Evaluating Robustness of Perceptual Image Hashing Algorithms

Andrea Drmic, Marin Silic, Goran Delac, Klemo Vladimir, Adrian S. Kurdija
University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, Croatia
andrea.drmic@fer.hr, marin.silic@fer.hr, goran.delac@fer.hr, klemo.vladimir@fer.hr, adrian.kurdija@fer.hr

Abstract - In this paper we evaluate the robustness of perceptual image hashing algorithms. The image hashing algorithms are often used for various objectives, such as images search and retrieval, finding similar images, finding duplicates and near-duplicates in a large collection of images, etc. In our research, we examine the image hashing algorithms for images identification on the Internet. Hence, our goal is to evaluate the most popular perceptual image hashing algorithms with the respect to ability to track and identify images on the Internet and popular social network sites. Our basic criteria for evaluation of hashing algorithms is robustness. We find a hashing algorithm robust if it can identify the original image upon visible modifications are performed, such as resizing, color and contrast change, text insertion, swirl etc. Also, we want a robust hashing algorithm to identify and track images once they get uploaded on popular social network sites such as Instagram, Facebook or Google+. To evaluate robustness of perceptual hashing algorithms, we prepared an image database and we performed various physical image modifications. To compare robustness of hashing algorithms, we computed Precision, Recall and F1 score for each competing algorithm. The obtained evaluation results strongly confirm that P-hash is the most robust perceptual hashing algorithm.

I. Introduction

The image tracking has recently become an interesting problem since a significant number of images get uploaded on various social network sites instantly. For instance, the recent statistics show that Facebook handles over 350 million image uploads each day. Once the image gets uploaded on the social network, other users and influencers share the image, modify it and upload it as their own content. In case the image becomes very popular, it is very hard to distinguish who is the original author (source) without having an insight in social network private data.

In general, it is very hard to obtain such insider statistics since social networks have very strict policy about their data. A possible approach to track images online would be to utilize steganography [1]. However, this approach is very limited due to a fact that social network usually performs image compression prior to image upload which results in all hidden data being lost. The only promising solution to track images on various social networks from outside is to utilize image hashing algorithms.

The classic hashing algorithms used in cryptography such as MD5 and SHA1 are not suitable for this purpose since their aim is to hash entities with high dispersity. This means that similar entities are very likely to hash to quite different hashes. Even small changes in input will result in completely different hash value which is a good property of cryptographic hashing algorithm. On the other hand, we need a different family of hashing algorithms that will hash similar entities to similar hashes. Generally, the hashing algorithms having such property are called locality-sensitive hashing algorithms [2] [3]. More specifically, for image entities, *perceptual* image hashing algorithms are used, which possess the property of invariance for "perceptually similar" media objects x and x' [4].

In this paper, we analyze robustness property of perceptual image hashing algorithms. We consider the following algorithms: A-hash, D-hash, W-hash and P-hash. We conducted experiments in which we evaluated robustness property of mentioned image hashing algorithms. In our experiments, we measured robustness of image hashing algorithms regarding visible image modifications (1), and with the respect to images upload to popular social network sites: Facebook, Instagram and Google+ (2). To compare robustness of competing algorithms, we used well established measures: *Precision*, *Recall* and F_1 *score*. The evaluation results obtained from conducted experiments strongly suggest that P-hash algorithm is the most successful algorithm for image identification upon visible modification is applied or image is uploaded on social network.

The rest of the paper is organized as follows. Section II presents the related work. The competing hashing algorithms are described in section III, while the evaluation is presented in section IV. Section V concludes the paper.

II. Related Work

The researchers have proposed a variety of image hashing algorithms that extract popular image features (i.e. HOG, DOG, SIFT, GIST etc.) as large high-dimensional vectors which are later reduced using some dimensionality reduction technique with the aim to detect near-duplicate images. For instance, in [5], the authors extract local features based on DOG for image representation, and then use locality sensitive hashing as the core indexing structure. In [6], spectral hashing is used by performing PCA to find the principal components of the data, and then the data is fit to a multidimensional rectangle. Similar as previous, the approach proposed in [7] finds the maximum variance direction using PCA, except that the original covariance matrix gets "adjusted" by another matrix

arising from the labeled data. Instead of representing an image using a single feature vector, the approach proposed in [8] independently indexes large number of local descriptors derived from PCA-SIFT which results in approach highly resistant to occlusions and cropping. In [9], the authors use bag-of-words techniques for text analysis for creation of bag-of-visual-words using vector quantized local feature descriptors (SIFT), and they propose Min-Hash algorithm for retrieval of similar images. In addition, geometric image hashing [10] is proposed to improve standard Min-Hash by considering the spatial dependency among visual words. Further improvement over geometric Min-Hash is a hashing scheme for partial duplicate image discovery [11] i.e. finding groups of images in a large dataset that contain the same object. An interesting novel graph-based approach which automatically discovers the neighborhood structure inherent in the data is introduced in [12]. Kernelized locality sensitive hashing for scalable image search is proposed in [13]. More recent works propose deep learning frameworks to generate binary hash codes for fast image retrieval [14] [15].

All mentioned approaches are quite complex and computationally challenging since they usually create high dimensional image representation which is later reduced. However, our goal is to focus on perceptual image hashing algorithms that can produce fast image fingerprint and still preserve image identity.

III. Perceptual Image Hashing Algorithms

In this section, we briefly describe the following perceptual image hashing algorithms: A-hash, D-hash, P-Hash and W-hash.

A. A-Hash

Average hash (A-hash) is a perceptual image hashing algorithm that creates compact 64-bit image hashes by focusing on properties of image structure. As image is decomposed into its underlying harmonics, the higher frequencies represent image details, while the lower frequencies represent image structure. To make the image fingerprint as small as possible, the higher frequencies are eliminated by reducing the image in its size. Specifically, prior to computing the hash, the image is scaled down to an 8x8 block and it thus contains a total of 64 pixels. Each pixel is then converted to greyscale. Note that all perceptual image hashing algorithms employ this step since the essential semantic information is preserved in the luminance component of an image. Next, an average color for all 64 pixels is computed. Subsequently, the hash is constructed so that each bit representing a single pixel is set based on whether the color value of that pixel is below or above the computed image average.

B. D-Hash

Difference hash (D-hash) is a perceptual hashing algorithm that leverages image structure, much like in the A-hash approach. The hashing principle focuses on the image structure and it achieves so by reducing the image size, i.e. by removing higher frequencies from the image spectrum. Unlike in the A-hash approach where the fingerprint was computed be averaging out the pixels, the D-hash approach tracks image gradient. Prior to hashing, each image is reduced to a 9x8 block and converted to gray scale, i.e. a total of 72 pixels. Then for each row, the difference between each two adjacent pixels is computed, a total of 8 differences per row. Thus, 64 differences are computed for each image and then subsequently used to construct the fingerprint. This is done by setting each bit based on the computed difference d. For instance, if $d < 0$, the hash bit is set to 0, and if $d \geq 0$, the bit is set to 1.

C. P-Hash

The P-hash algorithm [4] is based on discrete cosine transformation (DCT). The algorithm produces a 64-bit length binary sequence as an image hash. First, the image is converted to a greyscale representation using its luminance. Next, a mean filter (i.e. smoothing, averaging or box filter) is applied to the image. To apply the filter, a kernel with dimension 7x7 is used. The kernel is applied using a special convolution function [4].

Once the convolution is applied, the image is resized to 32x32 pixels. To extract the hash, 64 low frequency coefficients are used except the lowest frequency coefficients are omitted. The low frequency coefficients are used because they are mostly stable under various image modifications. Moreover, most of the signal information is preserved in low frequency components of the DCT. The reason the lowest frequency coefficients are omitted is they tend to be quite different from others and can significantly throw off the average.

To produce the hash, DCT coefficients from (1, 1), which is the upper left corner of 64-size matrix, to DCT coefficient (8, 8), representing the lower right corner of the same matrix, are concatenated to an array of length 64. Next, the median m of coefficients array is computed. Finally, the hash is transformed into a binary form using the following procedure:

$$h_i = \begin{cases} 0, & C_i < m \\ 1, & C_i \geq m \end{cases} \quad (1)$$

where C_i is the ith coefficient in the constructed array, and h_i is the ith bit in a 64-bit length perceptual hash.

D. W-Hash

Wavelet hash (W-hash) algorithm is a perceptual image hashing algorithm that transforms the original problem into frequency domain. Similar as P-hash, W-hash utilizes frequency domain, but instead of discrete cosine transform, W-hash uses discrete wavelet transform (DWT). Wavelet transform represents a signal using wavelet functions with different locations and scales. Wavelets are particularly well suited for the representation of signals with the aggressive change in input signal, thus resulting in smaller amount of information in the frequency domain. DWT is often used to remove redundancy in a data with highly correlated neighboring values, such as pixels in images. DWT is successfully used for noise reduction [16], image compression [17], dimensionality reduction [18], EEG analysis [19] and audio signal analysis [20]. Basic W-hash algorithm used

during the experiments works by first scaling the image followed by transformation of an image to frequency domain using the *Haar* wavelet (optionally removing the lowest low level frequency). Provided implementation of the W-hash algorithm is using the wavelet transform software for the Python programming language.

Due to the nature of the wavelet transformation, it is expected that W-hash will perform better on images with smaller amount of intense changes i.e. high contrast spatial data. Also, it is reasonable to assume that P-hash will perform better for images with more restrained spatial changes.

IV. EVALUATION

In this section, we evaluate the robustness of perceptual image hashing algorithms. The implementation we use is https://pypi.python.org/pypi/ImageHash. To evaluate the robustness property, we define two different experiments. In the first experiment, we evaluate the algorithms robustness with respect to various visible modifications applied to images. In the second experiment, we evaluate algorithms robustness regarding the ability to track images on the Internet. To be more specific, we uploaded, and subsequently downloaded, images on various social network sites such as: Facebook, Instagram and Google+.

A. *Evaluating robustness on visible image modifications*

In this experiment, our goal is to check which is the most robust algorithm when considering various visible image modifications. First, we collected an image dataset containing 1480 original images. We computed a local database containing hashes for each considered image hashing algorithm: A-hash, D-hash, W-hash and P-hash. Then, we performed various images modifications for all images in the dataset. More specifically, the following image modifications were applied: resizing, rotating, noise injecting, swirling, sharpening, adding border color, contrast-stretching, colorizing, thresholding and drawing. To introduce all mentioned modifications, we used a very popular open source tool *ImageMagick* [21] which exposes its functionalities using Python API. The modification parameters were chosen randomly under the appropriate constraints. All together, we created an amount of 8868 modified images. We introduced the appropriate naming convention for modified images so we can easily track which is the original image used to obtain each modified image. For instance, if the original name was "100201.jpg", for modified image obtained by resizing the original image was named "100201_resized_*N*.jpg", where *N* is the index of modification created from the respected original image.

To evaluate the sole robustness of each image hashing algorithm, we used well established measures for this purpose: *precision*, *recall* and F_1 *score*. Precision measure is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

where *TP* is the number of true positives and *FP* is the number of false positives. Recall measure is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

where *FN* is the number of false negatives. Finally, the F_1 score is defined as follows:

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{4}$$

Once we have the number of true positives, false positives and false negatives, we can easily compute measures for each competing hashing algorithm. To compute number of true positives, false positives and false negatives in our experiment, we try to reconstruct which is the original image, for each modified image. Specifically, for each modified image, we compute its hash. Then, we search the original images hashes database to see if the computed hash matches any of hashes in the database. To determine if the two hashes *A* and *B* are considered equal, we examine hashes' in binary representation and we compute their Hamming distance as follows:

```
for mod_img in modified_images:
  mod_hash = compute_hash(mod_img, alg)
  for org_hash, org_img in hashes_db:
    d = hamming_distance(mod_hash, org_hash)
    if mod_img.name.startsWith(org_img.name):
      if d <= N:
        TP = TP + 1
      else:
        FN = FN + 1
    else:
      if d <= N:
        FP = FP + 1
      else:
        TN = TN + 1
```

Figure 1 Procedure used during the evaluation for each algorithm

$$d(A, B) = \sum |A_i - B_i|. \tag{5}$$

To consider two hashes A and B equal, we require:

$$d(A, B) \leq N, \tag{6}$$

where *N* is a system parameter which can be tuned for each environment and each algorithm. During the evaluation, we perform the procedure depicted in Figure 1.

Knowing the ground truth based on a modified image name, and having a Hamming distance for each modified image and each original image hashes pair, we can easily figure out if some hash pair (i.e. original image hash, and modified image hash) is a true positive, false positive, true negative or false negative. We repeat the depicted procedure for each algorithm, varying the parameter *N* in range from 0 to 30. It should be noted that the hash size for each algorithm is 64 bits.

The precision, recall and F1 score results for algorithms A-hash, D-hash, W-hash and P-hash are depicted in Figure , Figure , Figure 5, and Figure 66, respectively. As can be seen in the figures, the most successful algorithm is P-hash, which achieves F_1 *score* value of 0.738 at *Precision* value of 0.856 and *Recall* value of 0.649 for a distance threshold $N = 14$. The second most successful algorithm is D-hash, which reaches F_1 *score* value of 0.641 at *Precision* value of

0.745 and *Recall* value of 0.563 for a distance threshold $N = 10$. The third place is reserved for A-hash algorithm, which gains F_1 *score* value of 0.406 at *Precision* value of 0.431 and *Recall* value of 0.385 for a distance threshold $N = 1$. Finally, the worst performance is obtained by W-hash algorithm, which obtains F_1 *score* value of 0.271 at *Precision* value of 0.249 and *Recall* value of 0.297 for a distance threshold $N = 0$.

As already described in the beginning of this section, the modifications introduced in this experiment setup were quite aggressive. This means that it is relatively hard to associate the original and the modified image for some examples by sole human visual inspection. Figure presents an assembled collage of some randomly chosen examples of original images (marked with yellow border), and their modifications. The modifications whose original was successfully identified using P-hash algorithm at distance threshold $N = 14$ are marked with green border, while modified images whose original could not have been discovered are marked with red border.

One could also argue the effectiveness of various algorithms depends of what the goal function is. In case the aim is finding very similar images, P-hash algorithm may be considered too tolerant to aggressive image modifications, i.e. the results will contain some images which are not similar to the original image. In this case, more appropriate choice would be A-hash algorithm which is less tolerant to visible image modifications.

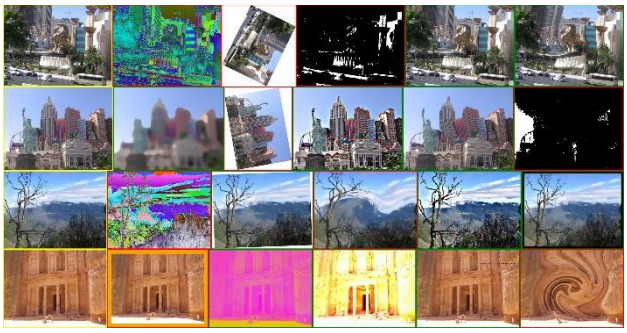

Figure 2 Examples of original images and their modifications

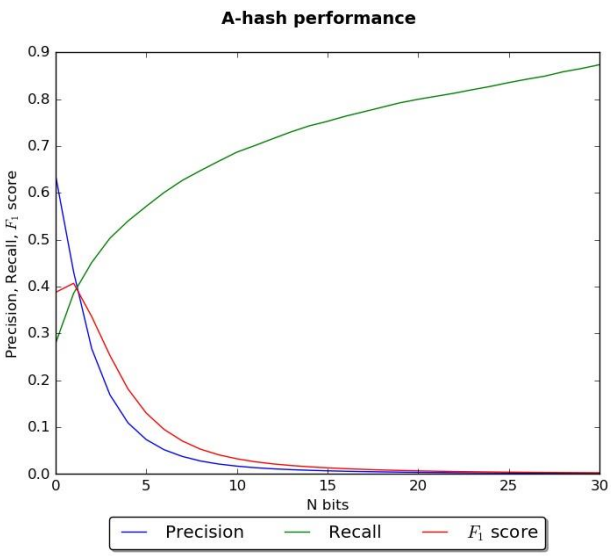

Figure 3 A-hash algorithm performance on modified images

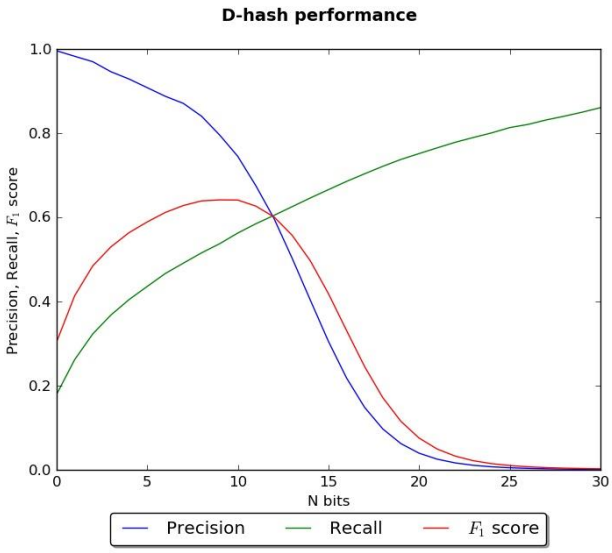

Figure 4 D-hash algorithm performance on modified images

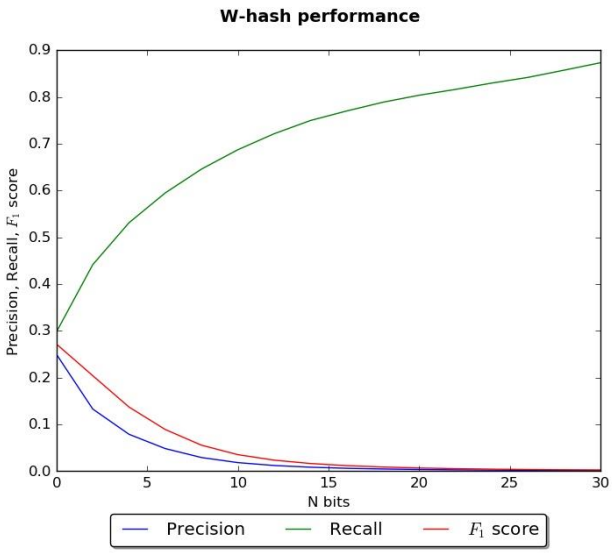

Figure 5 W-hash algorithm performance on modified images

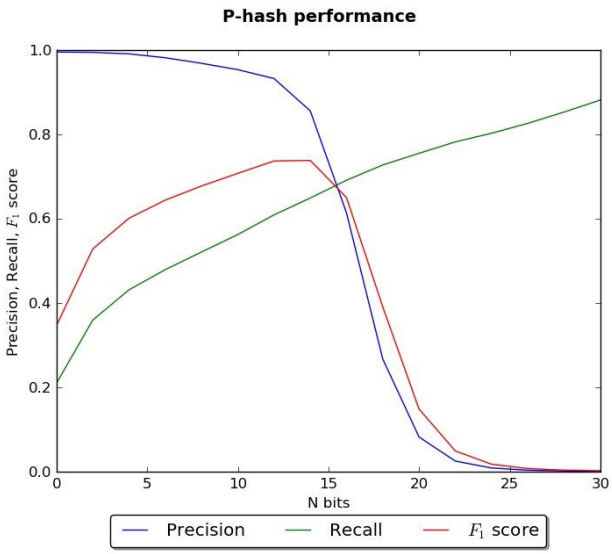

Figure 6 P-hash algorithm performance on modified images

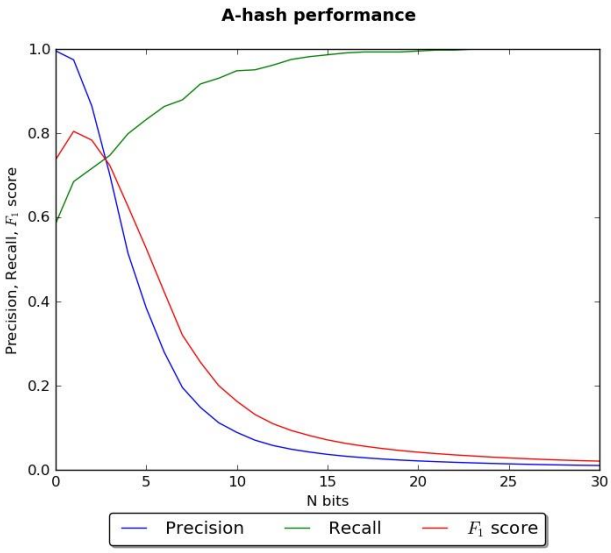

Figure 2 A-hash performance on social network dataset

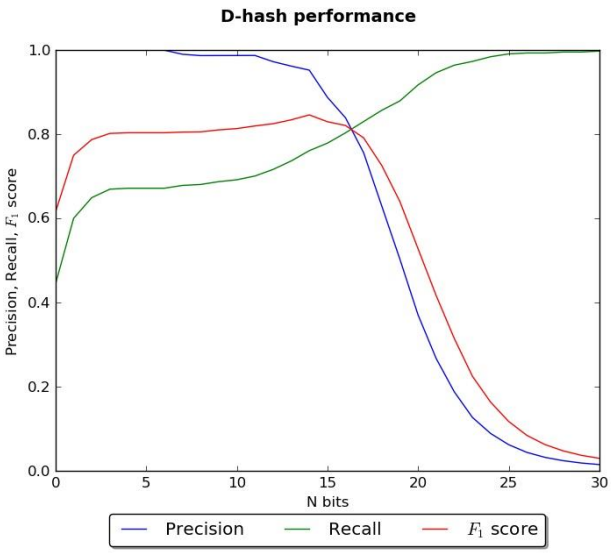

Figure 3 D-hash performance on social network dataset

B. *Evaluating robustness on images uploaded on social networks*

In this experiment, our goal is to analyze robustness on less aggressively modified images – images uploaded on social network sites. More specifically, in our experiment we included the following social network sites: *Facebook*, *Instagram* and *Google+*. It is well known that all considered sites provide users the ability to upload and share images as part of their profiles and feeds. However, while being uploaded the images are resized to dimensions dominantly used on uploading site (1), and the images are processed by *jpeg* compression engine which removes any irrelevant information (2). The resizing operation is quite likely to change the aspect ratio and the quality of the original image. Hence, it is likely that the image will change after being uploaded. Our goal is to determine the robustness of hashing algorithms with the respect to upload on social network sites.

We selected an amount of 150 original images from the dataset used in the previous experiment (see IV.A). Then, each image was uploaded to all considered social networks. After the upload, the images were downloaded from social networks and included to modified images dataset containing overall 450 images. In the same manner, as in the first experiment we created a database of hashes for each original image and each competing hashing algorithm. To measure the robustness for each competing algorithm, we used the same evaluation measures as in the first experiment: *Precision*, *Recall* and F_1 *score*. Furthermore, we used the same evaluation procedure described in pseudocode in shown Fig. 1.

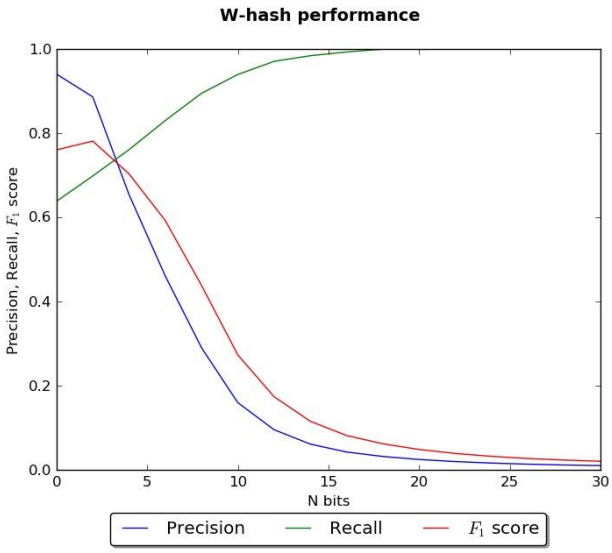

Figure 4 W-hash performance on social network dataset

The results for algorithms A-hash, D-hash, W-hash and P-hash are depicted in Figure 2, Figure 3, Figure 4, and Figure 5, respectively. Same as in the first experiment, the most successful algorithm is P-hash, which achieves F_1 *score* value of 0.864 at *Precision* value of 0.926 and *Recall* value of 0.81 for a distance threshold $N = 16$. The second most successful algorithm is D-hash, which reaches F_1 *score* value of 0.846 at *Precision* value of 0.952 and *Recall* value of 0.761 for a distance threshold $N = 14$. D-hash is followed by A-hash algorithm, which gains F_1 *score* value of 0.804 at *Precision* value of 0.974 and *Recall* value of 0.685 for a distance threshold $N = 1$. Finally, the worst performance is obtained by W-hash algorithm, which obtains F_1 *score* value of 0.781 at *Precision* value of 0.886 and *Recall* value of 0.698 for a distance threshold $N = 1$.

It is obvious that all algorithms perform significantly better on social network dataset when compared to the original image modification dataset. However, this behavior is expected since some of the introduced image modifications are quite intense, and it is hard for a human observer to associate modified image with its origin.

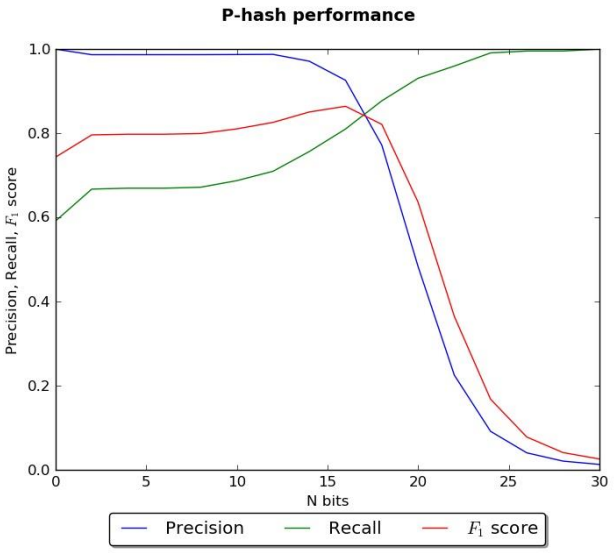

Figure 5 P-hash performance on social network dataset

V. CONCLUSION

In this paper, we studied robustness property of the following perceptual image hashing algorithms: A-hash, D-hash, W-hash and P-hash. To be precise, we analyze how robust image identification and tracking is with the respect to: visible physical image modifications (1), and image upload on social networks (2).

To assess robustness with the respect to visible image modifications, we created a dataset containing original images and their modifications. To evaluate robustness regarding upload on social network, we selected original images and uploaded them on the following social networks: Facebook, Instagram and Google+. To compare the performance of different algorithms, we used common measures: *Precision*, *Recall* and F_1 *score*.

The evaluation results, obtained on visible image modifications and both social networks uploaded dataset, strongly confirm that the most robust algorithm is P-hash which achieves F_1 *score* value of 0.738 on image modifications dataset, and F_1 *score* value of 0.864 on social networks uploaded dataset.

A detailed analysis of how a particular image hashing algorithm is robust with respect to a particular image modification is left for future work.

ACKNOWLEDGMENT

The authors acknowledge the support of the Croatian Science Foundation through the Recommender System for Service-oriented Architecture **(IP-2014-09-9606)** research project. The authors also acknowledge the support of Consumer Computing Laboratory of Faculty of Electrical Engineering and Computing at the University of Zagreb.

REFERENCES

[1] G. C. Kessler and C. Hosmer, "An overview of steganography," *Advances in Computers*, pp. 51-107, 2011.

[2] J. Leskovec, A. Rajaraman and J. D. Ullman, Mining of massive datasets, Cambridge University Press, 2014.

[3] K. Zhao, H. Lu and J. Mei, "Locality Preserving Hashing," *AAAI*, pp. 2874-2881, 2014.

[4] C. Zauner, Implementation and benchmarking of perceptual image hash functions, 2010.

[5] X. Yang, Q. Zhu and K.-T. Cheng, "MyFinder: near-duplicate detection for large image collections," in *ACM international conference on Multimedia*, 2009.

[6] Y. Weiss, A. Torralba and R. Fergus, "Spectral hashing," *Advances in neural information processing systems*, pp. 1753-1760, 2009.

[7] J. Wang, S. Kumar and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[8] R. Datta, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Computing Surveys*, 2008.

[9] O. Chum, "Near Duplicate Image Detection: min-Hash and tf-idf Weighting," *BMVC*, pp. 812-815, 2008.

[10] O. Chum, M. Perd'och and J. Matas, "Geometric min-hashing: Finding a (thick) needle in a haystack," in *Computer Vision and Pattern Recognition*, 2009.

[11] D. C. Lee, Q. KE and M. Isard, "Partition min-hash for partial duplicate image discovery," in *European Conference on Computer Vision*, 2010.

[12] W. Liu, "Hashing with graphs," in *International conference on machine learning*, 2011.

[13] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *International Conference on Computer Vision*, 2009.

[14] K. Lin, "Deep learning of binary hash codes for fast image retrieval," in *Conference on computer vision and pattern recognition*, 2015.

[15] R. Xia, "Supervised Hashing for Image Retrieval via Image Representation Learning," in *AAAI*, 2014.

[16] M. Lang, H. Guo, J. E. Odegard, C. S. Burrus and R. O. Wells, "Noise reduction using an undecimated discrete wavelet transform," *IEEE Signal Processing Letters*, vol. 3, no. 1, pp. 10-12, 1996.

[17] S. Grgic, M. Grgic and B. Zovko-Cihlar, "Performance analysis of image compression using wavelets," *IEEE Transactions on industrial electronics*, pp. 682-695, 2001.

[18] L. M. Bruce, C. H. Koger and J. LI, "Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction," *IEEE Transactions on geoscience and remote sensing*, pp. 2331-2338, 2002.

[19] H. Ocak, "Automatic detection of epileptic seizures in EEG using discrete wavelet transform and approximate entropy," *Expert Systems with Applications*, pp. 2027-2036, 2009.

[20] G. Tzanetakis, G. Essl and P. Cook, "Audio analysis using the discrete wavelet transform," in *Acoustics and Music Theory Applications*, 2001.

[21] "ImageMagick," ImageMagick Studio LLC, 2017. [Online]. Available: http://imagemagick.org/script/index.php.

[22] B. Wang, "Large-scale duplicate detection for web image search," in *International Conference on Multimedia and Expo*, 2006.