Virtually Augmenting Hundreds of Real Pictures: An Approach based on Learning, Retrieval, and Tracking

Julien Pilet*　　Hideo Saito†

Keio University

ABSTRACT

Tracking is a major issue of virtual and augmented reality applications. Single object tracking on monocular video streams is fairly well understood. However, when it comes to multiple objects, existing methods lack scalability and can recognize only a limited number of objects. Thanks to recent progress in feature matching, state-of-the-art image retrieval techniques can deal with millions of images. However, these methods do not focus on real-time video processing and can not track retrieved objects.

In this paper, we present a method that combines the speed and accuracy of tracking with the scalability of image retrieval. At the heart of our approach is a bi-layer clustering process that allows our system to index and retrieve objects based on tracks of features, thereby effectively summarizing the information available on multiple video frames.

As a result, our system is able to track in real-time multiple objects, recognized with low delay from a database of more than 300 entries.

1 INTRODUCTION

Object tracking is an important issue for many applications, especially in the domains of virtual, mixed, and augmented reality. The base process to visually integrate virtual elements on real ones is the following: A camera captures a scene. A registration technique provides the relative pose of the camera with respect to the target object. A standard CG method renders virtual contents from the appropriate point of view and integrates it on the camera image. Examples of such applications are numerous and include augmenting the pages of a book, playing cards, and trading cards [20, 24, 23]. In these cases, as often, the camera is the only registration device available.

Many methods have been proposed for visual registration [8]. However, when it comes to track multiple natural objects, they all have drawbacks. A classical approach to track multiple objects is to detect a part common to all objects, typically a highly contrasted square, and use the registration to ease searching the characteristic area in the database. However, the necessity of marking target objects severely restricts the application domain of such tracking approaches. Recently, wide-baseline point matching has been demonstrated to effectively detect and track naturally textured objects [15, 25]. However, when multiple objects are to be recognized, these approaches try to match each known object in turn. Such a linear complexity, both in computation and memory requirements, limits the number of known targets to a few objects [16].

Scalability is addressed by image retrieval approaches. These methods can deal with millions of images [13]. However, they focus on static images and are not designed for real-time tracking.

In this paper, we propose a real-time tracking method that integrates image retrieval techniques to achieve both accurate tracking and scalability with respect to the number of target objects. More specifically, we exploit information available in video stream, as well as the variability of feature descriptors, to efficiently establish correspondences with, index, and retrieve target objects. As a result, our method can augment more than 300 objects individually, as depicted by Fig. 1.

To better describe the algorithmic contribution of our paper, let us sketch a typical image retrieval system. The first step is detection of features such as SIFT [10] or MSER [12], to summarize an image with a set of vectors. These vectors are quantized, turning features into *visual words*. Quantization involves clustering a large number of representative features, forming a *visual vocabulary*. Once an image is expressed as a *bag of word*, it can be indexed and searched for using information-retrieval techniques [13, 22]. However, quantization errors combined with feature instability reduce retrieval performances. A key contribution of our work is a method that exploits quantization effects and the non-linear variations of features to improve retrieval and to reach a computation speed compatible with real-time tracking.

Our method observes the variability of descriptors over several frames, by tracking the keypoints from frame to frame. Inspired by Ozuysal *et al.* [15], we capture during a training phase the behavior of features. The collected data allows our method to create a stabilized visual vocabulary. At runtime, our system matches full point tracks, as opposed to neighborhood obtained from a single frame, against training data. It yields a stable entry that can serve as index key.

At a higher level, our contribution is a multiple natural ob-

*e-mail:julien@hvrl.ics.keio.ac.jp
†e-mail:saito@hvrl.ics.keio.ac.jp

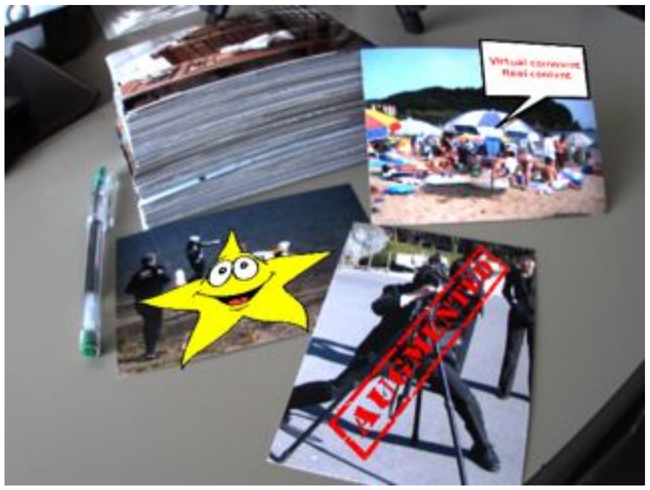

Figure 1: To produce this result, our system detects, identify, and track the photographs visible on the input video stream. Each picture is augmented with its corresponding virtual element, precisely registered. The stack of pictures visible on this image contains about 300 images. Our system can recognize all of them. Users can also augment new objects by simply showing them to the camera and clicking. Tracking and augmentation starts right away.

ject tracking system designed to scale well with the number of targets. Several of its properties make it perfectly suited for augmented reality applications: It can process live video stream, it can deal with a large number of target objects, adding new targets to the database is perceptually immediate, initialization is fully automatic, and it is robust to viewpoint changes, illumination effects, partial occlusion, and other typical tracking hazards.

We demonstrate the effectiveness of our approach with a toy application that can augment more than 300 pictures and that allows users to interactively augment with virtual elements a large number of new objects.

2 Related Work

Pixel level registration techniques nowadays reached maturity [11, 1]. Their association with point selection methods to concentrate computation efforts on interesting pixels forms the basis of many computer vision tasks such as object tracking [5, 21]. Tracking requires initialization, which remained an issue a few years ago. Because marker based approaches such as ARToolkit did not suffer from this drawback, they quickly became popular in the augmented reality community [7]. Among the many improved versions proposed, ARTag, for example, can recognize 2002 different markers [3].

In parallel, feature point methods grew powerful enough to achieve wide-baseline matching. The most representative of them is probably Lowe's scale invariant feature transform (SIFT [10]). Establishing correspondences between views with strong differences in pose and illumination addresses many issues, including tracking initialization. Lepetit *et al.* proposed a real-time tracking by detection method [9]. More information about model based tracking approaches can be found in [8].

The success of wide baseline feature matching also opened the way to large scale image retrieval [22, 14]. Using vector quantization of descriptors, recent approaches effectively retrieve images from databases containing more than a million of images [13, 17, 6]. Interestingly, Philbin *et al.* [18] note that quantizing several descriptions of the same physical point can be unstable. They tried to capture this instability by synthesizing affine and noisy deformations of image patches, inspired a classification based approach to wide baseline matching [9]. However, these attempts did not improve recognition performance, maybe because of an inappropriate choice of image perturbation, as the authors explain. Following a similar goal, our method observes descriptors tracked over several frames to mitigate the effect of their variability.

Feature matching has also been used to detect and track multiple 3D objects [16]. Recently, Wagner *et al.* proposed a method implemented on mobile phones that guarantees a minimum frame rate [26]. However, these methods are restricted to a limited number of objects, typically under 10. We focus on scaling the database size, while keeping a detection delay compatible with tracking tasks.

3 Method

The primitives upon which our method is built are feature detection and tracking. Keypoints[1] are detected at each frame, and matched between consecutive frames, using normalized cross-correlation (NCC). When such a simple NCC fails to find correspondences, the Lukas-Kanade (KLT) algorithm tracks lost features [1]. As a result, we obtain stable tracks of features, and a patch is extracted at every frame, as illustrated by Fig. 2a.

During a first training phase, we collect many features from a video stream. We cluster their descriptors with a recursive K-mean tree, as suggested by Nister *et al.* [13] and as depicted by Fig. 2b. This allows us to summarize a descriptor as the leaf it corresponds to or as an integer, since leaves are numbered.

In a second training phase, we collect tracks of features. The tree quantizes each descriptor, turning the feature tracks into leaf tracks. We then compute for each track a histogram of leaves. Because the training sequence contains effects such as motion blur, perspective distortion, or moving specular reflections, the collected histograms capture the descriptor's instability, including the quantization effects of the tree, in presence of such hazard. The set of collected histograms form a dictionary, or a *visual vocabulary*. To reduce its ambiguity, similar histograms are recursively merged until am-

[1]In this text, we define a *keypoint* as a location of interest on an image, a *descriptor* as a vector describing a keypoint neighborhood, and a *feature* as a keypoint and its descriptor.

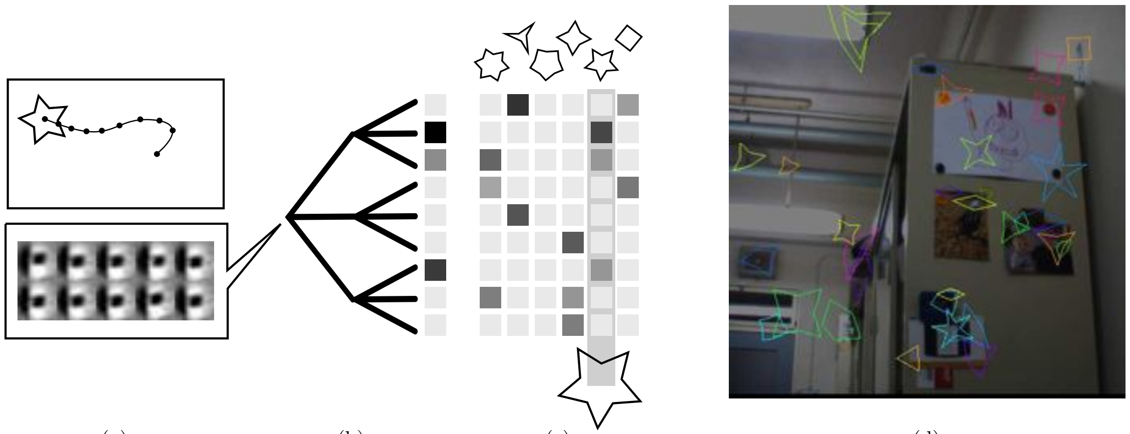
(a) (b) (c) (d)

Figure 2: Computing *visual words* by tracking features. (a) Features are tracked across frames, and patches are collected. (b) Patches pass through a tree and distribute over its leaves, forming a histogram. (c) The histogram is compared to the ones observed during the training phase. We use them as *visual words*, and visualize them with geometric shapes. In this example, the histogram matches a word represented with a star. (d) Some of the *visual words* detected on an input image.

biguity reaches an acceptable level.

At run-time, the whole history of each tracked feature is summarized by the most relevant histogram found in the dictionary. Each descriptor passes through the tree and ends up in a leaf, forming a histogram of leaves, as depicted by Fig. 2b. We then simply search for the most similar trained histogram (Fig. 2c). The advantage of this technique is double: It allows the algorithm to exploit both the variability of descriptors, which is usually viewed as a problem, and the large amount of data collected by tracking points over multiple frames. Fig. 2d shows an image and some of its detected features, with their associated histograms, represented as geometric shapes.

Indexing and retrieval follows a standard TF-IDF weighting scheme on a bag of word model. In simpler words, a table maps dictionary's entries to indexed objects. For retrieval, the table is looked up to collect all the frames in which the visual words of the query appear. The candidates are then ranked using an appropriate criterion.

The few best results are selected as candidates for geometric verification. If an object is successfully detected, matches are propagated to the next frame using motion flow estimation. Currently tracked objects are automatically appended to the list of candidates for next frame's geometric verification stage. Because we eliminate outliers during detection and propagate matches, few outliers remain on the next frame. Geometric verification therefore becomes simpler.

The remaining of this section details our system's components.

3.1 Low-Level: Repeatable Sparse Motion Flow

Our method relies on a particular type of motion flow estimation in which the tracked points are detected in a repeatable way. It means that a point that has previously been tracked is supposed to be tracked again when viewed from another angle. Stability and repeatability are the main goals of well known feature detectors such as SIFT, SURF, or FAST [10, 2, 19]. In our experiments, we used a GPU implementation of SIFT [27].

To compute the motion flow from frame to frame, we first detect the keypoints in both frames. We then compare their local neighborhoods using NCC. This quickly provides the flow of most features. However, when the feature detector detects a point on the first frame but fails to detect it on the following one, the KLT algorithm searches its new location. The decision to turn to KLT is taken when the best found correlation drops below a threshold.

The result of this process is a set of stable tracks. The KLT tracking mitigates the feature detector's failures, while the feature detector ensures repeatability and prevents the KLT tracker from drifting. In our experiments, this approach can track efficiently hundreds of points over hundreds of frames.

3.2 Describing tracked Keypoints

At this stage, our goal is to index target objects and to retrieve the ones visible on the current frame, relying on stable feature tracks. To do so, we aim at constructing a dictionary mapping feature tracks to indexed objects. Our approach has two stages of clustering: at descriptor level and at track level.

3.3 K-mean Tree

The descriptors extracted during the training sequence are clustered using a recursive K-mean, as proposed by [13]. The distance measure used is the L^2 norm. Each node has at most 4 children, and recursion stops either at depth 8 or when fewer than 64 descriptors are found in this cluster.

3.4 Learning and Exploiting Feature Variability

Our system tracks points and assigns them to a leaf of the recursive K-mean tree, making it possible to observe groups of leaves showing the same physical point. During the second training stage, such groups are collected from a video sequence and accumulated into histograms. Such histograms can capture the descriptor's non-linear variability, that might be caused by viewpoint, illumination, or noise effects.

Straightforward usage of these histograms as index keys is not possible due to the many redundant elements collected during training. We address this ambiguity issue with a second clustering stage. If two histograms are too difficult to discriminate, they are merged, creating a new, larger histogram. In practice, we compute the dot product between two normalized histograms and use agglomerative clustering. We recursively merge the histogram pair with the highest normalized dot product, and stop when it is lower than a threshold (typically 0.1, see Sec. 4). Merging two histograms a and b into a new histogram x gives: $x(l) = a(l) + b(l)$, where $a(l)$ is the number of times leaf l appears in a. At the end of this process, the remaining histograms form a stable visual vocabulary.

3.5 Feature Tracks as Index Keys

We build an inverted table mapping leaves of the recursive K-mean tree to trained histograms. Given one or several leaves observed when tracking a point, it becomes possible to efficiently fetch the most similar learned histogram. By doing so, our system assigns an index key to a *point track,* as opposed to a single feature. The following equation defines the score of the histogram h for the track t:

$$s(t,h) = \frac{1}{\sum_l t(l)} \frac{1}{\sum_l h(l)} \sum_l t(l)h(l)\mathrm{idf}(l)\,, \tag{1}$$

where $t(l)$ (respectively $h(l)$) is the number of features assigned to leaf l in the track t (respectively in the trained histogram h). The term $\mathrm{idf}(l) = -\log\left(\frac{f(l)}{F}\right)$, where $f(l)$ denotes the number of trained histograms involving leaf l, and F the total number of trained histograms. This score gives more importance to rare and discriminative leaves and decreases the weight of frequent ones.

The complexity of directly computing this score grows linearly with the track size. Therefore, we remember for each track the scores of potential matching histograms. When a new frame is available, the scores are updated regarding only the newly observed leaf. This incremental approach allows our system to efficiently exploit long tracks.

Soft visual word assignment, as suggested by Philbin *et al.* [18], can easily be achieved by considering not only the histogram with the highest score, but also the ones at least 90% as good.

3.6 Object Detection

To detect target objects entering the field of view, the database is queried with all point tracks visible on the current frame. As explained, each point track is assigned to a visual word. The histogram of the observed *visual words* in a frame forms a query q. The score of stored object d for the query q is:

$$s(q,d) = \frac{1}{\sum_w q(w)} \frac{1}{\sum_w d(w)} \sum_w q(w)d(w)idf(w)\,, \tag{2}$$

where $q(w)$ (respectively $d(w)$) is the number of words w in q (respectively in d), and $idf(w)$ the negative log of the proportion of the stored frames that contain the visual word w. The few objects with the best scores are kept as candidates for geometric verification.

From a computational point of view, we reduced the complexity of repeating this algorithm at each frame using incremental queries. We keep the scores of objects found in the previous frame, and update them with features that appeared or disappeared. The complexity of the query therefore depends on the number of feature addition or subtraction rather than the total number of features present on the current view.

3.7 Geometric Verification

Our algorithm builds a list of candidates for geometric verification. It is initialized with the set of objects successfully tracked on the previous frame. Then, the list is extended with at most three other candidates selected by their query score (Eq. 2).

For each object in the list, our system tries to match object and frame features. If the candidate was not detected on previous frame, correspondences are established based on the tracks index values (Sec. 3.5). Otherwise, correspondences are propagated from previous frame. The system also attempt to establish new correspondences with the object, but it uses for geometric verification at most 20% of new correspondences. Doing so controls the ratio of outliers.

Once the set of potential correspondences is created, the geometry consistency is verified. Each object has a geometric model, in our implementation either homography or epipolar constraints. For detection, the RANSAC [4] algorithm handles the potentially high outlier rate. During tracking, the outlier rate is controlled, and the least median of squares (LMedS) algorithm can replace RANSAC.

4 Results

We present in this section the experiments we conducted to evaluate our system. We focused on retrieval and tracking capabilities. We finally present a toy application demonstrating that our system fits augmented reality requirements.

4.1 Retrieval Evaluation

The goal of this experiment is to evaluate our system's ability to retrieve objects visible in a video stream. To do so, we ignore the geometric verification stage and concentrates on the best ranked candidate returned by the query.

We differentiate two scenarios. If the time to enter the objects in the database is not an issue, it is possible to include their views in the two clustering stages. This yields good performance, at the cost of a longer and less flexible

	Success/Direct	Success/Stabilized
untrained	2240	2351 (+5.0%)
trained	2389	2497 (+4.5%)

Table 1: To evaluate performance, we count the number of frames correctly retrieved for a test sequence. The rows correspond to the scenarios described in Sec. 4.1. The column Success/Direct contains results obtained without using our method. The column Success/Stabilized presents the results when using our approach. This table clearly shows that, for both scenarios, our approach improves performance.

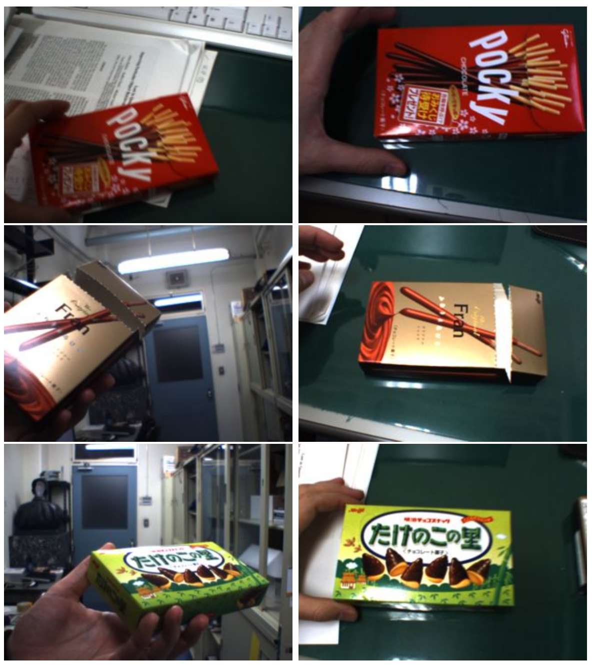

Figure 3: Pairs of query (left) and retrieved (right) images that our approach made possible to retrieve. The direct tree indexing approach failed to handle these frames, as opposed to our method. In the untrained scenario, our method improves the number of successfully retrieved frames of about 5%, as detailed by Sec. 4.1 and table 1. These three pairs are selected among these 5%. In the case of the first row, motion blur causes SIFT detector instability. In the second case, specular reflections alter the object appearance. In the third case, the perspective distortion and the moving specular reflection perturb retrieval if no tracking information is used.

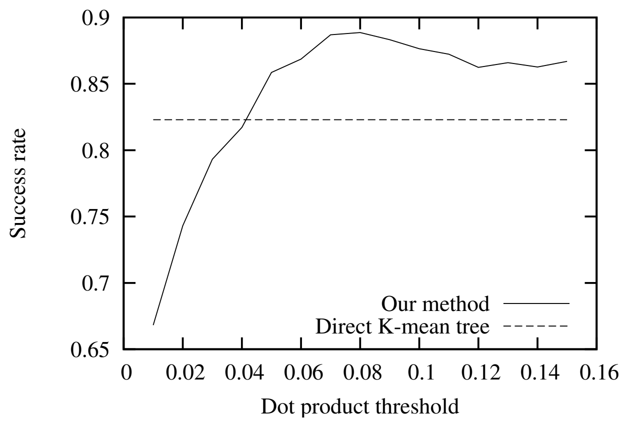

Figure 4: Changing the threshold that stops histogram merging during the second clustering phase. If the threshold is chosen above 0.06, our method improves retrieval rate by 4% to 6%. This graph was obtained in the untrained scenario, as described in Sec. 4.1.

procedure to add the objects to the database. Because some applications can not tolerate such a long process, we tested our method with and without including the target objects in the learning data. We call the tested scenarios trained and untrained.

Evaluation was conducted on video sequences showing 13 test objects, three of which are depicted by Fig 3. The image resolution used is 640×480. The first sequence does not show the test objects. It is used to build the visual vocabulary. The second sequence shows the objects in turn. Thirteen frames are manually added to the database. The third sequence is a test sequence. It also shows the objects, at most one at a time. Ground truth reference is obtained by manually noting the object visible on each frame. To test a scenario, we use every frame of the test sequence as a query to the database. We count the number of frames in which the first result matches the ground truth.

In the trained scenario, the visual vocabulary is built using both the first and the second sequences. It implies that the resulting K-mean tree is more complete. The untrained scenario does not use the second sequence for building the visual vocabulary. In this case, the system has to deal with previously unseen features.

Each scenario is tested twice. Once using directly the leaves of the K-mean tree as index keys, and once using our approach.

Table 1 presents the evaluation results. In both scenarios, our method improves retrieval. The results obtained in the trained scenario show that even when quantization errors are avoided by using the target objects to create the visual vocabulary, our method can still learn some remaining variability and provide a performance gain of about 4.5%.

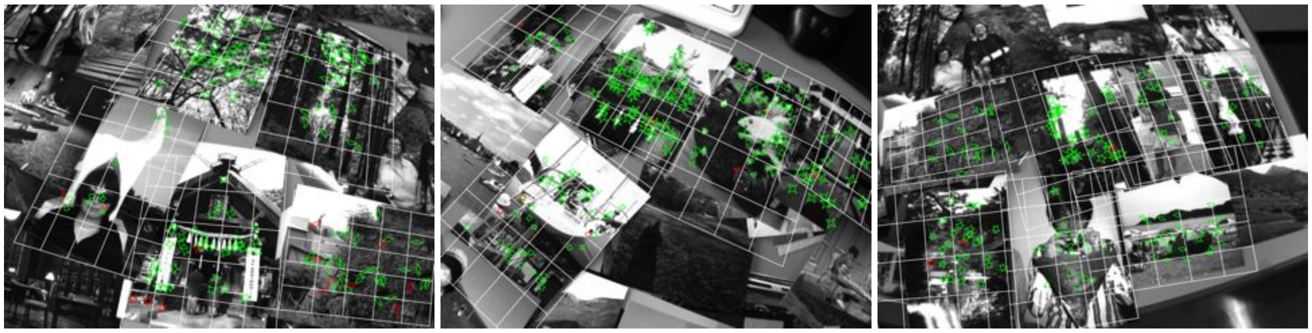

Figure 5: Tracking photographs on a desk. The frames are selected from a test sequence in which the user moves a camera above a desk on which lies several tens of photographs. The system recognizes and track the pictures when they enter the field of view. The white grid represent the detected homography. The darker marks show the newly established correspondences, and the brighter marks the ones propagated from previous frame. The shape of the marks are unique to each object.

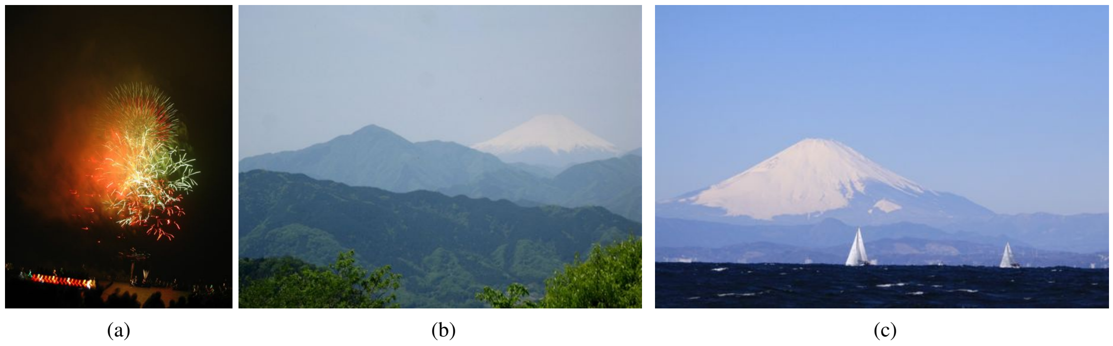

Figure 6: Cases of failure. The system fails to handle these images, due to the specific nature of the texture (a), low contrast due to atmospheric effects (b and c), and poorly discriminative texture (b and c). In total, only 10 pictures out of the 325 ones can not be detected effectively. Half of them are not recognized at all, while the three others are detected only on unoccluded, sharp, and frontal views.

4.2 Tracking Objects

For this experiment, we printed 325 photographs. We entered the pictures one by one into the system, by capturing them on a uniform background with a handheld video camera. It is important to note that once the learning stage is done, adding a new picture to the system is perceptually instantaneous. The user points the target to the camera, clicks, and tracking can start.

Based on the populated database, the system is able to recognize and track randomly chosen pictures. The recognition delay is short, typically 2-3 frames. Once detected, the photographs are tracked, subject to neither drift nor jittering, as depicted by Fig 5.

During the training stage, we transformed 125 pictures in their digital form with random homographies to generate synthetic views, out of which about 5 million features were extracted. We recursively applied K-mean clustering with $K = 4$, stopping at a maximum depth of 8 or when less than 32 descriptors remained in a branch. The resulting tree has 85434 nodes, 63955 of which are leafs. During the second training phase, 655970 histograms were extracted from new synthesized views. The agglomerative clustering process produced 39749 histograms.

The tree and cluster set produced during training allow our system to efficiently establish correspondences between an input view and the objects stored in the database. It is interesting to observe that the system can deal with objects that have not been used for training. We verified this behavior by populating the database with 200 unseen objects, in addition to the 125 ones used for training. Our system kept its performance and could successfully detect and track almost all of the 325 targets, except a few.

Out of 325 pictures, the system fails to detect only 10: 6 among the learned 125 image set, and 4 among the 200 other ones. A few of these pathological pictures are illustrated by Fig. 6.

In our experiments, the database contains 70185 keypoints distributed over 325 pictures. Objects have typically 100 to 300 features each, as depicted by Fig 7.

Fig. 8 depicts the ambiguity of keypoints. The histogram shows that, within our database, most of the keypoints have a quantized appearance that occurs only a few times. There-

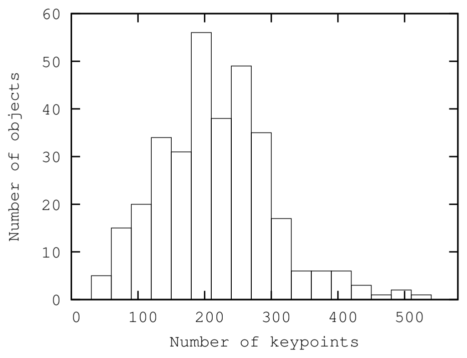

Figure 7: This histogram shows how many features are detected on indexed pictures.

fore, they provide a strong support to the retrieval process. However, Fig. 8 also shows that a few descriptors appear very often. These points are less discriminative but can still bring information for registration.

4.3 Application to Augmented Reality

To demonstrate our system's suitability to AR applications, we used it to augment pictures with virtual drawings. The database of target objects contains the 325 photographs mentioned in the previous section. When the system recognizes a known picture, it overlays it with its virtual counterpart warped with the appropriate homography.

As depicted by Fig. 9, our method's stable tracking yields convincing augmented reality, despite hazards such as viewpoint changes, illumination changes, partial occlusion, camera defocus, and specular reflection. The frames of figures 9 and 10 were produced directly and in real-time by our system fed with a live video stream (except cropping).

As illustrated by Fig. 10, when several photographs appear in the field of view, our system augments them as long as they appear large enough on the input image. Since an homography is computed for each target picture, the system can augment them even if they move independently.

The frame rate of our system is typically 6-8 frames per second. The computationally heaviest component in our implementation is the SIFT feature detector, despite its implementation on GPU.

5 Conclusion

In this paper, we presented an image retrieval approach to multiple object tracking. We demonstrated its effectiveness and scalability by running experiments on more than 300 target objects. Our system is user friendly because it is responsive and fully automated. For example, augmenting a new object simply amounts to pointing the camera at it and clicking. The augmentation starts immediately, and further

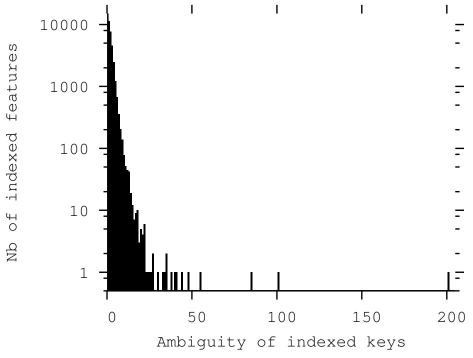

Figure 8: This histogram shows the ambiguity of the index dictionary. Most features are assigned to a unique index key. There is a small number of ambiguous features that appear very often. Because the number of index features ranges from 1 to 11213, we chose a logarithmic scale.

detection is 100% automatic. Our system can process live video streams, and is robust to partial occlusion, viewpoint changes, illumination effects, and other hazards.

These properties make our approach ideal for augmented reality applications that overlay virtual elements on real objects. Possible applications include: Animating pages of books, tagging virtual messages on real walls, virtually annotating objects for maintenance tasks, augmenting card games with virtual scenes, and augmenting panorama views with landmark names.

In future work, we aim to reduce our system's dependency on texture. Currently, only very textured objects can be detected easily. Taking into account the geometric relationship of keypoints could extend indexing and retrieval to printed text or uniform objects with a specific 3-D shape, such as a chair or a tripod.

References

[1] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, pages 221–255, March 2004.

[2] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, 2006.

[3] M. Fiala. ARTag, a fiducial marker system using digital techniques. In *Conference on Computer Vision and Pattern Recognition*, pages 590–596, 2005.

[4] M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications ACM*, 24(6):381–395, 1981.

[5] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Fourth Alvey Vision Conference, Manchester*, 1988.

[6] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, volume I of *LNCS*, pages 304–317, oct 2008.

[7] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana. Virtual Object Manipulation on a Table-Top AR Environment. In *International Symposium on Augmented Reality*, pages 111–119, 2000.

[8] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, October 2005.

[9] V. Lepetit, J. Pilet, and P. Fua. Point matching as a classification problem for fast and robust object pose estimation. In *Conference on Computer Vision and Pattern Recognition*, Washington, DC, June 2004.

[10] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 20(2):91–110, 2004.

[11] B. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

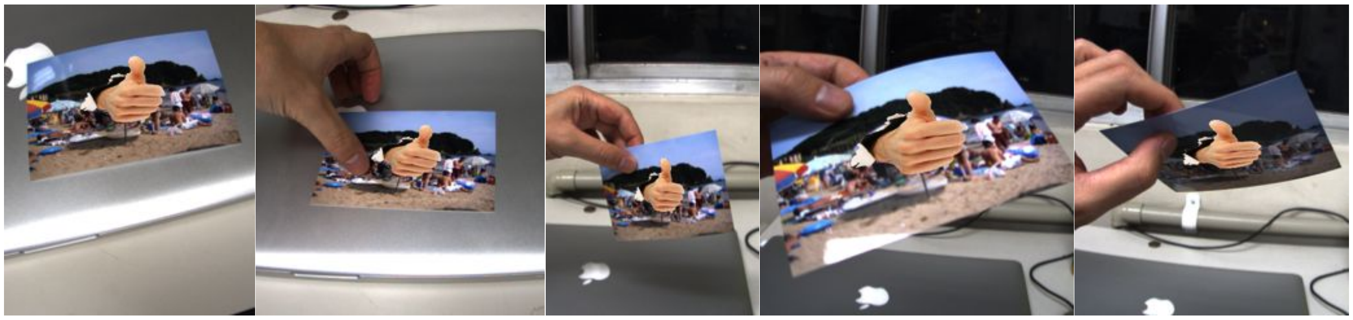

Figure 9: In these frames selected from a longer sequence, our system automatically augments the picture with a virtual hole through which a hand passes. The virtual hand appears stable on the moving photograph, despite illumination changes, partial occlusion, camera defocus, and specular reflection.

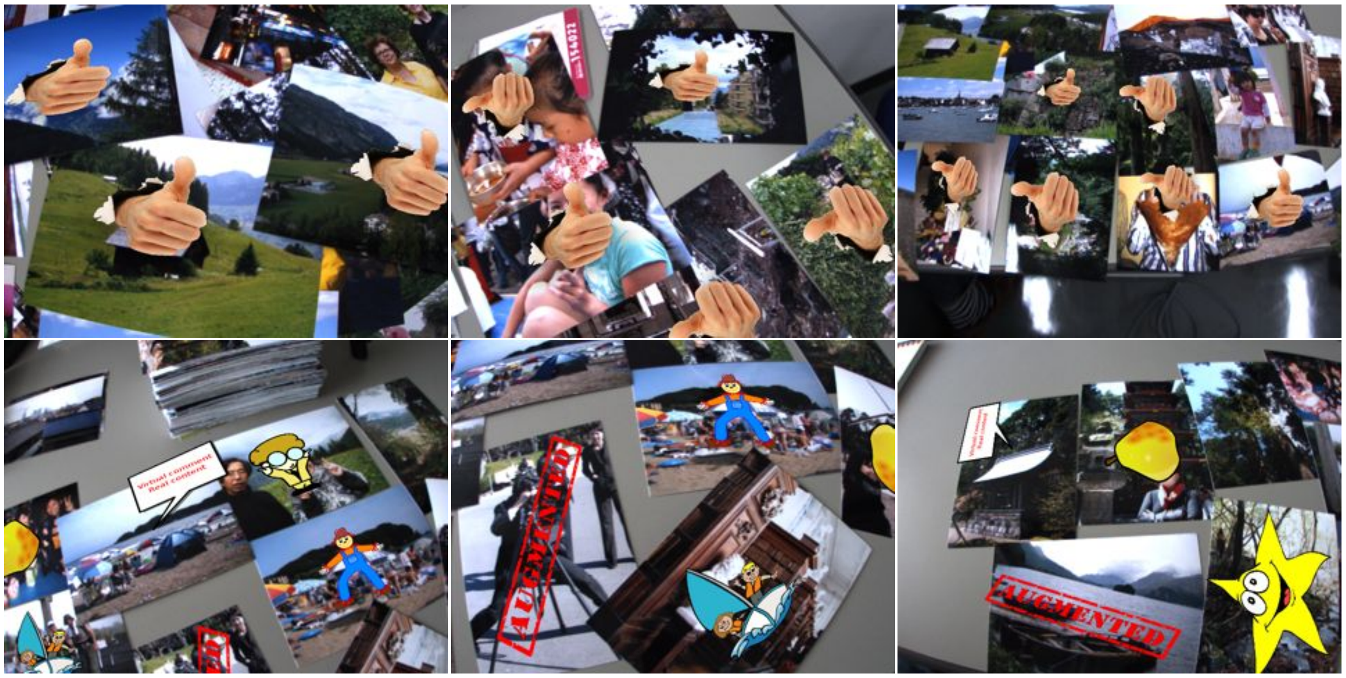

Figure 10: Augmenting multiple objects simultaneously. Our system retrieves, track, and augment the pictures lying on the desk. On the first row, all the objects are augmented with the same virtual content. On the second row, the virtual elements vary from object to object. In these examples, 3 to 6 pictures are tracked and augmented simultaneously.

[12] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *British Machine Vision Conference*, pages 384–393, London, UK, September 2002.

[13] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Conference on Computer Vision and Pattern Recognition*, 2006.

[14] Š. Obdržálek and J. Matas. Sub-linear indexing for large scale object recognition. In *British Machine Vision Conference*, 2005.

[15] M. Ozuysal, P. Fua, and V. Lepetit. Fast Keypoint Recognition in Ten Lines of Code. In *Conference on Computer Vision and Pattern Recognition*, Minneapolis, MI, June 2007.

[16] Y. Park, V. Lepetit, and W. Woo. Multiple 3d object tracking for augmented reality. In *International Symposium on Mixed and Augmented Reality*, pages 117–120, 2008.

[17] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Conference on Computer Vision and Pattern Recognition*, 2007.

[18] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Conference on Computer Vision and Pattern Recognition*, 2008.

[19] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, 2006.

[20] C. Scherrer, J. Pilet, V. Lepetit, and P. Fua. Souvenirs du monde des montagnes. *Leonardo, special issue on ACM SIGGRAPH*, 42(4):350–355, 2009.

[21] J. Shi and C. Tomasi. Good features to track. In *Conference on Computer Vision and Pattern Recognition*, Seattle, June 1994.

[22] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, Oct. 2003.

[23] Sony Computer Entertainment. The eye of judgment, 2007.

[24] Total Immersion. Topps 3d live baseball cards, 2008.

[25] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose Tracking from Natural Features on Mobile Phones. In *International Symposium on Mixed and Augmented Reality*, Cambridge, UK, Sept. 2008.

[26] D. Wagner, D. Schmalstieg, and H. Bischof. Multiple target detection and tracking with guaranted framerates on mobile phones. In *International Symposium on Mixed and Augmented Reality*, Orlando, USA, 2009.

[27] C. Wu. A GPU implementation of David Lowe's scale invariant feature transform, 2008.