Efficient Computation of the Most Probable Motion from Fuzzy Correspondences

Moshe Ben-Ezra   Shmuel Peleg   Michael Werman

Institute of Computer Science
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
Email: {moshe, peleg, werman}@cs.huji.ac.il

Abstract

An algorithm is presented for finding the most probable image motion between two images from fuzzy point correspondences. In fuzzy correspondence a point in one image is assigned to a region in the other image. Such a region can be line (aperture effect) or a convex polygon. Noise and outliers are always present, and points may belong to different motions. The presented algorithm, which uses linear programming, recovers the motion parameters and performs outlier rejection and motion-segmentation at the same time. The linear program computes the global *optimum without a need for initial guess.*

1. Introduction

Many methods have been developed for motion recovery, yet the recovery of motion parameters in the presence of noise, outliers and multiple-motions remains a challenge. This paper describes a new approach that overcomes many of the limitation of existing methods including errors due to outliers and to multiple motions, and the need for a good initial guess of the motion model to avoid local minima. Section 2 describes the new algorithm. Section 3 presents several test results, and finally, Section 4 summarizes the advantages of the new algorithm.

1.1. Previous Work

Given a set of matched points between two images (from an optical flow or from feature matching) a linear parametric image motion (such as an affine motion) can be recovered using a linear pseudo inverse equation system that minimizes the average error (RMS, L_2 metric). This RMS minimization is valid only if the errors have zero mean. It will fail in the presence of outliers and multiple motions. Moreover, motion computation that uses matched pairs of points cannot express uncertainty directly. In order to overcome these drawbacks several methods have been developed that utilize one or more of the following techniques:

Motion segmentation [1] - Techniques for outlier detection are used, usually combined with motion recovery by an iterative algorithm.

Probabilistic algorithm [9] - Algorithms that calculate the motion parameters from randomly selected pairs of matched points until they reach the desired accuracy.

Probabilistic matching [7, 8] - A point in one image corresponds to a distribution over locations in the second image, such distribution can be represented as a probability matrix over possible displacement. Motion recovery can be viewed as maximizing the combined likelihood of many local matches. When the motion consists of pure translation, the local motion given by each probability matrix matches the global motion. In this case the most probable global motion can be recovered directly from the local probability matrices. When the motion is more complex, the other parameters (rotation, scale) are recovered by extensive search over the parameter space.

Direct computation from grey level [4, 6] - Algorithms are based on the constant brightness assumption and the optical flow constraint. They are usually combined with motion segmentation methods in an iterative manner.

Global alignment of local measures [3] - This algorithm is a generalization of the direct grey level algorithms in the sense that it is not restricted to grey level minimization. The algorithm defines *match-measure sur-*

face over the local match field and uses Newton iterations to maximize (or minimize) the sum of the local measures.

2. Motion Computation from Fuzzy Correspondence

A point P_i in image I_1 is *fuzzy corresponding* to a group G_i in image I_2 if the target of P_i, denoted as P_i', is located within an area (or *a group*) in I_2 that is designated by G_i. There are various ways of defining G_i. We start our discussion with groups that are defined as *convex* polygons. Fig.1.a illustrates *fuzzy correspondence* of four points. Each group G_i is represented by its vertices $G_i^1..G_i^k$.

Point P_i is mapped to the (unknown) point P_i' where P_i' can be expressed as a linear combination of vertices: $G_i^1..G_i^k$.

The motion computation problem is defined as:
Given a set of pairs: (P_i, G_i) find the best parametric motion that maps the source points P_i in image I_1 into their target points P_i' in image I_2 where P_i' is a linear combination of G_i.

Subproblems are:

1. If the vertices G_i^k have different weights that represent likelihood, find the best motion that maximizes the combined likelihood.

2. If the set of pairs contains outliers or multiple motions, disregard the outliers while finding the parametric motion (and find the pairs that belong to the recovered motion). This problem that is called the motion segmentation problem, has an inherent difficulty: the parametric motion is easily recovered if the outliers are known and the outliers are easily found if the the parametric motion is known - solving for both presents a difficulty. The problem increases when multiple motion of similar number of points exists.

A well known special case of fuzzy correspondences is the aperture effect, or the recovery of global motion from normal flow. Fig.1.b illustrates the aperture effect problem and the normal flow for a pure translating object. The width of the groups represents the uncertainty in the magnitude of the normal flow and the length of the groups is the aperture effect uncertainty.

The normal flow vector can be derived directly from the image grey levels using the well known optical flow constraint [2]. The target points resides on the perpendicular line to the normal flow vector in an unknown position.

Motion computation from fuzzy correspondences is demonstrated in this paper using affine parametric motion. Fuzzy correspondences are also applicable to the linear 2D-quadratic motion model.

2.1. Implementation

The algorithm is implemented by mapping the motion computation problem into linear programming [5]. The linear program itself is solved using a standard linear programming algorithm. The most important features of the algorithm: global optimization and outlier rejection, are induced by the properties of the linear program and the robustness on the L_1 metric (that minimizes the median error and therefore is insensitive to extreme points, in contrast to L_2 metric that minimizes the average error).

We first describe the mapping of the geometrical motion into linear programming constraints (Section. 2.2), then we describe the *indexing relation* that connects the group interpolation constraints to a selection vector of linear programming variables (Section. 2.3), then we describe the linear programming objective function which optimizes the selection of the vertices in each group by their weight to get the maximum likelihood solution (Section. 2.4). Finally we will refine the program allowing it to reject outliers and have better control on the requested motion; for example, allow only rotation, translation, and scale (Section. 2.5)

The input for the mapping is a set of n pairs $\{P_i, G_i^k\}$. Each pair represent a mapping from point P_i in image I_1 into the group of $k_{(i)}$ vertices $G_i^{k(i)}$ in image I_2. Each group has its own number of vertices k which is a function of i, however in order to make the notation more readable we will just use a uniform k for all groups. When we refer to a group as a whole - we will use the notation G_i.

An optional weight vector C_i can be assigned to each group. This weight vector represent preference of the vertices of G_i.

2.2. The Geometrical Motion Constraint

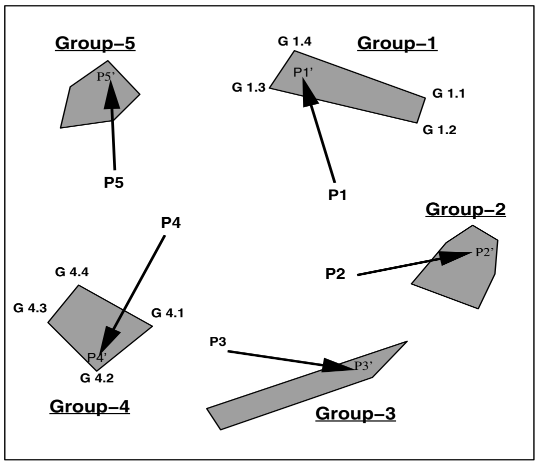
The affine transformation that maps point $P_i = (x_i, y_i)$ in image I_1 to the (unknown) point $P_i' = (x_i', y_i')$ in image I_2 is given by the following pair of constraints: $x_i' = ax_i + by_i + e, \quad y_i' = cx_i + dy_i + f$ Where: a, b, c, d, e, f are variables of the linear program that are *common* to all n pairs of these geometrical constraints.
The value of P_i' in unknown. However it is known that that P_i' is a convex combination of the vertices of the group G_i.

2.3. The Indexing relation

The relation between the point P_i' and its group G_i is given by the convex coefficient vector S_i as:

$$P_i' = \sum_{j=1}^{k} S_i^j G_i^j \quad 0 \le S_i^j \le 1 \qquad (1)$$

P_i can be defined by selection of at most 3 vertices of G_i.

a) 

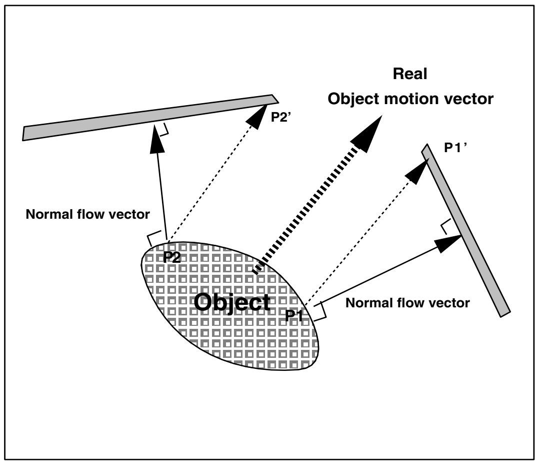
b) 

Figure 1. Fuzzy correspondence. A Point P_i in image I_1 is mapped to a group G_i in image I_2. a) Points to groups. b) Normal flow as fuzzy data.

These vertices are selected by three corresponding values of S_i^j that are non zero. The vector S_i is called the *selection vector* for the pair $\{P_i, G_i\}$ and each element G_i^j is the selection values for vertex j in group G_i.

2.4. The Maximal Selection constraint

Let $S = [S_n^k]$ be the selection matrix of all group vertices. S_i^j is the selection *variable* for vertex j in group i, $0 \leq S_i^j \leq 1$. Row i of the matrix S is the selection vector for group G_i.
S satisfies the *selection constraint*: $\forall i \; \sum_{j=1}^{k} S_i^j = 1$.
Let $C = [C_n^k]$ be the weight matrix for all group vertices. C_i^j is the predefined weight of vertex j in group i. For groups that represent convex probability matrices, C_i will satisfy: $0 \leq C_i^j \leq 1, \forall i \; \sum_{j=1}^{k} C_i^j = 1$

Let:

$$T = \sum_{i=1}^{n} \sum_{j=1}^{k} C_i^j S_i^j \tag{2}$$

Then $Max(T)$ over the selection matrix S satisfies the following properties: (The maximal selection properties).

1. If the assignment of S is constrained only by the the selection constraint and $\forall i, \quad C_i^j \neq C_i^l, \quad j \neq l$ then the assignment S that maximizes T will be an integral {0,1.0} matrix containing exactly n instances of the value 1.0. Each selection vector S_i will have a single instance of the value 1.0 corresponding to the maximal member of C_i. This is true since the maximum of C_i is larger than the average of any subset of C_i (that has more than one member).

2. If S is constrained by the selection constraint and by the geometrical constraint (via the indexing relation) than each row S_i will have at most 3 non-zero values, where one of the values corresponds to the maximum member C_i. (S_i will have at most two non-zero values if G_i is located on a line). This is true since P_i', that is located within the convex-hull of G_i is located in one of the two convex partitions created by the line that is defined by the maximal point and any other point of G_i.

Notes:

1. If some of the values of C are equal then there could be more non-zero values - but the objective function value and the recovered motion will not change (since the geometrical constraints can be satisfied with no impact on the objective function). equal

2. If the algorithm is forced to select more points than the maximum point m of some group (due to the geometrical constraint) - it will tend to select the other point w geometrically far from m since this will maximize the weight of m itself. This behavior only increases the selection of the maximal likelihood provided that the groups are convex shaped.

3. The Selection matrix S plays two roles - It is the *geometrical interpolation* values and it is also the *weight selection* values. S actually *selects* at most three vertices and interpolate only these weights. (This stands in contrast to to interpolation of all points by their distance which leads to L_2 metric that we wish to avoid).

4. In groups having convex shapes, the triangle defined by the three selected points (one of which has the max-

imum weight) is a continuous linear approximation to the surface near the maximum point - this property enables efficient and robust solution of surface like optimization using linear programming.

5. Non-convex shape groups are dealt with by splitting them into convex shaped groups. The outlier rejection will dispose of the wrong partitions. In extreme cases (checker-board) each group will be of size one and the algorithm will have no geometrical interpolation - it will be reduced to an L_1 selection of points (which is still much better that L_2 due to its outlier rejection property).

2.5. Outliers Rejection and Transformation Control

In order to be able to reject outliers (points that do not agree with the global motion that maximizes T). Free variables Z_i were added to each geometrical constraint. The variable Z_i corresponds to the geometrical error of the group G_i. The total number of Z variables is $2n$ (one for the X axis constraint and one for the Y axis constraint).
In order to limit the error we subtract $\alpha \sum_{i=1}^{2n} \beta_i |Z_i|$ from T, where α is a parameter used to adjust the units and metric weights between the selection vector units (0..1, L_∞) and the Z error units (0..image-radius, L_1), and β is an optional preference parameter for the whole group.
When T reaches its maximal value the Z variables contains match information and therefore can be used for segmentation purposes as feedback weights for iterative application of the algorithm (eliminating the need for threshold selecting).

2.6. The Linear Program

In order to get a linear program, the only changes required are the reshaping the C and S matrices into one dimensional vector.
In order to make the naming convention of the *vectors* C and S compatible with the matrix naming convention, double *vector* index is used: $S_i^j \equiv S_{ij}$.
The final linear program is given by:

$$max\ :\ C^t S - \alpha \sum_{i=1}^{2n} \beta_i |Z_i|\ \ s.t. \tag{3}$$

$\forall i$

$$\begin{array}{lcl} S_{ij} & \geq & 0 \\ \sum_{j=1}^{k} S_{ij} & = & 1 \\ \sum_{i=1}^{k} S_{ij} G_{ij}.x & = & ax_i + by_i + e + Z_i \\ \sum_{i=1}^{k} S_{ij} G_{ij}.y & = & cx_i + dy_i + e + Z_{n+i} \end{array}$$

The affine transformation can be limited, for example, to rotation + scale + translation *only* (no affine distortion) by adding the constraints: $a = d,\ \ c = -b$. (The conversion of the linear program into standard form is simple and requires two variables for each Z variable and two variables for each motion parameter).

3. Experiments

All tests have used a uniform $C = 1$, $\alpha = 0.001$, $\beta = 1$ which gives equal preferences to all vertices. (This is a worst case scenario - no preference information exists).

3.1. Synthetic Test Results

3.1.1. Outlier Rejection Test

In this test selected 59 pairs of points were selected randomly (group size = 1) in the range (-100, 100), belonging to the affine model-1 and 41 points belonging to the affine model-2 ($\approx 40 : 60\ ratio$). The accuracy of the input was up to 0.5 pixels, as only integer locations were used. The results are summarized in Table 1. Errors were calculated as Euclidean distances in the image plane between the ground truth and the image of the *recovered* affine transformation.

Fig. 2 shows the error of all the point sorted by distance. The points that belong to the recovered model are the first 59 points. The rest of the points are considered outliers. The segmentation into model and outliers is very clear.

	Affine Model-1	Affine Model-2
Original	1.055 -0.598 2.593 0.598 1.055 3.222	0.031 -0.199 -3.760 0.199 0.031 -1.951
Recovered	1.048 -0.597 2.806 0.597 1.048 3.175	(Outliers)
Points	59	41

Mean(Error)
Var(Error)
	0.823
	0.036
		10.77
		3.3

min(Error)
max(Error)
	0.371
	1.189
		6.49
		13.79

Table 1. Outliers Rejection. (Units: Pixels).

3.1.2. Polygon Uncertainty and Outlier Rejection

In this test we used the same original data as in the previous test. This time we gave the algorithm groups of four points which are bounding rectangles of the real destination. The bounding rectangle vertices were selected randomly using uniform distribution in the range (-3..+3) pixels. All vertices had equal weight of one. (The real location of each point can be anywhere within the bounding rectangle).

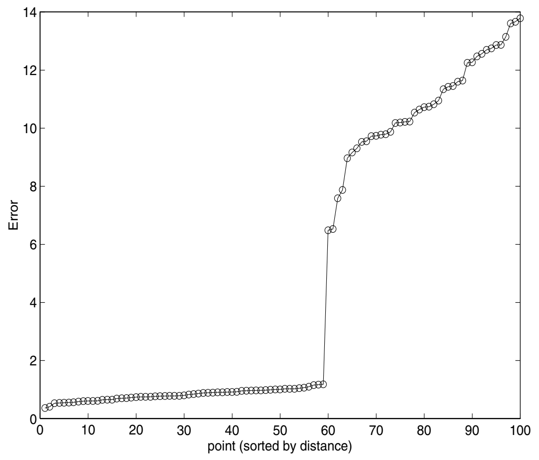


Figure 2. Outliers.

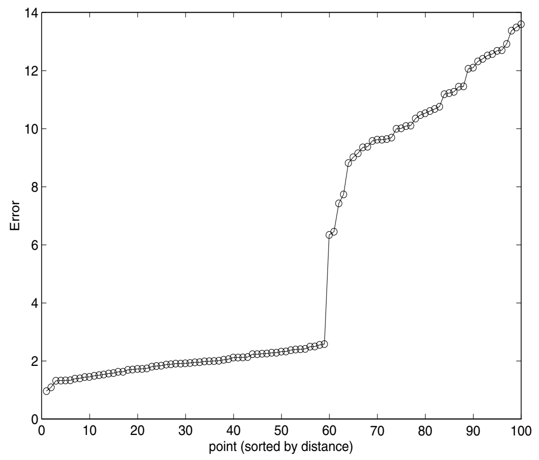


Figure 3. Outliers and uncertainty.

The results of this scenario is presented in Table 2 and Fig. 3. The segmentation is clear but the mean error of 1.9 pixels is still to large. To solve this, a second iteration was made giving weight of 0 to the outliers that were discovered by the segmentation at first iteration. The results of second iteration results are presented in Table 3 and Fig. 4.

	Affine Model-1	Affine Model-2
Original	1.055 -0.598 2.593 0.598 1.055 3.222	0.031 -0.199 -3.760 0.199 0.031 -1.951
Recovered	1.018 -0.587 3.082 0.587 1.018 2.815	(Outliers)
Points	59	41

Mean(Error)
Var(Error)
	1.90
	0.15
		10.60
		3.25

min(Error)
max(Error)
	0.96
	2.58
		6.34
		13.59

Table 2. Outliers and uncertainty. (Units: Pixels).

	Affine Model-1	Affine Model-2
Original	1.055 -0.598 2.593 0.598 1.055 3.222	N/A N/A
Recovered	1.056 -0.598 2.654 0.598 1.056 3.343	N/A
Points	59	N/A

Mean(Error)
Var(Error)
	0.47
	0.01
		N/A
		N/A

min(Error)
max(Error)
	0.19
	0.66
		N/A
		N/A

Table 3. 2nd iteration. (Units: Pixels).

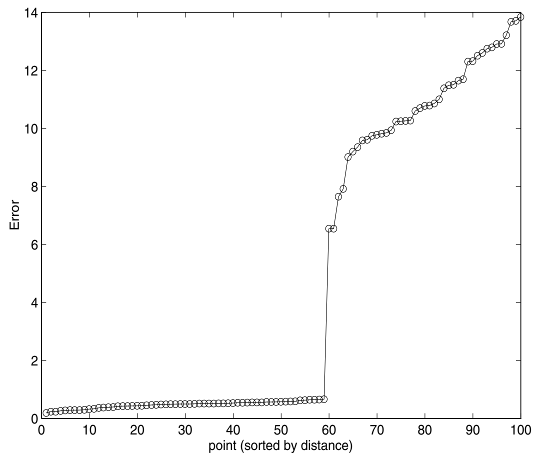


Figure 4. 2nd iteration.

3.2 Images test results

Two images with large displacement were selected form the “Puma” robot sequence. 14 points were manually selected from several locations in the image that have clear 3D structure (therefore they do not agree with any single affine transformation). The affine transformation was recovered using a pseudo-inverse (with the *exact* displacements) and by the linear programming algorithm (using group of uncertainty of one pixel at each direction). The results are shown in Fig. 5. We can see that the pseudo-inverse algorithm reduced the average error but couldn't fully register anything at the scene while the linear programming algorithm have fully registered half of the points after a single iteration (Z = 0) resulting in a much better registration.

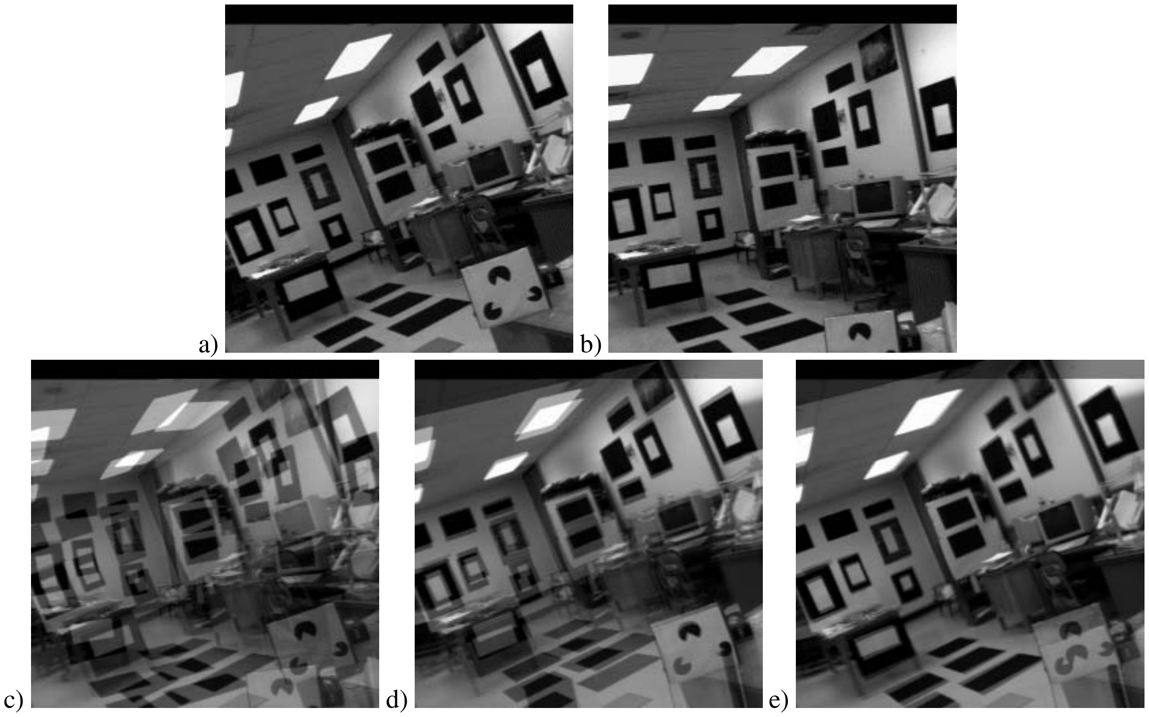


Figure 5. a) First frame. b) Second frame. c) Frames before registration. d) Pseudo-inverse registration. e) Linear programming registration.

4. Concluding Remarks

A linear programming algorithm for the recovery of parametric motion has been introduced. This algorithm has the following advantaged over conventional methods:

1. The algorithm utilizes fuzzy input data that can span over large displacements using L_1 metric for optimization.

2. The recovery of the motion parameters as well as the outlier rejection and motion segmentation is done simultaneously.

3. The algorithm does not require an a-priori initial guess of the transformation

4. The algorithm provides *global optimization* of the objective function which is maximal probability.

5. The algorithm solution can be controlled by adding more constraints (such as: pure scale, rotation and scale, rotation scale and translation).

References

[1] M. Ben-Ezra, S. Peleg, and B. Rousso. Motion segmentation using convergence properties. In *Image Understanding Workshop (ARPA94)*, pages II:1233–1235, 1994.

[2] B. Horn and B. Schunck. Determining optical flow. In *AI*, volume 17, pages 185–203, 1981.

[3] M. Irani and P. Anandan. Robust multi-sensor image alignment. In *ICCV98*, pages 959–966, 1998.

[4] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using region alignment. In *PAMI*, volume 19, pages 268–272, March 1997.

[5] H. Karloff. *Linear Programming*. Birkhäuser Verlag, Basel, Switzerland, 1991.

[6] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[7] Y. Rosenberg and M. Werman. Representing local motion as a probability distribution matrix and object tracking. In *DARPA Image Undersading Workshop*, pages 153–158, 1997.

[8] Y. Rosenberg and M. Werman. Real-time tracking from a moving camera: A software approach. In *IEEE Workshop on Applications of Computer Vision (WACV98)*, 1998.

[9] P. Torr and D. Murray. Outlier detection and motion segmentation. In *SPIE*, volume 2059, pages 432–443, 1993.