

Real-Time Motion Analysis with Linear-Programming¹

Moshe Ben-Ezra and Shmuel Peleg and Michael Werman

*School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
E-mail: {moshe, peleg, werman}@cs.huji.ac.il*

A method to compute motion models in real time from point-to-line correspondences using linear programming is presented. The reason for using point-to-line correspondences is that they are the most reliable measurements for image motion given the aperture effect, and it is shown how they can approximate other motion measurements as well.

An error measure for image alignment using the L_1 metric, and based on point-to-line correspondences, achieves results which are more robust than the commonly used L_2 metric. The L_1 error measure is minimized using linear programming. While estimators based on L_1 are not robust in the breakdown point sense, experiments show that the proposed method is robust enough to allow accurate motion recovery over hundreds of consecutive frames.

The L_1 solution is compared to standard M – estimators and Least Median of Squares (LMedS) and it is shown that the L_1 metric provides a reasonable and efficient compromise for different scenarios. The entire computation is performed in real-time on a PC without special hardware.

Key Words: motion analysis; linear-programming.

1. INTRODUCTION

Robust, real-time recovery of visual motion is essential for many vision based applications. Numerous methods have been developed for motion recovery from image sequences, among them are algorithms that compute the motion directly from the grey level values or local measures of them [14, 20, 16, 3, 17]. A second class of algorithms use feature points or optical flow to recover motion [1, 9, 15]. A probabilistic error minimization algorithm [26] can be used to recover motion in the presence of outliers. Another class of algorithms use explicit probability distributions of the motion vectors to calculate motion models [23]. Black and Anandan presented [7] a framework for robust calculation of optical flow based on the Lorentzian estimator and a Graduated Non-Convexity algorithm seeking an optimal solution. Bab-Hadiashar and Suter [4] presented a robust optical flow calculation based on LMedS (Least Median of Squares).

Most of the methods cited above have problems when computing high-order motion models (e.g. an affine motion model or a homography): either they are sensitive to outliers, or the execution speed is slow. Algorithms based on iterative re-weighting M – estimators have tuning and initial guess problems, especially in multiple model cases. The LMedS faces complexity and accuracy problems when it is difficult to obtain a good hypothesis by random selection.

In this paper an algorithm is presented to recover high-order motion models from point-to-line correspondences using linear-programming. Point-to-line correspondences are robust in the sense that they are largely insensitive to aperture effects and to T-junctions, unlike the common point-to-point correspondences. Point-to-line correspondences can also approximate other measurements as well, such as point-to-point correspondences, correspondences with uncertainty, and the spatio-temporal constraint.

The L_1 metric ($\sum |a_i - b_i|$) can be used with the point-to-line correspondences, and is much more robust than the L_2 metric ($\sqrt{\sum (a_i - b_i)^2}$). For example, the median minimizes the L_1 metric, while the centroid (average) minimizes the L_2 metric. L_1 based estimators are not robust in the sense of breakdown point [11, 24, 25] as they are sensitive to leverage points. However, our experiments show that in motion analysis the L_1 error measure is robust enough to compute accurate motion over hundreds of frames, even with large moving outlier objects in the scene. Moreover, this is done in real-time on a regular PC (300MHz).

The linear programming solver does not need an initial guess nor a noise scale estimate, which are required for iterative re-weighted least-square algorithms (such as M – estimators). Comparisons between estimators based on the L_1 metric and the robust LMedS estimators show that global motion analysis using the L_1 estimator is only slightly less robust than the LMedS estimator, but the L_1 computation is much faster.

The motion analysis consists of computing a model based alignment between successive frames. The alignment process consists of two steps: (i) Computing correspondences and representing them as point to line correspondences is described in

¹ Supported by Espirit project 26247 - Vigor

Section 2. (ii) Converting the alignment problem into a linear program using the point to line correspondences, and solving it, is described in Section 3. Section 4 describes experimental results and comparisons with other methods. Section 5 gives concluding remarks. The appendix describes a possible explanation for the experimental insensitivity of L_1 motion estimators to leverage points.

2. POINT TO LINE CORRESPONDENCES

Point to line correspondences are used due to their insensitivity to the aperture effect. This section describes the aperture effect, point selection for the point to line correspondence, and the use of point to line correspondences to represent normal flow and fuzzy correspondences.

2.1. Aperture Effect

A moving line viewed through a small aperture will have an apparent motion which is normal to the line. This phenomena is called the “aperture effect”. An example of the aperture effect is show in Figure 1. Given the aperture effect we express a constraint on the displacement (u, v) such that point $\mathbf{p} = (x, y)$ in the first image has moved to the straight line $\mathbf{s}(x + u, y + v)$ in the second image defined by the line equation:

$$\mathbf{s}(x + u, y + v) \equiv a(x + u) + b(y + v) + c = 0 \quad (1)$$

Without loss of generality we assume for the rest of the paper that $a^2 + b^2 = 1$ by normalization.

2.2. Point selection

For computation stability, selected points should be spread evenly over the image. Points should be located on strong features such as edges, and they should have balanced X-direction and Y-direction information. The following steps were carried out to select points in real-time:

1. N points were evenly spread over the image in a chess-board grid.
2. “Black” points were allowed to move slightly horizontally to find strong vertical edges. “White” points were allowed to move slightly vertically to find strong horizontal edges.
3. The best K “black” points and the best K “white” points ($2K < N$) were used as the selected points.

2.3. Normal Flow

An optical flow constraint can be derived directly from image intensities using the gray level constancy assumption. This optical-flow constraint is given by [14, 20]:

$$uI_x + vI_y + I_t = 0 \quad (2)$$

where (u, v) is the displacement vector for the pixel, and I_x, I_y, I_t are the partial derivatives of the image at each pixel with respect to X-axis, Y-axis and time.

Eq. (2) describes a line, which is the aperture effect line. When $I_x^2 + I_y^2$ is normalized to 1 the left hand side of Eq. (2) becomes the Euclidean distance of the point (x, y) from the line

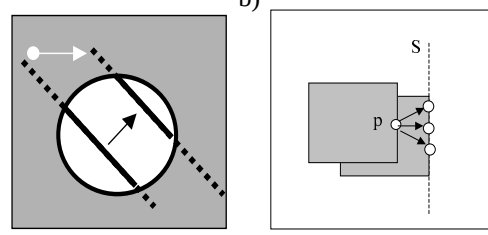


FIG. 1. Example of an aperture effect. a) The white arrow represents the actual motion, while the black arrow represents the apparent motion. b) Point to line correspondence.

passing through $(x + u, y + v)$, which is also called the normal flow [2].

2.4. Fuzzy Correspondence

An optical flow vector between successive images in a sequence represents the displacement between a point in one image to the corresponding point in the second image. While it is difficult to determine this point to point correspondence accurately from the images automatically, a correspondence is usually assigned to the most likely point. For example, given a point in one image, the corresponding point in the second image will be the one maximizing some correlation measure. However, such correspondences are error prone, especially when other points in the second image have a local neighborhood similar to the real corresponding point.

A possible solution to this problem is to postpone the determination of a unique correspondence to a later stage, and to represent the uncertainty as it is given by the correlation. In this case the correspondence will not be a unique point, but a fuzzy measure over a set of possible corresponding points.

The fuzzy correspondence of a point $\mathbf{p}$ can be represented as a matrix $\mathbf{M}(\mathbf{p})$ (the fuzzy correspondence matrix)[23]. $\mathbf{M}$ is computed using the sum of squared differences (SSD) values: $\mathbf{M}(u, v) = \alpha \sum_{i,j \in W} (\mathbf{I}_2(i+u, j+v) - \mathbf{I}_1(i, j))^2$ for a window W around point $\mathbf{p}$ in image $\mathbf{I}_1$. The parameter α is a normalization factor s.t. $\sum_{u,v} \mathbf{M}(u, v) = 1$. Each cell (i, j) of $\mathbf{M}(\mathbf{p})$ corresponds to the probability that point $\mathbf{p}$ has a displacement of (i, j) . In many cases the fuzzy correspondence matrix has a dominant compact shape: points on corners usually create a strong peak while edges form lines. A common case is an ellipse like shape. While the fuzzy correspondence matrix contains all correspondence uncertainty, utilizing this information to its full extent is difficult.

To enable computation of global motion with linear programming, we propose an approximation of the fuzzy correspondence matrix. An L_1 distance map obtained from a point to two lines correspondences. This approximation is given by two lines:

$\mathbf{s}_i(x, y) \equiv (A_i x + B_i y + C_i) = 0$ with two associated weights, w_i , ($i = 1, 2$). The weighted sum of both distances forms an L_1 “cone”. Each equidistant line on the cone is an L_1 “ellipse” with eccentricity proportional to the weights of the two lines. Figure 2 shows the approximation of a correspondence matrix by distance map. For the discrete case of the correspondence

matrix, L_1 distance is also called city block or Manhattan distance. The distance map is computed using lines and weights obtained by a weighted Hough transform over the original correspondence matrix [19]. The maximum bin corresponds to the main axis line. A second local maximum bin corresponds to the second line. The bin values are taken as the weights w_i . This approximation can also be used to express the point-to-point correspondence: $(x, y) \rightarrow (x', y')$ which can be approximated by point to two lines correspondences: between point (x, y) and the line $(x = x')$, and between the same point and the line $(y = y')$, with weights $w_1 = w_2$. The Hough transform is used to find both the lines and the weights resulting in the following constraint for the displacement of point (x, y) :

$$w_1 s_1(x + u, y + v) + w_2 s_2(x + u, y + v) = 0 \quad (3)$$

The geometrical meaning of this constraint is that point (x, y) has moved by a displacement of (u, v) and is now located on the intersection of the two lines s_1, s_2 .

3. L_1 ALIGNMENT USING LINEAR PROGRAMMING

Image alignment, or registration, is the process of recovering the coefficients of a global motion model using two images. A hierarchy (every model is an extension of the previous model) of common global motion models is [6]:

Translation - a two parameter model $(u, v): (x, y) \rightarrow (x + u, y + v)$. 2D horizontal and vertical motion.

Similarity - a four parameter model $(a, b, u, v): (x, y) \rightarrow (ax + by + u, -bx + ay + v)$. Translation, scale and rotation.

Affine - a six parameter model $(a, b, c, d, u, v): (x, y) \rightarrow (ax + by + u, cx + dy + v)$. Similarity and shear.

Homography - an eight parameter model $(a, b, c, d, e, f, u, v): (x, y) \rightarrow (\frac{ax+by+u}{ex+fy+1}, \frac{cx+dy+v}{ex+fy+1})$. True projective transformation of a plane.

Image registration is useful for many applications including (i) camera stabilization, (ii) ego-motion computation, (iii) detection of moving objects, (iv) mosaicing. The alignment process has two steps: (i) Computing correspondences and representing them as point-to-line correspondences. (ii) Converting the alignment problem into a linear-program using the point-to-line correspondences, and solving it. The first step was detailed in Section. 2, and in this section the second step is described. In particular, we show how to compute the eight parameter 2D homography, which corresponds to the transformation between different views of a planar surface.

A homography $\mathbf{H}$ is represented by a 3×3 matrix, whose i 'th row is designated $\mathbf{H}_i$. $\mathbf{H}$ is normalized by setting $\mathbf{H}_{3,3} = 1$ leaving eight independent parameters. A 2D point $\mathbf{p} = (x, y, 1)^t$ (in homogeneous coordinates) located at image $\mathbf{I}_1$ is mapped by the homography $\mathbf{H}$ into the 2D point $\mathbf{p}'$ in image $\mathbf{I}_2$ as follows:

$$\mathbf{p}' = \left(\frac{(\mathbf{H}_1 \cdot \mathbf{p})}{(\mathbf{H}_3 \cdot \mathbf{p})}, \frac{(\mathbf{H}_2 \cdot \mathbf{p})}{(\mathbf{H}_3 \cdot \mathbf{p})}, 1 \right)^t \quad (4)$$

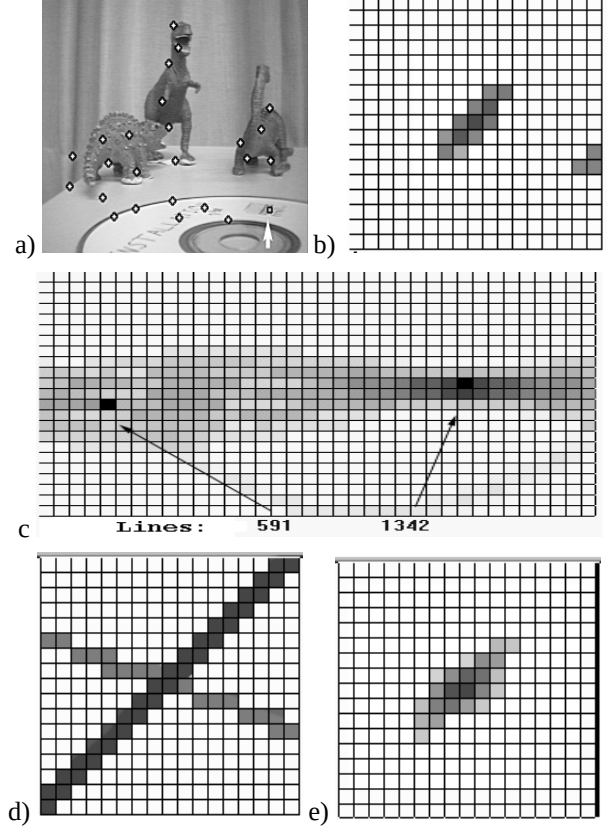


FIG. 2. Approximations for fuzzy correspondences. b) The fuzzy correspondence matrix between successive images for the point that is marked with a white arrow in (a). In this example the camera was not moving and the “ellipse” is located at the center of the correspondence matrix. c) Weighted Hough space of the correspondence matrix. Arrows point to the peak locations. Peak values (591, 1342) are used as weights. d) The lines used to approximate the fuzzy correspondence matrix. Intensity corresponds to weight. e) Weighted sum of city-block distance from the two lines in (c) is used as the approximation of the fuzzy correspondence matrix in (b).

The Euclidean distance of point $\mathbf{p}'$ from constraint line $\mathbf{s} = (Ax + By + C)$, using Eq. (4), is given by the following equation which is zero when the alignment is perfect.

$$d_H(\mathbf{p}', \mathbf{s}) = \left(\frac{A(\mathbf{H}_1 \cdot \mathbf{p})}{(\mathbf{H}_3 \cdot \mathbf{p})} + \frac{B(\mathbf{H}_2 \cdot \mathbf{p})}{(\mathbf{H}_3 \cdot \mathbf{p})} + C \right) \quad (5)$$

Multiplying Eq. (5) by $(\mathbf{H}_3 \cdot \mathbf{p})$ (which is non-zero for a finite size image) gives the following linear equation for the *residual error* of point $\mathbf{p}$:

$$\begin{aligned} r_H(\mathbf{p}', \mathbf{s}) &= d_H(\mathbf{p}', \mathbf{s})(\mathbf{H}_3 \cdot \mathbf{p}) \\ &= A(\mathbf{H}_1 \cdot \mathbf{p}) + B(\mathbf{H}_2 \cdot \mathbf{p}) + C(\mathbf{H}_3 \cdot \mathbf{p}) \end{aligned} \quad (6)$$

In order to get a linear equation, we multiplied the geometrical distance $d_H(\mathbf{p}', \mathbf{s})$ by the (unknown) value $(\mathbf{H}_3 \cdot \mathbf{p})$ resulting with an algebraic distance. The coordinates of $\mathbf{p}$ should be normalized to reduce bias [13, 12]. Setting the residual error $r_H(\mathbf{p}', \mathbf{s})$ to zero gives a linear constraint on the elements of the homography $\mathbf{H}$ that states that point $\mathbf{p}$ is mapped to point $\mathbf{p}'$ which is on the line $\mathbf{s}$.

In order to recover the eight parameters of the 2D homography $\mathbf{H}$, at least eight such point-to-line correspondences are required. Each point to two-lines correspondence (the linear approximation to fuzzy correspondence) gives one equation. When more than eight point-to-line correspondences are given $\mathbf{H}$ can be recovered by solving the following minimization problem:

$$\hat{\mathbf{H}} = \underset{\mathbf{H}}{\operatorname{argmin}} \sum_{i=1}^n w_i |r_H(\mathbf{p}'_i, \mathbf{s}_i)| \quad (7)$$

This error minimization problem, a minimization problems in the sense of L_1 (also called LAD for *least of absolute differences*) is converted into the following linear program, where one constraint equation is written for each point-to-line correspondence:

$$\begin{aligned} \min : & \sum_{i=1}^n w_i (z_i^+ + z_i^-) \\ \text{s.t.} & \\ & A_i(\mathbf{H}_1 \cdot \mathbf{p}_i) + B_i(\mathbf{H}_2 \cdot \mathbf{p}_i) + C_i(\mathbf{H}_3 \cdot \mathbf{p}_i) + (z_i^+ - z_i^-) = 0 \\ & z_i^+, z_i^- \geq 0 \end{aligned}$$

$(z_i^+ + z_i^-)$ is the absolute value of the residual error, $r_H(\mathbf{p}'_i, \mathbf{s}_i)$. This is done since only positive values are used in linear programming [18, 8]. Each residual error is represented by the difference of two non-negative variables: $r_H(\mathbf{p}'_i, \mathbf{s}_i) = z_i^+ - z_i^-$, one of which is always zero at the above minimum.

Notes:

1. When the constraints are of the form:
 $\mathbf{Ax} - \mathbf{b} + \mathbf{z} = 0$, a basic feasible solution that satisfies the constraints is given by $\mathbf{x} = 0, \mathbf{z} = \mathbf{b}$. This enables the use of an efficient one-phase simplex algorithm to solve the problem.
2. This linear program can be used to minimize any linear equation: $\operatorname{Min}(|\mathbf{Ax} - \mathbf{b}|)$. Parameter normalization or an additional constraint may be needed to avoid a zero root if $\mathbf{b} = 0$.
3. If $\mathbf{A}$ is an $(m \times n)$ matrix, m is the number of constraint equations and n is the number of parameters ($m > n$), then the linear program will have a total of $2(m + n)$ variables: $2n$ variables for the variable vector $\mathbf{x}$, and $2m$ variables for the slack variable vector $\mathbf{z}$. The factor of two is needed since each variable is represented by a difference of two non-negative variables.
4. The slack variable vector $\mathbf{z}$ contains the error measures for each point (the residuals).
5. Additional linear constraints can be added to the recovered model. For example we can define a motion model that is more general than similarity but has bounded affine/projective distortions.

4. EVALUATION

Three types of evaluation procedures were performed (i) testing the algorithm using real-image sequences. (ii) a comparison

between L_1 to other L_1 estimators and to the L_2 estimator. (iii) efficiency considerations. The tests were applied to different types of outliers and we show that L_1 provides a good compromise which is stable enough to cope with different categories of outliers and efficient enough to be used in real-time even for models with a large number of parameters. To balance the evaluation we also describe a scenario where L_1 fails completely. Section 5 discusses the results.

4.1. Image Registration and Mosaicing Experiments

Image registration was introduced in Section 3, mosaicing uses image registration to create large images by seamlessly stitching small images into a large image.

A major problem in image registration and mosaicing are outliers due to multiple models (either a complex static scene, or moving objects in the scene). One way of coping with this problem is *motion segmentation* which is the process of segmenting the image into different regions *according to their motion*. However motion segmentation is at least as difficult as motion recovery.

This section describes real image registration and mosaicing in the presence of outliers without motion segmentation. The motion analysis is done using point to line correspondences and linear programming LAD.

To compare our model to existing point-to-point methods, we converted each point-to-point correspondence to two point-to-line correspondences according to Section 2.4. The panorama example used point-to-line correspondences computed from fuzzy correspondence matrices.

4.1.1. Mosaicing with Similarity Model

A panoramic image was created in real-time (10 – 12 frames/seconds, image size of 320×240 pixels, motion limits by the correspondence matrix of ± 8 pixels) using a PC, as shown in Figure. 3.b. While the camera was scanning the scene, a large pendulum was swinging. The size of the pendulum was large enough to create about 15% outliers among the feature points. Since the stabilization algorithm used only frame to frame motion recovery, *any* error will cause the panorama to fail. Figure. 3 shows the pendulum (and its shadow) appearing/disappearing several times due to the swinging motion. However *all* frames were correctly aligned with a similarity model as can be seen by the objects that were not occluded by the pendulum.

4.1.2. Homographies: Comparison with L_2

This off-line experiment compares the computation of a 2D homography using L_1 registration to the least-squares method for point-to-point registration. Given two images, the feature points were selected automatically from a bi-directional optical-flow field. Each selected point had a strong intensity gradient, and the optical flow from the first to the second image agreed with the optical flow from the second to the first image. Selected points are shown in Figure. 4.a.



FIG. 3. Mosaicing examples. a) Point selected for motion computation. Four of the thirty points are located on the moving pendulum. b) Panoramic image that was created while a pendulum was swinging. The alignment was not affected by the outliers.

The alignment of the second image to the first image using the homography that minimizes the L_2 distance between the computed correspondences is shown in Figure. 4.d. It is completely useless due to outliers.

The L_1 alignment used the same data, but converted each point-to-point correspondence into two point-to-line correspondences (point (u, v) is converted to the two lines $(x = u)$ and $(y = v)$). Figure. 4.c shows the average of the two images after alignment. The alignment is now very good, and can be compared to Figure. 4.b where the two images were added before alignment.

4.1.3. Computing Affine Motion from Normal Flow

An affine alignment between two images can be computed from normal flow by an iterative method. In this example 112 points residing on strong edges and spread evenly across the image were selected automatically (off-line). The iterations were as follows:

1. The normal flow is computed from spatio-temporal derivatives and represented by a line constraint as described in Section. 2.3.
2. An affine motion model was computed using linear programming from the linear constraints.
3. The second image was warped toward the first image using the affine model.
4. Repeat steps 1-3 until convergence.

The iterations are necessary in this case since the accuracy of the normal flow depends on the accuracy of the spatio-temporal derivatives, which increases as the motion estimate becomes better. Figure. 5 shows the registration results for the L_1 registration by normal flow lines.

4.2. L_1 comparison to LMedS and the Tukey

M – estimator

M – estimators are not considered robust in the sense of breakdown point which is zero. [25, 11, 4]). L_1 is a particular M – estimator and therefore has a zero breakdown point as well. Moreover, even among M – estimators, L_1 is not considered the best one, other M – estimators such as the Tukey M – estimator have better resistance to outliers. The Least Median of Squares (LMedS, also called *LMS*) estimator is a well known robust estimator [21, 24] which has a breakdown point of 0.5, the highest

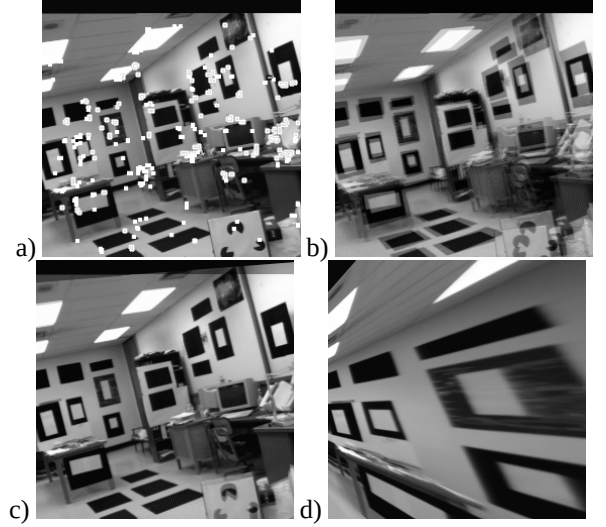


FIG. 4. Computing homography using L_2 registration compared to L_1 registration. The same feature points were used in the L_1 minimization and the L_2 minimization. Both examples are single iteration output, no segmentation and no re-weighting were used. a) Selected feature points are marked on one image. b) The average of the two original images to show their misalignment. c) The average of the images aligned with the homography obtained using linear programming. d) Warping the second image towards the first image with the homography obtained using a least-square method.

value possible. However, L_1 as a convex linear problem [8] has an advantage that the optimal solution can be found by a deterministic and efficient algorithm. The comparison is therefore between the following (estimator, algorithm) pairs:

(L_1 , *Simplex*) - The L_1 based estimator (LAD) is defined by:

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{x_i \in \mathbf{X}} |r_{i,\mathbf{a}}| \quad (8)$$

Where $\hat{\mathbf{a}}$ is the estimated model, $\mathbf{X}$ is the measurement matrix, $\mathbf{r}_{i,\mathbf{a}}$ is the residual error of measurements x_i with respect to the model $\mathbf{a}$. The algorithm used for minimization is the simplex algorithm. Note that this equation has no scale (variance) associated with it since the scale is not required by the simplex algorithm and it does not change the location of the global minimum.

(*TukeyM – estimator*, *IRLS*) - The Tukey M – estimator is defined by:

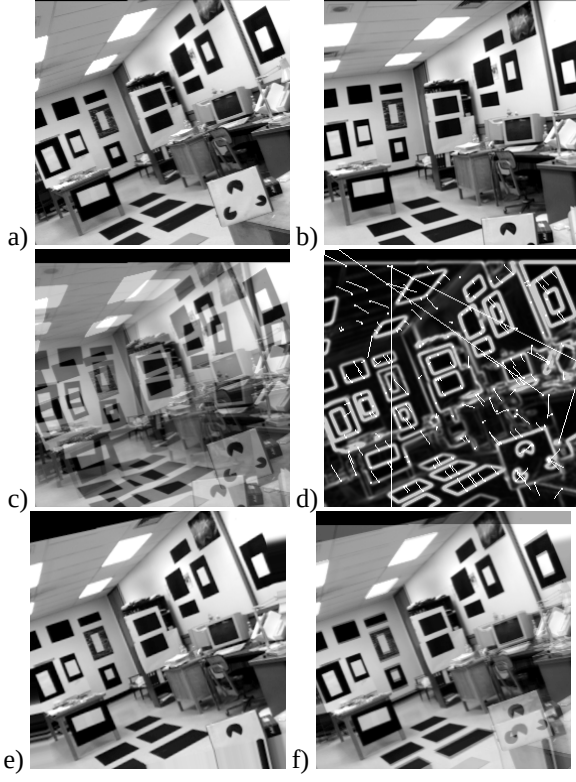


FIG. 5. Normal-flow point-to-line L_1 registration. a) First image. b) Second image. c) Averaging of (a) and (b) shows the displacement between the two images. d) Magnified normal flow of the selected points at the last iteration, displayed over a gradient map. The outliers are easy to spot. e) (b) warped towards (a) using the affine model computed by L_1 alignment. f) Averaging of (a) and (e) shows good alignment in most of the scene.

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{x_i \in \mathbf{X}} \rho(r_{i,\mathbf{a}}/\sigma_i) \quad (9)$$

$$\rho(u) = \begin{cases} \frac{B^2}{6} [1 - (1 - (\frac{u}{B})^2)^3] & |u| \leq B \\ \frac{B^2}{6} & |u| > B \end{cases}$$

Where $\rho(u)$ is the loss function, B is a tuning parameter and σ_i the scale associate with the value of $r_{i,\mathbf{a}}$.

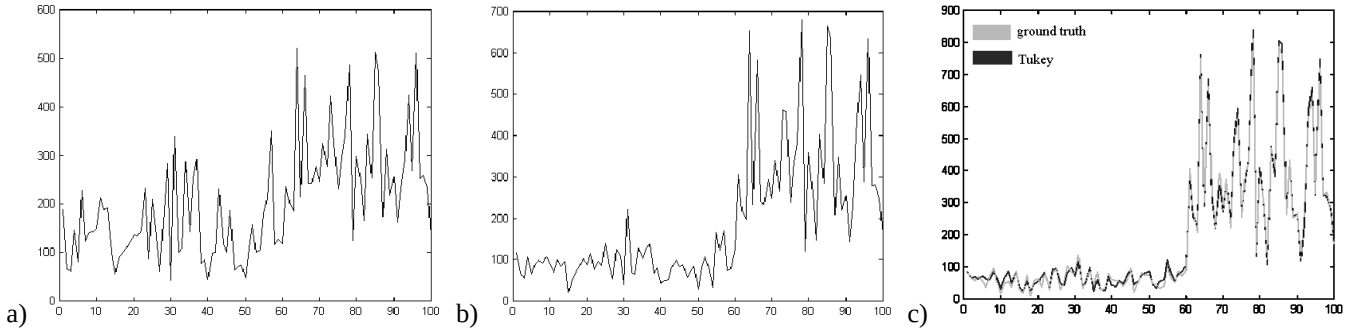


FIG. 6. Tukey M — estimator converges. a) Residuals of the initial guess (LSQ). b) Residuals after the first iteration. c) Residuals after the sixth iteration over the ground truth transformation residuals. The two graphs are almost identical.

Equation 9 is often solved by an iterative re-weighted least squares” (IRLS) [22] with the following weight function:

$$w(u = r_{i,\mathbf{a}}/\hat{\sigma}) = \psi(u)/u \quad (10)$$

$$\psi(u) = \rho(u)' = \begin{cases} u[1 - (\frac{u}{B})^2]^2 & |u| \leq B \\ 0 & |u| > B \end{cases}$$

Where $\hat{\sigma}$ is a scale estimate. The following scale estimation was used in the test

$$\hat{\sigma} = \sum_{i=1}^N \frac{w_i r_{i,\mathbf{a}}}{N} \quad (11)$$

Initially we set $w_i = 1$, $B = 4.8$ and we fine tuned σ to be $2\hat{\sigma} \dots 4\hat{\sigma}$. This fine tuning proved to be useful probably due to the aggressive outlier rejection of the Tukey M – estimator which rejects the “tail” of the inliers as well.

(LMedS, ProbabilisticAlgorithm) - The LMedS estimator is defined by:

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \operatorname{median}_{x_i \in \mathbf{X}} r_{i,\mathbf{a}}^2 \quad (12)$$

$\hat{\mathbf{a}}$ is computed using random sampling [10, 24, 26]. The algorithm, given that k data points are required to compute an model hypothesis $\hat{\mathbf{a}}^h$, then S k -points hypotheses are randomly selected from data set. The hypothesis $\hat{\mathbf{a}}^*$ with the smallest median is chosen as the estimate $\hat{\mathbf{a}}$.

4.2.1. Synthetic data Test configuration

The following three tests: “noise scenario test”, “spread-points bi-model scenario test” and the “bi-model shift error scenario test” were performed using synthetic data. The tests were general and were not specifically related to images. All three shared the same basic configuration:

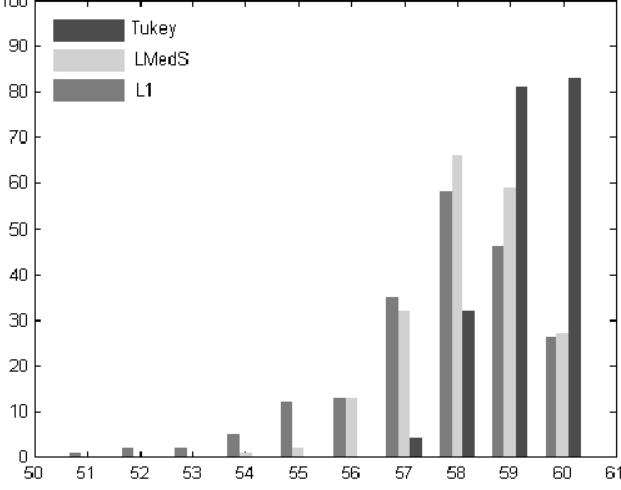


FIG. 7. Monte-Carlo test histogram for the noise scenario. The Tukey M – estimator produced the best results followed by LMedS which was closely followed by LAD.

$\mathbf{A} : R^4 \rightarrow R^4$ - the model to be estimated was a randomly selected linear transformation of sixteen parameters.

$\mathbf{X}$ - the measurements were a set of 100 source-target pairs of points in R^4 . The source points were randomly selected in the range $[-100 \dots 100]$. The inliers were the first 60 points which were transformed by $\mathbf{A}$, the rest 40 points were outliers and were set according to the specific test. Normal noise was added to the target points.

We evaluated the results using a *separation* criterion of simply counting the number of inliers that were included within the smallest ' n ' residuals, where n is the total number of inliers. The tests consists of 200 iterations, each time the number of inliers that were within the smallest 60 inliers was computed using a new set of $\mathbf{A}$ and $\mathbf{X}$ and the results were collected in a histogram of 60 bins.

4.2.2. Noise scenario

In this test the outliers consists of normal noise ($\bar{x} = 0, \sigma = 102$), inliers had an additive normal noise ($\bar{x} = 0, \sigma = 20$).

The resulting histogram is shown in Figure 7. We can see that for this scenario the Tukey M – estimator provided the best results, followed by LMedS which was closely followed by LAD.

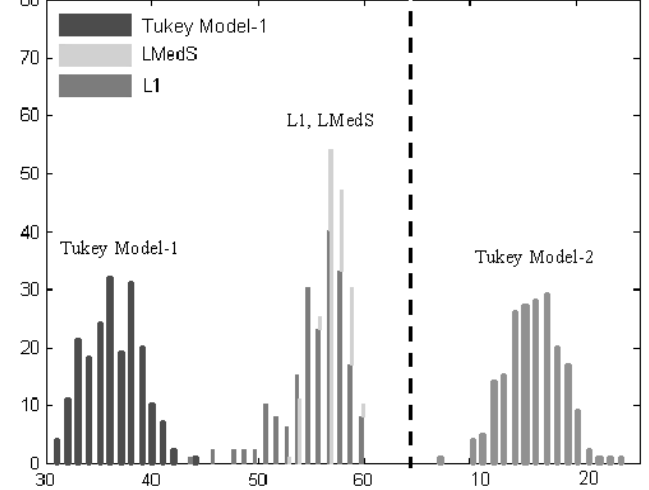


FIG. 8. Monte-Carlo test histogram for the bi-model scenario. Left: L_1 , LMedS and Tukey M – estimator histogram for model-1. Right: Tukey M – estimator histogram for model-2. A good separation would result in peaks around 60 at the first model and around 40 at the second model.

In many similar tests (with slightly different parameters) we found the same pattern - the Tukey M – estimator converges very fast (4-6 iterations) and gives the best estimate - provided that the scale is tuned and the initial guess is “good enough”. Figure 6 shows the initial guess (LSQ), the first and the sixth iteration of the TukeyM – estimator.

Experimental notes: (i) It is essential to have a good scale estimate, if the scale estimate is wrong the algorithm does not converge to the model. (ii) Wrong estimation, especially underestimation, can occur during iterations and spoil a previous good result.

4.2.3. Spread-Points Bi-Model scenario

In this test the outliers were transformed by a second linear transformation.

Both inliers and outliers had an additive normal noise ($\bar{x} = 0, \sigma = 20$).

The resulting histogram is shown in Figure 8. we can see that for this scenario the LMedS produced the best result, LAD closely followed it, and the Tukey M – estimator failed. The residuals of the LMedS, LAD and the Tukey M – estimator after

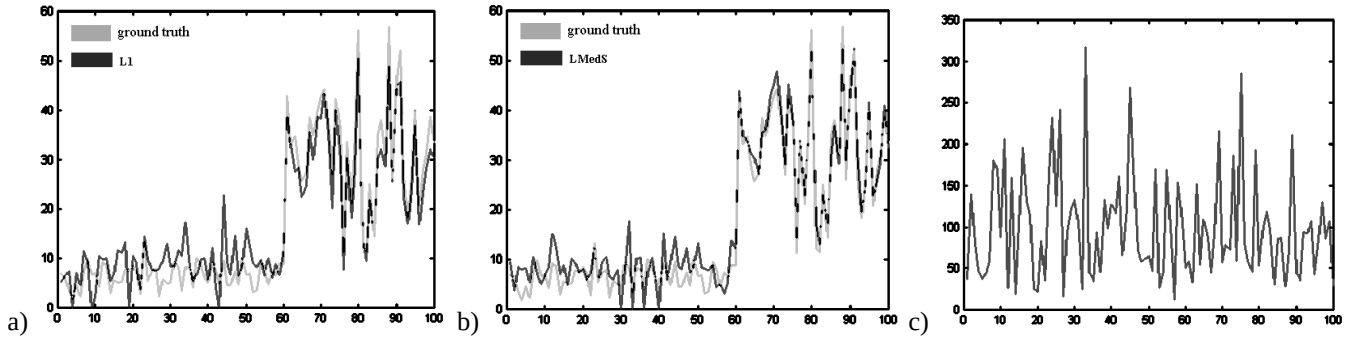


FIG. 9. Residuals for the bi-model scenario a) LAD residuals over the ground truth residuals. b) LMedS residuals over the ground truth residuals. c) Tukey M – estimator residuals after six iterations - failed to converge to any model.

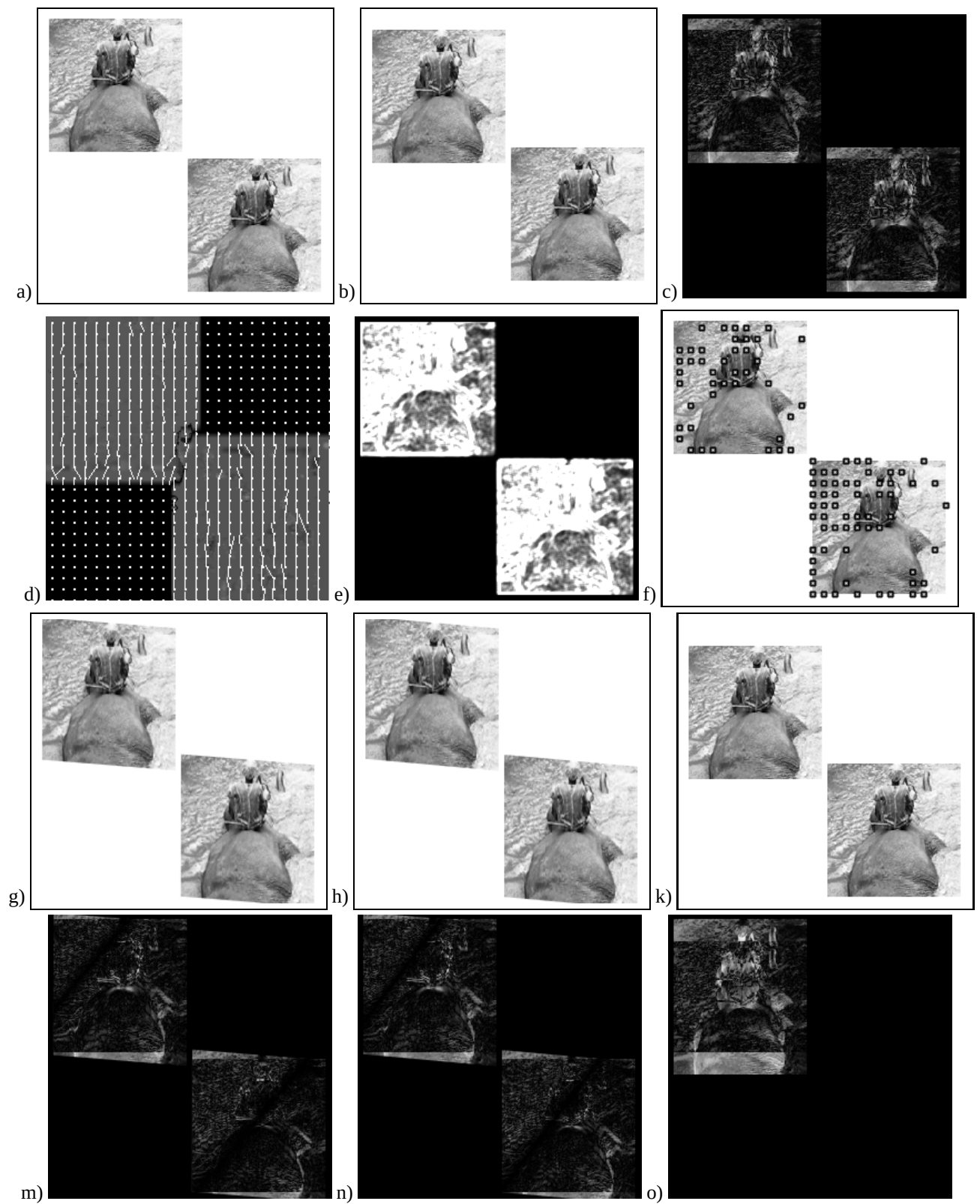


FIG. 10. Comparison between L_2 , optimal L_1 , and probabilistic LMedS affine image registration of the two block motion test case. a) First image (I_1). b) Second image (I_2). c) Difference before registration. d) Optical flow image. e) Reliability map. f) Point Selection. g) I_2 warped using L_2 recovered model. h) I_2 warped using L_1 recovered model. k) I_2 warped using LMedS recovered model. m) Error of the L_2 registration - failed to lock on one object. n) Error of the L_1 registration - failed to lock on one object. o) Error of the LMedS registration - succeeded.

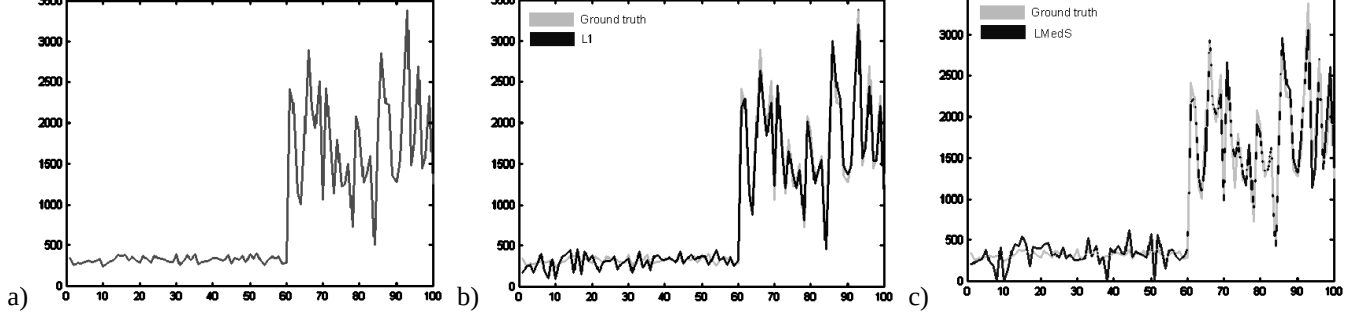


FIG. 11. Residuals for the shift+bi-model scenario a) Ground truth residuals. b) L_1 residuals over the ground truth residuals. c) LMedS (10,000 iterations) residuals over the ground truth residuals.

six iterations are shown in Figure 9. The Tukey M – estimator was initialized using LSQ and it couldn’t recover from the LSQ error. Having a two phase algorithm by using LMedS or LAD as an initializer may solve this problem.

4.2.4. Bi-Model, shift error scenario

The probabilistic LMedS solution is based on a K -point hypotheses, where K is usually the minimal number of points required to solve for $\hat{\mathbf{a}}$. What happens if none of the K -points hypotheses is close to $\hat{\mathbf{a}}$? For example, let $\mathbf{a}$ be the location of a single point in $\mathbf{R}^n$, $\mathbf{X}$ points with added noise vector $\mathbf{e}$ with a symmetric distribution of orientations but with magnitude greater than some positive radius R . $\mathbf{a}$ will be still the centroid of the measurements points therefore can be easily estimated by least-squares, however the best estimate the probabilistic LMedS can provide will be at least ‘ R ’ away from $\mathbf{a}$. This problem can be solved by increasing ‘ K ’ - however this will exponentially increase the computation time. In this test shift errors were added to the bi-model scenario to cause bad hypothesis. The residuals of the ground truth, LMedS, LAD are shown in Figure 11. We can see that the LAD produced a better estimate than the probabilistic LMedS however the LMedS still produced a good separation. Attempts to completely break down the LMedS resulted with a complete break down of LAD as well.

4.2.5. Close-points Bi-Model scenario

This scenario used a very simple, yet difficult synthetic scene that was specially designed to break down the LAD estimator (no dominant motion - see appendix). This scenario had two identical patches (from real images) sliding in opposite directions. This scenario is difficult in the sense that (i) both patches have identical texture and identical magnitude displacement (i-i) points are not spread across the image. About 100 “good” points were automatically selected, at about 42 : 58 ratio. “Good” points were points that were located on a strong texture using a gradient map (called reliability map in Figure 10) and the forward and backward optical flow agree. These points were fed into L_2 , L_1 and probabilistic LMedS solvers. The re-

sults are shown at Figure 10. We can see that the LMedS was the only estimator in the test that could *separate* the two motions.

4.3. Efficiency Considerations

This section discusses the time efficiency of IRLS, probabilistic LMedS and linear-programming. The IRLS algorithm used for M – estimators computation is very efficient and consists of few (3-10) least square, re-weight iterations.

4.3.1. Probabilistic LMedS time complexity

Given that the probability of choosing an outlier is q , then the probability p of having at least one perfect guess (no outliers in all k selected points) after t iteration is given by [10]:

$$p = 1 - [1 - (1 - q)^k]^t \quad (13)$$

Given the desired probability of success p , the number of necessary iterations t is given by

$$t = \ln(1 - p) / \ln(1 - (1 - q)^k) \quad (14)$$

The number of iterations required to reach a certain level of confidence is exponential in k and in q . Still the probabilistic LMedS is widely used in computer vision since the number of data points needed for obtaining a hypothesis is rather small, for example only 7 data points (triplets) are required for computing the 26 parameters (up to a scale factor) of the tri-linear tensor. There is, however, a hidden assumption here. The hidden assumption is that the data is dichotomic - if a point is not an outlier than *all* its coordinates are good. Clearly this assumption is not true due to several reasons including the aperture effect and camera and scene geometry. Breaking this dichotomy will be very expensive as k will become the number of parameters in the model. Another aspect of the probabilistic LMedS complexity is the required probability of success p . For example $p = 0.95$ is *not* considered good enough for video processing as it implies 1-2 bad frames every second (and more than 20 bad frames for even a simple mosaic which consists of several hundreds frames).

4.3.2. Linear-programming complexity

The complexity of the linear program is polynomial in the number of constraints, which equals the number of correspondences. In most practical cases, however, the complexity is

known to be nearly linear. During testing the number of pivot operation was approximately $n^{1.7}$ where n was the number of points.

4.3.3. Synthetic test performance comparison

In this test we tried to compare the actual performance of computing LAD using probabilistic algorithm and linear-programming. The test consisted of the following synthetic data:

Number of matched pairs: 100 point to point correspondences.

Rank of linear model: $k = 4$.

Outliers probability: $q = 0.4$ (three motion models, matched pairs are distributed as follows: 20, 60, 20).

Added Noise: Normal distribution with zero mean and variance is 5% of the range.

Even though the probabilistic algorithm was able to execute 7000 iterations during the time the single-iteration linear programming executed, the results obtained were inferior to linear programming as seen in Figure. A1.

4.3.4. Real-Time Performance

Programs for video mosaicing and for image stabilization were written based on fuzzy correspondences (Section 2.4). Execution was on a PC using windows NT with no special hardware. Image sequences were directly processed from the camera at 10-12 frames per second. The panorama in Figure 3 was created in real-time using this program. In order to find the time allocation within the program a profiler was used to measure the time of major program components. The profiling report is shown in Table A1, we can see that the linear programming solver used only 20% of the total time - less that the time required to warp the image using MMX technology.

5. SUMMARY

This paper presented a new approach to motion analysis by converting image measurements into point-to-line correspondences, and computing the motion model using LAD estimator computed by linear programming. The approach was robust enough and efficient enough to allow real-time mosaicing in the presence of outliers - a task that required correct alignment of all frames used to construct the mosaic. The paper compares the LAD estimator computed by linear-programming to the Tukey M - estimator computed by IRLS and to LMedS computed by the probabilistic algorithm. The LAD, linear programming approach was shown to be a good and stable compromise between efficiency and robustness. It performed very well in various scenarios while having a nearly linear time complexity. The comparison showed that:

IRLS, Tukey M - estimator - Provided superior results for noise only scenario, however it completely failed for bi-model scenario or when the noise scale estimator was not good enough.

LMedS, probabilistic LMedS - Never completely failed in tests, but behaved poorer then (LAD, LP) in a shift error sce-

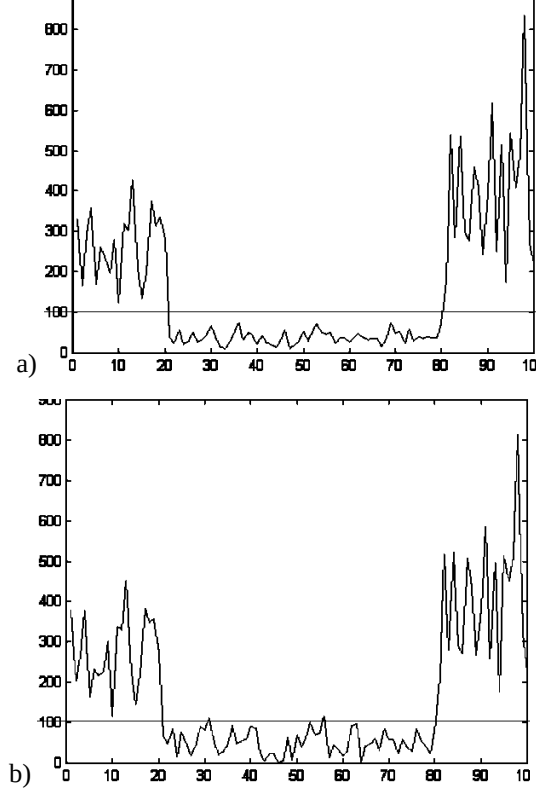


FIG. A1. Comparing performance of linear-programming and a probabilistic algorithm running for the same time. 60 of the 100 points are in the desired model ($q = 0.4$). a) Error plot for the linear programming solution. Clean separation is obtained. b) Error plot for the probabilistic LMedS solution. Separation between inliers and outliers is possible, but not for all points.

TABLE A1

Profiling results. 4156 frames were stabilized in 345.107 second = 12 Fps. The components were: Next-Frame: capturing the next frame time (using double buffer technique). Locate points: "Good" point selection for the point to line correspondence. Hough lines: Computing the correspondence matrix and find lines using Hough transform. Solve LP: Linear Programming solver. Warp: Image warping using MMX technology. Everything Else: Non profiled code including Operating System overhead.

Description	Total Time (sec)	Percentage	Average time (msec)
Next Frame	64.72	18.75%	15.56
Locate Points	3.12	0.90%	0.75
Hough Lines	76.52	22.17%	18.41
Solve LP	73.72	21.36%	17.74
Warp	78.85	22.85%	18.97
Display	14.32	4.15%	3.44
Everything Else	33.86	9.81%	1.36

nario, and less good then Tukey M - estimator in a noise only scenario. Is non deterministic and of exponential time complexity - but still very useful in vision due to the usually small number of guesses needed for hypothesis construction.

LAD, LP - Performed very close to LMedS. Failed when no dominant motion was present (See Appendix). Performed better than LMedS in the shift error scenario, and better than the

Tukey M-estimator in the 3D model scenario. The algorithm does not require an initial guess nor a noise scale estimator, it is deterministic, has a convex objective function (and hence a global minima is guaranteed), and has a nearly linear complexity. A good compromise in most practical scenarios.

A practical recommendation will be to use (LAD, LP) or a robust LMedS estimator if the complexity allows it - as an initial guess. Then iterate either by IRLS or by (LAD, LP) [5] to improve the accuracy of the result.

APPENDIX A

Dominance in Motion Analysis domain

M-estimators have a breakdown point of zero [4, 25]. Still as seen in the tests, LAD was able to resist 40% outliers quite similar to LMedS and the Tukey M-estimator and it was even better than LMedS in the noise outliers scenario. we try to explain this difference by looking at the relationship between *leverage points*, dominant motion, and the fact that images have **bounded region**. Leverage points are outliers that are far enough that even a single point can flip the recovered model very far from the real model (for the LAD estimator).

Such points are called leverage points. An example of a leverage point is shown in Figure. A1.a. Line *A* is the real model, points $q_1 \dots q_7$ are located on line *A*. Point *p* satisfies:

$$L_1(p, A) > \sum_{i=1}^n L_1(q_i, B) \quad (A.1)$$

where $L_1(a, b)$ represents the L_1 distance between *a* and *b*. This causes the model to flip into line *B*. Figure A1.b describes a very similar setup with points $q_1 \dots q_7$ spread along the line *A*. This time there is no single point in the bounded-rectangle that qualifies as a leverage point. There is no “room” for a “lever” long enough to flip the model. In this particular case the breakdown point of the L_1 metric is larger than zero - we then refer to line *A* as the “dominant line”. Figure A1c shows an “image” displacement map of five points. Points $q_1 \dots q_4$ belong to the motion model - in this case pure translation, while point *p* is a leverage point. Figure A1d show a 90° “solution” for the model recovery (shown in light-gray). Again, the error (measured from the base of the arrow to the head of the corresponding (gray) arrow) for the leverage point *p* becomes zero, while the sum of residual errors for the model points $q_1 \dots q_7$ is smaller than the original error of *p*. As in the previous example, if the points $q_1 \dots q_7$ were spread across the “image” as in Figure A1e, then within the bounds of the image there is no single displacement that can act as a lever (up to similarity model). and the *breakdown point* in this particular case is greater than zero.

The notion of *dominant region* or *dominant motion* forms a generalization. For example, line *A* in Figure A1.f is the dominant line since it has a lower minimum value than line *B* and **any superposition of the two model - for example line *C***. It is easy to see that the example satisfies these conditions (moving line *A* by ϵ in any direction and any orientation may reduce the error at most 3ϵ for $p_1 \dots p_3$ but will cost more than 3ϵ for

$q_1 \dots q_7$). As seen in the examples - for complex models such as Affine or Homography it is not easy to see if a dominant motion exist. A known strategy is to spread points as much as possible and not to over-parameterize the model.

In practice, the background of a scene usually forms a large model that is spread across the image, and thus is not subject to leverage points within the image boundaries.

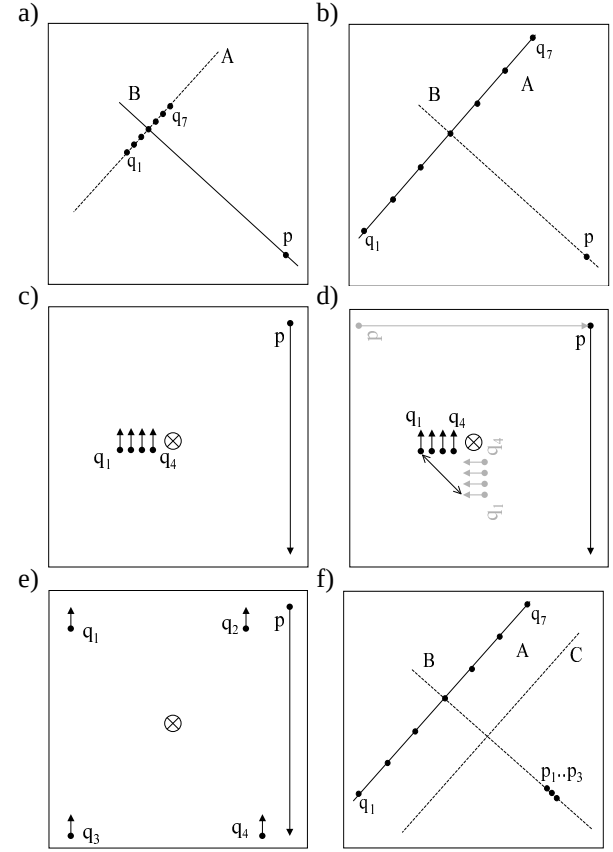


FIG. A1. Leverage Points for line and motion models. a) Point *p* is a leverage point that cause the model to switch to an incorrect model *B*. b) The points on line *A* are spread across the image. Point *p* cannot be far enough to switch the model and still be in the image boundaries. c,d) Point *p* is a leverage point that causes the model to switch to an incorrect rotation motion. e) Points $q_1 \dots q_4$ are spread across the image. Point *p* cannot move far enough to switch the model and still be in the image boundaries. f) Dominant line - line *A* is dominant since it has lower error then line *B* and any line *C* in between.

REFERENCES

1. G. Adiv. Determining 3-d motion and structure from optical flow generated by several moving objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(4):384–401, July 1985.
2. Y. Aloimonos and Z. Duric. Estimating the heading direction using normal flow. *Int. J. of Computer Vision*, 13(1):33–56, September 1994.
3. P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *Int. J. of Computer Vision*, 2:283–310, 1989.
4. A. Bab-Hadiashar and D. Suter. Optic flow calculation using robust statistics. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 988–993, 1997.

5. M. Ben-Ezra, S. Peleg, and M. Werman. Efficient computation of the most probable motion from fuzzy correspondences. In *IEEE Workshop on Applications of Computer Vision (WACV98)*, 1998.
6. J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conf. on Computer Vision*, pages 237–252, 1992.
7. M.J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow-fields. *Computer Vision and Image Understanding*, 63(1):75–104, January 1996.
8. Vašek Chvátal. *Linear Programming*. W.H. Freeman and CO., New York, 1983.
9. O.D. Faugeras, F. Lustman, and G. Toscani. Motion and structure from motion from point and line matching. In *Int. Conf. on Computer Vision*, pages 25–34, 1987.
10. M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.
11. Rousseeuw P.J. Hampel F.R., Ronchetti E.M. and Stahel W.A. *Robust Statistics: The Approach based on influence Functions*. New York: John Wiley, 1986.
12. R.I. Hartley. In defense of the eight-point algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997.
13. R.I. Hartley. Minimizing algebraic error in geometric estimation problems. In *Int. Conf. on Computer Vision*, pages 469–476, 1998.
14. B.K.P. Horn and B.G. Schunck. Determining optical flow. In *AI*, volume 17, pages 185–203, 1981.
15. T.S. Huang and A.N. Netravali. Motion and structure from feature correspondences: A review. *Proceedings of IEEE*, 82(2):252–268, February 1994.
16. M. Irani and P. Anandan. Robust multi-sensor image alignment. In *Int. Conf. on Computer Vision*, pages 959–966, 1998.
17. M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using region alignment. In *PAMI*, volume 19, pages 268–272, March 1997.
18. H. Karloff. *Linear Programming*. Birkhäuser Verlag, Basel, Switzerland, 1991.
19. Martin D. Levine. *Vision in Man and Machine*. McGraw-Hill, NY, USA, 1985.
20. B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. on Artificial Intelligence*, pages 674–679, 1981.
21. P. Meer, D. Mintz, and A. Rosenfeld. Analysis of the least median of squares estimator for computer vision applications. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 621–623, 1992.
22. Holland P.W. and Welsch R.E. Robust regression using iteratively reweighted least-squares. *Commun. Statist.-Theor. Meth*, A6:813–827, June 1977.
23. Y. Rosenberg and M. Werman. Representing local motion as a probability distribution matrix and object tracking. In *Image Understanding Workshop*, pages 153–158, 1997.
24. P.J. Rousseeuw. Least median of squares regression. *Journal of American Statistical Association*, 79:871–880, 1984.
25. P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection*. Wiley-Interscience, New York, 1987.
26. P.H.S. Torr and D.W. Murray. Outlier detection and motion segmentation. In *SPIE*, volume 2059, pages 432–443, 1993.