Searching for Sub-images using Sequence Alignment

Tomas Homola  
Faculty of Informatics  
Masaryk University  
Brno, Czech Republic  
Email: xhomola@fi.muni.cz

Vlastislav Dohnal  
Faculty of Informatics  
Masaryk University  
Brno, Czech Republic  
Email: dohnal@fi.muni.cz

Pavel Zezula  
Faculty of Informatics  
Masaryk University  
Brno, Czech Republic  
Email: zezula@fi.muni.cz

Abstract—The availability of various photo archives and photo sharing systems made similarity searching much more important because the photos are not usually conveniently tagged. So the photos (images) need to be searched by their content. Moreover, it is important not only to compare images with a query holistically but also to locate images that contain the query as their part. The query can be a picture of a person, building, or an abstract object and the task is to retrieve images of the query object but from a different perspective or images capturing a global scene containing the query object. This retrieval is called the sub-image searching. In this paper, we propose an algorithm for retrieving database images by their similarity to and containment of a query. The novelty of it lies in application of a sequence alignment algorithm, which is commonly used in text retrieval. This forms an orthogonal solution to currently used approaches based on inverted files. The proposed algorithm is evaluated on a real-life data set containing photographs where images of logos are searched. It was compared to a state-of-the-art method and the improvement of 20% in average mean precision was obtained.

Keywords—image matching; sub-image retrieval; local image features; sequence alignment;

I. INTRODUCTION

The complexity of search in current business intelligence systems, academic research, or even home audio-visual databases grows up rapidly and users of such systems require searching their data by content. For example, the user sees a cathedral while watching a movie and by taking a snapshot, his or her private collection of holiday photos can be searched for images containing that cathedral. Thus, it is not sufficient to store data and search it by exact match but rather by means of similarity. Similarity searching is especially requested in multimedia databases, but also in digital right management systems, computer aided diagnosis and natural sciences.

In this paper, we focus on the issue of similarity searching in images with a particular interest of searching for parts of images – sub-image searching, where a query contains a picture of an object and the task is to retrieve database images that contain it as their part. In general, current sub-image searching methods stands on three main pillars: (i) local features as the basis of image characteristic; (ii) similarity of the local features; and (iii) verification of spatial (geometrical) distribution of the local features. All these tasks are computationally intensive, since the number of local features obtained from a photograph is usually very high, ranging up to thousands. It leads to databases managing billions of data items. Moreover, individual local features are expressed as high-dimensional vectors, which further complicates algorithms from the performance point of view. In this paper, we do not tackle the issue of extracting local features from images. We rather focus on sub-image searching algorithms and we propose an algorithm that applies local/global sequence alignment methods to solve the issue of the latter two pillars, namely identification of similar local features and verification of their spatial correspondence, together into one procedure. This means finding a group of features between two images that are mutually similar and have appropriate geometrical arrangement.

This remaining parts of paper are structured as follows. The necessary background is given in the next subsection. In Section II, the proposed algorithm for sub-image searching is presented including a discussion on its properties and its improvement. In Section III, the paper continues with experimental evaluation and comparison with a state-of-the-art solution. The related work is summarized in Section IV. The paper concludes in Section V.

A. Necessary Background

Multimedia data objects are usually characterized as sets of local features, which are spatially distributed in a $d$-dimensional space ($d = 1$ in the case of sound, $d = 2$ for the images, etc.). Local features are likely to be viewed as tuples $(pos_1,\ldots,pos_d,desc)$, where $pos_i$ indicates the position in the $i^{th}$ coordinate and desc the actual local descriptor that describes the area surrounding the point at $(pos_1,\ldots,pos_d)$. In addition, the descriptor may contain other attributes, e.g. expressing its importance or orientation. In SIFT [1], the importance is expressed by the scale attribute.

II. SUB-IMAGE SEARCHING ALGORITHM

Our approach presented in this paper compares the descriptors of local features and their spatial distribution at the same time. Its idea is based on taking a set of local features as a sequence and applying a local/global sequence
alignment algorithm to reveal whether a query is contained in an image or not. The sequences are strings formed over a huge alphabet which contains all possible values of feature descriptors and the order of letters in sequences is given by projecting positions of features to one dimension, e.g. the $x$ or $y$ coordinate. An example of the transformation of a query image and a database image to sequences via the projection to the $x$ coordinate is given in Fig. 1. A sequence alignment algorithm then identifies an alignment of the query sequence and image sequence (letters in bold in the figure) and assigns a score to it. In general, the score denotes how much the query sequence is contained in the image sequence and is based on comparing individual letters. We define this comparison in terms of the original distance of two feature descriptors.

The algorithm for retrieving top-$k$ images containing a query image is available in pseudo-code in Fig. 2. We assume that a collection of images has been processed in advance and a sequence is available for each image. Thus, a database of all sequences is built. This algorithm is very straightforward and does not give any details from the query evaluation performance point of view. At least, there is no need to rank all database images, but a priority queue of the length $k$ can be used to decrease memory footprint as well as a part of computational costs. In this paper, we do not study this issue and we will investigate it in the future. However, an indexing structure for sequences, such as [2], can be applied to avoid the sequential scan over all database images. Provided the alignment algorithm used can be converted to a metric function, any metric index structure [3] can be employed.

In the following, we focus on the issue of sequence alignment and describe two well-known algorithms suitable for our task.

A. Sequence Alignment

For comparison of generated literal sequences several different methods can be used, however, they should compute alignment rather than similarity. For example, the edit distance is not convenient because it counts the number of changes needed to transform one sequence to another. We have to aim at counting the number of letters of the first (query) sequence to be contained in the other (database) sequence and how much they are permuted and distributed.

We have chosen the algorithms performing alignment of two sequences and allowing “gaps”. The property of assigning a special weight to gaps is very important for our task, since the query sequence is usually much shorter than any database sequence. Thus the leading and trailing letters should not be counted. This is depicted in Fig. 1 as the leading letter $A$ and trailing letters $FG$ of the database sequence. A gap can also be in the middle (the letters $BH$).

In the following, we recapitulate a general algorithm for sequence alignment, which is usually implemented in dynamic programming. Suppose two sequences obtained from two different images, let them denote as $M$ and $N$, their lengths as $|M| = m$ and $|N| = n$, and the $i$th letter of $M$ using brackets as $M^i$. We generate an alignment of $M$ and $N$ by maximizing a two-dimensional scoring function $V(i, j)$. The value $V(i, j)$ of this function denotes the best score for aligning the first $i$ letters of $M$, and the first $j$ letters of $N$. The final score of the whole alignment is stored in $V(|M|, |N|)$. The function $V(i, j)$ is based on a user-defined letter scoring function $score(x, y)$ that returns a score of replacing the letter $x$ for $y$. However due to the alphabet we use, an extension of this mechanism is necessary. We define a three-level letter scoring with respect to the distance between two letters, which are feature descriptors in our case. The score $S_e$ of equal letters is assigned if the distance between the corresponding descriptors is less than or equal to a threshold $T_e$. As in the protein sequence matching, the threshold $T_a$ defines a value of two descriptors being approximately the same but not equal, so the score of them is $S_a$. A pair of descriptors having their distance greater than $T_a$ is considered as a mismatch and a constant score $S_m$ is assigned. These scores and thresholds must satisfy the following inequalities: $0 \leq T_a \leq T_e$ and $S_e \geq S_a > S_m \geq 0$. 

---

**Input:** Query image $Q$; Database of image sequences $DB$

1. Extract local features from $Q$ and ignore those that are marginally important; E.g., the scale attribute in SIFT is less than one.
2. $S \leftarrow$ Create the query sequence by the projection to the $x$ axis;
3. Rank sequences in $DB$ using a sequence alignment algorithm with decreasing similarity score to $S$;
4. Return the first $k$ images.

---

**Figure 1.** Transformation of local features to a sequence of letters. The descriptors matching a query (left) are emphasized in bold in a DB image (right).

**Figure 2.** Sub-image Searching Algorithm in pseudo-code.
mon subsequence matching – and allows gaps to be present during matching. Each letter in a gap is scored with a non-negative value $S_g$. The scoring function $V(\cdot, \cdot)$ is computed as:

$$V(i, j) = \max \begin{cases} 
V(i - 1, j) - S_g \\
V(i, j - 1) - S_g \\
V(i - 1, j - 1) + \text{score}(M[i], N[j]) 
\end{cases}$$

2) Smith-Waterman: The Smith-Waterman algorithm [5], [6] performs the local alignment and gaps are penalized in two ways: opening a new gap $S_g$ and continuing the gap $S_{cg}$. This is very suitable for our task (sub-image searching), because it ignores gaps in the beginning and in the end of sequences. The scoring function $V(\cdot, \cdot)$ is computed as:

$$V(i, j) = \max \begin{cases} 
0, \\
E(i, j) \\
F(i, j) \\
V(i - 1, j - 1) + \text{score}(M[i], N[j]) 
\end{cases}$$

The functions $E$ and $F$ are “gap counters”. They are zero-initialized and defined as:

$$E(i, j) = \max \begin{cases} 
E(i, j - 1) - S_{cg} \\
V(i, j - 1) - S_g 
\end{cases}$$

$$F(i, j) = \max \begin{cases} 
F(i - 1, j) - S_{cg} \\
V(i - 1, j) - S_g 
\end{cases}$$

B. The Algorithm with Windowing

The comparison of two images described as sequences of descriptors using a sequence matching algorithm produces best results when a query is present in an image sequence with as few intermediate gaps as possible because they decrease the final score value. The leading and trailing gaps can be almost of any length if the Smith-Waterman algorithm with a small $S_{cg}$ is used. Since we use $x$-axis projection, a short query sequence in a long image sequence (containing thousands of letters) may be filled with many “noise” letters, so the final alignment score can be low. We handle this issue by "windowing" – slicing database images to windows of fixed sizes for which separate sequences are obtained. This leads to shorter image sequences and the “noise” is reduced. In case a large query image is posed, it is windowed too. Our windowing technique is different from [7], [8], where authors split images to a constant number of windows regardless the image size.

The windowing technique slices each image into fixed-sized windows. The preference on the window size is to contain as many queries as possible, because the searching must be done several times – once for each query window. Without the loss of generality, we can assume the query fits the window size. In this respect, the query sequence can be partly contained in two image sequences, which leads to lower alignment score. This can be handled by introducing window overlaps. A negative aspect of which is the increased number of windows and poorer query performance. We
study the influence of windowing on query effectiveness in the experiments.

We analyze the query efficiency in the following. Initially, let assume that no indexing technique is used to speed up the search in Lines 3 and 4 of the algorithm in Fig. 2. If the windows do not overlap, the same query performance is obtained. Otherwise it deteriorates linearly as overlap increases. However, we can suppose an indexing technique is used and it exhibits logarithmic complexity of searching for top-k items. The windowing is then convenient twofold. Firstly, fewer windows of each image (i.e. not the whole images) need to be checked, so the query performance is improved in comparison with the algorithm without windowing. Secondly, the impact of window overlapping on performance is logarithmic too. All these performance degradations are amortized by increased query effectiveness, which is proven in experimental trials that follow.

The algorithm with windowing is sketched out in Fig. 4. No special checks on how much the query is covered by the database images is done. Such a check can be easily implemented, however it is not crucial since we focus on similarity search and the final ranking assigns better ranks to database images that contain the whole query than a part of the query implicitly.

C. The Algorithm Complexity

The proposed algorithm can be decomposed to the following parts, for which we state their complexity. Firstly, the local features obtained from an image need to be projected and sorted, which leads to $O(N \log N)$ time complexity and $N$ space complexity, where $N$ is the number of local features. Secondly, the alignment algorithm is applied. The time complexity of Needleman-Wunsch and Smith-Waterman algorithms is the same and it is $O(mn)$, where $m$ and $n$ are the lengths of query and database sequences. The space complexity is $O(mn)$. Thirdly, the database lookup for $k$ most similar sequences is employed, where the complexity of a search algorithm depends on a technique used to organize the data. It is usually asymptotically logarithmic. We ignore the time to extract the local features from raw image data since all techniques based on local features need to obtain them. The applicability of the proposed algorithm is not constrained to any particular local features, e.g. SIFT – others like SURF, MSER or ASIFT can be used.

D. The Geometric Invariance

The proposed approach is inherently translation invariant – a possible cut of the query caused by the windowing is solved by window overlapping. The scale invariance is handled by the property of a sequence alignment algorithm which allows gaps in the compared sequences. Flipping invariance can be done by an additional evaluation with the query sequence inverted. The method is rotation invariant to few degrees only, which is not satisfactory yet. However, the rotation by 90° can be handled by projecting the query to the y-axis while database images are projected to the x-axis, and reevaluating the algorithm. The invariance of viewpoint change (tilt) is strongly influenced by this invariance of local features used.

III. EXPERIMENTS

In this section, we report on the trials we have performed to evaluate properties of the proposed sub-image search algorithm from different perspectives starting from score functions, window sizes and overlaps to adjusting threshold values on distance. We also compare our results with a state-of-the-art method on a real-life data set.

A. Data Set

To prove our ideas we have done several experiments with the Belga Logos dataset [9]. It contains 10,000 photographs and 26 logos. The logos are present in the photographs but they have not necessarily been taken from the photographs. The ground-truth for each logo is available in the form of “is/is not present” in a photograph. The position of logo in the photograph and possible multiple occurrences are not available. The average size of photographs is $800 \times 783$ pixels while the query logos are comparatively smaller and their dimension is $119 \times 77$ pixels on average – the smallest one is $74 \times 27$ pixels and the largest one is $150 \times 121$ pixels. From each image, SIFT local features [1] have been extracted.
B. Result Quality Measures

Since results obtained by our sub-image search algorithm are ranked and the ground-truth is available, we can apply widely-used measures to weight the result quality in terms of well-ordered true positives in answer. In particular, we use the Mean Average Precision (MAP) and the Averaged Normalized Modified Retrieval Rate (ANMRR) [10]. MAP for a set of queries $Q$ is defined as follows:

$$MAP(Q) = \frac{\sum_{q \in Q} AP(q)}{|Q|}$$

where $AP(q)$ stands for the average precision of the result of a query $q$.

$$AP(q) = \frac{\sum_{r=1}^{\# \text{of retrieved images}} P(r) \cdot rel(r)}{\# \text{of relevant images}}$$

where $r$ is the rank; $rel(r)$ tells whether the image at the rank $r$ is in the ground-truth or not; and $P(r)$ is the precision at a given cut-off rank:

$$P(r) = \frac{\# \text{of relevant retrieved images of rank } \leq r}{r}$$

The values returned by MAP are in $[0, 1]$ where the meaning is the same as for precision, i.e. one is the best.

ANMRR measures an average over all queries too, but not only does it indicate how many correct images are retrieved but also how highly they are ranked among all images retrieved. ANMRR returns a real value in $[0, 1]$, where the lower value is returned, the better the result is. We present inverted values of ANMRR to make it consistent with MAP and recall measures – it is denoted as 1-ANMRR.

C. Experiment Trials

We tested the influence of the following parameters to results: window size, window overlap, descriptor similarity and cost functions. In all experiments, we present MAP, ANMRR (as 1-ANMRR) and recall as the ranking-quality and effectiveness measures. In all graphs, zero indicates the worst and one the best results.

A score function needs to be specified in order to use a sequence alignment algorithm. We exemplified three possibilities (constant, linear and inverse) and tested their properties while varying values of scores. From a number of trials, we learned values of scores obtaining the best results in terms of the measures specified above: $S_c = 5$, $S_a = 2$, $S_m = -0.5$, $S_q = 1$ and $S_{cg} = 0$. We do not include these results here for space constraints. The penalty of continuing gaps is set to zero, which allows to diminish the influence of noise. This is markable in the results presented below. The influence of the score function to the results was marginal, thus we resorted to the constant variant. However, the score function may be important from the database indexing point of view. For example, the linear variant increases the number of returned values, which may lead to increase of standard deviation of distances. This is more convenient for indexing structures which perform more efficiently then.

Secondly, we varied the window size from very small windows of 64×64 pixels up to windows of 800×800 pixels – from windows slicing even the smallest queries up to windows covering the whole database images. The individual windows overlapped by one half, i.e. the next position was obtained by advancing the current position by a half of the window dimension. The results are presented in Fig. 5. The effectiveness obtained for the Needleman-Wunsch (N-W) algorithm deteriorates very fast with increasing window size – this was expected since it penalizes long gaps very much. Promising results were acquired for the Smith-Waterman (S-W) algorithm and the results are very stable even for large windows. Nonetheless, windows covering
whole database images introduce too much noise even for the S-W algorithm. As a result, the windowing technique is important and we use windows of 128×128 pixels in the following. Efficiency characteristics for different window size are summarized in Table I. These values represent a naive implementation of database search as the sequential scan over all possible windows. The time to compute the naive implementation of database search as the sequential scan over all possible windows. The time to compute the naive implementation of database search as the sequential scan over all possible windows. The time to compute the naive implementation of database search as the sequential scan over all possible windows.

Next, we studied the influence of window overlaps on the results. We changed the overlap size of several steps ranging from 75% of the window size via zero overlap up to −100%, which represents that window position is advanced by the distance thresholds to the results. Labels of horizontal axis are in form $T_a; T_e$. Window size was 128×128 and overlap was 50%.

### D. Comparison

In this section, we present comparison of our algorithm with a method by Joly et al. [9]. It is based on a locality sensitive hashing, so local features are transformed to hash codes. The geometric verification of features’ positions is done by a RANSAC algorithm [11]. Table II reports on values of MAP for all queries separately. Results of Joly’s method is for the “Qset2” query set (external logos). From the values reported, we can read that the S-W algorithm outperforms not only the N-W algorithm, which was expected, but also Joly’s approach. The improvement is by 20% on average values of MAP (from 25.67 to 30.95). Apart from this, Joly states that all results depend on the logos used in experiments and on their simplification as database objects. Our experience confirms it. For example, the “Ecusson” logo and text-only logos lead to significantly worse results compared to graphics-based ones. Logos characterized by a considerably lower amount of local features produce poor results in MAP as well as ANMRR measures too (even if the results exhibit high recall, the precision is low). Moreover, we analyzed the influence of projections and discovered that the results for the projection to $x$ axis and $y$ axis correlate by 0.96. This is very convenient since we can use just one to index images. This can be attributed to having square windows.
Our results are worse for logos which are blurred too much in the ground-truth database images. This is because we did not have SIFT descriptors used by Joly and we used a publicly available extractor by Lowe, so the descriptors we obtained are more prone to blur.

IV. RELATED WORK

Current state-of-the-art techniques, that use local image features, exploit an existing technology for text retrieval, i.e. inverted files and tf-idf measures. They also employ various simplifications to speed the process of sub-image searching. To use inverted files, local features must be quantized first and so-called visual words [12], [13] are produced. The quantization is usually done by a clustering algorithms, such as k-means, but due to enormous data volume in a hierarchical way. Without losing generality, we assume that visual words are in a form of IDs (numbers or letters) of centers of clusters located by the clustering algorithms. Each image is then described as a bag of visual words (IDs). For efficient retrieval, an inverted file is constructed, where a posting list is created for each visual word. The posting list consists of identifications of images that contain that particular visual word. Term frequency and inverted document frequency can be used to rank the results. Before exposing the results to a user, the visual words identified as matching are verified for spatial placements to filter out false positives. This stage is often implemented using a variant of RANSAC algorithm [11]. This algorithm automatically selects parameters for transforming one set of features to another. The transformation is usually bound to affine transformations in 2-D space. RANSAC is a randomized and iterative process, so it is not time upper-bounded and the solution is in every iteration improved, but it is still approximate.

Not only is this whole clustering procedure for obtaining visual words very computationally demanding, but it also incurs the following problems that influence the quality of results. Firstly, local features can be situated at borders of clusters (Voronoi cells) as depicted in Fig. 8(a). As a consequence, such features are more similar to local features from other clusters than to their cluster center. Secondly, the clustering is done on a sample of data items (training set), so it is crucial to select the set that has very similar, or the same, properties as the whole data set. Fig. 8(b) depicts the situation where the indexed data set has changed and all features are concentrated in one cluster. Thirdly, even if the clustering is optimal, it may be necessary to get features from neighboring cells to improve the results. However, this problem has not been solved yet. There are some heuristics [14], [15] that find neighboring Voronoi cells approximately.

Apart from clustering, there are methods that transform local features to hash codes [9] and apply a locality sensitive hashing [16] to assign similar features the same hash code. A searching procedure retrieves candidate images which are then checked by the RANSAC algorithm. Next, the approach presented in [17] takes two images as similar if they share a fixed number of unique descriptors. In [18], keypoints in an image are detected and other visual words are formed in concentric rings around the keypoints. The rings are then represented as histograms of visual words contained in concentric rings around the keypoints.

<table>
<thead>
<tr>
<th>Logo</th>
<th>Logo size [px]</th>
<th>Joly N-W</th>
<th>S-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addidas-text</td>
<td>129 × 33</td>
<td>1.10</td>
<td>11.85 17.82</td>
</tr>
<tr>
<td>Addidas</td>
<td>132 × 74</td>
<td>0.70</td>
<td>8.37 14.23</td>
</tr>
<tr>
<td>Base</td>
<td>129 × 57</td>
<td>58.20</td>
<td>60.34 68.88</td>
</tr>
<tr>
<td>Bougues</td>
<td>110 × 83</td>
<td>15.30</td>
<td>19.55 34.01</td>
</tr>
<tr>
<td>Citroen-text</td>
<td>150 × 27</td>
<td>0.10</td>
<td>7.82 6.45</td>
</tr>
<tr>
<td>Citroen</td>
<td>103 × 102</td>
<td>4.50</td>
<td>6.31 0.85</td>
</tr>
<tr>
<td>Coca-cola</td>
<td>150 × 52</td>
<td>0.10</td>
<td>6.33 12.0</td>
</tr>
<tr>
<td>Cotidis</td>
<td>92 × 95</td>
<td>65.30</td>
<td>75.01 85.53</td>
</tr>
<tr>
<td>Dexia</td>
<td>109 × 38</td>
<td>51.30</td>
<td>53.47 65.23</td>
</tr>
<tr>
<td>Ecusson</td>
<td>78 × 116</td>
<td>0.10</td>
<td>9.67 9.99</td>
</tr>
<tr>
<td>Elecerec</td>
<td>74 × 74</td>
<td>80.10</td>
<td>14.54 13.74</td>
</tr>
<tr>
<td>Ferrari</td>
<td>86 × 116</td>
<td>17.50</td>
<td>11.72 25.24</td>
</tr>
<tr>
<td>Gussi</td>
<td>97 × 62</td>
<td>0.00</td>
<td>0.91 14.65</td>
</tr>
<tr>
<td>Kia</td>
<td>150 × 77</td>
<td>75.60</td>
<td>48.55 61.47</td>
</tr>
<tr>
<td>Mercedes</td>
<td>114 × 114</td>
<td>19.20</td>
<td>36.96 51.19</td>
</tr>
<tr>
<td>Nike</td>
<td>150 × 71</td>
<td>2.60</td>
<td>7.97 8.08</td>
</tr>
<tr>
<td>President</td>
<td>121 × 121</td>
<td>23.20</td>
<td>13.36 27.2</td>
</tr>
<tr>
<td>Puma-text</td>
<td>117 × 44</td>
<td>2.20</td>
<td>5.54 6.97</td>
</tr>
<tr>
<td>Puma</td>
<td>89 × 65</td>
<td>0.70</td>
<td>5.11 8.01</td>
</tr>
<tr>
<td>Quick</td>
<td>104 × 103</td>
<td>56.60</td>
<td>12.43 24.92</td>
</tr>
<tr>
<td>Roche</td>
<td>144 × 77</td>
<td>0.20</td>
<td>0.30 0.20</td>
</tr>
<tr>
<td>SNCF</td>
<td>134 × 72</td>
<td>33.70</td>
<td>25.33 43.67</td>
</tr>
<tr>
<td>Stella</td>
<td>150 × 99</td>
<td>43.40</td>
<td>57.38 59.56</td>
</tr>
<tr>
<td>Tnt</td>
<td>150 × 55</td>
<td>4.40</td>
<td>3.95 8.35</td>
</tr>
<tr>
<td>VRT</td>
<td>128 × 72</td>
<td>11.20</td>
<td>17.29 39.52</td>
</tr>
</tbody>
</table>

Mean: 25.67 22.76 30.95

Table II

COMPARISON OF JOLY’S METHOD AND OUR ALGORITHM IN TERMS OF MAP (IN %). WINDOW SIZE WAS 128×128 AND OVERLAP WAS 50%.
in them. These rings form a profile of a keypoint. Similarity of two images is based on similarity of profiles of their keypoints. An advantage of this approach is that it includes the geometrical validation into the similarity function (i.e. definition of rings). In this respect, it is similar to our approach. However, it still needs visual words to be obtained.

On contrary to the previous approaches, in [8], database images as well as a query image are segmented to chunks of a pre-defined size and MPEG-7 global features [10] are extracted. A searching procedure then finds correspondence between the database images and the query image by utilizing a metric index structure [19], [20]. A disadvantage of this approach is that global features are very sensitive to image background thus it usually provides lower effectiveness.

V. CONCLUSION

We proposed an algorithm for retrieving images that contain a query image. This approach is unique in transforming the problem to sub-sequence matching and applying a sequence alignment algorithm. This approach was proved to be suitable for this task by running experiments on a real-life data set and comparing its results to a state-of-the-art algorithm. The experiments revealed a 20% improvement in mean average precision. Advantages of our approach are:

• similarity of local feature descriptors and their spatial positions is checked in one step;
• it is generic – the representation of local features may change (e.g. descriptors can be quantized – visual words, locality sensitive hashing, etc.);
• scores in the alignment algorithm can be tuned to support user feedback; and
• alignment methods can also be used for extremely small query images because only three features are needed to perform the geometric check, while RANSAC requires seven features to provide satisfactory results.

As for the future work, we plan to apply an indexing technique to speed up searching for similar sequences, to analyze performance when quantization is done and its influence, and to test implementations of alignment algorithms on GPUs [21] and FPGAs [22].

ACKNOWLEDGMENT

This research was supported by the national projects GAP 201/09/0683, GAP103/10/0886, VF20102014004 and MUNIA/A/0922/2009.

REFERENCES