Optimal Point Correspondence Through the Use of Rank Constraints

Ricardo Oliveira * João Costeira † João Xavier

Instituto de Sistemas e Robótica - Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisboa Codex, PORTUGAL
{*rco, jpc, jxavier*}*@isr.ist.utl.pt*

Abstract

We propose a solution to the n-frame correspondence problem under the factorization framework. During the matching process, our algorithm takes explicitly into account the geometrical constraints associated to the reconstruction process. To this end, a rank constraint is imposed on the measurement matrix.

Since our method relies solely on geometric constraints, it is not dependent on corners as image features and can consequently match generic points (e.g. contours). Outlier rejection is integrated as a part of the actual matching process.

In general the problem is formulated as combinatorial, but we develop a method which can provide a solution with a low computational complexity. Because of this, our algorithm is able to handle high-dimensional matching problems that are common in real-life applications.

1. Introduction

Establishing correspondences between the features of a pair of images has proven to be an essential task, deeply related to the recovery of 3D shape from 2D images. However, currently correspondence and reconstruction are viewed as separate problems. For example, successful algorithms like [3], [8], [9] or [10] assume that correspondences have already been established. On the other hand, feature trackers as [5] and stereo algorithms as [4] are not designed with reconstruction in mind. We propose to design a correspondence strategy that optimizes the reconstruction criterion. In the factorization framework this criterion is translated into a rank constraint on the measurement matrix.

Important gains in reconstruction performance can be obtained by performing correspondence in areas with poor texture. Because of this, traditional matching algorithms such as [5] are dependent on specific patterns (e.g. corners) in the image to identify and extract matching candidates. In this context, matching image points in scenes that do not present clear corners is of great difficulty. The use of geometric constraints eliminates difficulties usually associated with the selection of features in other methods. In particular, it allows the use of contour points as features, solving a problem frequently encountered in some classical flow-based methods.

Previous matching algorithms based on geometric constraints have used non-linear cost functions [6]. We formalize our criterion in such a way as to allow a solution based on the execution of a set of linear programs. This guarantees that the algorithm is computationally feasible even for large-scale problems. This aspect is particularly important since it allows the use of the method for real life problems, which are usually high-dimensional. Moreover, we provide a multi-frame formulation for this problem.

Our aim is to present a geometric matching algorithm based on a computationally feasible cost function, that is simulteaneously capable of handling a large number of image points. Moreover, the algorithm should be able to reject a significant number of outliers that usually arise in real-life applications.

2. Problem formulation

In this text we propose a global method that is capable of solving the correspondence problem in the factorization context. Our objective is to successively align the observations on each frame of a video sequence in a matrix W so that corresponding observations occupy the same column. Note that the alignment of each frame is a combinatorial problem. However, we will show that the solution to this problem can be found by imposing rank constraints on W, resulting in an alignment of the features in the current (e.g. most recent) frame with the matched points in the previously aligned images.

*This work was supported by FCT PhD Grant SFRH/BD/6434/2001.

†This work was partially supported by FCT under contract POSI/SRI/34121/1999) in the framework of QCAIII.

The matching candidates (feature points) in the current frame do not have to obey to any particular constraint - in opposition to feature points extracted in generic photometry based algorithms, they do not have to contain significant texture. To emphasize this issue, we use contour points as matching candidates. A feature of this type can potentially be located on a region that does not provide enough photometric information to allow for a correct match.

This paper presents a solution to the matching problem using the orthographic [10] and paraperspective [8] camera. Nevertheless, the procedure described herein is easily extendable to a generic affine camera model.

2.1. Feature representation

Observations on each frame are represented as a set of image coordinates containing the orthogonal projection of 3D feature points in the scene. It should be stressed that in this text the expression *feature point* applies to a generic contour point and should thus not be limited to a corner point. Assuming p_f feature points, we represent the u and v image coordinates of a frame f in the u^f and v^f vectors. We assume that each set of p_f feature points contains a certain number of outliers except for the first, which contains only the points that are to be tracked. The matrix corresponding to the feature positions in frame f is thus represented by

$$w_f = \begin{bmatrix} u_1^f & \cdots & u_{p_f}^f \\ v_1^f & \cdots & v_{p_f}^f \end{bmatrix} \tag{1}$$

Considering $W_1 = w_1$, W_f can be recursively defined as in (2), where W_{f-1} contains all correctly matched features up to scene $f-1$.

$$W_f = \begin{bmatrix} W_{f-1\,[2f-2\times p_0]} \\ w_{f\,[2\times p_f]} P_{f\,[p_f \times p_0]} \end{bmatrix} \tag{2}$$

Note that the outliers in w_f have to be rejected beforehand; moreover, the remaining points have to be aligned so that corresponding features share the same column in W_f. Both the alignment and the outlier rejection are achieved through matrix P_f. Figure 1. shows how the correct P_f allows the alignment of w_f with respect to W_{f-1} to be obtained.

In (2), P_f is a *rowwise partial permutation matrix* that is defined by the conditions in (3). In the remainder of the paper, for the sake of simplicity, we will refer to these matrices simply as partial permutation matrices.

$$\begin{aligned} &P_{f_{ij}} = \{0,1\}, \forall i = 1...p_f, \forall j = 1...p_0 \\ &\sum_i P_{f_{ij}} = 1, \forall j = 1...p_0 \\ &\sum_j P_{f_{ij}} \leq 1, \forall i = 1...p_f \\ &\sum_{i,j} P_{f_{ij}} = p_0 \end{aligned} \tag{3}$$

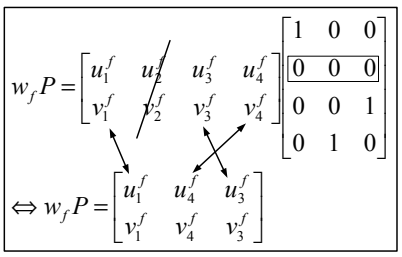

Figure 1. Rowwise Partial Permutation Matrix

In the previous equation p_0 identifies the number of features that will be matched, i. e. not discarded. Note that it is assumed that $p_0 \leq p_f$ - for this reason, P_f is usually a rectangular matrix, since a row is added for each outlier. Furthermore, it is assumed that these p_0 features are visible in every frame throughout the sequence.

2.2. Enforcing rank constraints

It has been shown in [10] and [8] that a measurement matrix similar to the one presented in (2) is highly rank deficient. More specifically, when including translation W_f is at most rank-4 in either model.

Work on factorization algorithms such as the ones referred in the previous paragraph is based on the assumption that a matching solution between the image points has already been found, so that image coordinates corresponding to the same feature point occupy the same column. In the presence of incorrect matches, the resulting W_f is (generally) of higher rank. Note that in the presence of a limited amount of noise the rank-4 constraint may still be assumed as valid, as shown in [6].

Our problem is thus equivalent to *finding a matching solution P_f for w_f that generates a rank-4 W_f*. Each new frame that is added to the stacked measurement matrix should thus be aligned so that the row and column spaces of the resulting matrix remain 4-dimensional (assuming that previous frames have been correctly matched).

The formulation of the matching problem as described above has a drawback: W_f grows with the number of frames, which means that an ever increasing amount of data has to be stored. This problem can be avoided by substituting W_{f-1} by a base for its row space. This alternative has the advantage of making the size of W_f independent of the number of frames (the base for the row space is of constant size). However, experiments have shown that error integration into the base may lead to a decrease in performance in comparison to the explicit use of W_{f-1}.

2.3. Generating the cost function

We consider the SVD decomposition of W_f

$$W_f = Q\Sigma V^T \tag{4}$$

and define Z as

$$Z = W_f W_f^T = Q\Sigma^2 Q^T = \begin{bmatrix} W_{f-1}W_{f-1}^T & W_{f-1}P_f^T w_f^T \\ w_f P_f W_{f-1}^T & w_f P_f P_f^T w_f^T \end{bmatrix} \tag{5}$$

Recall that the aim of our algorithm is to find the matching solution that creates the best rank-4 W_f in the least-squares sense. This can be achieved by minimizing the sum of all eigenvalues λ_i of Z, with the exception of the four largest ones. The eigenvalues of Z can be obtained, by definition, as the result of the following expression, where q_i represents the i^th column of Q, i. e. the i^th eigenvector of Z:

$$\lambda_i = q_i^T Z(P_f) q_i, P_f \in \mathcal{P}^f, \tag{6}$$

where $\mathcal{P}^f$ represents the set of partial permutation matrices of dimension $[p_f \times p_0]$.

It should be noted when minimizing the eigenvalues that the values of each λ_i depend on P_f itself. Our matching problem must thus be formalized as the search for the optimal partial permutation matrix P_f^* such that:

$$P_f^* = \arg\min_{P_f} \left(\sum_{i>4} \lambda_i(P_f) \right) = \arg\min_{P_f} \begin{pmatrix} \\ \\ \sum\limits_{i>4} \max\limits_{\substack{\hat{q}_i^T \hat{q}_i = 1 \\ s.t.\ \hat{q}_i^T \hat{q}_j = 0 \\ i<j}} \hat{q}_i^T Z(P_f) \hat{q}_i \end{pmatrix} \tag{7}$$

To solve the problem of the interdependency between P_f and $q_{nd} = \{q_5, ..., q_{2f}\}$ our algorithm runs iteratively, alternating updates on P_f and on q_{nd} (the eigenvectors are represented with a 'hat' to point out that these are *estimates* of the eigenvectors of the correct Z). At each iteration of the algorithm a new P_f is calculated using the current estimate of q_{nd}. The q_{nd} are then updated themselves by extracting the eigenvectors of Z corrected according to the newly found P_f. The algorithm will continue to iterate alternatively in P_f and q_{nd} until convergence is achieved. The use of this formulation requires that an initial estimate of q_{nd} be provided to start the iterative procedure.

A note should be made on the size of W_f: when processing the second frame (i.e. $f = 2$), W_2 contains only 4 rows. Under normal conditions, this matrix would be rank-4 for *all* P_2. To circumvent this issue, we provide an additional image, so that w_2 is actually matched against two aligned frames. The matching of these two frames can be achieved, for instance, with the algorithm described in [6]. Under these circumstances W_2 has 6 rows, so that only the *correct* P_2 will provide a suitable solution.

3. Recovering a matching solution

3.1. Solving for the permutation matrix

Given an estimate for q_{nd}, the cost function we obtained in the previous section is quadratic in P_f, as can clearly be seen by observing the structure of Z from (5). However, since P_f is a partial permutation matrix, the quadratic term $P_f P_f^T$ can be written as:

$$P_f P_f^T = \begin{bmatrix} \sum\limits_{i=1}^{p_0} {P_f}_{1i}^2 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sum\limits_{i=1}^{p_0} {P_f}_{p_f i}^2 \end{bmatrix} \tag{8}$$

Taking into account that the elements of P_f are either 0 or 1, the k^{th} term of the diagonal of the matrix in (8) can be simplified to $\sum_{i=1}^{p_0} {P_f}_{ki}$. In practice, this leads to a minimization procedure which is linear in P_f.

Given a matrix M, the *vec* operator stacks its columns in order to form a vector. Rearranging P_f as $x = vec(P_f)$, we can write a modified cost function as a linear function of x. Given q_{nd}, x can now be retrieved as the solution to the following linear program:

$$\begin{aligned} & x* = \arg\min_x c.x \\ & s.t. \\ & x = vec(P_f), P_f \in \mathcal{P}^f \end{aligned} \tag{9}$$

The coefficient vector c of the linear program can be determined by developing

$$\sum_{i>4} q_i^T Z(P_f) q_i \tag{10}$$

in order to the elements of P_f, given q_{nd}. Since this expression is linear in P_f it is possible to rewrite it as a dot product of two vectors c and x, where x gathers the elements of P_f as described in above. Under these conditions, c is given by the sum of the c_i in (11), each associated to the i^{th} non-dominant eigenvectors of Z:

$$\begin{aligned} c_i = 2\left[\left(q_1^T W_{f-1}\right) \otimes \left(q_2^T w_f\right)\right] + \\ 1_{[1\times p_0]} \otimes \left[\left(w_f^T q_2\right)^T \bullet \left(w_f^T q_2\right)^T\right] \end{aligned} \tag{11}$$

where q_1 represents the first $2f$ elements of the eigenvector and q_2 the remaining two elements. The details of the calculation of c are given in [7].

The formulation presented in (9) still remains an integer minimization problem and as such has no efficient solution. However, it can be demonstrated that an equivalent concave cost function can be built in the sense that it attains the same values as the original for all possible P_f in $\mathcal{P}^f$. It is also known that the minimum of a linear function over a compact convex set $\mathcal{C}$ is located at an extreme point of $\mathcal{C}$. Consequently, the constraint set of a minimization problem with a linear objective function can be relaxed into its convex-hull, provided that all the points in the original set are extreme points of the new set. In the present case, it can be shown that the convex-hull of the set of partial permutation matrices $\mathcal{P}^f$ is the set of rowwise substochastic matrices $\mathcal{S}^f$. This set can be defined by the second and third equations in (3) and by the condition

$$P_{ij} \geq 0, \forall i = 1...p_f, \forall j = 1...p_0 \tag{12}$$

The resulting problem is thus equivalent to the original, but for this class of problems (linear programming problems) there exist several efficient algorithms that can provide an adequate solution.

This method of solving the integer optimization problem has originally been presented in [6].

3.2. Processing Image Sequences

In this section we show that using the framework described in the previous sections it is possible to process an image sequence automatically by providing the initialization vectors for the first match only.

To bootstrap the matching of a frame in the sequence an estimate of the non-dominant eigenvectors of Z is required. However, since the matching solution to this new frame is as yet unknown, this calculaton cannot be performed directly. Instead, a rough estimate of the position of the feature points in the new frame is calculated by assuming an identical movement in 3D space relative to the previous frame. This estimate can then be used to recover an approximation to the non-dominant eigenvectors of Z.

The key assumption in this process is that the movement between consecutive images is small and smooth, so that q_{nd} is sufficiently close as to provide a good initialization. As can be seen in section 4., this assumption is true for most video sequences that have been built with a reasonable sampling rate.

When determining the correspondence for the first frame, no previous motion information is available and the above mentioned method to determine the eigenvectors can consequently not be used. In this case, it becomes necessary to estimate the initialization vectors for the first frame (which is often possible if some *a priori* knowledge is available) or to provide them externally, using an algorithm such as the one described in [6].

3.3. Correcting Matching Errors

When determining P_f, it is possible that a matching error occurs. This error will not only lower the quality of the solution in the frame where it occurs - since it is propagated into the algorithm it can also induce further errors in subsequent frames.

To prevent this error integration process, we use a correction algorithm that identifies mismatched features by reprojection. From previously matched images, a base for the row space of W_f can be calculated. Correctly matched features in the present frame are then used to calculate a corresponding base for the column space through a least-squares algorithm. The knowledge of corresponding bases for both the row and column spaces allows the recovery of the mismatched features.

This process is similar to that described in [1] for handling occlusions. Note that the algorithm is able to recover a complete set of features even if correctly matched features are marked as mismatched. We use the corrected matrix to generate new initialization vectors, with which we will rerun the matching problem until a stable matching solution is found.

4. Experiments

We tested our algorithm on two sets of data - the synthetic 'Sphere' sequence and the 'Hotel' sequence. The latter (real) sequence was originally obtained from CMU's Image database (*http://vasc.ri.cmu.edu/idb/*) and its images can be considered orthographic for all practical purposes.

In this set of experiments we intend to demonstrate the algorithms ability to successfully track a rigid object in a video sequence. In both experiments, the algorithm is initialized by the user: in the 'Sphere' Sequence theoretical values are used, while in the 'Hotel' sequence initial matches are selected manually for initialization. The matching solution to a frame will typically result in a linear program with tens of thousands of variables, that can take some time to solve. In order to speed up the process, we use *a priori* knowledge, by assuming that there is a limited disparity between consecutive images. This allows us to rule out matching solutions that would result in a very large movement of the features - in practice, this is equivalent to forcing some of the entries of the partial permutation matrix to 0. In this way, an important reduction of dimensionality is achieved. It has been verified that this reduction does not affect the final result, but only the time required to obtain it.

4.1. Sphere sequence

A sequence of 100 images of a translating and rotating wireframe sphere is used. The matching candidates are points on the meridians of the sphere. Two equidistant points are selected on each of the sphere's eight meridians as the points to be tracked - the structure of the sphere can be seen in Figure 2. No noise has been added to this sequence and its main purpose is to validate our method.

Note that with this set of matching candidates only two corner points are present - the poles of the sphere. Due to this fact, an algorithm dependent on corner points would not be able to return a solution suitable to be used for reconstruction, since only two adequate features would be available. Our algorithm, however, can use corner points and contour points alike without any loss in precision.

In each frame, the match for the 16 inliers is found among ca. 1200 candidates. Consequently, each frame would require the solution of a linear program with over 19000 variables. As explained previously, using *a priori* knowledge reduces the dimensionality of the problem.

4.2. Hotel sequence

This sequence is 30 frames long and contains images of a moving toy house. 37 points (features) have been chosen from the contour lines of the first image. As in the 'Sphere' sequence, these points have been chosen in such a way that they reside on contour lines. Since this sequence has been built from real images, a limited amount of noise is associated to each observation. The total number of matching candidates for each frame is ca. 11000 points, resulting in more than 400000 variables. Once more, knowledge on the maximum amplitude of movement between frames is used to radically reduce dimensionality. Due to the absence of ground truth, initialization for this sequence has been provided by manual matching of the feature points so that an initial value for the eigenvectors of Z in the first frame can be obtained.

4.3. Results

The following images in the sequence use the result of the preceding match for initialization, as described in section 3.3. The algorithm is run for the two above mentioned sequences, that present the disparity between consecutive frames as detailed in the next table.

'Sphere'		*'Hotel'*	
min	**max**	**min**	**max**
0.25 pxl	2.17 pxl	0.10 pxl	1.51 pxl

In the synthetic sequence, feature correspondence is done with zero error relative to ground-truth. This is expectable since the experiment corresponds to a perfectly orthographic and noiseless situation.

Matching candidates for the 'Hotel' sequence are represented as black pixels in Figure 3, on the first frame. Feature points have been singled out as black circles. Due to the absence of ground-truth, correspondence error in the 'Hotel' sequence cannot be evaluated directly. However, it is possible to ascertain by visual inspection that feature point trajectories are consistent with the movement of the scene, as seen in Figure 4. Correspondence quality can also be evaluated by recovering the 3D positions of the feature points using [10]. In Figure 5, a top view of the 3D point cloud clearly shows that they are aligned according to the walls of the house. The two 'rogue' points in the middle correspond to the points on the chimney.

Note that in the 'Hotel' sequence there exist a large number of outliers placed very near to the feature points. It is thus plausible that the matching algorithm chooses a neighbouring outlier over a feature point on some occasions, without this implying an error that would compromise the final result.

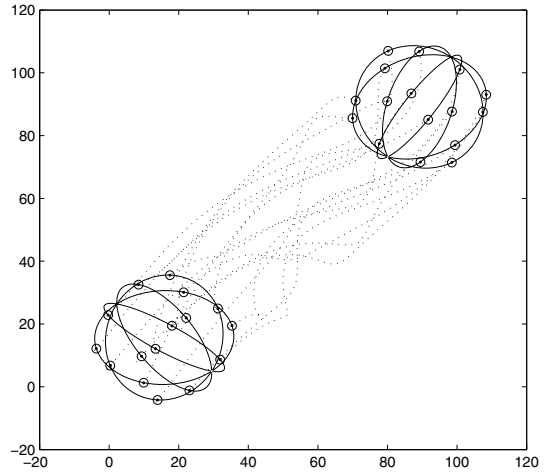

Figure 2. 1^{st}(lower left) and last (upper right) frame of the 'Sphere' Sequence. Dotted lines represent inlier trajectories.

5. Future work and conclusions

We have presented a method that is capable of matching image points extracted from generic contour points, without resorting to corners in the image.

Our algorithm is able to cope with a high percentage of outliers without any significant decrease in performance. We have run experiments with a high number of points, demonstrating that our method is computationally feasible. Reconstruction performance visually demonstrates the

Figure 3. Contours of the 1^{st} frame of the 'Hotel' sequence. Inliers are marked with a circle.

Figure 4. 30^{th} frame of the 'Hotel' sequence and associated inlier trajectories.

capibilities of this algorithm.

The fact that the initial set of features has to be visible over the whole sequence is an issue that becomes problematic if the sequence presents a very large movement. Handling occlusions is at the moment still an open issue. The increase in disparity between consecutive images (which is presently diminute) also requires further investigation.

References

[1] C. Branco and J. Costeira. A 3d image mosaiching system using the factorization method. *In IEEE ISIE*, Pretoria, South Africa, July 1998.

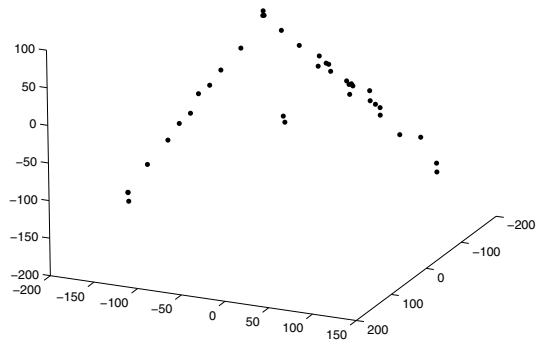

Figure 5. 3D point cloud generated by a SFM algorithm from the computed correspondences (top view).

[2] H. Lütkepohl. *Handbook of Matrices*, John Wiley & Sons 1996.

[3] A. Heyden, R. Berthilsson and G. Sparr. An iterative factorization method for projective structure and motion from image sequences. *Image and Vision Computing*(17), 13(1), pp. 981-991, November 1999.

[4] V. Kolmogorov and R. Zabih. Visual correspondence with occlusions using graph cuts. In *Proc. ICCV*, pp. 508-515, July 2001.

[5] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the* 7^{th} *International Joint Conference on AI*, 1981.

[6] J. Maciel and J. Costeira. A Global Solution to Sparse Correspondence Problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25(2), February 2003.

[7] R. Oliveira, J. Costeira and J. Xavier. Optimal Point Correspondence through the Use of Rank Constraints, *ISR Internal Report*, November 2004.

[8] C. J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In *Proc. ECCV*, pp. 97-108, August 1994.

[9] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proc. ECCV*, pp. 709-720, April 1996.

[10] C. Tomasi and T. Kanade. Shape from motion from image sreams under orthography: a factorization method. *IJCV*,9(2):137-154, November 1992.