A faster solution to the sliding semilandmarks equation

(updated)

According to Gunz (2005) and Gunz et al. (2005), the weighted least squares equation which minimizes the bending energy is:

$$-(\mathbf{U}^T\mathbf{L}_k^{-1}\mathbf{U})\mathbf{T} = \mathbf{U}^T\mathbf{L}_k^{-1}\mathbf{Y}^0 \quad (1)$$

This can be written more compactly as:

$$\mathbf{MT} = \mathbf{B} \quad (2)$$

where **T** are the unknown sliding parameters.

Let k be the total number of landmarks (in 3-D) and m the total number of sliding semilandmarks. We can consider three types of semilandmarks according to their directional constraints when sliding: a) sliding on curves, requiring 1 directional constraint vector (u, with direction cosines u^x, u^y, u^z), b) sliding on surfaces, requiring 2 directional vectors (u and v), and c) 'free' semilandmarks, not sliding on any surface or curve, but moving according to the thin-plate spline warp field. The last case can be considered a very general case of 'semilandmark' and needs three directional vectors (u, v, w), which can be set to correspond to the x, y and z axes of our coordinate system.

The **U** matrix has the following structure:

$$\left[\begin{array}{ccc:ccc:ccc}
u^x_{1,1} & \cdots & u^x_{1,m} & v^x_{1,1} & \cdots & v^x_{1,m} & w^x_{1,1} & \cdots & w^x_{1,m} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
u_{k,1} & \cdots & u_{k,m} & v^x_{k,1} & \cdots & v^x_{k,m} & w^x_{k,1} & \cdots & w^x_{k,m} \\
\hdashline
u^y_{1,1} & \cdots & u^y_{1,m} & v^y_{1,1} & \cdots & v^y_{1,m} & w^y_{1,1} & \cdots & w^y_{1,m} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
u^y_{k,1} & \cdots & u^y_{k,m} & v^y_{k,1} & \cdots & v^y_{k,m} & w^y_{k,1} & \cdots & w^y_{k,m} \\
\hdashline
u^z_{1,1} & \cdots & u^z_{1,m} & v^z_{1,1} & \cdots & v^z_{1,m} & w^z_{1,1} & \cdots & w^z_{1,m} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
u^z_{k,1} & \cdots & u^z_{k,m} & v^z_{k,1} & \cdots & v^z_{k,m} & w^z_{k,1} & \cdots & w^z_{k,m}
\end{array}\right] \quad (3)$$

Therefore, in the general case of all three semilandmark types, the **U** matrix has size $3k \times 3m$. It follows that the **M** matrix has size $3m \times 3m$ and is a symmetric matrix. Therefore, system (2) should be solvable as a system of $3m$ linear equations with $3m$ unknowns. Unfortunately, it is not possible to arrive at a solution because, in the general case, some of the rows (and columns) of **M** are zero, and therefore correspond to an equation of the type:

$$0x_1 + 0x_2 + 0x_3 + \cdots = 0$$

Such equations do not provide information for solving the system, which, actually has fewer equations than unknowns and can only be solved by the computationally expensive process of the pseudo-inverse.

Surprisingly (well, maybe not), the solution is simply to remove the zero rows and columns of the **M** matrix (and the corresponding rows of **B**). We thus arrive at a much smaller system of equations, which is well-behaved and can easily be solved as described below. The unknowns that are removed during this process are not needed

because we require them to be zero in any case, since they correspond to directional constraints that do not apply to the particular semilandmark type. To make this more apparent, consider the structure of **U** matrix again. In the case of many curve semilandmarks and only one surface semilandmark, the *v* columns will all be zero except for the one corresponding to the surface semilandmark. These zero *v* columns are not needed for specifying the sliding of the curve semilandmarks, and can therefore be removed. Let:

m1, the total number of curve, surface, and 'free' semilandmarks (i.e. with at least one directional constraint),

m2, the total number of surface, and 'free' semilandmarks (i.e. with at least two directional constraints),

m3, the total number of 'free' semilandmarks (three directional constraints).

Thus:

$m1 \geq m2 \geq m3$

Then, the **U** matrix can be written as:

$$\left[\begin{array}{ccc:ccc:ccc} u^x_{1,1} & \cdots & u^x_{1,m1} & v^x_{1,1} & \cdots & v^x_{1,m2} & w^x_{1,1} & \cdots & w^x_{1,m3} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u_{k,1} & \cdots & u_{k,m1} & v^x_{k,1} & \cdots & v^x_{k,m2} & w^x_{k,1} & \cdots & w^x_{k,m3} \\ \hdashline u^y_{1,1} & \cdots & u^y_{1,m1} & v^y_{1,1} & \cdots & v^y_{1,m2} & w^y_{1,1} & \cdots & w^y_{1,m3} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u^y_{k,1} & \cdots & u^y_{k,m1} & v^y_{k,1} & \cdots & v^y_{k,m2} & w^y_{k,1} & \cdots & w^y_{k,m3} \\ \hdashline u^z_{1,1} & \cdots & u^z_{1,m1} & v^z_{1,1} & \cdots & v^z_{1,m2} & w^z_{1,1} & \cdots & w^z_{1,m3} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u^z_{k,1} & \cdots & u^z_{k,m1} & v^z_{k,1} & \cdots & v^z_{k,m2} & w^z_{k,1} & \cdots & w^z_{k,m3} \end{array}\right] \quad (4)$$

Total size of **U** is now $3k \times (m1+m2+m3)$.

Size of **M** is $(m1+m2+m3) \times (m1+m2+m3)$. **M** is now symmetric and positive-definite and the system can be solved by a Cholesky decomposition and two back-substitutions, as follows:

$$\mathbf{M} = \mathbf{L}\mathbf{L}^T \quad (5)$$

This requires $O(n^3)$ multiplications.

Then we solve (2) by solving:

$$\mathbf{L}\mathbf{y} = \mathbf{B} \quad (6)$$

$$\mathbf{L}^T\mathbf{T} = \mathbf{y} \quad (7)$$

Equations (6) and (7) can be easily solved by back-substitution because **L** and $\mathbf{L}^T$ are triangular.

Implementation details

U is stored as a sparse matrix using a vector array of size $(m1+m2+m3)$ that contains each column of **U** as an entry of the form:

- row index (integer): equal to the row to which u^x corresponds, which is equal to the *k* index of the semilandmark.
- u^x, u^y, u^z (float): the directional constraint vector (directional cosines).

Matrix multiplications in (1) are performed by taking advantage of the sparsity of **U**. Careful bookkeeping of pointers and indices can eliminate the need to build $\mathbf{L}_k^{-1}$ as a 3×3 block matrix of L_k^{-1} and zero, and L_k^{-1} can be used directly.

The code has been implemented in Delphi (Pascal). Total run time (including matrix multiplications, Cholesky decomposition and back-substitution solving) on a Pentium IV 3GHz for sliding 401 semilandmarks (125 curve and 276 surface semilandmarks) is approximately 700 msec compared to more than 60 sec for calculating the pseudo-inverse. However, the pseudo-inverse calculation is not optimized because it does not take advantage of the symmetric nature of **M**. Using R, the pseudo-inverse of **M** is calculated in approximately 7 sec.

Previously it was required to use very small values for the directional constraints that are zero, because zero values could cause problems in the calculations (division by zero or numerical instability). This is no longer a problem, because the zero directional constraints do not appear in the matrices anymore. The smaller size of **U** makes all matrix multiplications faster and also reduces storage requirements.

Many thanks to Philipp Gunz for invaluable advice and to Michael Coquerelle for providing the incentive and for validating the computations using R.

References

Gunz P. Statistical and geometric reconstruction of hominid crania: Reconstructing the ontogeny of Australopithecines. PhD Thesis, Vienna 2005.

Gunz P, Mitteroecker P, Bookstein FL. Semilandmarks in 3D. In: Slice DE (editor). Modern Morphometrics in Physical Anthropology. Kluwer Academic, Plenum Publishers 2005.

Cholesky decomposition. http://en.wikipedia.org/wiki/Cholesky_decomposition.

Triangular matrix. http://en.wikipedia.org/wiki/Lower_triangular_matrix.

Demetris Halazonetis, 20th July 2008