Image Matching Algorithm Based on Hashes Extraction

Alberto Rivas[1], Pablo Chamoso[1], Javier J. Martín-Limorti[1], Sara Rodríguez[1(✉)], Fernando de la Prieta[1], and Javier Bajo[2]

[1] BISITE Research Group, Edificio I+D+i, University of Salamanca, Calle Espejo 2, 37007 Salamanca, Spain
{rivis,chamoso,limorti,srg,fer}@usal.es

[2] Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Campus Montegancedo, 28660 Boadilla del Monte, Madrid, Spain
jbajo@fi.upm.es

Abstract. Nowadays, the rise of social networks and the continuous storage of large of information are topical issue. But the main problem is not the storage itself, is the ability to process most of this information, so that it is not stored in vain. In this way, using the shared images within the scope of social networks, possible relationships between users could be identified. From this idea arises the present work, which focuses on identifying similar images even if they have been modified (applying color filters, rotations or even watermarks). The solution involves preprocessing to eliminate possible filters and then apply hashing techniques, just to obtain hashes that are unique for each image and allow the comparison of an abstract but effective way for the user.

Keywords: Image matching · Visual analysis · Social networks

1 Introduction

This article is focused on the analysis of images published in social networks. It has been developed to be integrated in a social network focused on work environments and job searches. The intention of this social network is to connect users with the same interests based on the contents they share. These contents can be published in different ways, for example images (which may or may not be accompanied by a descriptive text, so that text will not be taken into account).

Therefore, the problem that this article faces is the identification of images which are the same from a human point of view, but not from a computational point of view for one or more reasons: (a) the quality has been reduced or the image format has been changed; (b) a watermark has been included; (c) some changes in tonality have been applied; (d) a border has been added or removed; (e) the image has been rotated.

The solution presented in this article tries to solve this problem with an algorithm based on obtaining hashes from the images, so that the system is able to quickly compare the existing images and the new image at the moment it is sent by the user.

The rest of the article describes existing methodologies used in image matching and the mechanisms involved in processing large amounts of information in real time. Next, the proposed algorithm for image processing and matching is described, as well as the platform that supports real-time processing. This system is evaluated in the results section with a set of images. Finally, the article present the conclusions drawn and the lines of future work.

2 Background

As stated, the main focus of this article is to identify images that are the same from a human point of view but differ computationally, in order to put people who publish similar images in contact, as they may share common interests.

When identifying whether two images are the same, it is necessary to perform a series of checks because two apparently identical images may be computationally different due to problems of compression, different quality of the image, number of colors, size, slight modification of the image with filters or watermarks, changes in the tonality, insertion of borders, or rotations.

In the computer vision and the image processing fields, different methodologies have been presented to extract relevant information with an image as input. These techniques are catalogued under the concept of feature detection.

There are different types of image features including edges, corners or interest points, and blobs or region of interest. In this regard, there are multiple algorithms used to process images in search of features, the most common of which are:

– **Edges**: Canny, Sobel, Harris & Stephens, SUSAN [12].
– **Corners**: Harris & Stephens, SUSAN, Shi & Tomasi, Level curve curvature, FAST, Laplacian of Gaussian, Difference of Gaussians, Determinant of Hessian.
– **Blobs**: FAST, Laplacian of Gaussian, Difference of Gaussians, Determinant of Hessian, Maximally stable extremal regions (MSER) [8], Principal curvature-based region detector (PCBR) [3], Gray-level blobs and others algorithms [2,5].

The concept of perceptual hashing is similar to that of the classical paradigm of cryptographic hashes, where the tiniest changes quickly evolve into an entirely different hash. In perceptual hashing the image content is used to try to fingerprint the image, so that even if hashes are not identical they can be used to determine how "close" the images are to one and other.

Another important concept that has been applied when comparing different images is the Hamming distance [9]. It can be used on most of the resulting hashes to determine the perceived difference between two images, so that a perceptually similar image would have a short hamming distance, 0, for the same image. A quick definition for hamming distance, $d(x, y)$, is the number of ways in which x and y differ. In other words, the hamming distance is simply the number of positions in which they are different.

There are different proposed algorithms based on the hash value generation technique: pHash (also called "Perceptive Hash", with different variations) [6], aHash (also called Average Hash or Mean Hash) and dHash Also called Difference Hash) [7]. The typical hash-based algorithms flow diagram is shown in Fig. 1.

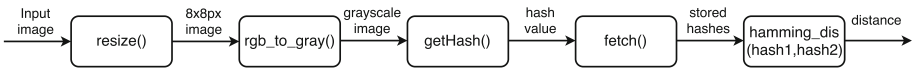

Fig. 1. Typical hash-based algorithms flow diagram.

However, all variations of this methodology present different problems when dealing with an image to which a border has been added, or one which has been rotated.

For the latter problem, a modification of these steps is proposed in [1,4] by introducing a rotational system whereby it is possible to differentiate images rotated 22.5°; however, this implies a loss of precision in the corners when the original images are rectangular or square, the most common situations with images uploaded to social networks, so its solution is not applicable to the present problem.

There are online platforms dedicated to the search of images that exist on the Internet and are similar to those provided by a user, without taking into account the meta-data or associated text. Their applicability is oriented to the search of image plagiarism. This is the case of TinEye [13], whose algorithm is not public, but is based on the analysis of hashes.

3 Proposed System

The proposed methodology is based on the application of techniques of image matching based on hash value generation, with certain transformations and preprocessing that are able to discriminate the possible transformations that a social network user may have applied to the image prior to uploading it.

One of the main characteristics of the proposed system, which makes it possible to improve the result of similar systems, is the preprocessing stage. This stage is focused on applying a series of transformations to the images that are received as input by the user. This is followed by a scheme similar to hash-based algorithms.

3.1 Preprocessing

The strategy followed in this first stage ensures that images which have been slightly transformed are stored in the system in the same way. This allows the system to start comparing the same or most similar image.

The present study considered the following possible transformations that a user could perform on an image, after which the image would still be considered the same: (i) insertion of an outer uniform border; (ii) rotation of the image; and (iii) insertion of a watermark. It should be noted that all hash-based algorithms are really robust if a uniform change is applied to the tonality. Therefore, such modifications were not considered for the comparison.

When a watermark is inserted, a hash-based algorithm application can be sufficient to determine if it is the same image or not despite the modification. Therefore, in this first stage of preprocessing, the proposed system focuses only on any modifications based on the insertion of an outer uniform border, and the rotation of the image.

- **Solid border addition:** The proposed system applies the Algorithm 1, allowing the following steps of the methodology to be performed without considering the uniform outer border. The first step is to transform the original image provided by the social network user I to a grayscale image gI, which will also be used in the following steps.

Algorithm 1. Solid border removal algorithm

```
1:  function borderRemoval($I$)
2:      $gI$ = grayscale($I$)
3:      if hasBorder($gI$) then                              ▷ Check border
4:          $value$ = getBorderTonality($gI$)                ▷ Get border tonality value
5:          $bI$ = toBinary($gI$, $value$)                   ▷ Border tonality as threshold
6:          $cnt$ = findContour($bI$)                        ▷ Get contour
7:          ⟨ x,y,width,height ⟩ = boundingRect($cnt$)       ▷ Find bounding rectangle
8:          $gI = gI[x : x + width, y : y + height]$         ▷ Crop grayscale image
9:      end if
10:     return $gI$
11: end function
```

- **Image rotation:** The most common rotations that a user applies to an image are based on 90° modifications. This part of the preprocessing is centered on precisely this type of rotation. The objective is for the images to follow a rotation pattern so that they always have the same orientation in the system. Different solutions are possible, depending on whether the shape of the image is rectangular or square.
  If the image is rectangular, the system will always work with the image in landscape mode (the two longest sides are in the x-axis) The system must then determine which side is placed on the top and which is placed on the bottom. If the image is square, the previous logic cannot be applied, since the four sides are the same length. In both cases, the key of the final orientation will be the tonality of the image, as described by the Algorithm 2. Although this step appears in the preprocessing section, it is applied in an intermediate step of the Algorithm 3, which will be detailed below, to avoid possible changes in the tonality resulting from the insertion of a watermark.

Algorithm 2. Image rotation algorithm

```
1: function ImageRotation($gI$,$sI$)
2:     $width$ = getWidth($gI$)
3:     $height$ = getHeight($gI$)
4:     if $width == height$ then                                  ▷ Square image
5:         $nsI = sI[0 : width, 0 : height/2]$                    ▷ Get North middle
6:         $ssI = sI[0 : width, height/2 : height]$               ▷ Get South middle
7:         $wsI = sI[0 : width/2, 0 : height]$                    ▷ Get West middle
8:         $esI = sI[width/2 : width, height/2 : height]$         ▷ Get East middle
9:         $highestMean$ = getHighestValue($\overline{nsI}, \overline{ssI}, \overline{wsI}, \overline{esI}$)
10:        if $highestMean == \overline{nsI}$ then                ▷ Highest tonality on top
11:            $rI$ = rotate($sI$,180)
12:        else if $highestMean == \overline{wgI}$ then
13:            $rI$ = rotate($sI$,270)
14:        else if $highestMean == \overline{egI}$ then
15:            $rI$ = rotate($sI$,90)
16:        else
17:            $rI = sI$
18:        end if
19:    else                                                       ▷ Rectangular image
20:        if $width < height$ then
21:            $gI$ = rotate($sI$,90)      ▷ Longest image side over x-axis (landscape)
22:        end if
23:        $nsI = sI[0 : width, 0 : height/2]$                    ▷ Get North middle
24:        $ssI = sI[0 : width, height/2 : height]$               ▷ Get South middle
25:        if $\overline{ngI} < \overline{sgI}$ then              ▷ Highest tonality on top
26:            $rI$ = rotate($sI$,180)
27:        else
28:            $rI = sI$
29:        end if
30:    end if
31:    return $rI$
32: end function
```

3.2 Hash-Based Transformations

Hash-based algorithms are the most suited for the problem of image matching because they are very fast. The pHash algorithm extends the aHash approach by using discrete cosine transform (DCT) [11] to reduce the frequencies. We have followed a similar schema; we defined the Algorithm as 3 and used it to obtain the hash associated with the image I, which is provided by a user of a social network. The input of this algorithm is the grayscale image gI, obtained in the preprocessing step.

Top-left 12×12 values are obtained because they represent the lowest frequency range. In contrast, the bottom right is the highest frequency range. The human eye is not very sensitive to high frequencies.

Algorithm 3. pHash-based algorithm

```
1: function getImageHash($gI$)
2:     $sI$ = reduceSize($gI, 32, 32$)                ▷ Reduce size to 32x32 pixels
3:     $rI$ = imageRotation($gI$,$sI$)                ▷ Rotate as defined in Algorithm 2
4:     $DCT$ = computeDCT($rI, 32, 32$)   ▷ Get a collection of frequencies and scalars
5:     $sDCT$ = reduceDCT($DCT, 12, 12$)      ▷ Get the lowest freq. (top-left 12x12)
6:     for each $px \in sDCT$ do
7:         if $px > \overline{sDCT}$ then                  ▷ Compare every pixel with sDCT mean
8:             $hash = hash + 1$
9:         else
10:            $hash = hash + 0$
11:        end if
12:    end for
13:    return $hash$
14: end function
```

As a result, we have the value of the hash composed of 144 values (12×12) 1 or 0 in order to evaluate the distance by using the Hamming distance algorithm, which simply compares each bit position and counts the number of differences.

4 Results

To perform the tests of the proposed system, a set of 200,000 images available in the public repository of Pixbay [10] was used as image dataset.

Figure 2 presents an example of the processing of two images obtained from the original. On the left side, there is an image with a yellowish hue, rotated 90°, with an outer border, and a watermark in the lower left corner. On the right, the processing of an image obtained directly from the original is shown. The result in both cases is a 144-digit value composed of 1 and 0, as detailed in the Algorithm 3. After calculating the Hamming distance, the system determines that both images are 99.3% equal.

To evaluate the performance of the algorithm, it was compared with the different implementations of hash-based algorithms. 1,000 images were obtained from the total set of the images to which different transformations were applied. The success rate was evaluated by considering the result a success for those cases in which the system associates the modified image with the original image of the dataset as the most similar, having a similarity value greater than 99%. The applied transformations and the images used are shown in Table 1.

All of these images were provided as input using the implementations of the pHash, aHas, and dHash algorithms, the proposed algorithm. Regarding to Tineye, whose algorithm is not public (although it has been published that it is based on hash), images have been processed by using its public API [13].

Following these indications, the results obtained are reflected in Table 2, where all images that have been catalogued as equal, and indeed were, are considered a success.

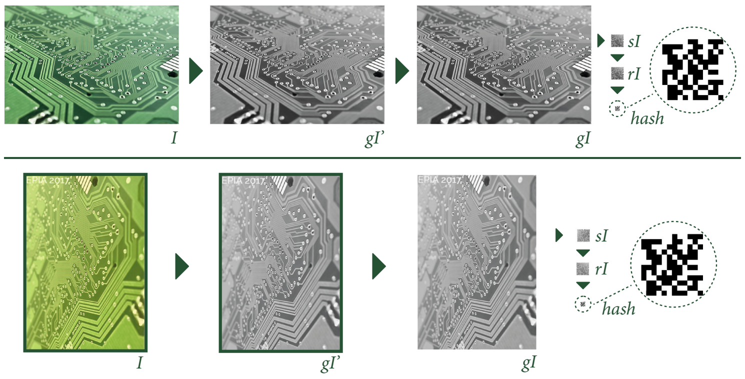

Fig. 2. Example of the system process. (Color figure online)

Table 1. Test dataset

	n	b	w	r	b + w	b + r	w + r	b + w + r	Total
Images	**125**	125	125	125	125	125	125	125	1000

Legend: n = none; b = border; w = watermark; r = rotation

Table 2. Hit rate for hash based algorithms

	n	b	w	r	b + w	b + r	w + r	b + w + r	Avg
pHash	**100%**	0%	75%	0%	0%	0%	0%	0%	22%
aHash	**100%**	0%	74%	0%	0%	0%	0%	0%	21%
dHash	**100%**	0%	75%	0%	0%	0%	0%	0%	22%
Tineye	**100%**	0%	**80%**	0%	0%	0%	0%	0%	22%
Proposed	**100%**	**90%**	75%	**100%**	**74%**	**90%**	**74%**	**74%**	**84%**

Legend: n = none; b = border; w = watermark; r = rotation; avg = average

It can be observed that the proposed system shows a better result in all transformations except when a watermark is included. In that case, Tineye and pHash show a higher success rate. In the case of Tineye, the details of its algorithm are not known. With respect to pHash, the improvement in the success rate is mainly due to the number of frequencies obtained when the DST is reduced (8×8), lower than for the proposed system (12×12).

5 Conclusion and Future Work

The proposed system improves current state of the art of image matching, by including images which have been slightly modified by the inclusion of a watermark, outer borders, or rotations of 90°, 180°, and 270°.

The results are robust in terms of the insertion of edges and rotations. However, with the insertion of watermarks which have considerably altered the image, none of the algorithms was able to associate the images with precision. In fact, in order not to introduce false positives (identify images as equal images when they are in fact not), it is necessary to compromise the detail with which one wants to perform the analysis.

As a future line of work, this solution will be incorporated into an existing job search social network in order to suggest contacts to users who have published or shared equal images. Regarding the image matching system, different solutions capable of associating images whose proportions have been modified by the user, either by trimming and removing part of the exterior of the image or by having deformed the image, are being evaluated. This evolution could make it possible to check rotations in each of the possible 360°.

Acknowledgments. This work was carried out under the frame of the project with ID RTC-2016-5642-6. The research of Pablo Chamoso has been financed by the Regional Ministry of Education in Castilla y León and the European Social Fund.

References

1. Aghav, S., Kumar, A., Gadakar, G., Mehta, A., Mhaisane, A.: Mitigation of rotational constraints in image based plagiarism detection using perceptual hash. Int. J. Comput. Sci. Trends Technol. **2**, 28–32 (2014)
2. De Paz, J.F., Rodríguez, S., Bajo, J., Corchado, J.M.: Mathematical model for dynamic case-based planning. Int. J. Comput. Math. **86**(10–11), 1719–1730 (2009)
3. Deng, H., Zhang, W., Mortensen, E., Dietterich, T., Shapiro, L.: Principal curvature-based region detector for object recognition. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE, June 2007
4. Hernandez, R.A.P., Miyatake, M.N., Kurkoski, B.M.: Robust image hashing using image normalization and SVD decomposition. In: 2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 1–4. IEEE, August 2011
5. Kamaruddin, S., Ghanib, N., Liong, C., Jemain, A.: Firearm classification using neural networks on ring of firing pin impression images. ADCAIJ: Adv. Distrib. Comput. Artif. Intell. J. **1**(3), 27–34 (2013). doi:10.14201/ADCAIJ20121312734
6. Krawetz, N.: Looks Like It (2011). http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-LikeIt.html. Accessed 12 Jan 2017
7. Krawetz, N.: Kind of Like That (2013). http://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html. Accessed 12 Jan 2017
8. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. Image Vis. Comput. **22**(10), 761–767 (2004)
9. Norouzi, M., Fleet, D.J., Salakhutdinov, R.R.: Hamming distance metric learning. In: Advances in Neural Information Processing Systems, pp. 1061–1069 (2012)
10. Pixabay.com. Free Images - Pixabay (2017). https://pixabay.com/. Accessed 17 Jan 2017
11. Rao, K.R., Yip, P.: Discrete Cosine Transform: Algorithms, Advantages, Applications. Academic Press, San Diego (2014)
12. Smith, S.M., Brady, J.M.: SUSAN—a new approach to low level image processing. Int. J. Comput. Vis. **23**(1), 45–78 (1997)
13. Tineye.com. TinEye Reverse Image Search (2017). https://www.tineye.com/. Accessed 12 Jan 2017