Improving document matching performance by local descriptor filtering

Joseph Chazalon, Marçal Rusiñol, Jean-Marc Ogier

To cite this version:

HAL Id: hal-01337262
https://hal.archives-ouvertes.fr/hal-01337262
Submitted on 24 Jun 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Improving Document Matching Performance by Local Descriptor Filtering

Joseph Chazalon*, Marçal Rusiñol†, and Jean-Marc Ogier*
*LSi Laboratory, Université de La Rochelle
Avenue Michel Crépeau
17042 La Rochelle Cédex 1, France
†Computer Vision Center, Dept. Ciències de la Computació
Edifici O, Univ. Autònoma de Barcelona
08193 Bellaterra (Barcelona), Spain

Abstract—In this paper we propose an effective method aimed at reducing the amount of local descriptors to be indexed in a document matching framework. In an off-line training stage, the matching between the model document and incoming images is computed retaining the local descriptors from the model that steadily produce good matches. We have evaluated this approach by using the ICDAR2015 SmartDOC dataset containing near 25 000 images from documents to be captured by a mobile device. We have tested the performance of this filtering step by using ORB and SIFT local detectors and descriptors. The results show an important gain both in quality of the final matching as well as in time and space requirements.

I. INTRODUCTION

Exact document image matching in mobile environments using local descriptors has become in the latest years quite a trend in our community. Applications ranging from model-guided mobile document capture and classification [1], marker-less augmented reality in documents and document retrieval [2] or document stitching and mosaicking [3] are powered by such techniques. At its core, the proposed techniques in the document image analysis domain are no different from classical object recognition scenarios, although many research efforts have been devoted to design accurate and discriminative local descriptors adapted to the specific problem of matching documents.

For instance, Nakai et al. proposed in [2], [4], the Locally Likely Arrangement Hashing (LLAH) method that encodes the word distribution by computing a set of perspective invariant geometric descriptors. In [5], Moraleda and Hull proposed a similar description based also on the way neighboring words are arranged and geometrically distributed.

However, despite the importance of designing new and performant local descriptors specifically tailored for document images, one should also consider the efficiency of such frameworks. Specially in the case in which such applications have to run in mobile devices at real time. Although LLAH performs queries in milliseconds in an indexed dataset of millions of pages, it also requires an incredible amount of RAM memory to store such descriptors which is an unrealistic requirement within today’s mobile devices specifications.

Binary local descriptors such as BRIEF [6] or ORB [7], in combination with efficient indexing schemes like Locality Sensitive Hashing [8] (LSH), allow to run real-time document matching applications in recent smartphone devices. However, such approach presents a couple of drawbacks. On the one hand, such binary descriptors are not so discriminant when compared with state-of-the-art local descriptors like SIFT [9] or SURF [10] for example. On the other hand, even though LSH allows a direct matching approach in real-time, it might not be that scalable when considering large amounts of documents to index in the dataset.

It is worth to note that usually the main bottleneck in terms of memory requirements is the vast amount of extracted local keypoints. Using default parameters, detecting local keypoints in a 300dpi scanned A4 page with either SIFT, SURF or ORB, usually results in thousands and even tens of thousands of keypoints to be indexed. Obviously, one could trim down such amount of keypoints by adjusting the goodness / saliency parameter of the local detector. However, reducing the amount of local keypoints by adjusting such parameter, does not guarantee that the really discriminative local descriptors are kept, just that the most reliable keypoint locations are retained. One can quickly realize that the most reliable keypoint locations in terms of cornerness, contrast or curvature (delivered by ORB, SIFT and SURF respectively) might not correspond to the most discriminative local descriptors. Specially when applying such frameworks in the document domain, where a lot of repetitive patterns (mostly coming from text) have to be expected.

In this paper we propose a simple but effective method aimed at reducing the amount of local descriptors to be indexed in a document matching framework. Inspired by the work by Kurz et al. [11], we propose to build a “training” sequence of images containing one specific instance of the model document to index. In such sequence, the matching between model and images is computed retaining the local descriptors from the model that steadily produced good matches against this training sequence of images. We have evaluated this approach by using the ICDAR2015 SmartDOC dataset containing near 25 000 images from documents to be captured by a mobile device. We have tested the performance of our proposal against the use of both ORB and SIFT local detectors and descriptors. The results show an important gain both in quality of the final matching as well as in time and space requirements.

The paper is organized as follows. In Section II we briefly overview the document matching framework when using local descriptors. Section III introduces the proposed local descriptor
filtering stage. In Section IV, we present our experimental setup and the associated results in Section V. Finally, Section VI presents our conclusions.

II. DOCUMENT MATCHING

We followed a standard architecture for document matching with local descriptors. Given a set of model documents \( D = \{d_1, d_2, ..., d_M\} \) to index, we compute local detectors to end up with \( N \) keypoints \( K = \{k_1, k_2, ..., k_N\} \) for each model document from \( D \). Since we have run our experiments using the SIFT and ORB detectors, the keypoints from \( K \) are at some extend invariant to rotation, scale, illumination and perspective changes. Each keypoint \( k_i \) is then described by a feature descriptor \( f_i \) from either SIFT or ORB descriptors. Such descriptors are then indexed in an inverted file efficiently implemented with the FLANN architecture [12], which either uses KD-trees for SIFT descriptors or LSH [8] for ORB binary descriptors.

For any incoming image from the mobile device, keypoints and local descriptors are extracted as well and matched against the inverted file. In order to produce reliable matches, we use the ratio-test proposed by Lowe [9], in which a match is considered to be correct if the ratio between the nearest and the second nearest local descriptor is above a certain threshold.

From this set of putative matches, a RANSAC [13] step is performed in order to filter out the outlier matches that do not agree geometrically and to find the homography between the recognized model document and its instance appearing in the scene.

III. LOCAL DESCRIPTOR FILTERING

As mentioned in the previous section, the document matching framework relies heavily on the keypoints extracted from each document model. Such approach suffers from two major drawbacks. First, the heuristics used by keypoint detectors are based on local criteria (cornerness, contrast or curvature) and do not guarantee a stable matching over various perspectives or illumination conditions. Second, common keypoint detectors tend to extract many ambiguous elements which are usually filtered along the steps of the process.

Those two claims are illustrated with Figure 1. Figure 1a) shows a document model and its keypoints (small circles) originally extracted from it. We appreciate the presence of keypoints covering text areas and other ambiguous parts of the image. Furthermore, this illustration shows the usage frequency of each keypoint with a color scale: blue and green indicate keypoints which are rarely used to estimate the perspective transformed, while orange and red indicate good supports for this estimation. We can see that an important part of the keypoints retained by the local detector are not relevant for the document matching problem. This is confirmed by Figure 1b), the histogram of usage for each keypoint as a support for the estimation of the perspective transform (i.e. not filtered by the ratio-test and the RANSAC stage).

To improve the selection of robust local descriptors, we followed an approach inspired by the work of Kurz et. al [11]: we propose to build a “training” sequence of images containing one specific instance of the document model to index. In such sequence, the matching between a document model and sample images is computed retaining the local descriptors from the model that steadily produced good matches against this training sequence of images.

More specifically, the steps of the method are the following, given a document model \( d_i \) and a train sequence \( S_{i,j} \):

1) use a local detector to extract an important amount of keypoints \( K_i \) candidates from \( d_i \);
2) build the histogram \( h_i,j \) of keypoints usage (inliers) when matching \( d_i \) (using \( K_i \)) with each image of \( S_{i,j} \);
3) build a new filtered keypoint set \( K_i \setminus t \) with the \( t \) keypoints which exhibit the highest histogram values in \( h_i,j \);
4) use the filtered keypoint set \( K_i \setminus t \).

It is worth noting that the training sequence does not require any special ground truth, except the reference of the document model captured. The selectivity of the matching ensures that only the best local descriptors are retained.

IV. EXPERIMENTAL SETUP

This section presents the dataset, evaluation protocol, metrics we used to evaluate the usefulness of local descriptor filtering.

A. Dataset

Our test dataset is the SmartDOC database for document capture (challenge 1) [14]. It consists of six different document types coming from public databases and five document images per class. The different types have been chosen so that they cover different document layout schemes and contents (either completely textual or having a high graphical content). In particular, the dataset consists of data-sheets documents and patent documents retrieved from the Ohega dataset [15], title-pages from medical scientific papers from the MARG dataset [16], color magazine pages from the PRIMA layout analysis dataset [17], american tax forms from the NIST Tax Forms Dataset (SPDB2) [18], and finally typewritten letters from the Tobaccoc800 document image database [19]. An example of each of those six different document types is shown
The evaluation we conducted compares the original key- point selection scheme based on local detectors heuristics (hereafter named baseline), and the filtering scheme under investigation, detailed in the Section III. We describe here the generation of the test results for each scheme, that we later evaluated with performance measures presented in Section IV-C.

### B. Evaluation Protocol

The generation of the test results for each scheme, that we later evaluated with performance measures presented in Section IV-C.

1) Baseline Scheme: The generation of document matching results using local descriptors selected with default heuristics is performed with the following algorithm, where $D$ is the set of document models, $T$ the set of the different numbers of keypoints to extract, $S_i$ is the set of video sequences (one for each background) for a given document model $d_i$.

```plaintext
for each document $d_i \in D$ do
    for each keypoint value threshold $t \in T$ do
        build $K_{i}^{t}$, a set of $t$ keypoints extracted from $d_i$ with default heuristics
        for each test sequence $s_{i,j} \in S_i$ do
            process $s_{i,j}$, matching each frame against $d_i$ using $K_{i}^{t}$
        end for
    end for
end for
```

2) Filtering Scheme: The generation of document matching results using local descriptors selected by a filtering based on there usage frequency on a training set is performed with the following algorithm, where $D$ is the set of document models, $T$ the set of the different numbers of keypoints to select, $S_i$ is the set of video sequences (one for each background) for a given document model $d_i$.

```plaintext
for each document $d_i \in D$ do
    extract $K_{i}^{\text{init}}$, a set of $\text{init}$ keypoints extracted from $d_i$ with default heuristics
    for each training sequence $s_{i,1} \in S_i$ do
        build $h_{i,j}$, the histogram of matches of $K_{i}^{\text{init}}$ against each frame of $s_{i,j}$
        for each keypoint value threshold $t \in T$ do
            build $K_{i}^{\text{init}} \setminus t$, a set of the $t$ most useful keypoints from $h_{i,j}$
            for each test sequence $s_{i,1} \in S_i, s_{i,j} \neq s_{i,1}$ do
                process $s_{i,j}$, matching each frame against $d_i$ using $K_{i}^{\text{init}} \setminus t$
            end for
        end for
    end for
end for
```

3) Tested Methods: To conduct our experiments, we studied the behavior of the ORB and SIFT detectors and descriptors. We used the following parameters.

- for ORB: $\text{init} = 2 \, 000$, and $T = \{200, 400, 600, \ldots, 2 \, 000\}$
- for SIFT: $\text{init} = 4 \, 000$, and $T = \{1 \, 000, 2 \, 000, 3 \, 000, 4 \, 000\}$
For each method, we extracted 1000 keypoints from each frame of the train and test sequences.

C. Performance Measures

1) Segmentation Accuracy: To assess the quality of the results produced by the variations of each method, we used the Jaccard index measure [21] that summarizes the ability of the different methods at correctly segmenting page outlines while also incorporating penalties for methods that do not detect the presence of a document object in some frames.

Using the document size and its coordinates in each frame, we start by transforming the coordinates of the quadrilaterals of a matching method $S$ and of the ground-truth $G$ to undo the perspective transform and obtain the corrected quadrilaterals $S'$ and $G'$. Such transform makes all the evaluation measures comparable within the document referential. For each frame $f$, we compute the Jaccard index (JI) that measures the goodness of overlapping of the corrected quadrilaterals as follows:

$$JI(f) = \frac{\text{area}(G' \cap S')}{\text{area}(G' \cup S')}$$

where $G' \cap S'$ defines the polygon resulting as the intersection of the detected and ground-truth document quadrilaterals and $G' \cup S'$ the polygon of their union. The overall score for each method will be the average of the frame score, for all the frames in the test dataset.

2) Processing Speed: The processing speed is estimated using the average time required to process each frame of the test set. This later is obtained using instrumented Python code and running all the tests on the same dedicated machine.

3) Memory Impact: The memory gain can be directly deducted from the relative reduction of the size of the set of local descriptor for each document model.

V. EXPERIMENTAL RESULTS

This section presents experimental results for ORB and SIFT local detectors and descriptors, and discusses how the training stage should be performed.

A. ORB

Experimental results exhibit an important performance gain for ORB when filtering local descriptors. Figure 4a) shows an important improvement when progressively removing unnecessary keypoints from the model, until a peak after which the discriminative power of the model starts being hindered. Table II summarizes the values obtained for the Jaccard Index measure. We can see here that a 5-10 factor reduction of the descriptor set size can be achieved almost without harming results’ quality. The processing time is also reduced significantly, with a gain of 25% for thresholds of 200 and 400.

B. SIFT

Contrary to ORB, SIFT benefits less from local descriptor filtering which only slows the performance drop, as illustrated by Figure 4b), and summarized in Table III. The slight gain in computing time may not balance the quality loss, and this can make ORB a competitive challenger for some situations.

<table>
<thead>
<tr>
<th># descr.</th>
<th>Time (s)</th>
<th>Mean JI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1.488</td>
<td>0.847 ± 0.004</td>
</tr>
<tr>
<td>2000</td>
<td>1.624</td>
<td>0.877 ± 0.004</td>
</tr>
<tr>
<td>3000</td>
<td>1.718</td>
<td>0.888 ± 0.004</td>
</tr>
<tr>
<td>4000</td>
<td>1.781</td>
<td>0.892 ± 0.004</td>
</tr>
</tbody>
</table>

Fig. 4. Influence of filtering over result quality.

<table>
<thead>
<tr>
<th># descr.</th>
<th>Time (s)</th>
<th>Mean JI</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>0.127</td>
<td>0.637 ± 0.006</td>
</tr>
<tr>
<td>400</td>
<td>0.127</td>
<td>0.771 ± 0.005</td>
</tr>
<tr>
<td>600</td>
<td>0.132</td>
<td>0.818 ± 0.004</td>
</tr>
<tr>
<td>800</td>
<td>0.137</td>
<td>0.837 ± 0.004</td>
</tr>
<tr>
<td>1000</td>
<td>0.143</td>
<td>0.846 ± 0.004</td>
</tr>
<tr>
<td>1200</td>
<td>0.149</td>
<td>0.852 ± 0.004</td>
</tr>
<tr>
<td>1400</td>
<td>0.155</td>
<td>0.856 ± 0.004</td>
</tr>
<tr>
<td>1600</td>
<td>0.161</td>
<td>0.857 ± 0.004</td>
</tr>
<tr>
<td>1800</td>
<td>0.166</td>
<td>0.860 ± 0.004</td>
</tr>
<tr>
<td>2000</td>
<td>0.171</td>
<td>0.861 ± 0.004</td>
</tr>
</tbody>
</table>

TABLE II. AVERAGE PERFORMANCE COMPARISON WHEN FILTERING 2000 LOCAL DESCRIPTORS, WITH AVERAGE PROCESSING TIME PER FRAME (ORB)

TABLE III. AVERAGE PERFORMANCE COMPARISON WHEN FILTERING 4000 LOCAL DESCRIPTORS, WITH AVERAGE PROCESSING TIME PER FRAME (SIFT)
### References


