Image Segmentation CS828 Spring '12

Angjoo Kanazawa

February 15, 2012

1 January 25th 2012 - Lecture 1

Definition: Segmentation (for this class):

- About low level vision in general
- Requires a lot of knowledge about th eworld, high level understanding, quite challenging.
- So we're going to focus on simpler segmentation that doesn't require that much knowledge about the world: Uniform surfaces, smooth shape. Still there will be varietion in intensity.
- Want to find uniform region in things (texture, color, motion, smoothness), not necessarily world property. Removed from true segmentation of objects but still useful.
- Image is an 2D geometric structure. Segmentation is clustering that takes advantage of this structure. Based on the assumption that near-by pixels have the same intensity.
-

We're going to look at

1. Diffusion
2. Anisotropic diffusion
3. Graph based algorithms: message passing, thinking of an image as a graph, every pixel is a node in a graph, edges to neighbors → Markov Random FIeld. Gives us a probablistic way to express the state of a node in relation to its neighbors. Usually NP-hard, but graph-cut and belief propagation algorithms still work. The biggest issue is when the number of labels is big.
4. Conditional Random Fields, a general version of MRF
5. Normalized Cut: form a graph
6. Wavelets

Math	Fourier transforms	Convolution	Diffusion
	Wavelets	Level sets	Riemannian Geometry
Current Research	Bilateral filtering (by Morel)		Texture Segmentation
	Cosegmentation		Affinity propagation

Workload

1. Reports (6 out of 8 papers):Be critical when reading papers, even if the paper is good, what is the really important. Learn to recognize, have a taste. (10%)
2. Presentations: 3 presentations per day, 15 min per paper 10 min each to discuss paper (15%)
3. a take home midterm, Final all on lecture material (50%)
4. Problem set/Project (25%)

2 January 30th - Lecture 2

2.1 Perceptual Grouping

- Putting pieces to preceive as a whole.
- Depends on the prior knowledge/statistics about the world.

History

- Behaviorists dominated in early 20th century, wanted to make psychology scientific, focused on quantifyiable things.
- Rejected anything introspective or mind building internal representations.
- AI, computers, chomsky killed behaviorists.
- Gestalt movement claimed visual system perceived world as a objects and surfaces, as a whole and not as raw atomic stimulus/intensities.

Classical principles/cues

- Knowing the role of edges is critical to how we perceive an image
- Similarity, Good continuation, Common Form, Connectivity, Symmetry (seems to jump out), Convexity, Closure, Common Fate, Paraallelism, Collinearity
- convexity beats symmetry? Connectivity also beats symmetry?

Theories

- We perceive shapes that are "good form": smooth curves,, pretty abstract
- Bayesian: organizaton that's most likely to be true. Not computationally friendly. Rather than checking all possible options, maybe we look for a certain small set of possiblities. Still doesn't explain everything
-

3 February 1st - Lecture 3: Fourier Transform

3.1 Mathematical representation

a point in a $\mathbf{R^2}$ can be represented in a coordinate. If $p = (7, 3)$, we really mean $p = 7(1, 0) + 3(0, 1)$. *Any* point can be represented by a linear combination of two vectors. The basis vectors are:

1. Span the entire space: every point in the space can be written by linear combinations of these vectors.
2. Orthogonal: If not, moving in one direction will mean you'll be moving in the another direction
3. Unit: if not, the distance from the origin will not be constant.

We can compute the bases by

1. Linear Projection (inner product with each basis)

$$p = (p \cdot (1, 0))(1, 0) + (p \cdot (0, 1))(0, 1) \tag{1}$$

2. Magnitude of a point $||p||^2 = x^2 + y^2$ v

3.2 Functions in $\mathbf{R}^1$

The domain of the function is $[0, 2\pi]$, and we'll deal with functions in $\mathbf{R}^1$.

Def: a delta function:

$$\delta_s(t) = \begin{cases} 0 & s \neq t \\ \infty & s = t \end{cases}, \int_0^{2\pi} \delta_s(t)dt = 1$$

We'll write functions by using delta functions as a basis.
In infinite dimensions,

$$f(t) = \int f_s(\delta_s(t))ds$$

is the same as (1) but in infinite dimensions. Tw basis are orthogonal if their inner products are 0, in infinite dimensions, this is taking the integral. So delta functions are orthogonal.

This is a bad representation in some ways. It doesn't converge to the right representation (the function) quickly: using countable number of delta functions will not be a good representation of the function because it will only be correct in those places. We also need a lot of co-efficients.

Differen Representation Divide the interval $[0, 2\pi]$ into short k intervals with width $\frac{2\pi}{k}$. Use a rectangle in a interval as basis. They are orthogonal, so we can scale these rectangles and set it to a height that is equal to the average of the function in that interval. We have a piece-wise representation of a function using a finite basis. As $k \to \infty$, the approximation gets better. The *Reimann integral*. Here, we're stuck with a cetain level of accuracy as we fix k.

To get an arbitrary accuracy, we can reuse basis from multiple ks. i.e. if we divide the interval in 2, then 4, etc, then we'll get many rectangles or infinite bases that are *not* orthogonal, but can represent any function with finite pieces.

Functions are uncountable, but we're trying to represent it as a countable set of bases. But this is okay because we enforce the functions to be continuous.

3.3 Fourier Series

The basis elements:

- Height of $\sqrt{\frac{1}{2\pi}}$
- $\frac{\cos(t)}{\sqrt{n}}$ all are multiplied by a constant so when integrated it is 1.
- $\frac{\sin(t)}{\sqrt{n}}$
- $\cos(2t)$, $\sin(2t)$

They are unit vectors (normalized) and they are orthonormal i.e. $\int \sin(t)\cos(t)dt = 0$. But better, draw them around π. sin is symmetric around π, cos is negative symmetric. So if they are multiplied together, the signs are different so they cancel and gives you 0.

Now, we can write any function as an infinite sum of these basis elements:

$$f(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos kt + \sum_{k=1}^{\infty} b_k \sin kt \tag{2}$$

If the sums were finite upto N, then $\lim N \to \infty ||f(t)|| = 0$. This is a better representation then the delta functions because if we use enough co-efficients we will get really good approximation to the function.

$\cos^{2n}(t/2)$: Look at waht $\cos(t/2)$ look like, then raise it to a higher power. Really quicly, it will peak and look more like a delta function. By adding a constant in, $\cos(t/2 + a)$, we can shift the peaks.

Becasue we know that we can approximate any function with infinite delta functions, this means we can also do it with these basis. There are couple of identities by trigonometry to write higher power trig functions as a single power functions. i.e. trig functions with different frequences: $sin^2(t/2) = \frac{1-\cos(t)}{2}$, $sin^2(t) = \frac{1-\cos 2t}{2}$

Intuition: In practice, functions are smooth and with very small coefficients we can get a very good approximations.

Notation

$$\cos kt + i \sin kt = e^{ikt} \tag{3}$$

There are simple ways of computing these coefficients a_k, b_k. If we want a_k, we **take the inner product** of the function and $\cos kt$ i.e. $\int f(t)\cos ktdt$.

Complex case Given

$$c_k = \langle f, e^{ikt}, \rangle = \langle f, \cos kt \rangle + i\langle f, \sin kt \rangle,$$

$c_{-k} = \langle f, e^{i-kt}, \rangle = \langle f, \cos kt \rangle - i\langle f, \sin kt \rangle$

Then

$$c_k e^{ikt} + c_{-k} e^{-ikt} = a_k \cos kt + b_k \sin kt \tag{4}$$

We get back to the fourier representation.

Following from $a \sin t + b \cos t = c \cos(t + k)$, k is the phase, or the shift of functions.

Parsevaal's Theorem: Same as the pythagorean theorem:

$$\int f^2(t)dt = \frac{\pi}{2} a_0^2 + \pi \sum (a_k^2 + b_k^2)$$

This is good to use to measure how good our approximation is. So We can do

$$||(\int f(t) - a_0 - \sum_{k=1}^{N} a_k \cos kt - \sum_{k=1}^{N} b_k)^2|| = ||(\sum_{N+1}^{\infty} a_k \cos kt - \sum_{N+1}^{\infty} b_k)^2||$$

3.4 Fourier Transform

Let $f(t)$ is periodic going from $[0, 2\pi l]$. Then, we can represent $f(t)$ by

$$f(t) = \sum c_k e^{ikt/l}$$

(By dividing with l, we're stretching the basis element in $[0, 2\pi]$.) As $l \to \infty$, this gives us every possible fraction, all of $\mathbf{Q}$. Which mean we write this as:

$$f(t) = \int_{-\infty}^{\infty} F(k) e^{ikt} dk \tag{5}$$

Remember: e^{ikt} carries the orthonromal basis, now extending to all of $\mathbf{R}$, this means the coefficients are now in the ∞ domain so we write coefficients as $F(k)$, and call this the **Fourier transform** of $f(k)$.

(5) is the approximation of $f(t)$, the inverse operation to get the fourier transform is;

$$F(k) = \int_{-\infty}^{\infty} f(k) e^{-ikt} dk \tag{6}$$

e^{-ikt} is negative because it's the complex conjugate of e^{ikt}, (square it we multiply it with the complex conjugate.)

4 February 6th - Lecture 4: Smoothing & Convolution

Why do we *smooth* images? It's a way of passing information around, also it connects it more to segmentation (looking for a uniform property). When we smooth, we can take things that are similar and make them more similar. It also allows us to represent images in multiple scales, it helps us get rid of fine details, giving us coarser representations of an image. That is we want to remove high frequency portion and analyze the low frequency part.

Smoothing can be done by *convolution*.

In vision we always assume vision. Given a noisy input, the true intensity + noise, say $P_i = 100 + n_i$, smoothing takes the average of all pixels, we'll have

$$\begin{aligned} &= \frac{1}{M}\sum P_i \\ &= \frac{1}{M}\sum 100 + n_i \\ &= 100 + \frac{1}{M}\sum n_i \end{aligned}$$

A simple example of smoothing, the average of a lot of random variables makes the std of noise smaller.

4.1 1-D image

Think of 1-D images as a function: $f(t)$. We want to replace a pixel by the average of its neighbors. We write this as:

$$h(t) = \frac{1}{2\delta}\int_{t-\delta}^{t+\delta} f(t^{'})dt^{'} \tag{7}$$

(Where $t^{'}$ is just another points, not derivatives)

Let us define,

$$g(t) = \begin{cases} \frac{1}{2\delta} & \text{for } -\delta \leqslant t \leqslant \delta \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Note that $g(t)$ sums to 1. (Like a pdf of $U(-\delta, \delta)$.)

Now, we can write (7) as

$$h(t) = \int_{-\infty}^{\infty} f(t-u)g(u)du \tag{9}$$

It's as if we take the function g and winding it up so that it's centered around t and taking the inner product to get h. It flips, the left side of the filter is applied to the right side of point t. The resulted h is just shifting g at everypoint and taking the inner product.

We write this as

$$h(t) = g(t) * f(t) \tag{10}$$

It's natural to take the weighted average of your neighbors, because it's more likely that people around you have more information. So we use a gaussian filter.

4.2 Convolution

Two properties:

1. Linear: $g * kf = k(g * f)$, $g * (f_1 + f_2) = g * f_1 + g * f_2$
2. Shit-invariant: Take my function and translate, then convolve with a filter is same as convolving with a filter and shifting it. i.e. if $f'(t) = f(t + k)$, $h = g * f$, and $h^{'} = g * f^{'}$, then $h^{'} = h(t + k)$

Convolution Theorem Given F as the fourier transform of f, (F is the function of frequency), and the same for G, g, and H, h.

$$f * g = h \Leftrightarrow FG = H \tag{11}$$

Proof: $f * g = \int f(t-u)g(u)du$, call this h. To take the fourier transform of this, we take the inner product of h and e^{-itw}. So

$$H(w) = \int e^{-itw}(\int f(t-u)g(u)du)dt$$

Define $v = t - u$. Now,

$$\begin{aligned} H(w) &= \int e^{-itw}(\int f(t-u)g(u)du)dt \\ &= \int e^{-i(v+u)w}(\int f(v)g(u)du)dv \\ &= \int e^{-iuw}g(u)du \int e^{-ivw}f(v)dv \\ &= GF \end{aligned}$$

The sines and cosines are eigenvectors of functions because when you convolve it with any filter it just scales it.

The narrower the gaussian, the broader the fourier transform, The broader my gaussian, the narrower my fourier transform. So the lower frequency part gets preserved and the higher frequency (the edge of gaussian) gets reduced more.

Intuition: If the gaussian is so sharp that it's like a delta function, it'll only scale the function at that point t. Then, the fourier transform of a delta function is uniformly 1 at all frequencies. Because convolving with a delta function doesn't change anything, so $f * \delta = f$, bug $F * G = H = F$, so G is a uniformly 1 that doesn't change anything.

Similarly, if the gaussian is so broad that it's like a uniform function, then the fourier transform is like a delta function.

example A sinc function: $G(w) = \int_{-\delta}^{\delta} \frac{1}{2\delta} e^{-iwt}dt = \frac{2\sin(\delta w}{w}$. Plot it, bad fourier transform because the high frequency components go in and out. Compared, the gaussian filter provides us a very good fourier transform.

Why remove higher frequency components? If we assume the noise is i.i.d., we can show that the fourier transform of the noise is uniform. i.i.d. noise has equal energy. Bc this noise has the same energy everywhere, it's called the *white noise*. If we think of our image as some smooth pixels with a white

noise. Images tend to have much more low frequencies than high frequencies. Noise is equal in low and high frequency, so if you reduce the high frequency components, it significantly reduces the noise.

Band-pass filter: Looks like $U(a, b)$, it perfectly preserves the low frequency component. It's fourier transform is the sinc function. So there's limitation to use these perfect filters.

High-pass filter: inverse of $U(a, b)$.

Fourier series:

$$f(t) = a_0 + \sum a_k \cos(kt) + \sum b_k \sin(kt)$$

(K is the frequency) The derivative:

$$f'(t) = \sum -a_k \sin(kt) + \sum b_k \cos(kt)$$

This is also a fourier series, *taking the derivative has the effect of scaling the coefficients by the frequency.* As frequency gets higher, it amplifies the coefficients.

This is why it's dangerous to take the derivative of a noisy image, because the derivative ampliflies the high frequency components with a lot of noise.

Taking the derivative is like convolution.

Gaussian Filter For any function, the more spatially localized (peaked) it is, the broader it is in frequency. Vice versa.

5 February 8th - Lecture 5: Diffusion

Sampling theorem Given

$$f(t) = a_0 + \sum_{k=1}^{T} a_k \cos(kt) + \sum_{k=1}^{T} b_k \sin(kt) \text{ for } t = 0, \frac{2\pi}{M}, 2\frac{2\pi}{M}, 3\frac{2\pi}{M}$$

This equation is linear in unknown coefficients ($2T+1$ many) so we need $2T+1$ samples to solve this equation.

If we have an analog version of someone's speech, you can digitize it by taking $2T+1$ samples knowing they are band-limited (otherwise we'll lose information). Speech is fine but the best thing is to apply a band-pass filter to make sure that it's band-limited.

If the signal isn't band-limited to begin with, i.e. $f(t) = a_0 + \sum_{k=1}^{3T} a_k \cos(kt) + \sum_{k=1}^{3T} b_k \sin(kt)$, you have more unknowns with $2T+1$ samples, and you're ignoring a lot of information and it's totally meaning less.

Aliasing when high frequency is indistinguishable from low frequency when sampled.

5.1 Diffusion

Diffusing is like smoothing, provides a physical analogy to smoothing. By setting it up and solving diffusion as a PDE, we'll see that writing smoothing as PDE can be modified so that edges can be preserved?

Again, everything followed is in 1D, we go from discrete, continuous, then back to discrete.

Discrete Imagine you have a lot of small buckets with lots of particles in it, describe each bucket by how many things are in it, the *concentration*, $C(x,t)$, which changes over discrete time steps. i.e. $C(1,0)$ tells us how many particles are in bucket 1 at time 0. Some of these particles can jump to neighboring buckets. A reasonable model for physical diffusion (milk and coffee, heat, etc). *Flux* , $J(x,t)$, is the net number of particles that are moving in the positive direction.

Diffusion is *isotropic* (equally likely to go to left and right) and *homogenous* (same things happen everywhere).

The relationship between flux and concentration can be modelled as such:

$$J(x,t) = -D\frac{\partial C}{\partial x} \tag{12}$$

If more stuff goes to the left than the right, the flux is negative. D is a constant diffusion coefficient, what fraction of things move around, the "diffusivity" of particles. If D is low, less stuff moves around.

$$\frac{\partial C}{\partial t} = -\frac{\partial J}{\partial x} \tag{13}$$

How much C changes over time? Then I want to count how much is coming in and how much is coming out (flux). So if flux is constant, then the concentration is not changing. If the flux is increasing, that means there's more stuff going out to the right. So if the change of J is positive, the change in C is negative.

If D wasn't constant, it would depend on x, to do more interesting kind of smoothing, we can make D into a function of x.

Combine the equation to get rid of J by taking PDE wrt x:

$$\frac{\partial J}{\partial x} = -D\frac{\partial^2 C}{\partial x^2}$$

Plugging this back to (13), we get

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} \tag{14}$$

A positive second derivative means concavity, a local minima, so the concentration increases, similarly, concentration goes down at a local maxima second derivative negative. This means that concentration is being smoothed over time.

Numerical Analysis A finite differential problem. Taking the taylor series for a fixed t, we get

$$c_{i+1} = c_i + \delta x\frac{\partial C}{\partial x} + \frac{1}{2}\delta x^2\frac{\partial^2 C}{\partial x^2} + \mathcal{O}(\delta x^3) \tag{15}$$

(also $c_{i-1} = c_i - \delta x\frac{\partial C}{\partial x} + \frac{1}{2}\delta x^2\frac{\partial^2 C}{\partial x^2} + \mathcal{O}(\delta x^3)$)

Ignoring the higher order terms, you get, in first-order,

$$\frac{\partial C}{\partial x} = \begin{cases} \frac{c_{i+1}-c_i}{\delta x} \\ \frac{c_i-c_{i-1}}{\delta x} \end{cases} \tag{16}$$

Adding them together, we get

$$\frac{\partial C}{\partial x} = \frac{c_{i+1}-c_{i-1}}{2\delta x} \tag{17}$$

Better because this is symmetric.

Doing the same thing to the second derivative (difference between the first derivative of left and right)

$$\frac{\partial^2 C}{\partial x^2} = \frac{(c_{i+1}-c_i)-(c_i-c_{i-1})}{\delta x^2} \tag{18}$$

We could say similar thing to wrt to t:

$$\frac{\partial C}{\partial t} = \frac{c(x,t+1)-c(x,t)}{\delta t} \tag{19}$$

Putting all of this together, (14) becomes

$$\begin{aligned}
\frac{\partial C}{\partial t} &= D\frac{\partial^2 C}{\partial x^2} \\
\frac{C(i,t_0+1)-C(i,t_0)}{\delta t} &= D\frac{C(i+1,t_0)-2C(i,t_0)+C(i-1,t_0)}{\delta x^2} \\
C(i,t_0+1) &= C(i,t_0) + \frac{\delta t D}{\delta x^2}(C(i+1,t_0)-2C(i,t_0)+C(i-1,t_0)) \\
&= (1-2\lambda)C(i,t_0) + \lambda(i+1,t_0) + \lambda C(i-1,t_0)
\end{aligned}$$

Where $\lambda = \frac{\delta t D}{\delta x^2}$. This is just another convolution with a filter that looks like $l = (\lambda, 1 - 2\lambda, \lambda)$.

Let $C(x, 0) = f(x)$, to get the concentration at time n, I get the initial concentration at time 0 and apply convolusion with filter l t_0 times. i.e.

$$C(x, t_0) = (l \times l \times \cdots \times l) \times f$$

But since convolution is associative, we can combine the filters together. Convolving l with l over and over gives us a gaussian.

So *diffusion is just a convolution with gaussian, same thing as low-pass filtering an image, smoothing.*

Limitation on λ:

$$\begin{aligned} 1 - 2\lambda &> 0 \\ 1 - 2\frac{\delta t D}{\delta x^2} &> 0 \\ \frac{\delta x^2}{2D} &> \delta t \end{aligned}$$

Another intuition Consider a single particle that is diffusion¡ This is a random variable x_i, where $x_i = -\delta x$ if it moves left at time i, δx if it moves to right. After T time steps, the position of the particle is $\sum_{i=1}^{T} x_i$, where by LLN, this is a r.v. with 0 mean Gaussian distribution. So the particles position after T time steps is a Gaussian, we can get it by convolving x_0 with a Gaussian or convolving it with a filter T times, same thing.

6 February 13th - Lecture 6: Edge Detection

Paper Presentation Topics

- Graph based, MRF/CRF
- Texture: (texton-boost)
- **Co-segmentation**
- **Layout** (3D surface estimation) by Hoeim, Efros
- Affinity propagation by Frey (tronto)
- Edge detection: Malik, Basiri
- Graph Cuts - Galun 'Detecting and Sketching the Common"
- Semantic

6.1 Edge Detection: Canny Edge Detector

Basic Idea: look at sudden changes in intensity and the first derivative I_x. But we'd expect some noise, so we always have to smooth the image with a gaussian before we take the derivative.

Algorithm in 1D:

1. Smooth with a gaussian
2. Take the first derivative
3. (a) Is it strong? (of large magnitude) Everything above a certain threshold is strong, where image is changing rapidly.
   (b) Pick points that are not only strong but also a local extrema

This procedure is optimal to minimize the number of false detections, while also optimizing how well we localize the edge.

Couple of parameters: The threshold is the tradeoff between false positive/negativees, std of the Gaussian, σ, is the width of the filter, the wider it is the smoother I and less noise, but less accurately we'll localize the edge.

6.2 In 2D

In 2D, things are little bit more complicated. A rapid intensity change depends on the direction. We need to figure out which direction the intensity changes most rapidly.

Compute the image gradient, (I_x, I_y), and the magnitude $||(I_x, I_y)|| = \sqrt{I_x^2 + I_y^2}$. We can represent the direction with the maximal change by a unit vector:

$$\frac{(I_x, I_y)}{||(I_x, I_y)||} \tag{20}$$

Intuition: When you take the first derivatie, you assume the image is locally linear, like a tilted plane, where there's a direction with big change, where orthogonal to that direction change is flat.

We take gradient ∇I instead of a derivative in 2D, and in asking is it strong, we ask if $||\nabla I||$ big. It's bad if you don't use a big enough gaussian. We want to renormalize so that the filter sums to 1, otherwise it makes the image dimmer or lighter than it actually is.

Smoothing in 2D

Gaussian in 1D:

$$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{x^2}{2\sigma^2}}$$

Gaussian in 2D:

$$\begin{aligned} G(x,y) &= \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{x^2}{2\sigma^2}}e^{-\frac{y^2}{2\sigma^2}} \end{aligned}$$

(Maybe midterm: prove that these are same things) We can use seperable filters because we can divide the filter into 2 filters.

Picking extrema in 2D What defined an edge is the local extrema *in the direction of the gradient.* The orthogonal direction is where the edge continues. A subtlety: Given a vector field, every pixel has a vector which is its gradient. We want to know what the image gradient is where each vector is pointing, but there might not be any pixel ther. We can calculate what it would be if the pixel grid were dense by interpolating (linear or bi-linear) between two image locations.

Hysteresis to figure out if $||\nabla I||$ is big: if i pick a threshold that's too big, things might get fragmented and miss some edges. With a lower threshold, we'll get unneccessarily edges from noise in the background. We want the best of both. One heuristic to get this is to do the high threshold, then the low threshold to get weaker edges but only keep them if they're close to the stronger edges. This is a very basic perceptual grouping based on connectivity.

This is the prominent method for edge detection but it still doesn't really work in real images. No matter how you pick threshold, we get too much or too less. Because edges/boundaries that are intuitive to us may not be strong locally. Also around pointy edges/corners, smoothing weakens them.

6.3 Corner Detection

A way to define a corner is a smal region in the image where you have image gradient change in both directions. Look at a small window (5 x 5), in this window look at the image gradients (25). Do PCA, principal component analysis, on the gradients and find out how much variations there are in image gradients in one direction. If image gradient only goes in one direction, only one PCA will be strong, but if it goes in x and y direction, both PCA will be strong.

Compute a scatter matrix,

$$H = \begin{pmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_y I_x & \sum I_y I_y \end{pmatrix}$$

Where the first eigen value λ_2, smallest one, gives you how much gradient is in the direction of least. if both λ_2, λ_1 big, it means gradients change in all direction. This is call the harris corner detector.

7 February 15th - Lecture 7: Non-linear Diffusion

Presentation points:

- Understand what's the biggest contribution about the paper. No need to pay attention to all of th edetails in the paper.
- Give context of this paper from state-of-the-art, past, and how it fits
- Where significant problems identified, addressed?
- Give opinions
- Always understand the questions before answering it

7.1 Nonlinear Diffusion

Goal for today Non-homogenous diffusion: when you're near the boundary don't smooth so much. Anisotropic (# of particles leaving is not same in all direction):.

In 2D isotropic, the flux is in the direction of the negative gradient, down hill. In anisotropic, the matrial isn't necessarily going down hill, the direction depends on a larger context.

7.2 Review: Isotropic Diffusion

$f(x)$ is the image at time 0, and $u(x,t)$ is the image at time t Flux in 1D:

$$j(x,t) = -D\frac{\partial u}{\partial x}$$

flows to the negative direction of the derivative. D is how fast stuff diffuse. Flux in 2D:

$$j = -D\nabla u$$

If D is constant, this is gaussian smoothing in 2D, isotropic and homogenous. If we make $D = D(x)$ a function of location x, it's non-homogenous. D could also be a tensor, a 2x2 matrix, giving us two vectors, now diffusion is non-homogenous and anisotropic. Intuition in 1D:

$$\frac{\partial u}{\partial t} = -\frac{\partial j}{\partial x}$$

in 2D:

$$\frac{\partial u}{\partial t} = -\text{div}j$$

Where

$$\text{div}j = \frac{\partial j}{\partial x} + \frac{\partial j}{\partial y}$$

Substituting things, the heat equation in 1D

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$

in 2D

$$\frac{\partial u}{\partial t} = \text{div}(D\nabla u)$$

Gradient is a vector $\nabla u = (\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y})$. div$j$ is the sum of these, saying how much material is piling up in this location.

7.3 2D Non-homogenous, isotropic diffusion

Let $D = g(x)$, a function of the image. The amount of diffusion is different in locations but the way things diffuse is the same. Write $\frac{\partial u}{\partial t} = \text{div}(g(x)\nabla u)$.

We want to diffuse but not around the boundary, i.e. when the gradient is big, not. To do this, we can make g:

$$g(||\nabla f||^2) = \frac{1}{\sqrt{1 + \frac{||\nabla f||^2}{\lambda^2}}}$$

When $||\nabla f|| \to \infty$, $\sqrt{1 + \frac{||\nabla f||^2}{\lambda^2}} \to \frac{\lambda}{||\nabla f||}$, very small. So we'll smooth less and less as the gradient gets bigger. λ is set by hand, controls what's the point where we make things non-homogenous (where to stop diffusing). This gives a linear PDE, and we can't do this with a convolution anymore (because convolution applies same thing everywhere).

Problem with this approach is that it depends on the original image. Say we have a small gradient that gets smoothed out. It'll have less influence but small structures like this leave artifacts. Instead, we can take the gradient of the actual image (not the original f).

$$g(||\nabla f||^2) = \frac{1}{\sqrt{1 + \frac{||\nabla u||^2}{\lambda^2}}} \tag{21}$$

This is the **Perona-Malik** model. This model is *unstable*. Why? When a gradient really big at one point, its left neighbor is not getting anything, so the neighbor goes down (with a slow bumpy slope, we can get staircases) i.e. things are going in the opposite direction than usual diffusion (instead of hi to low, it can go from low to hi). So this is unstable. We can go around this by smoothing the image before taking the gradient.

7.4 2D Non-homogenous, anisotropic diffusion

Let D a 2x2 matrix based on the image. So back to $\frac{\partial u}{\partial t} = \text{div}(D\nabla u)$. D has eigenvectors v_1, v_2, v_1, where $v_1 \parallel \nabla u_\sigma$ and $v_2 \perp \nabla u_\sigma$. Set the eigenvalues λ_1, λ_2 to $\lambda_1 = g(||\nabla u_\sigma||^2)$ (the term in Perona-Malik) and $\lambda_2 = 1$.

If $\nabla u_\sigma = \nabla u$, we're just scaling λ_1 by sometime, and this is just Perona-Malik. u_σ is the original image smoothed by gaussian of size σ.

Intuition: around the boundary, at a finer scale, the gradient is showing noise. If ∇u_σ large, means that we're near a boundary. ∇u_σ has the largest direction v_1 (pointing towards the boundary), and a component orthogonal v_2. We can always take ∇u and decompose it into v_1, v_2. If ∇u_σ big, v_1 gets scaled down, but v_2 is totally preserved (it's just 1), so ∇u *points away from the boundary.*

These approaches are all still heuristics compared to the bilateral filters.