search result 2
3. Search result 3

IT Design

Feature extraction process

- Preprocessing phase applied to the original icon
  - Rescaling and border extension
  - Gray scale transformation
  - Gaussian Blur filtering
  - Thresholding
- Rectangles or circles are searched into the icon
  - Circles : by Hough Circles technique
  - Rectangles : by analysing contours and vertices
  - 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



Performances

- Dataset of 80 icons :
  - 25 circle icons
  - 30 squared icons
  - 25 undefined shape icons

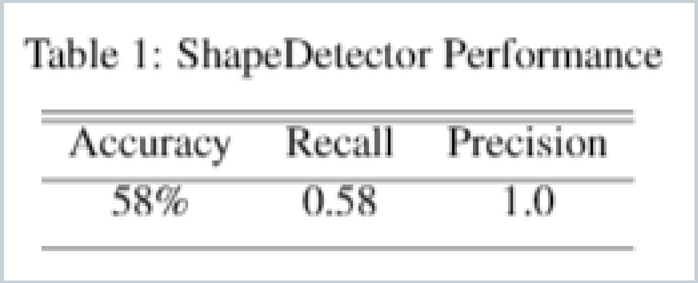


Table 1: ShapeDetector Performance

Accuracy	Recall	Precision
58%	0.58	1.0

Results and future improvements

Assumption : if two icons have different shape then they have different style.

⇨ 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.

⇨ Good way to start, can we follow by many improvements :

- Better accuracy by improving shape classification technique
- Speeding up by parallelizing search and indexing processes
- Adding more features extractors
  - (e.g. thickness descriptor),
  - Shadows recognition
  - Radius angles

References

- https://pdfs.semanticscholar.org/71d0/30d6a88b251ccb1a24f3ca058b151d5e2a3b.pdf
- https://tubikstudio.com/visual-perception-icons-vs-copy-in-ui/- Article on icons recognition
- https://www.nngroup.com/articles/classifying-icons/- Article on icons classification
- https://www.flaticon.com/packs - Website with free icons set
- https://docs.opencv.org - OpenCV documentation
- https://docs.opencv.org/trunk/d4/d70/tutorial_hough_circle.html - Circle detection
- http://answers.opencv.org/question/176614/detecting-shapes-using-opencv-with-java/ - Generic shape detection
- https://www.pyimagesearch.com/2016/02/08/opencv-shape-detection/ - Tutorial for shape detection in Python
- https://www.programcreek.com/java-api-examples/?class=org.opencv.imgproc.Imgproc&method=approxPolyDP - Examples of OpenCV with Java
- http://www.semanticmetadata.net/wiki/ - LIRe documentation
- https://github.com/dermotte/LIRE - GitHub repository of LIRe source code