

# Accurate Non-Iterative $O(n)$ Solution to the PnP Problem<sup>\*</sup>

Francesc Moreno-Noguer

Vincent Lepetit

Pascal Fua

Computer Vision Laboratory

École Polytechnique Fédérale de Lausanne – 1015 Lausanne, Switzerland

{francesc.moreno,vincent.lepetit,pascal.fua}@epfl.ch

## Abstract

*We propose a non-iterative solution to the PnP problem—the estimation of the pose of a calibrated camera from  $n$  3D-to-2D point correspondences—whose computational complexity grows linearly with  $n$ . This is in contrast to state-of-the-art methods that are  $O(n^5)$  or even  $O(n^8)$ , without being more accurate. Our method is applicable for all  $n \geq 4$  and handles properly both planar and non-planar configurations. Our central idea is to express the  $n$  3D points as a weighted sum of four virtual control points. The problem then reduces to estimating the coordinates of these control points in the camera referential, which can be done in  $O(n)$  time by expressing these coordinates as weighted sum of the eigenvectors of a  $12 \times 12$  matrix and solving a small constant number of quadratic equations to pick the right weights. The advantages of our method are demonstrated by thorough testing on both synthetic and real-data.*

## 1. Introduction

The aim of the Perspective- $n$ -Point problem—PnP in short—is to determine the position and orientation of a camera given its intrinsic parameters and a set of  $n$  correspondences between 3D points and their 2D projections. It has many applications in Computer Vision, Robotics, Augmented Reality and has received much attention in both the Photogrammetry [21] and Computer Vision [12] communities. In particular, applications such as feature point-based camera tracking [26, 18] require dealing with hundreds of noisy feature points in real-time, which requires computationally efficient methods.

In this paper, we introduce a non-iterative solution with better accuracy and much lower computational complexity than non-iterative state-of-the-art methods, and much faster than iterative ones with little loss of accuracy. Our approach is  $O(n)$  for  $n \geq 4$  whereas all other methods we know of are either specialized for small fixed values of  $n$ , very sensitive to noise, or much slower. The specialized methods include those designed to solve the P3P problem [9, 23]. Among those that handle arbitrary values of  $n$  [8, 6, 13, 11, 23, 28, 7, 2, 9], the lowest-complexity one [7] is  $O(n^2)$  but has been shown to be unstable for noisy

2D locations [2]. This is currently addressed by algorithms that are  $O(n^5)$  [23] or even  $O(n^8)$  [2] for better accuracy whereas our  $O(n)$  approach achieves even better accuracy and reduced sensitivity to noise, as depicted by Fig. 1 in the  $n = 6$  case and demonstrated for larger values of  $n$  in the result section.

A natural alternative to non-iterative approaches are iterative ones [19, 5, 14, 17, 20] that rely on minimizing an appropriate criterion. They can deal with arbitrary numbers of correspondences and achieve excellent precision when they converge properly. In particular Lu *et al.* [20] introduced a very accurate algorithm, which is fast in comparison with other iterative ones but slow compared to non-iterative methods. As shown in Fig. 1, our method achieves an accuracy that is almost as good, and is much faster and without requiring an initial estimate. This is significant because iterative methods are prone to failure if poorly initialized. For instance, Lu *et al.*'s approach relies on an initial estimation of the camera pose based on a weak-perspective assumption, which can lead to instabilities when the assumption is not satisfied. This happens when the points of the object are projected onto a small region on the side of the image and our solution performs more robustly under these circumstances. Furthermore, if maximal precision is required our output can be used to initialize Lu *et al.*'s, yielding both higher stability and faster convergence.

Our central idea was to write the coordinates of the  $n$  3D points as a weighted sum of four virtual control points. This reduces the problem to estimating the coordinates of the control points in the camera referential, which can be done in  $O(n)$  time by expressing these coordinates as weighted sum of the eigenvectors of a  $12 \times 12$  matrix and solving a small *constant* number of quadratic equations to pick the right weights. The accuracy of the resulting algorithm is better than the one of the more complex ones [23, 2]. Our approach also extends to planar configurations, which cause problems for some methods as discussed in [22, 24], by using three control points instead of four.

In the remainder of the paper, we first discuss related work focusing on accuracy and computation time. We then introduce our new formulation and derive our system of linear and quadratic equations. Finally, we compare our method against the state-of-the-art ones using synthetic data and demonstrate it using real data.

<sup>\*</sup>This work was supported in part by the Swiss National Science Foundation and by funds of the European Commission under the IST-project 034307 DYVINE (Dynamic Visual Networks).

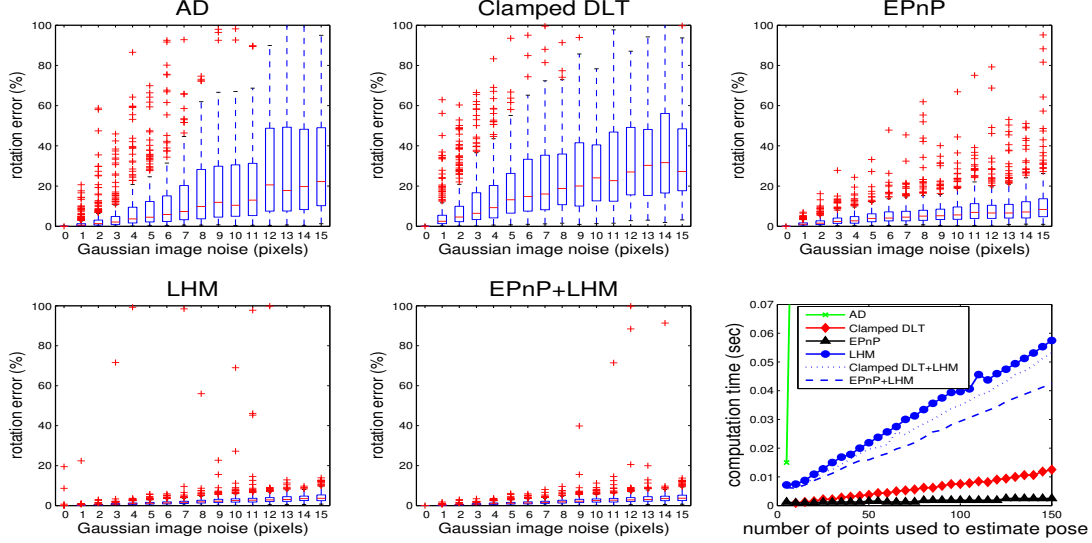


Figure 1. Comparing our method against state-of-the-art ones. **Top row.** Accuracy of non-iterative methods as a function of noise when using  $n = 6$  3D/2D correspondences: AD, the method of Ansar and Daniilidis [2]; Clamped DLT, the DLT algorithm after clamping the internal parameters with their known values; and EPnP, our method. **Bottom row.** Accuracy of iterative methods using  $n = 6$ : LHM, Lu *et al.*'s method [20] initialized with a weak perspective assumption; EPnP+LHM, Lu *et al.*'s algorithm initialized with the output of our algorithm. **Bottom-Right.** Computation times for all five methods –plus Clamped DLT+LHM– as a function of  $n$ , the number of correspondences, using MATLAB on a standard PC. The accuracy results are plotted using the boxplot representation: The blue boxes denote the first and third quartiles of the errors, the lines extending from each end of the box depict the statistical extent of the data, and the crosses indicate observations that fall out of it. Our method is both more accurate and faster than the other non-iterative ones, especially for large amounts of noise, and has almost similar accuracy than the iterative LHM. Furthermore, if maximal precision is required, the output of our algorithm can be used to initialize LHM and obtain faster convergence than using other techniques.

## 2. Related Work

There is an immense body of literature on pose estimation from point correspondences and, here, we focus on non-iterative approaches since our method falls in this category. In addition, we will also introduce Lu *et al.* [20] iterative method, which yields very good results and against which we compare our own approach.

Most of the non-iterative approaches, if not all of them, proceed by first estimating the points 3D positions in the camera coordinate system by solving for the points depths. It is then easy to retrieve the camera position and orientation as the Euclidean motion that aligns these positions on the given coordinates in the world coordinate system [15, 3, 29].

The P3P case has been extensively studied in the literature, and many closed form solutions have been proposed such as [6, 8, 9, 11, 23]. It typically involves solving for the roots of an eight-degree polynomial with no odd terms, yielding up to four solutions in general, so that a fourth point is needed for disambiguation. Fisher and Bolles [8] reduced this P4P problem to the P3P by taking subsets of three points and checking consistency. Similarly, Horaud *et al.* [13] reduced the P4P to a 3-line problem. For the 4 and 5 points problem, Triggs [28] derived a system of quadratic polynomials, which solves using multiresultant

theory. However, as pointed out in [2], this does not perform well for larger number of points.

Even if four correspondences are sufficient in general to estimate the pose, it is nonetheless desirable to consider larger point sets to introduce redundancy and reduce the sensitivity to noise. To do so, Quan and Lan [23] consider triplets of points and for each of them derive four-degree polynomials in the unknown point depths. The coefficients of these polynomials are then arranged in a  $\frac{(n-1)(n-2)}{2} \times 5$  matrix and singular value decomposition (SVD) is used to estimate the unknown depths. This method is repeated for all of the  $n$  points and therefore involves  $O(n^5)$  operations.<sup>1</sup> It should be noted that, even if it is not done in [23], this complexity could be reduced to  $O(n^3)$  by applying the same trick as we do when performing the SVD, but even then, it would remain slower than our method. Ansar and Daniilidis [2] derive a set of quadratic equations arranged in a  $\frac{n(n-1)}{2} \times \left(\frac{n(n+1)}{2} + 1\right)$  linear system, which, as formulated in the paper, requires  $O(n^8)$  operations to be solved. They show their approach performs better than [23].

The complexity of the previous two approaches stems from the fact that quadratic terms are introduced from the inter-point distances constraints. The linearization of these

<sup>1</sup>Following [10], we consider that the SVD for a  $m \times n$  matrix can be computed by a  $O(4m^2n + 8mn^2 + 9n^3)$  algorithm.

equations produces additional parameters, which increase the complexity of the system. Fiore’s method [7] avoids the need for these constraints: He initially forms a set of linear equations from which the world to camera rotation and translation parameters are eliminated, allowing the direct recovery of the point depths without considering the inter-point distances. This procedure allows the estimation of the camera pose in  $O(n^2)$  operations, which makes real-time performance possible for large  $n$ . Unfortunately, ignoring nonlinear constraints produces poor results in the presence of noise [2].

By contrast, our method is able to consider nonlinear constraints but requires  $O(n)$  operations only. Furthermore, in our synthetic experiments, it yields results that are more accurate than those of [2].

It is also worth mentioning that for large values of  $n$  one could use the Direct Linear Transformation (DLT) algorithm [1, 12]. However, it ignores the intrinsic camera parameters we assume to know here, and therefore generally leads to less stable pose estimate. A way to exploit our knowledge of the intrinsic parameters is to clamp the retrieved values to the known ones, but the accuracy still remains low.

Finally, among many other iterative methods, Lu *et al.*’s [20] is one of the fastest and most accurate. It minimizes an error expressed in 3D space, unlike many earlier methods that attempt to minimize reprojection residuals. The main difficulty is to impose the orthonormality of the rotation matrix. It is done by optimizing alternatively on the translation vector and the rotation matrix. In practice, the algorithm tends to converge fast but can get stuck in an inappropriate local minimum if incorrectly initialized. Our experiments show our method is slightly less accurate than Lu *et al.*’s when it find the correct minimum, but also that it is faster and more stable.

### 3. Our Approach to the PnP Problem

Let us assume we are given a set of  $n$  *reference points* whose 3D coordinates are known in the world coordinate system and whose 2D image projections are also known. As most of the proposed solutions to the PnP Problem, we aim at retrieving their coordinates in the camera coordinate system. It is then easy and standard to retrieve the orientation and translation as the Euclidean motion that aligns both sets of coordinates [15, 3, 29].

Most existing approaches attempt to solve for the depths of the reference points in the camera coordinate system. By contrast, we express their coordinates as a weighted sum of virtual *control points*. We need 4 non-coplanar such control points for general configurations, and only 3 for planar configurations. Given this formulation, the coordinates of the control points in the camera coordinate system become the unknown of our problem. For large  $n$ ’s, this is a much smaller number of unknowns than the  $n$  depth values that

traditional approaches have to deal with and is key to our efficient implementation.

The solution of our problem can be expressed as a vector that lies in the kernel of a matrix of size  $2n \times 12$  or  $2n \times 9$ . We denote this matrix as  $\mathbf{M}$  and can be easily computed from the 3D world coordinates of the reference points and their 2D image projections. More precisely, it is a weighted sum of the null eigenvectors of  $\mathbf{M}$ . Given that the correct linear combination is the one that yields 3D camera coordinates for the control points that preserve their distances, we can find the appropriate weights by solving small systems of quadratic equations, which can be done at a negligible computational cost. In fact, for  $n$  sufficiently large—about 15 in our implementation—the most expensive part of this whole computation is that of the matrix  $\mathbf{M}^T \mathbf{M}$ , which grows linearly with  $n$ .

In the remainder of this section, we first discuss our parameterization in terms of control points in the generic non-planar case. We then derive the matrix  $\mathbf{M}$  in whose kernel the solution must lie and introduce the quadratic constraints required to find the proper combination of eigenvectors. Finally, we show that this approach also applies to the planar case.

#### 3.1. Parameterization in the General Case

Let the reference points, that is, the  $n$  points whose 3D coordinates are known in the world coordinate system, be

$$\mathbf{p}_i, \quad i = 1, \dots, n.$$

Similarly, let the 4 control points we use to express their world coordinates be

$$\mathbf{c}_j, \quad j = 1, \dots, 4.$$

When necessary, we will specify that the point coordinates are expressed in the world coordinate system by using the  $^w$  superscript, and in the camera coordinate system by using the  $^c$  superscript. We express each reference point as a weighted sum of the control points

$$\mathbf{p}_i^w = \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^w, \text{ with } \sum_{j=1}^4 \alpha_{ij} = 1, \quad (1)$$

where the  $\alpha_{ij}$  are homogeneous barycentric coordinates. They are uniquely defined and can easily be estimated. The same relation holds in the camera coordinate system and we can also write

$$\mathbf{p}_i^c = \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^c. \quad (2)$$

In theory the control points can be chosen arbitrarily. However, in practice, we have found that the stability of our method is increased by taking the centroid of the reference points as one of the control point, and select the rest in such a way that they form a basis aligned with the principal directions of the data. This is normal because it amounts to conditioning the linear system of equations that are introduced below using a normalization technique very close to the one recommended for the classic DLT algorithm [12].

### 3.2. The Solution as Weighted Sum of Eigenvectors

We now derive the matrix  $\mathbf{M}$  in whose kernel the solution must lie given that the 2D projections of the reference points are known. Let  $\mathbf{A}$  be the camera internal calibration matrix and  $\{\mathbf{u}_i\}_{i=1,\dots,n}$  the 2D projections of the  $\{\mathbf{p}_i\}_{i=1,\dots,n}$  reference points. We have

$$\forall i, w_i \begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = \mathbf{A} \mathbf{p}_i^c = \mathbf{A} \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^c, \quad (3)$$

where the  $w_i$  are scalar projective parameters. We now expand this expression by considering the specific 3D coordinates  $[x_j^c, y_j^c, z_j^c]^\top$  of each  $\mathbf{c}_j^c$  control point, the 2D coordinates  $[u_i, v_i]^\top$  of the  $\mathbf{u}_i$  projections, and the  $f_u, f_v$  focal length coefficients and the  $(u_c, v_c)$  principal point that appear in the  $\mathbf{A}$  matrix. Eq. 3 then becomes

$$\forall i, w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^4 \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix}. \quad (4)$$

The unknown parameters of this linear system are the 12 control point coordinates  $\{(x_j^c, y_j^c, z_j^c)\}_{j=1,\dots,4}$  and the  $n$  projective parameters  $\{w_i\}_{i=1,\dots,n}$ . The last row of Eq. 4 implies that  $w_i = \sum_{j=1}^4 \alpha_{ij} z_j^c$ . Substituting this expression in the first two rows yields two linear equations for each reference point:

$$\sum_{j=1}^4 \alpha_{ij} f_u x_j^c + \alpha_{ij} (u_c - u_i) z_j^c = 0, \quad (5)$$

$$\sum_{j=1}^4 \alpha_{ij} f_v y_j^c + \alpha_{ij} (v_c - v_i) z_j^c = 0. \quad (6)$$

Note that the  $w_i$  projective parameter does not appear anymore in those equations. Hence, by concatenating them for all  $n$  reference points, we generate a linear system of the form

$$\mathbf{M} \mathbf{x} = \mathbf{0}, \quad (7)$$

where  $\mathbf{x} = [\mathbf{c}_1^c, \mathbf{c}_2^c, \mathbf{c}_3^c, \mathbf{c}_4^c]^\top$  is a 12-vector made of the unknowns, and  $\mathbf{M}$  is a  $2n \times 12$  matrix, generated by arranging the coefficients of Eqs. 5 and 6 for each reference point. Unlike in the case of DLT, we do not have to normalize the 2D projections since Eqs. 5 and 6 do not involve the image referential system.

The solution therefore belongs to the null space, or kernel, of  $\mathbf{M}$ , and can be expressed as

$$\mathbf{x} = \sum_{i=1}^N \beta_i \mathbf{v}_i \quad (8)$$

where the set  $\mathbf{v}_i$  are the columns of the right-singular vectors of  $\mathbf{M}$  corresponding to the  $N$  null singular values of  $\mathbf{M}$ . They can be found efficiently as the null eigenvectors of matrix  $\mathbf{M}^\top \mathbf{M}$ , which is of small constant ( $12 \times 12$ ) size. Computing the product  $\mathbf{M}^\top \mathbf{M}$  has  $O(n)$  complexity, and is the most time consuming step in our method when  $n$  is sufficiently large, about 15 in our implementation.

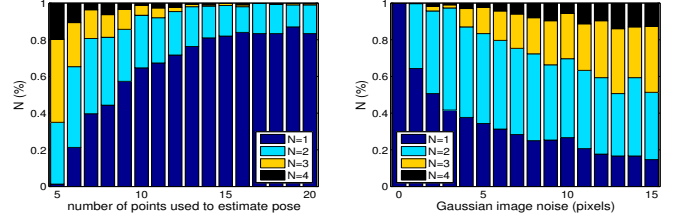


Figure 2. Effective number  $N$  of null singular values in  $\mathbf{M}$ . Each vertical bar represents the distributions of  $N$  for a total of 300 experiments. On the left, we plot the results for a fixed image noise of  $\sigma = 10$  pixels and an increasing number of reference points, and the results on the right correspond to a fixed  $n = 6$  number of reference points and increasing level of noise in the 2D projections.

### 3.3. Choosing the Right Linear Combination

Given that the solution can be expressed as a linear combination of the null eigenvectors of  $\mathbf{M}^\top \mathbf{M}$ , finding it amounts to computing the appropriate values for the  $\{\beta_i\}_{i=1,\dots,N}$  coefficients of Eq. 8. Note that this approach applies even when the system of Eq. 7 is under-constrained, for example because the number of input correspondences is 4 or 5, yielding to only 8 or 10 equations, which is less than the number of unknowns.

In theory, given perfect data from at least six points, the dimension  $N$  of the null-space of  $\mathbf{M}^\top \mathbf{M}$ , should be exactly one because of the scale ambiguity. However, in the presence of noise,  $\mathbf{M}^\top \mathbf{M}$  has no eigenvalue that is strictly zero but may have several that are very small.

In this section, we therefore show that the fact that the distances between control points must be preserved can be expressed in terms of a small number of quadratic equations, which can be efficiently solved to compute  $\{\beta_i\}_{i=1,\dots,N}$  for  $N = 1, 2, 3$  and 4. The techniques we describe here could be extended to the case  $N \geq 5$  but we have not found it necessary to do so in any of our experiments on real and synthetic data.

In practice, instead of trying to pick a value of  $N$  which would be error-prone if several eigenvalues had similar magnitudes, we compute solutions for all four values of  $N$  and keep the one that yields the smallest reprojection error

$$res = \sum_i \text{dist}^2(\mathbf{A}[\mathbf{R}|\mathbf{t}] \begin{bmatrix} \mathbf{c}_i^w \\ 1 \end{bmatrix}, \mathbf{u}_i), \quad (9)$$

where  $\text{dist}(\tilde{\mathbf{m}}, \mathbf{n})$  is the 2D distance between point  $\mathbf{m}$  expressed in homogeneous coordinates, and point  $\mathbf{n}$ . This improves robustness without any noticeable computational penalty because the most expensive operation is the computation of  $\mathbf{M}^\top \mathbf{M}$ , which is done only once, and not the solving of a few quadratic equations. The distribution of values of  $N$  estimated in this way is depicted by Fig. 2.

We now turn to the description of the quadratic constraints we introduce for  $N = 1, 2, 3$  and 4.



**Case  $N = 1$ :** We simply have  $\mathbf{x} = \beta \mathbf{v}$ . We solve for  $\beta$  by writing that the distances between control points as retrieved in the camera coordinate system should be equal to the ones computed in the world coordinate system when using the given 3D coordinates.

Let  $\mathbf{v}^{[i]}$  be the sub-vector of  $\mathbf{v}$  that corresponds to the coordinates of the control point  $\mathbf{c}_i^c$ . For example,  $\mathbf{v}^{[1]}$  will represent the vectors made of the three first elements of  $\mathbf{v}$ . Maintaining the distance between pairs of control points  $(\mathbf{c}_i, \mathbf{c}_j)$  implies that

$$\|\beta \mathbf{v}^{[i]} - \beta \mathbf{v}^{[j]}\|^2 = \|\mathbf{c}_i^w - \mathbf{c}_j^w\|^2. \quad (10)$$

The distances  $\|\mathbf{c}_i^w - \mathbf{c}_j^w\|$  are known and  $\beta$  computed in closed-form as

$$\beta = \frac{\sum_{\{i,j\} \in [1;4]} \|\mathbf{v}^{[i]} - \mathbf{v}^{[j]}\| \cdot \|\mathbf{c}_i^w - \mathbf{c}_j^w\|}{\sum_{\{i,j\} \in [1;4]} \|\mathbf{v}^{[i]} - \mathbf{v}^{[j]}\|^2}. \quad (11)$$

**Case  $N = 2$ :** We now have  $\mathbf{x} = \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2$ , and our distance constraints become

$$\|(\beta_1 \mathbf{v}_1^{[i]} + \beta_2 \mathbf{v}_2^{[i]}) - (\beta_1 \mathbf{v}_1^{[j]} + \beta_2 \mathbf{v}_2^{[j]})\|^2 = \|\mathbf{c}_i^w - \mathbf{c}_j^w\|^2. \quad (12)$$

$\beta_1$  and  $\beta_2$  only appear in the quadratic terms and we solve for them using a technique called “linearization” in cryptography, which was employed by [2] to estimate the point depths. It involves solving a linear system in  $[\beta_{11}, \beta_{12}, \beta_{22}]^T$  where  $\beta_{11} = \beta_1^2, \beta_{12} = \beta_1 \beta_2, \beta_{22} = \beta_2^2$ .

Since we have four control points, we can generate a linear system of six equations in the  $\beta_{ab}$ <sup>2</sup>:

$$\mathbf{L}\boldsymbol{\beta} = \boldsymbol{\rho}, \quad (13)$$

where  $\mathbf{L}$  is a  $6 \times 3$  matrix formed with the elements of  $\mathbf{v}_1$  and  $\mathbf{v}_2$ ,  $\boldsymbol{\rho}$  is a 6-vector with the squared distances  $\|\mathbf{c}_i^w - \mathbf{c}_j^w\|^2$ , and  $\boldsymbol{\beta} = [\beta_{11}, \beta_{12}, \beta_{22}]^T$  is the vector of unknowns. We solve this system using the pseudoinverse of  $\mathbf{L}$  and choose the signs for the  $\beta_a$  so that all the  $\mathbf{p}_i^c$  have positive  $z$  coordinates.

This yields  $\beta_1$  and  $\beta_2$  values that can be further refined by using the formula of Eq. 11 to estimate a common scale  $\beta$  so that  $\mathbf{c}_i^c = \beta(\beta_1 \mathbf{v}_1^{[i]} + \beta_2 \mathbf{v}_2^{[i]})$ . Because solving Eq. 13 is an indirect computation, the least-squares solution of Eq. 11 is more accurate in the presence of noise.

**Case  $N = 3$ :** As in the  $N = 2$  case, we use the six constraints of Eq. 12. This yields again a linear system  $\mathbf{L}\boldsymbol{\beta} = \boldsymbol{\rho}$ , although with different dimensionalities. Now  $\mathbf{L}$  is a square  $6 \times 6$  matrix formed with the elements of  $\mathbf{v}_1, \mathbf{v}_2$  and  $\mathbf{v}_3$ . The number of unknowns is extended to six, being  $\boldsymbol{\beta} = [\beta_{11}, \beta_{12}, \beta_{13}, \beta_{22}, \beta_{23}, \beta_{33}]^T$ . We follow the same procedure as before, except that we now use the inverse of  $\mathbf{L}$  instead of its pseudo-inverse.

<sup>2</sup>We use the indices  $a$  and  $b$  for the  $\beta$ ’s in order to differentiate from the indices  $i$  and  $j$  used for the 3D points.

**Case  $N = 4$ :** We now have four  $\beta_a$  unknowns and, in theory, the six distance constraints we have been using so far should still suffice. Unfortunately, the linearization procedure treats all 10 products  $\beta_{ab} = \beta_a \beta_b$  as unknowns and there are not enough constraints anymore. We solve this problem using a *relinearization* technique [16] whose principle is the same as the one we use to determine the control points coordinates.

The solution for the  $\beta_{ab}$  is in the null space of a first homogeneous linear system made from the original constraints. The correct coefficients are found by introducing new quadratic equations and solving them again by linearization, hence the name “relinearization”. These new quadratic equations are derived from the fact that we have, by commutativity of the multiplication

$$\beta_{ab}\beta_{cd} = \beta_a\beta_b\beta_c\beta_d = \beta_{a'b'}\beta_{c'd'}, \quad (14)$$

where  $\{a', b', c', d'\}$  represents any permutation of the integers  $\{a, b, c, d\}$ .

### 3.4. The Planar Case

In the planar case, that is, when the moment matrix of the reference points has one very small eigenvalue, we need only three control points. The dimensionality of  $\mathbf{M}$  and its eigenvectors  $\mathbf{v}_i$  are reduced, but the above equations remain mostly valid. The main difference is that the number of quadratic constraints drops from 6 to 3. As a consequence, we need use of the relinearization technique introduced in the  $N = 4$  case of the previous section for  $N \geq 3$ .

## 4. Results

We compare the accuracy and speed of our approach against that of state-of-the-art ones, both on simulated and real image data.

### 4.1. Synthetic Experiments

We produced synthetic 3D-to-2D correspondences in a  $640 \times 480$  image acquired using a virtual calibrated camera with an effective focal length of  $f_u = f_v = 800$  and a principal point at  $(u_c, v_c) = (320, 240)$ . We generated different sets for the input data. For the *centered* data, the 3D reference points were uniformly distributed into the  $x, y, z$  interval  $[-2, 2] \times [-2, 2] \times [4, 8]$ . For the *uncentered* data, the ranges were modified to  $[1, 2] \times [1, 2] \times [4, 8]$ . We also added Gaussian *noise* to the corresponding 2D point coordinates, and considered a percentage of *outliers*, for which the 2D coordinate was randomly selected within the whole image.

Given the true camera rotation  $\mathbf{R}_{true}$  and translation  $\mathbf{t}_{true}$ , we computed the relative error of the estimated rotation  $\mathbf{R}$  by  $E_{rot}(\%) = \|\mathbf{q}_{true} - \mathbf{q}\|/\|\mathbf{q}\|$ , where  $\mathbf{q}$  and  $\mathbf{q}_{true}$  are the normalized quaternions corresponding to the rotation matrices. Similarly, the relative error of the estimated translation  $\mathbf{t}$  is determined by  $E_{trans}(\%) = \|\mathbf{t}_{true} - \mathbf{t}\|/\|\mathbf{t}\|$ .

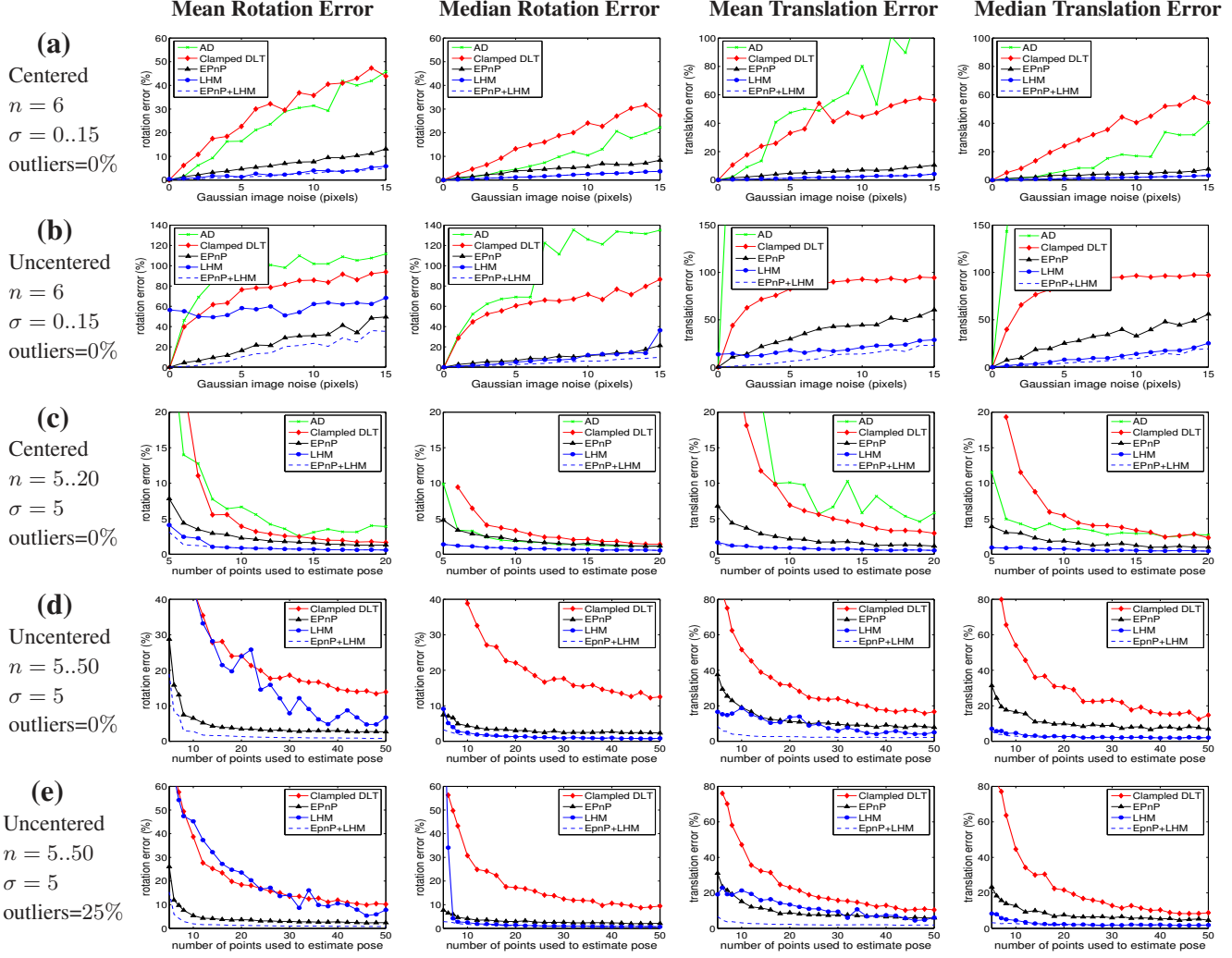


Figure 3. Non Planar case. Mean and median rotation and translation errors for different experiments.

All the plots discussed in this section were created by running 300 independent MATLAB simulations. To estimate running times, we ran the code 100 times for each example and recorded the average run time.

#### 4.1.1 The Non-Planar Case

For the non-planar case, we compared the accuracy and running times of our algorithm, which we denote as *EPnP*, to: *AD*, the non-iterative method of Ansar and Daniilidis [2]; *Clamped DLT*, the DLT algorithm after clamping the internal parameters with their known values; *LHM*, the Lu *et al.*'s [20] iterative method initialized with a weak perspective assumption; *EPnP+LHM*, Lu *et al.*'s algorithm initialized with the output of our algorithm.

On Fig. 1, we plot the rotational errors produced by the three non-iterative algorithms, and the two iterative ones as a function of noise when using  $n = 6$  points. We use the boxplot representation<sup>3</sup>, where each column depicts the dis-

tribution of the errors for the 300 different simulations. A concise way to summarize the boxplot results is to plot both the mean and median results for all simulations: The difference between the mean and the median mainly comes from the high errors represented as red crosses in the boxplots; the greater it is the less stable the method. This is shown in Fig. 3a, where in addition to the rotation error we also plot the translation error. Our method is consistently more accurate and stable than the other non-iterative ones, especially for large amounts of noise. It is only slightly less accurate than the LHM iterative algorithm. In fact, as shown in Fig. 3b, it ever performs better when instead of using well spread data as in the previous case, we simulate data that covers only a small fraction of the image.

In Fig. 3c, we plot the errors as a function of the number of reference points, when the noise is fixed to  $\sigma = 5$ .

third  $Q3$  quartiles, a horizontal line indicating the median, and a dashed vertical line representing the data extent taken to be  $Q3 + 1.5(Q3 - Q1)$ . The red crosses denote points lying outside of this range.

<sup>3</sup>The boxplot representation consists of a box denoting the first  $Q1$  and

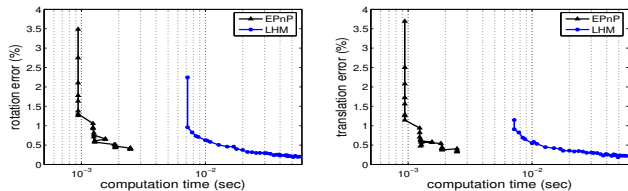


Figure 4. Median error as a function of required amount of computation, itself a function of the number of points being used, for our approach and the iterative method of Lu *et al.* [Lu00].

Again, our method performs better than the other non-iterative techniques and very nearly as well as LHM. It even represents a more stable solution when dealing with the uncentered data of Fig. 3d and data which includes outliers, as in Fig. 3e. Note that in all the cases where LHM does not converge perfectly, the combination EPnP+LHM provides accurate results. In the last two graphs, we did not compare the performance of AD, because this algorithm does not normalize the 2D coordinates, and hence, cannot deal well with uncentered data.

As shown in the bottom-right side of Fig. 1, the computational cost of our method grows linearly with the number of correspondences and remains much lower than all the others. It even compares favorably to clamped DLT, which is known to be fast. As shown, in Fig. 4, EPnP requires about a tenth of the time required by LHM to achieve similar accuracy levels. Although the difference becomes evident for a large number of reference points, it is significant even for small numbers. For instance, for  $n = 6$  points, our algorithm is about 10 times faster than LHM, and about 200 times faster than AD.

#### 4.1.2 The Planar Case

Schweighofer and Pinz [24] prove that when the reference points lie on a plane, camera pose suffers from an ambiguity that results in significant instability. They propose a new algorithm for resolving it and refine the solution using Lu *et al.* [20]’s method. Hereafter, we will refer to this combined algorithm as *SP+LHM*, which we will compare against to our method, *AD*, and *LHM*. We omit Clamped DLT because it is not applicable in the planar case.

Fig. 5 depicts the errors as a function of the image noise, when  $n = 10$  and for reference points lying on a plane with tilt of either 0 or 30 degrees. To obtain a fair comparison we present the results as was done in [24] and the errors are only averaged among those solutions not affected by the pose ambiguity. We also report the percentage of solutions which are considered as outliers. When the points are lying on a frontoparallel plane, there is no pose ambiguity and all the methods have a similar accuracy, with no outliers, as shown by the two left-most graphs of Fig. 5. The pose ambiguity problem appears only for inclined planes, as shown by the other three graphs of Fig. 5. Note that for AD, the number of outliers is really large, and even the errors for the

inliers are considerable. EPnP and LHM produce a much reduced number of outliers and similar results accuracy for the inliers. As before, the *SP+LHM* method computes the correct pose for almost all the cases. Note that we did not plot the errors for LHM, because when considering only the correct poses, it is the same as *SP+LHM*.

As in the non-planar case, the EPnP solution proposed here is much faster than the others. For example for  $n = 10$  and a tilt of  $30^\circ$ , our solution is about 200 times faster than AD, 30 times faster than LHM, even though the MATLAB code for the latter is not optimized.

## 4.2. Real Images

We tested our algorithm on noisy correspondences, that may include erroneous ones, obtained on real images with our implementation of the keypoint recognition method of [18]. We trained the method on a calibrated reference image of a textured box of known dimensions, using the model depicted by Fig. 6-left. At run time, the method generates about 200 correspondences per image. To filter out the erroneous ones, we use RANSAC on small subsets made of 7 correspondences from which we estimate the pose using our PnP method. This is effective because, even though our algorithm is designed to work with a large number of correspondences, it is also faster than other algorithms for small numbers of points, as discussed above. Furthermore, once the set of inliers has been selected, we use all of them to refine the camera pose. This gives a new set of inliers and the estimation is iterated until no additional inliers are found. Fig. 6-right shows two different frames of the sequence. In the corner of each frame, we display the reference image and the correspondences between the points which are considered as inliers.

## 5. Conclusion

We have proposed an  $O(n)$  non-iterative solution to the PnP problem that is faster and more accurate than the best current techniques. It is only slightly less accurate than one the most recent iterative ones [20] but much faster and more stable. Furthermore, our approach can be profitably be used to initialize the iterative algorithm.

Our central idea—expressing the 3D points as a weighted sum of four virtual control points and solving in terms of their coordinates—is very generic. We demonstrated it in the context of the PnP problem but it is potentially applicable to problems ranging from the estimation of the Essential matrix from a large number of points for Structure-from-Motion applications [27] to shape recovery of deformable surfaces. The latter is particularly promising because there have been many approaches to parameterizing such surfaces using control points [25, 4], which would fit perfectly into our framework and allow us to recover not only pose but also shape. This is what we will focus on in future research.

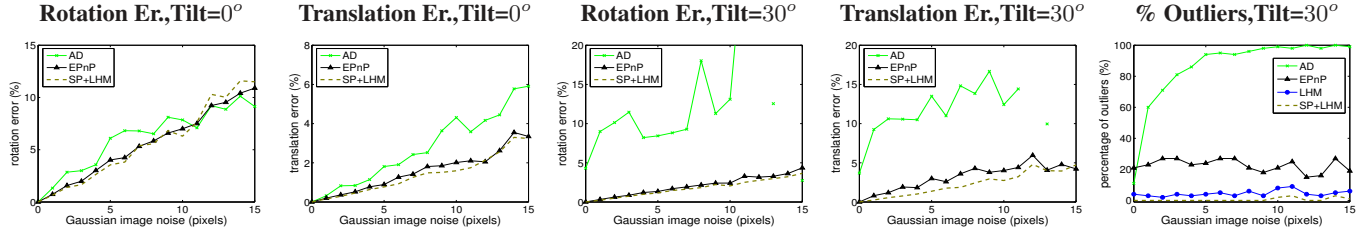


Figure 5. Planar case. Errors as a function of the image noise when using  $n = 6$  reference points. For Tilt=0°, there are no pose ambiguities and the results represent the mean over all the experiments. For Tilt=30°, the errors are only averaged among the experiments without the pose ambiguity. The right-most figure represents the number of solutions considered as outliers, which are defined as those for which the average error in the estimated 3D position of the reference points is larger than a threshold.

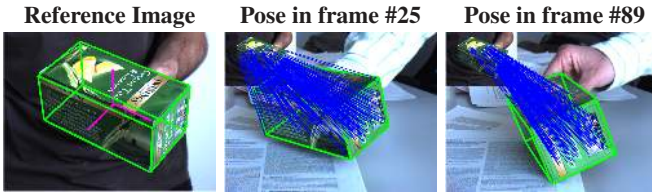


Figure 6. Real images. **Left.** Calibrated reference image. **Right.** Reprojection of the model of the box on two video frames. The camera pose has been computed using the set of correspondences depicted by the thin blue lines.

## References

- [1] Y. Abdel-Aziz and H. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Proc. ASP/UI Symp. Close-Range Photogrammetry*, pages 1–18, 1971. 3
- [2] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. *PAMI*, 25(5):578–589, 2003. 1, 2, 3, 5, 6
- [3] K. Arun, T. Huang, and S. Blostein. Least-Squares Fitting of Two 3D Points Sets. *PAMI*, 9(5):698–700, 1987. 2, 3
- [4] Y. Chang and A. P. Rockwood. A Generalized de Casteljau Approach to 3D Free-form Deformation. *ACM SIGGRAPH*, pages 257–260, 1994. 7
- [5] D. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. *IJCV*, 15:123–141, 1995. 1
- [6] M. Dhome, M. Richetin, and J. T. Lapreste. Determination of the attitude of 3D objects from a single perspective view. *PAMI*, 11(12):1265–1278, 1989. 1, 2
- [7] P. D. Fiore. Efficient linear solution of exterior orientation. *PAMI*, 23(2):140–148, 2001. 1, 3
- [8] M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications ACM*, 24(6):381–395, 1981. 1, 2
- [9] X. S. Gao, X. R. Hou, J. Tang, and H. F. Cheng. Complete solution classification for the perspective-three-point problem. *PAMI*, 25(8):930–943, 2003. 1, 2
- [10] G. Golub and C. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996. 2
- [11] R. Haralick, D. Lee, K. Ottenburg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Proc. CVPR*, pages 592–598, June 1991. 1, 2
- [12] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. 1, 3
- [13] R. Horaud, B. Conio, O. Le Boulleux, and B. Lacolle. An analytic solution for the perspective-4-point problem. *CVGIP*, 47(1):33–44, 1989. 1, 2
- [14] R. Horaud, F. Dornaika, and B. Lamiroy. Object pose: The link between weak perspective, paraperspective, and full perspective. *IJCV*, 22(2):173–189, 1997. 1
- [15] B. Horn, H. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA*, 5(7):1127–1135, 1988. 2, 3
- [16] A. Kipnis and A. Shamir. *Advances in Cryptology*, volume 1666/1999, pages 19–30. Springer, 1999. 5
- [17] R. Kumar and A. R. Hanson. Robust methods for estimating pose and a sensitivity analysis. *CVIU*, 60:313–342, 1994. 1
- [18] M. Özuysal, P. Fua, and V. Lepetit. Keypoint recognition in ten lines of code. In *Proc. CVPR*, June 2007. 1, 7
- [19] D. G. Lowe. Fitting parameterized three-dimensional models to images. *PAMI*, 13(5):441–450, June 1991. 1
- [20] C. P. Lu, G. D. Hager, and E. Mjølness. Fast and globally convergent pose estimation from video images. *PAMI*, 22(6):610–622, 2000. 1, 2, 3, 6, 7
- [21] C. McGloves, E. Mikhail, and J. Bethel, editors. *Manual of Photogrammetry*. Am. Soc. Phot. Remote Sensing, 2004. 1
- [22] D. Oberkampf, D. DeMenthon, and L. Davis. Iterative Pose Estimation using Coplanar Feature Points. *CVIU*, 63:495–511, 1996. 1
- [23] L. Quan and Z. Lan. Linear N-Point Camera Pose Determination. *PAMI*, 21(7):774–780, July 1999. 1, 2
- [24] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *PAMI*, 28(12):2024–2030, 2006. 1, 7
- [25] T. Sederberg and S. Parry. Free-Form Deformation of Solid Geometric Models. *ACM SIGGRAPH*, 20(4), 1986. 7
- [26] I. Skrypnik and D. G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *Proc. ISMAR*, pages 110–119, 2004. 1
- [27] H. Stewènius, C. Engels, and D. Nister. Recent developments on direct relative orientation. *Int. Soc. for Photogr. Remote Sensing*, 60:284–294, June 2006. 7
- [28] B. Triggs. Camera Pose and Calibration from 4 or 5 known 3D Points. In *Proc. ICCV*, pages 278–284, Sept 1999. 1, 2
- [29] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *PAMI*, 13, 1991. 2, 3