M1 Project - Image

Simon Mauras

March 29, 2016

Contents

---

1 Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobor- tis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris.

2 Implementation

2.1 Feature extraction

The feature extraction is done in C++ using the library DGtal. The file `features.cpp` provides several functions to extract a vector of features from an image. The program `shape_indexing` takes as input an image and write a vector of values in $\mathbb{R}^d$ on the standard output. The program `shape_distance` computes the similarity between the vectors of features of two images and write it on the standard outut. The python script `classifier_corpus.py` builds a vector of feature for each image of the directory `database` and store it in `corpus.csv`.

2.2 Learning

Once we have extracted a vector of values for each image of the database, we train a learning model on the corpus. The script `classifier_learning.py` reads the vectors of features, train a learning model and export it in the file `model.dump`.
We choose to use a k-nearest neighbors classifier (scikit-learn). To have an estimation of the score with the whole database, we use cross-validation.

2.2.1 k-nearest neighbors classifier

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobor- tis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris.

2.2.2 Cross-validation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobor- tis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris.

3 Normalization

During the classification, we need to analyse some noisy images. A preprocessing is therefore needed. Our preprocessing is done in 3 steps.

- Scale the image
- Smooth with a median filter
- Add a small (1px thick) black border to handle shapes that touch the border of the image (making sure that the background is connex).
- Compute the contour of the shape.
- Fill-in the shape.

We normalize images using function `normalize` (in `features.cpp`). The scaling is important because even if our features are scale-invariant, the digitization is a huge problem. Indeed after a few experiments, we realize that scaling then smoothing is better than just using the initial image. The result of normalization on a very noisy shape is satisfactory.

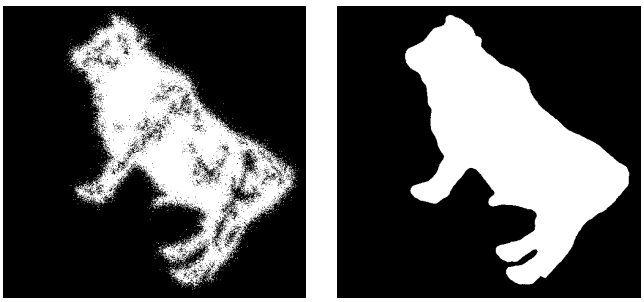


Figure 1: Normalization of a noisy shape

If there are several connex components, the biggest one is choose to be the shape. The filling part of the normalization is very usefull for classes like butterfly, cattle or dog as some images contains a lot of gaps and other don't.

Figure 2: Filling the shape

4 Feature design

4.1 Distance Transformation Percentile

Let's take a look at the distance transformation histogram for several shapes.

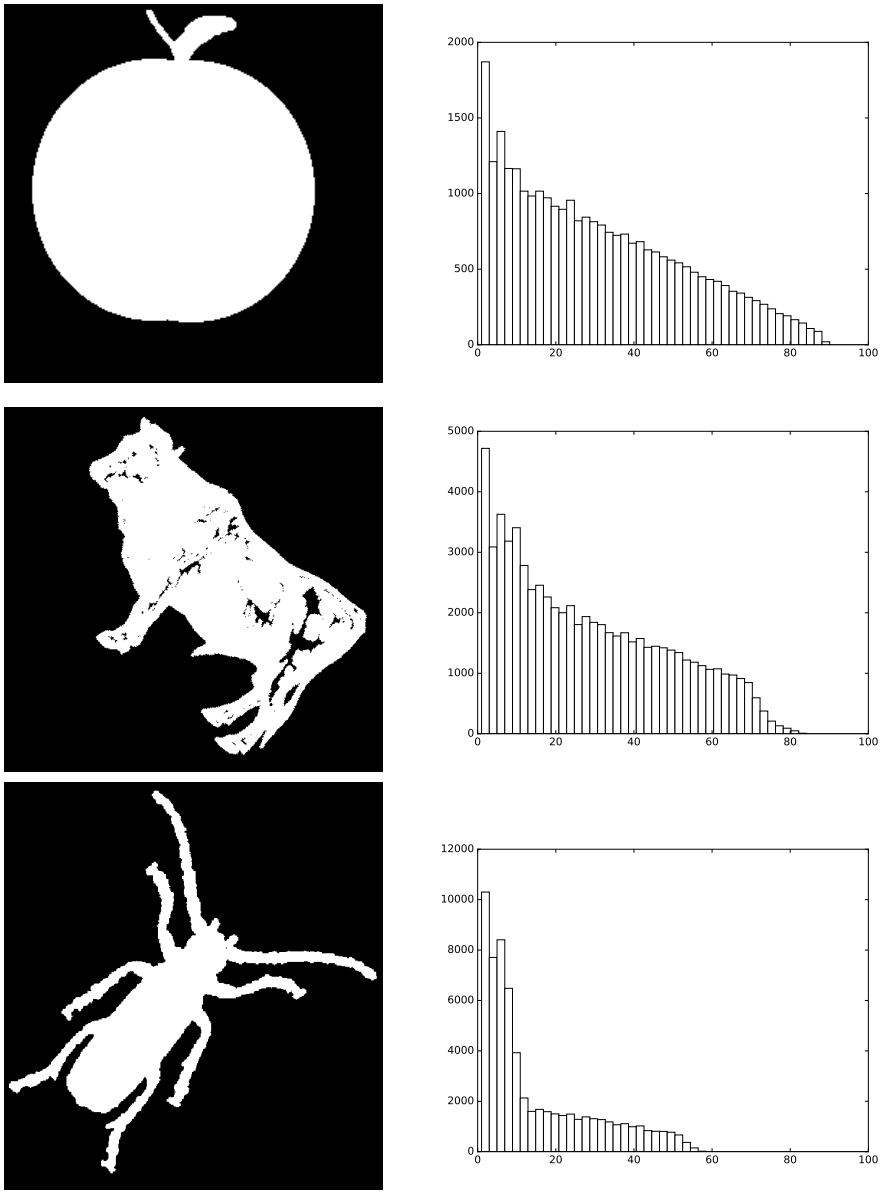


Figure 3: Distance Transformation Histogram

We can observe that a shape can be characterized by the distributon of its distance transformation.

Now we need to build an estimator satisfying the following requirement:

- Must be invariant under translation, scaling and rotation.
- We need some kind of continuity. If there is a small modification of the shape then the new estimation has to be close to the initial one.

We already have invariace by translation and rotation. In order to have the invariance under scaling, we can normalize the distance and the frequencies.

However we can notice that it is not enough to have a good estimator. Indeed the histograms $(1, 0, \ldots, 0)$ and $(0, 1, 0, \ldots, 0)$ are very close but the distance between those two vectors is huge. To fix this problem we are going to compute percentiles. The function `compute_dt` (in `features.cpp`) compute those features.

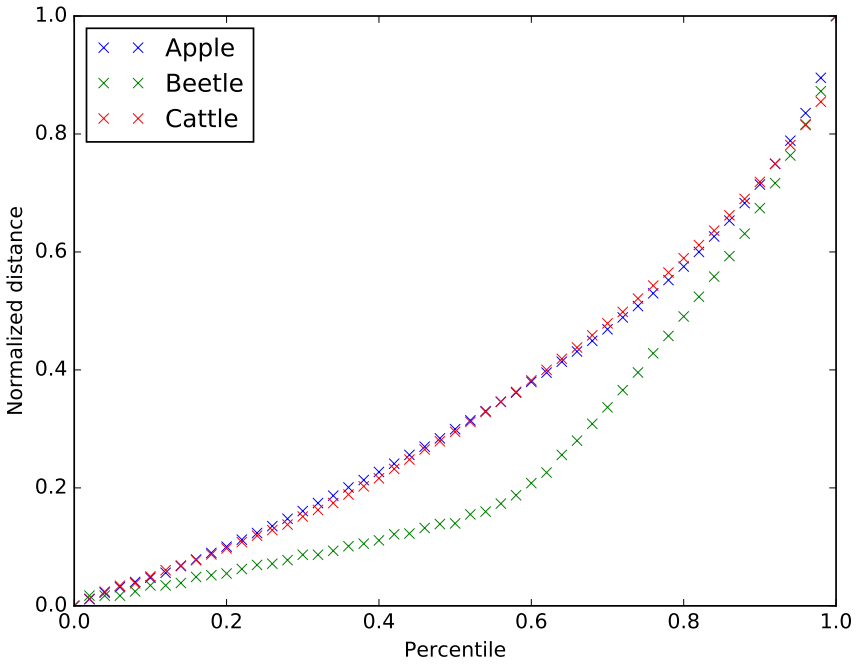


Figure 4: Distance Transformation Percentile

We just build an estimator that can help us to distinguish an apple from a beetle. But we can notice that this estimator is not very good to decide whether a shape is a cattle of an apple.

4.2 Rotation Distance

One caracteristic which is still not used is point reflexion. There is a lot of images with symetry and rotation invariance. Our idea is to compare the initial image with its rotation of θ degrees around its centroid. We can prove that if we can find a invariance under rotation for our image, then the centroid of the shape will be the center of the rotation.

For $\theta = 2\pi/k$ for $k = 2 \dots 10$ we chose to compute our features as follow:

- Compute the centroid c of the shape
- For each point p of the shape
  - Compute its rotation $q = c + e^{i\theta}(p - c)$
  - If there is a point in the shape at distance to q less than r, then add 1 to the score.
- Normalize the score (divide by the number of pixels in the shape)

The function `similarity_rotation` (in `features.cpp`) implement the algorithm we just describe.

The figure 5 contains the value of those features for five different classes. When k goes from 2 to 10, the angle goes from π to $\pi/10$. We can recognize the $\pi/6$ rotation invariance of device 1 and the circular shape of an apple. The results are very relevant for regular shapes (circles, squares, stars, ...). With distance transformation we had trouble to distiguish an apple from a cattle, now we can separate those two classes.

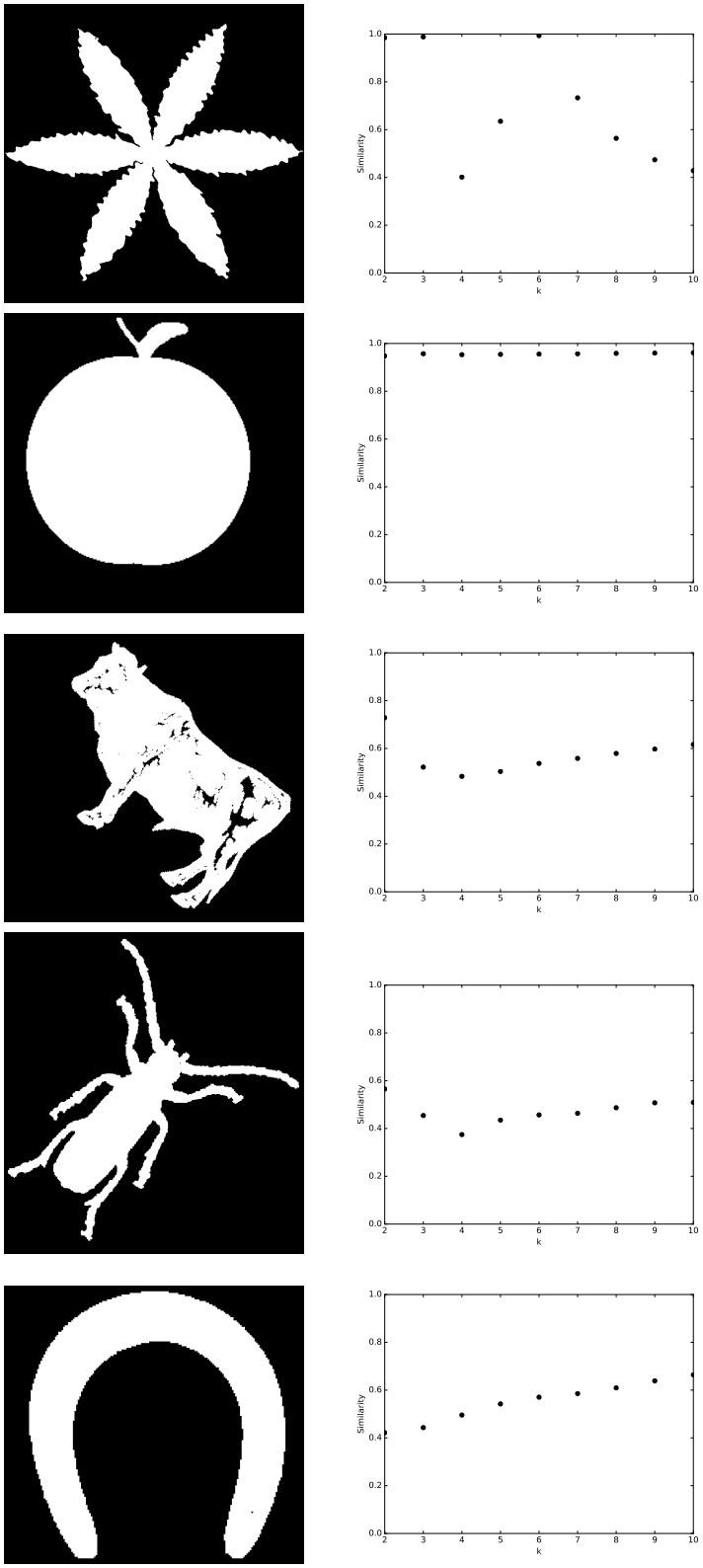


Figure 5: Rotation Distance

4.3 Convex hull

The convex hull of a set of point X is the smallest convex set that contains X. There exists several algorithms to compute the convex hull of a 2D finite set (Graham's scan, ...) We implemented the function `convex_hull` (in `features.cpp`) which computes the convex hull of a 2D shape in an image.

In the function `convex_hull` (in `features.cpp`) we implemented the monotone chain algorithm (aka Andrew's algorithm) as our points are aready sorted (2D matrix of 0/1). We first compute the upper hull and the lower hull. Then we simultaneously go through the domain, the upper hull and the lower hull to decide whether each point of the image is in the convex hull. The overall complexity is in $\mathcal{O}(n)$ where $n = a \times b$ is the size of the image.

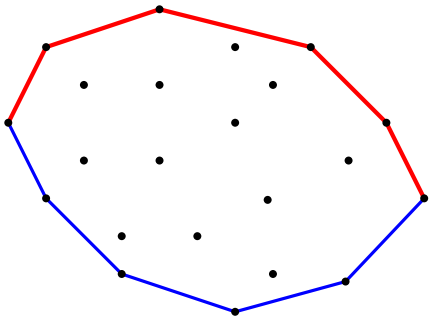
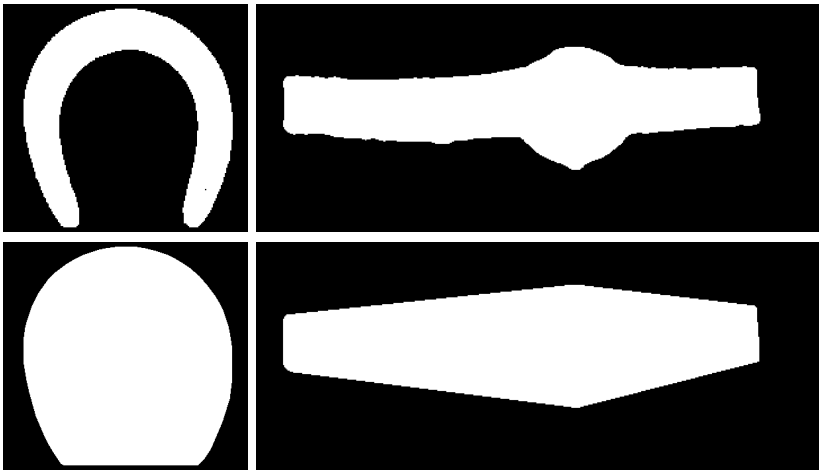


Figure 6: Upper and lower hull

One feature that give very good results is the ratio between the area of the initial image and the area of the convex hull of the shape. It is straighforward to show that this estimator is invariant under translation and rotation. We can also notice that the two area are multiplied by the same factor when scaling an image.



Ratio $= 0,46$ Ratio $= 0.79$

Figure 7: Convex Hull Area

We also duplicated all the features we designed previously and used them on the convex hull of the shape.

5 Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobor- tis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris.