Blind Logo and Identigram Exclusion for Image Matching

Krisztián Nagy and László Szirmay-Kalos[1]

Budapest University of Technology and Economics, Hungary
`szirmay@iit.bme.hu`

Abstract. To decide whether two different images partially depict the same object or the same scene, image descriptors are typically generated from the two images and these descriptors are tried to be paired. If there are sufficient number of pairs meeting also consistency requirements like homography and epipolar constraints, then we report a valid image match suggesting that the same object showed up in the two images. Such approaches are fundamental also in content based image retrieval or automatic image annotation. When this approach is used on large image databases, logos or identigrams pose problems since the standard algorithm may recognize that similar logos are present in two different images, so a false match is assumed. However, logos do not represent really similar content, but rather artifacts or signs of the authorship of the image. In this paper we examine an automatic solution of this problem. We call our proposed method blind because we usually do not know the logo in advance, so matches on logos must be recognized as an outlier without using any previous shape or feature information. What we can use instead is the typical 2D placement of logos, which is inherently different from the complex projective arrangements of 3D objects when they are photographed from two viewpoints. This paper proposes several methods to identify false matches of image comparisons due to logos, identigrams and watermarks. Having identified the keypoints on logos, they can be ignored during the matching process.

1 Introduction

Logos, identigrams or watermarks are often added to images to identify the source or the owner of the image. A logo can be of two distinct styles: overlapped, when the logo completely obscures a part of the image, and blended[7], when the content is transparently overwritten, partially preserving the underlying content. For a human observer, these logos do not pose problems because the human visual system easily identifies and filters them out. However, when these images are processed by algorithms, for example, when building large image databases or during content based retrieval, the algorithms can be misled by the features of such logos.

Let us assume a standard image matching scenario and suppose that our task is detect similar 3D objects or scenes in different images of a data base.

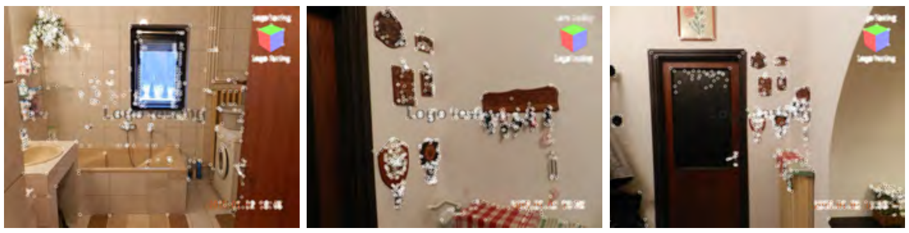

Fig. 1. Images to be matched with SIFT features generated on them.

The common approach is to extract image features, such as SIFT [2], SURF, SAFT, etc. features from both images and to find similar features in the two images (Fig. 1). If many feature pairs can be found, which also meet consistency criteria, then the similarity of object or scene can be reported. For consistency, we preserve only those keypoint pairs where the partner is the strongest match from both directions, i.e. a keypoint can participate only at most one pair, and we also check whether or not the keypoints of the two images can be mapped onto each other by modifying the extrinsic or intrinsic parameters of a camera. It means that valid pairs of feature points should satisfy the same epipolar constraint. If the same or similar logo is added to the two images, then features belonging to the logos will be present in both images and a simple homography, i.e. translation and minor scaling, will be consistent pairing, so such image pairs are reported as similar even when no common object is seen on them except for the logo.

Logos can cause both false positive and false negative matches. Two different images may be said to be similar because of logo matches. On the other hand, if the same scene is photographed from two different viewpoints, the valid matches will correspond to a different epipolar constraint than logo matches, so logo matches may invalidate real matches during the statistical check of the valid correspondences.

To make image comparison robust and appropriate also for images with logos, these logos should be identified automatically, and then the features belonging to the logos should be ignored during matching.

In this paper we propose several approaches for logo identification, which are based on the inherently 2D nature of logos. While logos are painted over an image depicting a 3D scene, 2D and 3D contents are mixed. If similar features are looked for on two images, consistently selected features of the 3D content satisfy epipolar constraint, while consistently selected features of the 2D content are related via simple translation and minor scaling.

2 Previous work

The issue of logo detection has been considered in different contexts of image processing. In [4], a set of grayscale features is extracted to construct a suite of rules for classifying the segmentation of the logos. In [6], a method to match logos based on positive and negative shape features is proposed. In [5], contour trees and line adjacent graphs are adopted for logo detection; recursive neural networks have also been applied to classify logos. The use of grayscale features prevents these methods from exploiting the unique characteristics of video logos [8]. The identified logo can be removed by image inpanting/completion [3] or simply ignored during further processing.

While the above approaches know the logo that is being searched or at least assume that the logo is present in all images, there are also blind approaches that separate the logo from the real signal by finding some discrepancy. For example, Zhu et al. [9] converts the image into wavelet domain and apply maximum likelihood estimation to decide which components are coming from a different source.

Logos can also be identified if we know that they are present in many images. In such cases, just the common part should be identified.

3 New methods

Our setup is different from what is usually attacked by the literature since we do not want to restore the original content, neither wish to eliminate logos, but want to ignore features that stem from the logos. This is a simpler task in this respect. On the other hand, unlike in other typical cases, we do not have a-priori information about the logos we are looking for. We wish to exclude false matches caused by logos if the same logo shows up at two different images.

To separate real image content from logos without knowing anything about the logo is possible by considering the inherent dimensionality behind real content and logos. While images are projections of 3D worlds, logos are two dimensional that are translated but usually neither scaled drastically nor rotated.

We propose three different approaches for logo recognition during image matching.

4 Logo filtering with area checking

The base of this method is the assumption that logos are usually placed in certain areas of a picture. These regions are the corners and the middle of the image. If the end points of a match are located in the same region on both of the pictures, the match is considered a logo match and thus it gets thrown away.

This method can yield good results without significant overhead but logos placed elsewhere or with a size bigger than the examined region won't get filtered out completely. On the other hand, good matches can be thrown out if they are located where usually the logos are placed.

5 Logo filtering using keypoint locations

Another approach is to consider the logic that the logos are usually placed with. Most of the time they are either placed at fixed distances in pixel coordinates from two sides (in a corner) of a picture or relatively along the x and y axis (e.g. in the middle). So if the distances from two sides or the relative placement are the same for a match on both pictures then it is considered a logo match and gets filtered out.

There are two problematic scenarios using this method. The first is that if the images already containing logos are scaled, then the pixel space position of the logos gets modified, so only relative positions work. On the other hand, if logos are placed after scaling the image, then the application of relative positions fails. The results of this method are shown by Figs. 2 and 3.

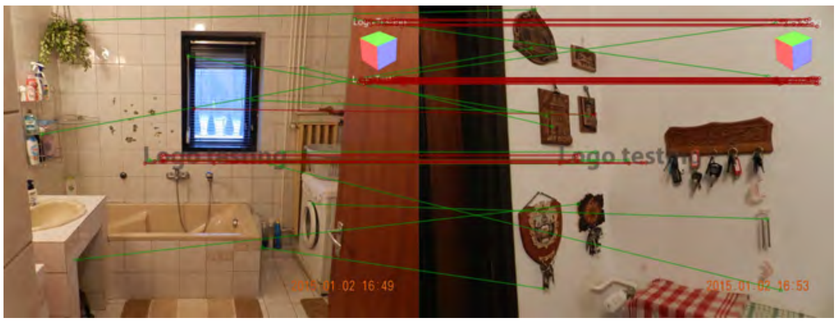

Fig. 2. Separating logo matches based on the keypoint locations for a pair of images depicting two different scenes. Logo matches are red, non-logo matches are green.

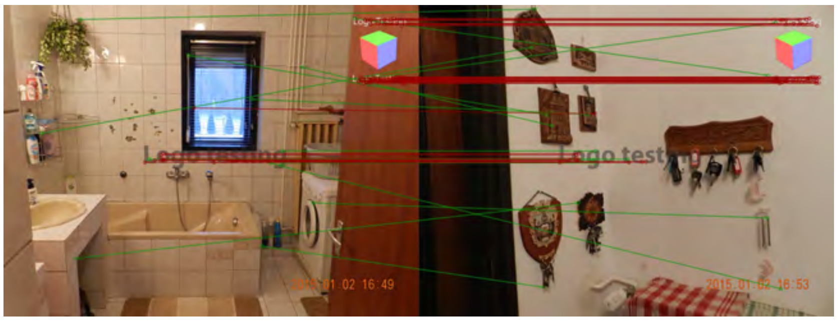

Fig. 3. Separating logo matches based on the keypoint locations for a pair of images depicting two different scenes. Logo matches are red, non-logo matches are green.

6 Logo filtering with clustering

In this method matches get placed in clusters based on the keypoint positions on the first image, translation vector between the corresponding keypoints and feature distances since we can expect that a logo is relatively small, so its keypoints are close to each other, the logo of the second image can be obtained with a single translation and a minor scaling from the logo of the first image, and the style and resolution inside a logo is relatively uniform, so features are detected on the same scale. The space of clustering is thus five dimensional where a "point" is the composition of translation $\boldsymbol{t}$ of the keypoints between the two images, location $\boldsymbol{p}$ of the keypoint in the first image, and scale difference s.

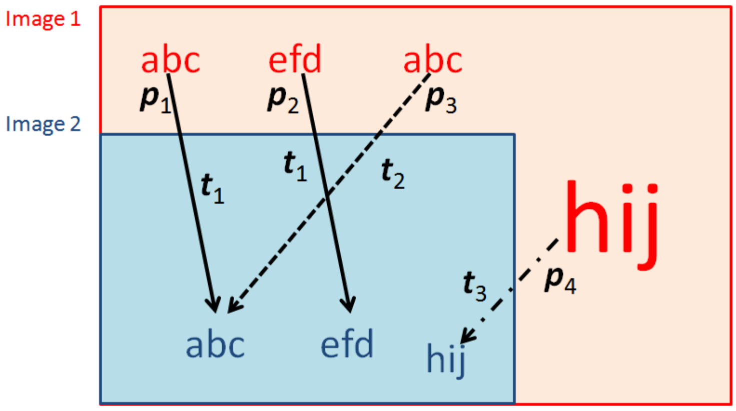

Fig. 4. Two overlayed images with similar features. The first image is depicted with red, the other with blue. Keypoints of "abc" can be found in both images, once with translation $\boldsymbol{t}_1$ once with translation $\boldsymbol{t}_2$. Keypoints of "efd" are also matched with translation $\boldsymbol{t}_1$. Finally, blue pictures's "hij" can be matched with the red picture's "hij" with translation $\boldsymbol{t}_3$. The initial 2D clustering method would use only the translation vectors, and would identify cluster 1 ($\boldsymbol{t}_1$, {abc,eft}), cluster 2 ($\boldsymbol{t}_2$, {abc}) and cluster 3 ($\boldsymbol{t}_3$, {hij}). Note that the statistical filtering of the SIFT keypoints would remove multiple keypoint usage, so one particular keypoint in the blue "abc" may belong either to cluster 1 or cluster 2. Having done the initial 2D clustering, keypoint locations and feature scale are also taken into account, resulting in the decomposition of cluster 1 into two new clusters cluster 11 ($\boldsymbol{t}_1$, $\boldsymbol{p}_1$, {abc}), cluster 12 ($\boldsymbol{t}_1$, $\boldsymbol{p}_2$, {efd}). Cluster 3 is preserved as it is and cluster 3 is deleted since the keypoints on the two images belong to very different scales.

We use a modified K-means algorithm [1] for clustering, consisting of the iteration of an assignment step and then an update step.

Initial clustering is done in 2D space using solely the match translation vector because this is a selective indicator of similarities as keypoints of similar regions (both real and logo matches) tend to have comparable translation vectors. Possible errors in placement are corrected during the refinement process. Elements are sorted using the x coordinate of these vectors and clusters are formed by successively selecting a desired number of items from the ordered array. This way the initial number and size of the clusters can be easily controlled.

Current clusters are given by the centroids obtained as the arithmetic mean of the elements assigned to this cluster. During assignment each element is paired with each centroid, and the element is assigned to that cluster from which centroid its distance is minimal. After assignment cluster centroids are recomputed.

After the refinement is done, clusters are searched for outliers (regarding endpoint positions on either or both of the images) using a robust statistical method. Two variables are used by the algorithm:

- $\boldsymbol{T}$: a running average of the translation, which is the arithmetic mean of the translations of current inlier elements.
- σ^2: a running average of the squared difference of the current translation vector $\boldsymbol{t}$ and the average translation $\boldsymbol{T}$, which is updated with the new inlier translation.

The method begins with finding the starting translation, which is the translation with the minimum summed distance from all other translations in the cluster. This element is added to an array reserved for inliers and its value is assigned to $\boldsymbol{T}$. No value is assigned to σ^2 yet. The second element will be the one with the lowest distance from the first. If there is only two elements in the cluster and their distance is above a threshold, then they are too far away to be related and the second element is separated into a new cluster. Otherwise it is added to the inliers then values of σ^2 and $\boldsymbol{T}$ are updated. This step is repeated iteratively. The next element $\boldsymbol{t}$ to evaluate is the one with the lowest difference from $\boldsymbol{T}$. This distance is then compared to σ. If it is much larger than σ, the element cannot be an inlier and thus it is placed in the outliers array along with the rest of the unchecked elements (because they are even farther from $\boldsymbol{T}$) and the algorithm stops. Otherwise, evaluated item $\boldsymbol{t}$ is placed in the inliers array, the values of σ^2 and $\boldsymbol{T}$ are updated and the next iteration starts.

The basis of outlier detection is the calculation of the probability whether the new element follows the same distribution as the already detected inliers. If the locations of points in a cluster are distributed with a Gaussian with mean $\boldsymbol{T}$ and standard deviation σ, then the probability that the distance $r = |\boldsymbol{t} - \boldsymbol{T}|$ is larger than the average distance σ of already identified inliers multiplied by confidence factor λ can be obtained as an integral of the 2D Gaussian:

$$P(r > \sigma\lambda) = \int\limits_{\phi=0}^{2\pi} \int\limits_{r=\sigma\lambda}^{\infty} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{r^2}{2\sigma^2}\right) r\mathrm{d}r\mathrm{d}\phi =$$

$$\left[-\exp\left(-\frac{r^2}{2\sigma^2}\right)\right]_{r=\sigma\lambda}^{\infty} = \exp\left(-\frac{\lambda^2}{2}\right).$$

With this equation we can convert a confidence probability into factor λ.

After every keypoint pair has been placed in either the array of inliers or outliers, outliers will form a new cluster (which can also be searched for outliers separately if deemed necessary) while inliers are kept as the new content of the searched cluster. Cluster centroids should be updated after outlier removal if they are to be used in subsequent operations. This refinement process can be repeated for increased precision.

As a post-processing step, clusters with too few elements or with a centroid too close to another centroid are removed and their elements are relocated to the next closest cluster. Then, cluster centroids are updated once more.

So far, we considered only the translation vectors between the keypoints to establish clusters. In the second phase, we further decompose clusters based on the position of the keypoints in the first image and the scale associated with the keypoints. The same clustering method is used again starting with the result of the first phase as applied for the translations.

The method of clustering presented so far can be used for logo removal but there is a scenario when the approach fails. If a logo consists of transparent letters, its background greatly affects how its keypoints are matched. For example, there can be a letter 't' in the beginning and the end of a logo that can be matched together if they are in similar contrast with their background but the same parts aren't. Or even the upper part of a 't' and 'h' can be matched. With clustering, these letter and letter part matches would end up in clusters which can be compared to each other. The translation vectors would be near identical among the elements of these clusters like it is expected with logos, so these clusters cannot be separated from the rest. On the other hand, the order of these separated clusters regarding their position on the two images would be different in such a case so they can be still identified as logos. This cannot be done reliably using matches solely without clustering because unrelated matches can occur between and around the transparent letters making the logo matches unseparable from their surroundings.

Similar images pose problems to this method as well, since in this case all pairs will belong to the same cluster. To attack this, we can use the method of Section 5.

Then we examine the clustering method discussed in Section 6. Fig. 5 shows the partial results of a single iteration step of the clustering method for a pair of images depicting two different scenes. Note that although these images depict two different scenes but due to the logos in the image center and right-bottom corner, a naive image matching algorithm would report many valid correspondences. Figs. 6 demonstrates the algorithm for a different image pair that depicts the same scene from two viewpoints.

7 Conclusions

In this paper we discussed methods to exclude false matches caused by identigrams or logos when scenes and objects are compared based on their images.

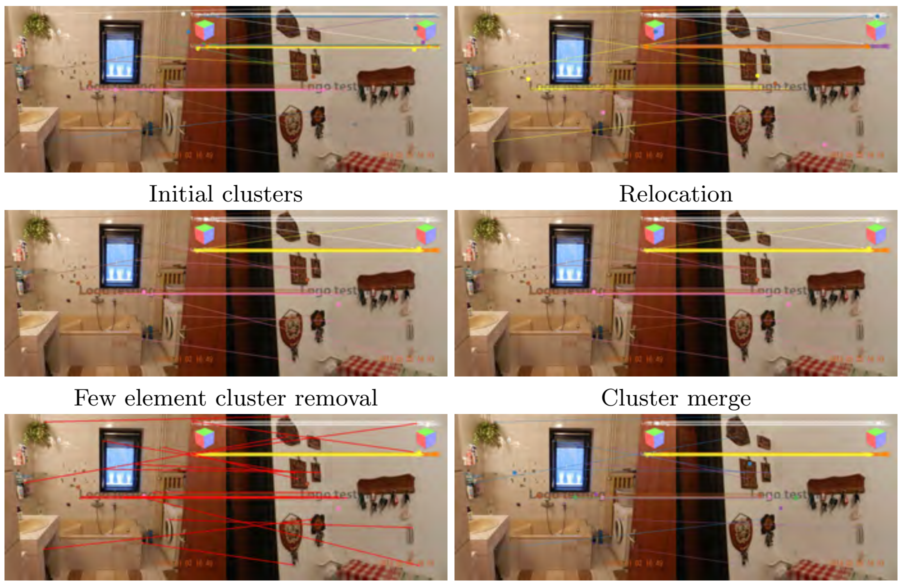

Fig. 5. Modification of clusters in a single iteration step for a pair of images depicting different scenes.

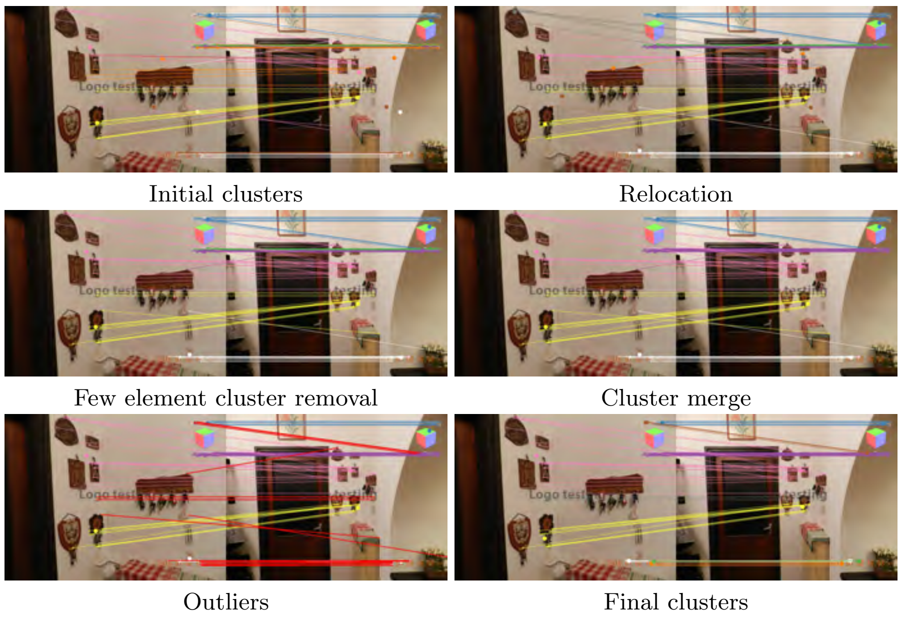

Fig. 6. Modification of clusters in a single iteration step for a pair of images depicting the same scene.

The proposed methods are implemented in OpenCV environment and built into a data mining application automatically processing images gathered from the World Wide Web. According to our experience, the clustering method robustly finds similar logos in the two images, so based on clustering results, false matching can be eliminated.

Acknowledgement

This work has been supported by OTKA K–104476.

References

1. T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Analysis and Mach. Int.*, 24(7):881–892, 2002.
2. David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.
3. Jaesik Park, Yu-Wing Tai, and In So Kweon. Identigram/watermark removal using cross-channel correlation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 446–453, June 2012.
4. S. Seiden, M. Dillencourt, and S. Irani. Logo detection in document images. In *Proceedings of the International Conference on Imag- ing Science, Systems, and Technology*, pages 446–449, 1997.
5. J. Sheng. *Graphical item recognition using neural networks.* PhD thesis, Thesis, Universita degli Studi Di Firenze, 1998.
6. A. Soffer and H. Samet. Using negative shape features for logo similarity matching. In *Proceedings of the 14th International Conference on Pattern Recognition (ICPR 1998)*, 1998.
7. László Szirmay-Kalos, György. Antal, and Ferenc Csonka. *Háromdimenziós grafika, animáció és játékfejlesztés.* ComputerBooks, Budapest, 2003.
8. Wei-Qi Yan, Jun Wang, and Mohan S. Kankanhalli. Automatic video logo detection and removal. *Multimedia Systems*, 10(5):379391, 2005.
9. Xiang-Wei Zhu and Liang Xiao. Research of blind watermark detection algorithm based on wavelet and contourlet transform domain. In *E-Business and Information System Security, 2009. EBISS '09. International Conference on*, pages 1–5, May 2009.