Rihamark: Perceptual image hash benchmarking

Christoph Zauner[a] and Martin Steinebach[b] and Eckehard Hermann[a]

[a]Upper Austria University of Applied Sciences, Campus Hagenberg, Department Secure Information Systems, Austria;
[b]Fraunhofer Institute for Secure Information Technology, Germany

ABSTRACT

We identify which hash function has the best characteristics for various applications. In some of those the computation speed may be the most important, in others the ability to distinguish similar images, and sometimes the robustness of the hash against attacks is the primary goal. We compare the hash functions and provide test results. The block mean value based image hash function outperforms the other hash functions in terms of speed. The discrete cosine transform (DCT) based image hash function is the slowest. Although the Marr-Hildreth operator based image hash function is neither the fastest nor the most robust, it offers by far the best discriminative abilities. Interestingly enough, the performance in terms of discriminative ability does not depend on the content of the images. That is, no matter whether the visual appearance of the images compared was very similar or not, the performance of the particular hash function did not change significantly. Different image operations, like horizontal flipping, rotating or resizing, were used to test the robustness of the image hash functions. An interesting result is that none of the tested image hash function is robust against flipping an image horizontally.

Keywords: Perceptual image hash, robust image hash, image fingerprint, benchmarking, evaluation

1. INTRODUCTION

Perceptual image hash functions produce hash values based on the image's visual appearance. A perceptual hash can also be referred to as e.g. a robust hash or a fingerprint. Such a function calculates similar hash values for similar images, whereas for dissimilar images dissimilar hash values are calculated. Finally, using an adequate distance or similarity function to compare two perceptual hash values, it can be decided whether two images are perceptually different or not. Perceptual image hash functions can be used e.g. for the identification or integrity verification of images.

Rihamark[1]*, a novel benchmarking framework for perceptual image hash functions, was used to benchmark four different perceptual image hash strategies. A **discrete cosine transform (DCT)**, a **Marr-Hildreth operator**, a **radial variance** and a **block mean value based perceptual image hash function**.

Rihamark is a benchmarking framework for perceptual image hash functions. It features a modular architecture and comes bundled with various default plugins. Plugins can be used e.g. to integrate perceptual image hash function libraries into Rihamark or to execute certain attacks on pictures. Test plans can be generated by using a graphical user interface.

pHash† provided the implementation of the previously mentioned perceptual image hash functions. pHash is an open source software library that implements several perceptual hashing algorithms, and provides a C-like API to use those functions. The first three hash functions were already implemented by pHash. The last hash function was implemented into pHash by us.

Further author information: (Send correspondence to Christoph Zauner)
Christoph Zauner: E-mail: christoph.zauner@nllk.net, WWW: http://zauner.nllk.net
Martin Steinebach: E-mail: martin.steinebach@sit.fraunhofer.de
Eckehard Hermann: E-mail: eckehard.hermann@fh-hagenberg.at

*Homepage: `http://rihamark.nllk.net`
†Homepage: `http://www.phash.org/`

Media Watermarking, Security, and Forensics III, edited by Nasir D. Memon, Jana Dittmann, Adnan M. Alattar, Edward J. Delp III, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7880 78800X · © 2011 SPIE-IS&T · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.876617

The motivation of our work is to identify which hash function has the best characteristics for various applications. In some of those the computation speed may be the most important, in others the ability to distinguish similar images, and sometimes the robustness of the hash against attacks is the primary goal. We compare the hash functions and provide test results.

1.1 The employed hash functions

1.1.1 DCT based hash

Various properties of the DCT can be utilized to create perceptual image hash functions. Low-frequency DCT coefficients of an image are mostly stable under image manipulations.[2] That is because most of the signal information tends to be concentrated in a few low-frequency components of the DCT. This property is also utilized by the Joint Photographics Experts Group (JPEG) image compression standard. There, the two-dimensional type-II DCTs of NxN pixel blocks are computed and the results are quantized. N is typically 8 and the type-II DCT formula is applied to each row and column of the block. The result is an $8x8$ transform coefficient array in which the elements close to the top-left (index position $(0, 0)$) represent low-frequency components and are therefore deemed to be perceptually most significant. Coefficients with increasing vertical and horizontal index values represent higher vertical and horizontal frequency components. Reference 3 shows that a feature code can be extracted from the relationship between two DCT coefficients of the same position in two separate blocks. This property is especially useful for image integrity verification systems, which are expected to pass only JPEG compression. In summary, that is because all DCT coefficient matrices are divided by the same quantization table in the JPEG compression process.

The pHash implementation is actually inspired by a DCT based perceptual video hash function.[4] It utilizes the former property discussed in the previous paragraph. The method first converts the image to grey scale using only its luminance. This step is common to all perceptual image hash functions, because the essential semantic information resides in the luminance component of an image. Then a mean filter[‡] is applied to the image. A kernel with dimension $7x7$ is used. After this operation the image is resized to $32x32$ pixels. Consequently, a DCT matrix is generated and the two-dimensional type-II DCT coefficients are calculated using matrix multiplications. The image is square. Therefore the two-dimensional DCT can be computed by multiplying the DCT matrix with the image and the transposed DCT matrix.

As proposed in ref. 4, 64 low-frequency DCT coefficients, omitting the lowest frequency coefficients, are selected for hash extraction. pHash therefore selects $8x8$ transform coefficients. The selected coefficients form a square matrix. The coefficient $\mathrm{DCT}(1, 1)$ being the upper left corner of the matrix and the coefficient $\mathrm{DCT}(8, 8)$ being the lower right corner of the matrix. The rows of the square matrix are stringed together forming a one-dimensional array of length 64. Let the DCT coefficients of the array be denoted as $C_i, i = 0, \ldots, 63$. Once the median m of the 64 DCT coefficients has been determined, the sequence can be normalized into a binary form as follows to form the final hash value

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

where h_i is the bit of the perceptual image hash at position i. To measure the distance between two hash values the hamming distance is used. Each hash value is 64 bits long.

1.1.2 Marr-Hildreth operator based hash

Several perceptual image hash functions that use edge detectors for feature extraction have been proposed (see e.g. ref. 5). The Marr-Hildreth operator, also denoted as the LoG, is a special case of a discrete Laplace filter. The filter kernel is constructed by applying the Laplace operator onto a Gauss function. Because of its visual representation, it is also called “mexican hat” filter. That is, because when visualized in three dimensions it looks like a sombrero hat. The LoG can be tuned to detect edges at a particular scale. The LoG filter, denoted as $h_c(x, y)$, can be defined as (cmp. ref. 6)

[‡]Other common names are smoothing, averaging or box filter.

$$
\begin{aligned}
h_c(x,y) &= \nabla^2 g_c(x,y) \\
&= \frac{x^2+y^2-2\sigma^2}{\sigma^4} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}.
\end{aligned} \tag{2}
$$

To implement the LoG in discrete form, one may construct a filter by sampling equation equ. (2) after selecting an value for σ. The filter then may be applied to an image by using 2D convolution.

The pHash implementation has not been proposed previously. The authors of pHash rather implemented their own approach with reagard to e.g. feature extraction. Before feature extraction, various pre-processing steps are applied to the image. First and foremost, the image is converted to grey scale. Then it is blurred using a Canny-Deriche filter. σ is set to 1.0. After that, the image is resized to a resolution of 512 x 512 pixels. Then a histogramm-equalized version of the image is calculated using 256 histogram levels. Finally the LoG kernel is applied to the image.

The normalized hamming distance is used to measure the distance between two hash values. Each hash value is 576 bit long.

1.1.3 Radial variance based hash

A perceptual image hash function based on the Radon transform was proposed by Lefèbvre and Macq in September 2002.[7] A few years later, in April 2005, both authors outlined[8] that their previously proposed algorithm suffers from some troubles. Thereupon they introduced a new algorithm[8, 9] to overcome these problems.

The Radon transform is the integral transform consisting of the integral of a function over a straight line. It is robust against various image processing steps (e.g. compression) and geometrical transformations (e.g. rotation). In ref. 7 a new visual content descriptor, based on the Radon transform, was presented. Let α denote the angle of the used projection line. x denotes the coordinate of a pixel along the x-axis, whereas y denotes the coordinate of a pixel along the y-axis. To extend the Radon transform to discrete images, the line integral along $d = x \cdot cos\alpha + y \cdot sin\alpha$ can be approximated by a summation of the pixels lying in the one pixel wide strip (see ref. 8):

$$
d - \frac{1}{2} \leq x \cdot cos\alpha + y \cdot sin\alpha \leq d + \frac{1}{2}. \tag{3}
$$

The algorithm proposed in ref. 8 uses the variance instead of the sum of the pixel values along the line projections. The variance captures luminace discontinuities along the projection lines much better. Such discontinuities result from edges, that are orthogonal to the projection direction. The so-called radial variance vector ($R[\alpha]$) is therefore defined as follows. Let $\Gamma(\alpha)$ denote the set of pixels (x, y) on the projection line corresponding to a given angle α. Let (x', y') denote the coordinates of the central pixel of the image. $(x, y) \in \Gamma(\alpha)$ if

$$
-\frac{1}{2} \leq (x - x') \cdot cos\alpha + (y - y') \cdot sin\alpha \leq \frac{1}{2}. \tag{4}
$$

Let $I(x, y)$ denote the luminance value of the pixel (x, y), the radial variance vector $R[\alpha]$, where $\alpha = 0, 1, \ldots, 179$, is then defined by

$$
R[\alpha] = \frac{\sum_{(x,y)\in\Gamma(\alpha)} I^2(x,y)}{\#\Gamma(\alpha)} - \Big(\frac{\sum_{(x,y)\in\Gamma(\alpha)} I(x,y)}{\#\Gamma(\alpha)}\Big)^2. \tag{5}
$$

As discussed in ref. 7, it is sufficient to extract 180 instead of 360 values. That is because the Radon transform is symmetric. Finally, in ref. 9, the perceptual image hash function was further improved by applying the DCT to the radial variance vector. The first 40 coefficients of the transformed radial variance vector form the so-called radial hash vector in the end. This omits redundant components of the radial variance vector and efficiently decorrelates it.

pHash implements the algorithm as proposed in ref. 9. At first, the image is converted to grey scale. After that pHash implements a few additional image pre-processing steps (blurring and gamma correction). Of the discussed perceptual image hash functions, the radial variance based image hash function is the only one which does not normalize the image with respect to resolution. None of the papers that proposed radial variance based hash functions[7–9] discusses any image normalization operations that may make sense when implementing such a hash function. In ref. 7 the term normalization is mentioned but no further details are outlined.

The hash is 320 bit (40 byte) long. The peak of cross correlation (PCC) is used to compare two hash values.

1.1.4 Block mean value based hash

In 2006, Bian Yang, Fan Gu and Xiamu Niu proposed a block mean value based perceptual image hash function.[10] Four slightly different methods are proposed. The latter two additionally incorporate an image rotation operation to enhance robustness against rotation attacks. This significantly increases the computational complexity of the latter two methods. To secure the perceptual image hash values encryption using a secret key is used.

1.1.5 Method 1

The first method is described as follows:

a) Convert the image to grey scale and normalize the original image into a preset size.

b) Let N denote the bit length (e.g. 256 bit) of the final hash value. Divide the pixels of the image I into non-overlapped blocks $I_1, I_2, \dots, I_N$.

c) Encrypt the indices of the block sequence $\{I_1, I_2, \dots, I_N\}$ using a secret key K to obtain a block sequence with a new scanning order $\{I'_1, I'_2, \dots, I'_N\}$. Reference 10 specifies no further details about what encryption algorithm to use. So it is up to the implementor of this perceptual image hash function to choose an adequate one.

d) Calculate the mean of the pixel values of each block. That is, from the block sequence $\{I'_1, I'_2, \dots, I'_N\}$ calculate the corresponding mean value sequence $\{M_1, M_2, \dots, M_N\}$. Finally obtain the median value M_d of the mean value sequence.

e) Normalize the mean value sequence into a binary form and obtain the hash value h as

$$h(i) = \begin{cases} 0 & , M_i < M_d \\ 1 & , M_i \geq M_d \end{cases} . \tag{6}$$

Reference 10 proposed four slightly different methods of this image hash function. The first two have been implemented into pHash by us. The encryption of the indices of the block sequence using a secret key is omitted by this implementation (step c of method 1. The normalized hamming distance is used to measure the distance between two hash values. As outlined in ref. 10 the image is converted to grey scale and resized to a square resolution. The default resolution of the pHash implementation is 256 x 256 pixels.

1.1.6 Method 2

The only difference to the first method is that the pixels of the image are divided into **overlapped** blocks. The degree of overlapping is set to be half the size of a block. If a preset size of 16x16 pixels is chosen and a block size of 4x4 pixels is used, the first method would yield a hash with a bit size of 16 bits. Using this method the pixels of the image would be divided into 49 blocks. Therefore the hash would have a size of 49 bits.

2. METHODOLOGY

Various topics were identified for benchmarking. The **speed**, **inter score distribution** and the **intra score distribution**. The topics will be reviewed in the following subsections.

2.1 Speed

The speed of the implementation of an algortihm can be used to judge it's computational complexity. The speed of a perceptual image hash function is especially important when a great number of images needs to be hashed and processed. This is e.g. the case when searching the World Wide Web for copyright infringements.

2.2 Inter score distribution

The inter score distribution (non-authentic attempts) can be used to measure and judge the discriminative capabilities of a perceptual hash function. When comparing two different images a perfect perceptual hash function would always yield a distance (or similarity score) of 0.5. An interesting question is whether the score distribution depends on the used images. When using very similar images it may be more difficult for a perceptual image hash function to achieve the "perfect" distance of 0.5 for every comparison. A thousand images, all of them depicting snow-covered mountains, can be considered as such a similar image set. Therefore the inter score distribution was benchmarked using two different image sets. The first set was the chaos image set (dissimilar images), whereas the second set was the duck image set (very similar images).

2.3 Intra score distribution

The intra score distributions (authentic attempts) were benchmarked using different operations. The intra score distribution can be used to judge the robustness of a perceptual hash function. An ideal perceptual hash function would calculate a distance score of 0 or rather a similarity score of 1 for images which are perceptually the same. JPEG compression and resizing are one of the most commonly used image operations which users employ to modify their images. They use these operations in order to reduce the file size of their images for instance. The rotation operation is especially often used in scientific papers to demonstrate the robustness of perceptual image hash functions. The horizontal flipping operation was used because it hardly changes the human perception of an image.

If the purpose of a benchmark is to measure the effects of an operation, it is important to keep in mind that the saving of an image using certain image formats (e.g. the JPEG image format) is an image modification or manipulation itself. Normally the process of applying an operation to an image is as follows. The image file is read from the hard disc. Then the image is decoded and stored in a custom raw format in the system memory. Subsequently, the operation (e.g. flipping the image horizontally) is applied. Finally, the image is converted from the custom raw format into a standardized image format and the result is written to the hard disc. It is important to use only image formats that do not use lossy compression for such benchmarks. An adequate image format would be e.g. the Portable Network Graphics (PNG) image format. For these benchmarks the used image sets were converted from the JPEG image format to the PNG image format.

3. DATA

Different image sets were used for the benchmarks. All the images were obtained from Wikimedia Commons[§]. The images are so-called "quality images". Quality images are images which meet certain quality standards (which are mostly technical in nature) and which are valuable for Wikimedia projects. The used image file format, if not stated otherwise, was JPEG. The first set, hereinafter referred to as the **event image set**, consists of images with very different motifs. The image set consists of 47 images. The images of the set depict various events. The mean dimension of the images is 2874 x 2260 pixels. Their mean file size is 3.19MiB. The total file size of the 47 images is 149.77MiB. This image set was taken from Wikimedia Commons quality images of events web site[¶]. The second image set, hereinafter referred to as **duck image set**, mainly consists of photographs that show ducks swimming in the water. It consists of 45 images. The mean dimension of the images is 2732 x 1802 pixels. The mean file size is 2.75MiB. The total file size of the 45 images is 123.79MiB. The image set is a selection of images from the Wikimedia Commons quality images of birds web site[‖]. The third image set, hereinafter referred to as **chaos image set**, consists of images with varying motifs. The images

[§]Homepage: `http://commons.wikimedia.org/`

[¶]`http://commons.wikimedia.org/wiki/Commons:Quality_images/Subject/Events`

[‖]`http://commons.wikimedia.org/wiki/Commons:Quality_images/Subject/Animals/Birds` were taken from various quality image sets from Wikimedia Commons. It consists of 45 images. The mean dimension of the images is 2502 x 2200 pixels. The mean file size is 2.44MiB. The total file size of the 45 images is 109.90MiB. This image set and the last image set described below were also converted to the lossless compression format PNG. This was neccessary because, as outlined in the previous section, images saved in lossy compression formats should not be used for intra score distribution benchmarks. After conversion, the total file size of the chaos image set increased from 109.90 to 343.85MiB. As a result, the mean file size also increased from 2.44 to 7.64MiB. The fourth and last image set is a subset of the chaos image set. Hereinafter it will be referred to as the **small chaos image set**. Three images were taken from the chaos image set to form this image set. The mean dimension of the images is 3003 x 2222 pixels. The mean file size is 3.51MiB. The total file size of the 3 images is 10.54MiB. The last image set was also converted to the PNG image format. The total file size of the small chaos image set increased from 10.54 to 34.88MiB. Consequently the mean file size also increased from 3.51 to 11.63MiB.

The following sections present and discuss the results of the benchmarks. Each topic is adressed in a separate section. All perceptual image hash functions were configured to use their default parameters (see table 2). Table 1 shows the hardware and software of the system on which the benchmarking was carried out.

Table 1. Hard- and software of the system used for benchmarking.

CPU	Intel Core 2 Duo T9300 (2.50GHz)
RAM	4096MiB
HDD	Seagate Momentus 5400.4 250GB (SATA, 3Gb/s), Model Nr.: ST9250827AS
OS	32-Bit GNU/Linux distribution

Table 2. pHash default parameters.

DCT based hash	
No parameters available.	n/a
Marr-Hildreth operator based hash	
Wavelet scale factor	2
Scale factor level	1
Radial variance based hash	
Sigma (radius) of the gaussian filter	1
Gamma correction	1
Number of angles to consider	180
Block mean value based image hash	
Method	1
Preset size X	256
Block size X	16

4. RESULTS

4.1 Speed

For benchmarking the event image set was used. The assembled test plan consisted of one test item. The test type option was set to “Intra”. Therefore each perceptual image hash function had to hash $47 \times 2 = 94$ images. An attack of type “Empty” was added to the attack chain. Therefore Rihamark copied the images from the “Image Directory” to the “Attack Image Directory” without changing them.

The result of the benchmark is shown in figure 8.1. The results are summarized in table 3. The newly implemented block mean value based perceptual image hash function is the fastest hash function. It needs **58** seconds to hash 94 images. The second fastest hash function, with **118** seconds, is the radial variance based hash function. Far behind are the Marr-Hildreth operator based (**343** seconds) and the DCT based (**911** seconds) hash functions. The great differences in speed can be explained by the fact that the former two hash functions only use pixel operations for feature extraction when calculating the hash. By contrast, the latter two use computationally more expensive convolution/correlation operations.

Table 3. Statistical results of the speed benchmark. The best result in each category is printed in bold.

	DCT	MH	Radial	BMB
Total time (sec.)	911	343	118	**58**
avg. sec. per image	9.7	3.6	1.3	**0.6**
MiB/sec.	0.33	0.87	2.54	**5.16**

5. INTER SCORE DISTRIBUTION

As previously outlined the inter score distribution was benchmarked using two different image sets. The chaos image set consists of perceptually very different images, whereas the duck set consists of perceptually very similar images. Each perceptual image hash function had to calculate $\binom{45}{2} = 990$ hash values per image set.

The results of the intra tests are depicted in figures 8.2 − 8.2 and summarized in table 4. Both the figures and the summary apparently indicate that the Marr-Hildreth operator based image hash has by far the most discriminative abilities. The DCT based image hash performs as second best. Figure 8.2 suggests that when using the DCT based hash function, specific distance scores occur very often (e.g. 0.531 or 0.469). The radial variance based image hash is on a par with the block mean value based image hash. Another interesting conclusion can be drawn from this benchmark. The composition of the image set does not significantly influence the performance of the benchmarked perceptual image hash function.

Table 4. Statistical results of the inter tests. The best result in each category is printed in bold.

	DCT	MH	Radial	BMB
Run 0 (chaos set):				
Mean dist.	**0.501**	**0.499**	0.565	0.482
Max. dist.	0.688	**0.578**	0.812	0.812
Min. dist.	0.250	**0.408**	0.135	0.109
Run 1 (duck set):				
Mean dist.	0.496	**0.500**	0.532	0.478
Max. dist.	0.750	**0.569**	0.835	0.844
Min. dist.	0.219	**0.439**	0.077	0.133

Furthermore, it was examined if the inter score values can be improved by combining the tested perceptual image hash functions. Table 5 shows the statistical results. The score values of each image were summed up and divided by the number of the used perceptual image hash functions. It can be concluded that, regardless of which combination of hash functions is used, the Marr-Hildreth operator based image hash function delivers the best results.

Table 5. Inter score values when combinging different perceptual image hash functions.

	DCT + MH + Radial + BMB	DCT + Radial + BMB
Mean dist.	0.512	0.516
Max. dist.	0.656	0.701
Min. dist.	0.302	0.231

6. INTRA SCORE DISTRIBUTION

Various common image operations were used to test the robustness of the perceptual image hash functions. The chaos image set was used for all the score distribution charts. The small chaos image set was used for the effect of attack charts. Because image formats using lossy compression methods should not be used for such benchmarks the PNG version of both image sets were used.

6.1 Horizontal Flipping

If an image is flipped, its binary representation is changed drastically, though its perception to the human visual system and its semantic meaning changes only minimally or not at all. Therefore, such an image operation is worth considering. Figure 8.3 depicts the results of this benchmark. Table 6 summarizes them. None of the tested perceptual image hash functions is robust against horizontal flipping.

Table 6. Statistical results of the intra test. The images were changed by horizontally flipping them. The best result in each category is printed in bold.

	DCT	MH	Radial	BMB
Run 0:				
Mean dist.	0.497	0.483	0.499	**0.315**
Max. dist.	**0.625**	0.658	0.732	0.703
Min. dist.	0.375	0.276	**0.042**	0.047

6.2 Resizing

Figure 8.3 shows the results of this benchmark and table 7 summarizes them. The images were changed by resizing the width to 1024 pixels. The height was adjusted proportionally. Bicubic interpolation was used. The radial variance based image hash function is not robust to resizing. This may stem from the fact that this hash function does not normalize the resolution of an image before extracting its features.

Table 7. Statistical results of the intra test. The images were changed by resizing the width to 1024 pixels. The height was adjusted proportionally. The best result in each category is printed in bold.

	DCT	MH	Radial	BMB
Run 0:				
Mean dist.	0.076	0.068	0.348	**0.012**
Max. dist.	0.219	0.271	0.670	**0.039**
Min. dist.	**0.000**	0.017	0.008	**0.000**

6.3 JPEG Compression

Figure 8.3 depicts the results of this benchmark. Table 8 summarizes them. The images were changed using a JPEG quality setting of 80. Furthermore, the impact of the JPEG quality setting on the robustness of the perceptual image hash functions was investigated. Therefore the JPEG quality was gradually varied from 100 to 0. For each value of the quality parameter, the hash functions had to calculate the distance scores of the given images. The average distance scores as a function of the quality parameter are depicted in figure 8.3. The radial variance based image hash function is almost not influenced at all by the quality parameter. Even when using a quality parameter of 0 the average distance score of this hash function is negligible. The same applies to the DCT and block mean value based image hash functions up to a quality parameter of 10. The Marr-Hildreth operator based image hash function performs the worst.

Table 8. Statistical results of the intra test. The images were changed using JPEG compression with a quality parameter of 80. The best result in each category is printed in bold.

	DCT	MH	Radial	BMB
Run 0:				
Mean dist.	0.001	0.045	**0.000**	0.001
Max. dist.	0.031	0.253	**0.000**	0.008
Min. dist.	**0.000**	0.002	**0.000**	**0.000**

6.4 Rotation

Figure 8.3 depicts the results of this benchmark. Table 9 summarizes them. The images were rotated by 5° and bicubic interpolation was used. Using this kind of image operation none of the tested image hash functions is robust. Figure 8.3 depicts the results when the angle of rotation is varied gradually. The block mean value based image hash function performs the best. When using a threshold of 0.15, it is robust against rotation up to 3°.

Table 9. Statistical results of the intra test. The images were changed by rotating them by 5°. The best result in each category is printed in bold.

	DCT	MH	Radial	BMB
Run 0:				
Mean dist.	0.335	0.486	0.358	**0.220**
Max. dist.	0.500	0.547	0.792	**0.375**
Min. dist.	0.062	0.337	**0.052**	0.117

7. CONCLUSION

The newly implemented block mean based perceptual image hash function is faster than all the other functions. With regard to JPEG compression, rotation and resize operations it is either the most robust one or at least on a par with the other functions. None of the tested image hash functions is robust against flipping an image horizontally. Although the Marr-Hildreth operator based hash function is behind other functions when it comes to robustness, it has by far the most discriminative abilities.

The different properties of these hash functions can be leveraged by combining them. For instance, an image identification system could be implemented by combining the block mean value based image hash function together with the Marr-Hildreth operator based function. The block mean value based function would process the images in the first cycle. The candidates it identifies would then be passed on to the Marr-Hildreth operator based function. Such an image identification system would offer excellent performance in terms of speed and possess a great discriminative capability. Of course the Marr-Hildreth operator based function would be the limiting factor in terms of the robustness of the identification system. The inter score distribution can not be improved any further by combining the tested hash functions.

8. CHARTS OF THE BENCHMARK RESULTS

8.1 Speed

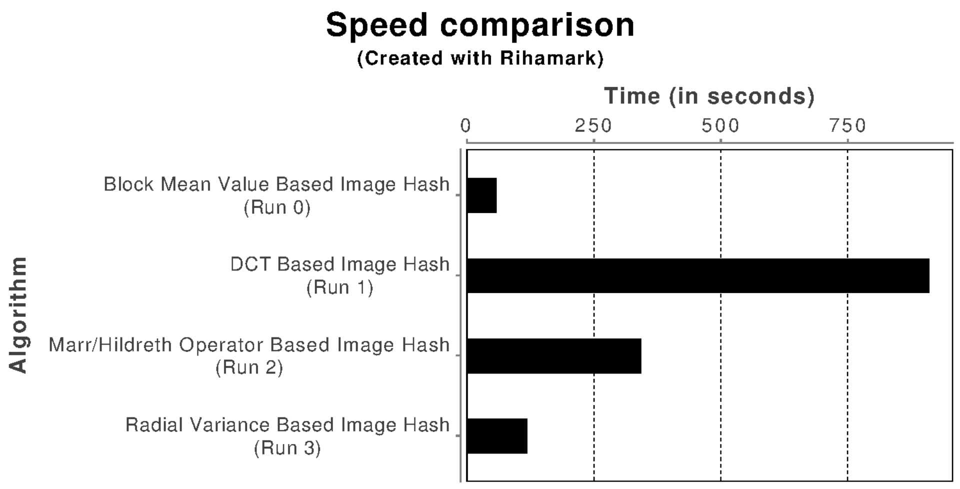

Figure 1. Results of the speed benchmark.

8.2 Inter Score Distribution

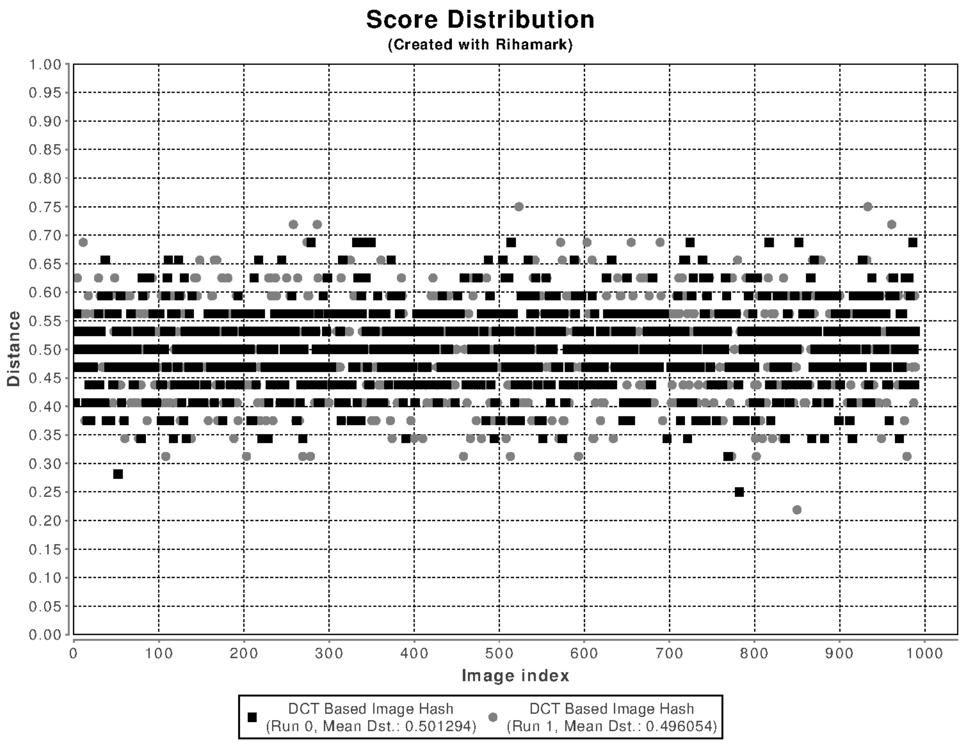

Figure 2. Results of the DCT based image hash function for two inter tests (chaos / duck image sets).

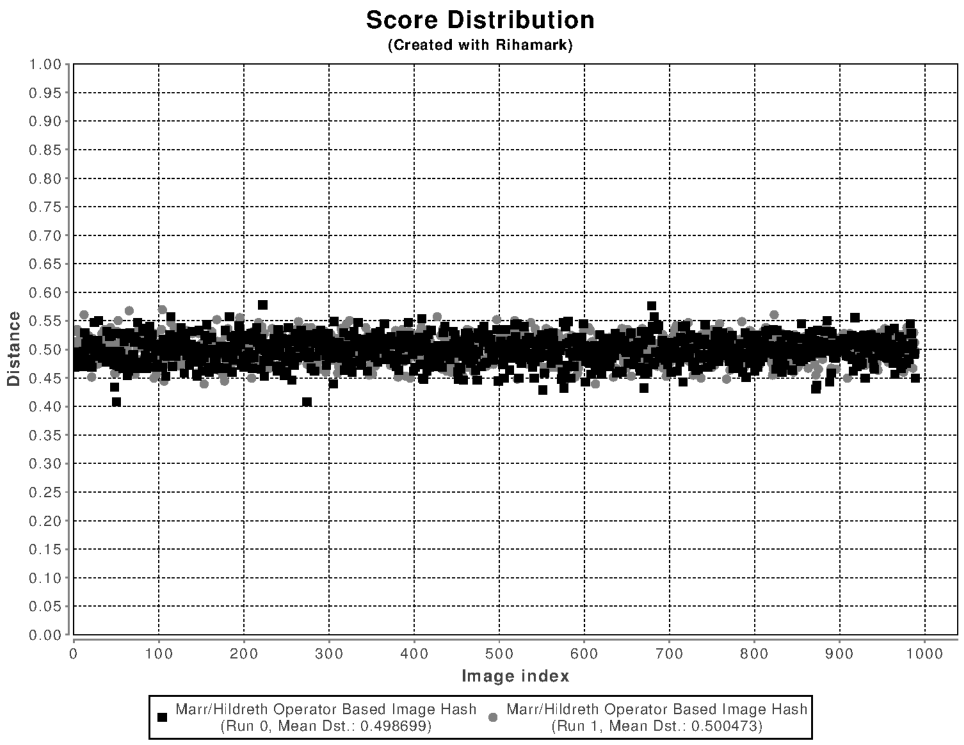

Figure 3. Results of the Marr-Hildreth operator based image hash function for two inter tests (chaos / duck image sets).

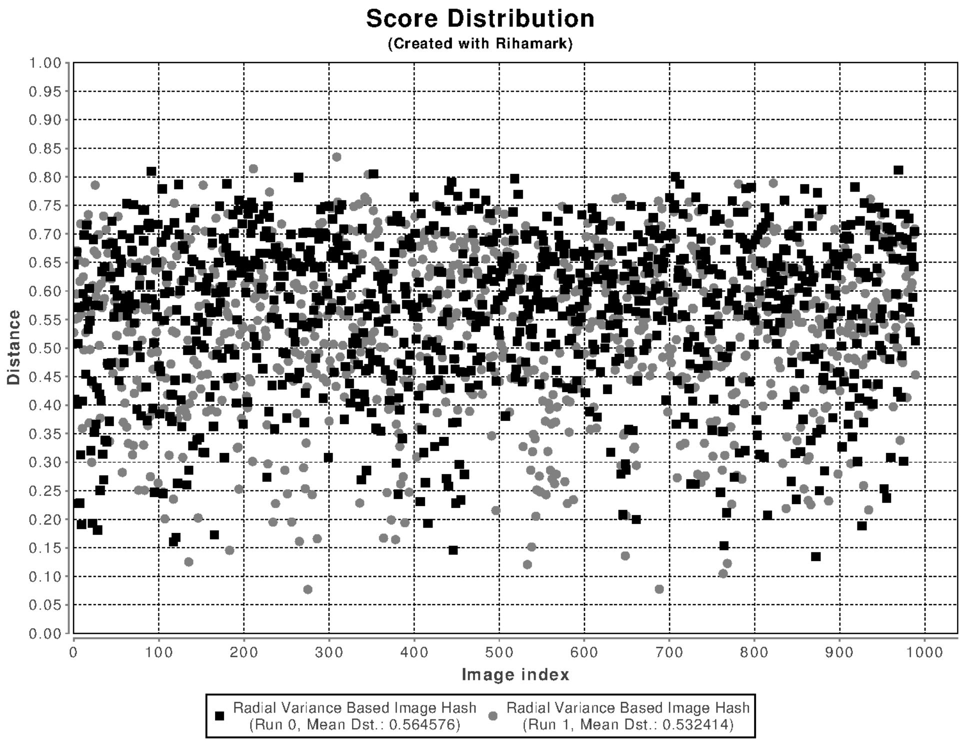

Figure 4. Results of the radial variance based image hash function for two inter tests (chaos / duck image sets).

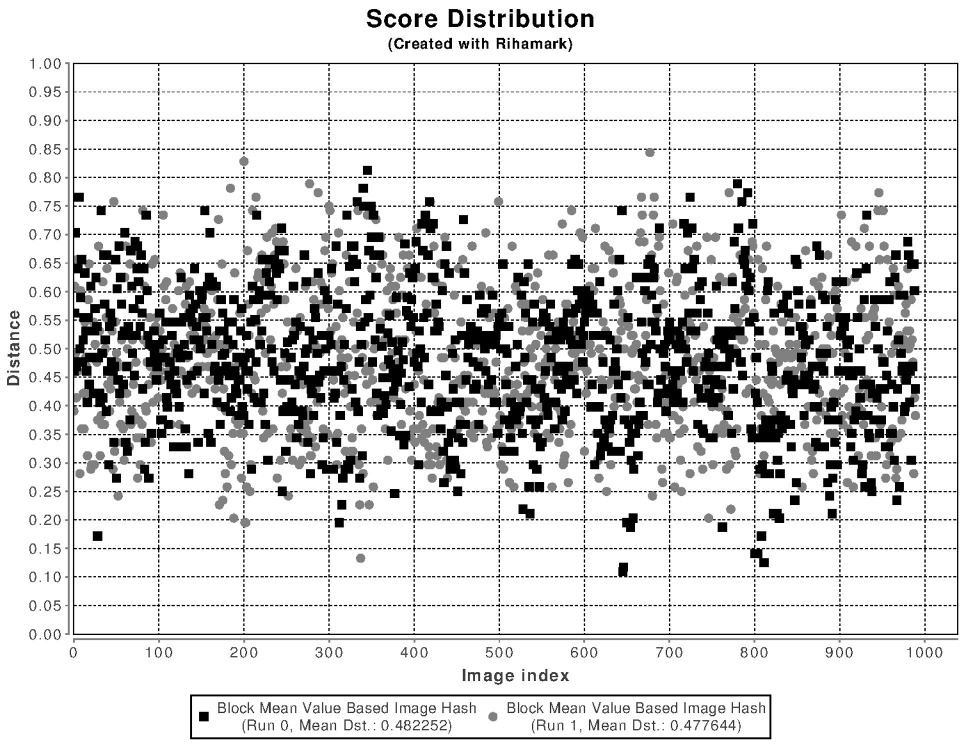

Figure 5. Results of the block mean value based image hash function for two inter tests (chaos / duck image sets).

8.3 Intra Score Distribution

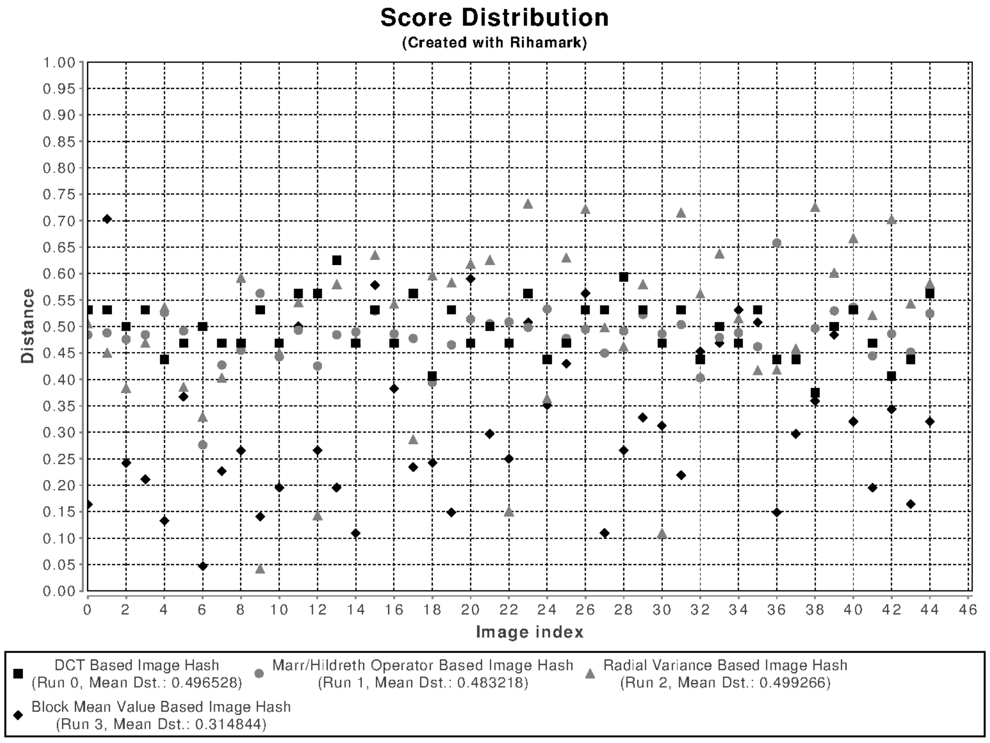

Figure 6. The images were changed by horizontally flipping them.

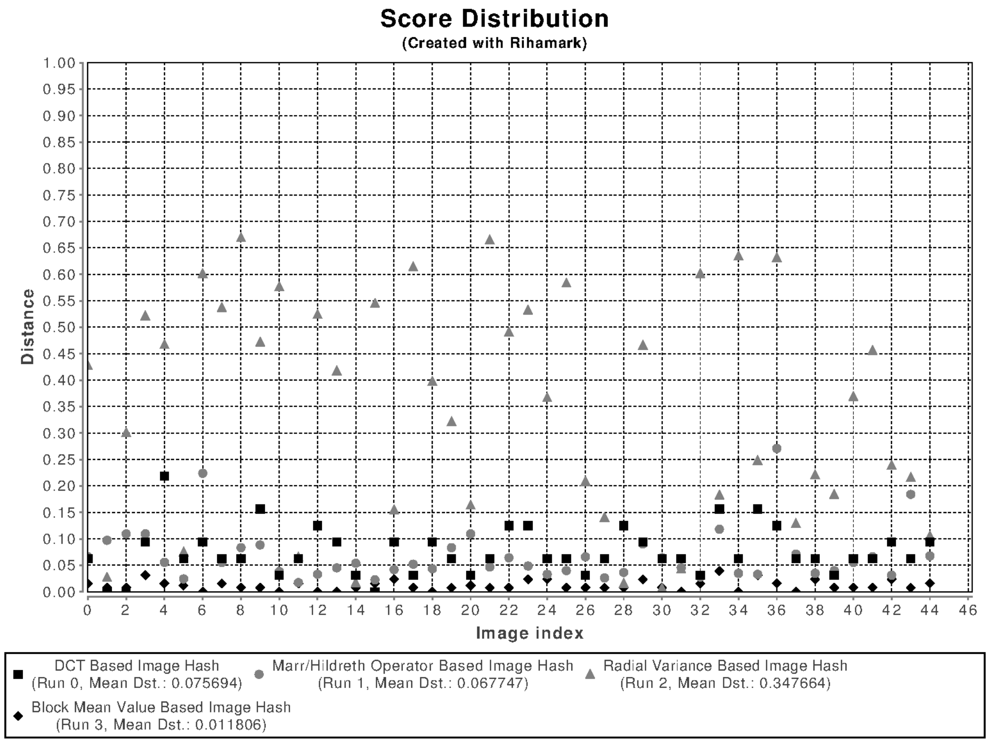

Figure 7. The width of the images was resized to 1024 pixels. The height was adjusted proportionally.

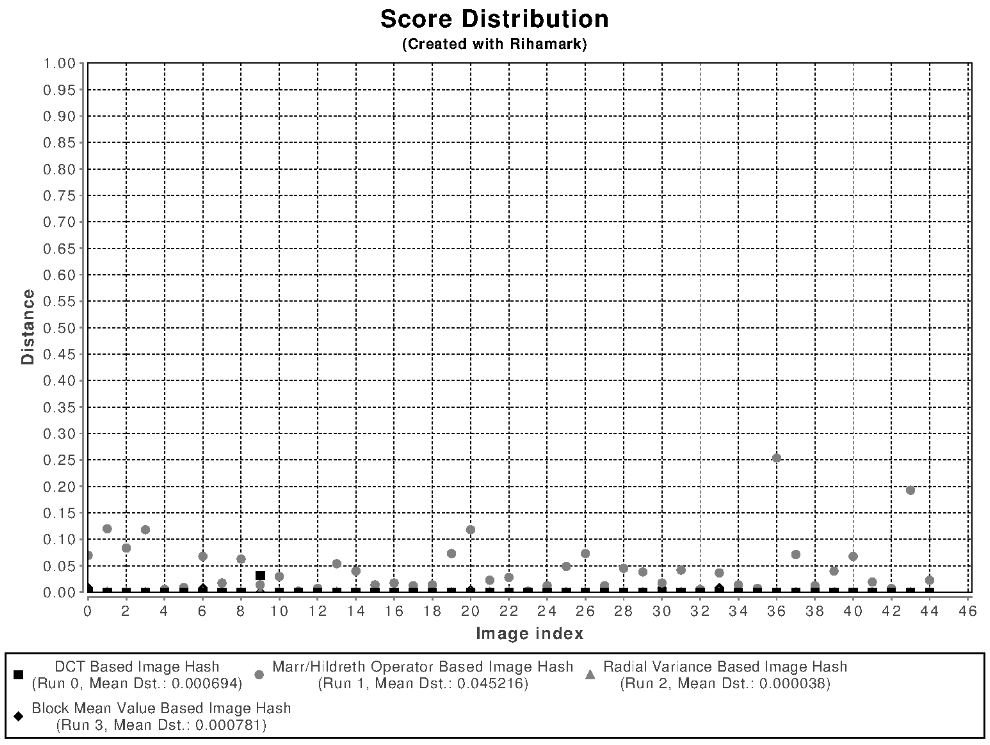

Figure 8. The images were changed using JPEG compression with a quality parameter of 80.

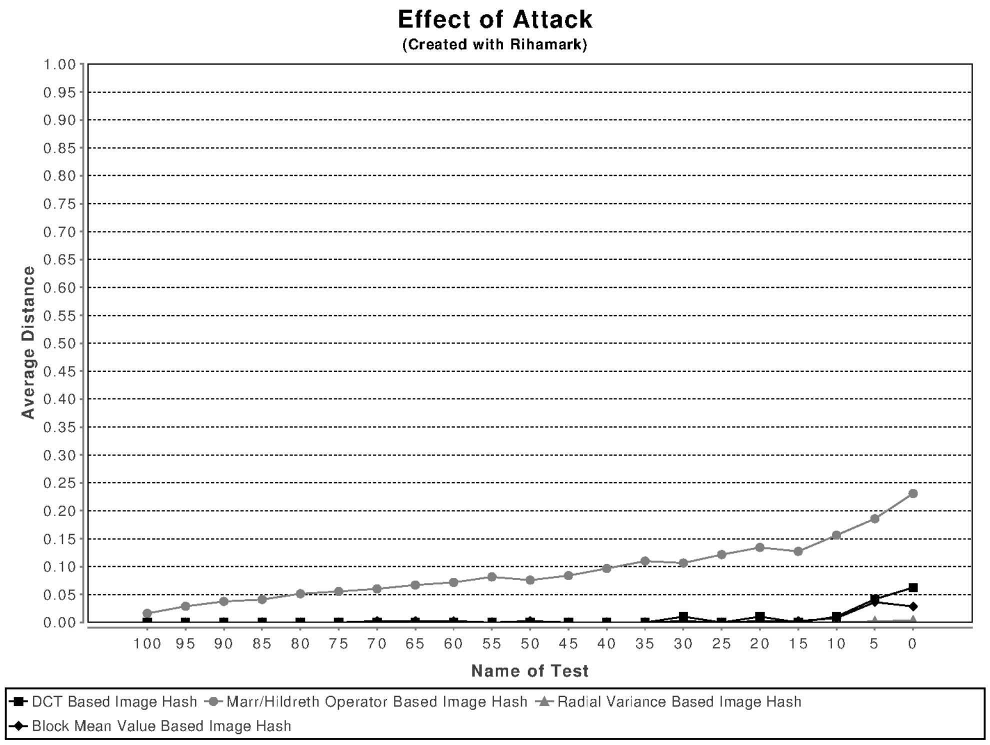

Figure 9. The JPEG quality parameter was gradually varied from 100 to 0.

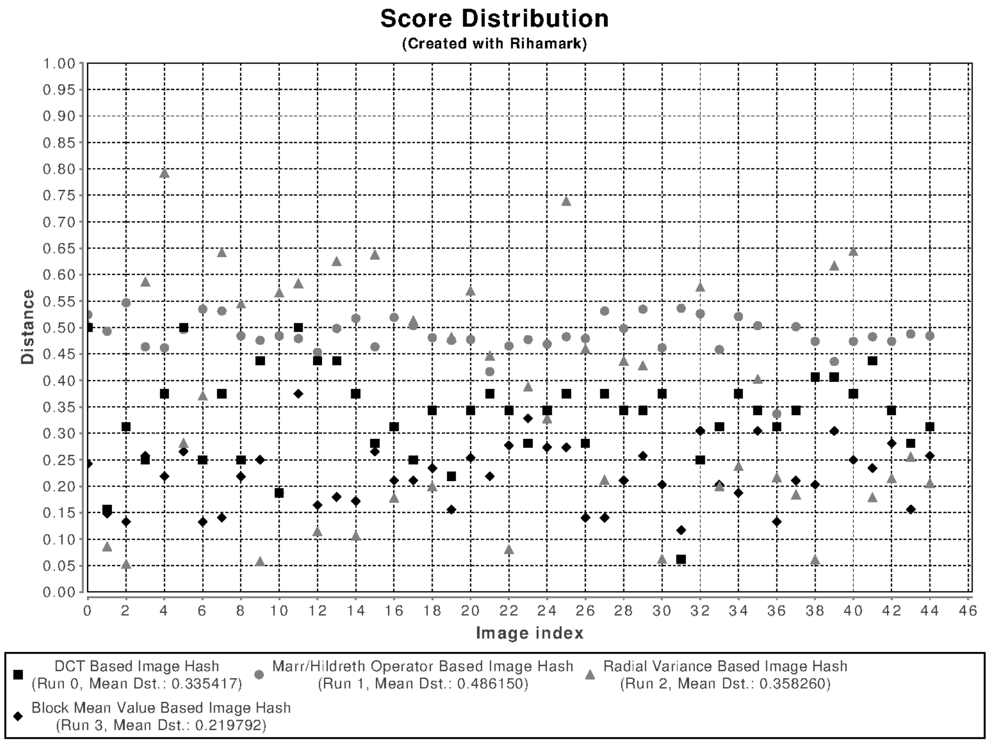

Figure 10. The images were rotated by 5 degrees.

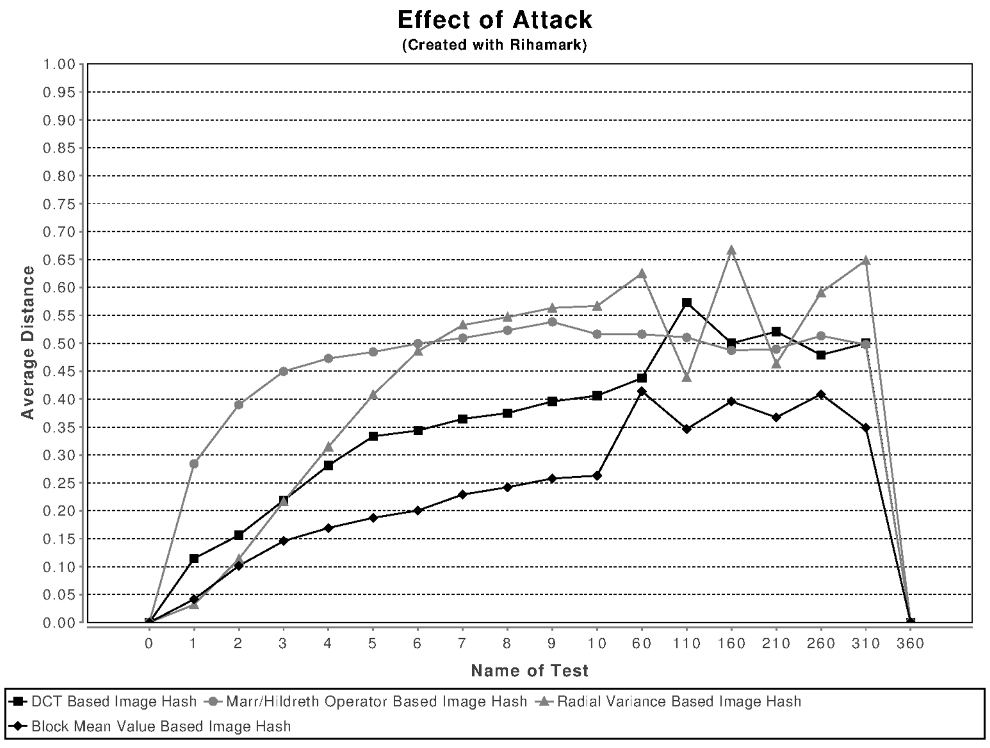

Figure 11. The angle was gradually varied $(0^\circ, 1^\circ, \ldots, 10^\circ, 60^\circ, \ldots, 360^\circ)$.

REFERENCES

[1] Zauner, C., *Implementation and Benchmarking of Perceptual Image Hash Functions*, Master's thesis, Upper Austria University of Applied Sciences, Hagenberg Campus (July 2010).

[2] Fridrich, J., "Robust bit extraction from images," in [*Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS)*], **2**, 536–540, IEEE (June 1999).

[3] Lin, C.-Y. and Chang, S.-F., "A robust image authentication method distinguishing JPEG compression from malicious manipulation," *IEEE Transactions on Circuits and Systems for Video Technology* **11**, 153–168 (Feb. 2001).

[4] Coskun, B. and Sankur, B., "Robust video hash extraction," in [*Proceedings of the Signal Processing and Communications Applications Conference*], 292–295, IEEE (Apr. 2004).

[5] Bhattacharjee, S. and Kutter, M., "Compression tolerant image authentication," in [*Proceedings of the International Conference on Image Processing (ICIP)*], **1**, 435–439, IEEE (Oct. 1998).

[6] Bovik, A., ed., [*The Essential Guide to Image Processing*], Academic Press (2009).

[7] Lefèbvre, F., Macq, B., and Legat, J.-D., "RASh: RAdon Soft Hash algorithm," in [*Proceedings of the European Signal Processing Conference (EUSIPCO)*], **I**, 299–302, European Association for Signal Processing (Sept. 2002).

[8] Standaert, F.-X., Lefèbvre, F., Rouvroy, G., Macq, B. M., Quisquater, J.-J., and Legat, J.-D., "Practical evaluation of a radial soft hash algorithm," in [*Proceedings of the International Conference on Information Technology: Coding and Computing*], **2**, 89–94 (Apr. 2005).

[9] Roover, C. D., Vleeschouwer, C. D., Lefèbvre, F., and Macq, B. M., "Robust image hashing based on radial variance of pixels," in [*Proceedings of the International Conference on Image Processing (ICIP)*], **3**, 77–80, IEEE (Sept. 2005).

[10] Yang, B., Gu, F., and Niu, X., "Block mean value based image perceptual hashing," in [*Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Multimedia Signal Processing (IIH-MSP)*], 167–172, IEEE (2006).