6.801/6.866: Machine Vision, Lecture 3

Professor Berthold Horn, Ryan Sander, Tadayuki Yoshitake
MIT Department of Electrical Engineering and Computer Science
Fall 2020

These lecture summaries are designed to be a review of the lecture. Though I do my best to include all main topics from the lecture, the lectures will have more elaborated explanations than these notes. Therefore, if you're looking for the most rigorous review and treatment of these topics, we encourage you to rewatch the lecture videos. With that said, we hope these summaries are beneficial for your learning. If you have any feedback for these lecture summaries, please submit it **here**.

1 Lecture 3: Time to Contact, Focus of Expansion, Direct Motion Vision Methods, Noise Gain

1.1 Noise Gain

Example/motivation: **Indoor GPS**. Rather than using localization of position with satellites, use indoor cellular signals.

Fun fact: EM waves travel at 1 ns/foot.

Dilution of Precision:

- How far off GPS is w.r.t. your location.
- Important to note that your dilution of precision can vary in different directions - e.g. horizontal precision is oftentimes greater than vertical position.

1.2 Forward and Inverse Problems of Machine Vision

1.2.1 Scalar Case

One way to conceptualize the goals of machine vision, as well as to highlight why noise gain is important, is by considering the following problems with some one-dimensional input x and an output $y = f(x)$:

- The **forward problem**: $x \rightarrow y \triangleq f(x)$
- The **inverse problem***: $y \triangleq f(x) \rightarrow x$
  *(this term comes up a lot in machine vision, computer graphics, and robotics)

In machine vision, we oftentimes focus on solving the **inverse problem**, rather than the **forward problem**. Intuitively, we usually observe some $y = f(x)$, and from this infer the latent parameters x using our model f.

When it is possible to express the inverse of a function in closed form or via a matrix/coefficient, we can simply solve the **inverse problem** using: $x = f^{-1}(y)$.

More importantly, to build a robust machine vision system to solve this inverse problem, it is critical that small perturbations in $y = f(x)$ do not need to large changes in x. Small perturbations need to be taken into account in machine vision problems because the sensors we use exhibit **measurement noise**. The concept of **noise gain** can come in to help deal with this uncertainty.

Consider a perturbation δy that leads to a perturbation δx when we solve the inverse problem. In the limit, as $\delta \in \mathbb{R} \rightarrow 0$, then we arrive at the definition of noise gain:

$$\text{noise gain} = \frac{\delta x}{\delta y} = \frac{1}{f'(x)} = \frac{1}{\frac{df(x)}{dx}} \tag{1}$$

Like other concepts/techniques we've studied so far, let's understand when this system fails. Below are two cases; we encourage to consider why they fail from both a mathematical and intuitive perspective (hint: for the mathematical component, look at the formula above, and for the intuitive component, think about how the effect on x from a small change in y in a curve that is nearly flat):

- $f'(x) = 0$ (flat curve)
- $f'(x) \approx 0$ (nearly flat curve)

1.2.2 Vector Case

Now that we've analyzed this problem in the scalar case, let us now consider it in the vector/multi-dimensional case. Since images are inherently multidimensional, this more general vector case is where we will find ourselves.

First, we can restate these problems:

- **Forward Problem**: $\mathbf{x} = \mathbf{Mb}, \mathbf{M} \in \mathbb{R}^{m \times n}$ for $m, n \in \mathbb{N}$
- **Inverse Problem**: $\mathbf{b} = \mathbf{M}^{-1}\mathbf{x}, \mathbf{M} \in \mathbb{R}^{m \times n}$ for $m, n \in \mathbb{N}$

But how good is this answer/approach? If $\mathbf{x}$ changes, how much does $\mathbf{b}$ change? We quantify noise gain in this case as follows:

$$\text{“noise gain”} \rightarrow \frac{||\delta \mathbf{b}||}{||\delta \mathbf{x}||}, \delta \in \mathbb{R} \tag{2}$$

*__NOTE__: This multidimensional problem is more nuanced because we may be in the presence of an **anisotropic** (spatially non-uniform) noise gain - e.g. there could be little noise gain in the x_1 direction, but a lot of noise gain in the x_2 direction.

As in the previous case, let's analyze when this approach fails. To do this, let's consider M^{-1} to help build intuition for why this fails:

$$M^{-1} = \frac{1}{\det|M|} \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \tag{3}$$

Let's ignore the specific entries of M^{-1} for now, and focus on the fact that we need to compute the determinant of M. When is this determinant zero? This will be the case whenever there exists **linear dependence** in the columns of $\mathbf{M}$. As we saw before, two cases that can yield to poor performance will be:

- $\det|M| = 0$: This corresponds to a non-invertible matrix, and also causes the noise term to blow up.
- $\det|M| \approx 0$: Though this matrix may be invertible, it may cause numerical instability in the machine vision system, and can also cause the noise term to blow up.

Let's also revisit, just as a refresher, the inverse of a 2×2 matrix:

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{4}$$

$$\tag{5}$$

$$\mathbf{A}^{-1} = \frac{1}{\det \mathbf{A}} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{6}$$

Now let's verify that this is indeed the inverse:

$$\mathbf{A}^{-1}\mathbf{A} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} ad - bc & -ab + ab \\ cd - cd & ad - bc \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}_2 \tag{7}$$

1.3 Review from Lecture 2

Before we dive into the next set of concepts, let's also revisit some of the concepts discussed in the previous lectures.

1.3.1 Two-Pixel Motion Estimation, Vector Form

First, let's recall the two pixel motion estimation set of equations:

$$uE_{x_1} + vE_{y_1} + E_{t_1} = 0 \tag{8}$$
$$uE_{x_2} + vE_{y_2} + E_{t_2} = 0 \tag{9}$$

Rewritten in matrix form:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{E_{x_1}E_{y_2} - E_{y_1}E_{x_2}} \begin{bmatrix} E_{y_2} & -E_{y_1} \\ -E_{x_2} & E_{x_1} \end{bmatrix} \tag{10}$$

Take note of the denominator on the right-hand side. Does this term look familiar to the determinant term above? We were indeed solving an instance of the **inverse problem**. If this determinant-like quantity ($E_{x_1}E_{y_2} - E_{y_1}E_{x_2}$) is small, the noise is greatly amplified. We saw that this happens when brightness gradients are similar to one another.

1.3.2 Constant Brightness Assumption, and Motion Equation Derivation

Recall the constant brightness assumption's mathematical formulation:

$$\textbf{Constant Brightness Assumption} \implies \frac{dE}{dt} = 0 \tag{11}$$

*(Please note the quantity above is a total derivative.)

Intuition Behind This: As the object/camera moves, the physical properties of the camera do not change and therefore the total derivative of the brightness w.r.t. time is 0. From the chain rule, we can rewrite this total derivative assumption:

$$\frac{dE}{dt} = 0 \implies uE_x + vE_y + E_t = 0 \tag{12}$$

*(Recall this is for when x and y are parameterized w.r.t. time, i.e. $x = x(t), y = y(t)$.)

The above constraint is known as the **Brightness Change Constraint Equation (BCCE)**.

1.3.3 Optical Mouse Problem

Recall our motion estimation problem with the optical mouse, in which our objective is no longer to find the point where the **BCCE** is strictly zero (since images are frequently corrupted by noise through sensing), but to minimize the LHS of the BCCE, i.e:

$$\min_{u,v}\{J(u,v) \triangleq \iint (uE_x + vE_y + E_t)^2 dxdy\} \tag{13}$$

We solve the above using unconstrained optimization and by taking a "least-squares" approach (hence why we square the LHS of the BCCE). Solve by setting the derivatives of the two optimizing variables to zero:

$$\frac{dJ(u,v)}{du} = 0, \frac{dJ(u,v)}{dv} = 0 \tag{14}$$

1.3.4 Perspective Projection

Recall the perspective projection equations in the scalar form:

$$\frac{x}{f} = \frac{X}{Z} \textbf{ (x-component)}, \frac{y}{f} = \frac{Y}{Z} \textbf{ (y-component)} \tag{15}$$

*(Note that capital coordinates are in the world space, and lowercase coordinates are in the image space.)

What if these quantities are changing in the world w.r.t. time? Take time derivatives:

$$\frac{1}{f}\frac{dx}{dt} = \frac{1}{Z}\frac{dX}{dt} - \frac{1}{Z^2}X\frac{dZ}{dt} \tag{16}$$

Writing these for x and y:

- **x**: $\frac{1}{f}u = \frac{1}{Z}U - \frac{W}{Z}\frac{X}{Z}$
- **y**: $\frac{1}{f}v = \frac{1}{Z}V - \frac{W}{Z}\frac{Y}{Z}$

Note again the following definitions:

- $u \rightarrow$ image velocity in the x direction
- $v \rightarrow$ image velocity in the y direction
- $U \rightarrow$ world velocity in the X direction
- $V \rightarrow$ world velocity in the Y direction

When are these points in (u, v) space interesting? When $u = v = 0$ - this is the **Focus of Expansion (FOE)**. The **FOE** given by (x_0, y_0) in two dimensions:

$$(x_0, y_0) = (\frac{f}{Z}\frac{U}{W}, \frac{f}{Z}\frac{V}{W}) \tag{17}$$

1.4 Time to Contact (TTC)

In the previous lecture, we discussed the derivation of Time to Contact (TTC) in terms of meters:

$$\frac{Z}{W} \triangleq= \frac{Z}{\frac{dZ}{dt}} = \frac{\text{meters}}{\frac{\text{meters}}{\text{seconds}}} = \text{seconds} \tag{18}$$

Let us express the inverse of this **Time to Contact (TTC)** quantity as C:

$$C \triangleq \frac{W}{Z} = \frac{1}{\text{TTC}} \tag{19}$$

Let us now suppose we parameterize our spatial velocities u and v according to $u = Cx, v = Cy$, where C is the inverse TTC value we introduced above. Then, substituting these into the BCCE equation, we have:

$$\textbf{Recall BCCE:} \quad uE_x + vE_y + E_t = 0 \tag{20}$$

$$\textbf{Substitute:} \quad C(xE_x + yE_y) + E_t = 0 \tag{21}$$

$$\textbf{Solve for } C\textbf{:} \quad C = -\frac{E_t}{xE_x + yE_y} \tag{22}$$

The denominator in the derivation of C is the **"radial gradient"**:

$$g = xE_x + yE_y = (x, y) \cdot (E_x, E_y) \tag{23}$$

Building Intuition: If we conceptualize 2D images as topographic maps, where brightness is the third dimension of the surface (and the spatial dimensions x and y comprise the other two dimensions), then the brightness gradient is the direction of steepest **ascent** up the brightness surface.

Another note: (x, y) is a radial vector, say in a polar coordinate system. Hence why the above dot product term is coined the name "radial gradient". This gradient is typically normalized by its L_2/Euclidean norm to illustrate the multiplication of the brightness gradients with a radial unit vector):

$$g = \sqrt{x^2 + y^2}(\frac{x}{\sqrt{x^2 + y^2}}, \frac{y}{\sqrt{x^2 + y^2}}) \cdot (E_x, E_y) \tag{24}$$

This g quantity can be thought of as: "How much brightness variation is in an outward direction from the center of the image?"

For a more robust estimate, let us again employ the philosophy that estimating from more data points is better. We'll again take a least-squares approach, and minimize across the entire image using the parameterized velocities we had before. In this case, since we are solving for inverse Time to Contact, we will minimize the error term over this quantity:

$$\min_C \{J(C) \triangleq \iint (C(xE_x + yE_y) + E_t)^2 dxdy\} \tag{25}$$

Without the presence of measurement noise, the optimal value of C gives us an error of zero, i.e. perfect adherence to the **BCCE**. However, as we've seen with other cases, this is not the case in practice due to noise corruption. We again will use unconstrained optimization to solve this problem.

Taking the derivative of the objective $J(C)$ and setting it to zero, we obtain:

$$\frac{dJ(C)}{dC} = 0 \implies 2\iint (C(xE_x + yE_y) + E_t)(xE_x + yE_y)dxdy = 0 \tag{26}$$

This in turn gives us:

$$C\iint (xE_x + yE_y)^2 dxdy + \iint (xE_x + yE_y)E_t dxdy = 0 \tag{27}$$

$$\frac{1}{\text{TTC}} = C = -\frac{\iint (xE_x + yE_y)E_t dxdy}{\iint (xE_x + yE_y)^2 dxdy} \tag{28}$$

A few notes here:

- E_t can be computed by taking differences between a pixel at different points in time.
- To implement a framework like this, we can do so with **accumulators**.
- This is an instance of **"direct computation"**.
- When objects/important components of a scene are far away, we can combine image pixels and run these algorithms at **multiple scales** - this allows us to compute/analyze motion velocities over different timescales. This can also be helpful for building more computationally-tractable motion estimation implementations, since the number of pixels over which computations must occur can be reduced quadratically with the scale.
- Note one problem we run into with this approach - each pixel we apply this estimation approach to introduces one equation, but two unknowns.
- Note that neighboring pixels typically exhibit similar behavior to one another.

Let's briefly return to the concept of the radial gradient. When is this zero?

- $E = 0$ everywhere (coal mine)
- $(x, y) \cdot (E_x, E_y) = 0$ (radial gradient is zero)

Now, let's take a more general case when we have world motion, i.e. $U \neq 0, V \neq 0$. For our two components x and y:

- **x-component**:

$$\frac{u}{f} = \frac{U}{Z} - \frac{X}{f}\frac{W}{Z} \tag{29}$$

$$u = \frac{fU}{Z} - X\frac{W}{Z} \tag{30}$$

$$u = A - XC \tag{31}$$

$$\text{Where: } A \triangleq \frac{fU}{Z}, C \triangleq \frac{W}{Z} \tag{32}$$

- **y-component**:

$$\frac{v}{f} = \frac{V}{Z} - \frac{Y}{f}\frac{W}{Z} \tag{33}$$

$$v = \frac{fV}{Z} - Y\frac{W}{Z} \tag{34}$$

$$v = B - YC \tag{35}$$

$$\text{Where: } B \triangleq \frac{fV}{Z}, C \triangleq \frac{W}{Z} \tag{36}$$

Note that for the world quantities A and B, we also have the following identities (note that the Focus of Expansion (FOE) is given by the point (x_0, y_0):

- $A = \frac{fU}{Z} = Cx_0$
- $B = \frac{fV}{Z} = Cy_0$

Building Intuition: "As I approach the wall, it will loom outward and increase in size."

Now returning to our **BCCE**, this time with A, B, and C:

$$AE_x + BE_y + C(xE_x + yE_y) + E_t = 0 \tag{37}$$

We can again use least-squares to minimize the following objective enforcing the BCCE. This time, our optimization aim is to minimize the objective function $J(A, B, C)$ using the quantities A, B, and C:

$$\min_{A,B,C}\{J(A,B,C)\} \triangleq \iint (AE_x + BE_y + C(xE_x + yE_y) + E_t)^2 dxdy\} \tag{38}$$

Use unconstrained optimzation with calculus and partial derivatives to solve. Since we have three variables to optimize over, we have three first-order conditions (FOCs):

- $\frac{dJ(A,B,C)}{dA} = 0$
- $\frac{dJ(A,B,C)}{dB} = 0$
- $\frac{dJ(A,B,C)}{dC} = 0$

Using the Chain rule for each of these FOCs, we can derive and rewrite each of these conditions to obtain 3 equations and 3 unknowns. Note that $G \triangleq xE_x + yE_y$.

- **A** variable:

$$2\iint (AE_x + BE_y + C(xE_x + yE_y))E_x = 0 \tag{39}$$

$$A\iint E_x^2 + B\iint E_xE_y + C\iint GE_x = -\iint E_xE_t \tag{40}$$

- **B** variable:

$$2\iint (AE_x + BE_y + C(xE_x + yE_y))E_x = 0 \tag{41}$$

$$A\iint E_yE_x + B\iint E_y^2 + C\iint GE_y = -\iint E_yE_t \tag{42}$$

- **C** variable:

$$2\iint (AE_x + BE_y + C(xE_x + yE_y))E_x = 0 \tag{43}$$

$$A\iint GE_x + B\iint GE_y + C\iint G^2 = -\iint GE_t \tag{44}$$

Putting all of these equations together:

$$\begin{aligned}
A\iint E_x^2 + B\iint E_xE_y + C\iint GE_x &= -\iint E_xE_t \\
A\iint E_yE_x + B\iint E_y^2 + C\iint GE_y &= -\iint E_yE_t \\
A\iint GE_x + B\iint GE_y + C\iint G^2 &= -\iint GE_t
\end{aligned}$$

This can be compactly written as a matrix-vector product equation:

$$\begin{bmatrix} \iint E_x^2 & \iint E_xE_y & \iint E_xG \\ \iint E_yE_x & \iint E_y^2 & \iint E_yG \\ \iint GE_x & \iint GE_y & \iint G^2 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = -\begin{bmatrix} \iint E_xE_t \\ \iint E_yE_t \\ \iint GE_t \end{bmatrix}$$

As in the time-to-contact problem above, this can again be implemented using **accumulators**.

Let's end on a fun fact: Did you know that optical mice have frame rates of 1800 fps?