6.801/6.866: Machine Vision, Lecture 9

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.

1 Lecture 9: Shape from Shading, General Case - from First Order Nonlinear PDE to five ODEs

In this lecture, we will begin by exploring some applications of magnification, shape recovery, and optics through Transmission and Scanning Electron Microscopes (TEMs and SEMs, respectively). Then, we will discuss how we can derive shape from shading using needle diagrams, which capture surface orientations at each (x, y) pixel in the image. This procedure will motivate the use of Green's Theorem, "computational molecules", and a discrete approach to our standard unconstrained optimization problem. We will conclude by discussing more about recovering shape from shading for Hapke surfaces using initial curves and rotated coordinate systems.

1.1 Example Applications: Transmission and Scanning Electron Microscopes (TEMs and SEMs, respectively)

We will begin with a few motivating questions/observations:

- **How do TEMs achieve amazing magnification?** They are able to do so due to the fact that these machines are not restricted by the wavelength of the light they use for imaging (since they are active sensors, they image using their own "light", in this case electrons.
- **What are SEM images more enjoyable to look at than TEM images?** This is because SEM images reflect shading, i.e. differences in brightness based off of surface orientation. TEM images do not do this.
- **How do SEMs work?** Rely on an electron source/beam, magnetic-based scanning mechanisms, photodiode sensors to measure secondary electron current. Specifically:
  - Many electrons lose energy and create secondary electrons. Secondary electrons are what allow us to make measurements.
  - Secondary electron currents vary with surface orientation.
  - Objects can be scanned in a raster-like format.
  - Electron current is used to modulate a light ray. Magnification is determined by the degree of deflection.
  - Gold plating is typically used to ensure object is conductive in a vacuum.
  - Inclines/angles can be used for perturbing/measuring different brightness values.
  - From a **reflectance map** perspective, measuring brightness gives is the **slope** (a scalar), but it does not give us the gradient (a vector). This is akin to knowing speed, but not the velocity.

1.2 Shape from Shading: Needle Diagram to Shape

This is another class of problems from the overarching "Shape from Shading" problem. Let us first begin by defining what a needle diagram is:

A **needle diagram** is a 2D representation of the surface orientation of an object for every pixel in an image, i.e. for every (x, y) pair, we have a surface orientation (p, q), where $p \triangleq \frac{dz}{dt}, q \triangleq \frac{dz}{dy}$. Recall from photometric stereo that we cannot simply parameterize $Z(x, y)$; we can only parameterize the surface gradients $p(x, y)$ and $q(x, y)$.

In this problem, our goal is that given (p, q) for each pixel (i.e. given the needle diagram), recover z for each pixel. Note that this leads to an overdetermined problem (more constraints/equations than unknowns) [1]. This actually will allow us to reduce noise and achieve better results.

For estimating z, we have:

$$x : z(x) = z(0) + \int_0^x p dx'$$
$$y : z(y) = z(0) + \int_0^y q dy'$$
$$x\&y : z(x, y) = z(0, 0) + \int p dx' + q dy'$$

Let us define $\delta x' = p dx' + q dy$. Next, we construct a contour in the (x, y) plane of our (p, q) measurements, where the contour starts and ends at the origin, and passes through a measurement. Our goal is to have the integrals of p and q be zero over these contours, i.e.

$$\oint (p dx' + q dy') = 0$$

This is equivalent to "z being conserved" over the contour.

But note that these measurements are noisy, and since we estimate p and q to obtain estimates for z, this is not necessarily true.

Note that an easy way to break this problem down from one large problem into many smaller problems (e.g. for computational parallelization, greater accuracy, etc.) is to decompose larger contours into smaller ones - if z is conserved for a series of smaller loops, then this implies z is conserved for the large loop as well.

1.2.1 Derivation with Taylor Series Expansion

Let us now suppose we have a point (x, y), centered around a square loop with lengths equal to δ. Then, applying the formula above, we have that:

$$p\Big(x, y - \frac{\delta y}{2}\Big)\delta x + q\Big(x + \frac{\delta x}{2}, y\Big)\delta y - p\Big(x, y + \frac{\delta y}{2}\Big)\delta x - q\Big(x - \frac{\delta x}{2}, y\Big)\delta y = 0$$

If we now take the first-order Taylor Series Expansion of this equation above, we have:

$$\textbf{Expansion}: \ \Big(p(x, y) - \frac{\delta y}{2}\frac{\partial p(x, y)}{\partial y}\Big)\delta x + \Big(q(x, y) + \frac{\delta x}{2}\frac{\partial q(x, y)}{\partial x}\Big)\delta y - \Big(p(x, y) + \frac{\delta y}{2}\frac{\partial p(x, y)}{\partial y}\Big)\delta x - \Big(q(x, y) - \frac{\delta x}{2}\frac{\partial q(x, y)}{\partial x}\Big)\delta y = 0$$

$$\textbf{Rewriting}: \ p(x, y)\delta x - \frac{\delta y \delta x}{2}\frac{\partial p(x, y)}{\partial y} + q(x, y)\delta y + \frac{\delta x \delta y}{2}\frac{\partial q(x, y)}{\partial x} = p(x, y)\delta x + \frac{\delta y \delta x}{2}\frac{\partial p(x, y)}{\partial y} + q(x, y)\delta y - \frac{\delta x \delta y}{2}\frac{\partial q(x, y)}{\partial x}$$

$$\textbf{Simplifying}: \ \delta y \delta x \frac{\partial p(x, y)}{\partial y} = \delta x \delta y \frac{\partial q(x, y)}{\partial x}$$

$$\textbf{Solution}: \ \frac{\partial p(x, y)}{\partial y} = \frac{\partial q(x, y)}{\partial x}$$

This is consistent with theory, because since our parameters $p \approx \frac{\partial z}{\partial x}$ and $q \approx \frac{\partial z}{\partial y}$, then the condition approximately becomes (under perfect measurements):

$$\frac{\partial p(x,y)}{\partial y} = \frac{\partial}{\partial y}(\frac{\partial z}{\partial x}) = \frac{\partial^2 z}{\partial y \partial x}$$

$$\frac{\partial q(x,y)}{\partial x} = \frac{\partial}{\partial x}(\frac{\partial z}{\partial y}) = \frac{\partial^2 z}{\partial x \partial y}$$

$$\frac{\partial p(x,y)}{\partial y} = \frac{\partial q(x,y)}{\partial x} \implies \frac{\partial^2 z}{\partial y \partial x} = \frac{\partial^2 z}{\partial x \partial y}, \text{ Which is consistent with Fubini's Theorem as well [2].}$$

Though this will not be the case in practice we are measuring p & q, and thus we will encounter some measurement noise.

1.2.2 Derivation with Green's Theorem

We can now show the same result via Green's Theorem, which relates contour integrals to area integrals:

$$\oint_L (Ldx + Mdy) = \iint_D \Big(\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y}\Big) dxdy$$

Where the term $Ldx + Mdy$ on the lefthand side is along the boundary of the contour of interest, and the term $\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y}$ is along the interior of the contour.

Green's Theorem is highly applicable in machine vision because we can reduce two-dimensional computations to one-dimensional computations. For instance, Green's Theorem can be helpful for:

- Computing the area of a contoured object/shape
- Computing the centroid of a blob or object in two-dimensional space, or more generally, geometric moments of a surface. Moments can generally be computed just by going around the boundary of a contour.

Let us now apply Green's Theorem to our problem:

$$\oint_L (pdx + qdy) = \iint_D \Big(\frac{\partial q(x,y)}{\partial x} - \frac{\partial p(x,y)}{\partial y}\Big) dxdy = 0$$

This requires $\frac{\partial q(x,y)}{\partial x} - \frac{\partial p(x,y)}{\partial y} = 0 \implies \frac{\partial q(x,y)}{\partial x} = \frac{\partial p(x,y)}{\partial y} \; \forall \, x, y \in D$.

We could solve for estimates of our unknowns of interest, p and q, using unconstrained optimization, but this will be more difficult than before. Let us try using a different tactic, which we will call "Brute Force Least Squares":

$$\min_{z(x,y)} \iint_D \Big(\frac{\partial z}{\partial x} - p\Big)^2 + \Big(\frac{\partial z}{\partial y} - q\Big)^2 dxdy$$

I.e. we are minimizing the squared distance between the partial derivatives of z with respect to x and y and our respective parameters over the entire image domain D.

However, this minimization approach requires having a finite number of variables, but here we are optimizing over a continuous function (which has an infinite number of variables). Therefore, we have infinite degrees of freedom. We can use **calculus of variations** here to help us with this. Let us try solving this as a discrete problem first.

1.2.3 Shape with Discrete Optimization

For this, let us first take a grid of unknowns $\{z_{ij}\}_{(i,j) \in D}$. Our goal is to minimize the errors of spatial derivatives of z with respect to p and q, our measurements in this case (given by $\{p_{ij}, q_{ij}\}_{(i,j) \in D}$. Our objective for this can then be written as:

$$\min_{\{z_{ij}\}} \sum_i \sum_j \Big(\frac{z_{i,j+1} - z_{i,j}}{\epsilon} - p_{i,j}\Big)^2 + \Big(\frac{z_{i+1,j} - z_{i,j}}{\epsilon} - q_{i,j}\Big)^2$$

Note that these discrete derivatives of z with respect to x and y present in the equation above use finite forward differences.

Even though we are solving this discretely, we can still think of this as solving our other unconstrained optimization problems, and therefore can do so by taking the first-order conditions of each of our unknowns, i.e. $\forall\ (k,l) \in D$. The FOCs are given by $|D|$ equations (these will actually be linear!):

$$\frac{\partial}{\partial z_{k,l}}(J(\{z_{i,j}\}_{(i,j)\in D}) = 0 \ \forall\ (k,l) \in D$$

Let us take two specific FOCs and use them to write a partial differential equation:

- **(k, l) = (i, j)**:

$$\frac{\partial}{\partial z_{k,l}}(J(\{z_{,j}\}_{(i,j)\in D}) = \frac{2}{\epsilon}\Big(\frac{z_{k,l+1} - z_{k,l}}{\epsilon} - p_{k,l}\Big) + \frac{2}{\epsilon}\Big(\frac{z_{k+1,l} - z_{k,l}}{\epsilon} - q_{k,l}\Big) = 0$$

- **(k, l) = (i+1, j+1)**:

$$\frac{\partial}{\partial z_{k,l}}(J(\{z_{,j}\}_{(i,j)\in D}) = \frac{2}{\epsilon}\Big(\frac{z_{k,l} - z_{k,l-1}}{\epsilon} - p_{k,l-1}\Big) + \frac{2}{\epsilon}\Big(\frac{z_{k,l} - z_{k-1,l}}{\epsilon} - q_{k-1,l}\Big) = 0$$

Gathering all terms (we can neglect the zs):

(1) $\frac{p_{k,l} - p_{k,l-1}}{\epsilon} \approx \frac{\partial p}{\partial x}$

(2) $\frac{q_{k,l} - q_{k-1,l}}{\epsilon} \approx \frac{\partial q}{\partial y}$

(3) $\frac{1}{\epsilon^2}((-z_{k,l+1} - z_{k+1,l} - z_{k,l-1} - z_{k-1,l}) + 4z_{k,l}) = 0 \approx -\Delta z = -\nabla^2 z$ **The Laplacian of** z

Where **(1)** + **(2)** + **(3)** $= 0$

Using the approximations of these three terms, our equation becomes:

$$\frac{\partial p}{\partial x} + \frac{\partial q}{\partial y} - \Delta z = 0 \implies \frac{\partial p}{\partial x} + \frac{\partial q}{\partial y} = \Delta z$$

This approach motivates the use of "computational molecules".

1.2.4 "Computational Molecules"

These are computational molecules that use finite differences [3] to estimate first and higher-order derivatives. They can be thought of as filters, functions, and operators that can be applied to images or other multidimensional arrays capturing spatial structural. Some of these are (please see the handwritten lecture notes for what these look like graphically):

1. $z_x = \frac{1}{\epsilon}(z(x,y) - z(x-1,y))$ (Backward Difference), $\frac{1}{\epsilon}(z(x+1,y) - z(x,y))$ (Forward Difference)
2. $z_y = \frac{1}{\epsilon}(z(x,y) - z(x,y-1))$ (Backward Difference), $\frac{1}{\epsilon}(z(x,y+1) - z(x,y))$ (Forward Difference)
3. $\Delta z = \nabla^2 z = \frac{1}{\epsilon^2}(4z(x,y) - (z(x-1,y) + z(x+1,y) + z(x,y-1) + z(x,y+1)))$
4. $z_{xx} = \frac{\partial^2 z}{\partial x^2} = \frac{1}{\epsilon^2}(z(x-1,y) - 2(x,y) + z(x+1,y))$
5. $z_{yy} = \frac{\partial^2 z}{\partial y^2} = \frac{1}{\epsilon^2}(z(x,y-1) - 2(x,y) + z(x,y+1))$

These computational molecules extend to much higher powers as well. Let us visit the Laplacian operator $\Delta(\cdot)$. This operator comes up a lot in computer vision:

- Definition: $\Delta z = \nabla^2 z = (\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y})^T(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}) = \frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2}$
- The Laplacian is the lowest dimensional **rotationally-invariant** linear operator, i.e. for a rotated coordinate system (x', y') rotated from (x, y) by some rotation matrix $\mathbf{R} \in SO(2)$, we have:

$$z_{x'x'} + z_{y'y'} = z_{xx} + z_{yy}$$

  I.e. the result of the Laplacian is the same in both coordinate systems.

As we can see, the Laplacian is quite useful in our derived solution above.

1.2.5 Shape with Discrete Optimization Cont.

Let us return to our discrete optimization problem. Our derivation above is a **least-squares** solution. This turns out to be the discrete version of our original continuous problem. Since these first-order equations are linear, we can solve them as a system of linear equations with Gaussian elimination. But note that these processes take $O(N^3)$ time. We can avoid this complexity by taking advantage of the sparsity in these equations.

Iterative Approach: The sparse structure in our First-Order Equations allows us to use an iterative approach for shape estimation. Our "update equation" updates the current depth/shape estimate $z_{k,l}$ using its neighboring indices in two dimensions:

$$z_{k,l}^{(n+1)} = \frac{1}{4}(z_{k,l+1}^{(n)} + z_{k+1,l}^{(n)} + z_{k,l-1}^{(n)} + z_{k-1,l}^{(n)}) - \epsilon(p_{k,l} - p_{k,l-1}) - \epsilon(q_{k,l} - q_{k-1,l})$$

A few terminology/phenomenological notes about this update equation:

- The superscripts n and $n+1$ denote the number of times a given indexed estimate has been updated (i.e. the number of times this update equation has been invoked). It is essentially the iteration number.
- The subscripts k and l refer to the indices.
- The first term on the righthand side $\frac{1}{4}(\cdot)$ is the local average of $z_{k,l}$ using its neighbors.
- This iterative approach converges to the solution much more quickly than Gaussian elimination.
- This iterative approach is also used in similar ways for solving problems in the **Heat and Diffusion Equations** (also PDEs).
- This procedure can be parallelized so long as the computational molecules do not overlap/touch each other. For instance, we could divide this into blocks of size 3 x 3 in order to achieve this.
- From this approach, we can develop robust surface estimates!

1.2.6 Reconstructing a Surface From a Single Image

Recall this other shape from brightness problem we solved for Hapke surfaces (Lecture 8). For Hapke surfaces, we have that our brightness in the image (radiance) L is given by:

$$L = \sqrt{\frac{\cos\theta_i}{\cos\theta_e}} = \sqrt{\frac{\hat{\mathbf{n}} \cdot \hat{\mathbf{s}}}{\hat{\mathbf{n}} \cdot \hat{\mathbf{v}}}}$$

Recall from our last lecture that this gives us a simple reflectance map of straight lines in gradient (p, q) space. By rotating this gradient space coordinate system from $(p, q) \to (p', q')$, we can simplify our estimates for shape.

With this rotation, we also claimed that rotating the system in gradient space is equivalent to using the same rotation matrix $\mathbf{R}$ in our image space (x, y). Here we prove this:

$$\textbf{Rotating Points}: \; x' = x\cos\theta - y\sin\theta, \; y' = x\sin\theta + y\cos\theta$$

$$\textbf{Reverse Rotating Points}: \; x = x'\cos\theta + y'\sin\theta, \; y = -x'\sin\theta + y'\cos\theta$$

$$\textbf{Taking Derivatives}: \; \frac{\partial z}{\partial x'} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial x'} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial x'}, \; \frac{\partial z}{\partial y'} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial y'} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial y'}$$

$$\textbf{Combining the Above}: p' = p\cos\theta - q\sin\theta, \; q' = p\sin\theta + q\cos\theta$$

Then, in our rotated coordinate system where p' is along the brightness gradient, we have that:

$$p' = \frac{p_s p + q_s q}{\sqrt{p_s^2 + q_s^2}} = \frac{r_s E^2 - 1}{\sqrt{p_s^2 + q_s^2}}$$

(Where $p' \triangleq \frac{\partial z}{\partial x'}$ is the slope of the surface of interest in a particular direction. This phenomenon only holds for Hapke surfaces with linear isophotes. We can integrate this expression out for surface estimation:

$$z(x) = z(x_0) + \int_x^{x_0} p'(x)dx$$

Integrating out as above allows us to build a surface height profile of our object of interest. Can do this for the y-direction as well:

$$z(x, y) = z(x_0, y) + \int_{x}^{x_0} p'(x, y)dx$$

A few notes from this, which we touched on in lecture 8 as well:

- Adding a constant to z does not change our profile integrals, except by an offset. Therefore, in order to obtain absolute height measurements of z, we need to include initial values.
- In this case, we need an initial condition for every horizontal row/profile. This is the same as requiring an "initial curve", and allows us to effectively reduce our computations from 2D to 1D. Note from our previous lecture that these initial conditions are needed to determine the surface orientation/shape at interfaces between these different profiles.
- Let us examine what this looks like when we parameterize an initial curve with η:

  $$\text{Take } x(\eta), y(\eta), z(\eta), \text{ and rotate with } \xi$$

  Then we can compute δz to recover shape, along with δx and δy:

  1. $\delta x = \frac{q_s}{\sqrt{p_s^2+q_s^2}}\delta\xi$
  2. $\delta y = \frac{p_s}{\sqrt{p_s^2+q_s^2}}\delta\xi$
  3. $\delta z = \frac{[r_s E^2(x,y)-1]}{\sqrt{p_s^2+q_s^2}}\delta\xi$

  Note that we can adjust the speed of motion here by adjusting the parameter δ.

Next time, we will generalize this from Hapke reflectance maps to arbitrary reflectance maps!

1.3 References

1. Overdetermined System, https://en.wikipedia.org/wiki/Overdetermined system
2. Fubini's Theorem, https://en.wikipedia.org/wiki/Fubini%27s theorem
3. Finite Differences, https://en.wikipedia.org/wiki/Finite difference