CS 477/577

Lecture Three

Spectral Image Formation II: Calibration
AND
Non-homogenous least squares

Review

Image Formation (deluxe version)

The response of an image capture system to a light signal $L(\lambda)$ associated with a given pixels is modeled by

$$G^{(k)} = F^{(k)}(C^{(k)}) = F^{(k)}\left(b^{(k)} + \underbrace{\int L(\lambda) \; S^{(k)}(\lambda) \; d\lambda}_{\text{from before}} \right)$$

where $S^{(k)}(\lambda)$ is the sensor response function for the k^{th} channel and $b^{(k)}$ is the k^{th} channel response to black.

$S^{(k)}(\lambda)$ includes the contributions due to the aperture, focal length, sensor position in the focal plane.

$F^{(k)}$ accounts for typical non-linearities such as gamma.

Review

Image Formation (deluxe version)

The discrete version says that response of an image capture system to a light signal represented by the vector $\mathbf{L}$ is modeled by

$$G^{(k)} = F^{(k)}(C^{(k)}) = F^{(k)}\left(b^{(k)} + \underbrace{\mathbf{L} \bullet \mathbf{S}^{(k)}}_{\text{Key part for today}} \right)$$

where $\mathbf{S}^{(k)}$ is now the vector representation of the sensor response function for the k^{th} channel.

Our focus today will be the linear part, $C^{(k)} = \mathbf{L} \bullet \mathbf{S}^{(k)}$, ignoring camera black, $b^{(k)}$.

Math aside, #1

Systems of Linear Equations

- Non-homogenous (likely most familiar)

$$
\begin{aligned}
2x_1 + 3x_2 &= 5 \\
x_1 - x_2 &= 0
\end{aligned}
$$

Solved by $x_1 = 1,\ x_2 = 1$

Math aside, #1

Systems of Linear Equations

- Non-homogenous (more equations than unknown)

$$\begin{bmatrix} 2 & 3 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \\ 3 \end{bmatrix}$$

Math aside, #1

Systems of Linear Equations

- Homogenous (next week)

$$\begin{bmatrix} 2 & 3 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \vec{0}$$

Such that
$x_1^2 + x_2^2 = 1$

Math aside, #1

Linear Least Squares

- Very common problem in vision: solve an over-constrained system of linear equations
  - e.g., $U\mathbf{x}=\mathbf{y}$, where U has more rows than needed

- No exact solution, but if the equations come from noisy data, we do not expect an exact solution to be correct anyway.
  - More equations allows multiple measurements to be used

- Least squares means that you minimize squared error (the difference between your model and your data)
  - Instead of $U\mathbf{x} = \mathbf{y}$, we say $U\mathbf{x} \approx \mathbf{y}$, and find the "best" $\mathbf{x}$,

Math aside, #1

Linear Least Squares

- Least squares minimization is (relatively) easy

- Not very robust to outliers (it assumes error is Gaussian)

Linear Least Squares

We will look at two problems

First, $U\mathbf{x} = \mathbf{y}$ where U has more rows than needed

Second, $U\mathbf{x} = 0$ subject to $\left|\mathbf{x}\right| = 1$ where U has more rows than needed

We can use the **first** for naïve spectral camera calibration.

We use the **second** problem for geometric camera calibration.

Non-homogeneous Least Squares*

Problem one: $U\mathbf{x} = \mathbf{y}$ where U has more rows than needed (this shows up as an optional question in HW 01)

U is not square, so inverting it does not work

In fact, usually **there is no solution**. We need to redefine what it means to “solve the equation”.

We seek the “best” answer but what is that?

* This is regression by a different name.

Non-homogeneous Least Squares

Define $\mathbf{e} = U\mathbf{x} - \mathbf{y}$ and $E(\mathbf{x}) = |\mathbf{e}|^{\mathbf{2}} = \mathbf{e}^T\mathbf{e}$

E() is one idea of "best", **U** and **y** are given by your problem and data.

The least squares solution is the **x** that has minimum E.

In other words, we solve $\arg\min_{\mathbf{x}}\left(E(\mathbf{x})\right)$

The answer is given by

$$\mathbf{x} = U^{\dagger}\mathbf{y} \text{ where } U^{\dagger} = (U^T U)^{-1} U^T \text{ is the pseudoinverse of U}$$

Non-homogeneous Least Squares

Define $\mathbf{e} = U\mathbf{x} - \mathbf{y}$ and $E(\mathbf{x}) = |\mathbf{e}|^2 = \mathbf{e}^T \mathbf{e}$

The least squares answer is given by

$\mathbf{x} = U^{\dagger}\mathbf{y}$ where $U^{\dagger} = (U^T U)^{-1} U^T$ is the pseudoinverse of U

We can derive this by differentiating with respect to each x_i, and setting all resulting equations to zero (see supplementary slides).

For speed and numerical stability, one may want to use a different approach to solve $U^T U\mathbf{x} = U^T \mathbf{y}$ without matrix inversion (e.g., Matlab `linsolve()` (or \ operator) which uses QR factorization).

Important

Non-homogeneous linear least squares summary
(the part you need to know)

You should be able to set up for problems where it makes sense to say:

$$U\mathbf{x} = \mathbf{y}$$

You should know that it can be solved in the least squares sense by

$\mathbf{x} = U^{\dagger}\mathbf{y}$ where $U^{\dagger}$ is the pseudoinverse of U

You should understand what it means to solve in the least squares sense.

For exams, you can assume that you can look up

$$U^{\dagger} = (U^T U)^{-1} U^T$$

Quick "derivation" of formula for linear least squares

$U\mathbf{x} \cong \mathbf{y}$ (U has more rows than columns)

$U^T U\mathbf{x} \cong U^T \mathbf{y}$ (Multiply both sides by U^T)

$U^T U$ is likely to be robustly invertable

So, $\mathbf{x} \cong \left(U^T U\right)^{-1} U^T \mathbf{y} = U^{\dagger}\mathbf{y}$

A more formal derivation **showing** that this answer minimizes the squared error is included in the posted notes as "supplemental material".

Example

Express this system of equations in matrix-vector form, provide the algebraic form of the answer, and use Matlab to compute it.

```
2a + 3b = 14
a  +  b = 5
a  +  b = 6
a  +  b = 7
2a -  b = 10
```

The answer (I)

$U\mathbf{x} = \mathbf{y}$,

where $U = \begin{bmatrix} 2 & 3 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 2 & -1 \end{bmatrix}$ and $y = \begin{bmatrix} 14 \\ 5 \\ 6 \\ 7 \\ 10 \end{bmatrix}$.

$$\mathbf{x} = U^{\dagger}\mathbf{y} = (U^T U)^{-1} U^T \mathbf{y}$$

The answer (II)

```
>> U=[2 3; 1 1; 1 1; 1 1;2 -1]

U =

     2     3
     1     1
     1     1
     1     1
     2    -1

>> y=[14;5;6;7;10]

y =

    14
     5
     6
     7
    10
```

```
>> x = inv(U'*U)*U'*y

x =

    5.4043
    0.9362

>> % Second method

>> x = U \ y

x =

    5.4043
    0.9362
```

Supplementary (optional) material (not covered in class).

Problem statement. Find **x** that minimizes E where

$$E = |\mathbf{e}|^2 = \mathbf{e}^T \mathbf{e} \quad \text{where} \quad \mathbf{e} = U\mathbf{x} - \mathbf{y}$$

For a minimum, $\dfrac{\delta E}{\delta x_i} = 0 \, , \; \forall x_i$

(given no boundary conditions)

$$E = \sum_j e_j^2$$

$$\frac{\delta E}{\delta x_i} = 2 \sum_j \frac{\delta e_j}{\delta x_i} \bullet e_j = 2 \frac{\delta \mathbf{e}^T}{\delta x_i} \mathbf{e}$$

Supplementary (optional) material (not covered in class).

$$\frac{\delta E}{\delta x_i} = 2\frac{\delta \mathbf{e}^T}{\delta x_i}\mathbf{e} = 0 \quad \text{(for minimum)}$$

This is true for all components, $x_{i,}$ so we get:

$$\begin{pmatrix} \cdots \\ \\ \dfrac{\delta \mathbf{e}^T}{\delta x_i} \\ \\ \cdots \end{pmatrix} \mathbf{e} = 0$$

Supplementary (optional) material (not covered in class).

The next step then is to evaluate $\dfrac{\delta \mathbf{e}^T}{\delta x_i}$

to get each row of a matrix, A, where A$\mathbf{e}$=0

$$\frac{\delta \mathbf{e}^T}{\delta x_i} = \left(\frac{\delta \mathbf{e}}{\delta x_i} \right)^T = \left(\frac{\delta}{\delta x_i} (U\mathbf{x} - \mathbf{y}) \right)^T = \left(\frac{\delta}{\delta x_i} U\mathbf{x} \right)^T$$

Supplementary (optional) material (not covered in class).

Each row of A is $\left(\frac{\delta}{\delta x_i} U\mathbf{x}\right)^T$

$$\left(U\mathbf{x}\right)_k = \sum_j U_{kj} x_j$$

(Let's study the k'th element of $U\mathbf{x}$)

$$\frac{\delta}{\delta x_i}\left(U\mathbf{x}\right)_k = U_{ki}$$

Supplementary (optional) material (not covered in class).

$$\frac{\delta}{\delta x_i}(U\mathbf{x})_k = U_{ki}$$ (k'th element of i'th column of U)

So $\frac{\delta}{\delta x_i}(U\mathbf{x})$ is the i'th column of U

And so $\frac{\delta \mathbf{e}^T}{\delta x_i} = \left(\frac{\delta}{\delta x_i} U\mathbf{x}\right)^T$ is the i'th row of U^T

So, the matrix referred to as A before, is U^T

Supplementary (optional) material (not covered in class).

$\frac{\delta \mathbf{e}^T}{\delta x_i} = \left(\frac{\delta}{\delta x_i} U \mathbf{x} \right)^T$ is the i'th row of U^T

So $\begin{pmatrix} \cdots \\ \\ \frac{\delta \mathbf{e}^T}{\delta x_i} \\ \\ \cdots \end{pmatrix} \mathbf{e} = 0$ becomes $U^T (U\mathbf{x} - \mathbf{y}) = 0$

Supplementary (optional) material (not covered in class).

From the previous slide our condition is $U^T(U\mathbf{x} - \mathbf{y}) = 0$

Or $U^T U\mathbf{x} = U^T\mathbf{y}$ (same as we got with our algebriac manipulation "proof")

So $\mathbf{x} = (U^T U)^{-1} U^T \mathbf{y}$

Thus

$\mathbf{x} = U^\dagger \mathbf{y}$ where $U^\dagger = (U^T U)^{-1} U^T$ is the pseudoinverse of U

Naïve spectral camera calibration

The camera has a spectral sensitivity $S^{(k)}(\lambda)$ for each k. So how do we find them out?

Notation. Use the discrete (vector) version of the formula, and do each k separately, and so use $\mathbf{S}^{(k)}$. Also, ignore camera black and the non-linear (gamma) mapping F().

Data. To find $\mathbf{S}^{(k)}$ we can measure many light signals, $\mathbf{L}_i$ and corresponding responses, $\mathbf{C}_i$.

Then, each measurement, i, provides an equation constraining $\mathbf{S}^{(k)}$.

$$C_i = \mathbf{L}_i \bullet \mathbf{S}^{(k)}$$

Important

Non-homogeneous linear least squares

(example one---naïve spectral camera calibration)

Strategy: measure some spectra entering the camera, $\mathbf{L}_i$, and note the response, C_i.

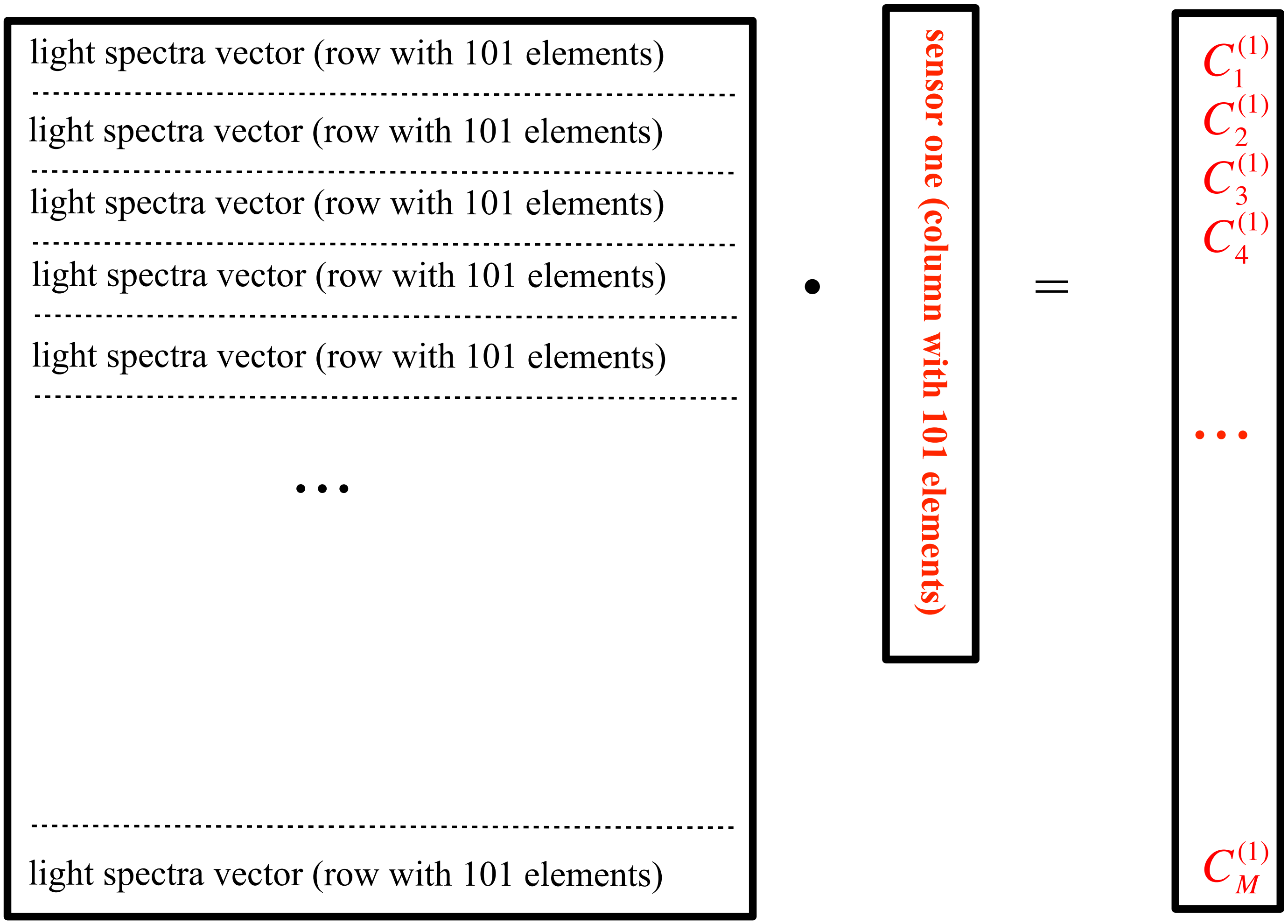
So we have, for a bunch of measurements, *i*:

$$C_i = \mathbf{L}_i \bullet \mathbf{S}^{(k)}$$

If we don't have enough measurements, then the problem is under constrained.

Because of noise, we want to use multiple measurements. IE, we want the problem to be over constrained (too many rows).

In pictures, for one sensor (say red, k=1)



Important

Naïve spectral camera calibration

From previous:

$$C_i = \mathbf{L}_i \bullet \mathbf{S}^{(k)}$$

(for a number of measurements indexed by *i*.)

We form a matrix L with rows $\mathbf{L}_i$, a vector **C** with elements C_i, and solve

$$L\,\mathbf{S}^{(k)} \cong \mathbf{C}$$

in the least squares sense (**S** is the unknown).

And we now know that this solutions is given by $\mathbf{S}^{(k)} = L^{\dagger}\mathbf{C}$

Spectral camera calibration improvements

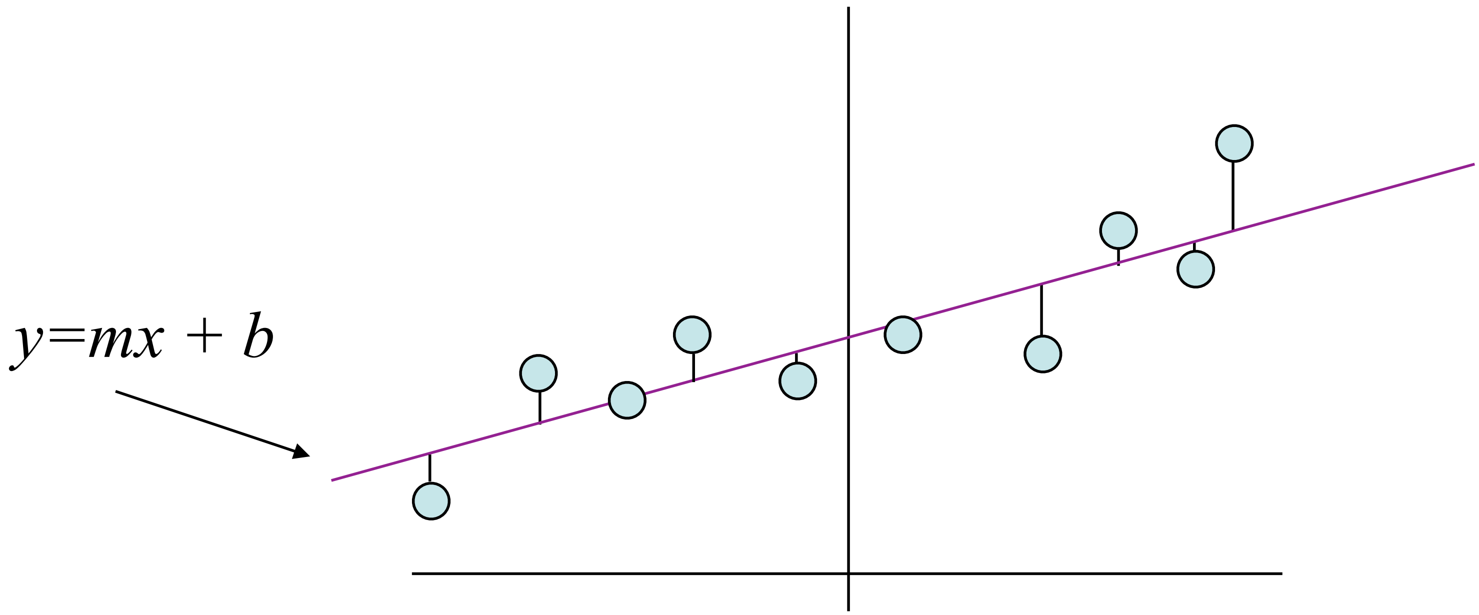
A) Constrain the sensitivities to be positive

B) Promote the sensitivity functions to be smooth
(This is often referred to as regularization).

How to do this? See grad student part of assignment two.

Important

Non-homogeneous linear least squares (example two---naïve line fitting)



The error contribution for each point is the vertical distance (illustrated) squared.

We fit the line by tweaking *m* and *b* so that the sum of the contributions is as small as possible.

Important

Non-homogeneous linear least squares (example two---naïve line fitting)

Can write $y=mx + b$ as:

$$(x\ 1)*(m\ b) = y$$

Important

Non-homogeneous linear least squares (example two---naïve line fitting)

Can write $y=mx + b$ as:

$(x\ 1)*(m\ b) = y$

So form

a matrix U with rows $(x_i\ 1)$

a vector $\mathbf{y}$ with elements y_i

a vector of unknowns $\mathbf{x}=(m, b)$

and use the formula to solve $U\mathbf{x}\cong\mathbf{y}$

Line fitting using non-homogeneous linear least squares

- We are minimizing $\sum_i \left(y_i - \left(m x_i + b \right) \right)^2$

- Only the vertical deviations count

- Asymmetric with respect to the axis
  - If you switch x and y, you will get a different answer

- Terminology note: This is standard *linear regression*

- Modeling note: The assumption here is that y_i are generated from from x_i, m, b, with added Gaussian noise.

Line fitting using non-homogeneous linear least squares (example)

Fit a line to (-1,2) (0,1) (1,2)

$$U = \begin{pmatrix} -1 & 1 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \qquad y = \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} m \\ b \end{pmatrix} = U^{\dagger} y = \left(U^T U\right)^{-1} U^T y$$

Using Matlab, we get

$$\begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 0 \\ 5/3 \end{pmatrix}$$ which is inutively reasonable (if we plot the points).