Combining Appearance and Topology for Wide Baseline Matching

Dennis Tell and *Stefan Carlsson*

Computational Vision and Active Perception Laboratory (CVAP)
Dept. of Numerical Analysis and Computer Science
KTH, SE-100 44 Stockholm, Sweden
Email: {dennis, stefanc}@nada.kth.se

Abstract. The problem of establishing image-to-image correspondences is fundamental in computer vision. Recently, several wide baseline matching algorithms capable of handling large changes of viewpoint have appeared. By computing feature values from image data, these algorithms mainly use appearance as a cue for matching. Topological information, i.e. spatial relations between features, has also been used, but not nearly to the same extent as appearance. In this paper, we incorporate topological constraints into an existing matching algorithm [1] which matches image intensity profiles between interest points. We show that the algorithm can be improved by exploiting the constraint that the intensity profiles around each interest point should be cyclically ordered. String matching techniques allows for an efficient implementation of the ordering constraint. Experiments with real data indicate that the modified algorithm indeed gives superior results to the original one. The method of enforcing the spatial constraints is not limited to the presented case, but can be used on any algorithm where interest point correspondences are sought.

1 Introduction

Recently, wide baseline image matching has received increased attention in computer vision. Unlike classical correlation based matching techniques, wide baseline matching algorithms can tolerate a large change in viewpoint between the images. The terms "wide baseline" and "large change" are quite vague, and available matching algorithms differ somewhat in the definition of these, but generally speaking, wide baseline algorithms are capable of handling more complicated transformations than image translations.

In applications such as 3D reconstruction or object recognition, better matching algorithms could potentially provide good results using only a few images to represent a scene or an object, as opposed to the 3D reconstruction algorithms of today [2] [3] which use image sequences, where the inter-frame difference between images is quite small. However, in cases where video sequences are not available, or difficult to record due to practical reasons, an algorithm capable of matching a

A. Heyden et al. (Eds.): ECCV 2002, LNCS 2350, pp. 68–81, 2002.

few, quite disparate, images is needed. Another area where wide baseline matching could boost performance is content-based image retrieval. Fewer images to represent objects means smaller databases and shorter search times.

Existing matching algorithms can mainly tolerate two classes of transformations, similarity and affine. The work of Schmid and Mohr [4] as well as that of Gouet et al [5] and Baumberg [6], both using differential invariants, belong to the former class. These types of algorithms are adequate for image plane translations and rotations, but for general camera movement the image transformations will be more complex. Hence, methods capable of handling more general transformations have been developed. Torr and Davidson [7] used a coarse to fine approach in conjunction with cross correlation to handle larger baselines. Tuytelaars et al [8] developed an algorithm in which small image regions were found in an affine invariant way. Features were then computed from those regions. The first step of this method is quite complicated and there is no way to guarantee that the same regions are found in both images. The authors solve this problem by selecting several overlapping regions of different sizes and shapes.

All these algorithms use interest points, most often Harris corners, and the textured neighborhoods to discriminate among possible corner correspondences. However, the spatial relationships between corners is also a source of valuable information for matching. This was recognised by Zhang et al. [9] in a short baseline setting where matches were disambiguated using a relaxation technique where local consistency of matches was enforced. Schmid and Mohr [4] used what they called semi-local constraints to aid matching. Given a possible point correspondence, these constraints required that neighboring points be good correspondences as well. As Schmid and Mohrs algorithm dealt with similarity transformations, they also required that angles in the triangles formed by triplets of points be consistent across the images. Since we are dealing with affine or even projective transformations here, it is not necessarily true that angles are preserved in any way.

A more general, although heuristic apprach was taken by Belongie et al. in [10]. In their method, images were sampled sparsly along edges and for each sample point, a histogram was formed, in which each bin represented the density of sample points of the rest of the shape in a particular direction and at a particular distance from the point. Essentially, this histogram captured the appearance of the image as "seen" from every sampled point. By aligning the histogram with the direction of the image gradient, rotational invariance was achieved. Although the histogram was quite coarse, this method does not seem likely to be succesful if significant affine or projective transformations relate the images. Also, the method of Belongie et al. does not use texture in any way, which has proved to be a very good cue for matching considering the numerous references on texture based matching in the computer vision literature.

The property that each point of interest has an attached descriptor, which encodes a large portion of the image "as seen" from that point, is appealing, but a less heuristic method of enforcing topological constraints would be desirable.

So what we want is a method combining the idea of spatial relations between points of interest and local and global texture information.

2 Profile Matching

We will use the work introduced in [1] as a starting point for our algorithm. In [1], Tell and Carlsson introduce a matching method using affine invariants in which the complicated search for invariant image regions needed in [8] is avoided. The method is based on a comparison of the intensity profiles between pairs of Harris corners across the images (see fig. 1).

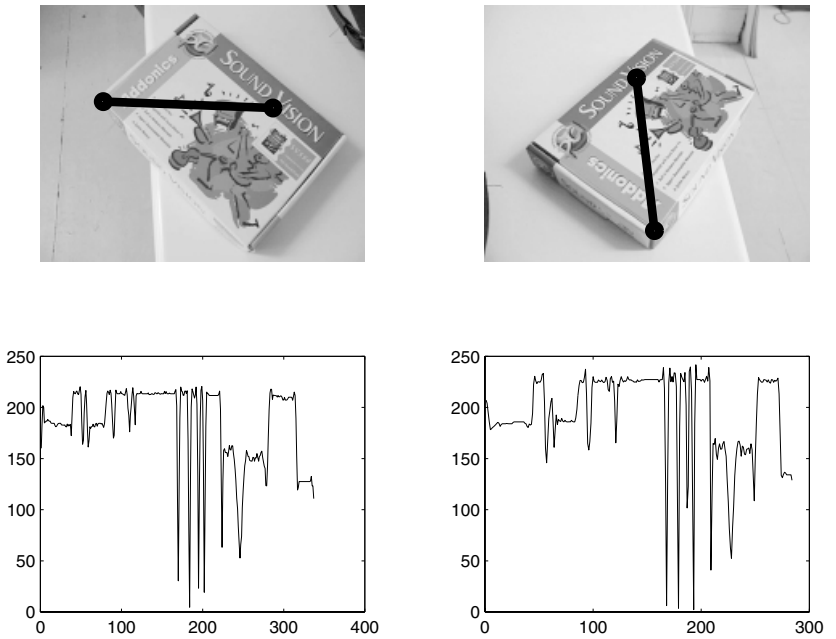
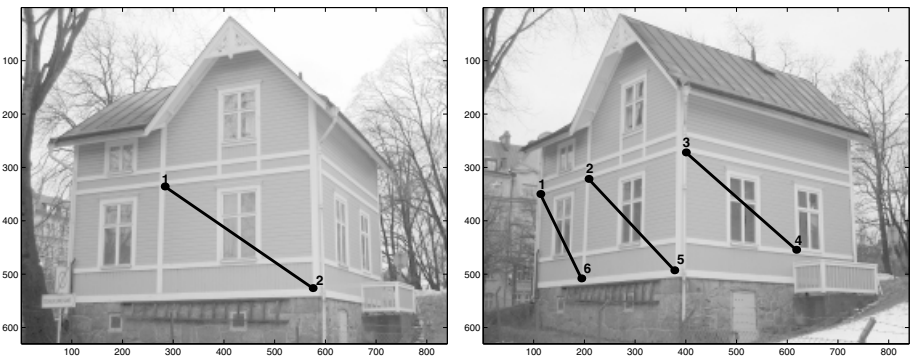


Fig. 1. The same intensity profile extracted from two different images. If the surface is planar along the profile, and if the images are related by an affine transformation, the texture of the profiles can be compared using any scale-invariant method.

The key idea is the observation that if an intensity profile lies along a planar surface, and if the viewing conditions are affine or if the images are related by an affine transformation, then corresponding intensity profiles are related by only one parameter - a scale change. This is because the profiles are related by a 1D affine transformation, which has 3 parameters. These parameters are completely specified by the known endpoints. Any scale-invariant way of comparing the

texture along the profiles can be used for matching. In [1], fourier features are used, since they exhibit good information packing properties and are easy to compute. The distance between fourier feature vectors is then used to compare intensity profiles.

Since the goal of the algorithm is to find corner correspondences, and each corner can form many different intensity profiles with neighboring corners, Tell and Carlsson use a voting strategy. For each intensity profile in the first image, the second image is searched for profiles with similar feature vectors. One can either find all feature vectors within a given distance, or simply select the K closest ones. For each of these profiles, votes are cast for the correspondences of the endpoints of the profiles. The result of this process is a table $\mathbf{V}$ of votes, where the entry $\mathbf{V}(i, j)$ is the number of times corner i was associated with corner j. Fig. 2 shows a simple example where the profile in the left image matches the three profiles shown in the right image, which gives rise to a small voting table.



		Right image corners					
		1	**2**	**3**	**4**	**5**	**6**
Left image corners	**1**	1	1	1	0	0	0
	2	0	0	0	1	1	1

Fig. 2. Illustration of voting. The profile in the left image matches the three profiles in the right image. Hence votes are cast for the correspondences of the endpoints of the profiles. This table also illustrates the idea of voting; using only the displayed profiles, the voting table is highly ambiguous. However, if the voting procedure is performed for all profiles in the images, many ambiguities are resolved.

When the voting process is done, a set of candidate matches can be extracted from the voting table. Several methods are available for this step, for instance the greedy algorithm of iteratively selecting the largest entry and then eliminating that column and row. If one wishes to maximise the sum of the selected entries, graph matching algorithms such as maximum weight matching are applicable.

3 Drawback of the Original Method

To see why this method can be improved upon, consider one single corner in each image. These two corners are characterised by all their outgoing intensity profiles, as shown in fig. 3, where a few of the profiles around corner i in the left image and corner j in the right image are displayed. This collection of profiles serves as a signature which uniquely identifies the corner. This is basically the same idea as in the "shape contexts" of Belongie et al. [10], i.e. that every point of interest has an attached descriptor which represents the rest of the image as seen from that point.

Typically, every corner is allowed to form intensity profiles with its K closest neighboring corners. Formulated this way, it becomes clear that the voting matrix entry $\mathsf{V}(i, j)$ for these two corners, is the number of times a profile from corner i in the left image matches a profile from corner j in the right image.

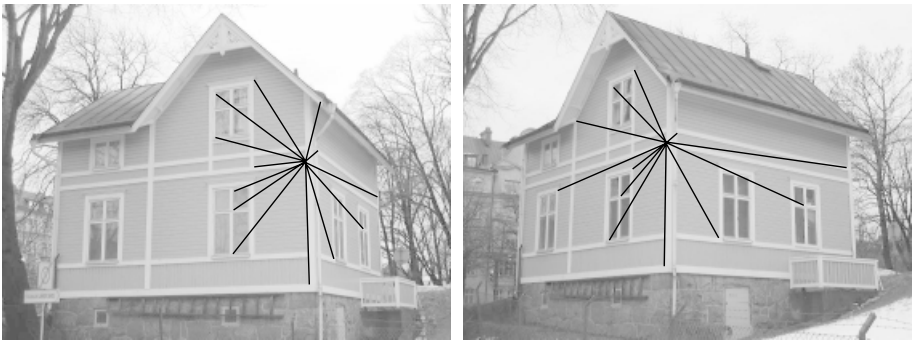


Fig. 3. Each corner is characterised by its outgoing intensity profiles. A few of the profiles for two corners are shown in the figure.

There are two problems with this measure of corner similarity. First, one of the profiles of the left image in fig. 3 may actually match several of the profiles in the right image, thereby voting for the same corner correspondence several times. This introduces a bias in the voting matrix. Second, the fact that intensity profiles should be cyclically ordered around the corners is not taken into account in any way. Again, considering fig. 3, given a subset of corresponding intensity profiles around the two corners, cyclic order should be preserved across the images. This is always true if the intensity profiles lie along planar surfaces, which is one of the basic assumptions of the method anyway.

To put this line of reasoning in a more formal context, first consider fig. 4, where the same profiles as in fig. 3 are shown with labels $\mathbf{p}_m$, $m \in (0, M)$, for the profiles around corner i in the left image, and $\mathbf{q}_n$, $n \in (0, N)$, for profiles around corner j in the right image.

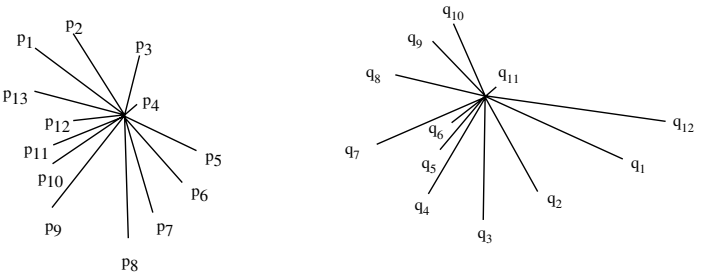


Fig. 4. The profiles extracted around corner i in the left image and corner j in the right image

Then, define a vote function v between two profiles, $\mathbf{p}_m$ and $\mathbf{q}_n$ as $v_{mn} = v(\mathbf{p}_m, \mathbf{q}_n)$. The form of the function v is defined by the voting strategy. For instance, by introducing the feature vectors of profiles as $\mathbf{p}_m^f$ and $\mathbf{q}_n^f$, and the covariance matrix C, the following vote function could be used:

$$v(\mathbf{p}_m, \mathbf{q}_n) = \begin{cases} 1 & \text{if } {\mathbf{p}_m^f}^\top \mathsf{C} \mathbf{q}_n^f < R^2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

given some suitable distance threshold R. This method might give different numbers of matches depending on how "common" a particular profile is, so another possibility without this bias is the following:

$$v(\mathbf{p}_m, \mathbf{q}_n) = \begin{cases} 1 & \text{if } \mathbf{q}_n \in \{\text{the K profiles closest to } \mathbf{p}_m\} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In this definition of the vote function, "closest to" means closest in the sense of Mahalanobis distance between feature vectors, as in eq. 1.

With these definitions, the vote matrix entry $\mathsf{V}(i,j)$ can be written as

$$\mathsf{V}(i,j) = \sum_{m=0}^{M} \sum_{n=0}^{N} v_{mn} \tag{3}$$

This equation makes it quite clear that one profile may vote several times in $\mathsf{V}(i,j)$, and that the angular order of profiles is not taken into account.

4 Suggested Solution - String Matching

It seems reasonable that the algorithm would be improved if cyclic order and one-to-one matching of intensity profiles were enforced somehow. However, since the method is already computationally quite expensive, it would be desirable to

do this without increasing the time complexity too much. We believe that this is indeed possible.

Both the problem of one-to-one matching and that of preservation of cyclic order can be solved in the same way – by applying string matching methods to the algorithm. String matching, or the string to string correction problem, is in its simplest form the problem of finding the longest common subsequence of two strings. For instance, given the string "EABBCD" and "BCED", the longest common subsequence is "BCD". This problem is well known [11] and can be solved by dynamic programming in $O(mn)$ time where m and n are the lengths of the two strings. In the string matching formulation, two letters either match or don't match, but the dynamic programming algorithm can also handle the case of arbitrary similarities of letters. Each possible letter correspondence is simply assigned a weight which expresses the similarity of the letters. In this case, the algorithm still finds a one-to-one sequentially preserved solution, but now one of maximum sum weight instead of one of maximum length.

In our case, each corner is represented by a "string". These "strings" are the collection of outgoing intensity profiles from the corner. The "letters" are the individual profiles, sorted by their angle, and the weights can for instance be the normalized cross correlation between feature vectors.

A complicating property of our problem is that we don't know the starting points of the "strings". One image may well be rotated in relation to the other. The brute-force solution would be to try all possible starting points, resulting in a $O(mn^2)$ algorithm, but better solutions exist for this so-called "cyclic string-to-string correction problem". In [12], an $O(mn\ logn)$ algorithm is proposed, and in [13] an $O(mn)$ algorithm is presented, but the latter one requires unit weights, which means that either letters match, or they don't. In the current implementation, the algorithm in [12] was used, since it allows arbitrary weights.

Using cyclic string matching, eq. 3 is changed to

$$\mathsf{V}(i,j) = \max_{S} \sum_{(m,n)\in S} v_{mn}. \tag{4}$$

The maximum in equation 4 is taken over all sequences S which are of the form $S = \{(m_1, n_1) \cdots (m_p, n_p)\}$, $p \leq \min(M, N)$, such that the following holds:

$$\begin{aligned} m_k &> m_{k-1} \quad 0 \leq m_k \leq M \\ n_k &> n_{k-1} \quad 0 \leq n_k \leq 2N \\ n_i &\neq n_j \bmod N \quad \text{if } i \neq j\ . \end{aligned}$$

The indexes n_k are used as $n_k \bmod N$ when accessing the intensity profiles. This complication is due to the fact that eq. 4 should be rotationally invariant.

This is a much more sensible measure of corner similarity than eq. 3. For instance, given the two strings "CDABBAA" and "ABCD", the original algorithm would find that the vote value is 7, whereas the one using cyclic string matching would find the correct value, which is 4. If ordinary string matching, and not cyclic string matching, was used, the result would be 2, which is not correct either, since rotational invariance is desired.

The next problem is how to apply the dynamic programming algorithm for string matching to the original matching algorithm. This is the topic of the next section. But before going into details about how the string matching methods were applied, an illustrative example might help to appreciate the results. Consider fig. 5, which shows the same row of a voting matrix both using the new approach and not using it. String matching clearly decreases the noise level in the vote matrix, and the correct match is more easily identified.

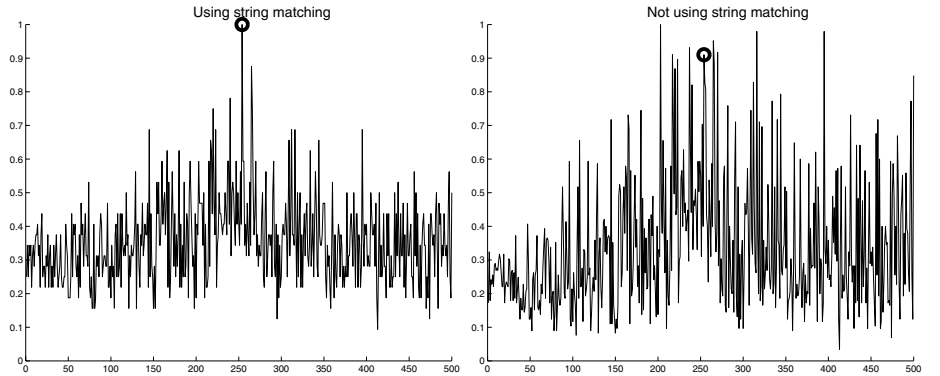


Fig. 5. Illustration of how the improved algorithm decreases noise in the votematrix. The graphs show an entire row of a voting matrix. The left image shows the results using the new approach, the right image shows the results using the original algorithm. The circles indicate the correctly matching column for the given row.

5 Applying the String Matching Algorithm

We see two possibilities of applying the string matching method. The first is the brute-force approach. Such an algorithm would consist of two steps. First, intensity profiles are extracted and features computed for each corner in order to find the "signature" (see fig. 4) of the corner. Second, the cyclic string matching problem is solved for every possible corner correspondence. Suppose that N corners are detected in each image, and that every corner will participate in K intensity profiles. Since there are N^2 possible corner correspondences, and each such candidate correspondence is evaluated using string matching with signatures of length K, the computational complexity of such an algorithm would be $O(N^2K^2 \log K)$. If we allow a complete set of profiles to be formed, i.e. every corner may form profiles with every other corner, the time complexity becomes $O(N^4 \log N)$. Typical running times in our C++ implementation of the brute force algorithm with $N = 500$ is in the order of 10 hours on a Sun Blade 100 workstation.

Clearly, this is not practical with todays computers, so as an alternative, one might apply the string matching methods after the original algorithm. To do this, one must keep track of which intensity profiles that contributed to the voting matrix entry $\mathsf{V}(i,j)$ (eq. 3). To do this, a list is kept for each entry in the voting matrix. Each time a vote is put in an entry, the list is appended with the identities of the two intensity profiles which gave rise to the vote.

Then for each voting matrix entry, the cyclic string matching problem is solved only for the contributing intensity profiles. With this approach, there are still N^2 string matching problems to solve, but each such problem is smaller than in the brute-force algorithm. If the voting process is such that each intensity profile may only put Q votes in the voting matrix, the time needed to solve each string matching problem is on average $Q^2 \log Q$, as opposed the the brute-force approach where each such problem took $K^2 \log K$ time. In our experiments, typical values of N, K and Q was $N = 500$, $K = 200$ and $Q = 40$, which reduced the running time from 10 hours to a few minutes. The trade-off is that more memory is used to keep the lists of contributing intensity profiles for each vote matrix entry.

6 Results

We have conducted a series of experiments designed to compare the performance of our extended algorithm using string matching to the original one. The first experiment compared the two algorithms by matching images of a piecwwise planar object using increasing differences in viewing angle. Both algorithms were run to find a candidate set of point correspondences, after which RANSAC was run to find a homography and a set of consistent matches. Fig. 6 shows the results and the number of consistent matches for the original algorithm, and fig. 7 shows the results using our string matching approach. Clearly, our method gives far more matches, and in the last image pair, where there is a very large difference in viewing angle, the original algorithm breaks down while the new approach still finds a reasonable solution.

The second experiment was made using a cylindrical object. Fig. 8(a) shows the results of the original algorithm and fig. 8(b) those of the new method. Here, the baseline can't be as wide as in the previous experiment for any of the algorithms, simply because the same part of the object can't be seen in both images if the difference in viewing angle is too large. Still, the method using string matching to enforce one-to-one matching and preservation of cyclic order of intensity profiles gives better results. It finds a solution in all three image pairs, while the original method only finds one in the first image pair.

The last experiment (fig. 6) shows a result on estimating a fundamental matrix instead of a homography. Again, the improved algorithm finds about 50% more matches. These may still contain outliers, since the epipolar constraint is quite weak, but that problem is present in both methods.

7 Conclusion

We have presented a general method of enforcing topological constraints such as one-to-one matching and cyclic order in matching algorithms. The method is based on applying cyclic string matching for every possible corner correspondence. Experiments have shown that the new method improved an existing wide baseline algorithm significantly, both by providing larger sets of correspondences, and by finding solutions in cases where the original algorithm failed completely.

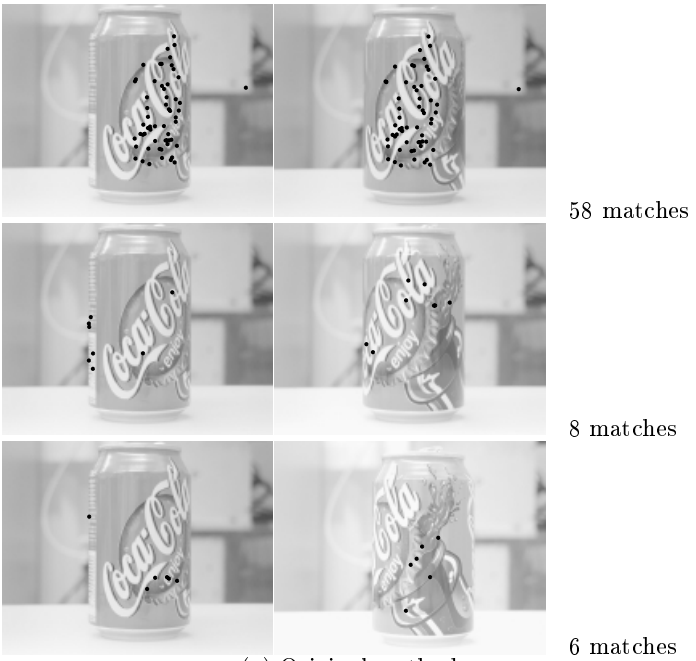
However, the method is not limited to this matching algorithm. It can in principle be applied to other correspondence algorithms where interest points are used in the matching process, particularly those using the more traditional approach of computing features from small neighborhoods around corners [4] [5] [6]. Investigating this could be the topic of future work.

References

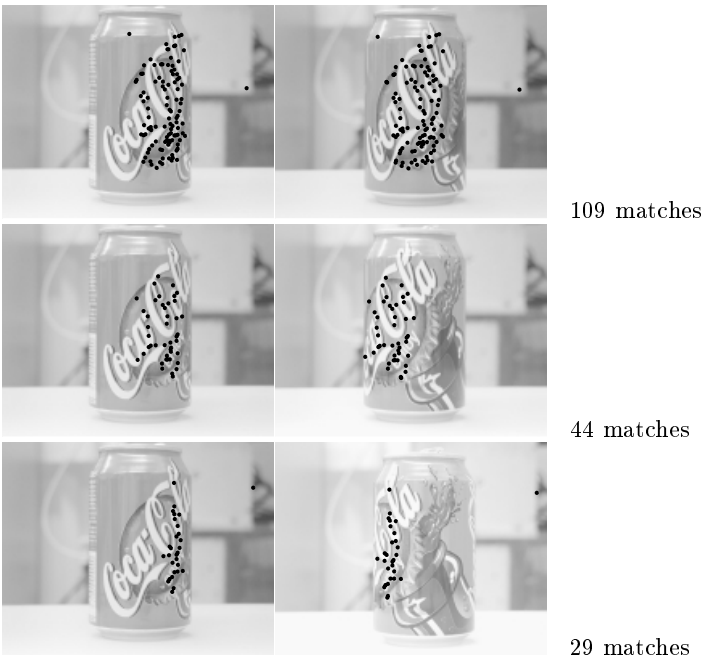
1. D. Tell and S. Carlsson. Wide baseline point matching using affine invariants computed from intensity profiles. In *ECCV*, volume 1, pages 814–828, 2000.
2. A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *Proc. European Conference on Computer Vision*, pages 311–326. Springer-Verlag, June 1998.
3. D. Nistér. *Automatic dense reconstruction from uncalibrated video sequences.* KTH, 2001.
4. C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *PAMI*, 19(5):872–877, 1997.
5. P. Montesinos, V. Gouet, and R. Deriche. Differential invariants for color images. In *ICPR*, 1998.
6. A. Baumberg. Reliable feature matching across widely separated views. In *CVPR*, pages 774–781, 2000.
7. P.H.S. Torr and C. Davidson. Impsac: Synthesis of importance sampling and random sample consensus. In *ECCV*, pages 819–833, 2000.
8. T. Tuytelaars and L. van Gool et. al. Matching of affinely invariant regions for visual servoing. In *ICRA*, pages 1601–1606, 1999.
9. Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence Journal*, 78:87–119, oct 1995.
10. S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *Proc. ICCV01*, pages 454–461. IEEE, July 2001.
11. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms.* MIT Press, 1992.
12. J. Gregor and M.G. Thomason. Dynamic programming alignment of sequences representing cyclic patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15:129–135, 1993.
13. G.M. Landau, E.W. Myers, and J.P.Schmidt. Incremental string comparison. *SIAM Journal of computing*, 27(2):557–582, 1998.

Fig. 6. Using the original matching method. Results on matching using increasing difference in viewing angle

Fig. 7. Using string matching. Results on matching using increasing difference in viewing angle

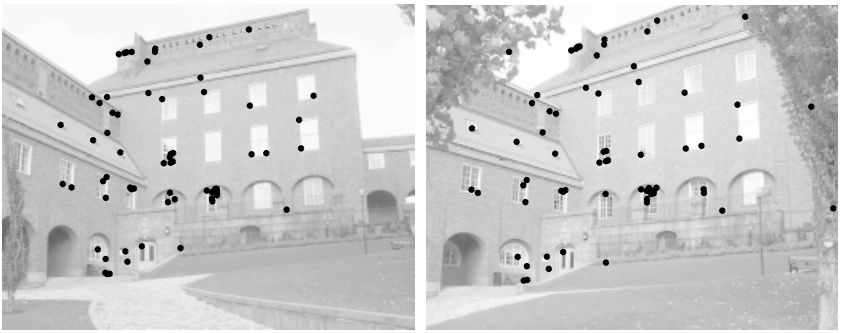


(a) Original method

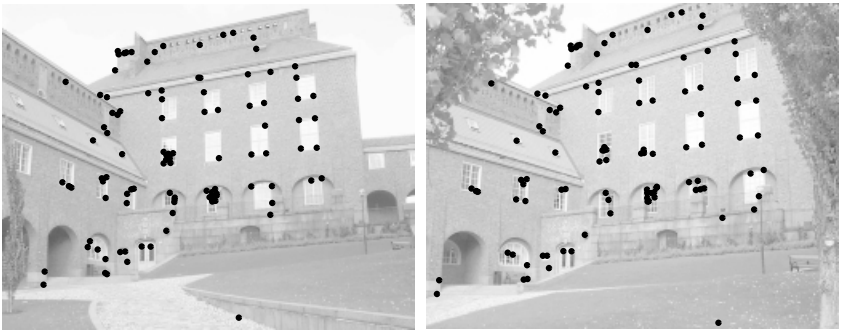


(b) String matching

Fig. 8. Results on matching using increasing difference in viewing angle.



(a) Original method, 66 matches



(b) String matching, 105 matches

Fig. 9. Results on matching using fundamental matrix estimation in RANSAC.