

Updating a Geographic Database

A Rubber-Sheet Adjustment Algorithm

Leendert van Gastel
Harry Uitermark

A rubber-sheet algorithm for warping is used to update a geographic database without altering points that are already known in the database. *Mathematica* allows a quick experimental validation of the theory and of existing FORTRAN legacy code.

■ Introduction

The number of geographic databases is increasing rapidly in many domains, varying from municipal management to tourism and archeology: anyone managing a territory needs their own geographic database [1].

Geographic databases form the data part of geographic information systems (GIS). At the Dutch Cadastre (Land Registration) we have built our own GIS, which is primarily a geographic database with houses and ownership boundaries.

In the beginning, paper maps were digitized as the data source for this geographic database. Now that all existing paper maps are in the database, it is important to keep the database up to date. For example, when a new house is built, the database will be updated (Figure 1). Central to this updating is that we do not want to change the coordinates of the information that is already in the geographic database.

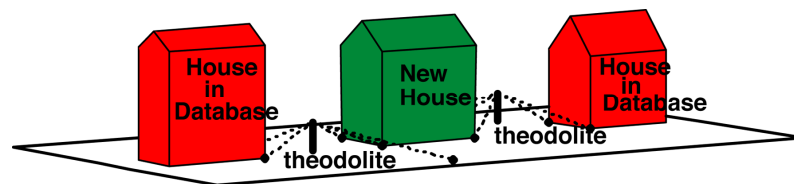


Figure 1. Measuring a new house (the green one) using a theodolite from two different positions.

However, we sometimes did not trust the results of existing update programs, which were coded in FORTRAN. We wondered if our formulae were correctly implemented, or were wrong from the beginning. So we checked our formulae by implementing them in *Mathematica* before checking the FORTRAN code. In the end the formulae was proven right and we found a bug in the FORTRAN code.

In this article we use this implementation to demonstrate how new information is added to the geographic database.

■ Warping

When collecting new information—most of the time using a theodolite, which is a surveying instrument that measures bearings and distances—all observations are done in a redundant way. This redundancy makes it possible to check the observations for errors. But at the same time, due to the stochastic nature of measurements, small discrepancies are introduced. So before adding new information to the database, we check our measurements for errors and filter out the small discrepancies by least squares methods. A statistical test is performed to ensure the quality of the data: the covariance of the measurements should be within standardized bounds.

A second stage, called *warping*, fits this internal local computation into the external global world of the database. There are several methods in use for warping. We mention:

- A similarity transformation, where the coefficients are determined by interpolation
- A quadratic bilinear extension
- An elastic rubber-sheet transformation [2]

At the Dutch Cadastre, we chose the last method because it deals best with the information already in the database. We will illustrate this later with an example.

■ Rubber-Sheet Adjustment

When adding a new measurement to the database, we discriminate between control points and free points.

Control points are points that already exist in the database; free points are new points to be added to the database. As we see in Figure 2, the points pt3, pt4, pt30, pt31, and pt504 are already in the database and therefore they are control points. The points pt20, pt21, pt25, and pt32 are from the new house and therefore free. Points pt1 and pt2 are the theodolite station points. These points are used in the computation but not stored in the database.

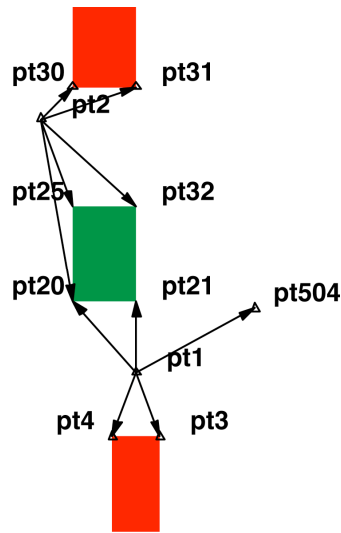


Figure 2. The control points (pt3, pt4, pt30, and pt504) and free points (pt20, pt21, pt25, and pt32) of the database.

The rubber-sheet transformation maps the measured coordinates of the control points exactly onto the coordinates of the control points in the database (this is the prime reason for choosing the rubber-sheet transformation).

There are two main steps in the operation.

- The first step is to link the coordinate system of the measurement with the database. This is done by a similarity transformation. This transformation can only map the measured coordinates of two control points exactly onto the coordinates in the database. So, there are differences for the other control points.
- The second step is to eliminate these differences.

The following formula is central to the second step:

$$e_f = H_{fc} \cdot H_{cc}^{-1} \cdot e_c \quad (1)$$

with

- e_f being the corrections to the coordinates of a free point
- H_{fc} representing the covariance matrix of a free point and the control points
- H_{cc} representing the covariance matrix of the control points
- e_c being the vector of differences between the coordinates of the control points in the measurement (after the similarity transformation) and the database.

This formula expresses the correction for any point as a weighted linear combination of the corrections of the control points. The weights depend on the distance

between the point and the control points. We explain below how the matrices H_{fc} and H_{cc} are constructed.

The most important property of this method is that it does not matter which basis you choose for the similarity transformation: after applying the above formula, you get the same result regardless of the basis. This is a form of invariance.

■ The Similarity Transformation

First, two control points r and s are chosen. These two points are mapped precisely onto the database by the similarity transformation S . A similarity transformation is by definition composed of rotation, scaling, and translation, so the transformation S is given by

$$x^{(2)} = Sx^{(1)} = \begin{pmatrix} p & -q \\ q & p \end{pmatrix} (x^{(1)} - r^{(1)}) + r^{(2)},$$

where p and q are found by solving $s^{(2)} = Ss^{(1)}$. Here $x^{(2)}$ denotes the coordinates of the point in the database and $x^{(1)}$ denotes the coordinates of the point in the measurement.

The *Mathematica* implementation is straightforward.

```
In[1]:= A = {{p, -q}, {q, p}};
r1 = {r1x, r1y};
r2 = {r2x, r2y};
s1 = {s1x, s1y};
s2 = {s2x, s2y};
S[{x_, y_}] := A.({x, y} - r1) + r2;
```

The numbers p and q are found by the following.

```
In[7]:= sols = Evaluate[Solve[S[s1] == s2, {p, q}]]][[1]]
Out[7]= {p -> -(-r1xr2x - r1yr2y + r2xs1x + r2ys1y + r1xs2x - s1xs2x + r1ys2y -
s1ys2y) / (r1x^2 + r1y^2 - 2r1xs1x + s1x^2 - 2r1ys1y + s1y^2),
q -> -((-r1y + s1y) (r2x - s2x) + (-r1x + s1x) (r2y - s2y)) /
((-r1x + s1x)^2 - (r1y - s1y) (-r1y + s1y)}
```

■ The Covariance Matrix

For two sets of points $\{u_1, \dots, u_m\}$ and $\{v_1, \dots, v_n\}$, the covariance matrix is made from 2×2 blocks:

$$H_{\{u_i\}\{v_j\}}^{r,s} = \begin{pmatrix} h_{xx}(u_1, v_1) & h_{xy}(u_1, v_1) & \dots & h_{xx}(u_1, v_n) & h_{xy}(u_1, v_n) \\ -h_{xy}(u_1, v_1) & h_{xx}(u_1, v_1) & \dots & -h_{xy}(u_1, v_n) & h_{xx}(u_1, v_n) \\ \vdots & & & & \\ h_{xx}(u_m, v_1) & h_{xy}(u_m, v_1) & \dots & h_{xx}(u_m, v_n) & h_{xy}(u_m, v_n) \\ -h_{xy}(u_m, v_1) & h_{xx}(u_m, v_1) & \dots & -h_{xy}(u_m, v_n) & h_{xx}(u_m, v_n) \end{pmatrix}$$

where r and s denote a basis and

$$b_{x,x}(u, v) = - | u - v | + (1 - a(v)) | u - r | + a(v) | u - s | + (1 - a(u)) | u - r | \\ + a(u) | v - s | + (-a(u) - a(v) + 2 a(u) a(v) + 2 b(u) b(v)) | s - r |$$

and

$$b_{x,y}(u, v) = b(v) | u - r | - b(v) | u - s | - b(u) | v - r | + b(v) | v - r | \\ + b(u) | v - s | - (b(u) - b(v) - 2 a(v) b(u) + 2 a(u) b(v)) | s - r |$$

with

$$a(u) = (u - r) \cdot (s - r) / (s - r) \cdot (s - r) \\ b(u) = (u - r)^\top \cdot (s - r) / (s - r) \cdot (s - r).$$

In equation (1), the term H_{f_c} is the matrix H for a single free point f and the set of control points c . The term H_{c_c} is the matrix H for both sets of control points c .

■ Mathematical Background

For the covariance matrix, we do not take the actual covariance of this particular measurement, but start from a standardized covariance matrix whose coefficients depend only on the distances between points. This has the advantage that the warping does not depend on the particular measurement. This choice is allowed because we know that the measurements are sufficiently reliable since they have passed a standardized test. In fact, the specifications on the covariance in this test are reflected in the new matrix.

The matrix is constructed in several steps. Let us say that the vicinity of two coordinate values x and x' is measured by

$$d(x, x') := D - (x - x')^2$$

for some fixed positive number D . If $d(x, x')$ would be negative, we set it equal to 0. The standardized covariance matrix for two points (x, y) and (x', y') has the form

$$\begin{pmatrix} d(x, x') & 0 \\ 0 & d(y, y') \end{pmatrix}.$$

This tells us that if two points are close, then the x -coordinates and the y -coordinates are close, but there is *a priori* no cross relation between the x -coordinate and the y -coordinate. For two sets of points $\{u_1, \dots, u_m\}$ and $\{v_1, \dots, v_n\}$ the matrix is just made from 2×2 blocks by taking as i, j -th block the covariance matrix of u_i and v_j :

$$\begin{pmatrix} d(u_{1,x}, v_{1,x}) & 0 & \dots & d(u_{1,x}, v_{n,x}) & 0 \\ 0 & d(u_{1,y}, v_{1,y}) & \dots & 0 & d(u_{1,y}, v_{n,y}) \\ \vdots & & & & \\ d(u_{m,x}, v_{1,x}) & 0 & \dots & d(u_{m,x}, v_{n,x}) & 0 \\ 0 & d(u_{m,y}, v_{1,y}) & \dots & 0 & d(u_{m,y}, v_{n,y}) \end{pmatrix}.$$

The above is the standard covariance matrix in the local coordinates of the measurement. During the warping we use the coordinate system of the database, so the matrix is affected by the similarity transformation and the result is $H_{\{u\}\{v\}}^{r,s}$. This is worked out in [2], where you may find more details.

■ Implementation

Here is the implementation of the rubber-sheet transformation.

```
In[8]:= a[x_, r_, s_] := (r - x) . (r - s) / (r - s) . (r - s);
b[x_, r_, s_] := T[r - x] . (r - s) / (r - s) . (r - s);
T[{a_, b_}] := {-b, a};
d2[x_List, y_List] := N[Sqrt[(x - y) . (x - y)]];

hxx[i_List, j_List, r_, s_] :=
  -d2[i, j] - (-1 + a[j, r, s]) * d2[i, r] + a[j, r, s] * d2[i, s] -
  (-1 + a[i, r, s]) * d2[j, r] + a[i, r, s] * d2[j, s] +
  d2[r, s] * (-a[i, r, s] - a[j, r, s] + 2 * a[i, r, s] * a[j, r, s] +
  2 * b[i, r, s] * b[j, r, s]);

hxy[i_List, j_List, r_, s_] :=
  b[j, r, s] * d2[i, r] - b[j, r, s] * d2[i, s] - b[i, r, s] * d2[j, r] +
  b[i, r, s] * d2[j, s] - (b[i, r, s] - 2 * a[j, r, s] * b[i, r, s] -
  b[j, r, s] + 2 * a[i, r, s] * b[j, r, s]) * d2[r, s];

h[p_, q_, r_, s_] := {{hxx[p, q, r, s], hxy[p, q, r, s]},
  {-hxy[p, q, r, s], hxx[p, q, r, s]}};

covarianceBlock[h_, u_, {v__}, r_, s_] := Module[{x}, Flatten/@
  Thread[Apply[h[#1, #2, r, s] &, Thread[{x, {v}}], {1}]] /. x -> u];

covarianceMatrix[h_, u_List, {v__}, r_, s_] :=
  Flatten[covarianceBlock[h, #, {v}, r, s] & /@ u, 1];

covarianceMatrix[h_, u_, {v__}, r_, s_] :=
  covarianceMatrix[h, {u}, {v}, r, s];
```

The definition of `covarianceBlock`, which makes a row of 2×2 blocks as in H_{fc} , is probably the only one that may need some explanation. For example, let us make the matrix for `u` and `{v1, v2}` with the help of `Thread`.

```
In[18]:= Thread[{u, {v1, v2}}]
```

```
Out[18]= {{u, v1}, {u, v2}}
```

Next we apply `h` on the two blocks at the right level.

```
In[19]:= Apply[h[#1, #2, r, s] &, %, {1}]
```

```
Out[19]= {{hxx[u, v1, r, s], hxy[u, v1, r, s]}, {-hxy[u, v1, r, s], hxx[u, v1, r, s]}},
  {{hxx[u, v2, r, s], hxy[u, v2, r, s]}, {-hxy[u, v2, r, s], hxx[u, v2, r, s]}}
```

Finally we flatten to get a matrix.

```
In[20]:= Flatten /@ %
```

```
Out[20]= {{hxx[u, v1, r, s], hxy[u, v1, r, s],
           -hxy[u, v1, r, s], hxx[u, v1, r, s]}, {hxx[u, v2, r, s],
           hxy[u, v2, r, s], -hxy[u, v2, r, s], hxx[u, v2, r, s]}}
```

To make this code work for a concrete vector (u_x, u_y) , as well as for a symbol u , we first introduce a free variable x instead of u and then use the substitution $x \rightarrow u$. This variable has been made local with the use of `Module`. All together we obtain `covarianceBlock`.

■ Example

Here is the example of Figure 2. Table 1 gives the measured coordinates and the coordinates of the database. First we choose `pt3` and `pt504` as a basis for the similarity transformation. Then we show that another basis, for example `pt30` and `pt31`, will give the same result for the final coordinates of a measurement.

	Measured coordinates		Database coordinates	
	x	y	x	y
pt20	1992.000	2009.000	to	to
pt21	2000.000	2009.000	be	be
pt25	1991.997	2021.000	com-	com-
pt32	2000.001	2021.001	puted	puted
pt3	2003.000	1992.000	155003.032	462992.079
pt4	1997.007	1991.998	154997.052	462992.039
pt30	1991.994	2036.000	154992.035	463036.095
pt31	1999.994	2036.002	155000.085	463036.052
pt504	2015.000	2008.000	155015.041	463008.000

Table 1. Measured coordinates and database coordinates.

We choose $r = \text{pt3}$ and $s = \text{pt504}$ as a basis. The other control points are the points `pt4`, `pt30`, and `pt31`. We transform the local coordinates of the control points and the free points (`pt20`, `pt21`, `pt25`, and `pt32`) with respect to the chosen basis. Here we only work out `pt20`.

```
In[21]:= r1x = 2003; r1y = 1992; r2x = 155003.032; r2y = 462992.079;
s1x = 2015; s1y = 2008; s2x = 155015.041; s2y = 463008.000;
pt20 = {1992, 2009};
pt21 = {2000, 2009};
pt25 = {1991.997, 2021.000};
pt32 = {2000.001, 2021.001};
pt20t = S[pt20] /. sols;
```

```

pt21t = S[pt21] /. sols;
pt25t = S[pt25] /. sols;
pt32t = S[pt32] /. sols;
pt4 = {1997.007, 1991.998};
pt30 = {1991.994, 2036};
pt31 = {1999.994, 2036.002};
pt4t = S[pt4] /. sols;
pt30t = S[pt30] /. sols;
pt31t = S[pt31] /. sols;
pt3db = {r2x, r2y};
pt4db = {154997.052, 462992.039};
pt30db = {154992.035, 463036.095};
pt31db = {155000.085, 463036.052};
pt504db = {s2x, s2y};

```

We transform the local coordinates of the control points and the free points (pt20, pt21, pt25, and pt32) with respect to the chosen basis. Here we only work out pt20.

```

In[42]:= pt20t = S[pt20] /. sols // N[#, 10] &
Out[42]= {154992., 463009.}

```

Next we compute the database coordinates of the new house with the following formula.

```

In[43]:= hfc = covarianceMatrix[h,
      {pt20t, pt21t, pt25t, pt32t},
      {pt4t, pt30t, pt31t}, pt3db, pt504db];
hcc = covarianceMatrix[h,
      {pt4t, pt30t, pt31t},
      {pt4t, pt30t, pt31t}, pt3db, pt504db];
ec = Flatten[{pt4db, pt30db, pt31db}] -
      Flatten[{pt4t, pt30t, pt31t}];
ef = hfc.Inverse[hcc].ec
Out[46]= {-0.0686549, 0.000513474, -0.0423258, 0.00651125,
      -0.0968291, 0.044021, -0.0634426, 0.0439568}

```

Here are the final coordinates of pt20, pt21, pt25, and pt32.

```

In[47]:= def3504 = Flatten[{pt20t, pt21t, pt25t, pt32t}] + ef // N[#, 10]&
Out[47]= {154992., 463009., 155000.,
      463009., 154992., 463021., 155000., 463021.}

```

Table 2 summarizes the result.

	Transformed		Database		Difference	
	x	y	x	y	x	y
pt20	154992.110	463009.060	154992.042	463009.060	-0.069	0.001
pt21	155000.087	463009.038	155000.045	463009.045	-0.042	0.007
pt25	154992.140	463021.025	154992.043	463021.069	-0.097	0.044
pt32	155000.121	463021.004	155000.057	463021.048	-0.063	0.044
pt3	155003.032	462992.079	155003.032	462992.079	0.000	0.000
pt4	154997.056	462992.093	154997.052	462992.039	-0.004	-0.054
pt30	154992.178	463035.982	154992.035	463036.095	-0.143	0.113
pt31	155000.155	463035.962	155000.085	463036.052	-0.070	0.090
pt504	155015.041	463008.000	155015.041	463008.000	0.000	0.000

Table 2. The transformed coordinates with respect to pt3 and pt504.

■ Another Basis

Choosing another basis for the similarity transformation, for example $r = \text{pt30}$ and $s = \text{pt31}$, leads to an entirely different computation, but the final results will be the same.

Again Table 1 is our starting point. Now the other control points are the points pt3, pt4, and pt504. We transform the local coordinates of these control points and the free points (pt20, pt21, pt25, and pt32) with respect to this basis. The results are in a vector def3031. Table 3 shows the results.

	Transformed		Database		Difference	
	x	y	x	y	x	y
pt20	154991.889	463008.926	154992.042	463009.060	0.152	0.134
pt21	154999.939	463008.881	155000.045	463009.045	0.106	0.163
pt25	154991.954	463021.001	154992.043	463021.069	0.090	0.068
pt32	155000.008	463020.957	155000.057	463021.048	0.050	0.091
pt3	155002.862	462991.758	155003.032	462992.079	0.170	0.321
pt4	155002.862	462991.758	154997.052	462992.039	0.220	0.249
pt30	154992.035	463036.095	154992.035	463036.095	0.000	0.000
pt31	155000.085	463036.052	155000.085	463036.052	0.000	0.000
pt504	155015.027	463007.791	155015.041	463008.000	0.014	0.209

Table 3. The transformed coordinates with respect to pt30 and pt31.

We redo the computation for this case.

```

In[48]:= r1x=1991.994; r1y=2036.000; r2x=154992.035; r2y=463036.095;
s1x=1999.994; s1y=2036.002; s2x=155000.085; s2y=463036.052;
pt20t=S[pt20] /. sols;
pt21t=S[pt21] /. sols;
pt25t=S[pt25] /. sols;
pt32t=S[pt32] /. sols;
pt4={1997.007, 1991.998};
pt3 = {2003, 1992};
pt504 = {2015, 2008};
pt4t= S[pt4] /. sols;
pt3t= S[pt3] /. sols;
pt504t= S[pt504] /. sols;
pt3db={155003.032, 462992.079};
pt4db = {154997.052, 462992.039};
pt30db={r2x, r2y};
pt31db={s2x, s2y};
pt504db={155015.041, 463008.000};

hfc = covarianceMatrix[h,
{pt20t, pt21t, pt25t, pt32t}, {pt3t, pt4t, pt504t}, pt30db, pt31db];

hcc = covarianceMatrix[h,
{pt3t, pt4t, pt504t}, {pt3t, pt4t, pt504t}, pt30db, pt31db];

ec = Flatten[{pt3db, pt4db, pt504db}] - Flatten[{pt3t, pt4t, pt504t}];

In[68]:= ef = hfc.Inverse[hcc].ec

Out[68]= {0.152425, 0.134159, 0.105645, 0.163329,
0.0895193, 0.067995, 0.0497575, 0.0911056}

Here are the final coordinates of pt20, pt21, pt25, and pt32 with respect to the
basis pt30 and pt31.

In[69]:= def3031 =
N[Flatten[{pt20t, pt21t, pt25t, pt32t}]+ ef, 10]

Out[69]= {154992., 463009., 155000.,
463009., 154992., 463021., 155000., 463021.}

```

■ Visualizing Results

To visualize and compare the results of the different bases, we take a corner of the house and show the images of the corner under the two similarity transformations together with the final result of both computations (Figure 3).

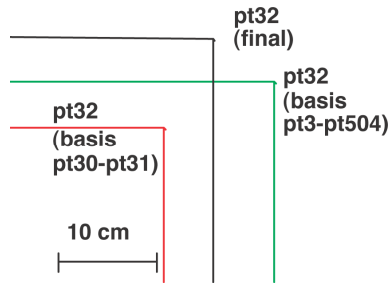


Figure 3. A zoomed-in picture of a corner of the new house, where you can see the effect of the different choices.

In Figure 3 the green corner is the image under the first transformation, the red corner is the image under the second transformation, and the black corner is the final result of both computations.

We can also perform a numerical check: the final results of the second computation should be numerically equal to those of the first computation.

```
In[70]:= def3504 - def3031
Out[70]= {-3.7835 × 10-10, 2.32831 × 10-10, -4.65661 × 10-10, -1.33878 × 10-9,
          9.31323 × 10-10, 1.16415 × 10-10, 2.44472 × 10-9, -1.28057 × 10-9}
```

We can make a picture of the warping by applying `PlotVectorField` from the package `Graphics`PlotField``. The vector field has been combined with the map using `Show`. Figure 4 shows the result: the “vector field” is like a vortex with two “sinks” at the basis points `pt30` and `pt31` where the corrections should be zero.

```
In[71]:= << Graphics`PlotField`
In[72]:= f = covarianceMatrix[h, {{x, y}}, {pt3t, pt4t, pt504t}, pt30db, pt31db].
          Inverse[hcc].ec;
```

```
In[73]:= PlotVectorField[f, {x, pt4t[[1]] - 20, pt504t[[1]] + 5, 10},
      {y, pt3t[[2]] - 15, pt31db[[2]] + 15, 10}]
```

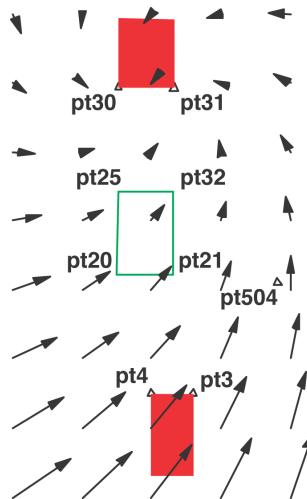


Figure 4. A “vector field” showing the relative size and direction of the corrections to the transformed coordinates.

■ Conclusion

The rubber-sheet algorithm for warping allows us to update a geographic database without altering points that are already known in the database. *Mathematica* offers a quick experimental validation of the theory and of the existing FORTRAN legacy code.

■ References

- [1] R. Laurini and D. Thompson, *Fundamentals of Spatial Information Systems*, London: Academic Press, 1992.
- [2] H. J. Buiten and P. Richardus, *Junction of Control Surveys by Adjustment Compared with Coordinate Transformation*, München: Hochschule der Bundeswehr, Schriftenreihe Wissenschaftlicher Studiengang Vermessungswesen, Heft 7, 1982, pp. 115–141.

About the Authors

Leendert van Gastel is a consultant specializing in the use of ICT in science and mathematics. He has been working with *Mathematica* since 1992, both in applied research and training.

Harry Uitermark is a GIS consultant. Currently his research interests include geographic information, ontologies, and XML Schema, a web standard for describing the structure and semantics of XML documents.

Leendert van Gastel

*Amstel Instituut
University of Amsterdam
gastel@science.uva.nl*

Harry Uitermark

*Kadaster, Box 9046
Apeldoorn 7300 GH, The Netherlands
harry.uitermark@kadaster.nl*