Contour Point Tracking by Enforcement of Rigidity Contraints

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

The aperture problem is one of the omnipresent issues in computer vision. Its local character constrains point matching to high textured areas, so that points in gradient - oriented regions (such as straight lines) can not be reliably matched. We propose a new method to overcome this problem by devising a global matching strategy under the factorization framework.

We solve the n-frame correspondence problem under this context by assuming the rigidity of the scene. To this end, a geometric contraint is used that selects the matching solution resulting in a rank-4 observation matrix.

The rank of the observation matrix is a function of the matching solutions associated to each image and as such a simulteaneous solution for all frames has to be found. An optimization procedure is used in this text in order to find the solution.

1. Introduction

Reconstructing a 3D scene from a sequence of 2D images is one of the most important problems in computer vision. However, a successful reconstruction is dependent on a precise matching of the points in the several frames. For instance, successful SFM algorithms like [3], [8] 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 ([3], [8], [9], [10]) 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, as the contour lines in an image. However, the aperture problem prevents the reliable extraction of matching candidates on generic contour poins. 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 a matching algorithm based on geometry is not dependent upon specific brightness patterns in the images and can thus use any contour point as a match candidate. Moreover, we provide a global solution.

Previous matching algorithms based on geometric constraints have used non-linear cost functions [6]. This aspect limits their applicability to problems with a reduced number of points, since calculation time rises rapidly with the dimension of the problem. This is an important issue since practical applications usually require the solution of high dimensional problems. Another aspect is that [6] does not provide an efficient framework to deal with image sequences.

Our aim is to match the images in a video sequence in such a way as to optimize the rigidity in the scene. To achieve this, we present a geometric correspondence 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

This work is an evolution of the approach presented in [7], in which we propose a global method that is capable of solving the *n-frame correspondence problem* in the factorization context. Our objective is to provide a set of matching solutions to features contained in the frames of a video sequence. These features are represented in an observation matrix W so that corresponding observations occupy the same column. We will show that the solution to this prob-

*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.

lem can be found in an efficient way by imposing rank constraints on W. It should be emphasized that the alignment of each frame is by itself a *combinatorial* problem. The multiple frame correspondence problem (i.e. the alignment of *all* images in W) is consequently an extremely complex task.

Subspace constraints on image coordinates have been established explicitly for the paraperspective camera in [8] and the orthographic camera in [10] - however, our formulation holds for any generic affine camera.

A drawback is that our method requires an initialization to bootstrap the matching procedure.

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 confused with 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 is corrupted by a certain number of outliers, except for w_1 which contains only the points that are to be tracked (i. e. inliers). The matrix corresponding to frame f is thus represented by

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

Measurements corresponding to several frames can be vertically stacked in order to create a measurement matrix W_f that incorporates the projection of the feature points up to scene f. However, the outliers in each frame have to be rejected beforehand; moreover, the remaining points have to be aligned so that corresponding features share the same column in W_f. Matrix P_f simultaneously aligns the feature points and rejects the outliers in the corresponding measurement matrix w_f. W_f can consequently be written as

$$W_f = \begin{bmatrix} w_1 & P_2 \\ w_2 & \\ \vdots & \vdots \\ w_f & P_f \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} u_1^1 & \cdots & u_{p_0}^1 \\ v_1^1 & \cdots & v_{p_0}^1 \end{bmatrix} & I_{[p_0 \times p_0]} \\ \begin{bmatrix} u_1^2 & \cdots & u_{p_2}^2 \\ v_1^2 & \cdots & v_{p_2}^2 \end{bmatrix} & P_{2[p_2 \times p_0]} \\ \vdots & \vdots \\ \begin{bmatrix} u_1^f & \cdots & u_{p_f}^f \\ v_1^f & \cdots & v_{p_f}^f \end{bmatrix} & P_f \left[p_f \times p_0 \right] \end{bmatrix} \tag{2}$$

In (2), each P_k 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*.

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

In each of the frames, the optimal P_k allows a correct alignment of w_k to be obtained, as depicted in Figure 1.

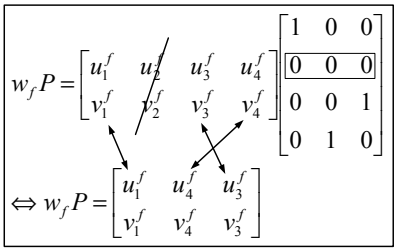

Figure 1. Rowwise Partial Permutation Matrix

In (2) p_0 identifies the number of features that will be matched, i. e. not discarded. p_0 is equivalent to the number of features in w_1, which contains no outliers. Note that it is assumed that $p_0 \leq p_k, \forall k$ - for this reason, P_k is usually a rectangular matrix, since a null 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 in [8] that a measurement matrix similar to the one presented in (2) is highly rank-defficient. 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.

When processing a video sequence up to a frame f, our problem is thus to find the set of partial permutation matrices $P_2, ..., P_f$ that chooses and orders features in $w_2, ..., w_f$ so as to generate a rank-4 W_f. Note that the afore-mentioned rank constraint, as it acts on W_f as a whole, requires that *all* permutation matrices be recalculated each time a new frame is processed. To avoid the

complexity of solving simultaneously for all P_k, we choose to solve the problem for each partial permutation matrix in sequence, while keeping the remaining matrices constant. In practice, we use a cyclic coordinate descent algorithm to solve an optimization problem in $\mathcal{P}^2 \times \mathcal{P}^3 \times ... \times \mathcal{P}^f$, where $\mathcal{P}^k$ represents the space of all partial permutation matrices of dimension $p_k \times p_0$.

2.3. A permutation-based cost function

At the k^{th} step of the cyclic coordinate descent algorithm, we solve for P_{f-k+1} individually. In this section we develop a cost function that solves the correspondence problem for the $f-k+1^{th}$ frame while assuming all other frames are correctly matched.

We consider the SVD decomposition of W_f

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

and define Z as

$$Z = W_f W_f^T = \begin{bmatrix} w_1 w_1^T & w_1 P_2^T w_2^T & \cdots & w_1 P_f^T w_f^T \\ w_2 P_2 w_1^T & w_2 P_2 P_2^T w_2^T & \cdots & w_2 P_2 P_f^T w_f^T \\ \vdots & \vdots & & \vdots \\ w_f P_f w_1^T & w_f P_f P_2^T w_2^T & \cdots & w_f P_f P_f^T w_f^T \end{bmatrix} \tag{5}$$

As can be seen in (5), Z is a function of all permutation matrices. However, all of them will be considered as constant except for matrix P_{f-k+1} when solving the problem. The aim of our cost function is to find the matching solution for frame $f-k+1$ that approximates a rank-4 W_f at the k^{th} step of the cyclic coordinate descent algorithm. This is equivalent to 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, from the following expression, where q_i represents the i^th column of Q, i. e. the i^{th} eigenvector:

$$\lambda_i = q_i^T Z(P_2, ..., P_{f-k+1}, ..., P_f) q_i \tag{6}$$

In (6), each $P_n \in \mathcal{P}^n$. As before, $\mathcal{P}^n$ represents the space of all partial permutation matrices of dimension $p_n \times p_0$.

Note that a cost function that relies solely on the minimization of λ_5 might not be effective in obtaining a rank-4 matrix, due to the fact that there would be no control on the behavior of the remaining non-dominant eigenvalues. The consequence of this would be that these eigenvalues could take on values that would induce a non-optimal solution solution, despite a low λ_5. Consequently, we choose to minimize the sum of all non-dominant eigenvalues. When processing frame f, this implies the minimization of the sum of the $2f-2$ smallest eigenvalues, since W_f has $2f+2$ rows when the f^{th} frame is processed (the practical need for two additional rows in W_f will be made clear in the section dedicated to initialization).

It should be noted when minimizing the eigenvalues that the value of each λ_i depends itself on the structure of P_{f-k+1}. Our matching problem must thus be formalized as the search for the optimal partial permutation matrix P^*_{f-k+1} such that:

$$P^*_{f-k+1} = \underset{P_{f-k+1}}{\arg\min} \left(\sum_{i>4} \lambda_i(\hat{P}_2, ..., P_{f-k+1}, ..., \hat{P}_f) \right) = \underset{P_{f-k+1}}{\arg\min} \left(\sum_{i>4} \underset{\substack{q_i^T q_i = 1 \\ s.t. \ q_i^T q_j = 0 \\ i<j}}{\max} q_i^T Z(\hat{P}_2, ..., P_{f-k+1}, ..., \hat{P}_f) q_i \right) \tag{7}$$

The partial permutation matrices that are not being optimized are represented with a *'hat'*, to emphasize that their values correspond to estimates that are being held constant at the current iteration. Note that there is an interdependency between the values of the permutation matrices and the eigenvectors of Z. To solve this problem, our algorithm runs iteratively, alternating the optimization on P_{f-k+1} and on q_{nd} (the set $\{q_5, ..., q_{2f+2}\}$ of non-dominant eigenvectors of Z). At each iteration of the algorithm a new P_{f-k+1} is calculated by minimizing the cost function using the current estimate of q_{nd}. The maximization in q_{nd} is subsequently solved by extracting the eigenvectors of Z corrected according to the newly found P_{f-k+1}. The algorithm will continue to iterate alternatively in P_{f-k+1} and q_{nd} until convergence is achieved. Our algorithm requires an initial estimate for q_{nd}. This is mathematically equivalent to estimating a base of the null space of W_f at the beginning of each iteration.

3. Methodology

3.1. Constructing a linear cost function

Each term (corresponding to the i^{th} eigenvalue) of the cost function we obtained in the previous section is nonlinear in P_{f-k+1}, due to the element $P_{f-k+1}P_{f-k+1}^T$ in the diagonal of Z. However, since P_{f-k+1} is a partial permutation matrix, this term can be written as:

$$P_{f-k+1}P_{f-k+1}^T = \begin{bmatrix} \sum_{i=1}^{p_0} {P_{f-k+1}}_{1i}^2 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sum_{i=1}^{p_0} {P_{f-k+1}}_{p_{f-k+1}i}^2 \end{bmatrix} \quad (8)$$

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

Given a matrix M, the *vec* operator stacks its columns in order to form a vector. Rearranging P_{f-k+1} as $x = vec\,(P_{f-k+1})$, 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:

$$\begin{aligned} & x* = \arg\min_x c.x \\ & s.t. \\ & x = vec\,(P_{f-k+1}), P_{f-k+1} \in \mathcal{P}^{f-k+1} \end{aligned} \quad (9)$$

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

$$\sum_{i>4} q_i^T Z(\hat{P}_2, ..., P_{f-k+1}, ..., \hat{P}_f) q_i \quad (10)$$

in order to the elements of P_{f-k+1}, given q_{nd}. Since this expression is linear in P_{f-k+1} it is possible to rewrite it as a dot product of two vectors c and x, where x gathers the elements of P_{f-k+1} as described in (9). Under these conditions, c is given by the sum of all c_i associated to the i^{th} eigenvector of Z. each c_i is given by:

$$\begin{aligned} & c_i = 2\left[\left(q_1^T W_c\right) \otimes \left(q_2^T w_k\right)\right] + \\ & 1_{[1\times p_0]} \otimes \left[\left(w_k^T q_2\right)^T \bullet \left(w_k^T q_2\right)^T\right] \end{aligned} \quad (11)$$

where the details of the calculation of c_i are given in the Appendix.

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 values of P_{f-k+1}. It is also known that the minimum of a concave 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 concave 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 $\mathcal{P}^k$ is the set of rowwise substochastic matrices $\mathcal{S}^k$. This set can be defined by the second and third equations in (3) and by the condition

$$P_{kij} \geq 0, \forall i = 1...p_k, \forall j = 1...p_0 \quad (12)$$

The resulting problem is thus equivalent to the original, but for this class of problems (concave programming problems) there exist several efficient algorithms that can provide an adequate solution. The process is depicted in Figure 2:

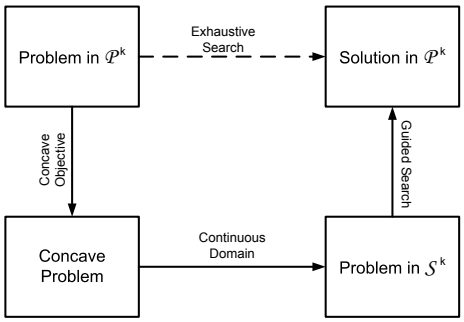

Figure 2. Relaxation Process

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

3.2. Finding an optimum set of permutation matrices

As described in the previous sections, for each new frame in the sequence all permutation matrices are recalculated. This is done by first determining P_f assuming P_2,...,P_{f-1} as constant. The new P_f is used to correct q_{nd} and P_{f-1} is then recalculated assuming P_2,...,P_{f-2}, P_f constant. This procedure is repeated until every permutation matrix has been recalculated. Convergence is achieved when all matrices have been recalculated and no change is recorded in any of them.

By using this process, we allow a matching error in an earlier frame to be corrected through the introduction of new frames.

3.3. Initializing the algorithm

Each time a new frame is made available, an estimate of the non-dominant eigenvectors q_{nd} of Z is required to bootstrap the algorithm. However, the last pair of rows of Z, corresponding to the new and still unmatched frame, is not known. To overcome this problem, we assume that the movement in the scene is smooth, in the sense that the 3D movement from frame $f-2$ to frame $f-1$ is equal to the

movement from frame $f-1$ to f. By using this approach we are able to determine a rough estimate of the position of the feature points in the new frame. The eigenvectors of the resulting $\bar{Z}$ can be used to start the method.

The key assumption in this process is that the movement between the images is small and smooth, so that the estimation of the initialization vectors can be precise. As can be seen in section 5., this assumption is true for most video sequences that have been built with a reasonable sampling rate.

Note that the subsequent steps of the cyclic coordinate descent algorithm do not have to be initialized, since they use the set of eigenvectors that results from the previous step. Consequently, the initialization problem only exists for the first step, i.e. once per frame.

Using the framework described in the previous sections, it is possible to process an image sequence automatically. However, when determining the correspondence for w_2, no previous initialization is available and the above mentioned method to determine the q_{nd} can consequently not be used. In this case, it becomes necessary to estimate the initialization vectors for the first case (which is often possible if the disparity is small enough) or to provide them externally.

Another problem arises when dealing with frame 2: since only two frames (1 and 2) would be available at the time, W_2 would, under normal conditions, be rank-4 for *all* P_2. This means that at the start of the algorithm, an additional pair of rows has to be available so that P_2 can be computed. To circumvent this issue several solutions are possible: we provide an additional image w_0, so that w_1* actually contains two aligned frames w_0 and w_1. The matching of these two frames can be achieved, for instance, with [6]. Under these circumstances W_2 has 6 rows, so that only the *correct* P_2 will cause this matrix to be rank deficient. If w_0 is chosen judiciously this also has the side effect of eliminating numerical problems.

3.4. Correcting matching errors

When determining a P_k, 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 steps of the cyclic coordinate descent algorithm.

To prevent this error integration process, we use a correction algorithm that identifies the mismatched features by reprojecting the obtained W_f into a rank-4 W_f^{fact}. By assuming a rank-4 approximation of W_f, the base R_f for the row space of W_f can be calculated from the rows corresponding to previously matched frames. By using the columns of W_f and R_f that correspond to correctly matched features we can determine, in the least-squares sense, a corresponding base C_f for the column space of W_f. The values of the incorrecly matched features can then be recovered by multiplying C_f by the submatrix of R_f containing 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. Summary of the algorithm

Based on the previous chapters, we present in this section an outline of the steps necessary to process a video sequence.

4.1. The sequencing process

1. Extract the set of observations w_f corresponding to a new frame. This is the current frame ($k = 1$).
2. Given q_{nd} and w_1, w_2, ..., w_f, run the Matching Process in order to obtain the partial permutation matrix P_{f-k+1} corresponding to the current frame.
3. Perform error correction by feature reprojection as described in 3.4 until a stable P_{f-k+1} is obtained.
4. Correct q_{nd} and repeat 2. and 3. with $k = k + 1$, until $k = f - 1$.
5. If any change in the set of permutation matrices is recorded, go to 2. and repeat the algorithm with k reset to 1.
6. Using the new results determine the initialization vectors q_{nd}^{new} for the next frame, by extracting the eigenvectors of $Z(P_2, ..., P_f)$.

4.2. The matching process

1. Given $w_1, ..., w_{f-k+1}, ..., w_f$ and the appropriate q_{nd}, build a linear cost function for P_{f-k+1} as detailed in 2.3 and 3.1.
2. Solve the integer optimization problem for P_{f-k+1} by using relaxation as referred in 3.1.
3. If P_{f-k+1} has converged, stop.
4. Given P_{f-k+1}, update q_{nd} as the eigenvectors of $Z(P_{f-k+1})$.
5. Return to 1.

5. Experiments

We tested our algorithm on two sets of data - the synthetic 'Cube' 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 algorithm's ability to successfully track a rigid object in a video sequence. In both experiments, the algorithm is initialized by the user: in the 'Cube' 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.

It can easily be seen that the number of rows of W_f grows linearly with the number of frames. Although the number of variables in *each* linear program does not suffer an increase the minimum number of cyclic coordinate descent iterations will increase significantly, thereby significantly slowing down the algorithm. In practice, this problem can be avoided by assuming that after a certain number of frames have been processed the first n matching solutions can be considered correct and will thus not be iterated upon. This approximation has led to a significant decrease in processing time, while not affecting the final result's precision.

5.1. Cube sequence

The synthetic sequence is composed by 55 frames of a rotating and translating cube. The matching candidates consist of the edges of the cube and three equidistant points are selected on each of the cube's edges as the points to be tracked. No noise has been added to this sequence and its principal objective is to validate our method.

Note that the cube's edges are the only contours present in the image: traditional photometric methods would suffer from the aperture problem when dealing with points on these straight lines. It should be emphasized that there are corners in the image (the vertices of the cube!); however, in order to prove our point, the features are selected so that they are not in the vicinity of these corners. In fact, due to the geometric nature of the algorithm corners and generic contour points are treated equally.

In each frame, the match for the 36 inliers is found among ca. 670 candidates. Consequently, each frame would require the solution of a linear program with over 24000 variables. As explained previously, using *a priori* knowledge reduces the dimensionality of the problem.

5.2. Hotel sequence

This sequence is 30 frames long and contains images of a moving toy house. 22 points (inliers) have been chosen from the contour lines of the first image. As in the 'Cube' sequence, these points have been chosen in such a way that they reside, as much as possible, on straight 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 242000 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.

5.3. 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 table below:

'Cube'		*'Hotel'*	
min	**max**	**min**	**max**
0.13 pxl	1.64 pxl	0,10 pxl	1,51 pxl

Note that in either sequence the outliers are 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 incorrect match.

As can be seen in Figures 3, 4, 6 and 7, all features are tracked with a diminute error. In the case of the 'Cube' Sequence the mean error of the feature points in relation to the theoretical values was always below one pixel. A null error was not achieved due to discretization errors that do not allow the pixel coordinates to coincide exactly with the theoretical values. For the 'Hotel' sequence no ground-truth exists; however, it is possible to assertain by visual inspection that the feature points move solidarily with the object.

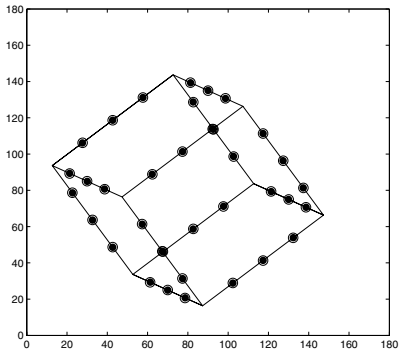

Figure 3. 1^{st} **frame of the 'Cube' Sequence**

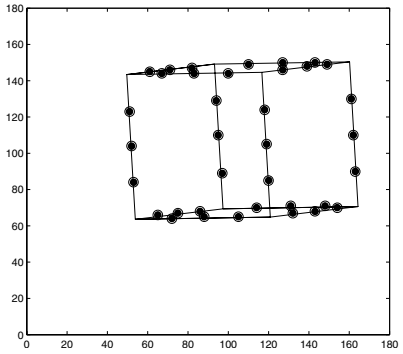

Figure 4. 55^{th} **frame of the 'Cube' Sequence**

6. Future work

The present version of our algorithm is limited by the fact that the disparity between images has to be small so that an adequate initialization in the subsequent frame is achieved. In this paper, the result of the previous matching process is used and the new initialization vector is extrapolated by assuming a smooth movement. Better results can be achieved by using more sofisticated ways to extrapolate future movement and thus calculate initialization vectors that are valid over larger disparities.

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. Even for small problems, this fact imposes restrictions on the choice of image points to match. Handling occlusions is at the moment still an open issue.

The algorithm can be made to run radically faster by subdividing the simplex problems so that only a subset of the image points are matched at a time. As the simplex algorithm is typically solved in polinomial time, processing n d-dimensional problems is usually much faster than processing a single $(n \times d)$-dimensional problem.

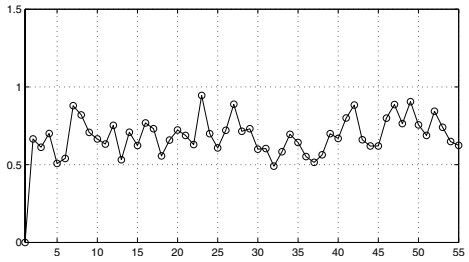

Figure 5. 'Cube' Sequence: Mean Error per frame in pixels

Figure 6. 1^{st} **frame of the 'Hotel' Sequence**

Figure 7. 30^{th} **frame of the 'Hotel' Sequence**

Our algorithm is presently constrained to use images obtained from affine cameras. The extension to projective cameras might be obtained using Heydens work in [3]. However, an analysis on the convergence properties of the altered method will have to be performed in order to ensure that performance is not affected.

7. Discussion 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. This algorithm is able to calculate the match of several images, thus optimizing the result in the whole image sequence.

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 capabilities of this method.

8. Appendix: Calculation of c

From (5), we can simplify the calculations by reordering W_f so that the frame that is to be aligned occupies the last pair of rows. We will consider the submatrix of the remaining rows W_c as constant, which allows us to disregard all of its elements since they do not depend on P_k. This allows considerable simplification of the calculations to be made and, since only constant terms are eliminated, does not affect the value of P_k^*.

$$Z_0 = \begin{bmatrix} 0_{[2f\times 2f]} & 1_{[2f\times 2]} \\ 1_{[2\times 2f]} & 1_{[2\times 2]} \end{bmatrix} \bullet \begin{bmatrix} W_c W_c^T & W_c P_k^T w_k^T \\ w_k P_k W_c^T & w_k P_k P_k^T w_k^T \end{bmatrix} \tag{13}$$

The i^{th} term of the cost function in (7) (which corresponds to the minimization of the i^{th} eigenvalue of Z) can then be written as:

$$q_i^T Z_0 q_i = \begin{bmatrix} q_i^1 \\ \vdots \\ q_i^{2f+2} \end{bmatrix}^T \begin{bmatrix} 0_{[2f\times 2f]} & W_c P_k^T w_k^T \\ w_k P_k W_c^T & w_k P_k P_k^T w_k^T \end{bmatrix} \begin{bmatrix} q_i^1 \\ \vdots \\ q_i^{2f+2} \end{bmatrix} \tag{14}$$

Note that this calculation need be performed $2f-2$ times, corresponding to the number of non-dominant eigenvalues that have to be minimized in order to obtain a rank-4 Z.

In order to present this problem as a linear program, we first divide q_i in the following manner (the subscript i on q will from now on be dropped in order to simplify notation):

$$q = \begin{bmatrix} q_{1_{[2f\times 1]}}^T & q_{2_{[2\times 1]}}^T \end{bmatrix}^T \tag{15}$$

Using (15) we can develop (14) as follows:

$$q^T Z_0 q = 2 \sum_{m=1}^{p_k} \sum_{n=1}^{p_0} \left(\sum_{i=1}^{2} q_{2_i}^T w_{k_{im}} \right) \left(\sum_{j=1}^{2f} (W_{cjn})^T q_{1_j} \right) P_{k_{mn}} + \sum_{m=1}^{p_k} \sum_{n=1}^{p_0} \left(\sum_{i=1}^{2} \left(w_{k_{mi}} \right)^T q_{2_i} \right)^2 P_{k_{mn}} \tag{16}$$

Note that in the second term we take advantage of the fact that $P_k P_k^T$ is a diagonal matrix. We can express (16) as a function of $x = vec(P)$. As a consequence, (16) assumes the form $c.x$, where c is given by:

$$c_i = 2\left[\left(q_1^T W_c \right) \otimes \left(q_2^T w_k \right) \right] + 1_{[1\times p_0]} \otimes \left[\left(w_k^T q_2 \right)^T \bullet \left(w_k^T q_2 \right)^T \right] \tag{17}$$

The complete c is given by the sum of all c_i.

References

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

[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 of Contour Points Through the Use of Rank Cosntraints. In *Proc. CVPR*, June 2005.

[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.