Least Squares

L27

Goal: To find out how the sum of squared differences cost function is used in image registration.

The "least squares" cost function, also known as the "sum of squared differences" (SSD), is a very popular cost function.

$$d_{SSD}(f,g) = \sum_{mn} (f_{mn} - g_{mn})^2$$

The reason for its popularity:

1. It is the optimal cost function in the presence of Gaussian-distributed additive noise.
2. It has an analytical solution (sort of).

Consider the rigid-body 2D transform with 3 parameters,

(r, c, θ) = (row shift, col shift, rotation)

Using Taylor's theorem (again),

$$T(r,c,\theta) = T(0,0,0)f + \frac{\partial T(0,0,0)f}{\partial r} r + \frac{\partial T(0,0,0)f}{\partial c} c + \frac{\partial T(0,0,0)f}{\partial \theta} \theta + O(r^2, c^2, \theta^2)$$

So, if r, c & θ are small,

$$T(r,c,\theta) \cong f + \frac{\partial f}{\partial r} r + \frac{\partial f}{\partial c} c + \frac{\partial f}{\partial \theta} \theta$$

$$= f + \nabla f \begin{bmatrix} r \\ c \\ \theta \end{bmatrix}$$

where $\nabla f = \left[\frac{\partial f}{\partial r}, \frac{\partial f}{\partial c}, \frac{\partial f}{\partial \theta}\right]$.

We can write this equation for every pixel (m, n)

$$\left[T(r,c,\theta) f\right]_{mn} = f_{mn} + \nabla f_{mn} \begin{bmatrix} r \\ c \\ \theta \end{bmatrix} \quad m = 1, \ldots, M \quad n = 1, \ldots, N$$

If we think of our image as a column vector $\vec{f}$, then

$$T(r,c,\theta)\,\vec{f} = \vec{f} + \nabla\vec{f}\begin{bmatrix} r \\ c \\ \theta \end{bmatrix}$$

Now this is an $(MN)\times 3$ matrix

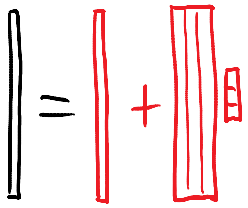
$$\nabla\vec{f} = \begin{bmatrix} \textcircled{1} & \textcircled{2} & \textcircled{3} \\ \frac{\partial f_{11}}{\partial r} & \frac{\partial f_{11}}{\partial c} & \frac{\partial f_{11}}{\partial \theta} \\ \frac{\partial f_{12}}{\partial r} & \frac{\partial f_{12}}{\partial c} & \frac{\partial f_{12}}{\partial \theta} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_{MN}}{\partial r} & \frac{\partial f_{MN}}{\partial c} & \frac{\partial f_{MN}}{\partial \theta} \end{bmatrix}$$

Let's call this A

Then we can approximate small motions using

$$T(r,c,\theta)\,\vec{f} = \vec{f} + Am$$

Each row of this matrix equation corresponds to one pixel.



The moved $\vec{f}$ equals the original plus a linear motion term... a nudge.

It's important to remember that Tf is nonlinear, and that we have merely approximated it for **small motions (nudges).**

We are looking for the motion parameters (r, c, θ) that minimize

$$\sum_{mn} \left(T(r,c,\theta) f_{mn} - g_{mn}\right)^2 = \left\| T(r,c,\theta)\,\vec{f} - \vec{g} \right\|_2^2$$

Replacing $T(r,c,\theta)\,\vec{f}$ with our linear approx $\vec{f}+Am$

$$\Rightarrow \min_m \left\| \vec{f}+Am-\vec{g} \right\|_2^2$$

This is a linear least-squares problem. I'll skip the details, but we can solve it like this:

$$A^T(\vec{f}+Am-\vec{g})=0$$

$$A^TAm = A^T(\vec{g}-\vec{f})$$

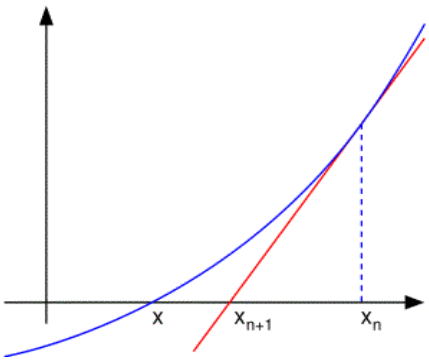
$$m = (A^TA)^{-1}A^T(\vec{g}-\vec{f})$$

A^TA is a 3x3 matrix, so this system is easy to solve.

As mentioned, we are only solving a linear approximation to our nonlinear problem. How can we claim to actually solve the nonlinear problem?

Newton's Method

To solve $f(x)=0$, $f:\mathbb{R}\to\mathbb{R}$,
iterate $x^{n+1} = x^n - \dfrac{f(x^n)}{f'(x^n)}$



until you reach a fixed point.
Let's derive a vector version.
We want to solve $L(m)=\vec{0}\in\mathbb{R}^N$ for $m\in\mathbb{R}^M$, where

$$L(m) = T(m)\vec{f}-\vec{g} \approx \vec{f}+Am-\vec{g}$$

Taylor's thm...

$$\vec{r}(m+\Delta m) = L(m) + \underbrace{\nabla L(m)}_{A\in\mathbb{R}^{N\times M}}\Delta m + O(\Delta m^2) = \vec{0}$$

$$L(m+\Delta m) = L(m) + \nabla L(m)\,\Delta m + O(\Delta m^2) = \vec{0}$$

$$\Rightarrow\ L(m) + A\Delta m \approx \vec{0}$$

$$A^T A \Delta m = A^T L(m)$$

$$\Delta m = (A^T A)^{-1} A^T (T(m)\vec{f} - \vec{g})$$

$$\therefore\ m^{t+1} = m^t + (A^T A)^{-1} A^T (T(m^t)\vec{f} - \vec{g})$$

Note that the A matrices also depend on m.

In this iterative scheme, one starts with a guess for m, then computes $T(m)\vec{f}$ and A. At each iteration, m is updated, and so is $T(m)\vec{f}$ and A.

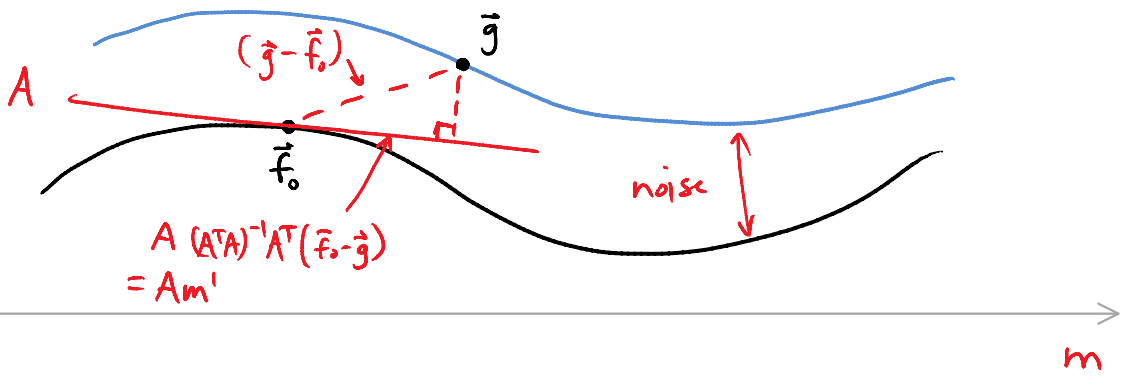
Once it converges to a fixed point, we have

$$(A^T A)^{-1} A^T (T(m)\vec{f} - \vec{g}) = \vec{0}$$

This is called **"Fixed-Point Iteration".**

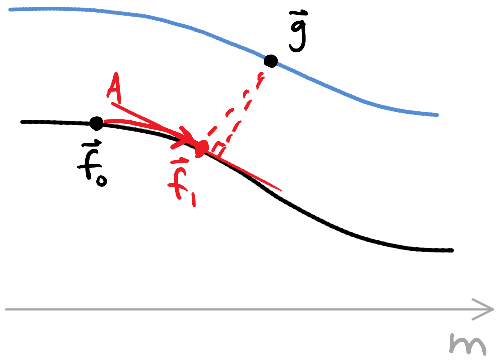
Geometrical Interpretation

Define $\vec{f}_t = T(m^t)\,\vec{f}$

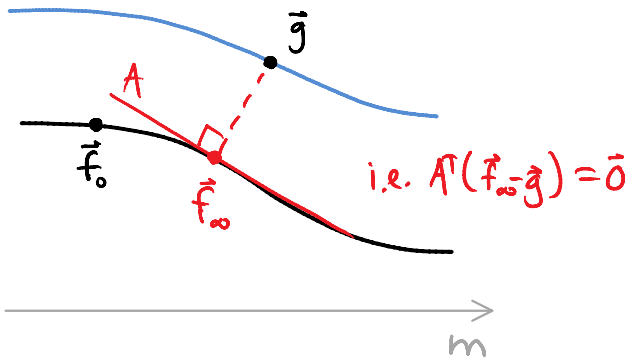


The curves represent all the different images you can get by moving $\vec{f}$ or $\vec{g}$ (ie. by moving along the m-axis). I call these curves **"motion manifolds".**
Their motion manifolds are similar, but separated because of noise.

Next iteration...



Convergence.



Implementation detail: You might have noticed that A has to be re-computed every iteration.
Here is a trick. Instead of approximating $T(m)\vec{f}$, we approximate

$$T^{-1}(m)\vec{g} = \vec{g} - \nabla_{\vec{g}} m = \vec{g} - \tilde{A}m$$

This is a slightly different $\tilde{A}$ than the one we get from approx. $T(m)\vec{f}$.

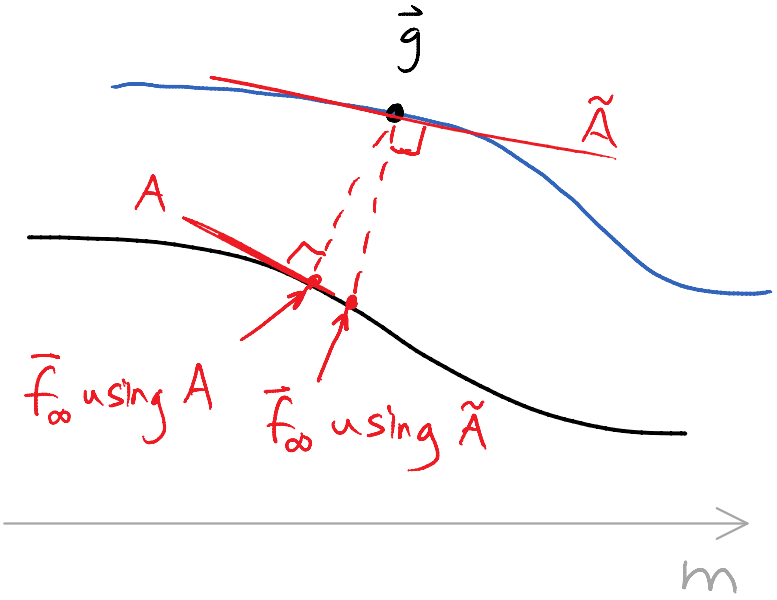
Now $\tilde{A}^T(\vec{f} - T^{-1}(m)\vec{g}) = \vec{0}$

$$\Rightarrow \tilde{A}^T\left(\vec{f} - (\vec{g} - \tilde{A}m)\right) = \tilde{A}^T(\vec{f} - \vec{g}) + \tilde{A}^T\tilde{A}m = \vec{0}$$

$$\Rightarrow m = (\tilde{A}^T\tilde{A})^{-1}\tilde{A}^T(\vec{f} - \vec{g})$$

Same as before, but with a different $\tilde{A}$.

$\vec{g}$


$\vec{g}$
$\tilde{A}$
A
$\vec{f}_\infty$ using A
$\vec{f}_\infty$ using $\tilde{A}$
m