Robust multi-view feature matching from multiple unordered views

Jian Yao\(^a\)*, Wai-Kuen Cham\(^b\)

\(^a\)IDIAP Research Institute, Rue du Simplon 4, Case Postale 592, CH-1920 Martigny, Switzerland
\(^b\)Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

Received 25 May 2006; received in revised form 4 February 2007; accepted 19 February 2007

Abstract

This paper explores the problem of multi-view feature matching from an unordered set of widely separated views. A set of local invariant features is extracted independently from each view. First we propose a new view-ordering algorithm that organizes all the unordered views into clusters of related (i.e. the same scene) views by efficiently computing the view-similarity values of all view pairs by reasonably selecting part of extracted features to match. Second a robust two-view matching algorithm is developed to find initial matches, then detect the outliers and finally incrementally find more reliable feature matches under the epipolar constraint between two views from dense to sparse based on an assumption that changes of both motion and feature characteristics of one match are consistent with those of neighbors. Third we establish the reliable multi-view matches across related views by reconstructing missing matches in a neighboring triple of views and efficiently determining the states of matches between view pairs. Finally, the reliable multi-view matches thus obtained are used to automatically track all the views by using a self-calibration method. The proposed methods were tested on several sets of real images. Experimental results show that it is efficient and can track a large set of multi-view feature matches across multiple widely separated views.

© 2007 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Organization of unordered views; Two-view matching; multi-view matching

1. Introduction

Recently a lot of effort has been devoted to developing new approaches for building visual scenes \([1,2]\), panoramic image mosaicing \([3]\), object recognition \([4]\) and image retrieval \([5]\) from an unordered set of uncalibrated views. Our objective in this paper is to develop an automatic scene reconstruction system that finds a large set of reliable multi-view matches over an unordered set of uncalibrated views taken from widely separated viewpoints, then automatically calibrate cameras of all the views and finally build a 3D scene model.

In the past few years, there has been a growing interest in wide-baseline stereo (WBS) matching, which builds up correspondences between two widely separated views. It has wide applications, such as view morphing, reconstruction from multiple views \([6,7]\) and visual navigation \([8]\). The main idea for WBS matching is to first extract local invariant features independently from the two views, then characterize them by invariant descriptors and finally build up correspondences between them \([5,9–16]\). Local invariant features have emerged as an invaluable tool in tackling the ubiquitous image correspondence problem. By using descriptors that are invariant not just to translation, but also to rotation \([5,11]\), scale \([14,16]\) and affine warping \([10,11,13,15–17]\), local invariant features provide much more robust matching than previous methods based on area based matching metrics including the SAD, SSD, NCC and their zero-meaned versions. The detailed evaluation and comparison for recently proposed local invariant feature detectors and descriptors can be found in Refs. \([18–20]\). Compared with traditional small-baseline stereo matching, WBS matching is more difficult due to more occlusions and larger deformations derived from large viewpoint changes. Fortunately, the recently developed local invariant feature detectors and descriptors overcome these difficulties to some extent.

To find reliable correspondences between two views, outliers can be first filtered out and more reliable matches can be then found under the epipolar geometry. However, in order to derive the epipolar geometry, a bootstrap phase involving
full image search especially under the wide-baseline condition is needed. One typical approach is to extract a set of feature points from the two views, attempt a matching, and then extract a subset of reliable matches by LMedS, M-estimators or RANSAC based on epipolar geometry. Before the epipolar geometry test, some people incorporated some constraints to reduce the number of matches. Zhang et al. [21] developed a class of techniques known as relaxation techniques by allowing the candidate matches to reorganize themselves by propagating some constraints, such as continuity and uniqueness, through the neighborhood. Ringer et al. [22] performed a similar operation to detect outliers between two views by assuming that features close to each other in one image should remain close to each other in another. Smith et al. [23] developed a new technique, called as the Median Flow Filter, to detect outliers by assuming that the image motion between two views is locally similar. These methods demonstrate good matching results between two small-baseline views. However, in the wide-baseline case, good matching results may not be obtained due to large deformation and occlusion derived from large viewpoint changes between two widely separated views. Tell et al. [10] proposed a wide-baseline matching algorithm using affine invariants. Based on this matching algorithm, they present a more robust matching method by enforcing topological constraints such as one-to-one matching and cyclic order based on applying cyclic string matching for every possible corner correspondence [24]. This newly proposed method has significant better performance than that of [10]. It provides a larger set of correspondences and finds solutions in cases where the original algorithm fails completely. Lu et al. [25] present a method for robust two-view WBS matching that efficiently limits the searching space for matching using the cross-epipolar ordering constraint. This algorithm has the two-fold advantage of reducing the risk of the computational cost as well as mismatches because only a subset of possible matches is searched.

While these methods focus on two views, recently some researchers have developed several algorithms for matching features across an unordered set of widely separated views, which forms a basis of many important visual tasks, such as calibration, 3D scene reconstruction and tracking. One way of computing multi-view matches is to first find matches for all view pairs using some two-view WBS matching algorithms and then sew them together into multi-view matches. This approach can obtain full multi-view matches across all the views but the computation becomes intractable when a large setting with hundreds or even thousand of views is considered. One solution is to perform WBS matching computations only on some of the view pairs, and then use the transitivity of matches to obtain further matching information among views that are not matched directly. A key aspect of such algorithm is the choice of view pairs to which the WBS matching algorithm will be applied. To reduce the number of WBS matching computations, Schaffalitzky and Zisserman [6] proposed a heuristic approach in which single-view invariants are first computed and mapped to a large features-versus-views hash table. The hash table can then guide the greedy choice of the view pairs on which to perform the WBS matching algorithm. Ferrari et al. [7] proposed another algorithm for finding multi-view feature correspondences across an unordered set of widely separated views, which can find more correspondences compared with that reported in Ref. [6], by extending the correspondence of two (or three) views to multiple (> 3) views and further filtering out the mismatches based on a topological constraint. Levi and Werman [26] considered the problem of computing the fundamental matrices between all view pairs in an unordered set of views from the fundamental matrices of a small number of view pairs. Their main contribution is an algebraic analysis of the constraints that can be extracted from a partial set of fundamental matrices among neighboring views. These constraints are then used to compute the missing fundamental matrices. While the above mentioned algorithms are designed to run on a single computer, Avidan et al. [27] present a distributed approach to compute multi-view matches across an unordered set of views by employing the theory of random graphs to provide an efficient probabilistic algorithm that performs WBS matching algorithm on a small number of view pairs where each view is attached to a separate computing unit and the views communicate over a network. In this scheme, computations on WBS matching and communication over a network are greatly reduced. Brown and Lowe [4] present a fully automatic system to recognize and reconstruct multiple 3D objects in an unordered image database by utilizing the local invariant features extracted by the SIFT detector [14] from images.

In this paper, we present a system for robust multi-view feature matching and automatic scene reconstruction from an unordered set of uncalibrated views taken from widely separated viewpoints. Fig. 1 shows the block diagram of the proposed system. Given an unordered set of views, we first organize them by using a new and efficient view-ordering algorithm which computes the view-similarity values of all view pairs based on both number of matches between each view pair and their mean distance by reasonably selecting part of extracted features on the unordered views to match. Then a robust two-view matching algorithm is proposed to find as many reliable matches as possible between two views in both the wide-baseline case and the small-baseline one. The global affine transformation between two views is first estimated to reduce the changes in translation, rotation, zooming or deformation between the two views. In most cases, both motion and feature characteristics of matches tend to have consistent changes after applying the
global affine transformation. This fact is efficiently utilized to
detect possible outliers and incrementally find more reliable
matches under the epipolar constraint from dense to sparse be-
cause a match with more neighbors around it can be detected as
an inlier or an outlier with a high-confidence level. In order to
reduce the computation time, we perform the robust two-view
matching algorithm only on a small number of view pairs. Af-
ter that, a large set of reliable multi-view matches are found by
using a new algorithm which reconstructs missing matches in
a neighboring triple of views, efficiently determining the states
of two-view matches and filtering out the outliers using the tri-
focal tensors over the neighboring triples of views by using a
small number of selected view pairs in the multi-view matching
process. Based on these found multi-view matches, metric re-
construction is automatically achieved by using a flexible self-
calibration method. Finally, we build a surface scene model by
using triangulation interpolation.

The rest of this paper is organized as follows. The new view-
ordering algorithm that organizes an unordered set of views
over the constructed view-spanning trees is presented in Sec-
tion 2. The proposed robust two-view matching algorithm is de-
scribed in Section 3. In Section 4, the new matching algorithm
that can find a large set of reliable multi-view matches across
views over the constructed view-spanning trees is presented. A
brief description for 3D scene reconstruction system from an
unordered set of uncalibrated views is described in Section 5.
In Section 6, we briefly analyze the computational complexity
of the whole matching algorithm we propose. Experimental re-
sults are reported in Section 7. Finally, conclusions are given
in Section 8.

2. Ordering views

In this section, we present a new view-ordering algorithm
that, given an unordered set of views, can group them into clus-
ters of related views. The views in each cluster come from the
same scene, which are organized in one view-spanning tree like
the examples as shown in Figs. 2 and 3. To organize a set of \( N \)
unordered views \( \{ v_i, i = 1, \ldots, N \} \), in general, we first perform
two-view matching algorithm for each view pair among these
views based on the extracted features and then compute their
view similarities using the matched features. However, applying
two-view matching exhaustively for a large set of views is
very expensive computationally. To reduce the number of two-
view matching computations, Schaffalitzky and Zisserman [6]
proposed a heuristic approach in which single-view invariants
are first computed and mapped to a large features-versus-views
hash table. The hash table can then guide the greedy choice of
the view pairs on which to perform the two-view matching al-
gorithm. Although their proposed method greatly reduces the
computation time of ordering views, it still is a bit unacceptable
when the number of views to be organized is large. In addition,
their proposed method uses the number of matched features
in each view pair to measure the similarity between the view
pair without taking into consideration of their reliability. In this
section, a new view-ordering algorithm is proposed to over-
come these problems to some extent, which is an extension of
the Schaffalitzky et al.’s method with three major differences.
The first one is that we combine the numbers of matched fea-
tures found and their mean Euclidean distances into the more
efficient measure, the similarity between two views. The sec-
ond one is that the proposed view-spanning tree construction
scheme is simpler as we do not need to check whether there
exists a cycle in the graph for added views, which is needed
in their method. The last and the most important one is that
we propose a general feature-reducing method for most kinds
of invariant features to significantly reduce the time of or-}

![Fig. 2. Two-view connections found between the views of the Valbonne church sequence: (a) by using all the extracted features (the whole searching and constructing time is 14.43 s) and (b) by using 20% features, i.e. the feature-reducing factor \( s = 0.2 \) (the whole searching and constructing time is 1.48 s).](http://www.cs.umd.edu/~mount/ANN/)
The total number of all the extracted SIFT features from these 15 images is 49,885. The results are shown in Tables 1(a)–(c). The view-similarity value between two views \( V_i \) and \( V_j \) is computed by

\[
S_{im}(V_i, V_j) = \frac{N(V_i, V_j)}{N_{\text{max}}} + (1 - \alpha) \frac{d_{\text{max}} - d(V_i, V_j)}{d_{\text{max}}},
\]

where \( N(V_i, V_j) \) denotes the total number of matched features between views \( V_i \) and \( V_j \), \( d(V_i, V_j) \) denotes the mean distance of all matched features between views \( V_i \) and \( V_j \), \( N_{\text{max}} \) and \( d_{\text{max}} \) are the highest \( N(V_i, V_j) \) and the largest \( d(V_i, V_j) \) among all view pairs. The balancing factor \( \alpha \in [0, 1] \) was chosen to be 0.8. Note that the second term in Eq. (1) measures the reliability of the matched features, which increases when \( d(V_i, V_j) \) decreases. The view-similarity table that contains the similarity value between any view pair among \( N \) views is given, as shown in Table 1(c). Actually the three tables shown in Tables 1(a)–(c) are symmetrical across the primary diagonal which are blank. For convenience, we only show the values of the upper triangle.

Based on the computed view-similarity table, these \( N \) unordered views can be organized by constructing one or several view-spanning trees across these \( N \) views. Each view-spanning tree corresponds to a same scene. A tree is constructed as follows. Starting from the view pair with the highest view-similarity value (i.e. the two most similar views), we add a new view from the remaining views, which has the highest view-similarity value with an existing view on the view-spanning tree. However, if the number of matches corresponding to the new view to be linked is fewer than 50, we check the validity of this link through testing the epipolar geometric constraint. If a fundamental matrix can be successfully determined between the two views on this link and the number of reliable matches constrained with the estimated fundamental matrix is larger than 20, this link will be added onto the view-spanning tree. This process continues until the view-spanning tree is completely constructed. If there are still some views not to be added onto this view-spanning tree, we start to construct another new view-spanning tree by selecting the view pair with the highest view-similarity value from the remaining un-added views and repeat the process until all the views have been added. The proposed view-ordering approach was tested on two sets of real data [31]: the Valbonne church sequence of 15 images and the Raglan Castle sequence of 46 images. For the Valbonne church sequence, the constructed view spanning tree is shown in Fig. 2(a). For the Raglan Castle sequence, we obtained five view-spanning trees which have 2, 6, 8, 12 and 18 views, respectively. The longest constructed view-spanning tree which has 18 views is shown in Fig. 3.

Although the ANN searching strategy can achieve significant improvement in the searching time, it is still a bit unacceptable with the increase of both the number of views to be organized and the number of extracted features. When we match one view to other \( N - 1 \) views, one natural way to reduce the searching time is to randomly select part of features on this view for searching the approximate nearest neighbors (features) among all the features on other \( N - 1 \) views when the number of the extracted features from this view is not too small. In this way, the query set is reduced but the data set is kept unchanged. To reduce both the query set and the data set simultaneously, which do not affect the final ordering result of these \( N \) unordered views, there are two natural ways to complete it. One way is to shrink the original views of size \( W \times H \) (in pixels) into smaller views of size \( \delta W \times \delta H \) where the image shrinking factor \( \delta \in (0, 1) \). Suppose that the number of extracted features from an image is proportional to the image size \( W \times H \). We may infer that the total number of detected features will be decreased to \( \delta^2 \) times of that of the original images. Our experiments on the SIFT feature detector have verified that this inference is basically reasonable in most cases when \( \delta \) is not too small. Another way is to strengthen the feature extraction criteria in order to extract less or sparser features for efficiently ordering views. Of course, sometimes the extraction results by some feature detectors provide some additional information which can be used to suitably select part of features on all the views in the same criterion for efficiently ordering views. Here, we propose a general approach to select part of features on all the views so that the ordering result of all the views is basically unchanged. Let the extracted features be represented by an independent \( d \)-dimension vector. For example, \( d = 128 \) for the SIFT represented features and \( d = 20 \) for the PCA-SIFT represented ones. Our objective here is to reduce the number \( N(V_i) \) of features on the view \( V_i \) into \( sN(V_i) \), \( s \in (0, 1) \), where we call \( s \) the feature-reducing factor. The proposed algorithm is described in Fig. 4. First we select the central \( pN(V_i) \) features of all the \( N(V_i) \) features sorted by their first dimensional
Table 1
Six 15 × 15 tables which are used for ordering the 15-image Valbonne church sequence by using all the extracted features (the first three tables) and by using 20% features (the last three tables).

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>245</td>
</tr>
<tr>
<td>1</td>
<td>503</td>
</tr>
<tr>
<td>522</td>
<td>317</td>
</tr>
<tr>
<td>3</td>
<td>462</td>
</tr>
<tr>
<td>4</td>
<td>517</td>
</tr>
<tr>
<td>5</td>
<td>592</td>
</tr>
<tr>
<td>6</td>
<td>305</td>
</tr>
<tr>
<td>7</td>
<td>171</td>
</tr>
<tr>
<td>8</td>
<td>314</td>
</tr>
<tr>
<td>9</td>
<td>173</td>
</tr>
<tr>
<td>10</td>
<td>703</td>
</tr>
<tr>
<td>11</td>
<td>581</td>
</tr>
<tr>
<td>12</td>
<td>646</td>
</tr>
<tr>
<td>(a)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1721</td>
</tr>
<tr>
<td>1</td>
<td>1493</td>
</tr>
<tr>
<td>1489</td>
<td>1597</td>
</tr>
<tr>
<td>1473</td>
<td>1395</td>
</tr>
<tr>
<td>1470</td>
<td>1555</td>
</tr>
<tr>
<td>1439</td>
<td>1600</td>
</tr>
<tr>
<td>1610</td>
<td>1537</td>
</tr>
<tr>
<td>1622</td>
<td>1610</td>
</tr>
<tr>
<td>1609</td>
<td>1806</td>
</tr>
<tr>
<td>1718</td>
<td>1548</td>
</tr>
<tr>
<td>1435</td>
<td>1280</td>
</tr>
<tr>
<td>1408</td>
<td>1705</td>
</tr>
<tr>
<td>1513</td>
<td>1791</td>
</tr>
<tr>
<td>1604</td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.310</td>
</tr>
<tr>
<td>1</td>
<td>0.626</td>
</tr>
<tr>
<td>2</td>
<td>0.648</td>
</tr>
<tr>
<td>3</td>
<td>0.581</td>
</tr>
<tr>
<td>4</td>
<td>0.644</td>
</tr>
<tr>
<td>5</td>
<td>0.733</td>
</tr>
<tr>
<td>6</td>
<td>0.389</td>
</tr>
<tr>
<td>7</td>
<td>0.235</td>
</tr>
<tr>
<td>8</td>
<td>0.400</td>
</tr>
<tr>
<td>9</td>
<td>0.228</td>
</tr>
<tr>
<td>10</td>
<td>0.530</td>
</tr>
<tr>
<td>11</td>
<td>0.862</td>
</tr>
<tr>
<td>12</td>
<td>0.713</td>
</tr>
<tr>
<td>13</td>
<td>0.778</td>
</tr>
<tr>
<td>(c)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>84</td>
</tr>
<tr>
<td>1</td>
<td>97</td>
</tr>
<tr>
<td>2</td>
<td>115</td>
</tr>
<tr>
<td>3</td>
<td>122</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>157</td>
</tr>
<tr>
<td>6</td>
<td>81</td>
</tr>
<tr>
<td>7</td>
<td>66</td>
</tr>
<tr>
<td>8</td>
<td>73</td>
</tr>
<tr>
<td>9</td>
<td>61</td>
</tr>
<tr>
<td>(d)</td>
<td></td>
</tr>
</tbody>
</table>
Table 1 (Continued)

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>118</td>
<td>125</td>
<td>90</td>
<td>46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>156</td>
<td>82</td>
<td>53</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>122</td>
<td>63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>117</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(e) $N(v_i, v_j)$, the number of matched features between views $v_i$ and $v_j$; (b),(e) $\bar{d}(v_i, v_j)$, the mean Euclidean distance of matched features between views $v_i$ and $v_j$; (c),(f) $\text{Sim}(v_i, v_j)$, the view-similarity value of view pair $v_i$ and $v_j$.

\[ \text{(a),(d)} N(v_i, v_j), \text{the number of matched features between views } v_i \text{ and } v_j; \text{(b),(e)} \bar{d}(v_i, v_j), \text{the mean Euclidean distance of matched features between views } v_i \text{ and } v_j; \text{(c),(f)} \text{Sim}(v_i, v_j), \text{the view-similarity value of view pair } v_i \text{ and } v_j. \]

Fig. 4. Flowchart of reducing the number $N(v_i)$ of features represented by $d$-dimension vectors on $v_i$ into $sN(v_i)$ where $s = p^d$. 

$N(v_i)$ features

\[ \text{select central } 100p\% \text{ features} \]

\[ \text{sorted by first dimensional elements} \]

$pN(v_i)$ features

\[ \text{select central } 100p\% \text{ features} \]

\[ \text{sorted by second dimensional elements} \]

$p^2N(v_i)$ features

\[ \text{select central } 100p\% \text{ features} \]

\[ \text{sorted by } d\text{-th dimensional elements} \]

$p^dN(v_i)$ features

\[ \text{select central } 100p\% \text{ features} \]

\[ \text{sorted by } d\text{-th dimensional elements} \]
elements where $p$ is the reducing factor in each dimension, $p \in (0, 1]$. Then we select the central $p^2 N(V_i)$ features of all the remaining $p N(V_i)$ features sorted by their second dimensional elements. Finally, we expect to select $p^d N(V_i)$ features after handling all the $d$ dimensions in this way, satisfying the following equation:

$$p^d N(V_i) = s N(V_i) \Rightarrow p = \sqrt[2d]{s}.$$  \hspace{1cm} (2)

The feature-reducing approach was tested on both the Valbonne church sequence and the Raglan Castle sequence. We set $s = 0.2$ and $d = 20$ and so $p = \sqrt[20]{0.2} = 0.92268$. The three feature tracking tables for the Valbonne church sequence are shown in Tables 1(d)–(f). The final constructed view-spanning tree for this sequence is shown in Fig. 2(b), which has a very similar structure with the original view-spanning tree as shown in Fig. 2(a). The whole searching and constructing time is 1.48 s which is about 10% of the original 14.43 s, which was tested on a P4-1.8 GHz PC. By using other different feature-reducing factors, the needed searching and constructing times are shown in Fig. 5(a). For the Raglan Castle sequence, we extracted 523,980 SIFT features from 46 images. The needed searching and constructing times for different feature-reducing factors are shown in Fig. 5(b).

3. Robust two-view matching

The main idea for two-view WBS matching is to first extract local invariant features independently from the two views and then characterize them by invariant descriptors. The nearest neighbor method is often used to identify candidate matches and the epipolar geometry is imposed to filter out the outliers. Finally, more matches can be found by guided matching. Unfortunately, the nearest neighbor is not always the best match under the wide-baseline conditions due to occlusion and deformation derived from large viewpoint changes and repeated structures in the scenes. In this section, a robust two-view matching algorithm is proposed to overcome this problem. Based on the assumption that changes of both motion and feature characteristics of one match are consistent with those of neighbors after applying the global affine transformation on the two views, the proposed algorithm is used to filter out possible outliers and incrementally find more reliable matches under the epipolar constraint from dense to sparse because a match with more neighbors around it can be detected as an inlier or an outlier with a high-confidence level. Major steps of the proposed algorithm are simply described in Algorithm 1. Details will be given in the subsequent sections.

Algorithm 1. The proposed robust two-view matching

1. Initial matching
   (a) Extract two sets of local invariant features independently from two views;
   (b) For each feature $f = (x, c, d)$ on one view, we find an approximate nearest neighbor $f' = (x', c', d')$ on another view, satisfying the Euclidean distance between $f$ and $f'$ is not larger than $R_f = 2500$, i.e. $\|d - d'\| \leq R_f$;
   (c) Find geometrically consistent feature matches using RANSAC to solve for the fundamental matrix between two views;

2. Outlier detection
   (a) Compute density values for all the matched features on two views;
   (b) Detect outliers from dense to sparse based on an assumption that changes of both motions and feature characteristics of one match are consistent with those of neighbors after applying the global affine transformation on two views;

3. Guided matching
   (a) Compute density values for all the unmatched features on two views;
   (b) Find up to $k$ nearest neighbors in a relaxed threshold $R_f$ under the epipolar constraint for each unmatched feature and then select one neighbor as the reliable one by filtering out the outliers using the proposed outlier detection algorithm;
   (c) Incrementally find more reliable matches from dense to sparse.
3.1. Initial matching

Given a view pair \( \mathcal{V} \) and \( \mathcal{V}' \), local invariant features are extracted by the PCA-SIFT descriptor [28] and then represented by the \( \mathcal{H} \) descriptor [14] and \( \mathbf{c} \) denotes the location of the feature, \( \mathbf{d} \) is a 20-vector invariant descriptor and \( \mathbf{c} \) includes some additional feature characteristics such as scale and orientation for the features. Thus, the two sets of matches are expressed by \( \mathcal{M} = (\mathbf{x}, \mathbf{c}, \mathbf{d}) \) where \( \mathbf{x} = (x, y)^T \) denotes the location of the feature, \( \mathbf{d} \) is a 20-vector invariant descriptor and \( \mathbf{c} \) includes some additional feature characteristics such as scale and orientation for the features. Thus, the two sets of features extracted from \( \mathcal{V} \) and \( \mathcal{V}' \) are expressed by \( \mathcal{F} = \{ \mathbf{f}_l | \mathbf{f}_l = (x_k, c_k, d_k), k = 1, \ldots, N(\mathcal{V}) \} \) and \( \mathcal{F}' = \{ \mathbf{f}_l' | \mathbf{f}_l' = (x'_k, c'_k, d'_k), l = 1, \ldots, N(\mathcal{V}') \} \), respectively, where \( N(\mathcal{V}) \) and \( N(\mathcal{V}') \) are the total numbers of features extracted from \( \mathcal{V} \) and \( \mathcal{V}' \), respectively. For one feature \( \mathbf{f}_l \) in \( \mathcal{F} \), the ANN searching strategy is used to find an approximate nearest neighbor \( \mathbf{f}_l' \) in \( \mathcal{F}' \) with the Euclidean distance \( \text{dist}(\mathbf{f}_l, \mathbf{f}_l') = \| \mathbf{d}_l - \mathbf{d}_l' \| \leq R_f, R_f = 2500 \) was used in our work. Later on, the outliers are filtered out by the fundamental matrix between two views, which is automatically estimated using RANSAC. The final set of matches are expressed by \( \mathcal{M}_c = \{ \langle \mathbf{f}_l, \mathbf{f}_l' \rangle | \mathbf{f}_l \in \mathcal{F}, \mathbf{f}_l' \in \mathcal{F}' \} \) where \( \langle \bullet, \bullet \rangle \) denotes a two-view match.

3.2. Outlier detection

After initial matching, maybe there still exist a few outliers (incorrect matches). In this section, an outlier detection algorithm is developed to further detect possible outliers. In this paper, we will consider to match two views taken from two widely separated viewpoints, which may result in large changes in rotation, translation, zooming or deformation between these two views. To eliminate these changes to some extent, we estimate the global affine transformation between these two views based on the matches constrained by the fundamental matrix. Given a 2D point \( \mathbf{x} = (x, y)^T \) on \( \mathcal{V} \), the affine-transformed point \( \hat{\mathbf{x}} = (\hat{x}, \hat{y})^T = \mathbf{A}\mathbf{x} + \mathbf{B} \) with the affine transformation parameters \( (\mathbf{A}, \mathbf{B}) \) where \( \mathbf{A} \) is a \( 2 \times 2 \) matrix consisting of scaling and skew and \( \mathbf{B} \) is a \( 2 \times 1 \) translation vector. \( \mathbf{A} \) and \( \mathbf{B} \) are solved by minimizing the sum of the distances \( \text{dist}(\mathbf{f}_l, \mathbf{f}_l') \) where \( \langle \mathbf{f}_l, \mathbf{f}_l' \rangle \in \mathcal{M}_c \). Thus, the two sets of affine-transformed features corresponding to \( \mathcal{F} \) and \( \mathcal{F}' \) are defined by \( \hat{\mathcal{F}} = \{ \hat{\mathbf{f}}_l = (\hat{x}_k, \hat{c}_k, \hat{d}_k), k = 1, \ldots, N(\mathcal{V}) \} \) with the affine transformation parameters \( (\mathbf{A}, \mathbf{B}) \) and \( \hat{\mathcal{F}}' = \{ \hat{\mathbf{f}}_l' = (\hat{x}_l', \hat{c}_l', \hat{d}_l'), l = 1, \ldots, N(\mathcal{V}') \} \) with the affine transformation parameters \( (\mathbf{A}', \mathbf{B}') \), respectively. To more efficiently describe the proposed outlier detection algorithm, we define two sets of affine-transformed matches \( \mathcal{M}_c = \{ \langle \mathbf{f}_l, \mathbf{f}_l' \rangle | \mathbf{f}_l \in \mathcal{F}, \mathbf{f}_l' \in \mathcal{F}' \} \) from \( \mathcal{V} \) to \( \mathcal{V}' \) and \( \hat{\mathcal{M}}_c = \{ \langle \hat{\mathbf{f}}_l, \hat{\mathbf{f}}_l' \rangle | \hat{\mathbf{f}}_l \in \hat{\mathcal{F}}, \hat{\mathbf{f}}_l' \in \hat{\mathcal{F}}' \} \) from \( \mathcal{V}' \) to \( \mathcal{V} \) after applying the global affine transformation. Consider a two-view match \( \langle \mathbf{f}_l, \mathbf{f}_l' \rangle \in \mathcal{M}_c \), we get two two-view matches \( \langle \mathbf{f}_l, \hat{\mathbf{f}}_l' \rangle \in \hat{\mathcal{M}}_c \) and \( \langle \hat{\mathbf{f}}_l, \mathbf{f}_l' \rangle \in \hat{\mathcal{M}}_c \). Based on our observation and understanding, the matches in a small neighborhood tend to have the consistent location changes in views after the global affine transformation. Four typical examples are shown in Fig. 6 from which we can observe that the outliers with obvious inconsistent motion directions relative to those of other matches in their small neighborhoods are more easily to be detected. Given a two-view match \( \langle \mathbf{f}_l, \mathbf{f}_l' \rangle \in \mathcal{M}_c \), we check whether it has the consistent motion with its neighbors by considering two aspects: (1) whether the motion from \( \mathbf{f}_l \) to \( \mathbf{f}_l' \) is consistent with the neighbors around \( \mathbf{f}_l \) and (2) whether the motion from \( \mathbf{f}_l' \) to \( \mathbf{f}_l \) is consistent with the neighbors around \( \mathbf{f}_l' \).
where $M_{\text{otion}}(\bullet)$ is a motion consistent function, which is $\text{True}$ for consistence or $\text{False}$ for inconsistence. AND is the logical AND-operator. The value of $M_{\text{otion}}(\bullet)$ will be determined by the logical conjunction of $M_{\text{otion}}^{(x)}(\bullet)$ and $M_{\text{otion}}^{(y)}(\bullet)$ as follows:

$$
\begin{align*}
&M_{\text{otion}}(f_k \rightarrow \hat{f}_l) = M_{\text{otion}}^{(x)}(f_k \rightarrow \hat{f}_l) \ \text{AND} \ M_{\text{otion}}^{(y)}(f_k \rightarrow \hat{f}_l), \\
&M_{\text{otion}}(f_k \leftarrow \hat{f}_l) = M_{\text{otion}}^{(x)}(f_k \leftarrow \hat{f}_l) \ \text{AND} \ M_{\text{otion}}^{(y)}(f_k \leftarrow \hat{f}_l),
\end{align*}
$$

(4)

where $M_{\text{otion}}^{(x)}(\bullet)$ and $M_{\text{otion}}^{(y)}(\bullet)$ denote two motion consistent functions in the xth direction and the yth direction, respectively. For simplicity, here we only describe how to determine the value of $M_{\text{otion}}^{(x)}(f_k \rightarrow \hat{f}_l)$ and the values of other motion consistent functions can be determined in the same way. First, we select a suitable window with the width $2h+1$ (typically $h=\frac{1}{10} \max(W, H)$ where $W$ and $H$ denote the width and height in pixels of views $\mathcal{V}$ and $\mathcal{V}'$, respectively). Then we count the number of matched features in the given neighborhood $\mathcal{N}_h(f_k) = \{f_m | x_m-x_k| \leq h, |y_m-y_k| \leq h, f_m \in \mathcal{F}, (f_m, \hat{f}_m) \in M_c, m \neq k\}$ around the feature $f_k$. If it is less than 5, we broaden the half-width by $h = 2h$ and count it again. If it still is less than 5, we determine the value of $M_{\text{otion}}^{(x)}(f_k \rightarrow \hat{f}_l)$ by

$$
M_{\text{otion}}^{(x)}(f_k \rightarrow \hat{f}_l) = \begin{cases} 
\text{True}, & |x^{(c)}_k - \mu| \leq \max(8\sigma, \theta_g), \\
\text{False}, & |x^{(c)}_k - \mu| > \max(8\sigma, \theta_g),
\end{cases}
$$

(5)

where $x^{(c)}_k$ denotes the motion change in the xth direction from $f_k$ to $\hat{f}_l$, $x^{(c)}_k = \hat{x}^{(c)}_l - x_k$, and $\mu$ and $\sigma$ denote the mean and standard deviation of the data set $\{x^{(c)}_m | x_m-x_k| = \hat{x}^{(c)}_l - x_k, (f_m, \hat{f}_m) \in \mathcal{M}_c\}$, respectively. $\theta_g$ is the minimum acceptable length in pixels for the global motion change in the xth direction, typically $\theta_g = 10$. Otherwise, we calculate the mean absolute relative motion change $\bar{x}^{(c)}_k$ in the xth direction from $f_k$ to $\hat{f}_l$ relative to motion changes of matches $(f_m, \hat{f}_m), f_m \in \mathcal{N}_h(f_k)$ in the given neighborhood $\mathcal{N}_h(f_k)$ as follows:

$$
\bar{x}^{(c)}_k = \frac{\sum_{f_m \in \mathcal{N}_h(f_k)} w_m |x^{(c)}_m - x^{(c)}_k|}{\sum_{f_m \in \mathcal{N}_h(f_k)} w_m},
$$

(6)

where $x^{(c)}_k$ and $x^{(c)}_m$ denote the motion changes in the xth direction from $f_k$ to $\hat{f}_l$ and from $f_m$ to $\hat{f}_m$, respectively, $x^{(c)}_k = \hat{x}^{(c)}_l - x_k$ and $x^{(c)}_m = \hat{x}^{(c)}_l - x_m$. The coefficient $w_m$ is a weighting factor for the feature $f_m$ which is defined as

$$
w_m = K \left( \frac{x_m-x_k}{h} \right) K \left( \frac{y_m-y_k}{h} \right),
$$

(7)

where $K(\bullet)$ denotes a so-called kernel function and $h$ denotes the bandwidth. In this paper, the epanechnikov kernel function is selected for use, which is defined as $K(u) = \frac{3}{4}(1-u^2)$ when $|u| \leq 1$ or 0 when $|u| > 1$. The mean absolute relative motion change $\bar{x}^{(c)}_k$ in the xth direction from $f_m$ to $\hat{f}_m$ is calculated in the similar way but it does not include the match $(f_k, \hat{f}_l)$ whose matching state (inlier or outlier) is unknown and is to be determined. Thus, the utilized matched features in the given neighborhood around the feature $f_m$ is defined as $\mathcal{N}_h(f_m) = \{f_p | x_p-x_m| \leq h, |y_p-y_m| \leq h, f_p \in \mathcal{F}, (f_m, \hat{f}_m) \in M_c, p \neq m, p \neq k\}$. The weighted mean $\hat{\mu}$ and the weighted standard deviation $\hat{\sigma}$ of these mean absolute relative motion changes $\{x^{(c)}_m | f_m \in \mathcal{N}_h(f_k)\}$ are calculated by

$$
\hat{\mu} = \frac{\sum_{f_m \in \mathcal{N}_h(f_k)} w_m x^{(c)}_m}{\sum_{f_m \in \mathcal{N}_h(f_k)} w_m} \quad \text{and}
$$

$$
\hat{\sigma} = \sqrt{\frac{\sum_{f_m \in \mathcal{N}_h(f_k)} w_m (x^{(c)}_m - \hat{\mu})^2}{\sum_{f_m \in \mathcal{N}_h(f_k)} w_m}},
$$

(8)

where $w_m$ is a weighting factor for the feature $f_m$ which is computed using (7).

After that, we can determine the value of the motion consistent function $M_{\text{otion}}^{(x)}(f_k \rightarrow \hat{f}_l)$ by the following equation:

$$
M_{\text{otion}}^{(x)}(f_k \rightarrow \hat{f}_l) = \begin{cases} 
\text{True}, & |x^{(c)}_k - \hat{\mu}| \leq \max(3\hat{\sigma}, \theta_h), \\
\text{False}, & |x^{(c)}_k - \hat{\mu}| > \max(3\hat{\sigma}, \theta_h),
\end{cases}
$$

(9)

where $\theta_h$ is the minimum acceptable length in pixels of the absolute relative motion change in the xth direction in the given neighborhood with the width $2h+1$, typically $\theta_h = 3$.

To more efficiently detect possible outliers, we also check whether the match has the consistent changes over feature characteristics with their neighbors in the given neighborhood as

$$
F_{\text{c}}(f_k \leftrightarrow \hat{f}_l) = F_{\text{c}}(f_k \rightarrow \hat{f}_l) \ \text{AND} \ F_{\text{c}}(f_k \leftarrow \hat{f}_l),
$$

(10)

where $F_{\text{c}}(\bullet)$ is a feature characteristic consistent function, which is $\text{True}$ for consistence or $\text{False}$ for inconsistence. For example, the features extracted by the SIFT detector provide two additional feature characteristics: scale and orientation. The consistent changes over scale and orientation can be observed in Fig. 7. The value of the function $F_{\text{c}}(\bullet)$ can be determined in the same way as we determine the value of $M_{\text{otion}}(\bullet)$. For the SIFT features, $F_{\text{c}}(f_k \rightarrow \hat{f}_l) = F_{\text{s}}^{(s)}(f_k \rightarrow \hat{f}_l) \ \text{AND} \ F_{\text{s}}^{(o)}(f_k \rightarrow \hat{f}_l)$ and $F_{\text{c}}(f_k \leftarrow \hat{f}_l) = F_{\text{s}}^{(s)}(f_k \leftarrow \hat{f}_l) \ \text{AND} \ F_{\text{s}}^{(o)}(f_k \leftarrow \hat{f}_l)$ where the superscripts $s$ and $o$ stand for scale and orientation characteristics for the SIFT features.

Now, we can detect the outliers based on two consistent constraints as

$$
I_{\text{inlier}}(f_k, \hat{f}_l) = M_{\text{otion}}(f_k \leftrightarrow \hat{f}_l) \ \text{AND} \ F_{\text{c}}(f_k \leftrightarrow \hat{f}_l),
$$

(11)

where $I_{\text{inlier}}(f_k, \hat{f}_l) = \text{True}(\text{inlier})$ or $\text{False}(\text{outlier})$.

For detecting the outliers more efficiently, we test all the matches from dense to sparse because a match with more matches around it can be detected as an inlier or an outlier with a high-confidence level. The density values of each matched features will be computed by using the same kernel function and the width used in Eq. (7).

3.3. Guided matching

By the guidance of the estimated fundamental matrix, we search more matches from one view around their corresponding epipolar lines on another view. To find more matches, we
relax the threshold $R_f, R_f = 5000$ was used in this paper. Then we sort all the un-matched features on views $r'$ and $r''$ by their corresponding density values in the descending order and handle successively them. In most cases, given a feature on one view, the nearest neighbor (feature) in the Euclidean distance on another view is the best match. However, the nearest neighbor is not always the best match due to occlusion and deformation derived from large viewpoint changes and repeated structures in the scene especially after the threshold $R_f$ is relaxed. In order to find more reliable feature matches, for one feature on one view, we search $k$ nearest effective neighbors (features), satisfying that their Euclidean distances are less than the given threshold $R_f$ as the initial candidate matches. For an un-matched feature on one view, one of $k$ nearest neighbors (features) on another view may be the best match. Among these $k$ nearest neighbors, maybe some features have been matched and they are first filtered out. Then we check the remaining matches in the ascending order by their corresponding Euclidean distances based on the above described outlier detection algorithm. When a reliable match is found, we add it into the current set of matches $\mathcal{M}_c$. Then we re-calculate the density values of all the features affected by the newly added match and sort all the remaining un-matched features again. Later on, we handle all the remaining un-matched features in the above order again. In this way, we can find a mass of reliable matches between two views. With the increase of the number of matches in $\mathcal{M}_c$, the proposed outlier detection algorithm tends to be more efficient. Before performing the guided matching, some matches, which are correct, may be detected as the outliers due to a small quantity of neighboring features around them. Fortunately, most of them can be collected as reliable matches again with the increase of the number of the added matches. Of course, before performing the guided matching, some matches, which are wrong matches, may not be filtered out due to the same reason. So we test all the matches found again based on the proposed outlier detection algorithm after the guided matching.

The robust matching algorithm proposed in this section is a general matching scheme, which is also suitable for matching other kinds of features between two views by only selecting different feature similarity metrics and correspondingly adjusting the thresholds.

4. Multi-view matching

Given a small number of view pairs selected on a constructed view-spanning tree obtained by applying the proposed view-ordering algorithm described in Section 2 on an unordered set of views, our objective in this section is to build up many more reliable multi-view feature matches in order to support further camera calibration and 3D scene reconstruction. Based on the construction approach of the view-spanning tree, we thought that two views across more links on the constructed view-spanning tree will have less similarities. The view pairs across at most $L$ links are selected for further robust two-view matching. Based on different requirements, we set different $L$, $L = 1, 2, \ldots, N_t - 1$ where $N_t$ is the total number of views on the constructed view-spanning tree. In general, more view pairs selected in this way will produce more feature matches across multi-views. Of course, the needed computation time will also increase. After we select some view pairs, we perform the proposed robust two-view matching algorithm on them.

After matching all the selected view pairs, generally people directly utilize the trifocal constraint among a triple of neighboring views to filter out outliers. Here, we propose a new algorithm to track as many reliable matches as possible across multiple views in three stages. In the first stage, we reconstruct the missing matches among any one triple of views $(V_a, V_b, V_c)$ based on a fact that if there exist two matches $(f_a, f_b)$ between $V_a$ and $V_b$ and $(f_b, f_c)$ between $V_b$ and $V_c$ then there must exist a match $(f_a, f_c)$ as well between $V_a$ and $V_c$. If the fundamental matrix between $V_a$ and $V_c$ has been estimated, it is used to determine whether the reconstructed match is added. Otherwise, the reconstructed match will be added unconditionally. We iteratively reconstruct all the missing matches among all the triples of views at least containing two selected view pairs, which have been selectively matched, until no more missing matches can be found. Since there exist some unreliable matches between view pairs, many unreliable matches are reconstructed as well.

In the second stage, we want to determine the states of all the two-view matches found in the first stage as follows. We define three states for reliable, unreliable and unknown-stated matches: RELIABLE, UNRELIABLE and UNKNOWN. First we compute the fundamental matrices using the reconstructed matches for those view pairs which are not directly selected for two-view matching. It is often needed when we select neighboring view pairs on the constructed view-spanning tree. All those matches satisfying the two-view constraint based on the estimated fundamental matrices are marked as UNKNOWN. Other matches are marked as UNRELIABLE. Then we convert the states of the triples of matches all marked by UNKNOWN into RELIABLE as shown in Fig. 8(a). Later on, we discover the spurious matches based on a fact that a feature on one view cannot be matched to multiple different features on another view and these matches hold UNKNOWN or RELIABLE.
When we find a triple of matches consisting of multi-view matches over the constructed view-spanning tree are consisting of at least two selected view pairs. Then all the focal tensor, which is automatically estimated by a set of feature system from an unordered set of views is described in Sections 2, 3 and 4, respectively, were used to organize these unordered views into several view-spanning trees, then build up correspondences between view pairs selected on the constructed trees and finally establish multi-view feature matches. The found multi-view matches are used to automatically create the projective reconstruction of the rigid scene [32]. A flexible self-calibration method is applied to upgrade the projective scene reconstruction to metric [33]. Then we use the bundle adjustment technique [34] to simultaneously refine both 3D structure and viewing parameters (camera pose and calibration) to obtain an optimal visual reconstruction. A generic sparse bundle adjustment software package provided by Lourakis et al. [35] was used to efficiently complete this task. Finally, the VRML scene models are generated by triangulation interpolation.

### 6. Computational complexity analysis

In this section, we briefly describe the computational complexity of the whole matching algorithms we propose.

Our proposed view-ordering algorithm mainly in time depends on the SIFT detector [14], the PCA-SIFT descriptor [28], the ANN searching strategy and the proposed feature-reducing approach. The construction time of the ordered view-spanning trees for an unordered set of images can be ignored. Here, we do not discuss the complexity of the SIFT detector and the PCA-SIFT descriptor and the interested readers can refer to their papers [14,28]. Next we focus on the discussion of the computational complexity of the used ANN search strategy. Given any positive real ε, a data point $p \in \mathbb{R}^d$ is a $(1+\epsilon)$-approximate nearest neighbor of a query point $q \in \mathbb{R}^d$ if its distance from $q$ is within a factor of $(1+\epsilon)$ of the distance to the true nearest neighbor. Arya et al. [29,30] found that given an integer $k \geq 1$, $(1+\epsilon)$-approximations to the $k$ near-

![Fig. 8. Conversion of matching states for: (a) the triples of consistent matches; (b) the one-to-more matches and (c) the triples of conflicted matches.](image)
est neighbors of a query point \( q \in \mathbb{R}^d \) can be found in a set of \( N \) data points in \( \mathbb{R}^d \) in \( O(kd \log N) \) time. Based on this analyzed result, the view-versus-feature table for \( N \) views, which should be used to organize all the \( N \) views, can be computed in \( O(\sum_{i=0}^{N-1} k d N(\hat{Y}_i)) \log \left( \sum_{i=0}^{N-1} j \neq i (s N(\hat{Y}_j)) \right) \) time where \( N(\hat{Y}_i) \) is the number of feature points on the view \( \hat{Y}_i \) and \( d = 20 \) is the dimension of the PCA-SIFT represented features. Furthermore, the proposed feature-reducing approach can greatly reduce the computational time for organizing an unordered set of images, which should be reduced to \( O(\sum_{i=0}^{N-1} k d (s N(\hat{Y}_i)) \log \left( \sum_{i=0}^{N-1} j \neq i (s N(\hat{Y}_j)) \right) \) where \( s \leq 1 \) is the feature-reducing factor. It can be rewritten as \( O(sT - s \log(1/s)kd\sum_{i=0}^{N-1} N(\hat{Y}_i)) \) where \( T = \sum_{i=0}^{N-1} k d N(\hat{Y}_i) \log \left( \sum_{i=0}^{N-1} j \neq i (s N(\hat{Y}_j)) \right) \) is the original computational complexity before reducing the number of features. This computational complexity analysis was examined successfully by the results shown in Fig. 5.

The computational time of the proposed two-view matching algorithm depends on the computational times of initial matching, outlier detection and guided matching. In initial matching step, it depends on the used ANN searching strategy which has been analyzed above and the RANSAC method for computing the fundamental matrix between two views. In outlier detection step, outliers among \( c \) two-view matches found in initial matching step can be detected in \( O(c^2_{\hat{r}_n}) \) where \( c_{\hat{r}_n} \) is the average number of used neighboring feature points around all the matched points. In guided matching step, we can find all the matched point from the \( r_1 \) and \( r_2 \) remaining points in two views, respectively, in \( O(k c_{\hat{r}_n}^2 (r_1 + r_2)) \) where \( c_{\hat{r}_n} \) has the same meaning above and \( k \) is the number of nearest neighbors which should be used to match.

Finally, for the proposed multi-view matching algorithm, it is difficult for us to definitely analyze its computational complexity. But it mainly depends on the number of input images, the number of chosen view pairs spanning at most \( L \) links over the constructed view-spanning tree and the verification of multi-view matches in three stages described in Section 4.

7. Experimental results

In this section we present and discuss the experimental results of the two-view matching algorithm proposed in Section 3 and the multi-view matching algorithm proposed in Section 4.

7.1. Testing on the proposed two-view matching algorithm

The proposed two-view matching algorithm was tested on many pairs of real images and the matching results of the selected eight view pairs in case of zooming, rotation, translation,

<table>
<thead>
<tr>
<th>Images (( \hat{Y}_i ), ( \hat{Y}_j ))</th>
<th>Size (in pixels)</th>
<th>( (N(\hat{Y}_i), N(\hat{Y}_j)) )</th>
<th>( N(0)(\hat{M}_i) )</th>
<th>( N(1)(\hat{M}_i) )</th>
<th>( N(2)(\hat{M}_i) )</th>
<th>( N(3)(\hat{M}_i) )</th>
<th>( N N(%) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{Y}_i )</td>
<td>760 \times 555</td>
<td>(5185,1588)</td>
<td>119</td>
<td>86</td>
<td>84</td>
<td>272</td>
<td>76 (27.9%)</td>
</tr>
<tr>
<td>( \hat{Y}_j )</td>
<td>850 \times 680</td>
<td>(13114,7100)</td>
<td>1401</td>
<td>1159</td>
<td>1154</td>
<td>2828</td>
<td>874 (30.9%)</td>
</tr>
<tr>
<td>( \hat{Y}_k )</td>
<td>768 \times 576</td>
<td>(7086,7599)</td>
<td>2367</td>
<td>1988</td>
<td>1985</td>
<td>2581</td>
<td>321 (12.4%)</td>
</tr>
<tr>
<td>( \hat{Y}_l )</td>
<td>800 \times 640</td>
<td>(8849,8187)</td>
<td>1268</td>
<td>663</td>
<td>658</td>
<td>2248</td>
<td>1376 (61.2%)</td>
</tr>
<tr>
<td>( \hat{Y}_m )</td>
<td>850 \times 680</td>
<td>(5886,7058)</td>
<td>641</td>
<td>435</td>
<td>418</td>
<td>1123</td>
<td>447 (39.8%)</td>
</tr>
<tr>
<td>( \hat{Y}_n )</td>
<td>800 \times 640</td>
<td>(5978,7612)</td>
<td>285</td>
<td>64</td>
<td>64</td>
<td>710</td>
<td>316 (44.5%)</td>
</tr>
<tr>
<td>( \hat{Y}_o )</td>
<td>850 \times 680</td>
<td>(5020,4195)</td>
<td>1008</td>
<td>759</td>
<td>756</td>
<td>1262</td>
<td>295 (23.4%)</td>
</tr>
<tr>
<td>( \hat{Y}_p )</td>
<td>768 \times 512</td>
<td>(8423,7772)</td>
<td>1603</td>
<td>1325</td>
<td>1317</td>
<td>3160</td>
<td>948 (30%)</td>
</tr>
</tbody>
</table>
Fig. 9. Robust two-view matching for two scenes—Inria Model (left) and Cityhall (right): (a) original images; (b) the images with the extracted SIFT features marked by red points; (c) a set of initial matches obtained by ANN with $R_f = 2500$; (d) a set of matches consistent with the estimated fundamental matrix, in which the matches marked by dark lines are the outliers detected by the proposed outlier detection algorithm; (e) the final set of found matches $M_c$ after guided matching by relaxing $R_f = 5000$ and (f) the two sets of the affine-transformed matches $\hat{M}_c$ and $\hat{M}_c'$, corresponding to $M_c$.

Viewpoint changes or their combination [31,36,37] are shown in Table 2. The matching results on two scenes as shown in the fifth and eighth rows in Table 2 are presented in Fig. 9 in the graphical form again. In this experiment, the threshold $R_f = 2500$ was used in the initial matching stage and $R_f = 5000$ was used in the guided matching stage. In all experiments in this paper, the same thresholds were used. From these experimental results shown in Table 2 and Fig. 9, we observe that there still exist a few outliers consistent with the estimated fundamental matrix. Two-view matches in all the experiments in this paper are confined within one pixel distance derived from the epipolar line computed from the estimated fundamental matrix between two views. The proposed outlier detection algorithm can efficiently detect the outliers. After guided matching, we can find a mass of reliable matches. In these eight view pairs, about 30–40% matches shown in the last column in Table 2, which are not the nearest neighbors each other under the epipolar geometric constraint, over all the final matches found were robustly detected. These matches cannot be found by using traditional guided matching algorithms.

In the above experiment on real images, we do not know whether the final matches found are correct correspondences. For more sufficient evaluation, synthetic image pairs were utilized to test our algorithm again and compare the results with other methods. In general, the best candidate match for one feature in one image is found by identifying its first nearest neighbor (feature) in another image. For comparison, this method is labeled as “nearest neighbor”. To remove the incorrect matches, Lowe [14] proposed to verify a match by comparing the match distance $e_{1-NN}$ of the first best match (first nearest neighbor) to the match distance $e_{2-NN}$ of the second best match (second nearest neighbor). This strategy was also adopted in Ref. [4]. Here, the first best match for one feature was identified.
Fig. 11. Three synthetic images by projection from a three-orthogonal-rectangle model with texture mapping.

as a correct match only when the ratio $e_{1-NN}/e_{2-NN} < 0.8$.

In this experiment, synthetic images were generated as follows. A model consisting of three orthogonal rectangles with one intersection point in 3D space was first created and then three real images were selected to be mapped to these three rectangles as shown in Fig. 10. Three synthetic images $V_1$, $V_2$ and $V_3$, which were obtained by projection from the three-orthogonal-rectangle model at different positions, are shown in Fig. 11. The SIFT detector and the PCA-SIFT descriptor were used to extract the features and represent them in invariant vectors, respectively. For these three synthetic images, manually identifying matches between them would be extremely time-consuming and potentially error-prone.

Fortunately, three rectangles can be used to check whether a match is a correct one. These three virtual cameras corresponding to three synthetic images as shown in Fig. 11 were automatically calibrated by using 1673 three-view matches within one pixel distance [33]. These three-view matches were obtained by first performing the proposed two-view matching algorithm on two view pairs $(V_1, V_2)$ and $(V_2, V_3)$, then connecting these two-view matches into three-view matches and finally rejecting outliers by trifocal tensor. In most cases, these three-view matches are correct. Three planes were estimated by manually marking the correspondences at seven vertices as shown in Fig. 10. Then we grouped the reconstructed scene 3D points onto three estimated planes. These planes

Fig. 12. Comparing results of our proposed method, nearest neighbor method and Lowe’s method on three pairs of synthetic images: (a)–(c) the curves of the numbers of correct matches versus different $z$-scores on view pairs $(V_1, V_2)$, $(V_2, V_3)$ and $(V_1, V_3)$, respectively; and (d)–(f) the curves of the percentages of correct matches over all the matches found versus different $z$-scores on image pairs $(V_1, V_2)$, $(V_2, V_3)$ and $(V_1, V_3)$, respectively.
Table 3
Numbers of multi-view matches (tracks) and reconstructed 3D scene points for the Valbonne church sequence

<table>
<thead>
<tr>
<th>$k$</th>
<th>Views</th>
<th>$L = 1$ (14 links)</th>
<th>$L \leq 2$ (33 links)</th>
<th>$L \leq 3$ (54 links)</th>
<th>$L = \text{any}$ (105 links)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tracks</td>
<td>Points</td>
<td>Tracks</td>
<td>Points</td>
<td>Tracks</td>
</tr>
<tr>
<td>2</td>
<td>11,12</td>
<td>578</td>
<td>578</td>
<td>663</td>
<td>666</td>
</tr>
<tr>
<td>3</td>
<td>9,11,12</td>
<td>401</td>
<td>728</td>
<td>443</td>
<td>918</td>
</tr>
<tr>
<td>4</td>
<td>6,9,11,12</td>
<td>272</td>
<td>926</td>
<td>323</td>
<td>1167</td>
</tr>
<tr>
<td>5</td>
<td>6,9,11,12,13</td>
<td>157</td>
<td>1054</td>
<td>173</td>
<td>1286</td>
</tr>
<tr>
<td>6</td>
<td>5,6,9,11,12,13</td>
<td>128</td>
<td>1200</td>
<td>150</td>
<td>1456</td>
</tr>
<tr>
<td>7</td>
<td>3,5,6,9,11,12,13</td>
<td>96</td>
<td>1407</td>
<td>118</td>
<td>1644</td>
</tr>
<tr>
<td>8</td>
<td>2,3,5,6,9,11,12,13</td>
<td>64</td>
<td>1471</td>
<td>78</td>
<td>1706</td>
</tr>
<tr>
<td>9</td>
<td>1,2,3,5,6,9,11,12,13</td>
<td>40</td>
<td>1501</td>
<td>50</td>
<td>1731</td>
</tr>
<tr>
<td>10</td>
<td>1,2,3,5,6,9,11,12,13,14</td>
<td>19</td>
<td>1533</td>
<td>20</td>
<td>1757</td>
</tr>
<tr>
<td>15</td>
<td>All 15 views</td>
<td>3</td>
<td>1602</td>
<td>3</td>
<td>1838</td>
</tr>
</tbody>
</table>

were further estimated by the RANSAC method and the least square method over three sets of scene points on these three planes. Subsequently, we computed the mean and the standard deviation of the (signed) distances between these grouped scene points and their corresponding planes, which are denoted by $\mu_d$ and $\sigma_d$, respectively. Given a two-view match, we first reconstructed its corresponding 3D scene point and then determined which plane it is most likely located on by finding out which plane gives a minimal absolute point-to-plane distance. In this experiment, this match was considered as a correct one only when the (signed) point-to-plane distance is limited in the range $[\mu_d - z\sigma_d, \mu_d + z\sigma_d]$ where larger $z$-score results in that the match is less reliable. We tested our proposed method, the nearest neighbor method and Lowe’s method over different $z$-scores on three view pairs $(V_1, V_2)$, $(V_2, V_3)$ and $(V_1, V_3)$. We evaluated these three methods from two aspects. The first one is the total number of correct matches whose corresponding (signed) point-to-plane distances are limited in the range $[\mu_d - z\sigma_d, \mu_d + z\sigma_d]$. The second one is the percentage of correct matches over all the matches found. The experimental results are shown in Fig. 12. From the curves as shown in Figs. 12(a)–(c), we observe that the proposed method can detect more correct matches at different $z$-scores compared with other two methods. Also, we can observe that the advantage is more prominent under larger variations in viewing conditions, for example, the advantage of the proposed method on the image pair $(V_1, V_3)$ as shown in Fig. 12(c) is more obvious than the view pairs $(V_1, V_2)$ and $(V_2, V_3)$ as shown in Figs. 12(a)(b). This is because that larger view deformation results in larger texture deformation around the extracted features in this case that the nearest neighbor is inadequate for matching. From the curves as shown in Figs. 12(d)–(e), we see that the percentages of correct matches over all the matches found by using the proposed method are higher than the ones provided by the nearest neighbor method. However, it is lower than the ones provided by Lowe’s method. From this aspect, Lowe’s method holds higher reliability but it pays heavy cost on the numbers of correct matches found as shown in...

Fig. 13. 356 4-view matches across views 3, 5, 6 and 9 of the Valbonne church sequence.
Figs. 12(a)–(c). From the curves as shown in Fig. 12(f), the percentages of correct matches over all the matches found by using the proposed method are lower than the ones provided by the nearest neighbor method and Lowe’s method but the proposed method can detect much more correct matches than these two methods as shown in Fig. 12(c). Under the z-score of 3.0, the proposed method detected 1043 correct matches between two images $Y_1$ and $Y_3$ while the nearest neighbor method and Lowe’s method only detected 676 and 331 correct ones, respectively. As a whole, the proposed method can detect a mass of correct matches and hold good reliability especially under large variations in viewing conditions.

7.2. Testing on the proposed multi-view matching algorithm

In this section, the proposed multi-view matching scheme was tested on many sets of images. Here we report some results on three different sets of real images where no ordering information is provided. The first one is the Valbonne church sequence, which presents a case with an unordered set of images of the environment. The second one is the Statue sequence consisting of five images [38]. The final one is the cityhall sequence with strong wide baseline.

Valbonne: In this experiment, we used 15 input images of size 512 × 768 pixels, available at [31]. First we extracted the features independently from each input image by the SIFT feature detector [14] and then represented them by the PCA-SIFT descriptor [28]. Then we organized these unordered input images throughout constructing a view-spanning tree over them with the feature-reducing factor $s = 0.2$. The constructed view-spanning tree is shown in Fig. 2(b). To evaluate how the number of selected view pairs affects the final matching results, we selected view pairs across at most one link ($L = 1$), two links ($L \leq 2$), three links ($L \leq 3$) and all view pairs ($L = \text{any}$), respectively. Based on these four sets of view pairs consisting of 14, 33, 54 and 105 view pairs, respectively, we matched them using the robust two-view matching algorithm described in Section 3 and then found a large set of reliable matches across multiple views. Finally, we automatically calibrated all the views using these tracked multi-view matches and then filtered out some outliers by bundle adjustment. The numbers of multi-view matches (tracks) found and reconstructed...
Table 4
Numbers of multi-view matches (tracks) and reconstructed 3D scene points for the cityhall sequence

<table>
<thead>
<tr>
<th>k</th>
<th>Views</th>
<th>Tracks ($L=1$)</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3,4</td>
<td>1539</td>
<td>1539</td>
</tr>
<tr>
<td>3</td>
<td>0,3,4</td>
<td>363</td>
<td>1563</td>
</tr>
<tr>
<td>4</td>
<td>1,0,3,4</td>
<td>361</td>
<td>3102</td>
</tr>
<tr>
<td>5</td>
<td>2,1,0,3,4</td>
<td>358</td>
<td>3115</td>
</tr>
<tr>
<td>6</td>
<td>2,1,0,3,4,5</td>
<td>122</td>
<td>3331</td>
</tr>
<tr>
<td>7</td>
<td>2,1,0,3,4,5,6</td>
<td>49</td>
<td>3331</td>
</tr>
</tbody>
</table>

3D scene points across the selected multiple views are shown in Table 3. We observe that the proposed multi-view matching scheme can find a large number of matches across multiple widely separated views, which is essential for further camera calibration and scene reconstruction. As k, the number of views used, increases, the number of complete multi-view matches (tracks) across all these k views decreases rapidly. However, the number of reconstructed points tends to increase. When we selected only 14 neighboring view pairs (i.e. $L = 1$) for the multi-view matching process, we detected 18 matches across 10 views, which is about two times of the uncalibrated tracking result (11 tracks) from all view pairs reported in Ref. [7]. There are more improvements on the tracking results compared with the results reported in Ref. [6] where no 10-view match can be successfully found in the case that the total number of extracted features from 15 images is about the same with number of extracted SIFT features in this experiment. In addition, we also observe when we selected more view pairs, the number of found tracks across multiple views do not obviously increase especially when we selected view pairs across three or more links. In the proposed multi-views matching scheme, generally we only need to select neighboring view pairs (i.e. $L = 1$) or select view pairs across at most two links (i.e. $L \leq 2$). The tracking results across views 3, 5, 6 and 9 when we selected neighboring view pairs (i.e. $L = 1$) are shown in Fig. 13. The reconstructed point clouds and tracked cameras for all the views with $L = 1$ are shown in Fig. 14. In the same case, a 3D model expressed by a triangular wire-frame mesh with texture mapping is shown in Fig. 15.

**Statue:** This is a set of five images taken under wide-baseline conditions [38]. This scene contains a statue in front of a far wall. The proposed two-view matching algorithm was first
performed on four neighboring view pairs (i.e. \( L = 1 \)). Then multi-view matches were found and structure and motion were recovered. The reconstructed point clouds and tracked cameras for all five views are shown in Fig. 16. A VRML scene model is shown in Fig. 17.

Cityhall: Seven images of size 768 × 512 pixels, which are \( \frac{1}{4} \) of original size 3072 × 2048 ones [36,37], were used in this experiment. Totally 47,352 SIFT features were extracted from these seven views. This sequence presents a case of multiple strong wide-baseline views taken from a complicated 3D structure and large depth discontinuity. The constructed view-spanning tree is 2-1-0-3-4-5-6 with the feature-reducing factor \( s = 0.2 \). The numbers of multi-view matches found and reconstructed 3D scene points across the selected views are shown in Table 4. The tracking results across views 1–3 are shown in Fig. 18. The reconstructed point clouds and tracked cameras are shown in Fig. 19 and a VRML scene model is shown in Fig. 20.

8. Conclusions

An efficient and robust multi-view feature matching scheme is proposed in this paper. Given an unordered of uncalibrated views taken from different widely separated viewpoints, we organize them on the view-spanning trees by utilizing only part of extracted features to match, which will greatly reduce the whole computation time and obtain a satisfactory ordering result. A robust two-view matching algorithm is proposed to find reliable matches between a small number of view pairs. We assume that both motion and feature characteristics of matches tend to have consistent changes after applying the global affine transformation. This fact is utilized to detect possible outliers and incrementally find more reliable matches from dense to sparse. Experimental results on real data and synthetic data show that the proposed two-view matching algorithm can detect a large set of reliable feature matches in most cases. Based on the two-view matches found between a small number of view pairs, a large set of reliable multi-view matches can be established by reconstructing missing matches and efficiently determining the states of matches in neighboring triples of views. A large set of reliable multi-view matches can help to establish good metric reconstruction and also become a good initialization for further producing dense matching.

Acknowledgements

The work described in this paper was substantially supported by a grant from the Research Grants Council (RGC) of the Hong Kong Special Administrative Region (Project no. CUHK4167/01E).

References


[31] [http://www.robots.ox.ac.uk/~vgg/data.html].


About the Author—JIAN YAO received the B.Sc. degree and M.Sc. degree from the Department of Automation, Xiamen University in September 1997 and the School of Computer, Wuhan University in September 2000 respectively. From 2000 to 2002, he has ever worked on chaos-based encryption and chaos-based secure communication at Shenzhen R&D Centre of City University of Hong Kong. He received the Ph.D. degree in the Department of Electronic Engineering, The Chinese University of Hong Kong in August, 2006. Currently he is working in IDIAP Research Institute, Switzerland as a PostDoc. His current research interests mainly include Image-based Modeling and Rendering, Multi-view Feature Matching, Multi-view Dense Matching, 3D Face Modeling, 3D Person Tracking, Chaos-based Communication, etc.

About the Author—WAI-KUEN CHAM graduated from The Chinese University of Hong Kong in 1979 in Electronics. He received his M.Sc. and Ph.D. degrees from Loughborough University of Technology, U.K., in 1980 and 1983, respectively. His research work was on image coding and in close collaboration with British Telecom. From June 1984 to April 1985, he was a senior engineer in Datacraft Hong Kong Limited and a lecturer in the Department of Electronic Engineering, Hong Kong Polytechnic (now The Polytechnic University of Hong Kong). Since May 1985, he has been with the Department of Electronic Engineering, the Chinese University of Hong Kong. His research interests include image and video signal processing and face recognition. He has published over hundred papers in international journals and proceedings of major conferences and is the inventor of a number of patents. He has acted as a consultant to a number of government and industrial organizations, such as the Hong Kong Consumer Council, Grace Everett Ltd., COMedia HK Ltd., and Jet Propulsion Laboratory, NASA. He is the inventor of the Integer Cosine Transform which was used in the NASA Galileo space mission and is adopted in the international video coding standard H.264/AVC and the emerging national standard of China AVS (Audio Video Coding Standard). He is a Chartered Engineer and a senior member of IEEE. His current interests include Digital Signal Processing, Image Coding, Image Processing, Video Segmentation, Edge-model-based Image Processing, 3D Head Model Reconstruction from Video, Multi-view Matching, Facial Feature Extraction and Face Recognition.