Multi-view Matching for Unordered Image Sets, or "How Do I Organize My Holiday Snaps?"

F. Schaffalitzky and A. Zisserman

Robotics Research Group
University of Oxford
{fsm,az}@robots.ox.ac.uk

Abstract. There has been considerable success in automated reconstruction for image sequences where small baseline algorithms can be used to establish matches across a number of images. In contrast in the case of widely separated views, methods have generally been restricted to two or three views.

In this paper we investigate the problem of establishing relative viewpoints given a large number of images where no ordering information is provided. A typical application would be where images are obtained from different sources or at different times: both the viewpoint (position, orientation, scale) and lighting conditions may vary significantly over the data set.

Such a problem is not fundamentally amenable to exhaustive pair wise and triplet wide baseline matching because this would be prohibitively expensive as the number of views increases. Instead, we investiate how a combination of image invariants, covariants, and multiple view relations can be used in concord to enable efficient multiple view matching. The result is a matching algorithm which is linear in the number of views.

The methods are illustrated on several real image data sets. The output enables an image based technique for navigating in a 3D scene, moving from one image to whichever image is the next most appropriate.

1 Introduction

Our objective in this work is the following: given an unordered set of images, divide the data into clusters of related (i.e. from the same scene) image and determine the viewpoints of each image, thereby spatially organizing the image set. The need for solving this problem arises quite commonly. For example, the image set may have been acquired by a person photographing a scene (e.g. a castle or mountain) at various angles while walking back and forth around the area. Or the set may be the response from a query to an image database (e.g. a web search engine). Typical examples of such image sets are given in figures 1 and 2.

Much of the research on structure and motion recovery has concentrated on image sequences, and there are two consequences of this: first, there is an *ordering* on the image set, and it is natural to use this ordering, at least initially, in the processing; second, it has allowed small baseline algorithms to be used

A. Heyden et al. (Eds.): ECCV 2002, LNCS 2350, pp. 414–431, 2002.

Fig. 1. Fifteen images of the church in Valbonne, France.

Fig. 2. Fifteen images (from a set of 46) of Raglan Castle, Wales.

between consecutive frames of the sequence. Many successful systems strongly use this ordering [1,3,4] as a means of sewing together long feature tracks through the sequence and generating initial structure and camera estimates. Given these estimates further matches may then be sought between non-contiguous frames based on approximate spatial overlap of the views [7,17].

In our case we do not have an ordering, so there is no natural sequence in which to process the images, and also we cannot know *a priori* that between-view motions are small. Consequently we will require wide baseline matching methods [2,10,12,11,15,16,18,21,23,24] to compute multiview relations. In practice, computing view clusters will be difficult because wide baseline matching is difficult but in this paper we will see how far we can go by applying imperfect wide baseline stereo techniques to multi-view data sets.

How to proceed? We could attempt to mimic the image sequence (ordered set) processing paradigm and compute a fundamental matrix between all N-choose-2 views (using a suitable wide baseline algorithm), then follow this by computing trifocal geometry over all N-choose-3 views, and finally sew together the resulting matches into tracks throughout our view set. Clearly we would rather not do this as it will become prohibitively expensive as N increases – a complexity of at least $O(N^3)$ for trifocal geometry and to this must be added the cost of sewing together all three-view matches.

Instead we divide the problem into three stages: first (section 2) we develop an efficient indexing scheme based on invariant image patches. The output is a table of features vs views, so that for each feature point its putative matches over multiple views are known. The table at this stage will contain many ambiguous (i.e. not one-to-one matches) and many erroneous matches. The overall complexity of this stage is data dependent but the main parameter is the total number of features in the data set, which is linear in the number of images.

In the second stage (section 3) the quality of the matches is improved by a number of global "clean-up" operations such as selective use of two-view and three-view matching constraints. The output is a feature vs view table with considerably more correct matches, and fewer incorrect matches. The complexity of this stage, which is opportunistic, is linear in the number of views.

In the third stage (section 4) a 3D reconstruction of cameras and points is computed for connected sub-sets of views using the multiple view tracks. The difficulty is that between some views there will be no matches because the viewpoint footprints do not overlap or the images might even have been taken from two different scenes. In either case, the set of input images should be expected to split into clusters and the objective is to find those clusters, the larger the better.

Once the cameras are computed the images can be viewed coherently by embedding them in a VRML model, or by driving an image based renderer directly from the estimated scene geometry.

2 From Images to Multiview Matches

In this section our objective is to efficiently determine putative multiple view matches, i.e. a point correspondence over multiple images.

To achieve this we follow the, now standard, approach in the wide baseline literature, and start from features with viewpoint invariant descriptors. The viewpoint transformations which must be considered are an affine geometric transformation and an affine photometric transformation.

Features are determined in two stages: first, regions which transform covariantly with viewpoint are detected in each image, second, a vector of invariant descriptors is then computed for each region. The invariant vector is a label for that region, and will be used as an index into an indexing structure for matching between views — the corresponding region in other images will (ideally) have an identical vector.

These features are determined in all images independently. The descriptors for all images are then stored in the indexing structure. Features with 'close' descriptors establish a putative multiple view match. These steps are described in more detail in the following subsections.

Fig. 3. Invariant neighbourhood process. Left: close-up of original image. Middle: three detected interested points, their associated scale indicated by the size of the circles. Right: the corresponding affine-adapted neighbourhoods.

2.1 Covariant Regions

We use two types of features: one based on interest point neighbourhoods, the other based on the "Maximally Stable Extremal" (MSE) regions of Matas and Paleček [11]. Both features are described in more detail below. Each feature defines an elliptical region which is used to construct an invariant descriptor.

Invariant neighbourhoods: In each image independently we compute interest points, to each of which is associated a characteristic scale. The scale is computed using the method of [13] and is necessary in order to handle scale changes between the views. For each point we then attempt to compute an affine invariant

neighbourhood using the method proposed by Baumberg [2]. The method is an adaptive procedure based on isotropy of the second moment gradient matrix [8]. If successful, the output is an image point with an elliptical neighbourhood which transforms co-variantly with viewpoint Similar neighbourhoods have been developed by Mikolajczyk and Schmid [14].

For a 768×512 image the number of neighbourhoods computed is typically 2000 but the number depends of course on the image. See figure 3 for an example. The computation of the neighbourhood generally succeeds at points where there is signal variation in more than one direction (e.g. near "blobs" or "corners").

To illustrate "what the computer sees", figure 4 shows just those parts of an image which are in a support region of some invariant neighbourhood.

Fig. 4. The support regions used to compute invariants are shown here using original image intensities. The representation is clearly dense and captures most of the salient image parts while discarding smooth regions, such as the sky. The large "sea" in the right-most image is a featureless lawn.

MSE regions: The regions are obtained by thresholding the intensity image and tracking the connected components as the threshold value changes. A MSE region is declared when the area of a component being tracked is approximately stationary. The idea (and implementation used here) is due to Matas and Paleček [11] (see figure 5 for an example). Typically the regions correspond to blobs of high contrast with respect to their surroundings. For example a dark window on a grey wall. Once the regions have been detected we construct ellipses by replacing each region by an ellipse with the same 2nd moments.

2.2 Choice of Invariant Descriptor

Given an elliptical image region which is co-variant with 2D affine transformations of the image, we wish to compute a description which is *invariant* to such geometric transformations *and* to 1D affine intensity transformations. The choice of descriptors we use is novel, and we now discuss this.

Invariance to affine lighting changes is achieved simply by shifting the signal's mean (taken over the invariant neighbourhood) to zero and then normalizing its power to unity.

The first step in obtaining invariance to image transformation is to affinely transform each neighbourhood by mapping it onto the unit disk. The process

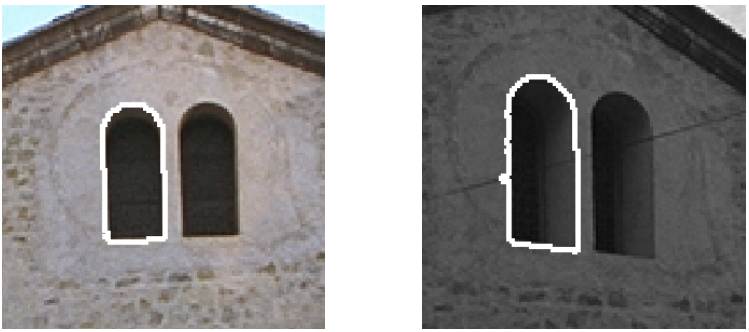

Fig. 5. MSE regions (shown in white) detected in views 3 and 7 of the Valbonne data set.

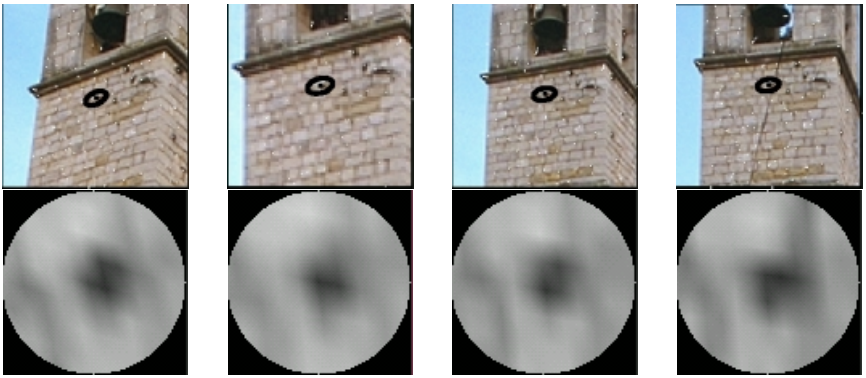

Fig. 6. Illustration of the invariant indexing stage. The query point is taken from the second image (from the left) and the hits found in the index structure are shown in the other images. Below each image is the corresponding affine normalized image patch. Note that the patches are approximately rotated versions of each other. This shows only four of the eight "hits" correctly found for this particular point.

is canonical except for a choice of rotation of the unit disk, so this device has reduced the problem from one of affine invariance to computing rotational invariants of a scalar function (the image intensity) defined on the unit disk. This idea was introduced by Baumberg in [2].

The objective of invariant indexing is to reduce the cost of search by discarding match candidates whose invariants are different. While two very different features can have similar invariants, similar features cannot have very different invariants. Conceptually, the "distance" in invariant space predicts a lower bound on the "distance" in feature space. Our invariant scheme is designed so that Euclidean distance between invariant vectors actually (and not just conceptually) provide a lower bound on the SSD difference between image patches. By contrast Schmid [20] and Baumberg [2] both learn a distance metric in invariant space from training data, which has the disadvantage of tuning the metric to the domain of training data.

We apply a bank of linear filters, similar to derivatives of a Gaussian, and compute rotational invariants from the filter responses. The filters used are derived from the family

$$K_{mn}(x, y) = (x + iy)^m (x - iy)^n G(x, y)$$

where $G(x, y)$ is a Gaussian. Under a rotation by an angle θ, the two complex quantities $z = x + iy$ and $\bar{z} = x - iy$ transform as $z \mapsto e^{i\theta} z$ and $\bar{z} \mapsto e^{-i\theta} \bar{z}$, so the effect on K_{mn} is simply multiplication by $e^{i(m-n)\theta}$. Along the "diagonal" given by $m - n = \text{const}$ the group action is the same and filters from different "diagonals" are orthogonal so if we orthonormalize each "diagonal" separately we arrive at a new filter bank with similar group action properties but which is also orthonormal. This filter bank differs from a bank of Gaussian derivatives by a linear coordinates change in filter response space. The advantage of our formulation is that the group acts separately on each component of the filter response and doesn't "mix" them together, which makes it easier to work with. Note that the group action doesn't affect the magnitude of filter responses but only changes their relative phases. We used all the filters with $m + n \leq 6$ and $m \geq n$ (swapping m nd n just gives complex conjugate filters) which gives a total of 16 complex filter responses per image patch.

Taking the absolute value of each filter response gives 16 invariants. The inequality $||z| - |w|| \leq |z - w|$ guarantees (by Parseval's theorem – the filter bank is orthonormal) that Euclidean distance in invariant space is a lower bound on image SSD difference. Unfortunately, this ignores the relative phase between the components of the signal.

Alternatively, following [13] one could estimate a gradient direction over the image patch and artifically "rotate" each coefficient vector to have the same gradient direction. This would give twice as many (32) invariants, but doesn't work well when the gradient is close to zero.

Instead, we find, among the coefficients for with $p = m - n \neq 0$ the one with the largest absolute value and artificially "rotate" the patch so as to make the phase 0 (i.e. the complex filter response is real and positive). When $p > 1$ there are p ways to do this (p roots of unity) and we just put all the p candidate invariant vectors into the index table. The property of distance in invariant space being a lower bound on image SSD error is also approximately true for this invariant scheme, the source of possible extra error coming from feature localization errors.

Summary: We have constructed, for each invariant neighbourhood, a feature vector which is invariant to affine intensity and image transformations. Morever, the Euclidean distance between feature vectors directly predicts a lower bound on the SSD distance between image patches, obviating the need to learn this connection empirically.

2.3 Invariant Indexing

By comparing the invariant vectors for each point over all views, potential matches may be hypothesized: i.e. a match is hypothesized if the invariant vec-

tors of two points are within a threshold distance. See figure 6 for illustration. These are the "hits" in the indexing structure, and since each must be attended to, the overall complexity depends at the very least on the total number of hits.

Indexing structure: The query that we wish to support is "find all points within distance ε of this given point". We take ε to be one tenth of the image dynamic range (recall this is an image intensity SSD threshold).

For the experiments in this paper we used a binary space partition tree, found to be more time efficient than a k-d tree, despite the extra overhead. The high dimensionality of the invariant space (and it is generally the case that performance increases with dimension) rules out many indexing structures, such as R-trees, whose performances do not scale well with dimension.

2.4 Verification

Since two different patches may have similar invariant vectors, a "hit" match does not mean that the image regions are affine related. For our purposes two points are deemed matched if there exists an affine geometric and photometric transformation which registers the intensities of the elliptical neighbourhood within some tolerance. However, it is too expensive, and unecessary, to search exhaustively over affine transformations in order to verify every match. Instead we compute an approximate estimate of the local affine transformation between the neighbourhoods from the characteristic scale and invariants. If after this approximate registration the intensity at corresponding points in the neighbourhood differ by more than a threshold, or if the implied affine intensity change between the patches is outside a certain range, then the match can be rejected. Table 1 shows the time taken by, and the number of matches involved in, this process.

Table 1. Typical timings for the initial stages on a 2.0GHz Xeon processor. Most of the time is taking up by querying the index table, although it is an empirical observation that the majority of multi-tracks are small. Verification by correlation typically removes 30% of the putatives matches.

	Valbonne	Raglan
Total number of features	39544	402622
Intra-image hashing (ms)	8130	159400
Distintive features	37628	384068
Inter-image hashing (ms)	30760	2513520
Number of matches	14349	717721
Correlation (ms)	6930	313530
Number of matches	9168	332063

The outcome of the indexing and verification stages is a large collection of putative "multi-tracks"; a multi-track is simply the set of matches resulting from

a query to the index table. A multi-track thus gives rise to a number of feature track hypotheses by selecting a subsets with at most one feature taken from each image. The index table matches two features if they merely "look" similar up to an affine transformation, so the possible putative tracks contain many false matches (e.g. one corner of a window in one image will match all the corner of the window in another image). To reduce the confusion, we only consider features which are "distinctive" in the sense that they have at most 5 intra-image matches. Thus, we use within-image matching (a process that is linear in the number of views) to reduces the cost inter-matching.

For the 15-image Valbonne sequence, the result of hashing and verification is summarized in table 7. Each entry counts the number of putative two-view matches implied by the list of multi-tracks. Note, this is only a *picture* of the feature vs view data structure; one could also generate from it an $N \times N \times N$ table of 3-view match counts, or 4-view match counts etc.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	106	103	98	99	86	74	27	84	65	90	67	54	51	83
1	0	0	174	105	104	67	76	34	77	74	80	95	69	63	67
2	0	0	0	169	103	108	86	34	101	77	74	94	55	59	61
3	0	0	0	0	146	164	114	28	84	105	58	92	52	35	61
4	0	0	0	0	0	135	111	42	113	111	74	101	65	62	83
5	0	0	0	0	0	0	161	37	86	107	58	95	52	33	56
6	0	0	0	0	0	0	0	62	135	155	57	96	66	52	63
7	0	0	0	0	0	0	0	0	67	46	32	27	33	25	43
8	0	0	0	0	0	0	0	0	0	141	104	117	68	81	74
9	0	0	0	0	0	0	0	0	0	0	77	141	72	54	64
10	0	0	0	0	0	0	0	0	0	0	0	258	204	126	77
11	0	0	0	0	0	0	0	0	0	0	0	0	217	123	83
12	0	0	0	0	0	0	0	0	0	0	0	0	0	148	81
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	185
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	89	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	350	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	460	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	381	424	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	635	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	566	0	0	0	0	0
7	0	0	0	0	0	0	0	0	85	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	321	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	540	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	214	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	613	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	180	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	434
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 7. Left: Number of initial two-view connections found between the view of the Valbonne image set. Right: Number of new two-view connections found (see section 3.3).

The problem now is that there are still many false matches. These can be resolved by robustly fitting multi-view constraints to the putative correspondences but to do so naively is prohibitively expensive since the number, say, of fundamental matrices is quadratic in the number of views. Instead, we will use the table to guide a heuristic for singling out pairs of views which it will be worth spending computational effort on.

For each pair of views that we do single out, we want out algorithm to have the best chance of working. This is the subject of the next section.

3 Improving the Multiview Matches

Given the putative multiple matches of section 2 our objective in this section is to build up a sufficient number of *correct* point matches between the views in order to support camera computation over a significant sub-set of the views. The computation of cameras from these matches is then the subject of section 4.

Our task here then is to "clean-up" the multiple view matches: remove erroneous and ambigous matches and add in new correct matches.

The matching constraint tools at our disposal range from semi-local to global across the image. Semi-local constraints are on how sets of neighbouring points

transform (for example a similar photometric or geometric transformation), and global are the multi-view relations which apply to point matches globally across the image (such as epipolar and trifocal geometry). These constraints can be applied at various points and order in a matching strategy, and can be used both to generate new matches and to verify or refute existing matches.

The strategy we employ here is to improve the matches between selected view *pairs*. There are three steps (1) select view pair, (2) grow additional matches, (3) compute the epipolar geometry to reject incorrect matches. These three steps are described below.

However, it is important to note that in improving matches between a particular view pair has consequences for the matches between other pairs of images (by tracking a feature from view to view) in the view set. This is a key point because it means that a linear number of operations results in improvements that naively would require quadratic, or cubic etc time.

3.1 Growing Matches

Given a verified match between two views, the estimated affine transformation between the patches can be refined using direct photometric registration (similar to the Lucas-Kanade [9] algorithm) with correction for affine intensity changes. This verification is expensive as it involves a six-parameter (four geometric and two photometric) numerical optimization which is carried out using the Levenberg-Marquardt algorithm. Even a special-purpose implementation of this is quite slow and it is unwise to apply it indiscriminantly.

Once computed, the fitted local intensity registration provides information about the local orientation of the scene near the match; for example, if the camera is rotated about its optical axis, this will be reflected directly by cyclo-rotation in the local affine transformation. The local affine transformation can thus be used to guide the search for further matches. This idea of growing matches [16] enables matches which have been missed as hits, perhaps due to feature localization errors, to be recovered and is crucial in increasing the number of correspondences found to a sufficient level.

Growing is the opposite of the approach taken by several previous researchers [20,25], where the aim was to measure the consistency of matches of neighbouring points as a means of verifying or refuting a particular match. In our case we have a verified match and use this as a "seed" for growing. The objective is to obtain other verified matches in the neighbourhood, and then use these to grow still further matches etc.

We use the estimated affine transformation provided by the initial verified match to guide the search for further matches (see figure 8) which are then verified in the standard manner. Figure 9 demonstrates that many new matches can be generated from a single seed.

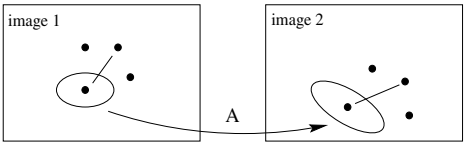

Fig. 8. Illustration of the use of local registration to guide the search for further matches.

Fig. 9. A seed match (left) and the 25 new matches grown from it (right).

3.2 Robust Global Verification

Having grown matches the next step is to use fundamental matrix estimation between pairs of views with a sufficient number of matches. This is a global method to reject outlying two-view matches between each pair of views. A novelty here is to use the affine transformations between the patches in the computation, by imposing the constraint that points transferred under the affine transformation agree with the fundamental matrix in an infinitesimal neighbourhood. The consequence is that only 3 points are needed for each RANSAC sample, as opposed to 7 if only the position of the point is used [6].

3.3 Greedy Algorithm

The next objective is to efficiently select pairs of views to process using two-view matching constraints in order to "clean up" the putative matches in the feature vs view table. Our approach will be to construct a spanning tree on the set of images in the data set. How do we select the edges of the tree? Starting from the pair of images with the most putative two-view matches, we robustly impose the epipolar constraint and then join up those images in the graph. Then we do the same for the pair of images with the highest number of two-view matches, subject to the constraint that joining those images will not create a cycle in the graph. If there are N images, the spanning tree will have $N - 1$ edges so this process is linear in the number of views.

Figure 10 shows the spanning tree obtained in this way and table 7 shows the difference between the number of two-view matches before and after this process has been applied to the Valbonne data set. It can be seen that in all cases considered, the number of matches increased, sometimes significantly. Once the spanning tree has been constructed, we delete any edges corresponding to fewer than 100 matches. In the Valbonne example, this has no effect but in

Fig. 10. Two-view connections found between the views of the Valbonne image set.

the Raglan data set it causes the graph to break into 11 connected components corresponding to distinct scenes, or views of the same scene that were difficult to match. In the next stage, each such component is treated separately.

In summary, we have described a method for singling out particular views for processing which allows us to split the data set into subsets that are likely to be related. The complexity of all stages is linear in the number of views. This process is of course sub-optimal compared to enforcing epipolar constraints between all pairs of images but on the data sets tried, it gives almost comparable performance.

4 From Matches to Cameras

The objective now is to compute cameras and scene structure for each of the components from the previous section separately. Our sub-goal is to find many long tracks (i.e. correspondences across many views) as this is likely to correlate with large a baseline, and a large baseline increases the chance of a successful reconstruction. To achieve this we use a greedy algorithm.

Since we wish to use the reconstruction as a means of spatially organizing the views we require a metric (or at least quasi-affine [5]) reconstruction, and we use the standard approach of first computing a projective reconstruction, followed by auto-calibration [6] using the constraint that the camera has square pixels.

Estimating cluster geometry: In general the cluster of views we are faced with will not have many complete tracks across them, which makes it necessary to compute structure for a sub-set of views first and then enlarge to more views using common scene points. The question is how to select the initial sub-set of views.

Our approach is to order the views in each component, again greedily, by starting with the pair with the most matches and sequentially adding in the next view with the largest number of matches, subject to it being adjacent to a view already included. Now we have an ordering on our image set.

We next look at the initial subsequences of length two, three, four, ... in each ordered image set and compute the number of tracks that can be made across the whole subsequence. We take the longest subsequence with at least 25 complete tracks and then use the 6-point algorithm from [19] to robustly compute projective structure for the subsequence, bundle and then sequentially re-section the remaining cameras into the reconstruction, bundling [22] as we go. The process of selecting a subsequence is linear in the number of views but, of course, structure estimation with repeated bundle adjustment is not.

For the Valbonne sequence, the ordering of the views and the number of complete tracks is shown in table 2.

Table 2. As the number k of views used increases, the number of complete tracks (those seen in all views) decreases. The number of reconstructed points tends to increase initially, then plateau and eventually decrease. A compromise must be reached.

k	views	tracks	points
2	11,12	377	377
3	9,11,12	94	423
4	6,9,11,12	52	666
5	5,6,9,11,12	33	869
6	3,5,6,9,11,12	18	1072
7	2,3,5,6,9,11,12	11	1208
8	2,3,4,5,6,9,11,12	8	1200
9	1,2,3,4,5,6,9,11,12	6	1319

For the resectioning it is important to have Euclidean reconstruction because when image overlap is small the scene is often nearly flat so one needs to make calibration assumptions in order to be able to re-section the cameras. We assume square pixels.

Overall we have moved from a sparse "backbone" of good two-view matches (end of section 3) to a "weaving" together of the views [7,17] via the computed structure. Instead of sparse connections between pairs of views we now have a global view of our data set, facilitated by being able to quickly look up relationships in the feature vs view table.

4.1 Results

In the case of the Valbonne image set the views have sufficient overlap to form a single cluster, but the Raglan image set splits into three clusters of size eight, five and two.

Valbonne: This is a 15 image set, and cameras are computed in a single reconstruction over all the views, shown in figure 11.

Raglan: This is a set of 46 images which breaks into several clusters, the largest consisting of 8 views. Some such clusters are shown if figures 12 and 13

5 Algorithm Summary

The strategy involves hypothesizing matches (the hash table) with hypotheses being refuted, verified or grown based on progressively more global image neighbourhood and multiple view constraints.

1. Detect two types of feature independently in each image, and compute their invariant descriptors.
2. Use hashing (followed by correlation, but no registration) to find initial putative matches and make a table counting two-view matches.
3. Greedy spanning tree growing stage:
   a) Choose the pair i, j of images with the largest number of matches, subject to i, j not already being in the same component.
   b) Apply full two-view matching to images i and j, that is:
      i. Increase correlation neighbourhood sizes if this improves the score.
      ii. Intensity based affine registration.
      iii. Growing using affine registrations.
      iv. Robustly fit epipolar geometry.
   c) Join images i and j in the graph.
   d) Repeat till only one component is left.
4. Form connected components of views as follows:
   a) Erase from the spanning tree all edges corresponding to fewer than 100 matches.
   b) Greedily grow connected components as before; this induces an ordering on the images in each component.
   c) From each ordered component, choose the largest initial subsequence of images with at least 25 complete tracks.
   d) Compute (Euclidean) structure for that subsequence.
   e) Re-section the remaining views into the reconstruction in order, bundling the structure and cameras at each stage.

5.1 Discussion, Conclusion, and Future Work

We have achieved our aim of constructing an $O(N)$ method of organizing an unordered image set. The algorithm is greedy but performs reasonably well. There are still plenty of issues to be addressed, though:

Three-view constraints: Applying two-view matching constraints exhaustively is expensive because there are $O(N^2)$ pairs of images and each computation is independent of the other. For three-view matching constraints there is a qualitative difference, which is that while there are $O(N^3)$ triples of images, the outcome of outlier rejection for one triplet *can* affect the outcome for another triplet (if the triplets share two views). Essentially, the RANSAC robust fitting is able to terminate more quickly as the number of mismatches is reduced. Empirically, it has been observed that the robust fits speed up towards the end of the set of triplets.

This is an example of how structure constraints reduce complexity of matching; quantifying when estimated structure is reliable enough to be used for guiding the matching is an important problem.

Once a spanning tree has been formed (e.g. by our greedy method) it might be beneficial to run a multi-view structure constraint filter "along" the tree to clear up bad matches. For example, at each vertex of the tree one could apply a filter to the vertex and its neighbours. The complexity of this would still be linear in the number of views.

Fig. 11. VRML model of the scene computed for the Valbonne image set. Note that the distribution of images in the computed scene reflects the impression of proximity, adjacency etc that one gets by looking at the original images. There number of 3D scene points is 1034.

Cleaning up the matches: The initial match candidates produced by the indexing structure are based on appearance alone so when trying to form tracks across a few images one is easy prey for mismatched features. The larger the set of images, the greater the risk of false matches. Applying two- or three-view matching constraints can help illuminate the global picture, though. For example, if we believe in the results of two-view matching between image A and B and between image B and C then we can infer matches between image A and C and also reject matches between these images that were previously putative. The practical difficulty with this sort of approach is knowing when estimated two- or three-view can be trusted for inference with and when it can't.

Image based viewer: Although our aim is to compute the cameras, and results are given for this below, if our only requirement is an approximate spatial organization (in terms of relations such as "to-the-left-of", or "a-close-up-of"),

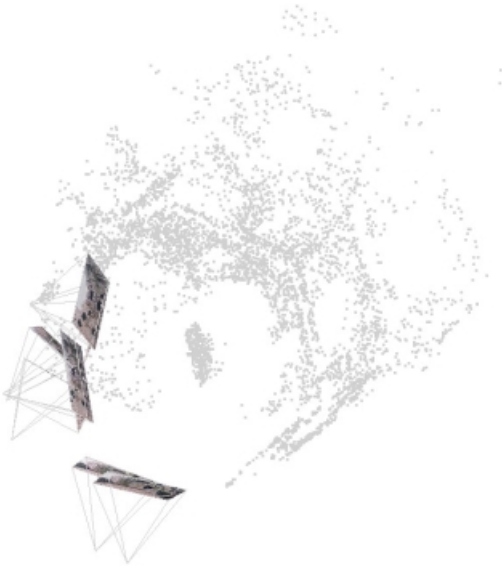

Fig. 12. The largest cluster (8 views), for the Raglan image set, showing the interior of the hexagonal keep.

Fig. 13. Two smaller clustesr (5 views and 2 views) from the Raglan image set, the left one showing the exterior of the keep and the right one showing the top of the keep. Ideally the latter should have been connected to the 8-view cluster of views.

then simply computing homographies between the views and organizing the results with a spherical topology, will suffice. This is a sufficient basis for an image based navigator.

Variations and extensions: Our framework could be specialized (in transformation) by reducing the degree of invariance of the interest point neighbourhood descriptors (e.g. from affine to similarity only) or generalized (in scene type) by

including other types of invariant descriptor, e.g. those developed by [12,21,23, 24].

Other heuristics: Section 2 outlined one strategy for obtaining sufficient multiview matches, and are now looking at other methods to quickly find subsets of hits that are likely to yield good structure estimates which can then be brought in earlier to guide matching, such as the voting techniques used by Schmid [20].

To conclude, we have achieved our objective of spatially organizing an unordered image set, and have set up a framework which is applicable to large data sets. We are currently investigating if the scheme can be extended as far as, say, video shot-matching where there can be thousands of shots in a single movie.

Acknowledgements. This paper is dedicated to Richard Szeliski. We are very grateful to Jiri Matas for supplying the MSE region code and to Andrew Fitzgibbon for his assistance with the structure-from-motion engine used for the experiments. The Valbonne images were provided by INRIA Sophia, and the Raglan images by Julian Morris. Funding was provided by Balliol College, Oxford, and EC project Vibes.

References

1. S. Avidan and A. Shashua. Threading fundamental matrices. In *Proc. ECCV*, pages 124–140. Springer-Verlag, 1998.
2. A. Baumberg. Reliable feature matching across widely separated views. In *Proc. CVPR*, pages 774–781, 2000.
3. P. Beardsley, P. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In *Proc. ECCV*, LNCS 1064/1065, pages 683–695. Springer-Verlag, 1996.
4. A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *Proc. ECCV*, pages 311–326. Springer-Verlag, Jun 1998.
5. R. Hartley, L. de Agapito, E. Hayman, and I. Reid. Camera calibration and the search for infinity. In *Proc. ICCV*, pages 510–517, September 1999.
6. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521623049, 2000.
7. R. Koch, M. Pollefeys, B. Heigl, L. Van Gool, and H. Niemann. Calibration of hand-held camera sequences for plenoptic modeling. In *Proc. ICCV*, pages 585–591, 1999.
8. T. Lindeberg and J. Gårding. Shape-adapted smoothing in estimation of 3-d depth cues from affine distortions of local 2-d brightness structure. In *Proc. ECCV*, pages 389–400, May 1994.
9. B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.
10. J. Matas, J. Burianek, and J. Kittler. Object recognition using the invariant pixel-set signature. In *Proc. BMVC.*, pages 606–615, 2000.

11. J. Matas, O. Chum, and T. Urban, M. an Pajdla. Distinguished regions for wide-baseline stereo. Research Report CTU–CMP–2001–33, Center for Machine Perception, K333 FEE Czech Technical University, Prague, Czech Republic, November 2001.
12. J Matas, M Urban, and T Pajdla. Unifying view for wide-baseline stereo. In B Likar, editor, *Proc. Computer Vision Winter Workshop*, pages 214–222, Ljubljana, Sloveni, February 2001. Slovenian Pattern Recorgnition Society.
13. K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proc. ICCV*, 2001.
14. K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. ECCV*. Springer-Verlag, 2002.
15. P. Pritchett and A. Zisserman. Matching and reconstruction from widely separated views. In R. Koch and L. Van Gool, editors, *3D Structure from Multiple Images of Large-Scale Environments, LNCS 1506*, pages 78–92. Springer-Verlag, Jun 1998.
16. P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *Proc. ICCV*, pages 754–760, Jan 1998.
17. H. S. Sawhney, S. Hsu, and R. Kumar. Robust video mosaicing through topology inference and local to global alignment. In *Proc. ECCV*, pages 103–119. Springer-Verlag, 1998.
18. F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *Proc. ICCV*, Jul 2001.
19. F. Schaffalitzky, A. Zisserman, Hartley, R. I., and P. H. S. Torr. A six point solution for structure and motion. In *Proc. ECCV*, pages 632–648. Springer-Verlag, Jun 2000.
20. C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE PAMI*, 19(5):530–534, May 1997.
21. D. Tell and S. Carlsson. Wide baseline point matching using affine invariants computed from intensity profiles. In *Proc. ECCV*. Springer-Verlag, Jun 2000.
22. W. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment: A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS. Springer Verlag, 2000.
23. T. Tuytelaars and L. Van Gool. Content-based image retrieval based on local affinely invariant regions. In *Int. Conf. on Visual Information Systems*, pages 493–500, 1999.
24. T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *Proc. BMVC.*, pages 412–425, 2000.
25. Z. Zhang, R. Deriche, O. D. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78:87–119, 1995.