INFORMATICA, 2012, Vol. 23, No. 4, 645–663

Fast Convex Layers Algorithm for Near-Duplicate Image Detection

Smiljan ŠINJUR, Damjan ZAZULA, Borut ŽALIK
Facultaty of Electrical Engineering and Computer Science, University of Maribor
Smetanova ul. 17, 2000 Maribor, Slovenia
e-mail: smiljan.sinjur@uni-mb.si, zazula@uni-mb.si, zalik@uni-mb.si

Received: November 2010; accepted: July 2012

Abstract. This paper builds on a novel, fast algorithm for generating the convex layers on grid points with linear time complexity. Convex layers are extracted from the binary image. The obtained convex hulls are characterized by the number of their vertices and used as representative image features. A computational geometric approach to near-duplicate image detection stems from these features. Similarity of feature vectors of given images is assessed by correlation coefficient. This way, all images with closely related structure and contents can be retrieved from large databases of images quickly and efficiently. The algorithm can be used in various applications such as video surveillance, image and video duplication search, or image alignment. Our approach is rather robust up to moderate signal-to-noise ratios, tolerates lossy image compression, and copes with translated, rotated and scaled image contents.

Keywords: near-duplicate image detection, feature extraction, geometric features, convex layers, similarity measure.

1. Introduction

Image matching compares images in order to assess their similarity. A subclass of image similarity assessment deals with near-duplicate images. The problem of near-duplicate image detection is important in many applications, such as object and scene recognition, texture classification, stereo matching, feature tracking, content-based image retrieval, and data mining. First image matching approaches were developed back in fifties (Hobrough, 1959), with early solutions based on cross-correlation (Hannah, 1959). Also the approaches are known where images are described by keywords as metadata. The keywords for the similarity searches are generated either by the user or by the computer (Viitanierni and Laaksonen, 2005). Another alternative is to extract and compare for similarity local and/or global features of images (Wang *et al.*, 2007; Yang *et al.*, 2006). Similarity between images can be measured by an approximation of the Earth Mover's Distance (EMD), which leads to a fast computation of minima l-cost correspondences between two sets of local features (Grauman and Darrell, 2005; Chum *et al.*, 2008). Several algorithms for area-based matching have also been proposed. The area-based matching approaches use a probabilistic formulation of image matching in terms of maximum-likelihood estimation (Zhang and Chang, 2004) that can be applied to both the edge template matching

and grey-level image matching (Olson, 2002). Some approaches use colour histogram or edge pixels and consider images as a whole (Heipke, 1996). These approaches are robust to small variations of viewpoint, illumination changes or small changes on the image, but are sensitive to outliers which occur on the edges of image objects.

A number of commercial products exist for image retrieval (Google Image Search, 2011; Bing Image Search, 2011; Tiltomo, 2011), and (Incogna Image Search, 2011). Most of them are metadata search engines, where each image in the database has attached a set of metadata that are extracted from image name, user defined, or extracted from image content by one of feature extraction methods, e.g., like SIFT (Lowe, 2004). Another solution (Tiltomo, 2011), supports searching the images by similarity of either subject/colour/texture or colour/texture to a reference image.

Our paper describes an alternative method applied to near-duplicate image detection. It is based on the convex layers constructed on the points whose values exceed a given threshold. The image foreground is, thus, represented with only one geometric feature, which results in fast computational times. Image matching is determined by the similarity of image features, i.e., the vertices on convex layers, compared by their correlation coefficient. The proposed approach is fast and robust, but cannot cope with image similarity in general, such as images similar in only some separate regions. It, however, is fairly efficient with images that match as a whole. As we prove in the sequel, high additive noise corruption, scaling and small rotations around a multiple of $\pi/2$ do not degrade the performance of our approach. It can, therefore, be applied to extensive image database retrievals to find out whether a selected image is contained in the database or not.

The proposed algorithm is composed of the following five steps: image scaling, colour reduction, geometric feature extraction, feature dimension reduction, and duplicate detection test. Image scaling and colour reduction are the two basic steps of our solution. They are crucial to gain suitable input for the third step, i.e., geometric feature extraction. By scaling, the sizes of tested images are unified. Both compared images are further transformed into binary form which is most suitable for the geometric feature extraction based on convex layers. Since convex layers represent two-dimensional features, further dimension reduction is performed in order to correlate and assess them for possible near-duplication.

Time complexity of described algorithm is linear. Each image is indexed – its feature vector is created – only once in time $O(n)$, where n is the number of pixels in the image. If the feature vector length equals m, any subsequent search for near-duplicate images by our algorithm shows the complexity $O(m)$. Because m is always much smaller than n (m never exceeds $\frac{1}{2}\sqrt{n}$), fast inquiries to the databases with image features (image convex hulls) mean an evident advantage of our approach over, for example, direct image cross-correlation whose time complexity is always $O(n)$.

This paper is organised in 6 sections. Section 2 describes the convex layer construction algorithm on a grid of points. This procedure has been made very efficient and speeds up the near-duplicate image detection presented in Section 3. Section 4 estimates the computational complexity of the proposed algorithm. Section 5 interprets the experimental results, while Section 6 concludes the paper.

2. Convex Layers

Various algorithms for building the convex layers have been developed (Preparata and Shamos, 1985) with different computational complexities. Nevertheless, being faced with vast image databases to be retrieved for duplication, we propose a new simple but very efficient solution. Each pixel of a binary image is transformed into a geometric feature. Each black pixel at position (i, j) is characterised by its coordinates and plotted as a point in a $p \times q$ rectangular area, where p and q define the image size. All pixel positions are integers. Consequently, all points are placed in a grid with unit element distance.

Convex layers, nested convex hulls or onion peeling are the terms used in computational geometry (Preparata and Shamos, 1985). Let $S = \{p_0, p_1, \ldots, p_{n-1}\}$ be a set of n points in a plane. The set of convex layers is a set of nested convex hulls defined on S. Each layer is obtained by computing the convex hull of the current points in S and removing its vertices from S. Therefore, the problem of convex layers is an extension of the convex hull problem. By definition, convex layers can be constructed by recursive convex hull algorithms. A brute force approach uses the Graham scan (Graham, 1972) or Quick Hull (Preparata and Shamos, 1985) algorithm. Their computational complexity in the worst case is $O(n^2 \log(n))$, where n stands for the number of points on the plane. The original algorithm by Jarvis (1973) was developed for constructing the convex layers, but was later described as a convex hull algorithm. Andrew (1979) described an algorithm similar to Graham scan for constructing the convex hulls with linear computational time, if points are pre-sorted according to one of the coordinates. Computational complexity of both recursive implementations is reduced to $O(n^2)$. An optimal algorithm for constructing the convex layers with $O(n \log(n))$ time complexity was introduced by Chazelle (1985). As described by the author, his algorithm expects all points to be non-collinear.

All the above-mentioned algorithms are general and work best on uniformly distributed points. In our case all the points lie on a grid (example in Fig. 1) and their co-ordinates are integers. This property is an important advantage for constructing the convex layers with linear computational complexity.

2.1. *Convex-Layer Construction*

All points assigned to a grid in the co-ordinate system describe the image objects. Their geometric arrangement is used to characterise the objects and, consequently, the image. For this purpose, nested convex layers are constructed on the set of points belonging to the individual images.

To complete the convex-layer computation in linear time, an appropriate data structure has to be considered. All vertices of an individual row are connected into a doubly linked list. The first (resp. last) vertex is also linked to the first (resp. last) vertex of the neighbouring upper and lower rows. The described double connections of vertices make them possible to be tested against, and assigned to, potential convex hulls in constant time. We took advantage of this fact and developed a novel convex-layer construction algorithm. Some preliminary results on near-duplicate image searching have been published in two conference papers (Sinjur and Zazula, 2006; Sinjur and Zazula, 2008).

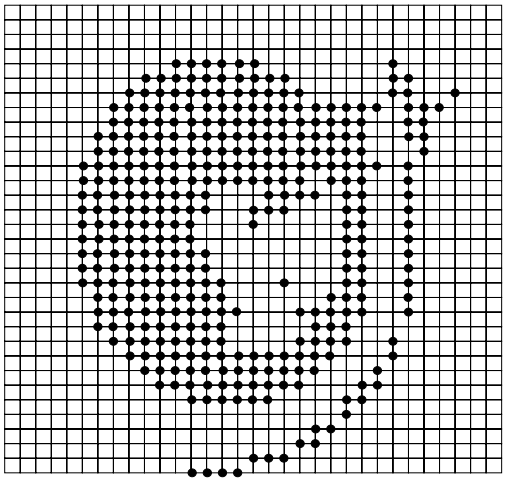


Fig. 1. Points on a grid corresponding to the image in Fig. 4(b).

Linked lists are generated directly from binary images, where each image row represents one doubly linked list in the data structure. Dynamic lists determine only a part of convex hulls at their tops and bottoms. Additional left- and right-hand side parts of the hulls have to be constructed. After the first, i.e., the outmost convex hull, is formed, the data structure has to be altered appropriately, in order to generate recursively the inner convex layers.

The convex layer construction starts with generating the most outer convex hull. The convex-hull candidate vertices are the most left and right vertices (first and last, respectively) of dynamically linked lists. Vertices, which belong to the most upper and lower lists, automatically form a part of convex hull. Left and right vertices, however, form two monotone chains (Preparata and Shamos, 1985). From each of them, the missing left and right parts of convex hull are produced. This step is analogous to the Graham's (1972) or Andrew's scan (1979). The leftmost vertices in linked lists form monotone chains. Denote the top two vertices of temporary convex hull by vertices V_i and V_{i+1}, and the leftmost vertex of kth linked list by V_k. The algorithm begins with V_1 and V_2, which stand for the leftmost vertices from the first and the second linked list. Every step for $k > 2$ verifies the following iteration: until vertices V_i, V_{i+1}, V_k form a concave hull, V_{i+1} is removed from current convex hull and i is decremented. In each step, after removing concave vertices, the leftmost vertex V_k is added to currently constructed hull and i is incremented. Same method is used to form right side of the convex hull, where the rightmost vertices of all linked lists are used. All four parts of convex hulls are finally concatenated, and all the vertices included in a hull are removed from double dynamically linked lists.

Figure 2 shows the vertices and their convex layers as obtained from the example shown in Fig. 1. Our example consists of 11 convex layers. In general, the number of convex hulls is at most $\lceil \frac{\min(p,q)}{2} \rceil$, where p and q are dimensions of the grid.

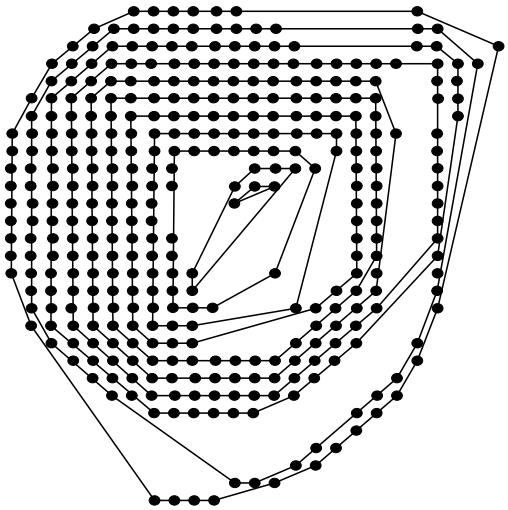


Fig. 2. Convex layers generated from the points in Fig. 1.

3. Near-Duplicate Search Algorithm

The definition of near-duplicate images has been previously provided in several articles, for example by Zhang and Chang (2004).

DEFINITION 1. Near-duplicate images is a pair of images in which one is close to the exact duplicate of the other, but differs slightly due to variations of capturing conditions (camera, camera parameter, view angle, etc), acquisition time, rendering conditions, or editing operations.

Our matching algorithm tends to characterise objects in an image by transforming the regions they cover into the sets of convex layers. These are considered image features to be compared for an image in order to decide whether we deal with very closely related images. However, no special image segmentation is necessary, while our algorithm only follows the convex layers whose features uniquely encompass the image contents. The images are binarized and their content is characterized by black pixels. Pixels are then collected into convex layers by our algorithm described in Section 2. The similarity of two images is measured by comparing the corresponding convex layers. The two-dimensional convex layers are transformed into one-dimensional features whose correlation defines the similarity of the compared images.

To clearly illustrate the basic operation of our algorithm, two simple images are used (Fig. 3). Section 5 exemplifies the results on near-duplicate image detection in a few more complex images from the Star Wreck movie. Corresponding features are depicted. This movie is also used for various experiments in Section 5.

(a) (b)

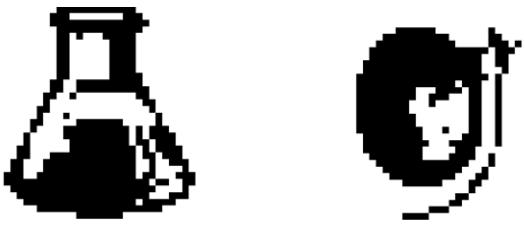
Fig. 3. Example images.

3.1. *Preprocessing*

To be able to construct the convex layers from the content of an image, a suitable preprocessing has to be performed. Two images being compared for duplication can be of different dimensions and have to be initially scaled to equal sizes. If the size of the first image is $p_1 \times q_1$ and of the second $p_2 \times q_2$, then the dimensions of both are scaled to $\min(p_1, p_2) \times \min(q_1, q_2)$ (Muñoz *et al.*, 2001).

An input image, regardless of the colour depth, is transformed to a grey-level image. To binarize an image a suitable threshold method follows the example (Otsu, 1979). The computed threshold minimizes the intraclass variance of black and white pixels. Characteristic features of an image are expected to appear in black and the background in white (see the example in Fig. 4). We additionally test the central area of the image, where objects should appear in black according to our expectations. A further check is performed on the image margins, where the background is expected in white colour. If the central area and margins of the image do not comply with expectations, the binarized image is inverted.

The pixels for Fig. 4(b), as transformed into the grid points, are depicted in Fig. 1. Figure 5 shows the constructed convex layers based on the image from Fig. 4(a).



(a) (b)

Fig. 4. Binarized images from Fig. 3.

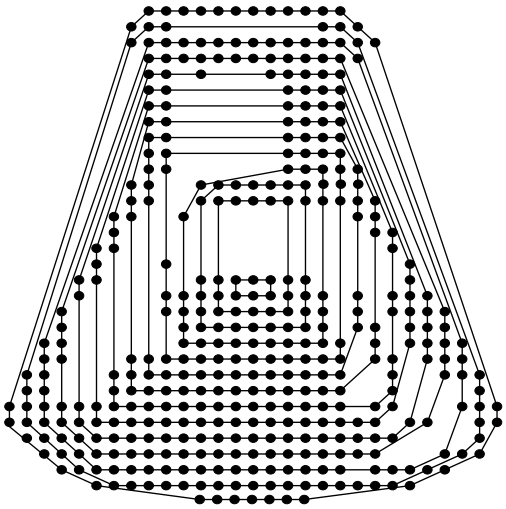


Fig. 5. Convex layers for the image in Fig. 4(a).

3.2. *Feature Extraction*

After preprocessing, the image features are constructed by the algorithm described in Section 2. Because a direct comparison of the two-dimensional convex layers is rather computationally complex, a reduction of feature dimension is performed first. There are several options to reduce the feature dimensionality, such as counting the number of convex hulls, length of each convex hull, number of vertices a convex hull contains, etc.

Most simple would be to consider the number of nested convex hulls. It is always in range $[0, \lceil \min(p, q)/2 \rceil]$, where p and q are dimensions of the rescaled images. Since many different images have the same number of nested convex hulls, this measure is used more as a measure of dissimilarity than similarity. We can assume that when the difference in the number of convex layers of two images exceeds a threshold, images are not duplicated. If the difference is close to zero, further tests for duplication have to be performed.

If the lengths of convex hulls, or the number of vertices on convex hulls, are considered, they are observed versus the sequential hull numbers. The lengths of convex hulls monotonically decrease. However, the number of vertices in the hulls of very similar lengths can be very different.

It has become clear that the most discriminate information is given by the number of vertices (pixels) on convex hulls. Therefore, we introduced a feature vector whose elements correspond to the number of vertices on the consecutive convex layers. To test two images for duplication, the distance between the corresponding feature vectors is calculated.

Figure 6 shows the feature vector obtained from convex layers in Fig. 5. There are 33 vertices on the first convex hull, 37 on the second, etc. In total, we have 14 convex hulls.

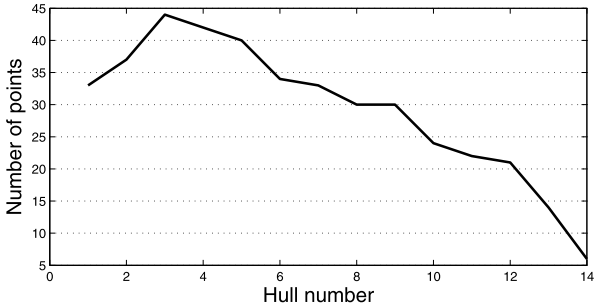


Fig. 6. Feature vector for convex layers in Fig. 5: the number of vertices versus layers

3.3. *Feature Matching*

We define a new image similarity measure by the correlation between two compared feature vectors. Let $\mathbf{x} = [x_1, x_2, \ldots, x_p]$ and $\mathbf{y} = [y_1, y_2, \ldots, y_q]$ be the two feature vectors obtained for two images. In general, the lengths of vectors $\mathbf{x}$ and $\mathbf{y}$ are different. The vector of shorter length is padded by l zeros, where $l = |p - q|$. New vector length is $L = \max(p, q)$.

The correlation coefficient of vectors $\mathbf{x}$ and $\mathbf{y}$ is computed as:

$$d(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{L}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{L}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{L}(y_i - \bar{y})^2}}, \tag{1}$$

where $\bar{x}$ and $\bar{y}$ stand for the mean values of $\mathbf{x}$ and $\mathbf{y}$.

Unfortunately, (1) hides an unwanted property: because the image feature vectors show a decreasing tendency, like the one in Fig. 6, the differences between the feature means, $\bar{x}$ and $\bar{y}$, and the tendency regression line surpass by far the differences which carry the information on the image local properties. This means that the computed correlation coefficients lose their resolution; there is only a narrow gap between those of different and those of similar images.

To solve the problem, the regression line is computed and subtracted from each feature vector prior to the comparison. Consequently, a number of points on a hull in the detrended feature vector can become negative. Let us illustrate this step for vector $\mathbf{x}$, which is converted to $\mathbf{x}'$:

$$\mathbf{x}' = \mathbf{x} - \mathbf{t}, \tag{2}$$

where $\mathbf{t} = [t_1, t_2, \ldots, t_L]$ and t_i is defined as

$$t_i = \frac{L + 1 - i}{L} \cdot b, \tag{3}$$

where L stands for the corrected length of $\mathbf{x}$ and b for the regression coefficient.

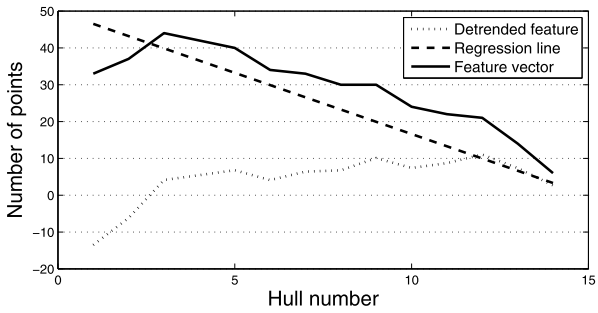


Fig. 7. An example of the feature vector detrending for the image in Fig. 4(a).

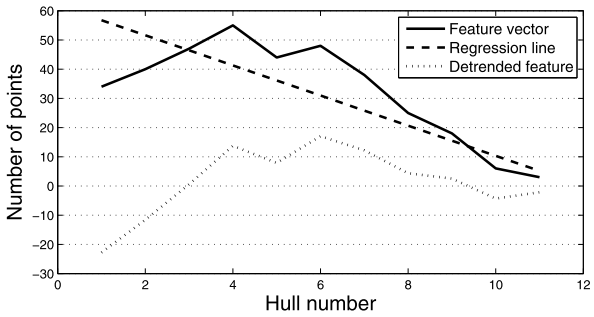


Fig. 8. An example of the feature vector detrending for the image in Fig. 4(b).

The regression coefficient b is calculated as follows:

$$b = \frac{\sum_{i=1}^{L}(x_i - \bar{x}) \cdot (i - \frac{L+1}{2})}{\sum_{i=1}^{L}(i - \frac{L+1}{2})^2}. \tag{4}$$

Figure 7 shows three graphs: feature vector $\mathbf{x}$ for the image depicted in Fig. 4(a), regression line $\mathbf{t}$ according to (3), and the detrended feature vector $\mathbf{x}'$. Analogously, Figure 8 shows three similar graphs in conjunction with Fig. 4(b). To gain the same feature vector length, the shorter one in Fig. 8 is padded by 3 zeros.

Correlation coefficient for the feature vectors in Figs. 7 and 8 yields 0.529. This value is considered as the measure of similarity between the corresponding images in Figs. 3(a) and 3(b). In our tests, the images are considered a duplicate only if their correlation coefficient is above 0.9 (Bewick *et al.*, 2003). For the images from Fig. 3, nonduplication has been established, which is certainly obvious.

4. The Algorithm's Computational Complexity

In this section we show that the overall computational complexity of the described algorithm is linear. It is obvious for all steps, except perhaps for convex layer construction,

that the computation takes linear time. Let's summarise the computational complexity of convex-layer construction in three lemmas.

Theorem 1. *The construction of a double dynamically linked lists based on the image frame with dimensions $p \cdot q$ and at most n pixels comprising the image content, where $n = p \cdot q$, has linear computational complexity, $O(n)$.*

Proof. To generate a dynamic list, each of the q image rows has to be processed once. Because a binary image has at most p black pixels in any row, the maximum length of this list is at most p. All first and last vertices can also be linked with the first and last vertices of the previous and next list, respectively, in constant time. Since we have at most q linked lists, the overall time for this operation is $T = p \cdot q + 2 \cdot q \Rightarrow O(n)$, where $n = p \cdot q$. □

Theorem 2. *Maximum number of convex hulls inside a convex layer is $\lceil \min(p, q)/2 \rceil$.*

Proof. The most outer convex hull contains vertices from the most upper and lower linked lists. Both lists are removed before the next convex hull is generated recursively. Because q dynamic lists exist, they can contribute to at most $\lceil q/2 \rceil$ convex hulls. Now, consider a creation of the same data structure, where the lists comprise the column pixels. In this case, at most $\lceil p/2 \rceil$ convex hulls can be created. □

Theorem 3. *Computational complexity for the construction of convex layers is $O(n)$.*

Proof. In general, the proposed data structure (the number of rows in Fig. 2) has at most q dynamic lists. Most left (resp. right) vertices of dynamic lists are candidates for the left (resp. right) part of the convex hull. Vertices form a monotone chain. The algorithm for computing a convex hull from a monotone chain takes at most $2 \cdot q$ steps (Preparata and Shamos, 1985; Graham, 1972; Andrew, 1979). Upper (resp. lower) part of the convex hull corresponds to the most upper (resp. lower) dynamic list. By concatenating all four parts, a convex hull is formed. In Theorem 2, it was proven that $\lceil q/2 \rceil$ convex hulls can appear for an image, which means computational complexity of $T = 2 \cdot q \cdot \lceil \min(p, q)/2 \rceil \leqslant n$, where n is always equal to $p \cdot q$. □

5. Experimental Results

We experimented with a large database of images. To create the database, 154891 frames were extracted from the film entitled Star Wreck: In the Pirkinning (Star wreck, 2010) published under (Creative commons license, 2010). All images were of the same bitmap format with resolution 640×272. Although the film itself occupies 541 MB, the storage needed for separate images in best JPEG quality increased to 10.6 GB.

On the other hand, the database of the convex-hull feature vectors for all images was remarkable smaller. For each image, the convex layers were computed and their feature

(a)

(b)

(c)

Fig. 9. A set of three example images from Star Wreck.

vectors stored in the database. The size of this database was 68 MB, which resulted in only 0.63% of image storage space needed otherwise. The reading and transformation to binary images and the correlation coefficient computation were written in Matlab, while the feature extraction was written in the C programming language. The transformation of all images to their feature vectors took 3528 seconds, or 23 milliseconds per image, on average, running a Dual Core Pentium processor with system clock 1.8 GHz and 2 GB of memory.

Two types of experiments were conducted. Firstly, we were interested in computational complexity when all duplicate images from our feature database referring to the same reference image were looked for. One thousand random feature vectors were chosen from the database as separate references. The time needed to find all near-duplicated feature vectors (images) for all randomly chosen feature vectors totalled in 279 seconds on average, with standard deviation of 17 seconds. Therefore, by considering the total number of tests taken, our correlation test for an individual pair took 1.8 microseconds. The average number of 193 near-duplicated vectors per a reference feature vector was determined.

Figure 9 shows three images from the Star Wreck movie. Fig. 9(a) is considered a reference image in a search for all near-duplicated images. Figure 9(b) is an image that was recognized near-duplicate, while the image from Fig. 9(c) was labelled as nonduplicate.

Figure 10 depicts three detrended feature vectors for images from Figs. 9(a)–9(c). While the decision based on correlation coefficients between vectors for Figs. 9(a) and 9(b) is similar and therefore duplicate, the vector for Fig. 9(c) differs from others and is recognized not similar.

The near-duplication tests were repeated in different situations after translations, rotations, and scaling, with additive noise, and after lossy compression. In all our experi-

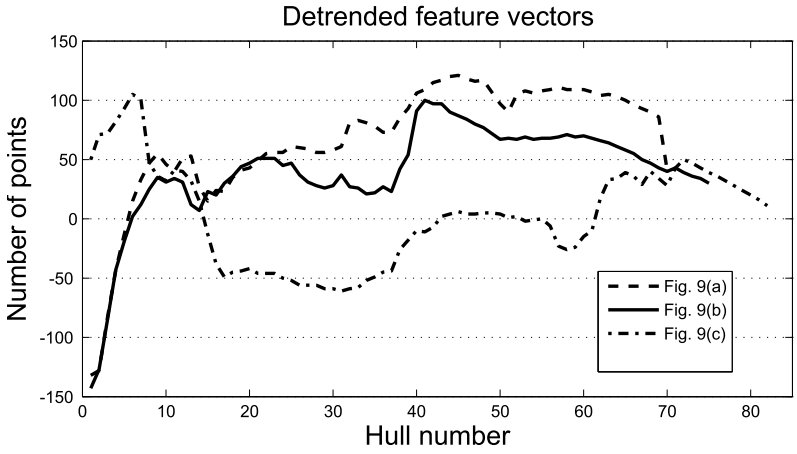


Fig. 10. Detrended feature vectors for images on Figs. 9(a)–9(c).

ments, if not otherwise noticed, the duplication test was considered positive if the correlation coefficient for feature vectors exceeded, or was equal to, 0.9.

For each test two classes of images were considered: those near-duplicate and those nonduplicate to the reference image. Statistical significance of the obtained results was determined by the ratio of recognized near-duplicate images versus all near duplicate images (sensitivity) and the ratio of the images classified nonduplicate versus all nonduplicate images (specificity).

5.1. *Efficiency of Near-Duplicate Detection on Translated, Rotated, and Scaled Images*

Our first search was constructed to establish specificity for a selected reference feature vector in our film database. One thousand feature vectors were randomly selected from the database. Each reference was compared to all database images, i.e., feature vectors, except those appearing close to the reference one. A neighbourhood of 600 frames, i.e., 24 seconds of film at 25 frames/s, centred at every reference image was considered near-duplicate and excluded from this experiment. Our search found 97.61% of images different, on average (Table 1).

At the same time, sensitivity tests were performed in the neighbourhoods of 600 frames centred at the selected reference image. In this case, only 10 random images were chosen, since we had to check all the images reported as duplicate visually. Our verification confirmed that 96.68% of near-duplicated images were recognized near-duplicated, on average (Table 1).

The second experiment was performed with translated, rotated, and scaled image contents. One hundred images were randomly chosen from the database and transformed. For the translation tests, the content of chosen images was randomly translated within the image dimensions, with no content changes. The influence of rotation was tested on images rotated by 6 angles, $i\frac{\pi}{6};\ i = 1, 2, 3, 4, 5, 6$. Each test with scaling applied a different pair

Table 1

Near-duplicate test performance: sensitivity and specificity were computed on 10 and 1000 randomly selected images, respectively, while translations, rotations, and scaling were tested for 100 randomly chosen images

	Statistical index			
	Sensitivity		Specificity	
	Mean	Standard deviation	Mean	Standard deviation
Original images	0.9668	0.0032	0.9761	0.0209
Translation	0.9581	0.0065	0.9662	0.0165
Rotation	0.6379	0.0298	0.9504	0.0444
Scaling	0.9663	0.0083	0.9652	0.0213

of random scaling factors from the interval [0.2, 2.0], one along the x-axis and another along the y-axis.

As expected, the translation test gave results similar to the test on original images. The sensitivity in the rotation test, however, shows only 63% of matching image pairs as near-duplicate. The rotation interpolates the image pixel values, which may change the outcome of binarization. Therefore, if the angle of rotation is $i\frac{\pi}{2};\ i = 1, 2, \ldots$, the interpolation does not occur and the results for sensitivity and specificity are not affected. When rescaling an image, the interpolation is also performed. If the same rescaling algorithm is used for both compared images the influence of interpolation is minimized. The comparison results are shown in Table 1.

5.2. *Noise Influence*

The robustness of our duplicate search test was checked on images with additive zero-mean Gaussian noise. The noise was added 10 times to each of 100 randomly chosen images with signal-to-noise ratios (SNR) from 50 dB to 0 dB.

Figures 11 and 12 depict the sensitivity and specificity versus SNR. While the sensitivity decreases significantly with the increasing level of Gaussian noise, the specificity is quite immune to additive noise. This means that even if the recognition rate drops very low, the positive duplicate results can be considered highly confident.

5.3. *Influence of the Lossy Compression on Image Near-Duplicate Checks*

Image compression plays an indispensable role in most today's communication and data exchange standards. Therefore we also verified how lossy compression influences our algorithm and the image comparison results. Ten percent of images from our database in their original form were randomly chosen and compressed using different JPEG compression levels (Wallace, 1992). Lossy compression was used, so that after the decompression the images were degraded to a certain extent. The convex-layer feature vectors were computed for these images and, afterwards, compared with the feature vectors of the originals (taken from the database).

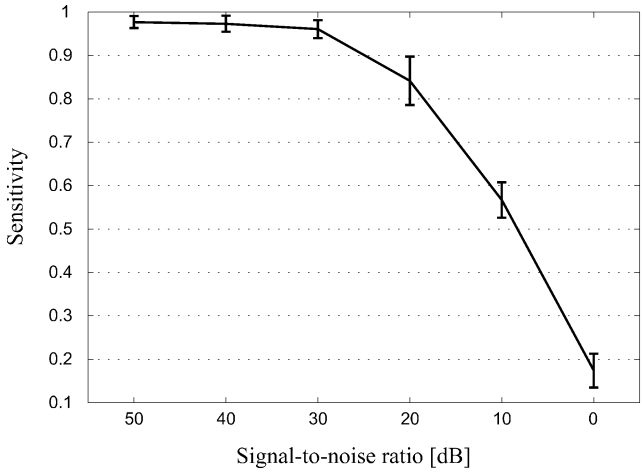


Fig. 11. Influence of noise on the algorithm's sensitivity: 100 images were included in 10 Monte-Carlo simulation runs per each SNR

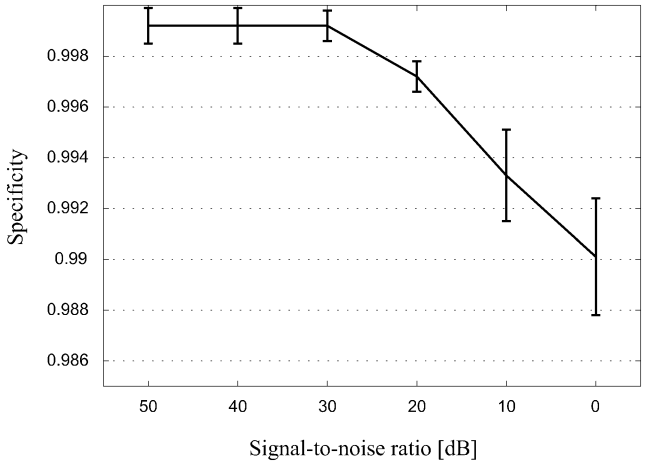


Fig. 12. Influence of noise on the algorithm's specificity: 100 images were included in 10 Monte-Carlo simulation runs per each SNR.

The applied compression levels were 100%, 90%, ..., 0%, where the smallest value means good compression ratio and bad image quality, while the largest value means the opposite. Figure 13 shows decreasing sensitivity of our algorithm when the image quality decreases. The highest compression level pushes the sensitivity down to 74% and it stays above 90% for compression levels greater or equal to 10. Specificity is more insensitive to compression levels (Fig. 14). It remains above 99%, even with the images of inferior quality.

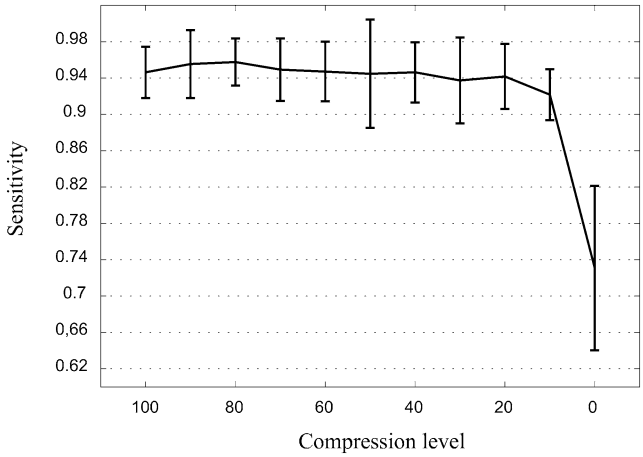


Fig. 13. Sensitivity versus compression level: 10% of randomly chosen images were compressed and decompressed with levels from 100 (best quality) down to 0 (worst quality) in steps of 10.

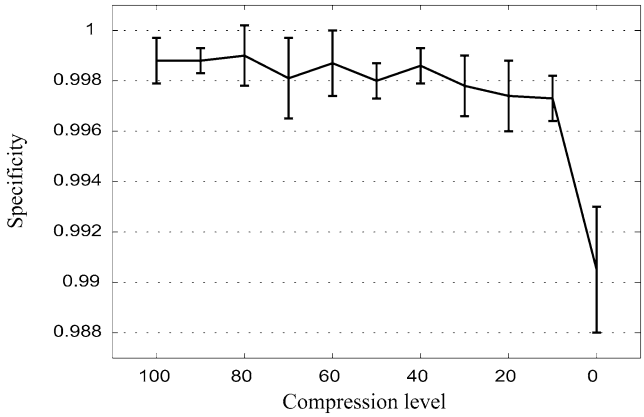


Fig. 14. Specificity versus compression level: 10% of randomly chosen images were compressed and decompressed with levels from 100 (best quality) down to 0 (worst quality) in steps of 10.

5.4. *Comparison Results*

We compared our proposed method for image feature extraction to four published methods:

- The method derived in Lowe (2004) describes Scale-Invariant Feature Transform (SIFT). SIFT features are local features that belong to the points of interest on an image object.
- Paper (Bay *et al.*, 2008) introduces the Speeded Up Robust Feature (SURF) approach, whose features are inspired by SIFT and mean an approximation of a sum of Haar wavelet coefficients. Their extraction is several times faster than extraction of SIFT features and they are more independent of image transformations.

- The method in Thomee *et al.* (2008) suggests feature generation based on hierarchical averaging.
- Another SIFT-based feature extraction algorithm is explained in Chum *et al.* (2008).

We downloaded the experimental database with 250 different desktop images from (Blirk Images, 2011). All original images are of size 800×600. Twenty subsets of 20 randomly chosen images were generated from the database. For each image of any subset 8 different transformations were applied 5 times with random selection of parameters, dependent on the transformation applied:

- Blur – pixel values were changed by a Gaussian local operator with randomly chosen standard deviation (σ) in $[1, 5]$.
- Median – pixel values were replaced by median pixel value in the local area of radius randomly chosen in $[1, 5]$.
- Oil painting effect – pixel is replaced by the most frequent colour in a circular neighbourhood of random radius $[1, 5]$.
- JPEG compression – quality percentage was randomly decided from interval $[30, 100]$.
- Crop – each edge is removed by a random quantity from 1 to 10% of original image size.
- Scale – image size was changed by a random percent from interval $[50, 250]$.
- Rotation – image is rotated by a random angle from interval $[0, 2\pi]$.
- Rotation with crop – same as above, except that image size after rotation was cropped to its original size.

All cited algorithms were implemented in the C programming language and executed on a computer with Pentium i2600K 3.4 GHz CPU and 8 GB DDR3 RAM, running Linux. Also in this experiment we observed sensitivity, specificity, and computational time needed for feature extraction (Table 2).

Table 2 shows comparison results for $20 \times 8 \times 5 = 800$ (number of subsets $\times$ number of transformations $\times$ number of repetitions) simulation runs. Means and standard deviations for sensitivity and specificity were calculated on a subset basis. All tested algorithms

Table 2
Comparison results: specificity, sensitivity and feature extraction time

	Sensitivity		Specificity		Time[s]	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
Thomee *et al.* (2008)	0.71	0.02	0.81	0.03	0.052	0.007
Chum *et al.* (2008)	0.68	0.03	0.90	0.03	0.462	0.023
Bay *et al.* (2008)	0.72	0.06	0.83	0.04	0.295	0.042
Lowe (2004)	0.74	0.04	0.85	0.03	3.731	0.081
Proposed method	0.73	0.02	0.82	0.03	0.030	0.004

yield similar sensitivity, only Chum *et al.* (2008) is slightly lower. Also the specificity of all algorithms is very similar, only the results of Chum *et al.* (2008) are slightly higher. The main difference appears in the last table column. Our algorithm is the fastest when dealing with feature extraction, which is its most important advantage over all other algorithms. It is almost twice as fast as the second best algorithm, (Thomee *et al.*, 2008), and it outperforms the SIFT-based algorithm described in Lowe (2004) by more than 100 times.

6. Conclusion

In this paper, a novel computationally efficient algorithm for image matching was presented. The method determines a set of convex hulls for every image to be compared. These two-dimensional features can be calculated with linear computational complexity. For the comparison reasons, they are transformed into feature vectors whose elements contain the number of vertices in each convex hull. To be able to detect all subtle image variations, the feature vectors must have their regression line subtracted prior to the test of image similarity.

By the described approach, all duplicated or nonduplicated images from a database of images can be extracted if the assumption of structurally and contextually similar images is respected. This means that images with only some similar objects, or similar objects at different image co-ordinates, are assumed to be nondplicated. However, there are numerous situations where this property proves beneficial, in particular because the proposed algorithm is very fast.

Experimental results show that duplicated images can be found even if they are translated or scaled. Our algorithm also works perfectly for rotated images if the rotation does not evidently change the images by interpolation. This happens when the angle of rotation equals multiples of 90 degrees. Considering no image preprocessing and noise suppression, additive noise with SNRs of down to 30 dB does not disturb the construction of convex layers and successful image matching. Our convex-layer-based image matching approach is also insensitive to lossy compression even in worst quality.

References

Andrew, A.M. (1979). Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5), 216–219.

Bay, H., Ess, A., Tuytelaars, T., Van Gool, L. (2008). SURF: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3), 346–359.

Bewick, V., Cheek, L., Ball, J. (2003). Statistics review 7: correlation and regression. *Critical Care*, 7(6), 451–459.

Bing Image Search (2011). Online available: `http://images.bing.com/`.

Chazelle, B. (1985). On the convex layers of a planar set. *IEEE Transactions on Information Theory*, 31(4), 509–517.

Chum, O., Philbin J., Zisserman, A. (2008). Near duplicate image detection: min-hash and tf-idf weighting. In: *Proceedings of the British Machine Vision Conference*, pp. 493–502.

Creative Commons License (2010). Online available: `http://creativecommons.org`.

Desktop Wallpapers, Free Desktop Background, Free Stock Photos and Images (2011) . Online available: `http://www.blirk.net/`.

Google Image Search (2011). Online available: `http://images.google.com/`.

Graham, R.L. (1972). An efficient algorithm for determining the convex hull of a finite planar set. *Informayion Processing Letters*, 1(4), 132–133.

Grauman, K., Darrell, T. (2005). Efficient image matching with distributions of local invariant features. In: *In CVPR (2)*, IEEE Computer Society, pp. 627–634.

Hannah, M.J. (1989). A system for digital stereo image matching. *Photogrammetric Engineering Remote Sensing*, 55(12), 1765–1770 .

Heipke, C. (1996). Overview of image matching techniques. *OEEPE Workshop on the Application of Digital Photogrammetric Workstations*, 33, 173–189.

Hobrough, G.L. (1959). Automatic stereo plotting. *Photogrammetric Engineering Remote Sensing*, 25(5), 763–769.

Incogna Image Search (2011). Online available: `http://www.incogna.com/`.

Jarvis, R.A. (1973). On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1), 18–21.

Lowe, D.G. (2004) . Distinctive image features from scale-invariant keypoints. *International Journal of Computing Vision*, 60(2), 91–110.

Muñoz, A., Blu, T., Unser, M. (2001). Least-squares image resizing using finite differences. *IEEE Transactions on Image Processing*, 10(9), 1365–1378.

Olson, C.F. (2002). Maximum-likelihood image matching. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 24(6), 853–857.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 62–66.

Preparata, F. P., Shamos, M. I. (1985). *Computational Geometry – An Introduction*. Springer, New York.

Šinjur, S., Zazula, D.(2006). A novel approach to image matching using convex layers. *WSEAS Transactions on Information Science and Applications*, 3(12), 2381–2385 .

Šinjur, S., Zazula, D. (2008). Image similarity search in large databases using a fast machine learning approach. In: *First International Symposium on Intelligent Interactive Multimedia Systems and Services (KES-IIMSS 2008)*, Springer, Berlin, pp. 85–93.

Star Wreck: In the Pirkinning (2010). Online available: `http://www.starwreck.com`.

Tiltomo Image Search (2011). Online available: `http://www.tiltomo.com/`.

Thomee, B., Huiskes, M.J., Bakker, E., Lew, M.S. (2008). Large scale image copy detection evaluation. In: *Proceeding of the 1st ACM International Conference on Multimedia Information Retrieval (MIR 2008)*, ACM, New York, pp. 59–66.

Viitanierni, V., Laaksonen, J. (2005). Keyword-detection approach to automatic image annotation. *Integration of Knowledge, Semantics and Digital Media Technology*, EWIMT 2005, 15–22.

Wallace, G.K. (1992). The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1), 28–34.

Wang, Z., Josephson, W., Lv, Q., Charikar, M., Li, K. (2007). Filtering image spam with near-duplicate detection. In: *Proceedings of the 4th Conference on Email and Anti-Spam*.

Yang, M., Qiu, G., Huang, J., Elliman, D. (2006). Near-duplicate image recognition and content-based image retrieval using adaptive hierarchical geometric centroids. In: *18th International Conference on Pattern Recognition*, pp. 958–961.

Zhang, D., Chang, S. (2004). Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In: *ACM Conference of Multimedia*, pp. 877–884.

S. Šinjur was born in 1976. He received the BSc degree in computer science from the University of Maribor, Slovenia in 2002. He is currently working as a teaching assistant at Faculty of Electrical Engineering and Computer Science at University of Maribor. His research interests include computational geometry and image processing, especially near-duplicate image and video detection.

D. Zazula graduated from the University of Ljubljana, Slovenia, where he also received his MSc and a PhD degrees in signal processing. He has worked in industrial research for twelve years and in 1987 joined the Faculty of Technical Sciences, University of Maribor. He is currently holding a position of a full professor lecturing on system software, computer system performance, operating systems, and digital signal processing. His main research interests are digital signal and image processing, especially in biomedical applications.

B. Žalik is a professor of computer science at University of Maribor, Slovenia. He obtained PhD in computer science in 1993 from the University of Maribor, Slovenia. He is the head of Laboratory for Geometric Modelling and Multimedia Algorithms. His research interests include computational geometry, geometric data compression, scientific visualization and geographic information systems. He is an author or co-author of more than 60 journal and 100 conference papers.

Greitas iškiliais sluoksniai grįstas algoritmas panašių vaizdų atpažinimui

Smiljan ŠINJUR, Damjan ZAZULA, Borut ŽALIK

Šiame straipsnyje pristatomas naujas, greitas algoritmas gebantis generuoti iškiliuosius sluoksnius ant pateikto tinklelio. Iškilūs sluoksniai yra gaunami iš juodai balto vaizdo. Apskaičiuoti iškilieji kontūrai yra aprašomi vektoriais ir vėliau yra naudojami kaip žymenys nusakantys vaizdo požymius. Atsižvelgiant į šiuos požymius yra skaičiuojamas geometrinis vaizdų panašumo matas. Požymių panašumas vertinamas atsižvelgiant į požymių vektorių koreliacijos koeficientus. Tos būdas leidžia vaizdus su panašiomis objektų struktūromis ir turiniu efektyviai aptikti didelėse duomenų bazėse. Pateikiamas algoritmas gali būti pritaikytas sprendžiant vaizdų sutapdinimo, dublikatų aptikimo, video stebėjimo klausimus. Algoritmas yra nejautrus esant vidutiniam signalo atsako į triukšmą lygiui, gali būti naudojamas su suspaustais vaizdais. Algoritmui įtakos neturi vaizdų pasukimas, perkėlimas ir mastelio pakeitimas.