# EFFICIENT METHODS FOR POINT MATCHING WITH KNOWN CAMERA ORIENTATION

*João F. C. Mota, Pedro M. Q. Aguiar*

Institute for Systems and Robotics / IST, Lisboa, Portugal
{jmota,aguiar}@isr.ist.utl.pt

## ABSTRACT

The vast majority of methods that successfully recover 3D structure from 2D images hinge on a preliminary identification of corresponding feature points. When the images capture close views, *e.g.*, in a video sequence, corresponding points can be found by using local pattern matching methods. However, to better constrain the 3D inference problem, the views must be far apart, leading to challenging point matching problems. In the recent past, researchers have then dealt with the combinatorial explosion that arises when searching among $N!$ possible ways of matching $N$ points. In this paper we overcome this search by making use of prior knowledge that is available in many situations: the orientation of the camera. This knowledge enables us to derive $\mathcal{O}(N^2)$ algorithms to compute point correspondences. We prove that our approach computes the correct solution when dealing with noiseless data and derive an heuristic that results robust to the measurement noise and the uncertainty in prior knowledge. Although we model the camera using orthography, our experiments illustrate that our method is able to deal with violations, including the perspective effects of general real images.

***Index Terms*—** Stereo vision, Image matching, Image registration, Permutations, Image sequence analysis

## 1. INTRODUCTION

Methods that infer three-dimensional (3D) information about the world from two-dimensional (2D) projections, available as ordinary images, find applications in several fields, *e.g.*, digital video, virtual reality, and robotics, motivating the attention of the image analysis community. Using single image brightness cues, such as shading and defocus, researchers have proposed methods that work in highly controlled environments, like laboratories, but result sensitive to the noise and unable to deal with more general scenarios. As a result, the effort of the past decades was mainly on the exploitation of a much stronger cue: the motion of the brightness pattern between images. In fact, the image projections of objects at different depths move differently, unambiguously capturing the 3D shape of the scene. This lead to the so-called 3D Structure-from-Motion (SfM) methods.

SfM splits the problem into two separate steps: **i)** 2D motion estimation, from the images; **ii)** inference of 3D structure (3D motion of the camera and 3D shape of the scene), from 2D motion. Usually, the 3D shape of the scene is represented in a sparse way, by a set of pointwise features, thus the 2D motion is represented by the corresponding set of trajectories of image point projections. When dealing with video sequences, consecutive images correspond to close views, and those trajectories can be obtained through tracking, *i.e.*, by using *local* motion estimation techniques. However, since very distinct viewpoints are required to better constrain the 3D inference problem, in many situations, there is the need to process a single pair of distant views. In this scenario, the 2D motion estimation step **i)**, *i.e.*, the problem of matching pointwise features across views, becomes very hard and, in fact, the bottleneck of SfM (step **ii)** has been extensively studied and efficient methods are available [1]).

Researchers have then addressed the problem of computing point correspondences in a *global* way, by incorporating the knowledge that the feature points belong to a 3D rigid object. However, the space of correspondences to search grows extremely fast: considering $N$ feature points, there exist $N!$ ways to match them. Due to this combinatorial explosion, only sub-optimal methods have been proposed to solve the problem, see, *e.g.*, [2], for an iterative approach that strongly depends on the initialization. Curiously, in the simpler scenario of dealing with noisy observations of *geometrically equal* point clouds, the optimal solution can be efficiently obtained as the solution of a convex problem [3]. The challenge in SfM is that the point clouds from which we must infer the correspondences have *distinct shape* because they are different 2D projections of the (unknown) 3D shape.

In this paper, we overcome the difficulty pointed out in the previous paragraph by using as prior knowledge the orientation of the camera. In fact, in many situations, that knowledge is available from camera calibration or can be computed without using feature points and their correspondences. For example, in scenarios where many edges are aligned with three orthogonal directions, *e.g.*, indoor or outdoor urban scenes, the orientation of the camera can be reliably obtained from the vanishing lines of a single image, see, *e.g.*, [1], or even directly from the statistics of the image intensities [4]. We show how the knowledge of camera orientation simplifies the problem, enabling us to derive an algorithm of complexity $\mathcal{O}(N^2)$. We prove that this algorithm computes the optimal set of correspondences for the orthographic camera projection model in a noiseless scenario and propose a modified version that results robust to uncertain measurements and violations of orthography.

## 2. PROBLEM FORMULATION

Consider the scenario of Fig 1, where two cameras $C_1$ and $C_2$ (or, equivalently, the same camera in two different positions) capture two different views of the world. As usual when recovering SfM, we assume that a set of $N$ feature points was extracted from each of the images, and their coordinates in the image plane are represented by

$$I_1 := \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_N^{(1)} \\ y_1^{(1)} & y_2^{(1)} & \cdots & y_N^{(1)} \end{bmatrix}, \quad I_2 := \begin{bmatrix} x_1^{(2)} & x_2^{(2)} & \cdots & x_N^{(2)} \\ y_1^{(2)} & y_2^{(2)} & \cdots & y_N^{(2)} \end{bmatrix}, \quad (1)$$

where the superscript $(i)$ indexes the points to $C_i$, for $i = 1, 2$. Each feature point has 3D coordinates $(X_n, Y_n, Z_n)$, with respect to some fixed coordinate frame. Let that frame be attached to $C_1$ such that:

Partially supported by FCT, under ISR/IST plurianual funding (POSC program, FEDER) and grant MODI-PTDC/EEA-ACR/72201/2006.

**1)** the axes $X$ and $Y$ are parallel to the axes $x'$ and $y'$ of the camera frame; **2)** the optical center of the camera $C_1$ is aligned with the axis $Z$ (see Fig. 1). The major challenge when attempting to recover $\{(X_n, Y_n, Z_n), n = 1, \ldots, N\}$ from $I_1$ and $I_2$ is the correspondence problem. In fact, we do not know the pairwise correspondences between the columns of $I_1$ and $I_2$ in (1) because there is not a "natural" way to automatically order the feature point projections. Although estimating this ordering leads to a combinatorial problem whose solution, in general, becomes a quagmire for large $N$, we show in this paper that, when the relative orientation of the cameras is known and the perspective projection is well approximated by the orthographic projection model, an efficient solution can be found.

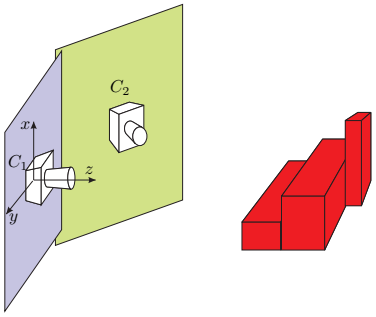


**Fig. 1**. Our scenario, with a choice for the reference frame.

Consider the orthographic model of a camera [1]: $\mathbf{x} = P\mathbf{X}$, where $\mathbf{X} \in \mathbb{P}^3$ and $\mathbf{x} \in \mathbb{P}^2$ are, respectively, the homogeneous coordinates of the points in space and in the image plane. The matrix $P \in \mathbb{R}^{3\times4}$ is given by

$$P = \begin{bmatrix} R & t \\ 0_3^T & 1 \end{bmatrix}, \tag{2}$$

where $R \in \mathbb{R}^{2\times3}$ contains the first two rows of a 3D rotation matrix, $t \in \mathbb{R}^2$ is a translation vector and $0_3$ is the zero vector in $\mathbb{R}^3$. With the choice of reference frame of the previous paragraph, it is straightforward to see that camera $C_1$ captures the first two coordinates of the feature points, *i.e.*, that $(x_n^{(1)}, y_n^{(1)}) = (X_n, Y_n), n = 1, \ldots, N$. Naturally, camera $C_2$ captures projections that depend on the relative position of the cameras, the 3D coordinates of the points, and their correspondences:

$$\begin{bmatrix} I_2 \\ 1_N^T \end{bmatrix} = \begin{bmatrix} R & t \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} X_1 & X_2 & \cdots & X_N \\ Y_1 & Y_2 & \cdots & Y_N \\ Z_1 & Z_2 & \cdots & Z_N \\ 1 & 1 & \cdots & 1 \end{bmatrix} \Pi, \tag{3}$$

where $1_N \in \mathbb{R}^N$ has all its entries equal to 1, and $\Pi \in \mathbb{R}^{N\times N}$ is a permutation matrix, *i.e.*, a matrix with exactly one entry equal to 1 per row and per column and the remaining entries equal to 0 (when we multiply a matrix $M$ by $\Pi$, we get a matrix with the same columns of $M$ but arranged in a possibly different order).

By using (3), we obtain the model relating the projections of the feature points in images $I_1$ and $I_2$ with all the unknowns:

$$I_2 = \left[\hat{R}I_1 + \hat{r}Z^T + t1_N^T\right]\Pi, \tag{4}$$

where $Z = [Z_1, Z_2, \ldots, Z_N]^T$ and $R$ was decomposed as $R = [\hat{R}, \hat{r}]$, with $\hat{R} \in \mathbb{R}^{2\times2}$ and $\hat{r} \in \mathbb{R}^{2\times1}$. When the relative orientation of the cameras is known (which, as discussed in the previous section, occurs in several practical situations), *i.e.*, when $\hat{R}$ and $\hat{r}$ are known, the problem becomes to find a permutation matrix $\Pi$, a set of 3D point depths $\{Z_1, \ldots, Z_N\}$, and a translation vector $t$ that solve (4). In general, the problem is hard due to the huge cardinality of the set of all $N \times N$ permutation matrices: $N!$.

## 3. CLOSED-FORM SOLUTION FOR TRANSLATION

The choice of the reference frame in the previous section leaves one degree of freedom: we can place the frame at any point along the axis $Z$. We now choose this position in such a way that the problem is simplified: let it be such that $\sum_{n=1}^N Z_n = 1_N^T Z = 0$, *i.e.*, that the plane $XY$ contains the center of mass of the feature points.

Multiplying both sides of (4) by $1_N$ and simplifying, we get

$$I_2 1_N = \left[\hat{R}I_1 + \hat{r}Z^T + t1_N^T\right]1_N \tag{5}$$

$$= \hat{R}I_1 1_N + Nt. \tag{6}$$

where (5) uses the fact that $\Pi\, 1_N = 1_N$ (permutation of a vector with all equal entries) and (6) uses equalities $Z^T 1_N = 0$ (from the choice of reference frame) and $1_N^T 1_N = N$. From (6), we see that the solution for the translation vector $t$ does not depend on the remaining unknowns ($\Pi$, $Z$):

$$t = \frac{1}{N}\left(I_2 - \hat{R}I_1\right)1_N. \tag{7}$$

By removing the (now known) translation from the problem, *i.e.*, by replacing the solution (7) in (4) (and using $1_N^T\,\Pi = 1_N^T$), we get

$$I_2 = \left[\hat{R}I_1 + \hat{r}Z^T\right]\Pi + \frac{1}{N}\left(I_2 - \hat{R}I_1\right)1_N 1_N^T. \tag{8}$$

To simplify notation, we re-define our observations by introducing matrices $\tilde{I}_1$ and $\tilde{I}_2$, both computed from known data:

$$\tilde{I}_2 := I_2 - \frac{1}{N}\left(I_2 - \hat{R}I_1\right)1_N 1_N^T, \qquad \tilde{I}_1 := \hat{R}I_1. \tag{9}$$

With these definitions, problem (4) is re-written as

$$\tilde{I}_2 = \left[\tilde{I}_1 + \hat{r}Z^T\right]\Pi, \tag{10}$$

where the unknowns are the depths $Z_1, \ldots, Z_N$, in $Z$, and the correspondences, coded by $\Pi$.

## 4. OPTIMAL SOLUTION FOR NOISELESS DATA

We first present an efficient algorithm to compute the solution to our problem when there is no noise, meaning that there exists at least one pair $(Z, \Pi)$ that solves (10).

Naturally, the solution for the permutation matrix $\Pi$ is given by the association of each column of $\tilde{I}_1$ with a column of $\tilde{I}_2$, for the correct value of $Z$. Let column $n$ of $\tilde{I}_1$ (resp. $\tilde{I}_2$) be represented by $[\tilde{X}_n, \tilde{Y}_n]^T$ (resp. $[\tilde{x}_n, \tilde{y}_n]^T$) and consider the error $E_{ij}$ of associating column $j$ of $\tilde{I}_1$ with column $i$ of $\tilde{I}_2$, *i.e.*,

$$E_{ij} = \min_{Z_j}\left[\tilde{x}_i - \tilde{X}_j - \hat{r}_1 Z_j\right]^2 + \left[\tilde{y}_i - \tilde{Y}_j - \hat{r}_2 Z_j\right]^2, \tag{11}$$

where $\hat{r} = [\hat{r}_1, \hat{r}_2]^T$. The minimizer $Z_j^*$ solving (11) is straightforwardly obtained in closed-form:

$$Z_j^* = \frac{\hat{r}_1(\tilde{x}_i - \tilde{X}_j) + \hat{r}_2(\tilde{y}_i - \tilde{Y}_j)}{\|\hat{r}\|^2}. \tag{12}$$

Our algorithm, detailed and analyzed in the sequel, computes for each column $i$ of $\tilde{I}_2$, the column $j^*$ of $\tilde{I}_1$ that minimizes error $E_{ij^*}$ (11) (without noise, for each $i$ there exists at least one $j^*$ such that $E_{ij^*} = 0$). In the algorithm description below, the $N \times N$ permutation matrix $\Pi$ is simply parameterized by a $N \times 1$ vector $\mathrm{perm}$: the $j$th column of $\Pi$ has entry $\mathrm{perm}_j$ equal to 1 (and, obviously, the others equal to zero); also, $|S|$ denotes the cardinality of set $S$ and $S_1 \backslash S_2$ the set of elements of $S_1$ that do not belong to $S_2$.

**Algorithm 1.**

**Inputs** *Matrices $\tilde{I}_1$ and $\tilde{I}_2$, organized into the corresponding sets of columns $\mathcal{B}_1 = \{[\tilde{X}_1, \tilde{Y}_1]^T, \ldots, [\tilde{X}_N, \tilde{Y}_N]^T\}$ and $\mathcal{A} = \{[\tilde{x}_1, \tilde{y}_1]^T, \ldots, [\tilde{x}_N, \tilde{y}_N]^T\}$, and vector $\hat{r}$.*

**Procedure** *For $i = 1, \ldots, N$ ($N = |\mathcal{A}|$)*

- *For all $j = 1, \ldots, |\mathcal{B}_i|$, compute $Z_j^*$ (12) and $E_{ij}$ (11);*
- $j^* = \arg\min_j E_{ij}$;
- $\mathrm{perm}_{j^*} = i$, $Z_{j^*} = Z_{j^*}^*$;
- $\mathcal{B}_{i+1} = \mathcal{B}_i \backslash [\tilde{X}_{j^*}, \tilde{Y}_{j^*}]^T$.

**Outputs** *Vectors* $\mathrm{perm}$ *and* $Z$.

Algorithm 1 consists of $N$ loops where, in each loop, a column of $\tilde{I}_2$ is assigned to a column of $\tilde{I}_1$. Each assignment requires a search over, at most, $N$ possibilities. It is then clear that our algorithm has complexity of $\mathcal{O}(N^2)$, in particular, we obtain the total number of floating point operations (flops) as $7N^2 + 7N - 14$. Before proving optimality of Algorithm 1, we interpret it in a geometric way. Defining the "displacement" as $a_{ij} := [\tilde{x}_i - \tilde{X}_j, \tilde{y}_i - \tilde{Y}_j]^T$, the cost to be minimized in (11) can be rewritten as $\|a_{ij} - Z_j \hat{r}\|^2$. So, for each column $[\tilde{x}_i, \tilde{y}_i]^T$ of $\tilde{I}_2$, our algorithm searches the column $[\tilde{X}_j, \tilde{Y}_j]^T$ of $\tilde{I}_1$ that minimizes $\|a_{ij} - Z_j \hat{r}\|^2$ for all possible values of $Z_j$. Since this expression achieves its minimum (zero) when $a_{ij}$ is collinear with $\hat{r}$ (which we synthetically denote by $a_{ij} // \hat{r}$), Algorithm 1 assigns pairs of columns such that their difference is "as parallel as possible" to $\hat{r}$. This collinearity is a re-statement of the fact that the trajectories of image projections of a rigid scene can be represented in a rank 1 matrix [5].

**Theorem 1** (Optimality of Algorithm 1)**.** *If there exists at least one pair $(Z, \Pi)$, such that* (10) *holds, then the outputs of Algorithm 1 determine a pair $(\bar{Z}, \bar{\Pi})$ that that solves* (10)*.*

*Proof.* Suppose the pair $(Z^*, \Pi^*)$ is such that (10) holds. For each $i = 1, \ldots, N$, there exists one and only one $k$ such that

$$\Pi_{ki}^* = 1 \tag{13}$$

(because $\Pi^*$ is a permutation matrix). We now denote by $j^*(i)$ the assignment produced by Algorithm 1, *i.e.*, we make explicit the dependence of $j^*$ on $i$. Obviously, if $j^*(i) = k$ for all $i = 1, \ldots, N$, then the algorithm returned an optimal solution. So, for the remaining of the proof, we assume there is an index $i$ such that $j^*(i) \neq k$. We will see that, even in this case, (10) holds for the solution provided by the algorithm, because $E_{ij^*(i)} = 0$, for all $i$.

A simple way to complete the proof is using contradiction. Assume $i$ is the smallest index such that $E_{ij^*(i)} > 0$ (obviously $j^*(i) \neq k$). If $E_{ij^*(i)} > 0$, then $[\tilde{X}_k, \tilde{Y}_k]^T \notin \mathcal{B}_i$ (at the $i$th loop). Thus, there exists $1 \leq l < i$ such that $[\tilde{x}_l, \tilde{y}_l]^T // [\tilde{X}_k, \tilde{Y}_k]^T$ (because $E_{lj^*(l)} = 0$ for all $1 \leq l < i$). According to the assignment defined by (13), we have $[\tilde{X}_k, \tilde{Y}_k]^T // [\tilde{x}_i, \tilde{y}_i]^T$, thus $[\tilde{x}_l, \tilde{y}_l]^T // [\tilde{x}_i, \tilde{y}_i]^T$. Also, since $\Pi^*$ is a permutation matrix, there exists an index $m = 1, \ldots, N$, such that $\Pi_{ml}^* = 1$, or, equivalently, such that $[\tilde{X}_m, \tilde{Y}_m]^T // [\tilde{x}_l, \tilde{y}_l]^T$, thus, $[\tilde{X}_m, \tilde{Y}_m]^T // [\tilde{x}_i, \tilde{y}_i]^T$. We now consider two cases: **1)** if $[\tilde{X}_m, \tilde{Y}_m]^T \in \mathcal{B}_i$, there is a contradiction because $E_{im} = 0$; **2)** if $[\tilde{X}_m, \tilde{Y}_m]^T \notin \mathcal{B}_i$, it is straightforward to find a vector $[\tilde{X}_{m'}, \tilde{Y}_{m'}]^T \in \mathcal{B}_i$ such that $[\tilde{X}_{m'}, \tilde{Y}_{m'}]^T // [\tilde{x}_i, \tilde{y}_i]^T$, by performing steps like the ones above, which brings us back to case **1)**. □

## 5. APPROXIMATE SOLUTION FOR NOISY DATA

In practice, not only the knowledge of the camera orientation is uncertain but also the feature point projections are noisy. Since Algorithm 1 is based on the collinearity of a vector that depends on the camera orientation ($\hat{r}$) with vectors that depend on the feature point projections ($[\tilde{x}_i - \tilde{X}_j, \tilde{y}_i - \tilde{Y}_j]^T$), its behavior is sensitive to disturbances affecting these vectors. We now propose a modification of this algorithm, which results robust not only to the noise but also to violations of the orthogaphic projection model.

From model (10) we note that the clouds of points in $\tilde{I}_1$ and $\tilde{I}_2$ differ by $\hat{r} Z^T$. Since $\hat{r}$ contains entries of a rotation matrix, *i.e.*, entries with magnitude smaller than 1, in practice, the patterns of points in $\tilde{I}_1$ and $\tilde{I}_2$ will almost coincide when the depth of the scene is not too large (more rigorously, when $\hat{r} Z^T$ is negligible if compared to the minimum distance between points), even if the corresponding points in $I_1$ and $I_2$ are very distant (see an insightful example in Fig. 4). This motivated us use the matching criterion of minimizing the Euclidean distance between points in $\tilde{I}_1$ and $\tilde{I}_2$,

$$E'_{ij} = \left\| \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} X_j \\ Y_j \end{bmatrix} \right\|^2, \tag{14}$$

rather than the less robust collinearity (11).

**Algorithm 2.**

**Inputs** *Matrices $\tilde{I}_1$ and $\tilde{I}_2$, organized into the corresponding sets of columns $\mathcal{B}_1 = \{[\tilde{X}_1, \tilde{Y}_1]^T, \ldots, [\tilde{X}_N, \tilde{Y}_N]^T\}$ and $\mathcal{A} = \{[\tilde{x}_1, \tilde{y}_1]^T, \ldots, [\tilde{x}_N, \tilde{y}_N]^T\}$, and vector $\hat{r}$.*

**Procedure** *For $i = 1, \ldots, N$ ($N = |\mathcal{A}|$)*

- *For all $j = 1, \ldots, |\mathcal{B}_i|$, compute $E'_{ij}$ (14);*
- $j^* = \arg\min_j E'_{ij}$;
- $\mathrm{perm}_{j^*} = i$, $Z_{j^*} = Z_{j^*}^*$ (12);
- $\mathcal{B}_{i+1} = \mathcal{B}_i \backslash [\tilde{X}_{j^*}, \tilde{Y}_{j^*}]^T$.

**Outputs** *Vectors* $\mathrm{perm}$ *and* $Z$.

Our experiments, some of them singled out in the following section, demonstrate that Algorithm 2 successfully infers correct feature point correspondences when dealing with real images. In spite of correctly determining correspondences, the accuracy of the depth estimates in $Z$ strongly depend on the magnitude of the components of $\hat{r}$. In fact, assuming the correspondences are known, for example, $\Pi = I_{N \times N}$ (for simplicity), model (10) becomes $\tilde{I}_2 - \tilde{I}_1 = \hat{r} Z^T$, making clear that the accuracy in the estimation of $Z$ depends not only on the accuracy of the measurements $(\tilde{I}_1, \tilde{I}_2, \hat{r})$ but also on the magnitude of the components of $\hat{r}$. In particular, we obtain an upper-bound for the depth estimation error as $\rho_Z = \max |\tilde{I}_2 - \tilde{I}_1| / \min |\hat{r}|$. Naturally, when $\rho_Z$ is large, we can still use our algorithm to estimate the correspondences between the feature points (in general, the bottleneck of the problem), which are not affected by $\rho_Z$, and then use a standard algorithm to recover SfM from known correspondences, eventually using a larger set of images to reduce ambiguity, see, *e.g.*, [1].

## 6. EXPERIMENTS

To test the algorithms with ground truth, we synthesized data. In particular, we generated the 3D world as a set of 50 points randomly distributed in $[-200, 200]^3$ and relative orientations between the cameras by specifying random rotation matrices. Then, we synthesized measurements according to the model in expression (3), for random permutation matrices. As expected, according to our theoretical derivation of Section 4, Algorithm 1 always produced the correct result: it successfully recovered the permutation, *i.e.*, the correct correspondences between the points, and their depth. To test robustness to disturbances, we then run experiments by considering inaccurate knowledge of camera orientation and noisy feature point projections. As anticipated in Section 5, we observed that Algorithm 2 results more robust than Algorithm 1. The plot in Fig. 2 illustrates this point by showing the average number of wrong correspondences as functions of the (white Gaussian) measurement noise standard deviation (st.dv.). Note that, even for noise st.dv. of 5 pixels, Algorithm 2 almost always recovers totally correct correspondences. In what respects to depth estimation accuracy, the magnitudes of the errors were smaller than the magnitudes of the measurement noise.

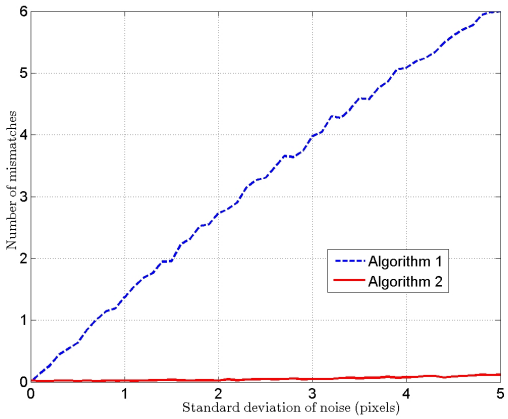


**Fig. 2**. Number of incorrect correspondences as functions of the observation noise power (mean over 1000 runs).

We tested our algorithms with real images. Two examples are shown in Fig. 3, which contains the two pairs of images with feature points superimposed. Note that, in both examples, corresponding features are far from being close to each other, preventing thus the usage of "local" methods. We used standard calibration techniques to compute camera orientation [6] and then run our algorithms. The plots in Fig. 4 provide insight over our approach: while the feature point projections of corresponding features in $I_1$ and $I_2$ are in general far apart, their "version" in $\tilde{I}_1$ and $\tilde{I}_2$ are close. As a consequence, Algorithm 2 recovered the correct correspondences in both cases. We emphasize that these examples strongly depart from the assumed orthographic projection, see the perspective effects between the pairs of images in Fig. 3, thus, that our approach is able to deal with a wide range of real life scenarios.

## 7. CONCLUSION

We proposed efficient algorithms for finding, simultaneously, the correspondences between points in two images and their depth in the 3D world. Our approach is based on the facts that, in many situations, the relative orientation of the cameras is available, or can be easily inferred, and the camera model can be approximated by an orthographic projection. The resulting complexity is $\mathcal{O}(N^2)$, where $N$ is the number of feature points (compare with $N!$, the number of possible correspondences). We prove the optimality of a first algorithm when dealing with noiseless data and develop a modified version that results more robust to uncertainty in the measurements.

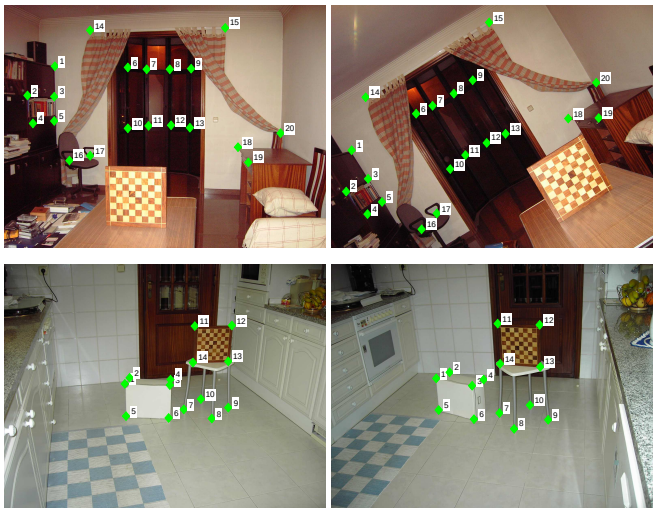

**Fig. 3**. Two pairs of real images with feature points superimposed.

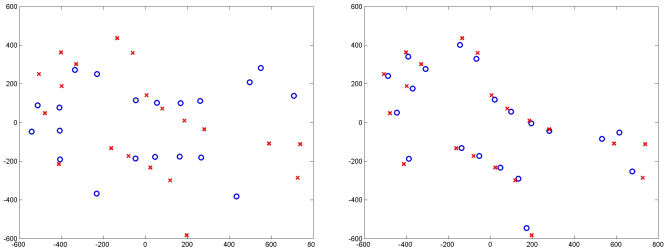

**Fig. 4**. Left: feature point coordinates in $I_1$ and $I_2$, extracted from the pair of images in the top of Fig. 3 (the blue circles are from the left image and the red crosses from the right one). Right: corresponding entries of $\tilde{I}_1$ and $\tilde{I}_2$, computed from known data, see (9).

## 8. REFERENCES


[1] R. Hartley and A. Zisserman, *Multiple View Geometry In Computer Vision*, Cambridge University Press, 2003.

[2] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun, "Structure from motion without correspondence," in *IEEE CVPR*, 2000.

[3] P. Shivaswamy and T. Jebara, "Permutation invariant SVMs," in *Int. Conf. on Machine Learning*, Pittsburgh PA, USA, 2006.

[4] A. Martins, P. Aguiar, and M. Figueiredo, "Orientation in Manhattan," *IEEE Trans. on PAMI*, vol. 27, no. 5, 2005.

[5] P. Aguiar and J. Moura, "Rank 1 weighted factorization for 3D structure recovery," *IEEE Trans. on PAMI*, vol. 25, no. 9, 2003.

[6] J. Bouguet, "Camera calibration toolbox for Matlab," `www.vision.caltech.edu/bouguetj/calib_doc`, 2008.