Procrustean point-line registration and the NPnP problem

Andrea Fusiello*, Fabio Crosilla$^+$ and Francesco Malapelle*
$^{(*)}$DIEGM / $^{(+)}$DICA - University of Udine
Via Delle Scienze, 208 - 33100 Udine, Italy

Abstract

In this paper we formulate the point-line registration problem, which generalizes absolute orientation to point-line matching, in terms of an instance of the orthogonal Procrustes problem, and derive its solution. The same formulation solves the Non-Perspective-n-Point camera pose problem, which in turn generalizes exterior orientation to non-central cameras, i.e., generalized cameras where projection rays do not meet in a single point. Our Procrustean solution is very simple and compact, and copes also with scaling. Experiments with simulated data demonstrate that our method compares favourably with the state-of-the-art in terms of accuracy.

1. Introduction

The problem of estimating the orientation (position and angular attitude) of a perspective camera given its intrinsic parameters and a set of world-to-image correspondences is known as the *Perspective-n-Point* camera pose problem (PnP) in computer vision or *exterior orientation* problem in Photogrammetry. It can be seen as special case of a point-line registration problem, where the lines form a bundle. Indeed, perspective cameras are modelled as a bundle of rays and an image plane. If we remove the requirement that rays form a bundle, we are left with a set of rays rigidly connected and the imaging model is known as a *generalized camera*. In a generalized camera, every ray has its own origin, which is – in general – different from the others. These cameras can model all non-central devices [13] and also multi-camera systems, composed by many central cameras rigidly connected (the overall set of rays do not intersect a in single point, although they intersect group-wise). In turn, these multi-camera systems can represent an actual device composed of many cameras rigidly connected by a mechanical rig, or a set of images (taken by any number of cameras) whose relative orientation is known.

The Non-Perspective PnP (NPnP) problem generalizes the PnP problem (defined for central cameras) to generalized cameras, and requires to find the position, angular attitude and – possibly – scale of a non-perspective camera with respect to a set of known control points whose projections are also known.

The NPnP solution is particularly relevant in SLAM [15] and Structure from Motion (SfM). Consider for example a hierarchical SfM pipeline such as [16]: a cluster A of cameras can be glued to cluster B by solving a NPnP as soon as 4 points reconstructed in A are visible in some images of B, if B is modelled as a generalized camera. Please note that the points in B need not to be reconstructed. In contrast, the solution reported in [16] merges A and B with absolute orientation (+ scale), requiring that the same 4 points are reconstructed in A *and* B.

In general, the main advantage of multi-camera systems, when compared to the single camera, is that the visibility of the minimal number of control points required for the problem solution must be satisfied within the whole network and not by the single camera. This, of course, becomes very useful due to the restricted field of view of each camera.

While the problem has deep roots in the PnP, the literature relative to its non-perspective variant is fairly recent. Early work focused on minimal solvers (NP3P or NP4P problems, depending whether the scale is considered or not) [11, 1, 17], while recent work [9, 15] in particular concentrated on general solvers. The formulations in [8, 9] and [15] build independently on DLS [6]. However [8, 9] do not deal with scale, whereas [15] does. The latter feature is important in the context of structure from motion, where the scale is not consistent among generalized cameras.

In this paper we propose a new formulation of the NPnP problem based on Procrustes analysis, and two solutions that generalize PnP Fiore's algorithm [3] and PPnP [4] respectively. Both cope with scale, use only elementary linear algebra and have a very compact MATLAB implementation (Appendix C). The first one (Sec. 4) is a direct, closed form solution but it requires at least six point matches in general position and it minimizes an algebraic error. The second one (Sec. 3) is iterative but converges from an uninformed initialization in all cases (in our experiments). It minimizes a geometric error in object (3D) space, and works with the minimum number (four) of points.

2. Non-Perspective PnP formulation

Given a number n of 3D point-line correspondences $\mathbf{m}_j \leftrightarrow \ell_j$ the Non-Perspective PnP (NPnP) problem requires to find a rotation matrix R, a translation vector $\mathbf{t}$ and (possibly) a scale α such that:

$$\zeta_j \hat{\mathbf{p}}_j + \mathbf{o}_j = \alpha R \mathbf{M}_j + \mathbf{t} \quad \text{for} j = 1 \dots n, \tag{1}$$

where a line in space is described by means of its origin $\mathbf{o}_j$, its versor $\hat{\mathbf{p}}_j$ (or direction), and parameterized by ζ_j:

$$\ell_j : \ \zeta_j \hat{\mathbf{p}}_j + \mathbf{o}_j \quad \zeta_j \in \mathbb{R}. \tag{2}$$

All these lines are rigidly connected but the global pose parameters $R, \mathbf{t}, \alpha$ with respect to the external reference system (in which $\mathbf{M}_j$ are expressed) are missing, together with the values of ζ_j for $j = 1 \dots n$.

More in general, Problem (1) is that of registering the set of 3D points $\mathbf{M}_j$ with the set of lines given by $(\mathbf{o}_j, \hat{\mathbf{p}}_j)$, hence it can be also seen as a generalization of a 3D point-point registration, a.k.a. absolute orientation.

If the lines intersect in one point (forming a bundle), the problem reduces to the well-known *Perspective-n-Point* camera pose problem (PnP) (or exterior orientation), where the centre of bundle is the perspective centre of the camera and image points are identified with the rays. Therefore Problem (1) can also be seen as a generalization of PnP to non-central cameras, where the projective rays do not form a bundle.

In turn, a generalized camera can represent a cluster of rigidly connected central cameras, where the relative poses are fixed and known but the exterior pose parameters R, $\mathbf{c}$, α of the cluster with respect to the external reference system are missing.

After some rewriting Eq. (1) yields:

$$\underbrace{\mathrm{diag}(\zeta_1 \dots \zeta_n)}_{Z} \underbrace{\begin{bmatrix} \hat{\mathbf{p}}_1^\mathsf{T} \\ \vdots \\ \hat{\mathbf{p}}_n^\mathsf{T} \end{bmatrix}}_{P} R + \underbrace{\begin{bmatrix} \mathbf{o}_1^\mathsf{T} \\ \vdots \\ \mathbf{o}_n^\mathsf{T} \end{bmatrix}}_{O} R + \underbrace{\begin{bmatrix} \mathbf{c}^\mathsf{T} \\ \vdots \\ \mathbf{c}^\mathsf{T} \end{bmatrix}}_{\mathbf{1c}^\mathsf{T}} = \alpha \underbrace{\begin{bmatrix} \mathbf{M}_1^\mathsf{T} \\ \vdots \\ \mathbf{M}_n^\mathsf{T} \end{bmatrix}}_{S},$$

where $\mathbf{c} = -R^\mathsf{T}\mathbf{t}$. In matrix form:

$$ZPR + OR + \mathbf{1c}^\mathsf{T} = \alpha S, \tag{3}$$

where P is the matrix by rows of line versors, O is the matrix by rows of line origins, S is the matrix by rows of point coordinates defined in the external system, Z is the diagonal parameters matrix, $\mathbf{c}$ is the vector representing the position of the lines, R is the orthogonal rotation matrix, and $\mathbf{1}$ is a column vector of n ones.

In the point-line registration formulation the entries of Z can have any sign, whereas cheirality imposes that $Z \geq 0$ in the NPnP formulation.

3. Procrustean solution

The Procrustean solution minimizes the following geometric error:

$$\|ZPR + OR + \mathbf{1c}^\mathsf{T} - \alpha S\|_F^2. \tag{4}$$

This objective function is the sum of the squared norm of the difference vectors (Δ in Fig. 1) between reference 3D points (S) and the points belonging to the lines determined by Z (i.e., $ZP + O$) based on their estimated attitude, position and scale ($R, \mathbf{c}, \alpha$).

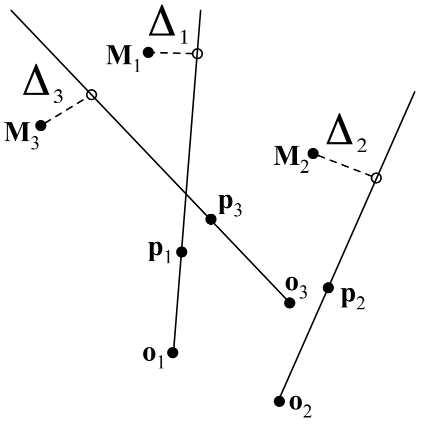


Figure 1. The estimated parameter ζ_i defines a 3D point (empty circle) along the line $(\mathbf{o}_i \mathbf{p}_i)$. The segment (perpendicular to the line) joining this point and the corresponding reference 3D point $\mathbf{M}_i$ is Δ. The position, attitude and scale of the set of rays are estimated in such a way to minimize the length of the Δs for all the points, in a least squares sense.

The minimization is accomplished with an alternating scheme (also called "block relaxation" [2]) as in the PPnP [4] case, where each variable is alternatively estimated while keeping the others fixed.

As a matter of fact, the Procrustean solution can be seen as iterating between two stages, namely:

- assuming Z is known, this fixes a point on each line and the problem becomes a point-point registration;
- given R, $\mathbf{c}$ and α, solve for Z by finding the position along the (fixed) line that minimizes the distance to the 3D points S.

Solving for $R, \mathbf{c}$ and α. When Z is known, Eq. (3) reduces to the standard EOPA model (see Appendix A) where one set of points is $B = S$ and the other one is $A = (ZP + O)$. Hence $R, \mathbf{c}, \alpha$ are computed according to the following formulas, that are instances of those reported in Appendix A:

$$R = U \mathrm{diag}\left(1,1,\det(UV^\mathsf{T})\right) V^\mathsf{T} \tag{5}$$

with

$$UDV^{\mathsf{T}} = (ZP+O)^{\mathsf{T}}\left(I - \mathbf{1}\,\mathbf{1}^{\mathsf{T}}/n\right)S; \tag{6}$$

$$\alpha = \frac{\operatorname{tr}((ZP+O)^{\mathsf{T}}\left(I - \mathbf{1}\,\mathbf{1}^{\mathsf{T}}/n\right)(ZP+O))}{\operatorname{tr}(R^{\mathsf{T}}(ZP+O)^{\mathsf{T}}\left(I - \mathbf{1}\,\mathbf{1}^{\mathsf{T}}/n\right)S)} \tag{7}$$

$$\mathbf{c} = (\alpha S - (ZP+O)R)^{\mathsf{T}}\mathbf{1}/n. \tag{8}$$

Solving for Z. When $R, \mathbf{c}$ and α are known, it is simply a matter of solving Eq. (3) for Z. Eq. (3) can be written as

$$ZP = (\alpha S - OR - \mathbf{1}c^{\mathsf{T}})R^{\mathsf{T}} \tag{9}$$

or equivalently

$$P^{\mathsf{T}}Z = Y \tag{10}$$

with $Y = R(\alpha S - OR - \mathbf{1}c^{\mathsf{T}})^{\mathsf{T}}$. Since Z is diagonal, it can be obtained as the solution of (see Appendix B):

$$(I \odot P^{\mathsf{T}})\operatorname{diag}^{-1}(Z) = \operatorname{vec}(Y) \tag{11}$$

where $\odot$ denotes the Khatri-Rao product, and diag^{-1} returns a vector containing the diagonal elements of its argument.

Any non-negativity constraint on Z must be enforced a-posteriori by clipping to zero negative values.

Please note that in the original formulation of PPnP given in[4], the solution for Z writes

$$\operatorname{diag}^{-1}(Z) = (I \circ PP^{\mathsf{T}})^{-1}\operatorname{diag}^{-1}(PY) \tag{12}$$

which (see Appendix B) corresponds to the least squares solution of $P^{\mathsf{T}}Z = Y$. However, since the Eq. (10) is exactly determined, the two formulations coincides.

The algorithm, henceforth dubbed gPPnP, is summarized in Algorithm 1.

Algorithm 1 gPPnP

```
Input: 3D points $S$ and lines $(O, P)$ (origin, direction)
Output: parameters $R$, $\mathbf{c}$ and $\alpha$ of the similarity transformation that aligns points and lines
  1. Start with $Z = \operatorname{diag}(\mathbf{1})$ (or any sensible guess);
  2. Compute $R = U\operatorname{diag}\left(1,1,\det(UV^{\mathsf{T}})\right)V^{\mathsf{T}}$
     with $UDV^{\mathsf{T}} = (ZP+O)^{\mathsf{T}}\left(I - \mathbf{1}\,\mathbf{1}^{\mathsf{T}}/p\right)S$;
  3. Compute $\alpha$ with (7);
  4. Compute $\mathbf{c} = (\alpha S - (ZP+O)R)^{\mathsf{T}}\,\mathbf{1}/n$;
  5. Compute $Z = \operatorname{diag}((I \odot P^{\mathsf{T}})^{-1}\operatorname{vec}(Y))$
     where $Y = R(\alpha S - OR - \mathbf{1}c^{\mathsf{T}})^{\mathsf{T}}$;
  6. Iterate over steps 2, 3, 4 until convergence.
```

Although being iterative, gPPnP always converged, in our experiment, to a global minimum of the cost function, starting with $Z = \operatorname{diag}(\mathbf{1})$.

4. Algebraic, direct solution

This direct solution can be regarded as a generalization of the PnP Fiore's algorithm derived in [3], from which it borrows the main idea. The Z values are computed directly via SVD, and the rest of the algorithm follows a standard EOPA solution.

Let us rewrite Eq. (3) as (w.l.o.g. let us assume $\alpha = 1$ for the moment) :

$$\begin{aligned} ZP + O &= (S - \mathbf{1}c^{\mathsf{T}})R^{\mathsf{T}} = \\ &= SR^{\mathsf{T}} - \mathbf{1}c^{\mathsf{T}}R^{\mathsf{T}} = [S, \mathbf{1}][R^{\mathsf{T}}, -c^{\mathsf{T}}R^{\mathsf{T}}] \end{aligned} \tag{13}$$

The matrix $[S, \mathbf{1}]$ has – in general – rank 4, hence it is rank deficient as soon as $n > 4$. Let us consider its SVD decomposition:

$$[S, \mathbf{1}]^{\mathsf{T}} = UDV^{\mathsf{T}} \tag{14}$$

and in particular let V_r contain the last $n-r$ columns of V corresponding to zero singular values. Hence $[S, \mathbf{1}]^{\mathsf{T}}V_r = 0$ and $V_r^{\mathsf{T}}[S, \mathbf{1}] = 0$ and also

$$V_r^{\mathsf{T}}(ZP+O) = 0 \tag{15}$$

from which Z can be recovered by solving the linear system of equations:

$$(P^{\mathsf{T}} \odot V_r^{\mathsf{T}})\operatorname{diag}^{-1}(Z) = -\operatorname{vec}(V_r^{\mathsf{T}}O) \tag{16}$$

where $\odot$ denotes the Khatri-Rao product [7], diag^{-1} returns a vector containing the diagonal elements of its argument.

The coefficient matrix has dimension $3(n-r) \times n$ and there are n unknowns. If points are in general position $r = 4$, hence at least 6 points are needed. If $n \geq 6$ the least squares solution in closed form is (see Appendix B):

$$\operatorname{diag}^{-1}(Z) = (PP^{\mathsf{T}} \circ V_rV_r^{\mathsf{T}})^{-1}\operatorname{diag}^{-1}(V_rV_r^{\mathsf{T}}OP^{\mathsf{T}}). \tag{17}$$

Once Z is known the parameters of the similarity transformation R, $\mathbf{c}$ and α are computed with EOPA (see Appendix A) as in Sec. 3. The algorithm, henceforth dubbed gFiore, is summarized in Algorithm 2.

5. Experiments

We run experiments on simulated data, obtained as follows. Line origins were generated randomly in the unit cube centred on zero; 3D points were randomly placed on a sphere of unit radius; line directions were computed as unit vectors from camera origins to 3D points. A random similarity transformation was applied to 3D points, where rotations is obtained from three random Euler angles, translation is a random vector with norm in $[0.5, 10]$ and scale is a random number between 0.1 and 10. Noise was added by perturbing line versors with a Gaussian random vector with

Algorithm 2 gFiore

```
Input: 3D points $S$ and lines $(O, P)$ (origin, direction)
Output: parameters $R$, $\mathbf{c}$ and $\alpha$ of the similarity transformation that aligns points and lines
  1. Obtain $V_r$ from the SVD of $[S, \mathbf{1}]$;
  2. Compute $Z$ with
     $\mathrm{diag}^{-1}(Z)=(PP^\mathsf{T} \circ V_r V_r^\mathsf{T})^{-1}\,\mathrm{diag}^{-1}(V_r V_r^\mathsf{T} O P^\mathsf{T})$;
  3. Compute $R = U\mathrm{diag}\left(1,1,\det(UV^\mathsf{T})\right)V^\mathsf{T}$
     with $UDV^\mathsf{T} = (ZP+O)^\mathsf{T}\left(I - \mathbf{1}\,\mathbf{1}^\mathsf{T}/p\right)S$;
  4. Compute $\alpha$ with (7);
  5. Compute $\mathbf{c} = (\alpha S - (ZP+O)R)^\mathsf{T}\,\mathbf{1}/n$.
```

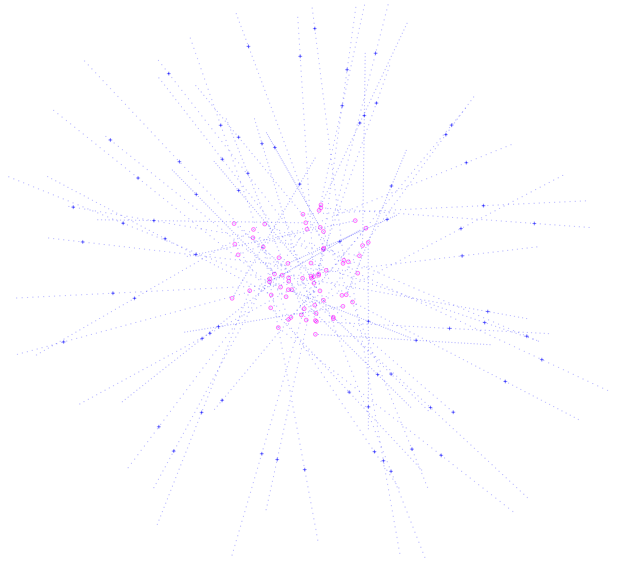


Figure 2. An instance of simulated scene, where magenta circles are the origins O and blue crosses are the 3D points S. Dotted lines join centres and points.

increasing standard deviation σ from 0 to 0.1 units. Every point of the graph is the average of 100 trials.

We evaluated the rotation error in degrees (similar results were obtained for the translation error) as a function of noise level and number of points.

We compared the two algorithms proposed in this paper, namely gFiore (Sec. 4) and gPPnP (Sec. 3) with gDLS[1], the only recent non-minimal solver for NPnP that deals with scale.

Results are reported in Fig. 3. The three algorithms have substantially equivalent performances, and the rotation error grows almost linearly with the noise standard deviation, and decreases with the number of points. However, when the noise level increases the differences becomes more noticeable and gPPnP achieves the lowest error, followed by gFiore.

Comparing running times is not feasible as gFiore and gPPnP are implemented in MATLAB, while gDLS in C++.

[1]The C++ code is available on the web as part of the Theia library [14].

6. Conclusions

The contribution of the paper is both theoretical and practical: on the theoretical side it provides a Procrustean formulation of the NPnP problem; on the practical side it describes two algorithms that solve it, one direct and one iterative, both very easy to implement (MATLAB code provided in Appendix C) and with state-of-the-art accuracy, as experiments have shown.

A. Extended Orthogonal Procrustes Analysis

The terms *Procrustes Analysis* (e.g. [5]) is referred to a set of least squares mathematical models used to compute transformations among corresponding points belonging to a generic k-dimensional space, in order to achieve their maximum agreement. In particular, the *Extended Orthogonal Procrustes Analysis* (EOPA) model allows to recover the least squares similarity transformation between two point sets.

Let us consider two matrices A and B containing two sets of numerical data, e.g., the coordinates of p-points of $\mathbb{R}^k$ by rows. EOPA allows to directly estimate the unknown rotation matrix R, a translation vector $\mathbf{t}$ and a global scale factor λ for which the residual:

$$\left\| B - \lambda A R - \mathbf{1}\mathbf{c}^\mathsf{T} \right\|_F^2 \tag{18}$$

is minimum, under the orthogonality condition: $R^\mathsf{T} R = RR^\mathsf{T} = I$.

The minimization proceeds by defining a Lagrangean function and setting the derivatives to zero (details can be found in [12]). In the following we report the main results.

The rotation is given by

$$R = U\mathrm{diag}\left(1,1,\det(UV^\mathsf{T})\right)V^\mathsf{T} \tag{19}$$

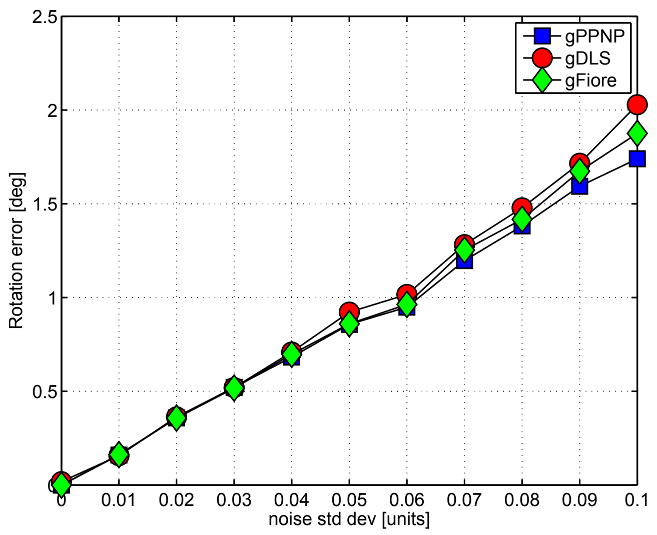
where U and V are determined from the SVD decomposition:

$$A^\mathsf{T}\left(I - \frac{\mathbf{1}\mathbf{1}^\mathsf{T}}{p}\right)B = UDV^\mathsf{T} \tag{20}$$

The $\det(UV^\mathsf{T})$ normalization guarantees that R is not only orthogonal but has positive determinant [18].

Then the scale factor can be determined with:

$$\lambda = \frac{\mathrm{tr}\left(R^\mathsf{T} A^\mathsf{T}\left(I - \frac{\mathbf{1}\mathbf{1}^\mathsf{T}}{p}\right)B\right)}{\mathrm{tr}\left(A^\mathsf{T}\left(I - \frac{\mathbf{1}\mathbf{1}^\mathsf{T}}{p}\right)A\right)} \tag{21}$$



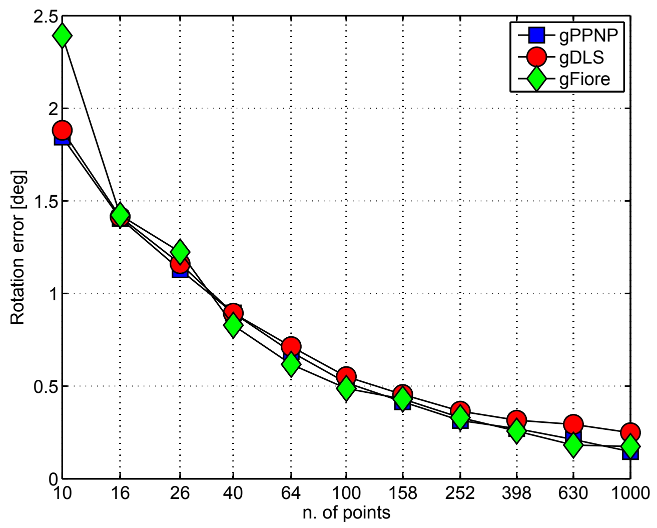


Figure 3. The rotation error in degrees is plotted as a function of the the noise level σ (on the left) with 64 points and the number of points (on the right) with $\sigma = 0.04$.

And finally the translation writes:

$$\mathbf{c} = (B - \lambda AR)^{\mathsf{T}} \mathbf{1}/p \tag{22}$$

To reconcile this notation with the one that is more customary in Computer Vision, it is sufficient to note that:

- points are represented by rows, hence linear operators (e.g., rotations) are represented by post-multiplication with a matrix;
- $A\mathbf{1}/p$ where A is $n \times p$ corresponds to taking the average of the rows;
- $A\left(I - \frac{\mathbf{1}\mathbf{1}^{\mathsf{T}}}{p}\right)$ has the effect of subtracting to A its rows average. The matrix $\left(I - \frac{\mathbf{1}\mathbf{1}^{\mathsf{T}}}{p}\right)$ is symmetric and idempotent.

B. Khatri-Rao product

The Khatri-Rao product [7], denoted by $\odot$, is in some sense a partitioned Kronecker product, where by default the column-wise partitioning is considered.

Let us consider two matrices A of order $p \times r$ and B of order $q \times r$ and denote the columns of A by $\mathbf{a}_1 \cdots \mathbf{a}_r$ and the those of B by $\mathbf{b}_1 \cdots \mathbf{b}_r$. The Khatri-Rao product is defined to be the partitioned matrix of order $pq \times r$:

$$A \odot B = [\mathbf{a}_1 \otimes \mathbf{b}_1, \cdots \mathbf{a}_r \otimes \mathbf{b}_r] \tag{23}$$

where $\otimes$ denotes the Kronecker product.

A useful property involves diagonal matrices and the vec operator, that transform a matrix into a vector by stacking its columns. If X is diagonal, then

$$\operatorname{vec}(AXB) = (B^{\mathsf{T}} \odot A) \operatorname{diag}^{-1}(X) \tag{24}$$

where diag^{-1} returns a vector containing the diagonal elements of its argument. Please contrast this with a similar property involving a general X and the Kronecker product.

With $B = I$ one obtains

$$\operatorname{vec}(AX) = (I \odot A) \operatorname{diag}^{-1}(X). \tag{25}$$

It it is easy to see that

$$(I \odot A) = \operatorname{blockdiag}(\mathbf{a}_1 \dots \mathbf{a}_n) \tag{26}$$

where $\mathbf{a}_1 \dots \mathbf{a}_n$ are the columns of A and blockdiag is the operator that construct a block diagonal matrix with its arguments as blocks.

The solution of $AXB^{\mathsf{T}}{=}Q$, with X diagonal is equivalent to solving:

$$(B \odot A) \operatorname{diag}^{-1}(X) = \operatorname{vec} Q \tag{27}$$

In the over-determined case the least squares solution is given by

$$\operatorname{diag}^{-1}(X) {=} \left((B{\odot}A)^{\mathsf{T}}(B{\odot}A)\right)^{-1}(B{\odot}A)^{\mathsf{T}} \operatorname{vec} Q \tag{28}$$

which is equivalent to this more compact formulation [10] with the Hadamard product, denoted by $\circ$:

$$\operatorname{diag}^{-1}(X) = \left(B^{\mathsf{T}}B \circ A^{\mathsf{T}}A\right)^{-1} \operatorname{diag}^{-1}(A^{\mathsf{T}}QB). \tag{29}$$

C. Matlab code

```
function [R t, a] = gPPnP(P,O,S,tol,pz)
% input
% P : matrix (nx3) of line versors
% O : matrix (nx3) of line origins
% S : matrix (nx3) 3D coordinates
% tol: exit tolerance
% pZ : positive Z flag
% output
% R: rotation matrix
% t: tanslation vector
% a: scale factor (only if required)

unit_scale = (nargout <3);
n = size(P,1); Z = diag(ones(1,n));
e = ones(n,1); II = eye(n)-((e*e')./n);
err = +Inf; E_old = 1000*ones(n,3);
D = kr(eye(n),P');

while err>tol
    [U,~,V] = svd((Z*P+O)'*II*S);
    R=U*[1 0 0; 0 1 0; 0 0 det(U*V')]*V';
    A = Z*P+O; AR=A*R;
    if unit_scale
        a=1;
    else
        a = trace((A)'*II*(A))/trace(AR'*II*S);
    end
    c = mean((a*S-AR),1)';
    Y = a*S - O*R - e*c';
    b = vec(R*Y'); v = D\b;
    if pz,  v(v<0) = 0; end
    Z = diag(v);
    E = Y-Z*P*R;
    err = norm(E-E_old,'fro'); E_old = E;
end
t = -R*c;
end

function [R t, a] = gFiore(P,O,S)
% same usage as gPPnP

unit_scale = (nargout <3);
n = size(P,1); Z = diag(ones(1,n));
e = ones(n,1); II = eye(n)-((e*e')./n);

M = [S,e]'; [~,~,V] = svd(M);
V2 = V(:,rank(M)+1:end);
D = ((P*P').* (V2*V2')); b=-diag(V2*V2'*O*P');
v = D\b; Z = diag(v);
[U,~,V] = svd((Z*P+O)'*II*S);
R=U*[1 0 0; 0 1 0; 0 0 det(U*V')]*V';
A = Z*P+O; AR=A*R;
if unit_scale
    a=1;
else
    a = trace((A)'*II*(A))/trace(AR'*II*S);
end
c = mean((a*S-AR),1)';
t = -R*c;
end
```

References

[1] Chu-Song Chen and Wen-Yan Chang. On pose recovery for generalized visual sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):848–861, 2004.

[2] Jan de Leeuw. Block-relaxation algorithms in statistics. In H. H. Bock, W. Lenski, and M. M. Richter, editors, *Information Systems and Data Analysis*, pages 308 – 325. Springer-Verlag, 1994.

[3] Paul D. Fiore. Efficient linear solution of exterior orientation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):140–148, 2001.

[4] V. Garro, F. Crosilla, and A. Fusiello. Solving the PnP problem with anisotropic orthogonal procrustes analysis. In *Second Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIM-PVT)*, pages 262–269, 2012.

[5] John C. Gower and Garmt B. Dijksterhuis. *Procrustes problems*. Oxford University Press, Oxford, UK, 2004.

[6] J.A. Hesch and S.I. Roumeliotis. A direct least-squares (DLS) method for PnP. In *Proc. of the International Conference on Computer Vision*, pages 383–390, 2011.

[7] C. G. Khatri and C. Radhakrishna Rao. Solutions to some functional equations and their applications to characterization of probability distributions. *Sankhyā: The Indian Journal of Statistics*, 30(2):167–180, 1968.

[8] L Kneip, P Furgale, and R Siegwart. Using multi-camera systems in robotics: Efficient solutions to the NPnP problem. In *Proc. of the IEEE International Conference on Robotics and Automation*, 2013.

[9] Laurent Kneip, Hongdong Li, and Yongduek Seo. UPnP: An optimal O(n) solution to the absolute pose problem with universal applicability. In *13th European Conference on Computer Vision*, pages 127–142, 2014.

[10] Hanoch Lev-Ari. Efficient solution of linear matrix equations with application to multistatic antenna array processing. *Communications in Information & Systems*, 05(1):123–130, 2005.

[11] David Nistèr. A minimal solution to the generalised 3-point pose problem. In *Computer Vision and Pattern Recognition, Conference on*, pages 560–567, 2004.

[12] Peter Schönemann and Robert Carroll. Fitting one matrix to another under choice of a central dilation and a rigid motion. *Psychometrika*, 35(2):245–255, 1970.

[13] Peter Sturm, Srikumar Ramalingam, Jean-Philippe Tardif, Simone Gasparini, and João Barreto. Camera models and fundamental concepts used in geometric computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6:1–183, 2011.

[14] Chris Sweeney. *Theia Multiview Geometry Library: Tutorial & Reference*. University of California Santa Barbara. http://www.theia-sfm.org/.

[15] Chris Sweeney, Victor Fragoso, Tobias Höllerer, and Matthew Turk. gDLS: A scalable solution to the generalized pose and scale problem. In *13th European Conference on Computer Vision*, pages 16–31, 2014.

[16] Roberto Toldo, Riccardo Gherardi, Michela Farenzena, and Andrea Fusiello. Hierarchical structure-and-motion recovery from uncalibrated images. *Computer Vision and Image Understanding*, 2015.

[17] Jonathan Ventura, Clemens Arth, Gerhard Reitmayr, and Dieter Schmalstieg. A minimal solution to the generalized pose-and-scale problem. In *Computer Vision and Pattern Recognition, Conference on*, 2014.

[18] Grace Wahba. A Least Squares Estimate of Satellite Attitude. *SIAM Review*, 7(3), July 1965.