University "POLITEHNICA" of Bucharest

Automatic Control and Computers Faculty,
Computer Science and Engineering Department

BACHELOR THESIS

A scalable algorithm for similar image detection

Scientific Advisers:
Prof. Dr. Ing. Nicolae Ţăpuş

Author:
Andrei-Bogdan Pârvu

Abstract

Here we can write the abstract of this extraordinary paper

Contents

Chapter 1

Design

1.1 Analysis of an image

1.1.1 Using Harris corner detector

The analysis of an image begins with applying the Harris detector on it. It will compute a given score for each pixel of the image, the higher the score, the greater the chance that pixel represents a corner.
Using these values we will have to determine a subset of pixels that will represent the Harris mask of the image. Experimentally, I have established that all the points which have a value greater than $0.01 * max\_image\_value$ are to be part of the subset.
As an example, Figure 1.1 shows a normal image of a woman's face, while Figure 1.2 shows the corresponding pixels that form the corner mask.

Figure 1.1: Sample image

Figure 1.2: Harris corner mask applied

As it can be seen from the image, the corner mask centers around the interest points in the image, the eyes and the eyelashes, while leaving the smooth surfaces (the skin) unmarked.

Another observation is that possible watermarks can be present in the image (as seen above), which, of course, the mask detects (they are center pieces of the image, and the corner algorithm cannot determine that they have no real connection with the image).

Furthermore, in some cases the watermark might contain all (or at least a vast majority of) the points in the mask (as seen in Figure 1.3). To avoid such a behavior, I have split the image into 9 sub images (three rows and three columns of identical dimensions) and the Harris mask is applied to each of these.

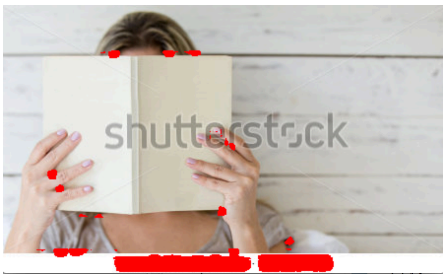

Figure 1.3: Single image

Figure 1.4: 9 subimages

This determines the corner mask to also contain pixels from the center of the image (some of which, unfortunately, are also a watermark).

1.1.2 Using SIFT keypoints and descriptors

As we have seen, the SIFT algorithm computes the keypoints of an image, and then the descriptors of these keypoints. Due to the high similarity nature of our problem, we do not want to compute the keypoints for the entire image, but filter them based on the corner mask determined in the previous section (as observed in Figure 1.5 and Figure 1.6)

Figure 1.5: SIFT keypoints for the entire image

Figure 1.6: SIFT keypoints for corner mask

The SIFT descriptors are then computed for each keypoint located in the corner mask and this will be the information stored for a certain image.

1.2 Analysis of a pair of images

In order to analyze a pair of images, we shall use the SIFT descriptors determined in the previous section. The two sets of descriptors are compared in order to obtain the best matches between pairs of keypoints. A distance is computed between each pair of keypoint descriptors, which is the Euclidian norm between the SIFT descriptors of the keypoints. Experimentally, I

have concluded that a match between two keypoints has a high similarity if the Euclidian norm is less than 100.
For a pair of images, we first find the set of corner-mask keypoints and then compute the best matches between these keypoints. We shall keep only the best 10 matches, and compute the arithmetic mean between the distances of these matches. As stated before, if this mean is smaller than 100, the two images are considered similar. We shall name this algorithm the *pair similarity algorithm*.
Figure 1.7 shows the corresponding matches between two images with two different watermarks, one of which is rotated 90^o clockwise.

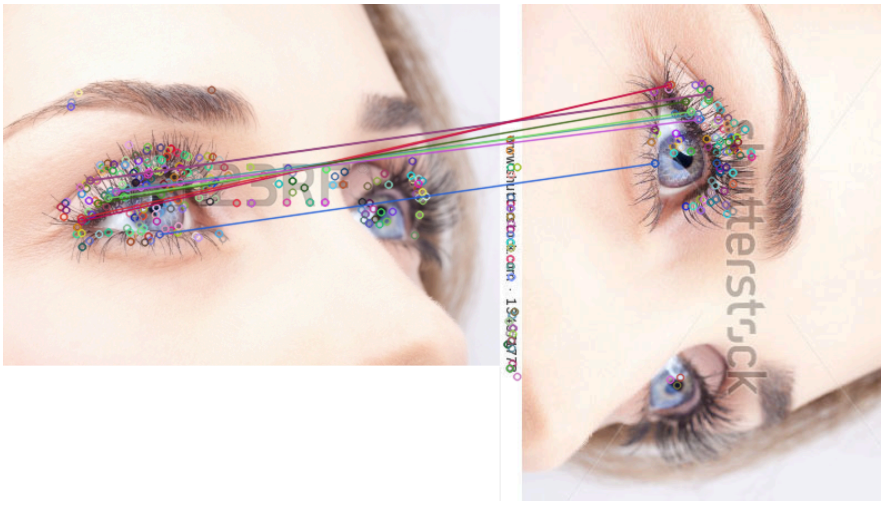

Figure 1.7: Comparison between two images

1.3 Analysis of a set of images

Suppose that we have a set of images, and we want to compare a test image with the set and detect whether we have a similar image within the set. Of course, the first possibility is the brut force one: we iterate through all the images and apply the *pair similarity algorithm* described in the previous section. Although this provides a correct result, it has a complexity of $O(number\_of\_images * image\_match\_time)$. We shall name this basic algorithm as the *linear algorithm*. Although this algorithm is very straightforward, its complexity is undesirable if the number of images becomes large. Moreover, most of the images in our set will likely have a big similarity distance with our searched image, so maybe we don't want to apply the full *pair similarity algorithm*.
Thus, we need to determine an efficient algorithm which can filter the initial set of images to a smaller set which contains very likely matching candidates with our test image.
The filtering algorithm is implemented as follows: we will maintain a maximum number of M descriptors for each image in the initial set and create a KD-tree with the set of descriptors of the initial images. When a query for a test image arrives, we will compute its descriptors and then perform a N nearest-neighbor search on our KD-tree. Then we will select the top T images with the most descriptors returned by the KD-tree and perform the *linear algorithm*. Experimentally, I have determined that the optimal values for M, the number of descriptors, N, the number of neighbors and T, the number of images is 40, 5 and 10.