

Re: a rigid transformation problem

Source: <http://sci.tech-archive.net/Archive/sci.math/2006-03/msg04668.html>

- *From:* "Robert Israel" <israel@xxxxxxxxxxx>
 - *Date:* 24 Mar 2006 09:26:57 -0800
-

Thomas Mautsch wrote:

In news:<[dvccvn\\$ktm\\$1@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:dvccvnktm1@xxxxxxxxxxxxxxxxxxxxxxxx)>
schrieb Robert Israel <israel@xxxxxxxxxxx>:

In article <44195a1c@xxxxxxxxxxx>, Thomas Mautsch wrote:

In news:<[dva3jn\\$pk\\$1@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:dva3jnpk1@xxxxxxxxxxxxxxxxxxxxxxxx)>
schrieb Robert Israel <israel@xxxxxxxxxxx>:

<jayzhuo@xxxxxxxx> wrote:

given two point sets $\{p_i\}$
 $\{q_i\}$, determining the rotation
matrix R and
translation matrix T , so that
the error E is minimized:

$$E = \text{square sum of } d(R \cdot p_i + T, q_i)$$

Do you mean sum of the squares?

Let us suppose he does.

the distance is defined as
follow:

$$d(p, q) = \|p - q\| \text{ if } \|p - q\| < M$$
$$M \text{ otherwise}$$

where M is a constance.

That could be rather nasty, as it makes the

Re: a rigid transformation problem

objective function
non-differentiable.

But not badly non-differentiable – I personally would always prefer a piecewise linear function to a fully nonlinear function like \tanh in this context.

In theory you could solve this optimization problem as follows:

Say the number of given pairs of points (p_i, q_i) is n .
For each of the 2^n subsets S of the index set $\{1, 2, \dots, n\}$,

Not very practical, of course, unless n is rather small...

Yes. Maybe there are shortcuts that allow to dispose of some of the 2^n cases without calculating an SVD?

Would a smooth cut-off such as
 $d(p, q) = M \tanh(\|p - q\|/M)$ work for your purpose?

It might be better to take two different values for the two values M in this formula,
maybe even different from the value of M in the OP's definition of distance, or might it not?

how to solve this problem?

Using a numeric solver.
You have your choice of parametrizing the rotation matrix, e.g.
in terms of Euler angles, or using constraints
 $R^T R = I$ and
 $\det(R) > 0$.

I wonder how good numeric solvers can cope with the fact that the gradient of \tanh is so small at larger values. – Might there not be the possibility that the solver gets stuck in local maxima where

Re: a rigid transformation problem

(most of) the points $(R p_i + T)$ and q_i are far apart from each other.

Maybe you could provide an example. –
That would be marvellous!

OK, here's an example in Maple 10.

Thank you very much for the example.

```
with(LinearAlgebra): with(Optimization):
```

I still have doubts about the usefulness of the Optimization package. –
How can you check that the solution is correct? – Maple's solver certainly doesn't tell when the solution set is not unique...

On non-convex problems such as this, the Optimization solver can get stuck in a local optimum. You can use GlobalSolve in the Global Optimization Toolbox to find global optima. Sometimes it can be quite slow, especially on high-dimensional problems, but in this particular case it's quite fast and returns essentially the same result as LSSolve.

```
# Here is a parametrization of a 3D rotation matrix
# in terms of Euler angles a,b,c

Rabc:= Matrix([[ cos(c)*cos(b)*cos(a)-sin(c)*sin(a),
-sin(c)*cos(b)*cos(a)-cos(c)*sin(a), sin(b)*cos(a) ],
[ cos(c)*cos(b)*sin(a)+sin(c)*cos(a),
-sin(c)*cos(b)*sin(a)+cos(c)*cos(a), sin(b)*sin(a) ],
[ -cos(c)*sin(b), sin(c)*sin(b), cos(b) ]]);
```

I wonder what will happen when the rotation R_t is near a critical point of this parametrization...

```
V:= <v1,v2,v3>;
```

```
# Make up some data. Ten random vectors for the p_i
```

```
P:= [seq(RandomVector(3,generator=rand(-5..5)),i=1..10)];
```

Re: a rigid transformation problem

```
[1] [-4] [ 0] [-5] [ 4] [-4] [ 5] [ 4] [ 2] [ 5]
[] [] [] [] [] [] [] [] [] []
P := [[4], [ 5], [-1], [ 2], [ 5], [-2], [-3], [-4], [4], [-5]]
[] [] [] [] [] [] [] [] [] []
[0] [-2] [ 5] [-1] [-4] [ 2] [ 3] [ 5] [3] [ 0]

# The true transformation has Euler angles 1, 2, 3 and translation
# <1,2,3>
```

```
Rt:= eval(Rabc,{a=1.0, b=2.0, c=3.0});
```

```
Tt:= <1.0,2.0,3.0>;
Q:= map(v -> Rt.v+Tt, P);
```

```
# An extra translation of <1,1,1> is added to the first three vectors in Q
# so the "cutoff" will have something to do.
```

```
for i from 1 to 3 do Q[i]:= Q[i] + <1,1,1> od;
```

^^^^^^

That's a **VERY** LONG vector when compared with the cut-off value $M = 1$.
No wonder your approximate solution turns out
to almost the original rotation.

```
# Now get the residuals  $d(p,q) = \tanh \|p - q\|$ 
```

```
E:= [seq(Rabc.P[i]+V-Q[i], i=1..10)];
```

```
ds:= map(v -> tanh(sqrt(v^%T . v)), E);
```

```
# Minimize the sum of squares of residuals using the least-squares
# solver
```

```
LSSolve(ds);
```

```
[1.3090743731944, [a = 1.00382746183031957, c = 2.99466288305908712,
```

^^^^^^^^^^^^^^^^^^ Strange!

```
b = 1.99673288295575734, v1 = 1.04762504047361582,
```

```
v2 = 2.03025329876483252, v3 = 3.02999015731672605]]
```

It was very quick, and the result is very close to the "true"

Re: a rigid transformation problem

transformation.

For comparison:

It took my computer minutes to calculate the exact solution
in the spirit of the original posting:

The optimal solution for the data given here
are really the matrix R_t and the vector T_t themselves –
the total error is 3 (because the three points $P[1]$, $P[2]$, $P[3]$
are "far off"). – This situation remains the same
as long as the cut-off bound M stays ≤ 1.3 .
For $M \geq 1.4$, the optimal solution contains no excluded points.

Funny that the error for your tanh-functional given by `LSSolve`
is with 1.3 so much smaller than 3, or isn't it.

The first value returned by `LSSolve` is supposed to be $1/2$ times the sum
of
the squares of the residuals at the minimum it finds. I don't know why
the $1/2$,
but that's what it is.

Robert Israel israel@xxxxxxxxxxxx
Department of Mathematics <http://www.math.ubc.ca/~israel>
University of British Columbia Vancouver, BC, Canada

.