# Client-side Mobile Visual Search

Andreas Hartl, Dieter Schmalstieg and Gerhard Reitmayr
*Institute for Computer Graphics and Vision, Graz University of Technology, Inffeldgasse 16, Graz, Austria*

Keywords: Visual Search, Mobile Phone, Descriptors, Vocabulary Tree, Geometric Verification, Augmented Reality.

Abstract: Visual search systems present a simple way to obtain information about our surroundings, our location or an object of interest. Typically, mobile applications of visual search remotely connect to large-scale systems capable of dealing with millions of images. Querying such systems may induce considerable delays, which can severeley harm usability or even lead to complete rejection by the user. In this paper, we investigate an interim solution and system design using a local visual search system for embedded devices. We optimized a traditional visual search system to decrease runtime and also storage space in order to scale to thousands of training images on current off-the-shelf smartphones. We demonstrate practical applicability in a prototype for mobile visual search on the same target platform. Compared with the unmodified version of the pipeline we achieve up to a two-fold speed-up in runtime, save 85% of storage space and provide substantially increased recognition performance. In addition, we integrate the pipeline with a popular Augmented Reality SDK on Android devices and use it as a pre-selector for tracking datasets. This allows to instantly use a large number of tracking targets without requiring user intervention or costly server-side recognition.

## 1 INTRODUCTION

Visual search is a way of obtaining information about objects in the proximity of the user by taking an image of the object and using the image to index into a database of known objects. With advancements in processing power, screen size and connectivity, mobile devices such as smartphones or tablets have become an interesting platform for this kind of service. It is important to note that in this case, mobile visual search may replace standard input methods up to a certain degree. This means, that information retrieval may take place considerably faster than in a traditional keyboard or touchscreen-driven setup. Today, mobile visual search is available through services like Google Goggles[1] or kooaba[2], dealing with large numbers of categories or classes such as products, logos, printed text, but also places and faces. While the former is available as an application on major mobile platforms, the latter can be queried through a web API, processing a given image.

Currently server-side visual search operation exhibits considerable delays. This may be due to incomplete coverage of suitable mobile networks (3G onwards) but also due to delays in the recognition engine, which are independent of the connection quality. Such delays can severely harm usability or even lead to complete rejection of the application by the user.


Figure 1: Mobile prototype: top-left: client-side visual search, top-right: tracking and augmenting an image target recognized by client-side visual search, bottom-left: fast switching of tracking datasets with the local pipeline (268 ms); bottom-right: slow recognition using the server-side approach (1984 ms).

We seek to mitigate this situation by performing visual search directly on the mobile device. Consequently, the time required for getting an initial result can be considerably reduced. In this work we present

[1]http://www.google.com/mobile/goggles
[2]http://www.kooaba.com

Hartl A., Schmalstieg D. and Reitmayr G..
Client-side Mobile Visual Search.
DOI: 10.5220/0004672901250132
In *Proceedings of the 9th International Conference on Computer Vision Theory and Applications* (VISAPP-2014), pages 125-132
ISBN: 978-989-758-009-3

a visual search pipeline for embedded devices and extensively evaluate it w.r.t. recognition rate and runtime. We also provide a realistic comparison with kooaba, indicating the benefits of the proposed solution. Finally we use this pipeline to overcome the limit of a state-of-the-art solution for mobile Augmented Reality on the number of tracking targets, extending its scale significantly without introducing huge delays caused by server-side recognition.

## 2 RELATED WORK

Objects captured with mobile phone cameras may differ largely in appearance when compared with images obtained in a controlled environment. Consequently, local image features are a reasonable choice for representation, abstracting from custom acquisition conditions. Local image features typically require initial keypoint localization and can be divided into two broad groups. While the first group can be represented as a feature-vector (e.g., SIFT (Lowe, 2004), SURF (Bay et al., 2008)), the second group is computed from pixel differences and stored as a binary string. VLAD (Jégou et al., 2010) is a low-dimensional descriptor designed for large-scale retrieval employing product quantization for compression. However, this requires pre-computing SIFT descriptors, which is slow on current mobile devices. Binary feature descriptors like BRIEF (Calonder et al., 2010), ORB (Rublee et al., 2011), BRISK (Leutenegger et al., 2011) and FREAK (Alahi et al., 2012) can be more efficiently computed and matched. Although they take up only a fraction of the space of traditional approaches, they are in general not as accurate. A recent approach called BinBoost (Trzcinski et al., 2013) finds a low-dimensional but highly discriminative binary descriptor using supervised learning. The resulting descriptors feature accuracy comparable to floating point descriptors, however, the generation process is of computational complexity similar to SIFT.

Recognition with local features can be realized by feature matching and subsequent robust (e.g., RANSAC (Fischler and Bolles, 1981)) verification of the spatial layout by a suitable model. For a larger number of images, an approximation of nearest neighbor computation or quantization of features is required (Marius Muja and Lowe, 2009). The latter is realized in the popular bag of words model (BOW), where features are quantized using a vocabulary of visuals words obtained by clustering all feature descriptors (Sivic and Zisserman, 2003). Images are then classified using a suitable scoring scheme (e.g., TF-IDF). However, the vocabulary can become very large, making both storage and retrieval infeasible. By hierarchically clustering the available features, the BOW model is applicable to problems of larger scale (Nister and Stewenius, 2006).

Performance can be further enhanced by taking into account the context of local features (Wang et al., 2011). Although this gives better recognition performance, the overhead in memory consumption is prohibitive in a mobile context. Initial results obtained by the vocabulary tree may be improved using tree-based re-ranking as an additional step before performing geometric verification (Tsai et al., 2010). With a larger number of classes, the dominant factor is the size of the inverted index in the vocabulary tree, which can be compressed (Chen et al., 2010).

It must be noted that binary descriptors generally violate the assumption that points can be represented by cluster centers (Trzcinski et al., 2012). This causes lower performance compared with floating point descriptors in typical scenarios. This can be overcome by randomizing at the cost of additional overhead.

The aim of previous work in mobile visual search was mainly to reduce the amount of data that needs to be transferred to a server performing the actual search operation (Girod et al., 2011) (Ji et al., 2011). This applies to the standard pipeline using the vocabulary tree, but also to alternative approaches, which convert the feature-vector to a binary representation (Zhou et al., 2012) or perform hashing (He et al., 2012). Compressing keypoint locations (Tsai et al., 2009) or using special descriptors further help to reduce transmission time (CHOG (Chandrasekhar et al., 2012)). Still, the initial latency caused by current mobile networks may degrade usability, which is critical in a mobile context.

Current services for visual search have large delays with only little dependence on image resolution (see Section 4.3). Consequently, it seems interesting to investigate visual search from a client-side perspective. We choose the traditional vocabulary tree pipeline for reasons of efficiency, extensibility and popularity. However, we neither transfer the image nor descriptors to a server and perform all processing locally on the mobile device.

Prior art closest to the proposed approach is given by (Henze et al., 2009). They use heavily optimized local features that are known to sacrifice scale-invariance. The authors only provide a user study on the performance of the system that deals with a rather small number of images. In contrast, in our work we modify selected parts of the pipeline to account for special requirements of mobile setups such as limited processing power and storage capabilities, but

also to allow better scaling to a larger number of images. We provide an extensive evaluation of standard datasets with current-off-the-shelf hardware. Thus we describe a system that is half-way between an online visual search solution and a real-time system. Performing the search locally on the device allows for instant responses, while we are able to limit the memory consumption on current off-the-shelf smartphones for image databases of reasonable size.

# 3 MOBILE APPROACH

The major goal of performing mobile visual search on the client is to reduce the large round-trip time of current server-side solutions. Runtime is a very critical factor for mobile applications, and failure to deliver in this area may lead to immediate rejection by the user. Due to constraints in processing power and memory, it is not possible to duplicate a conventional server-side solution onto a mobile device. This also means that the scale of a mobile solution will be considerably smaller than a server-side system, as all information needs to be stored on the device itself. The size of applications packages is also critical, as they are typically downloaded by the device over 3G or Wi-Fi networks. Consequently, we need to keep both runtime and storage requirements at a reasonable level so that the problem remains computationally feasible on current mobile devices. With these considerations in mind, we first implemented a suitable pipeline for visual search and ported it to mobile devices. We then added various modifications so that the pipeline can be used in a realistic scenario employing a large number of image classes or categories, still working in instant time entirely on the mobile device.

## 3.1 Overview

Our pipeline largely follows the standard concept for visual search. We perform keypoint detection and feature extraction on an input image resized to a desired maximum dimension and hand over this data to a vocabulary tree structure for initial classification. For reasons of efficiency, we perform flat scoring at the leafs. In this step, we pipe each feature descriptor down the tree, accumulate normalized visit counts stored during training for each class and weight them by the corresponding entropy. The final result is reported as the index which corresponds to the maximum of the accumulation vector.

We improve our results in a subsequent verification step with a suitable number of candidates (see Figure 2). We serialize both tree data and keypoint/feature descriptor data created during training. While the first is kept in main memory for reasons of speed, the latter is read on demand from flash memory during geometric verification. Since we mainly target planar objects in this work, we employ robust homography estimation using RANSAC to re-rank the list obtained from the vocabulary tree (Hartley and Zisserman, 2003).

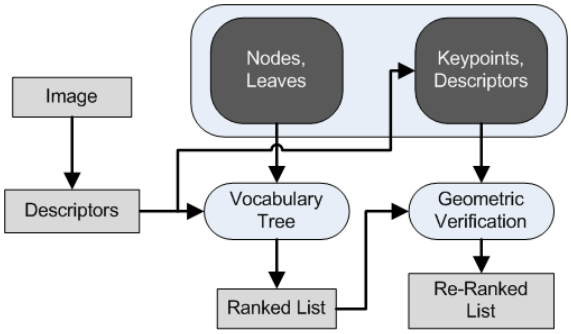


Figure 2: Overview of a local visual search pipeline: image descriptors are extracted and initially classified using a vocabulary tree built from suitable training data; this result is subsequently refined during geometric verification.

## 3.2 Modifications for Mobile Application

We made various enhancements to the standard approach for visual search to improve both runtime and memory requirements. Descriptor computation is a critical task in this type of application, as it tends to have a comparatively large runtime. We modified the current implementation of OPEN-SURF (Evans, 2009) by speeding up integral image computation, but also by employing modifications in the final step of descriptor computation. More specifically, we use a grid-size of 3x3 for 36 dimensions in the feature vector. This yields considerable savings in runtime during descriptor computation and geometric verification, but also in terms of storage. Memory consumption is critical, as it influences both installation time and startup time. We reduce requirements in main memory (tree structure) but also in flash memory (keypoints/descriptors) by using half-precision float values throughout the pipeline. In particular, this affects keypoint data and vocabulary tree data (see Figure 3). In addition, we compress descriptor data by

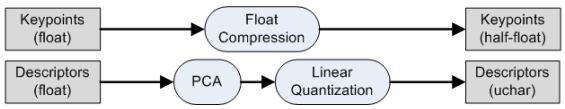


Figure 3: Pipeline for compressing local features: keypoint locations are compressed into half precision values and descriptors are linearly quantized with optional PCA.

linear quantization into a single byte per dimension. Optionally, we perform PCA (Pearson, 1901) to reduce the number of initial dimensions before linear quantization. We also employ compression of inverted index data by recursive integer coding (Moffat and Anh, 2005), targeting specifically the burden on main memory caused by a large number of image classes. It must be noted that we decompress all data on-the-fly during program execution, working solely on the mobile CPU. We will evaluate this client-side pipeline w.r.t. runtime and memory consumption in detail in the next section.

# 4 EVALUATION

We first evaluate the local pipeline w.r.t. recognition performance, runtime and memory requirements directly on a Samsung Galaxy S3 mobile phone. This is a an off-the-shelf smartphone with an ARM-Cortex A9 CPU (up to 1.4 GHz) and 1 GB of main memory running Android. Information about the performance on this device allows to estimate behavior on most smartphones or tablets currently in use. We then evaluate recognition performance using the commercial recognition service kooaba. This allows to clearly show the behavior of our pipeline compared to a state-of-the-art solution for image retrieval.

## 4.1 Metrics and Datasets

In general, we report recognition rate (relative amount of candidates classified correctly), runtime (descriptor computation, vocabulary tree, geometric verification) and the size of serialized data for the vocabulary tree and keypoints/descriptors. If not noted otherwise, runtime is given in milliseconds (ms) and memory usage is reported in megabytes (MB). Based on informal experiences with acceptable recognition latency, we set the upper runtime limit of a local pipeline at approximately 500 ms on current off-the-shelf devices.

We use several datasets in our evaluation (see Table 1). The posters dataset was created mainly to be able to evaluate behavior with various image transformations and serves for initial testing. The Missouri (Wang et al., 2011) and in particular the Stanford (Chandrasekhar et al., 2011) dataset represent typical objects and operating conditions encountered in mobile visual search. Especially the latter is interesting in our context, as it contains more than 1000 classes. Finally, the UK-Bench[3] dataset is included here to be able to evaluate the behavior of the pipeline with a larger number of image classes. This dataset is not very representative for mobile visual search, however, as it also contains different views of non-planar objects, sometimes captured on very textured background. Since there is no test set given, it requires computation of a different metric for evaluation (ukyscore).

[3]www.vis.uky.edu/ stewe/ukbench

Table 1: Most datasets used in our evaluation represent typical operating conditions for mobile visual search; the ukbench dataset allows evaluations of larger scale.

| Name | Categories | Images | Light | Clutter | Distortion |
|---|---|---|---|---|---|
| Posters | 11 | 11 | x | | x |
| Missouri Mobile | 5 | 400 | x | x | x |
| Stanford MVS | 8 | 1193 | x | x | x |
| UK-Bench | 2550 | 10200 | | x | x |

Table 2: Local pipeline: performance and runtime of various local features on the posters dataset; R...resolution, D...type of feature, P...recognition performance, F...feature computation, T...vocabulary tree, V...geometric verification.

| R train/test | D | P | F [ms] | T [ms] | V [ms] | Sum [ms] |
|---|---|---|---|---|---|---|
| 320/320 | SIFT | 0.8170 | 1269 | 21 | 180 | 1470 |
| 320/320 | SURF | **0.8568** | 604 | 9 | 58 | 671 |
| 320/320 | OSURF | 0.7829 | 208 | 8 | 59 | 275 |
| 320/320 | OSURF36 | 0.7784 | **126** | **4** | **43** | **173** |
| 320/480 | OSURF36 | 0.8409 | 208 | 6 | 34 | 248 |

Although the scale of these experiments is relatively small compared to server-side systems from literature, it seems to be a common practice to create larger datasets by insertion of an arbitrary amount of distractor images. In contrast to our evaluation methodology, a comparison is much more difficult in these cases. The posters dataset is available from the authors upon request.

## 4.2 Evaluation of the Local Pipeline

We first evaluate the system to determine suitable parameters for feature descriptors and geometric verification. Then, we determine the influence of compression on recognition rate, runtime and memory consumption. In a final step, we evaluate our pipeline with a considerably larger number of classes. This allows to come up with a clear statement on performance and practical usability on current mobile hardware.

### 4.2.1 Descriptors and Geometric Verification

We evaluated the influence of the number of candidates used in geometric verifcation on recognition rate and runtime for the posters dataset (see Figure 5). We evaluate various feature descriptors for use in our pipeline with the posters dataset (compression switched off). In order to facilitate the comparison of

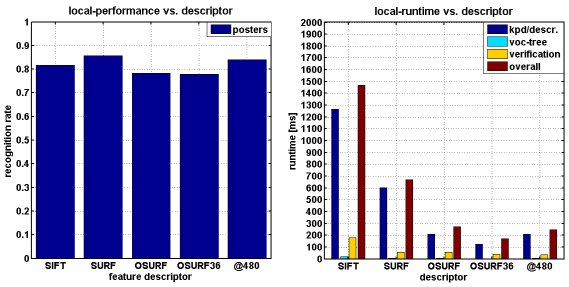


Figure 4: Local pipeline: performance and runtime of various feature descriptors on the posters dataset; our modified SURF descriptor provides reasonable performance but takes up less runtime compared to the unmodified variants.

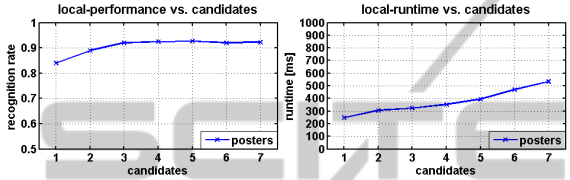


Figure 5: Local pipeline: effect of geometric verification on the posters dataset; performance saturates around three candidates

results, we also evaluate SIFT and SURF[4]. We use a maximum extension for the input image of 320 pixels and limit the maximum number of keypoints to 256. Geometric verification is enabled, but configured to just use one candidate. From Table 2 and Figure 4, it is evident that we obtain reasonable recognition performance with the evaluated feature types. However, runtime of certain setups such as SIFT or SURF is prohibitive for current mobile devices considering our runtime budget of approx. 500 ms. Our modified OPEN-SURF descriptor with just 36 dimensions takes only a fraction of runtime compared to SURF. However, recognition rate is around 10% lower. As runtime is comparatively low, we can also process images of higher-resolution (e.g., 480 pixels). In this case, we can roughly match the recognition rate of SURF. Still, runtime is less than 50% compared to SURF. In particular, runtime for geometric verification is shorter, which is also due to the reduced size of the descriptor. So, it is possible to use more candidates for a given runtime budget. We see that runtime scales approx. linearly in the number of candidates. Similarly, recognition rate improves with an increasing number of candidates. Although performance seems to saturate starting with three candidates for the posters dataset, we choose to use six candidates for our modified OPEN-SURF descriptor, as runtime is still around 500 ms. Based on our runtime budget, we can only compare the performance

[4]http://opencv.org

Table 3: Local pipeline: effect of compressing keypoints and descriptors; N...name of dataset, D...type of feature, F...feature computation, P...recognition performance, T...tree, V...geometric verification.

| N | D | P | F [ms] | T [ms] | V [ms] | T [MB] | F [MB] |
|---|---|---|---|---|---|---|---|
| Post. | SURF | 0.8568 | 604 | 9 | **58** | 0.98 | 0.79 |
| Post. | OSURF36 | 0.9204 | 212 | **6** | 135 | 0.49 | 0.41 |
| Post. | OSURF36C | **0.9329** | **211** | 16 | 136 | **0.15** | **0.12** |
| Miss. | SURF | 0.6751 | 742 | 15 | **68** | 33.2 | 27.5 |
| Miss. | OSURF36 | 0.8623 | 248 | **9** | 180 | 16.9 | 14.6 |
| Miss. | OSURF36C | **0.8759** | **225** | 26 | 151 | **5.14** | **4.61** |
| Stanf. | SURF | 0.6550 | 640 | 14 | **58** | 84.1 | 72.2 |
| Stanf. | OSURF36 | 0.6940 | 216 | **9** | 134 | 36.4 | 33.5 |
| Stanf. | OSURF36C | **0.7000** | **216** | 26 | 134 | **11.1** | **10.5** |

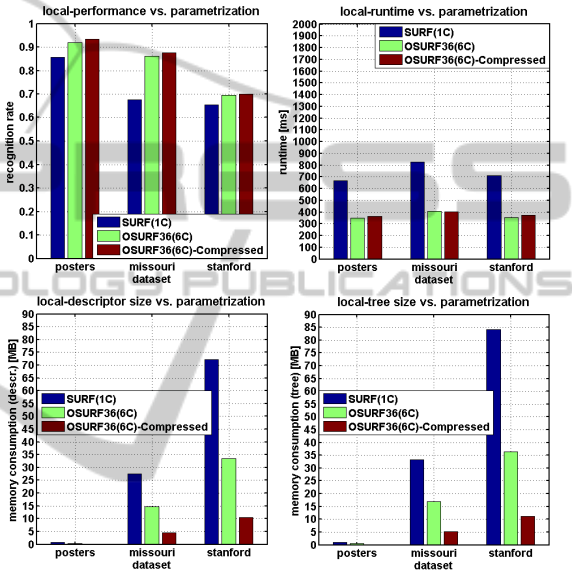


Figure 6: Local pipeline: detailed evaluation of compression on various datasets; compared with standard SURF, up to 85% of storage space can be saved at negligible runtime overhead and slightly increased performance

of our modified descriptor to SURF using a single candidate for verification. In Table 3 and Figure 6, we present the results of this setup with the Missouri and Stanford datasets. Compared with the baseline, our modified OPEN-SURF descriptor offers 5%-20% better performance in this setup, but takes up only half of the runtime of SURF. Still, memory requirements for tree and descriptors are comparatively high, especially for the Stanford dataset.

### 4.2.2 Compression

In order to tackle increasing memory requirements, we compress both descriptors and the tree structure. The effect of these measures can be seen in Table 3 and Figure 6. Our compression efforts significantly reduce memory requirements, while the effect on runtime is negligible. By employing the proposed mod-

ifications, up to 85% of storage space can be saved over standard SURF. Interestingly, there is a small increase in recognition performance when compression is enabled. This may be due to a reduction in noise caused by our quantization scheme.

#### 4.2.3 Scalability

In this experiment, we determine large-scale performance on the UK-Bench dataset. We perform this test on a Samsung Galaxy S3 smartphone and enable compression of keypoints and descriptors, but also the inverted index stored in the vocabulary tree. From Figure 7, it is evident that our pipeline scales well concerning recognition rate, runtime and main memory consumption. It is possible to manage more than 10000 classes with the current pipeline, using less than 110 MB of total storage space. Only a fraction (approx. 50 MB) of data needs to be loaded into main memory. On the one hand, the overall scores obtained in this experiment are lower than those reported in literature, as our parametrization is targeted towards practical applicability on mobile devices. On the other hand, it does not seem reasonable for this kind of application to train a class for each view of an object. As current mobile devices feature 1-2 GB of main memory and at least 16 GB of flash storage, this purely client-side approach is estimated to be able to handle an amount of images that is around 1-2 magnitudes higher.

### 4.3 Comparison with Kooaba

For this experiment, we uploaded relevant reference images into a single group and deactivated all images not relevant to the current experiment or dataset. We then performed queries over a Wi-Fi internet connection. This can be considered a very optimistic setup compared to current mobile phone networks.

According to initial tests, the query resolution has little influence on runtime and recognition rate. We scale down query images to a maximum extension of 320 pixels, which is rather common for mobile applications. From Figure 8, we see that the posters dataset performs best (approx. 0.7) on kooaba. With the Missouri and Stanford datasets, performance drops by around 10%. Compared to our client-side approach, overall performance per dataset is significantly lower (5-25%).

However, the client-side approach has lower performance for the print category of the Stanford dataset (see Figure 9). As text in general features many keypoints, this drop is likely to be caused by the imposed keypoint limit of our approach. It must be noted that the performance for the landmark category is low for

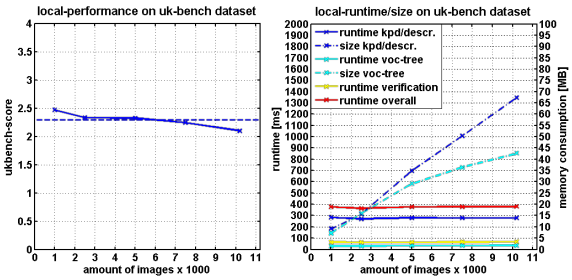


Figure 7: Local pipeline: testing scalability with the UK-Bench dataset on a Samsung Galaxy S3 mobile phone; our pipeline scales well concerning recognition rate, runtime and main memory consumption

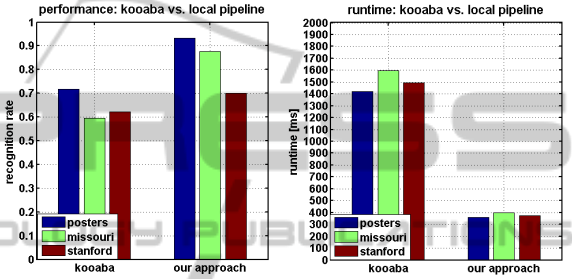


Figure 8: Comparison of server-side mobile visual search (kooaba) and client-side mobile visual search; our client-side solution offers significantly better recogniton and runtime performance

both approaches. This may be caused by the fact that the publicly available training set consists of several images of the same object, each having a separate class. This is not a common application scenario for visual search, however.

For this experiment, runtime of kooaba is around 1500 ms, where the Missouri dataset has a higher runtime than the other two (see Figure 8). For our setup the bottleneck currently seems to lie in the recognition engine itself, rather than connection speed.

All in all, the client-side solution offers significantly better recognition performance on the evaluated datasets compared with a state-of-the-art server-side solution. However, the latter performs better in the print category. A local pipeline giving a result in approximately 500 ms, can therefore compete in recognition performance with a server-side solution which takes three times the runtime.

## 5 MOBILE PROTOTYPE APPLICATION

We built a mobile prototype for Android smartphones and tablets demonstrating client-side mobile visual

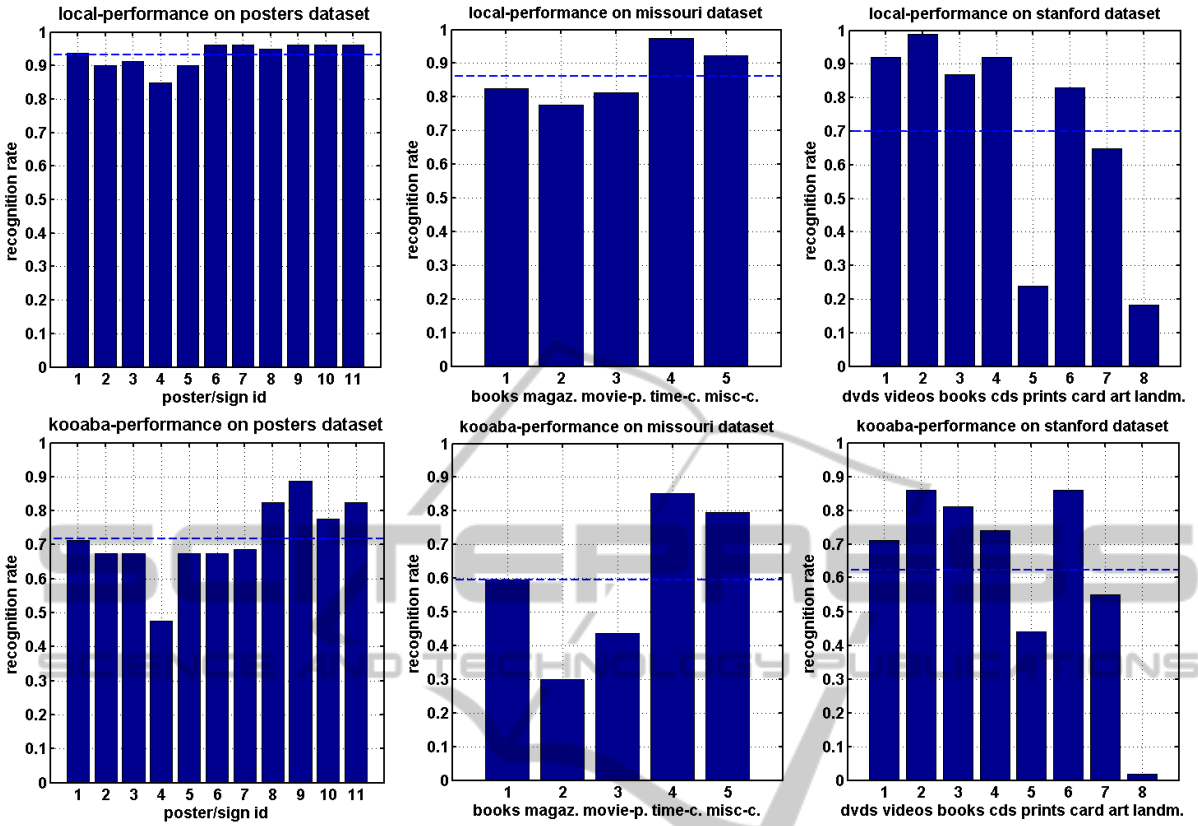


Figure 9: Detailed evaluation for the local pipeline and kooaba; top row: local pipeline, bottom row: kooaba; our client-side approach delivers significantly improved recognition performance for all datasets and categories except print (keypoint limit)

search but also server-side visual search using kooaba (see Figure 1). Similar to popular search engines, we give a list of candidates in the form of preview images, which may be activated to trigger a web-based search in order to get additional information.

We also use our client-side module for visual search to extend the amount of realistically usable tracking targets within the Vuforia SDK[5] for mobile Augmented Reality. In this case, we can instantly select a matching tracking dataset without requiring user intervention or costly server-side recognition. We successfully tested this setup with several hundred image targets. Due to limitations in the SDK, we cannot provide a detailed evaluation, however.

## 6 CONCLUSIONS

We optimized a traditional visual search system to decrease runtime and also storage space in order to scale to thousands of training images on current off-the-shelf smartphones. Compared with a standard pipeline, we achieve up to a two-fold speed-up in runtime, save 85% of storage space and provide substantially increased recognition performance. We compared performance and runtime with the commercial service kooaba and deliver considerably better recognition performance at a fraction of runtime. We demonstrated practical applicability in a prototype for mobile visual search on mobile devices running Android. In addition, we integrated the pipeline with a popular AR SDK and used it as a preselector for tracking datasets. Consequently it is possible to instantly use a large number of tracking targets without requiring user intervention or costly server-side recognition

Improvements could be made on various levels. We treat those as future work and will list them briefly: Descriptor computation should be accelerated further, possibly by using the GPU for part of the processing. This would certainly lead to an even more responsive system, but might also improve recognition rate by relaxing the current limitation on the number of keypoints/feature descriptors. As their size again poses a problem for huge numbers of classes, they should be further compressed (e.g., variable-rate quantization).

[5]https://www.vuforia.com

For current server-based visual search systems it seams reasonable to run a combination of server-side and client-side recognition. Then, the client could be configured to instantly recognize just a small subset of currently popular classes. This would provide both large-scale capability and instant recognition.

## ACKNOWLEDGEMENTS

This work is supported by Bundesdruckerei GmbH.

## REFERENCES

Alahi, A., Ortiz, R., and Vandergheynst, P. (2012). Freak: Fast retina keypoint. In *CVPR*, pages 510–517.

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

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: binary robust independent elementary features. In *ECCV*, pages 778–792.

Chandrasekhar, V.and Takacs, G., Chen, D. M., Tsai, S., Reznik, Y.and Grzeszczuk, R., and Girod, B. (2012). Compressed histogram of gradients: A low-bitrate descriptor. *International Journal of Computer Vision*, 96(3):384–399.

Chandrasekhar, V. R., Chen, D. M., Tsai, S. S., Cheung, N.-M., Chen, H., Takacs, G., Reznik, Y., Vedantham, R., Grzeszczuk, R., Bach, J., and Girod, B. (2011). The stanford mobile visual search data set. In *MMSys*, pages 117–122.

Chen, D. M., Tsai, S. S., Chandrasekhar, V., Takacs, G., Vedantham, R., Grzeszczuk, R., and Girod, B. (2010). Inverted index compression for scalable image matching. In *IEEE DCC*, page 525.

Evans, C. (2009). Notes on the opensurf library. Technical Report CSTR-09-001, University of Bristol.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

Girod, B., Chandrasekhar, V., Chen, D. M., Cheung, N.-M., Grzeszczuk, R., Reznik, Y. A., Takacs, G., Tsai, S. S., and Vedantham, R. (2011). Mobile visual search. *IEEE Signal Processing Magazine*, 28(4):61–76.

Hartley, R. and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2nd edition.

He, J., Feng, J., Liu, X., Cheng, T., Lin, T.-H., Chung, H., and Chang, S.-F. (2012). Mobile product search with bag of hash bits and boundary reranking. In *CVPR*, pages 3005–3012.

Henze, N., Schinke, T., and Boll, S. (2009). What is that? object recognition from natural features on a mobile phone. In *Workshop on Mobile Interaction with The Real World*.

Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311.

Ji, R., Duan, L.-Y., Chen, J., Yao, H., Rui, Y., Chang, S.-F., and Gao, W. (2011). Towards low bit rate mobile visual search with multiple-channel coding. In *ACM MM*, pages 573–582.

Leutenegger, S., Chli, M., and Siegwart, R. (2011). Brisk: Binary robust invariant scalable keypoints. In *ICCV*, pages 2548–2555.

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

Marius Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, pages 331–340.

Moffat, A. and Anh, V. N. (2005). Binary codes for non-uniform sources. In *IEEE DCC*, pages 133–142.

Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: an efficient alternative to sift or surf. In *ICCV*, pages 2564–2571.

Sivic, J. and Zisserman, A. (2003). Video google: a text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477.

Trzcinski, T., Christoudias, M., Fua, P., and Lepetit, V. (2013). Boosting binary keypoint descriptors. In *CVPR*, pages 2874–2881.

Trzcinski, T., Lepetit, V., and Fua, P. (2012). Thick boundaries in binary space and their influence on nearest-neighbor search. *Pattern Recognition Letters*, 33(16):2173–2180.

Tsai, S., Chen, D. M., Takacs, G., Chandrasekhar, V., Vedantham, R., Grzeszczuk, R., and Girod, B. (2010). Fast geometric re-ranking for image-based retrieval. In *ICIP*, pages 1029–1032.

Tsai, S. S., Chen, D., Takacs, G., Chandrasekhar, V., Singh, J. P., and Girod, B. (2009). Location coding for mobile image retrieval. In *MMCC*, pages 8:1–8:7.

Wang, X., Yang, M., Cour, T., Zhu, S., Yu, K., and Han, T. (2011). Contextual weighting for vocabulary tree based image retrieval. In *ICCV*, pages 209–216.

Zhou, W., Lu, Y., Li, H., and Tian, Q. (2012). Scalar quantization for large scale image search. In *ACM MM*, pages 169–178.