Available online at www.sciencedirect.com
ScienceDirect
Computer Physics Communications 176 (2007) 694–700
Computer Physics Communications
www.elsevier.com/locate/cpc

Alignment of fiducial marks in a tomographic tilt series with an unknown rotation axis ☆

Zachary H. Levine *, Alex Volkovitsky [1], Howard K. Hung

National Institute of Standards and Technology, Gaithersburg, MD 20899, USA

Received 1 August 2006; accepted 24 January 2007

Available online 4 April 2007

Abstract

Alignment for tomography using a transmission electron microscopy frequently uses colloidal gold particles as fiducial reference marks. Typically, there is an implicit assumption that the tilt axis of the tomographic series is orthogonal to the beam direction. However, this may not be true, either intentionally, if a tilt-rotate stage is used, or unintentionally, because of mechanical errors in the rotation stage or the sample fixture.

Here, we provide a computer code which takes as input a set of two-dimensional (2D) observations of fiducial reference marks at various tilt angles and the values of those tilt angles. It produces as output a three-dimensional model of the observations, 2D shifts for each view, and the tilt axis direction.

Program summary

Title of program: particleTilt
Catalogue identifier: ADYW_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/ADYW_v1_0
Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland
Computers: IBM[2] compatible desktop PC; SGI[2] Octane
Operating system: Red Hat[2] WS 3 Linux (with 2.4.21-40.EL kernel); IRIX 6.5 IP30
Program language used: Fortran 90
No. of bits in a word: 32
No. of processors used: one
Has the code been vectorized: no
No. of lines in distributed program, including test data, etc.: 2397
No. of bytes in distributed program, including test data, etc.: 47 017
Distribution format: tar.gz
Peripherals used: one
Typical running time: 350 ms (larger included example, on 2.8 GHz 32-bit PC)
Nature of problem: The program is used to assist the alignment step in tomography. The samples should be prepared with spherical particles (typically gold beads) which are observed in several views. (Not every particle need be observed in every view.) The program reports coordinates of a 3D model of the particles as well as the direction of the tilt axis as a point on the unit sphere.
Method of solution: Our package minimizes an objective function whose free variables are a set of 3D model points and 2D shifts of the views as well as two parameters characterizing of tilt axis as a point on the unit sphere. The objective function is decomposed into a pure quadratic form

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: zlevine@nist.gov (Z.H. Levine).
[1] Present address: Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

[2] Commercial organizations identified in this paper are for the purpose of identification only and are not endorsed by NIST. The associated equipment or software identified is not necessarily the best available for the purpose.

0010-4655/$ – see front matter Published by Elsevier B.V.
doi:10.1016/j.cpc.2007.01.008

which encompasses the model points and shifts, and a more complicated form which has only two degrees of freedom. The Broyden–Fletcher–Goldfarb–Shanno algorithm is applied alternately and iteratively to minimize the objective function with respect to the two sets.
Published by Elsevier B.V.

PACS: 68.37.Lp; 06.90.+v

1. Introduction

Alignment is a critical phase of tomography on the micrometer or nanometer scale [1,2]. It contrast to, say, medical imaging, where it is possible to hold the samples and detectors fixed on the scale of the pixel (about 100 μm), this is impossible in microtomography where the pixels are about 1 nm for electron microscopy or 10 nm for X-ray microscopy. In practice, every image is subject to an arbitrary translation which must be removed by an analysis of the scene or real-time control of the instrument [3].

Solutions to the alignment problem either include or exclude fiducials. If no fiducials are present, correlation or moment techniques are used [4–6]. These require that the whole object be imaged in every view. Here, we will consider the case of fiducial marks which are usually implemented with colloidal gold particles [3,7,8]. In addition, we assume rigid-body motion of all the fiducials. The problem of radiation damage to a biological cell and the attendant plastic flow has been addressed within the IMOD package [7]. We also assume that the fiducial marks may be distinguished from one another in each image and identified with a single three-dimensional object.

Typically in tomography using a single-axis tilt series, the experimentalist reports the tilt angle as a single value. However, the direction of the tilt axis itself is not reported. It is typically assumed that the tilt axis is in the plane orthogonal to the beam direction [7]. However, this need not be the case [9]. In the present work, the tilt axis may be in any direction on the unit sphere (except aligned with the beam). To the best of our knowledge, there is no publicly available software in the case in which the beam and the tilt axis are not orthogonal.

2. Method

We seek a 3D model of the fiducials (as points in space) as well as 2D shifts of each image and the direction of the tilt axis as a point on the unit sphere. A suitable objective function is

$$F = \sum_{mj} \left(r_{mj} - \delta_{vj} - \sum_{ii'} P_{ji} R_{ii'}(\phi_v \hat{n}) X_{i'p} \right)^2, \tag{1}$$

where the symbols are defined in Table 1. What this objective function says is: consider the projection of each 3D particle onto every view in which it may be seen. (This is the third term in parenthesis.) We wish to minimize the squared deviation of the projected point from its observed location r_{mj} subject to the proviso that every image is subject to an unknown rigid shift δ_{vj}. To clarify the minimization problem, we write

$$F = F(X, \delta, \hat{n}; \phi, r), \tag{2}$$

Table 1
Symbols used herein. Primed symbols have the same meaning as the initial one, the prime meaning "another instance of". If all particles were present in all the views, m would simply be $v \times p$. In our case, some particles are allowed to be missing, so m determines both v and p, i.e. implicitly $v = v(m)$ and $p = p(m)$. Note that $N_m \leqslant N_p N_v$

i	3D Cartesian index	index 1 to 3
j	2D Cartesian index	index 1 to 2
v	index of the views (or, equivalently, tilts)	index 1 to N_v
p	index of the fiducial points (taken to be objects in 3D)	index 1 to N_p
m	combined view and particle index	index 1 to N_m
r_{mj}	coordinates of the fiducial marks as observed	known
P_{ji}	projection matrix	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$
$R_{ii'}$	rotation matrix	determined by $\phi_v \hat{n}$
ϕ_v	tilt angles	known
$\hat{n}$	tilt axis	2 unknowns
X_{ip}	coordinates of the fiducial marks	$3N_p$ unknowns
δ_{vj}	shifts of each image	$2N_v$ unknowns
F	objective function	dependent variable

where the symbols appearing without their subscripts are to be understood as arrays whose components are to be determined and the arguments after the semicolon are fixed for a given problem. (In the case of ϕ, these are independent variables in the measurement; for r, these are the dependent variables.) Note that the objective function is invariant under $X_{ip} \to X_{ip} - \Delta_i$, $\delta_{vj} \to \delta_{vj} + \sum_{ii'} P_{ji} R_{ii'}(\phi_v \hat{n}) \Delta_{i'}$. There is sufficient flexibility in the co-ordinate systems of the views to accommodate a translation of the 3D model. In practice, we will minimize the objective function and then (optionally) normalize the solution at the end. We may set $X_{ip} = 0$ for any one particle indexed by p and all three Cartesian directions indexed by i, or set the centroid to the origin via $\sum_p X_{ip} = 0$.

The rotation matrix $R(\phi_v \hat{n})$ is conveniently computed from the Rodrigues parameters $\phi_v \hat{n}$ for each v [10]. We construct an antisymmetric matrix J

$$J(\vec{\omega}) = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{3}$$

Then the rotation matrix is given by

$$R(\vec{\omega}) = e^{J(\vec{\omega})}, \tag{4}$$

where $\vec{\omega}$ is the generator of the rotation [10]. (As is standard, vectors are denotes as $\vec{\omega}$; their magnitude is denoted as $\omega = |\vec{\omega}|$; and their associated unit vectors are denoted as $\hat{\omega} = \vec{\omega}/\omega$.) The matrix exponential need not be performed explicitly to construct the rotation matrix [10]. Instead,

$$R(\omega) = I + \sin(\omega) J(\hat{\omega}) + 2 \sin^2\left(\frac{\omega}{2}\right) J^2(\hat{\omega}), \tag{5}$$

where I is the identity matrix.

Whereas the Rodrigues parameterization is sufficiently rich to carry one 3D co-ordinate system into another (as are the matrices constructed from Euler angles), the problem here is simpler. We are concerned with describing a 2D tilt axis (as a direction on the unit sphere). Hence, it suffices to consider a two-dimensional space of generators with $\omega_z = 0$. The tilt axis is described in terms of a generator of a rotation

$$\hat{n} = e^{J(\vec{\omega})}\hat{z}, \tag{6}$$

where $\hat{z}$ is the unit vector in the Cartesian z direction. If $\hat{n}$ is the unit vector characterized by θ and φ in polar co-ordinates, i.e.

$$\hat{n} = \sin\theta\cos\varphi\hat{x} + \sin\theta\sin\varphi\hat{y} + \cos\theta\hat{z}, \tag{7}$$

then

$$\vec{\omega} = -\theta\sin\varphi\hat{x} + \theta\cos\varphi\hat{y}. \tag{8}$$

Following Taylor and Kriegman [11], we will minimize with respect to $\vec{\omega}$, with $\omega_z = 0$, instead of using alternative descriptions such as the polar angles of the tilt axis or Euler angles which have problems of discontinuities. Eq. (5) is used at two distinct stages in the code: once to find the tilt axis in terms of the parameter $\vec{\omega}$ itself and a second time with $\phi_v\hat{n}$ playing the role of $\vec{\omega}$ to generate the rotation matrices $R(\phi_v\hat{n})$.

With the tilt axis direction fixed, the objective function is a quadratic function of the other variables. Moreover, we anticipate all of the eigenvalues of the Hessian of the objective function will be positive except for three zero eigenvalues corresponding to the arbitrary center of coordinates of the particles. Rapidly convergent algorithms such as the one due to Broyden–Fletcher–Goldfarb–Shanno (BFGS) exist for this case [12]. (The presence of the zero eigenvalues at the minimum leads the algorithm to converge to an arbitrary point in the 3D space of minima. However, as discussed above, the solutions represent translations of the model and a normalization can be imposed.) With the particle coordinates and the shifts fixed, the objective function may be expanded in spherical harmonics with $\ell \leqslant 4$ (i.e. is limited to a fourth-degree polynomial in the cosine of the polar angle and the sine and cosine of the azimuthal angle). Early attempts to minimize all variables jointly were dismal failures. However, the following algorithm emerged:

Set $\hat{n}$ from input.
$X_{ip} = 0$
$\delta_{vj} = 0$
Loop {
    minimize $F(X_{ip}, \delta_{vj}; \hat{n}(\vec{\omega}))$
    update X_{ip} and δ_{vj} for all (i, p) and (j, v)
    minimize $F(\hat{n}(\vec{\omega}); X_{ip}, \delta_{vj})$
    update $\hat{n}$
}

Arguments to the right of the semicolon are treated as parameters. The parameters fixed for the whole problem, r_{mj} and ϕ_v are suppressed. In practice, although the second minimization is over a somewhat complicated function, it seems to have two minima. One corresponds to the rotation axis. A second corresponds to its negative. We have not explored this in detail. The user is required to specify an initial guess for the tilt axis which has some positive projection along the correct solution.

To converge, there must be at least as many input constraints as degrees of freedom. The former are the number of observations $N_m \approx 2N_pN_v$, where the approximate equality assumes there is not too much missing data, and the 2 comes from the 2 co-ordinates per observation. Excluding translational invariance, there are $3(N_p - 1) + 2N_v + 2$ degrees of freedom. If N_m exceeds the number of degrees of freedom only slightly, the algorithm converges slowly. If there are about twice as many constraints as degrees of freedom, the objective function is reduced by approximately a factor of 2 per iteration.

The work in this paper is similar to that of Díez et al. [9]. They also propose an algorithm, with an experimental implementation in their case, for determining the extent to which a tilt axis is not orthogonal do the direction of projection. They use 1-parameter to describe the direction of the tilt axis, allow for an additional rotation of each image but do not allow for a rigid shift of each image, allow for a magnification of each image, and they require every particle to be present in every view. In the present work, we use two-parameters to describe the direction of the tilt axis, we do not allow for additional rotation of each image but do allow for a rigid shift of each image, do not allow for magnification of each image, and do not require every particle to be present in every view. Technically, our algorithms are similar in that our objective functions are very similar, and we both split out the minimization objective function with respect to the angular variable(s) from the other variables. We differ in that we use the Rodrigues parameterization for the rotation matrices whereas they use the Euler form.

3. Implementation details

We have implemented the code in Fortran 90. The principle language features of Fortran 90 which we used are dynamic storage allocation, modules, and implicit loop on array indices. The main program ("particleTiltMain.f90") directs the input to be read, the problem to be solved, and the solution to be printed. The principle module ("findTiltSolnMod.f90") is only loosely coupled to the main program. It would be relatively simple to use the principle module in another program so the user is not necessarily dependent on the interface described below. This module does not read or write any files (other than some messages about the solution progress which are not essential to its operation). A companion program, "particleTiltDataGen.f90", generates test data for "particleTiltMain.f90" based on a known 3D model, shifts, tilt angles, and tilt axis. It also allows Gaussian random noise to be added to the projected positions using the ziggurat method [13]. The main code relies on the routine "lbfgs.f" [12]. The companion code uses random number generator from the code "ziggurat.f90" [13].

The executable is created by invoking "make" after the file is unzipped (on some systems, using the command "gunzip particleTilt.tar.gz") and untarred (on some systems, using the command "tar -xlf particleTilt.tar"). The user's system should have a compiler whose name is "f90", or the variable "FORT"

in the file "Makefile" may be set to the name of the Fortran 90 compiler.

3.1. User interface

The main code, whose executable is "particleTilt.x", requires three fixed-name files: "tilt.in", "particle.in", and "control.in" all of which are read by subroutines in the module "particleTiltInMod.f90". The file "tilt.in" contains

fmtGot the number 20060614 which serves as a file identifier (i).
nTilt the number of tilt angles in the file (i).
tiltDeg the tilt angles in degrees (nTilt*f).

The symbols in parenthesis give the number and type of variables with i for integer and f for real. The input is read with free format conventions. Each item may be on one or more lines, but two different items may not share a line. The file "particle.in" contains

fmtGot the number 20060209 (i).
nm the number of observations (i).
pp(m), **vv(m)**, **rObs(:,m)** the particle index, the view index, and the 2D position of each observation [(2*i,2*f), for m = 1, ..., nm].

The 1-based indices pp and vv may be in any order, but it is assumed that every particle appears in at least 1 observation and every view has at least one particle in it. The file "control.in" contains

fmtGot the number 20060621 (i).
nHat the initial estimate of the tilt axis direction (3*f).
convFactor if the objective function is reduced by this factor from the initial guess, the program halts (f).
nIterOuter if this many iterations of the loop described in the method section occurs, the program halts (i).
iShift normalization convention for the solution. -1: omit normalization; 0: set $\sum_p X_{ip} = 0$; $1 \leqslant p \leqslant N_p$: set $X_{ip} = 0$.
mBFGS parameter which controls how large the Hessian matrix is in the lbfgs algorithm; see comments in lbfgs.f; recommended value: 7.
iPrint print control; see comments in lbfgs.f (2*i).
epsBFGS controls how close to 0 the gradient must be for BFGS to be considered converged; see comments in lbfgs.f; recommended value: 1.e–5 (f).
xTolBFGS estimate of machine tolerance; see comments in lbfgs.f; recommended value: 1.e–15 (f).

In addition to the human-readable output file generated to the standard output, the solution is summarized using free-format write statements in its own file called "particleTiltSoln.out" which may be read by a program written by the user, possibly in another language. See "subroutine prtSoln" in "findParticleTiltSoln.f90" for details.

The companion code, whose executable is "particleTiltDataGen.x", requires one input file called "particleTiltDataGen.in". The file contains

fmtGot the number 20060622 (i).
seed a seed for the random number generator (i).
sigma the standard deviation of the observations (i).
uHatUnnorm the direction of the tilt axis (an unnormalized vector used as a unit vector) (3*f).
np the number of particles (i).
xPart the co-ordinates of the particle (3*np*f); the 3D Cartesian index is the fast index.
nv the number of views or, equivalently, tilt angles.
tiltDeg the tilt angles in degrees (nv*f).
deltaView the shifts added to each view (2*nv*f); the 2D Cartesian index is the fast index.
nSkip number of observations to skip (can be 0).
pSkip(iSkip), **vSkip(iSkip)** particle and view indices to omit (2*i, for iSkip = 1, ..., nSkip).

The variable sigma tells the standard deviation of the zero-mean Gaussian noise to be added to the observed particle positions. It may be set to 0. The companion code produces the files "tilt.in" and "particle.in" as output. The companion code may be used to generate test data for the main code. Of course, the main code may take observed data as input. In particular, if the IMOD code [7] is used, model positions generated using the command "imodinfo -a modelFile" can be converted to the format required by "particle.in" with a small script, perhaps written in a language such as AWK or Perl.

3.2. Code structure

Only the main program "particleTiltMain.f90" is described in this section, the companion program "particleTiltDataGen.f90" being fairly simple. The modules of "particleTiltMain.f90" are

constantMod.f90 Contains global constants.
findTiltSolnMod.f90 Contains the heart of algorithm as well as code for finding the objective function and its gradient which is contained in the included file "objFtn.f90" which also includes code to normalize the solution. Code to print the solution is also included, but is invoked by the main program.
lbfgs.f Subroutine for BFGS solution (by J. Nocedal).
particleTiltInMod.f90 Module for data reading.
particleTiltMain.f90 Main program. Calls for data reading, solution, and printing of solution.
printMod.f90 Formats array printing.
rodriguesMod.f90 Contains code to generate rotation matrices from their Rodrigues parameters.
vecMod.f90 Contains low-level routine for manipulation of vectors, including one which uses the Rodrigues representation.

4. Test runs

4.1. Test case 1: 4 particles

The first test case places particles at the origin and $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ and the tilt axis in the $\hat{x}$ direction. These particles are viewed at 5 tilt angles, whose values are given in "tilt.in".

```
20060614 fmtGot
5        nTilt then tiltDeg
0.
90.
180.
270.
45.
```

The words each the input files are not read by the program; they are in effect comments.

Next "particle.in" is presented.

```
20060209   fmtGot
20         nv then pp vv rObs
1 1  0.  0.
1 2  0.  0.
1 3  0.  0.
1 4 10.  0.
1 5  0.  0.
2 1  1.  0.
2 2  1.  0.
2 3  1.  0.
2 4 11.  0.
2 5  1.  0.
3 1  0.  1.
3 2  0.  0.
3 3  0. -1.
3 4 10.  0.
3 5  0.  0.7071067811865475244
4 1  0.  0.
4 2  0. -1.
4 3  0.  0.
4 4 10.  1.
4 5  0. -0.7071067811865475244 0.
```

The view at the fourth tilt angle is shifted by $(10, 0)$. Note $1/\sqrt{2} \approx 0.707\ldots$.

Finally, "control.in" contains

```
20060621    fmtGot
+1. +1. +1. nHat
1.e-20      convFactor
270         nIterOuter
+1          iShift (-1: no shift, 0: avg to
                origin, 1:np, ip to origin)
7           mBFGS
1 1         iPrint (see lbfgs.f90 comment)
1.d-07 epsBFGS (see lbfgs.f90 comment)
1.d-15 xTolBFGS (see lbfgs.f90 comment)
```

The program prints to standard output. The final few lines include

```
findTiltSoln: iIterOuter 179 funcVal after
   omega minimization 4.00067E-18
Tilt axis direction cosines 1.00000 0.00000
   0.00000

Particle 1  0.00000 0.00000 0.00000
Particle 2  1.00000 0.00000 0.00000
Particle 3  0.00000 1.00000 0.00000
Particle 4  0.00000 0.00000 1.00000
Shift 1  0.00000 0.00000
Shift 2  0.00000 0.00000
Shift 3  0.00000 0.00000
Shift 4 10.00000 0.00000
Shift 5  0.00000 0.00000
```

4.2. Test case 2: 11 particles

A more realistic test case includes 11 particles viewed in 20° intervals between ±70°. The positions of the particles were generated as random triples of integers from 0 to 9, except for the final particle which was placed at the origin. In addition, each particle was omitted from some view, chosen at random. The input files "particle.in" and "tilt.in" were generated by "particleTiltDataGen.x" from its input file "particleTiltDataGen.in" which contains

```
20060622       fmtGot
12345          seed
0.             sigma
0.999949999 0. 0.01 uHatUnnorm
11             np
8 3 0
9 4 8
9 5 4
9 9 4
7 2 8
7 1 3
1 6 2
0 7 2
4 7 7
2 1 9
0 0 0          xPart
8              nv
-70.
-50.
-30.
-10.
+10.
+30.
+50.
+70.           tilt
4 1
4 4
0 3
-1 4
1 -3
2 -4
2 2
3 -3           deltaView
0              nSkip
1 1
```

```
2 6
3 3
4 3
5 1
6 2
7 2
8 5
9 1
10 4
11 4
```

The file "control.in" contains

```
20060621
+1. +1. +1. nHat
1.e-20      convFactor
125         nIterOuter
11          iShift (-1: no shift, avg to
0:               origin, 1:np, ip to origin)
7           mBFGS
-1 0        iPrint (see lbfgs.f90 comment)
1.d-05      epsBFGS (see lbfgs.f90 comment)
1.d-15      xTolBFGS (see lbfgs.f90 comment)
```

The output file has the following lines near the end:

```
findTiltSoln: iIterOuter 102 funcVal
   after omega minimization 8.48246E-17
Tilt axis direction cosines 0.99995 0.00000
   0.01000
Particle  1  8.00000 3.00000 0.00000
Particle  2  9.00000 4.00000 8.00000
Particle  3  9.00000 5.00000 4.00000
Particle  4  9.00000 9.00000 4.00000
Particle  5  7.00000 2.00000 8.00000
Particle  6  7.00000 1.00000 3.00000
Particle  7  1.00000 6.00000 2.00000
Particle  8  0.00000 7.00000 2.00000
Particle  9  4.00000 7.00000 7.00000
Particle 10 2.00000 1.00000 9.00000
Particle 11 0.00000 0.00000 0.00000
Shift 1  4.00000  1.00000
Shift 2  4.00000  4.00000
Shift 3  0.00000  3.00000
Shift 4 -1.00000  4.00000
Shift 5  1.00000 -3.00000
Shift 6  2.00000 -4.00000
Shift 7  2.00000  2.00000
Shift 8  3.00000 -3.00000
```

It is important to recall that the program works with real numbers and the integers are chosen only to make it easier for the reader to verify the input and final output agree. Note also that the 10 mrad out-of-plane tilt axis was accurately found by the program, admittedly working with noise-free data. The PC computer (32-bit, 2.8 GHz) required 350 ms to do the larger example. Although an example is not provided here, the program degrades gracefully in the presence of noise if the number of redundant constraints is substantial (e.g., at least 50 % more than the minimum).

5. Concluding remarks

The code presented here has two applications within tomography using a transmission electron microscope. In the case in which the tilt axis is supposed to be orthogonal to the beam direction, this code enables the user to estimate whether a particular set of observations is consistent with such an assumption. It is important to be aware of the possibility of a nonorthogonal tilt axis because a code such as IMOD [7] might misinterpret the effects of a nonorthogonal tilt axis as a plastic flow [9].

Second, a tilt series might be acquired with the tilt axis intentionally nonorthogonal to the beam direction. This could be implemented with a tilt-rotate holder. If the sample were held at a fixed angle (say 45°) from the beam direction, and rotated about the azimuth of the holder, a slab-shaped sample would have a constant thickness in contrast to the usual case in which the thickness varies as the reciprocal of the cosine of the tilt angle [8]. Three-dimensional Bayesian codes for tomography are becoming available to process such data [14,15].

Acknowledgements

The authors gratefully acknowledge correspondence with David Mastronarde and funding from the NIST Director and the NIST Office of Microelectronic Programs.

References

[1] V. Lucic, F. Forster, W. Baumeister, Structural studies by electron tomography: From cells to molecules, Ann. Rev. of Biochem. 74 (2005) 833–865.
[2] R. McIntosh, D. Nicastro, D. Mastronarde, New views of cells in 3d: and introduction to electron tomography, Trends in Cell Biol. 15 (2005) 43–51.
[3] U. Ziese, A.H. Janssen, J.L. Murk, W.J.C. Geerts, T. Van der Krift, A.J. Verkleij, A.J. Koster, Automated high-throughput electron tomography by pre-calibration of image shifts, J. Microscopy-Oxford 205 (2002) 187–200 (Part 2).
[4] C.H. Owen, W.J. Landis, Alignment of electron tomographic series by correlation without the use of gold particles, Ultramicroscopy 63 (1996) 27–38.
[5] S. Lanzavecchia, F. Cantele, P. Luigi, Alignment of 3d structures of macromolecular assemblies, Bioinformatics 17 (2002) 58–62.
[6] H. Winkler, K.A. Taylor, Accurate marker-free alignment with simultaneous geometry determination and reconstruction of tilt series in electron tomography, Ultramicroscopy 106 (2006) 240–254.
[7] J.R. Kremer, D.N. Mastronarde, J.R. McIntosh, Computer visualization of three-dimensional image data using imod, J. Struct. Biol. 116 (1996) 71–76; see also http://bio3d.colorado.edu/imod/.
[8] H.B. Zhang, C. Yang, A. Takaoka, Measuring the top-bottom effect of a tilted thick specimen in an ultrahigh-voltage electron microscope, Rev. Sci. Inst. 76 (2005) 056106.
[9] D.C. Díez, A. Seybert, A.S. Frangakis, Tilt-series and electron microscope alignment for the correction of the non-perpendicularity of beam and tilt-axis, J. Struct. Biol. 154 (2006) 195–205.
[10] S.L. Altmann, in: Rotations, Quaternions, and Double Groups, Clarendon, New York, 1986, p. 74ff.
[11] C.J. Taylor, D.J. Kriegman, Yale University Technical Report No. 9405, 1994; http://www.cis.upenn.edu/~cjtaylor/.
[12] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Programming (Ser. B) 45 (3) (1989) 503–528; J. Nocedal, lbfgs.f90. We have included the version lbfgs from netlib, http://www.netlib.org to avoid licensing issues. A more advanced ver-

sion, L-BFGS-B, may be obtained for certain purposes at http://www.ece.northwestern.edu/~nocedal/lbfgs.html.

[13] G. Marsaglia, W.W. Tang, The ziggurat method for generating random variables, J. Statist. Software 5 (8) (2000); http://www.jstatsoft.org/v05/i08;
Fortran 90 code by A. Miller http://www.netlib.org/random/ziggurat.f90.

[14] Z.H. Levine, A.J. Kearsley, J.G. Hagedorn, Bayesian tomography for projections with an arbitrary transmission function with an application in electron microscopy, J. Res. Nat. Inst. of Stand. and Technol. 111 (2006) 411–417.

[15] K. Thielemans, Software for Tomographic Image Reconstruction, http://stir.hammersmithimanet.com/, Version 1.4, 2006.