Fixing WTFs: Detecting Image Matches caused by Watermarks, Timestamps, and Frames in Internet Photos

Tobias Weyand Chih-Yun Tsai Bastian Leibe
Computer Vision Group, RWTH Aachen University, Germany
weyand@vision.rwth-aachen.de chi.tsai@rwth-aachen.de leibe@vision.rwth-aachen.de

Abstract

An increasing number of photos in Internet photo collections comes with watermarks, timestamps, or frames (in the following called WTFs*) embedded in the image content. In image retrieval, such* WTFs *often cause false-positive matches. In image clustering, these false-positive matches can cause clusters of different buildings to be joined into one. This harms applications like landmark recognition or large-scale structure-from-motion, which rely on clean building clusters. We propose a simple, but highly effective detector for such false-positive matches. Given a matching image pair with an estimated homography, we first determine similar regions in both images. Exploiting the fact that* WTFs *typically appear near the border, we build a spatial histogram of the similar regions and apply a binary classifier to decide whether the match is due to a* WTF*. Based on a large-scale dataset of* WTFs *we collected from Internet photo collections, we show that our approach is general enough to recognize a large variety of watermarks, timestamps, and frames, and that it is efficient enough for large-scale applications. In addition, we show that our method fixes the problems that WTFs cause in image clustering applications. The source code is publicly available*[1] *and easy to integrate into existing retrieval and clustering systems.*

1. Introduction

In recent years, the number of Internet images containing specific artifacts has been increasing dramatically (Fig. 1). One reason is a trend in smartphone camera apps to allow the user to add photo frames or worn-out border effects to their photos to simulate a vintage look. Amateur and professional photographers often add visible watermarks in the shape of logos or signatures to their photos. Moreover, many users set their cameras to embed a visible timestamp in the image's pixel data. Such *WTFs* (Watermarks, Timestamps and Frames) are becoming an increasing problem for Internet Vision applications.

If a WTF is present in two otherwise unrelated images, it often causes a false-positive match. This particularly affects local feature based image retrieval methods [18, 19, 27], because they only require images to share a small part of their content to form a match. For example, [17] observed that logos and frames in X-ray images are a frequent cause of false-positive matches in medical image retrieval. Furthermore, WTFs affect image clustering and landmark mining techniques [2, 4, 10, 21, 22, 23, 31, 32, 37]. These approaches typically operate on a *matching graph* in which two images are connected if they have shared content. This graph is built using local feature based image retrieval methods. A WTF can cause two images to share an edge, even though they show different objects, and this false-positive edge can cause unrelated clusters to be joined into one (Fig. 9). Image clustering often serves as the basis for Landmark Recognition [2, 10, 22, 37] which relies on pure clusters that each contain just one object or building. If this is not the case due to WTFs randomly linking unrelated images, a recognition system will confuse the joined buildings. Moreover, WTFs can form non-object clusters. For example, Chum *et al*. [5] report that timestamps form *pseudo* clusters when applying a small-object discovery algorithm. Finally, WTFs can disturb large-scale structure-from-motion engines based on Internet photos [1, 7, 9, 28, 29]. If WTFs are present in the input photos, reconstruction will either fail completely or produce incorrect reconstructions where unrelated buildings are connected by photos containing WTFs. WTFs practically affect all vision datasets crawled from the web, such as OXFORD BUILDINGS [19] and PARIS BUILDINGS [20], IMAGENET [6], the DUBROVNIK and ROME datasets [13], EUROPEAN CITIES 1M [2], and PARIS 500K [30], to name but a few. Moreover, companies like Google, Facebook or Yahoo (Flickr) rely on massive corpora of Internet images to build their vision-based services.

While several approaches exist for detecting and reading timestamps in photos [3, 12, 8, 26], most approaches for detecting visible watermarks are targeted at videos [33, 16, 11]. Because they make strong assumptions about the appearance and position of the timestamps or water-

[1] http://www.vision.rwth-aachen.de/software

Figure 1. Examples of Watermarks (left), Timestamps (middle), Frames (right). Top: Original images, Bottom: WTFs in higher resolution.

marks, they would fail when applied to the variety of WTFs present in Internet photos. So far, there is no unified approach for detecting watermarks, timestamps, and frames.

In this paper, we propose a simple but effective method to detect whether a match between two images is due to a WTF. Our approach can be applied to filter image retrieval results and to prevent false-positive matches when building a matching graph for image clustering. It is general enough to detect watermarks, timestamps, and frames alike, while being fast enough to be integrated into an existing retrieval or clustering system without introducing too much computational overhead. Given an image pair and its estimated homography, our approach computes a map of highly similar image regions and builds a spatial histogram from it. This histogram serves as input for a binary classifier that decides whether the match was caused by a WTF.

To evaluate our approach, we created a challenging dataset containing 3.6k real-world WTF and 33k non-WTF image pairs mined from Internet photos of Paris. Furthermore, we show that in image clustering, our method effectively prevents clusters of different objects from being joined and eliminates pseudo-clusters formed by WTFs.

2. Related Work

Most of the work on detecting and reading *timestamps in photos* has focused on scanned analog photos and thus on dot font and 7-segment timestamps exposed onto the analog film. While Chen *et al*. [3] and Fumin *et al*. [8] perform template matching with manually created digit templates, Li [12] uses Self-Generating Neural Networks (SGNNs) to model digit appearance. Chen *et al*. [3] make no assumptions on timestamp position, Fumin *et al*. [8] and Li [12] limit their search to only the four image corners, while Shahab *et al*. [26] learn location priors from a training set. Unlike these works, our goal is not to segment or read timestamps, but merely to detect their presence.

Müller *et al*. [17] observed that false positives in radiological image retrieval are often caused by text, logos and frames in the X-ray images. Because in this application, the overlays are separated from the content and the background is always black, connected components analysis and thresholding suffice to remove the WTFs. While the goal and motivation of [17] are very similar to ours, general Internet photos require more elaborate methods since content and overlays have a much higher variability in appearance.

A related line of research is to detect *text and timestamps in videos*. Sato *et al*. [24] and Yin *et al*. [34] exploit the constantly changing background and apply a running temporal minimum over the image intensity to isolate timestamps. This is not applicable in our setting where only two images with an overlay are given that might not have as much background variation. Another approach that only works on videos was used by Li *et al*. [14] and Yu *et al*. [35]. In both their cases, the timestamp includes seconds, which enables searching for parts of the image that change every second. Li *et al*. [14] do not even use OCR, but infer the passed time based on the changing pattern of the clock digits.

Meisinger *et al*. [16] and Yan *et al*. [33] address the problem of removing TV station logos from videos. Both use a simple frame differencing approach, which is however only effective with dynamic backgrounds and non-transparent logos. Therefore, Yan *et al*. [33] additionally use a Bayesian classifier with a location prior that favors the image corners.

Zhang *et al*. [36] aim to detect objects that have been pasted onto images. Like ours, their method works on pairs of images related by a homography. Assuming that the pasted object does *not* obey the homography, they warp the images onto each other and subtract them. Image regions with high pixel-wise difference are assumed to be the pasted object. In contrast to our work, their method assumes that the image pair shows the *same subject* and an additional pasted object. WTF matches, on the other hand, have *different* subjects because the match is on the WTF itself. Therefore, in our case, WTFs are the most *similar* image regions.

In summary, most previous work has focused on either detecting timestamps in photos or watermarks (station logos) in videos, but these approaches are too specialized and thus not applicable to our problem. Moreover, to the best of our knowledge, no one has addressed the problem of detecting frames in photos, and we are not aware of any previous paper that addresses the detection of watermarks, timestamps, and frames with a single approach.

3. Method

Our goal is a detector that is *general* enough for all kinds of watermarks, timestamps and frames. It needs to have *high accuracy* since a single missed WTF can already cause clusters of unrelated objects to be merged (Fig. 9), and it needs to be *fast* so that it can be applied in large-scale settings. Our detector is used as a post-processing step in im-

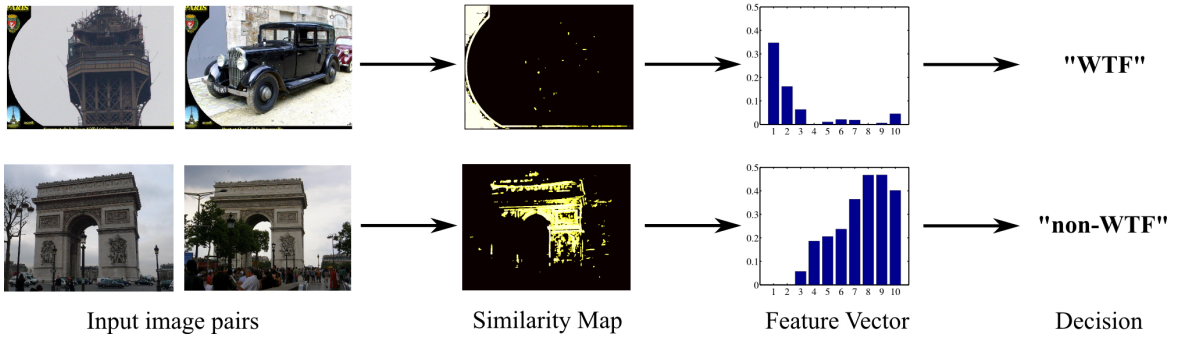


Figure 2. Workflow of the WTF detector. Top: WTF match, bottom: non-WTF match.

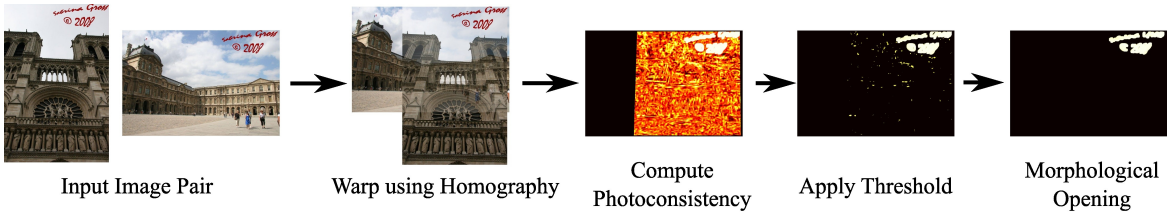


Figure 3. Similarity map computation based on photoconsistency.

age retrieval that decides whether an image *match*, *i.e.* a pair of images related by a homography, is caused by a WTF. An alternative would be to detect WTFs in every *single* image. However, if an image with a WTF has matches due to its actual content, this method would also discard them.

WTFs come in many shapes and sizes (Fig. 1), but after visually inspecting thousands of them, we found that the following two assumptions generally hold. (i) Watermarks and frames always have the exact *same appearance*, since they are simply templates pasted onto the image. Timestamps are slightly more challenging since the background is usually visible behind them and their appearance differs depending on the time. (ii) All WTFs are typically *close to the border* of the image. However, they may have varying positions and scales in each of the two matching images. For example, a professional photographer might always place her logo to not occlude the subject.

Based on the above assumptions, we now propose a method to detect WTF matches. The basic steps of our detector are as follows (Fig. 2): 1. Detect highly similar image regions. 2. Compute a feature vector consisting of a spatial histogram of the similar regions and some summary statistics. 3. Classify the match as WTF or non-WTF. In this section, we propose different choices for each of these three steps and evaluate them in Sec. 4.

Detecting similar image regions. In the first step, we compute an image similarity map that we later extract features from. We propose two approaches for this: one based on *photoconsistency* and one based on homography *inliers*.

The process of computing similarity maps based on *photoconsistency* is shown in Fig. 3. Because WTFs can appear at different positions in the image, we first warp image 2 onto image 1 using the homography between them. We then compute pixel-wise photoconsistency scores based on normalized cross-correlation (NCC). To isolate WTF regions, we set all NCC values below a threshold θ to 0. Finally, we use morphological opening to eliminate small regions that likely do not belong to a WTF. Fig. 4 (3rd column) shows example similarity maps computed with this method. While it generally yields very precise results, we found that uniform image regions with high NCC can often occur randomly across the image. Moreover, this method is expensive and requires the choice of three parameters, namely the NCC window size r, the NCC threshold θ, and the size of the structuring element for morphological opening.

Another way to find similar image regions exploits the fact that the homography *inliers*, *i.e.* corresponding local features that obey the homography transformation, are available to our detector at no extra cost. Assuming that feature correspondences are a sufficient indication of the similarity of their respective image regions, we create the similarity map simply by forming the union of all inliers' interest regions in image 1. While this yields less precise similarity maps than the *photoconsistency* method, it is much faster and does not randomly fire in uniform regions. Fig. 4 (4th column) shows some example *inlier* similarity maps.

Spatial Histogram of Similar Regions. Based on these similarity maps, we now compute feature vectors suitable as input for a classifier. As mentioned above, we assume that for WTFs, similar regions appear mainly at the image borders, while for valid matches, similar regions appear all over the image (Fig. 4). Therefore, we compute spatial histograms of the similarity maps to detect WTFs. We investigate four histogram shapes (Fig. 5). `dist2border` and `dist2centre` are histograms of the distance to the image border and center, respectively. `cake` is an an-

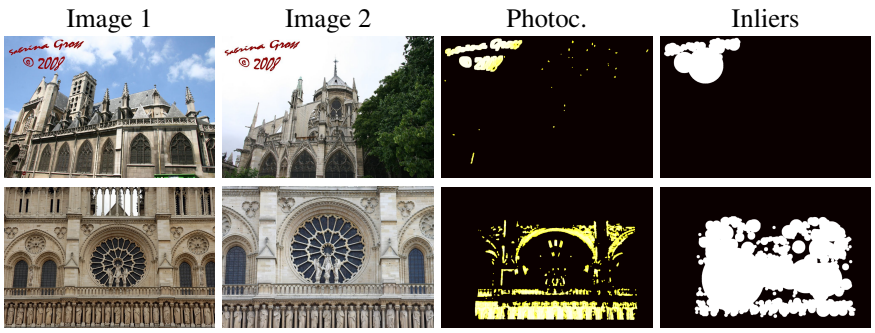


Figure 4. Similarity maps computed with the *photoconsistency* and *inlier* methods.

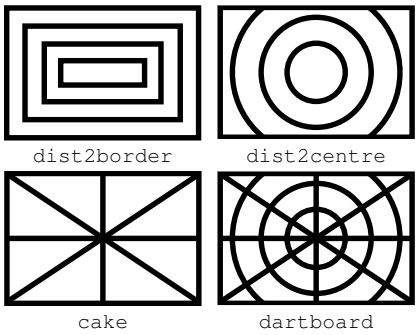


Figure 5. Spatial histogram shapes.

gular histogram, and `dartboard` is the combination of `dist2centre` and `cake`.

We also experiment with additional attributes, namely the *mean photoconsistency* and the *coverage*, *i.e.* the fraction of active pixels in the similarity map. The resulting feature vector serves as input for AdaBoost with decision trees as weak classifiers. This delivered performance superior to both linear SVMs and SVMs with RBF kernels.

Two-way Matching. Sometimes the matching region between two images is at the border of one image but in the middle of the other (See Fig. 7b). In such a case, depending on which of the images we choose for computing the similarity map, the match can be falsely detected as a WTF. To prevent this, we compute the similarity maps and histograms in two directions; once by projecting image 2 onto image 1, and once by projecting image 1 onto image 2. We then only call the image pair a WTF if *both* feature vectors are labeled as positive by the classifier. This way, we enforce that the WTF regions are close to their learned positions in both images. This not only reduces the number of false positives, but also doubles the amount of training data, since each image pair now generates two training samples.

Integration into Image Retrieval and Clustering. In visual-words-based *image retrieval* [27, 19], potential matches of a query image are ranked w.r.t. their tf-idf score and then verified by fitting a geometric transformation. Our detector is simply used as a *second* verification step that rejects WTF matches. This integration directly benefits *image clustering* approaches that are based on a *matching graph*. This graph is built by performing image retrieval and linking matching images by edges. By performing WTF detection after image retrieval, false-positive edges in the matching graph can be avoided directly during graph construction.

4. Experiments and Results

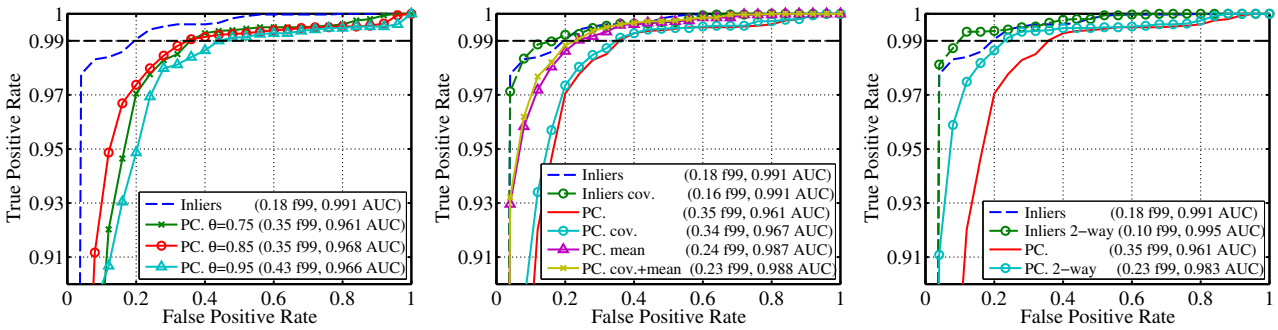
We first give a detailed analysis of our method in a classification setting and then show how it can improve image clustering results by filtering false matches.

Dataset and Ground Truth. We collected a realistic dataset[2] of 36,240 image matches for evaluating our WTF detector. Each image pair has a binary label: 90% of the matches are *correct*, or *non-WTF* matches, and 10% are *WTF* matches. The basis for this dataset is the PARIS500K dataset [30], collected from Flickr and Panoramio, that consists of 500k photos taken in the inner city of Paris. To collect our WTF dataset, we first generated a large pool of image matches in PARIS500K by performing a pairwise matching using standard image retrieval techniques [18, 19, 27] and then mined this pool for WTF matches.

To perform the pairwise matching, we computed bags of visual words [27] from the images by quantizing their SIFT features [15] using a visual vocabulary of 1M visual words. We then constructed an inverted file index [19] and queried it once with each image. We ranked query results by their tf-idf score and spatially verified the top-300 matches by establishing feature correspondences using the 2nd-nearest-neighbor criterion [15] and fitting a homography using SCRAMSAC [25]. If an image pair had 15 or more inliers w.r.t. the homography, we considered it a *match* and added it to the pool along with its homography and inliers, which are required for the similarity map computation.

We then used four methods to mine WTF matches from this pool. (i) We searched for users whose photos frequently match among themselves. (ii) We searched for image pairs where all inliers are in one of the four image corners. (iii) We clustered the dataset using Iconoid Shift [31] and identified clusters containing multiple unrelated buildings, since these are typically caused by WTF matches that create invalid edges in the matching graph. (iv) We manually went through a large number of images from the dataset and collected the matches of all images containing a WTF. We then manually inspected all our collected image matches and filtered out the remaining non-WTF matches. Because we found that some users were responsible for a large number of matches in the pool, we limited the number of WTF matches from the same user to 250 to avoid a bias towards certain kinds of WTFs. The resulting set of 3,624 image matches consists of 61% Watermarks, 23% Timestamps and 16% Frames. By visual inspection of 1,000 random matches from this set we found that that 99.6% of the WTFs were near the image border, validating our initial assump-

[2] Available at http://www.vision.rwth-aachen.de/data



(a) (b) (c)

Figure 6. Effect of different parameters on classification performance. (Note that the y-axis ranges from 0.9 to 1.0.) (a) *Photoconsistency* vs. *inlier* similarity maps (using a `dist2border` histogram (Fig. 5) with 5 bins). (b) Effect of additional features. (using θ=0.75 for *photoconsistency*) (c) Effect of two-way matching.

tion. The exception cases were semitransparent watermarks placed in the middle of the image. We then added 32,616 non-WTF matches from our pool of matches, so the final dataset has 10% WTF matches. For cross-validation, we then split up the dataset into 5 folds, taking care that WTFs from the same user are all in the same fold to avoid training on the test data.

Evaluation Procedure. We now evaluate our method in a *binary classification* setting using 5-fold cross-validation. We plot receiver operator characteristic (ROC) curves by *vertical averaging*, *i.e.* we average the true positive rates of the five folds for each false positive rate. Additionally, we consider the area under the ROC curve (AUC) and the false positive rate at a true positive rate of 0.99, which we call *f99*. This is motivated by our target application, image clustering. While the effect of a few missing matches is negligible since an image cluster is usually very densely connected, a single false-positive match caused by a WTF can already result in the merging of two unrelated clusters (Fig. 9). Our primary goal is therefore to reach high recall. We now analyze the different parameters and design choices introduced in Sec. 3 and how they affect performance and efficiency.

Photoconsistency vs. Inliers. The first pipeline step is detecting similar image regions. We compare the *photoconsistency* and *inliers* methods using different NCC thresholds θ for *photoconsistency* (Fig. 6a). Surprisingly, *inliers* outperforms *photoconsistency* by a large margin (0.18 *f99* vs. 0.35 *f99*). The reason is that *photoconsistency* is much more prone to false-positives, because high photoconsistency regions are likely to occur by accident, *e.g.*, in uniform regions (Fig. 7a). *Inliers* does not have this problem, because interest points are mostly detected in textured regions and because SIFT matches are more discriminative.

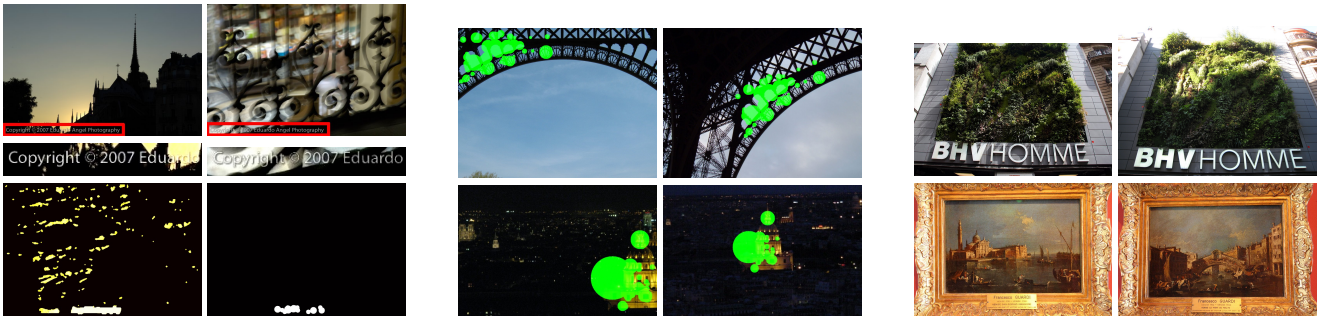
Efficiency. The main performance difference lies in the similarity map computation. The *photoconsistency* method needs to compute the NCC by applying a sliding window over both images and computing a dot product at each position. Let N be the number of pixels in the overlap of the warped images and r be the window size, then NCC computation requires $O(N * r^2)$ operations. The *inliers* method only needs to fill the interest regions of the inlier features. Let s be the average diameter of an interest region and R the number of interest regions, then the similarity map computation takes $O(R * s^2)$ operations. Since there are much less interest points than pixels ($R \ll N$), the *inliers* method is much faster. In our measurements, the average time to compute the feature vector for an image pair is 1.5 seconds using *photoconsistency* and 0.15 seconds using *inliers*[3]. Therefore, *inliers* is highly preferable both in terms of performance and efficiency.

Additional Features. The effect of the additional features *coverage* and *mean* is shown in Fig. 6b. While *coverage* has only a small effect on both methods, the *mean* strongly increases the performance of the *photoconsistency* method. The reason is that accidentally photoconsistent regions (yellow regions in Fig. 7a, bottom right) typically have lower photoconsistency, which decreases the mean.

Two-way Matching yields a large performance improvement (Fig. 6c). The *f99* improves from 0.18 to 0.10 for *inliers* and from 0.35 to 0.23 for *photoconsistency*. This is because firstly, twice the amount of training data is available, and secondly, image matches where the similar region is at the center in one image, but at the border in the other are filtered out by this method. In the examples in Fig. 7b, only the left images were considered when not using two-way matching, causing them to be falsely classified as WTFs.

Histogram Shapes. We investigate four spatial histogram shapes (Fig. 5). As Fig. 8a shows, `dist2centre` performs highest. Although `dartboard` with 5 distance and 16 angular steps achieves similar performance, it has 80 dimensions, whereas `dist2centre` only has 5 dimen-

[3]Timings were measured on a 2.7 GHz Intel Core i7 CPU and were based on prototype Matlab implementations. Our open source library is a faster C++ implementation of the *inliers* method.



(a) (b) (c)

Figure 7. (a) The *inlier* method can prevent errors in similarity map estimation. Top: WTF match. Middle: Closeup of WTF. Bottom left: *photoconsistency* map with errors in homogeneous image regions (yellow). Bottom right: *inlier* map. (b) False-positive WTFs that two-way matching prevents (similarity map overlaid in green). (c) Two unusual false-positive WTF detections in PARIS 500K.

2-way	hist.	f99	AUC
	d2c	0.228	0.989
X	d2c	**0.184**	**0.993**
	d2b	0.355	0.961
X	d2b	0.228	0.983

(a) Photoconsistency

2-way	hist.	f99	AUC
	d2c	0.045	0.995
X	d2c	**0.034**	**0.998**
	d2b	0.183	0.991
X	d2b	0.095	0.995

(b) Inliers

Table 1. Comparison of different detector settings.

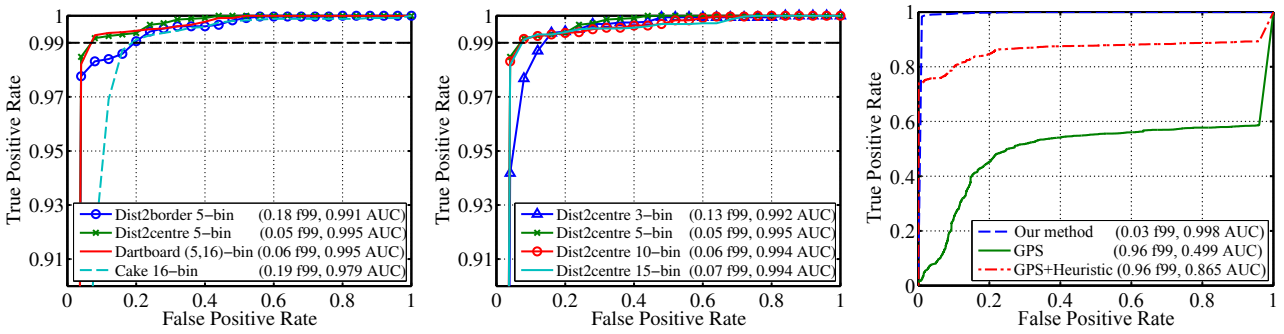
sions. Fig. 8b shows the effect of the number of bins for the `dist2centre` histogram and the *inliers* method. The strong performance increase from 3 to 5 bins shows that a certain spatial resolution is required to reliably detect WTFs. However, more bins do not bring an advantage, which allows us to keep feature dimensionality low.

Summary. Tab. 1 summarizes our results. For both *photoconsistency* and *inliers*, `dist2centre` with 2-way matching performs highest. The best setup achieves 3.4% *f99*.

Comparison against Baselines. Since the task of detecting watermarks, timestamps and frames has not been addressed before and previous methods are not applicable to Internet photos (Sec. 2), we compare against two simple methods used in practice. The first is to use GPS tags to restrict the candidate matches geographically [2, 22, 37], assuming that images taken far apart cannot show the same object. If they match anyway, it must be a false positive. Since geotags are available for our images, we apply this method by classifying all image pairs with a geographic distance above a threshold t as WTFs. Fig. 8c shows the ROC curve drawn by varying t, compared against our best performing setup. We found that the main reasons for the much lower performance of this method are: (i) Inaccurate GPS tags causing false-positive WTF detections. For example, two pictures of a sculpture in a museum may have far-away geotags due to bad GPS reception, causing the match to be classified as a WTF. (ii) Although two images have close-by geotags, they do not always show the same object. Hence, matches between them can still be caused by a WTF. In our experiments in [31] we found that the following heuristic, combined with the GPS-based method, helped reduce WTF matches: An image match is removed if at least 50% of its inliers are in the top or bottom 10% of the image. While this additional step drastically increases the *tpr*, its performance, especially the *f99*, is still not comparable to our method.

Application to Clustering. We now demonstrate how our detector can improve the results of image clustering and thus benefit applications such as landmark recognition or structure-from-motion. We use two datasets, PARIS 500K [30] and OXFORD 105K [19]. For each dataset, we build one baseline matching graph using standard image retrieval and one using image retrieval with subsequent WTF removal. We then cluster both graphs using Iconoid Shift [31]. The clustering based on the baseline matching graph is called *clustering 1* and the clustering based on the matching graph with WTF removal is called *clustering 2*. We now compare these clusterings for both datasets. For WTF removal, we use *inlier* similarity maps, `dist2border` histograms with 5 bins and two-way matching. We selected the classifier score threshold to an operating point of 0.99 *tpr* determined using cross-validation on the WTF dataset.

Results on Paris 500k. *Clustering 1* of PARIS 500K has 14,254 clusters covering a total of 111,892 images. 87 clusters, and in particular the three largest clusters, are affected by WTFs. For example, the largest cluster (5,435 images) contains the Eiffel Tower, the Louvre, Notre Dame, the Arc de Triomphe, several paintings and sculptures, as well as many non-landmark tourist photos such as portraits or pictures of food. In *clustering 2*, 38 of these clusters are completely removed since they only contained WTFs. Two WTF clusters survived unharmed, but they contained only 2 and 3 images respectively. 39 clusters, covering 17,857 images, were split into a total of 317 clusters. Only 14 of them still contained WTF matches, but all of these clusters were smaller than 5 images. The remaining 303 clusters were all pure, each containing only one object. Fig. 9 shows an example of a successfully split cluster. 8 clusters from *clustering 1* contained only a small fraction of WTF images, but were pure otherwise. In *clustering 2*, all WTFs were



(a) (b) (c)

Figure 8. (a) Performance of different histogram shapes (using *inliers* and the respective best-performing bin counts). (b) Performance of different histogram sizes. (c) Comparison with two baseline methods. The settings for our method are the best performing setup: `dist2centre`, 5-bin, inliers, 2way. (Note that the y-axis is scaled differently in this plot, so the curves of the other methods are visible.)

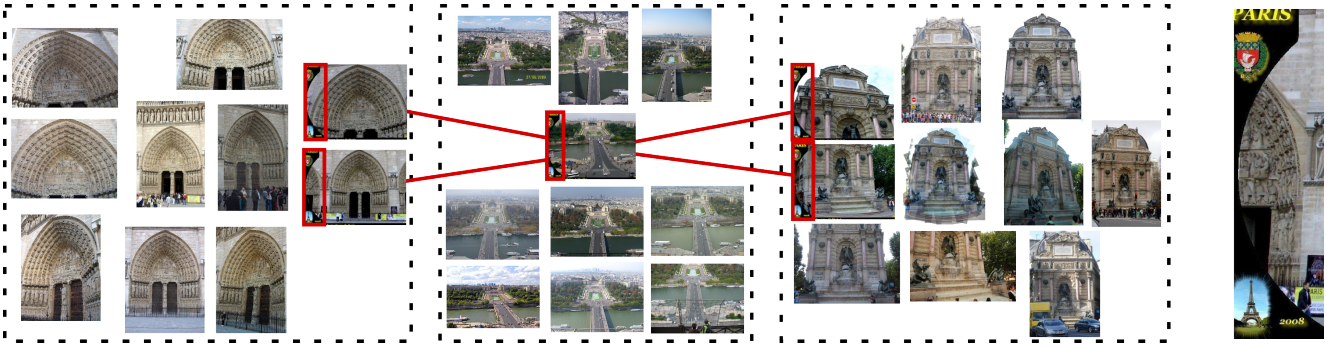


Figure 9. Subset of a cluster successfully split by our algorithm. All images were initially in the same cluster due to WTF matches (red, closeup on the right) that linked images of different buildings. The cluster was split into the three clusters denoted by the dashed rectangles.

removed from these clusters.

Due to false positives, 79 non-WTF clusters were removed, but all of them were very small: 70 of them only had size 2 and the largest had size 7. False positives that are removed in larger clusters usually do not have any effect, since these clusters are densely connected. Most false positive WTF detections are due to image pairs that have matching features only at the border. This can happen, *e.g.*, if there is extreme panning between the two photos such that they have only very little overlap. Fig. 7c shows two rather unusual cases. (Top: A planted facade resembling a frame. Bottom: Two paintings with similar *physical* frames.)

Results on Oxford 105k. The OXFORD 105K dataset, used to evaluate image clustering in [4, 5, 21], consists of 5k images of *Oxford* showing 11 landmarks, and 100k *distractor* images collected from Flickr by random keyword searches.

Clustering 1 of OXFORD 105K has 6,225 clusters covering 17,599 images. Since the *Oxford* part of the dataset is relatively clean of WTFs, none of the Oxford building clusters were merged. We found that 7 clusters in *clustering 1* were merged by WTFs, 8 contained only WTFs, and one cluster contained some WTFs but was otherwise pure. In *clustering 2*, 6 of the 7 merged clusters were successfully split. In the remaining cluster, all photos had a common background object that caused them to match. 7 of 8 clusters containing only WTFs were removed. The missed cluster contained 4 identical images with a frame, which were not detected because they had inliers over the whole image area. Finally, the cluster consisting partially of WTFs was successfully cleaned of unrelated images (Fig. 10). 203 clusters were removed although they did not contain WTFs, but all of them were only of size four or less.

Summary. We have shown that our method prevents the formation of clusters whose images match only due to WTFs and prevents clusters from being merged due to WTFs. Though trained on PARIS 500K, our detector was able to generalize to OXFORD 105K where it was just as successful in removing WTFs.

5. Conclusion

We considered the so far neglected problem of detecting watermarks, timestamps, and frames (WTFs) in Internet photos. WTFs cause false-positive image matches that harm image retrieval and image clustering applications. We proposed a simple yet effective method to detect WTF matches based on spatial histograms of similar image regions and showed that it achieves high performance on a realistic dataset and successfully fixes the problems that WTFs cause in image clustering.

Figure 10. A cluster in the OXFORD 105K dataset that contains unrelated images due to false-positive matches caused by timestamps (example on the right). The crossed-out images were removed by our detector, leaving only the relevant images.

Acknowledgments. We gratefully acknowledge support by the DFG Excellence Cluster UMIC (DFG EXC 89) and by a Google Faculty Research Award.

References

[1] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building Rome in a Day. In *ICCV*, 2009.

[2] Y. Avrithis, Y. Kalantidis, G. Tolias, and E. Spyrou. Retrieving Landmark and Non-Landmark Images from Community Photo Collections. In *ACM Multimedia*, 2010.

[3] X. Chen and H.-J. Zhang. Photo time-stamp detection and recognition. In *ICDAR*, 2003.

[4] O. Chum and J. Matas. Large-scale discovery of spatially related images. *PAMI*, 32(2):371–377, 2010.

[5] O. Chum, M. Perdoch, and J. Matas. Geometric min-Hashing: Finding a (Thick) Needle in a Haystack. In *ICCV*, 2007.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[7] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a Cloudless Day. In *ECCV*, 2010.

[8] B. Fumin, L. Aiguo, and Q. Zheng. Photo Time-Stamp Recognition Based on Particle Swarm Optimization. In *WI*, 2004.

[9] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards Internet-scale Multi-view Stereo. In *CVPR*, 2012.

[10] S. Gammeter, T. Quack, and L. Van Gool. I Know What You Did Last Summer: Object-Level Auto-Annotation of Holiday Snaps. In *ICCV*, 2009.

[11] C.-M. Kuo, C.-P. Chao, W.-H. Chang, and J.-L. Shen. Broadcast Video Logo Detection and Removing. In *IIHMSP*, 2008.

[12] A. Li. Fast Photo Time-Stamp Recognition Based on SGNN. In *Advances in Neural Networks - ISNN 2006*, volume 3972, pages 316–321. Springer Berlin Heidelberg, 2006.

[13] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Location Recognition using Prioritized Feature Matching. In *ECCV*, 2010.

[14] Y. Li, K. Wan, X. Yan, and X. Yu. Video Clock Time Reconition Based on Temporal Periodic Pattern Change of the Digit Characters. In *ICASSP*, 2006.

[15] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2), 2004.

[16] K. Meisinger, T. Troeger, M. Zeller, and A. Kaup. Automatic TV logo removal using statistical based logo detection and frequency selective inpainting. In *ESPC*, 2005.

[17] H. Müller, , J. Heuberger, and A. Geissbuhler. Logo and text removal for medical image retrieval. In *Bildverarbeitung für die Medizin 2005*, 2005.

[18] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *CVPR*, 2006.

[19] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *CVPR*, 2007.

[20] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases. In *CVPR*, 2008.

[21] J. Philbin and A. Zisserman. Object Mining using a Matching Graph on Very Large Image Collections. In *ICVGIP*, 2008.

[22] T. Quack, B. Leibe, and L. Van Gool. World-Scale Mining of Objects and Events from Community Photo Collections. In *CIVR*, 2008.

[23] R. Raguram, C. Wu, J.-M. Frahm, and S. Lazebnik. Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs. *IJCV*, 95(3):213–239, 2011.

[24] T. Sato, T. Kanade, E. K. Hughes, and M. A. Smith. Video OCR for Digital News Archive. In *CAIVD*, 1998.

[25] T. Sattler, B. Leibe, and L. Kobbelt. SCRAMSAC: Improving RANSAC's Efficiency with a Spatial Consistency Filter. In *ICCV*, 2009.

[26] A. Shahab, F. Shafait, and A. Dengel. Bayesian Approach to Photo Time-Stamp Recognition. In *ICDAR*, 2011.

[27] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *ICCV*, 2003.

[28] N. Snavely, S. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *SIGGRAPH*, 2006.

[29] N. Snavely, S. Seitz, and R. Szeliski. Modeling the World from Internet Photo Collections. *IJCV*, 80(2), 2008.

[30] T. Weyand, J. Hosang, and B. Leibe. An Evaluation of Two Landmark Building Discovery Algorithms. In *ECCV RMLE Workshop*, 2010.

[31] T. Weyand and B. Leibe. Discovering Favorite Views of Popular Places with Iconoid Shift. In *ICCV*, 2011.

[32] T. Weyand and B. Leibe. Discovering Details and Scene Structure with Hierarchical Iconoid Shift. In *ICCV*, 2013.

[33] W.-Q. Yan, J. Wang, and M. S. Kankanhalli. Automatic video logo detection and removal. *MMS*, 10(5):379–391, 2005.

[34] P. Yin, X.-S. Hua, and H.-J. Zhang. Automatic Time Stamp Extraction System for Home Videos. In *ISCAS*, 2002.

[35] X. Yu, J. Cheng, W. Song, and B. He. An Algorithm for Timestamp Removal for Panorama Video Surveillance. In *ICIMCS*, 2013.

[36] W. Zhang, X. Cao, Y. Qu, Y. Hou, H. Zhao, and C. Zhang. Detecting and Extracting the Photo Composites Using Planar Homography and Graph Cut. *IFS*, 5(3):544–555, 2010.

[37] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the world: Building a Web-Scale Landmark Recognition Engine. In *CVPR*, 2009.