Mini Project
on
Discrete-Time Kalman Filter

The project solves a given problem by applying several different methods. The problem is a simple one: Given two sets X and Y of 2-dimensional coordinates. The coordinates are *observations* of the same points (with common weight) in two different coordinate systems. The observations are contaminated by random errors which we assume are normally distributed and with zero mean.

We look for formulas that transform coordinates from one set to another. The transformation shall be a similarity transformation which allows for translations t_x, t_y, a rotation ϕ, and a change of scale h.

We start by solving the problem using classical least squares and additional matrix algebra. These are powerful tools; next we use vector updating as formulated by various filters.

The Classical Least-Squares Solution

The coordinates of point i are given as rows i in both X and Y. We call them $(x_{i,1}, x_{i,2})$ and $(y_{i,1}, y_{i,2})$. The transformation is

$$\begin{bmatrix} y_{i,1} \\ y_{i,2} \end{bmatrix} = h \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_{i,1} \\ x_{i,2} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \text{error}. \tag{1}$$

We introduce the abbreviations $a = h\cos\phi$ and $b = h\sin\phi$. After a, b are estimated, we compute h and ϕ as follows

$$h = \sqrt{a^2 + b^2}$$
$$\phi = \text{atan2}(b, a).$$

The introduced change of variables eliminates the non-linearity introduced by the trigonometrical functions! We now get

$$\begin{bmatrix} y_{i,1} \\ y_{i,2} \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} x_{i,1} \\ x_{i,2} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \text{error}. \tag{2}$$

Next we use a remarkable trick of rearrangement, so that the vector of $(x_{i,1}, x_{i,2})$ is substituted by (a, b):

$$\begin{bmatrix} y_{i,1} \\ y_{i,2} \end{bmatrix} = \begin{bmatrix} x_{i,1} & x_{i,2} \\ x_{i,2} & -x_{i,1} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \text{error}. \tag{3}$$

We use data from Example 12.6 in the book by Strang and Borre, page 418:

$$X = \begin{bmatrix} 277\,722.02 & -230\,855.15 \\ 275\,956.87 & -231\,105.84 \\ 277\,563.37 & -235\,447.40 \\ 278\,608.53 & -233\,945.92 \\ 276\,163.68 & -236\,471.63 \\ 273\,578.80 & -230\,941.42 \\ 274\,533.96 & -235\,063.72 \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} 6\,310\,000.53 & 562\,940.82 \\ 6\,308\,231.26 & 562\,725.63 \\ 6\,309\,749.96 & 558\,354.12 \\ 6\,310\,824.66 & 559\,833.89 \\ 6\,308\,330.47 & 557\,358.46 \\ 6\,305\,857.71 & 562\,937.59 \\ 6\,306\,729.80 & 558\,798.28 \end{bmatrix}.$$

The least-squares solution for these data is

$$\begin{aligned} a &= 0.999\,484\,492 & b &= 0.020\,032\,209 \\ t_x &= 6\,037\,046.21 & t_y &= 799\,240.35. \end{aligned}$$

A General Remark on Numerical Stability

The above method is of the batch processing type where all available observations are included into one and the same step/solution. The A and x fit together with large and small numbers:

$$b = Ax$$

$$\begin{bmatrix} y_{i,1} \\ y_{i,2} \end{bmatrix} = \begin{bmatrix} x_{i,1} & x_{i,2} & 1 & 0 \\ x_{i,2} & -x_{i,1} & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ t_x \\ t_y \end{bmatrix}. \tag{4}$$

In A the $x_{i,1}$ and $x_{i,2}$ are large compared to 1 and 0; a and b lie in the interval from -1 to $+1$ while t_x and t_y often are large numbers. This situation is handled well by the normal equations; the solution creates no further problem.

The subsequent methods work *recursively*. This may cause numerical problems. One way of repairing the situation is to reduce all coordinates to their gravity centers. Let the number of rows in X and Y be r, then the gravity centers $(x_{s,1}, x_{s,2})$ and $(y_{s,1}, y_{s,2})$ are defined as

$$\begin{aligned} x_{s,1} &= \sum x_{i,1}/r, & x_{s,2} &= \sum x_{i,2}/r \\ y_{s,1} &= \sum y_{i,1}/r, & y_{s,2} &= \sum y_{i,2}/r \end{aligned}$$

and the *reduced coordinates* are

$$\begin{aligned} x_{i,1} &= x_{i,1} - x_{s,1}, & x_{i,2} &= x_{i,2} - x_{s,2} \\ y_{i,1} &= y_{i,1} - y_{s,1}, & y_{i,2} &= y_{i,2} - y_{s,2} \end{aligned}$$

or in Matlab code

```
[r,s] = size(X);
gammax = ones(r,1)*sum(X,1)/r;
X = X-gammax;
gammay = ones(r,1)*sum(Y,1)/r;
Y = Y-gammay;
```

Now all column sums of X and Y are zero, and in the following we exclusively work with the *reduced coordinates*. In the filter t_x and t_y are close to zero and all $x_{i,1}$ and $x_{i,2}$ are small numbers.

In case the X and Y coordinates are reduced to the gravity centers we need an alternative way of computing t_x and $_y$. This is done by using (3) which is valid for any set of points, hence also for the gravity centers:

$$\begin{bmatrix} y_{s,1} \\ y_{s,2} \end{bmatrix} = \begin{bmatrix} x_{s,1} & x_{s,2} \\ x_{s,2} & -x_{s,1} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \tag{5}$$

A Modern Solution

Recently an alternative theory was developed by Krarup and Borre. This theory makes use of the ***singular value decomposition*** and involves two steps: Firstly, all coordinates in X and in Y are reduced to their gravity centers, and secondly the mixed product of the reduced coordinate matrices is used as input for a SVD. The product of the orthogonal matrices is the rotation matrix! The change of scale is computed as a fraction between traces of certain matrix products. In Matlab, the core code is

```
[Phi, Sigma, Psit] = svd(X'*Y);
Omega = Phi*Psit;
h = trace(Y'*Y)/trace(X'*Y*Omega');
```

The numerical result is

$$h\Omega = \begin{bmatrix} 0.999\,484\,492 & -0.020\,032\,209 \\ 0.020\,032\,209 & 0.999\,484\,492 \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$$

and from (5) the translations are

$$t_x = 6\,037\,046.208 \qquad t_y = 799\,240.351.$$

The present method once again demonstrates the power and usefulness of SVD.

Recursive Least Squares

We assume that we have reduced X and Y to their gravity centers. So the first guess for the unknowns is $x = (a, b, t_x, t_y) = (1, 0, 0, 0)$, and we put $\Sigma_b = I$.

A code may result in the following numbers:

Filtering No	a	b
1	1.008 389 45	0.015 080 03
2	0.999 484 73	0.020 033 59
3	5 87	3 76
4	5 83	4 02
5	5 96	3 25
6	4 11	2 95
7	4 19	2 88
8	4 38	3 08
9	4 38	3 08
10	4 20	3 15
11	4 82	2 70
12	4 59	2 39
13	4 47	2 28
14	4 51	2 24

The result for the translation once again comes from (5):

$$t_x = 6\,037\,046.21 \qquad t_y = 799\,240.36.$$

The Kalman Filter

The Kalman filter version again works from the reduced coordinates. The core code is

```
innovation_variance = A * P * A' + R;
K = (A * P)'/innovation_variance;
x = x + K * (b - A * x);
P = P - K * A * P;
```

First we run a filter with four parameters (a, b, t_x, t_y) and initial values $x_0 = (0, 0, 0, 0)$, $P = I_4$, and $\Sigma_b = R = 0.01 I_4$. The result is

Filtering No	a	b
1	1.008 389 45	0.015 080 03
1	0.999 484 49	0.020 033 38
2	0.999 485 41	0.020 033 92
3	0.999 484 08	0.020 032 87
4	0.999 484 37	0.020 033 00
5	0.999 484 18	0.020 033 08
6	0.999 484 57	0.020 032 35
7	0.999 484 49	0.020 032 20

$$t_x = 6\,037\,046.21 \qquad t_y = 799\,240.35.$$

With one equation at a time and a and b as unknowns:

Filtering No	a	b
1	0.244 711 48	0.439 902 71
2	2.472 135 30	2.667 326 53
3	0.999 484 18	0.020 032 62
4	0.999 484 68	0.020 032 95
	...	
14	0.999 484 39	0.020 032 03

$$t_x = 6\,037\,046.19 \qquad t_y = 799\,240.28.$$

Note that this result deviates from all earlier ones.

The Bayes Filter

We get the Bayes formulation from Example 15.3

```
P = inv(inv(P) + A' * inv(R) * A);
K = P * A' * inv(R);
x = x + K * (b - A * x);
```

The numerical results are similar to the ones from the Kalman filter.

Your Contribution

The mini project asks for your analysis and Matlab code for the various formulations of the problem described:

1. Classical Least-Squares Solution
2. Advanced Modern Solution
3. Recursive Least-Squares Solution
4. The Kalman Filter Solution
   (a) Write a Kalman filter with state vector (a, b) and which updates one row at a time
   (b) Investigate if updating with two rows at a time makes any difference compared to one row at a time
   (c) Investigate if an augmentation of the state vector from $x = (a, b)$ to $x = (a, b, t_x, t_y)$ makes any difference
5. The Bayes Filter Solution. Same comments as above for the Kalman filter

Reuse of code is allowed, and even recommend if you see a possibility. I wish you much joy with the task.

Kai Borre
September, 2008