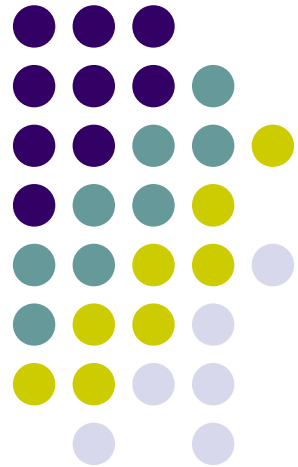
变分与泛函方法应用

Hongxin Zhang

2007-06-28

State Key Lab of CAD&CG, ZJU



主要内容

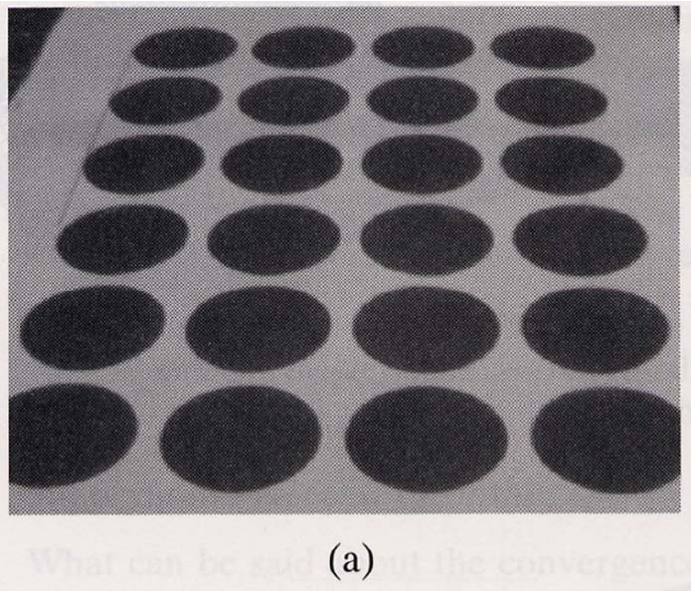
- Shape from shading
- Kernel methods

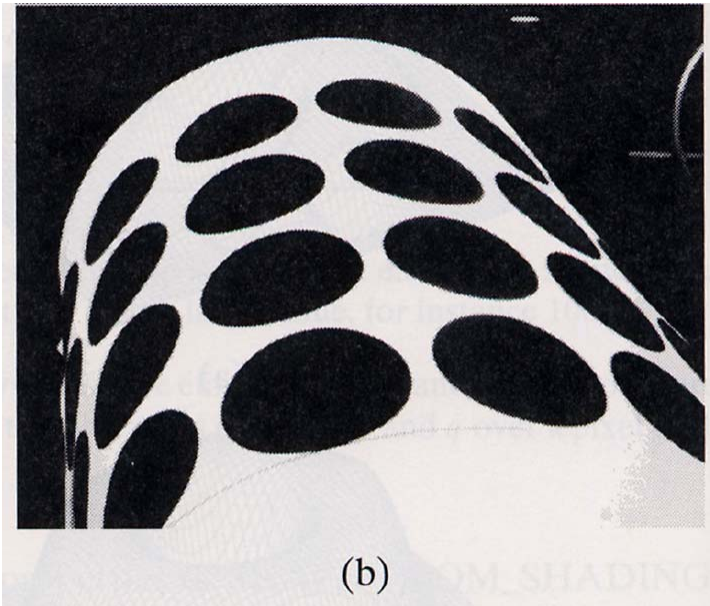
Shape from X

- Many cues can be used for inferring object shapes from images.
- Based on the number of images used, there are two categories of methods:
  - methods using multiple images
  - methods using a single image.
- A list of so called shape-from-X algorithms are shown in the following table.

Shape from	How many images	Method type
Stereo	2 or more	passive
Motion	a sequence	active/passive
Focus/defocus	2 or more	active
Zoom	2 or more	active
Contours	single	passive
Texture	single	passive
Shading	single	passive

Shape from texture



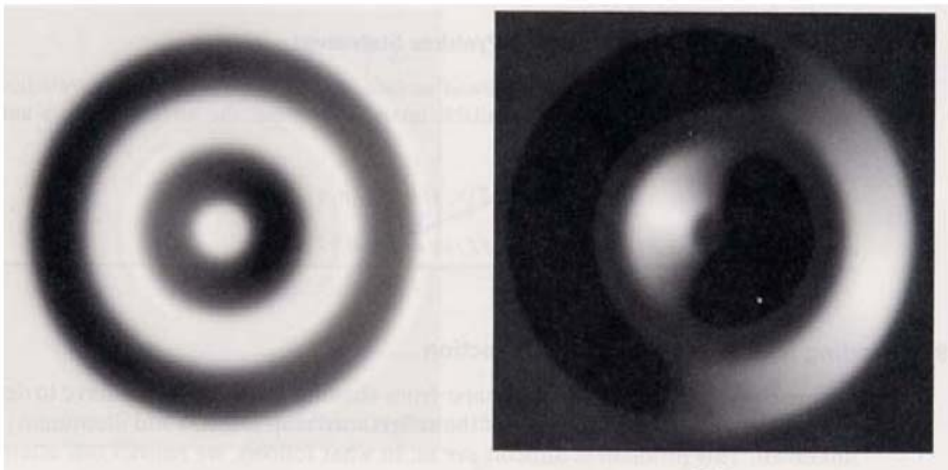


(a) image of a plane covered by a deterministic texture.

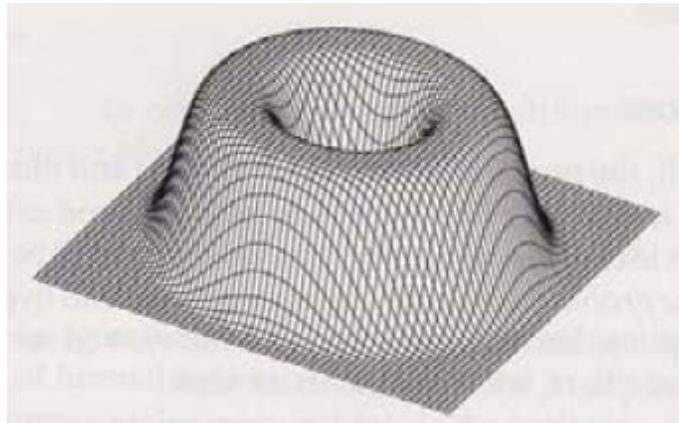
(b) the same texture on a curved surface

Shape from shading

- Shape can be recovered from a single image by human visual system based on the shading information
- To estimate the shape from a single image, some assumptions have to be made, e.g. constant albedo (surface color)
- This technique is very useful for reconstructing surfaces of planets from photographs acquired by spacecrafts

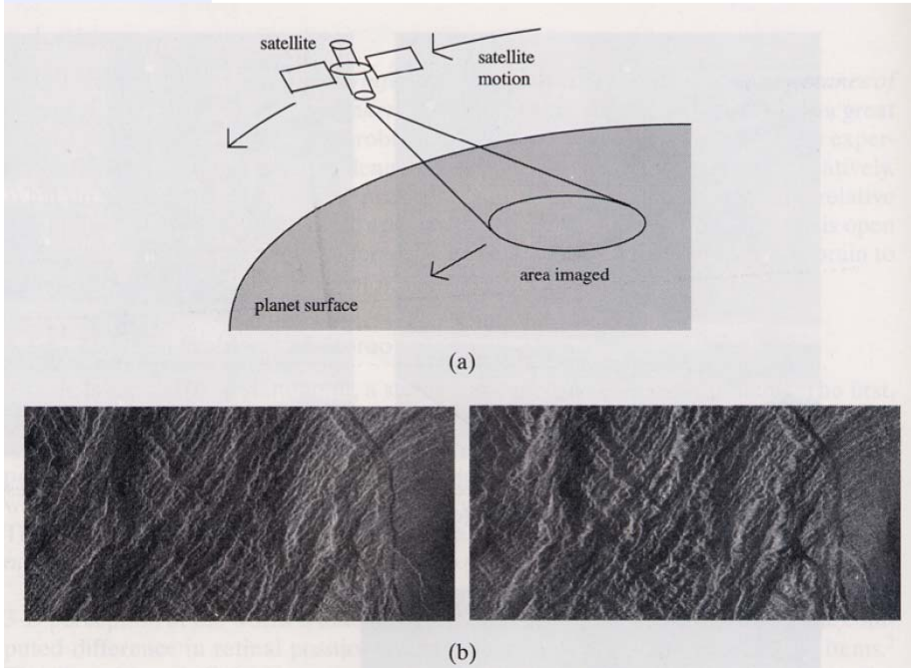


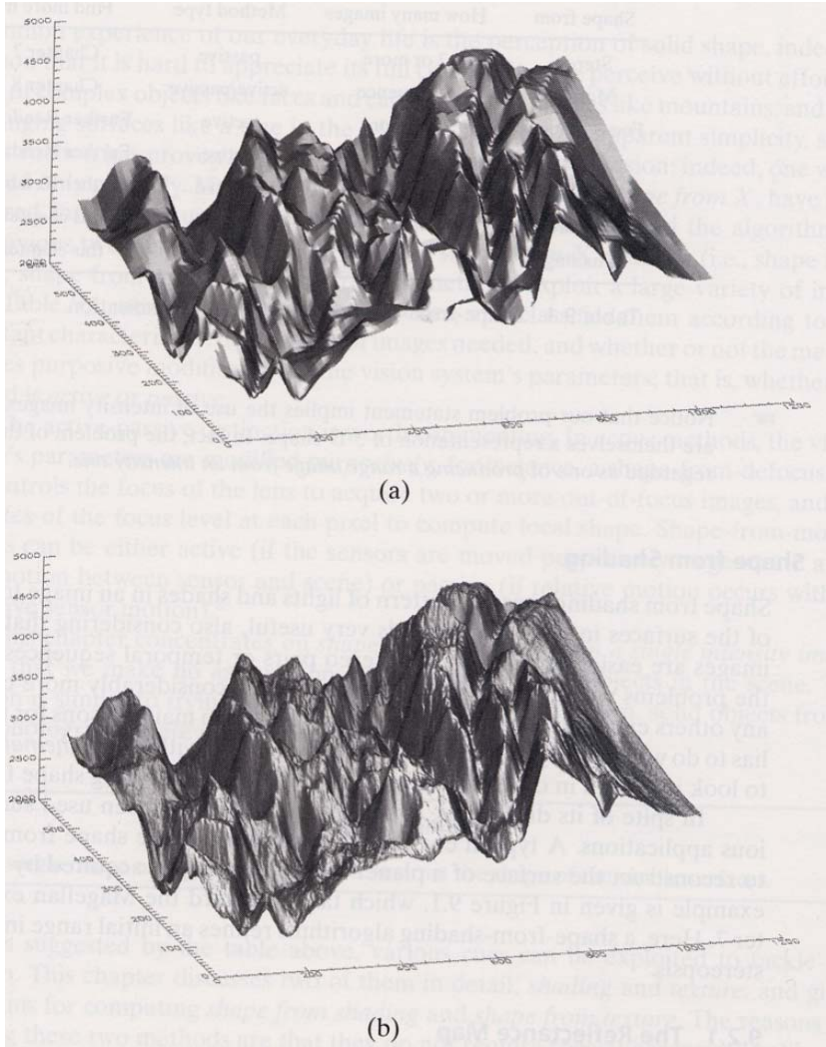
two images of the same Lambertian surface seen from above but illuminated from different directions



3D rendering of the surface

Shape from shading - Example





Reflectance of a Lambertian surface

- The radiance at a 3D point is proportional to the cosine of the angle between the surface normal and the direction of the illuminant

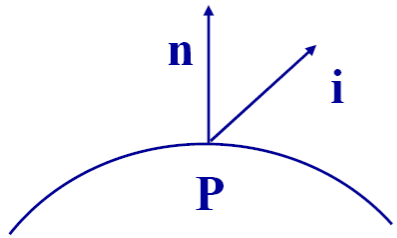
$$L(\mathbf{P}) = R_{\rho,\mathbf{i}} = \rho\ \mathbf{i}\bullet\mathbf{n}$$

- ρ is called the **effective albedo(反照率）**: the real albedo times the intensity of illuminant

Illuminance 照度

Luminance 亮度

Intensity 强度，光强



Fundamental equation for shape from shading

- The pixel intensity at $p=[x, y]^T$ is

$$E(\mathbf{P}) = L(\mathbf{P})\left(\frac{d}{f}\right)^2 \cos^4 \alpha$$

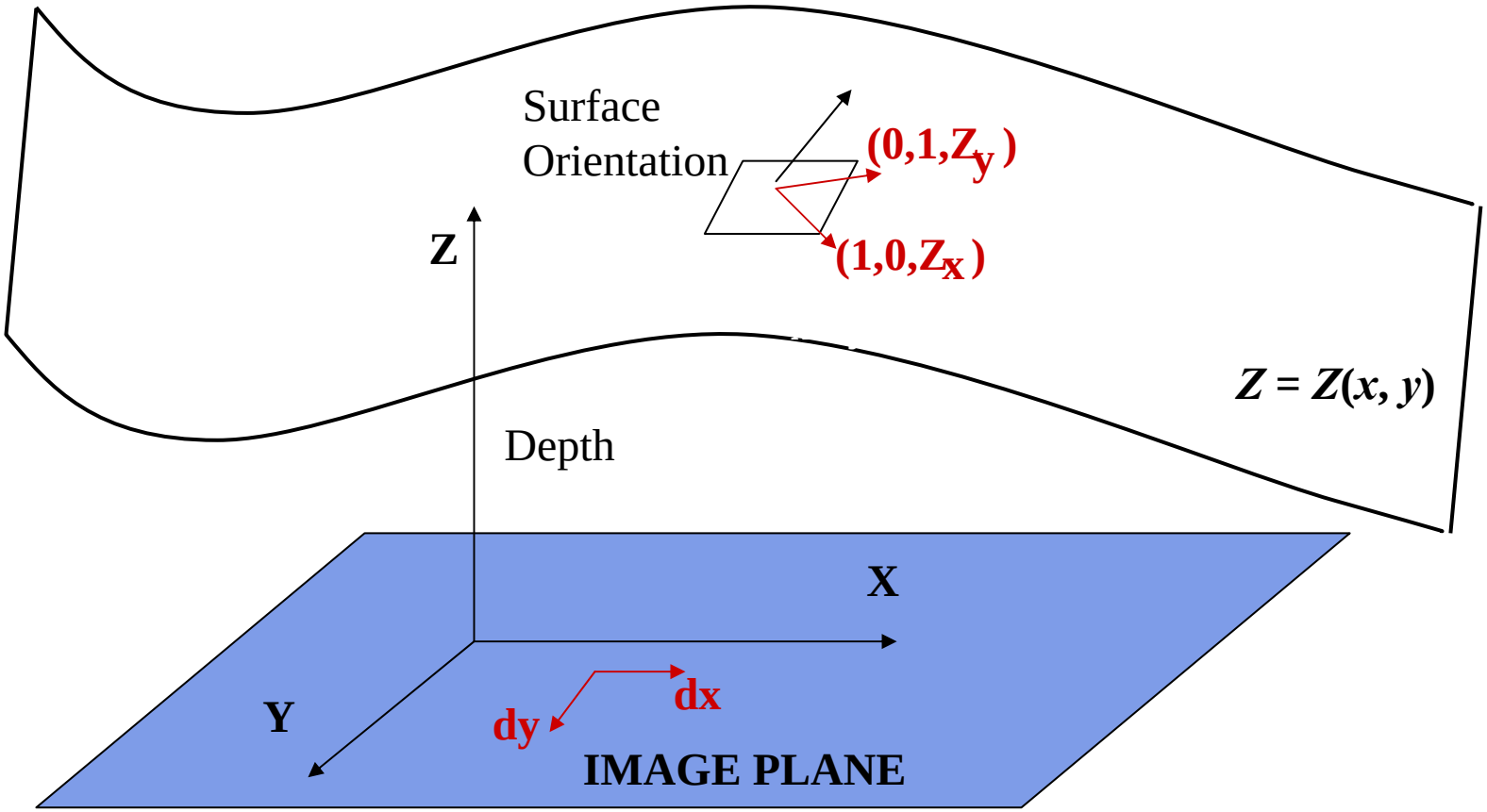
- Assumptions:
  - If we neglect the constant term and if we assume the optical system has been calibrated to compensate the $cos^4\ \alpha$ effect, and in addition,
  - if we assume that all the visible points of the surface receive direct illumination, we have the fundamental equation for shape from shading

$$E(\mathbf{P}) = R_{\rho,\mathbf{i}}(\mathbf{n})$$

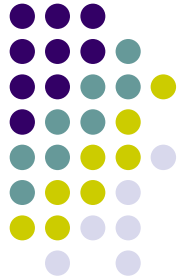
- Notice that the image intensity is determined only by the surface normal vector

Reflectance map is a viewer-centered representation of reflectance

$$\mathbf{n} = [1, 0, p]^T \times [0, 1, q]^T = \frac{1}{\sqrt{1 + p^2 + q^2}} [-p, -q, 1]^T$$



Representing normal vectors



- Assume that
  - the scene is far away from the camera, we can use a weak-perspective camera model to describe the projection.
  - the average depth of the scene is Z_0.
- The weak-perspective projection can be written as

$$x = f\frac{X}{Z_0} \qquad y = f\frac{Y}{Z_0}$$

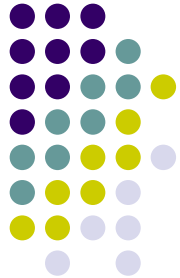
- Therefore, with some rescaling, the scene surface can be thought of as a function of the (x, y), or

$$Z = Z(X,Y) = Z(x\frac{Z_0}{f}, y\frac{Z_0}{f}) = Z(x,y)$$

- Then the slopes along the x axis and y axis are

$$\left[1,0,\frac{\partial Z}{\partial X}\right]^T = \left[1,0,\frac{\partial Z}{\partial x}\frac{\partial x}{\partial X}\right]^T = \left[1,0,\frac{\partial Z}{\partial x}\frac{f}{Z_0}\right]^T \text{ and } \left[0,1,\frac{\partial Z}{\partial y}\frac{f}{Z_0}\right]^T$$

Representing normal vectors



- We denote the above vectors as $[1, 0, p]^T$ and $[0, 1, q]^T$. The normal vector is the cross product of these two vectors, therefore

$$\mathbf{n} = [1, 0, p]^T \times [0, 1, q]^T = \frac{1}{\sqrt{1 + p^2 + q^2}} [-p, -q, 1]^T$$

- Now we can rewrite the fundamental equation as

$$E(x, y) = R_{\rho, \mathbf{i}}(p, q) = \frac{\rho}{\sqrt{1 + p^2 + q^2}} \mathbf{i}^T [-p, -q, 1]^T$$

The problem of shape from shading

- Assumptions
  - The imaging system is calibrated so that the $cos^4 \alpha$ effect is compensated
  - All the visible surface receive direct illumination
  - The surface is imaged under weak-perspective projection
  - The surface can be parameterized as $Z=Z[x, y]$
- Shape from shading
  - Given the reflectance map of the surface $R_{\rho,\mathbf{i}}(p,q)$, and full knowledge of the parameters ρ and i relative to the available image, reconstruct the surface slopes, p and q, for which

$$E(x,y) = R_{\rho,\mathbf{i}}(p,q)$$

  - and the surface $Z=Z[x, y]$ s.t.

$$p = \frac{\partial Z}{\partial x}\frac{Z_0}{f} \text{ and } q = \frac{\partial Z}{\partial y}\frac{Z_0}{f}$$

- To simplify the formulation, we will reconstruct a scaled version of the original surface so that

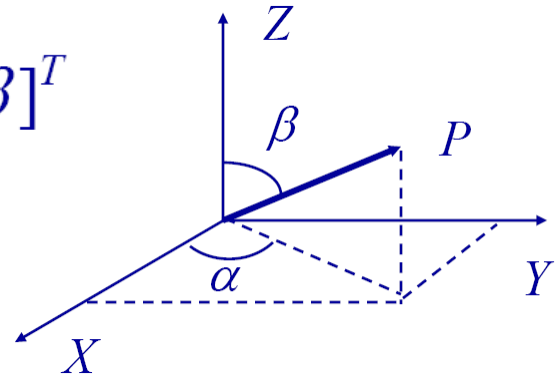
$$p = \frac{\partial Z}{\partial x} \text{ and } q = \frac{\partial Z}{\partial y}$$

Finding albedo and illuminant direction

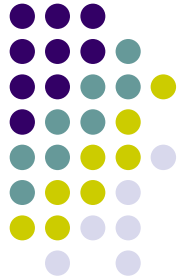
- Assumption
  - Albedo is the same for all surface points
  - The surface is Lambertian
  - The direction of the surface normal vectors are uniformly distributed in the 3D space
- A normal vector is represented by two angles, α and β, where $\alpha \in [0, 2\pi]$ and $\beta \in [0, \pi/2]$

$$\mathbf{n} = [\cos\alpha \sin\beta, \sin\alpha \sin\beta, \cos\beta]^T$$

As seen from the image plane, the probability of a normal vector with $\frac{\cos\beta}{2\pi}$



Finding albedo and illuminant direction



- If we also represent the illuminant direction using two angles $\tau \in [0, 2\pi]$ and $\sigma \in [0, \pi/2]$. As the result, the illuminant vector is

$$\mathbf{i} = [\cos\tau \sin\sigma, \sin\tau \sin\sigma, \cos\sigma]^T$$

- The image brightness of a surface point with normal angles α and β is

$$E(\alpha,\beta) = \rho \mathbf{i}^T \mathbf{n} = \rho(\cos\alpha \sin\beta \cos\tau \sin\sigma + \sin\alpha \sin\beta \sin\tau \sin\sigma + \cos\beta \cos\sigma)$$

- The average image intensity becomes

$$\langle E \rangle = \int_0^{2\pi}\int_0^{2\pi} E(\alpha,\beta) p(\alpha,\beta) d\alpha d\beta$$

$$= \rho \int_0^{\pi/2}\int_0^{2\pi} \{\cos\alpha \sin\beta \cos\tau \sin\sigma + \sin\alpha \sin\beta \sin\tau \sin\sigma + \cos\beta \cos\sigma\} \frac{\cos\beta}{2\pi} d\alpha d\beta$$

$$= \rho \int_0^{\pi/2}\int_0^{2\pi} \{\cos\beta \cos\sigma\} \frac{\cos\beta}{2\pi} d\alpha d\beta = \rho \cos\sigma \int_0^{\pi/2} \cos^2\beta d\beta = \frac{\pi}{4} \rho \cos\sigma$$

Finding albedo and illuminant direction

- Similar derivation gives us

$$\left\langle E^2 \right\rangle = \int_0^{2\pi}\int_0^{2\pi} E^2(\alpha,\beta)\,p(\alpha,\beta)\,d\alpha\,d\beta = \frac{1}{6}\rho^2(1+3\cos^2\sigma)$$

- From above two equations, we can recover the albedo and the angle σ

$$\gamma = \sqrt{6\pi^2 < E^2 > -48 < E >^2} = \sqrt{\pi^2\rho^2(1+3\cos^2\sigma) - 3\pi^2\rho^2\cos^2\sigma} = \pi\rho$$

- Therefore

$$\rho = \frac{\gamma}{\pi}$$

$$\cos\sigma = \frac{4 < E >}{\gamma}$$

Finding albedo and illuminant direction

- To estimate the angle τ, we compute the average image spatial gradient and estimate

$$\tan\tau = \frac{<\hat{E}_y>}{<\hat{E}_x>}$$

- Algorithm_Approximate_Albedo_Illuminant
  - Compute the average of the image intensity $<E>$ and of its square, $<E^2>$
  - Compute the average spatial image gradient $<E_x, E_y>$
  - Estimate the albedo and the illuminant angle as

$$\rho = \frac{\gamma}{\pi} \qquad \cos\sigma = \frac{4<E>}{\gamma} \qquad \tan\tau = \frac{<\hat{E}_y>}{<\hat{E}_x>}$$

where

$$\gamma = \sqrt{6\pi^2 <E^2> - 48 <E>^2}$$

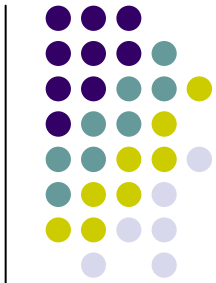
Shape from shading

- The fundamental equation of shape from shading is

$$E(x,y) = R_{\rho,\mathbf{i}}(p,q) = \frac{\rho}{\sqrt{1+p^2+q^2}}\mathbf{i}^T\left[-p,-q,1\right]^T$$

- The problem of shape from shading
  - Given an image $E(x,y)$,
  - and full knowledge of the parameters ρ and $\mathbf{i}$ relative to the available image,
  - reconstruct the surface slopes p and q, and the surface $Z(x,y)$

Shape from shading



- For each pixel,
  - there are two unknown variables and one known quantity.
- Shape from shading is a highly under-constrained problem

- Solution
  - To make the problem tractable, we add a constraint that a “smoother” surface is preferred. This soft constraint can be translated into a regularization term in an optimization problem
  - Variational method for shape from shading: we will find the smoothest solution that satisfies the fundamental equation. In other words, the solution should minimize the following energy function

$$\varepsilon = \int \{(E(x,y) - R(p,q))^2 + \lambda(p_x^2 + p_y^2 + q_x^2 + q_y^2)\} dxdy$$

Euler-Lagrange equation

- If J is defined as

$$J = \int f(t, q, \dot{q}) dt$$

- where $\dot{q} = \mathrm{d}q / \mathrm{d}t$

The derivative of J vanishes if the Euler-Lagrange equation

$$\frac{\partial f}{\partial q} - \frac{d}{dt}\left(\frac{\partial f}{\partial \dot{q}}\right) = 0$$

is satisfied. If the time-derivative $\dot{q}$ is replaced by space-derivative q_x, the equation becomes

$$\frac{\partial f}{\partial q} - \frac{d}{dx}\left(\frac{\partial f}{\partial q_x}\right) = 0$$

Euler-Lagrange equation

- In our case, the equation is

$$\varepsilon = \int \{(E(x,y) - R(p,q))^2 + \lambda(p_x^2 + p_y^2 + q_x^2 + q_y^2)\} dxdy$$

- Using the Euler-Lagrange equation with two independent variables, the stationary value can be achieved when

$$\frac{\partial f}{\partial p} - \frac{d}{dx}\left(\frac{\partial f}{\partial p_x}\right) - \frac{d}{dy}\left(\frac{\partial f}{\partial p_y}\right) = 0 \quad \text{and} \quad \frac{\partial f}{\partial q} - \frac{d}{dx}\left(\frac{\partial f}{\partial q_x}\right) - \frac{d}{dy}\left(\frac{\partial f}{\partial q_y}\right) = 0$$

where

$$f(x, y, p, q, p_x, p_y, q_x, q_y) = (E(x,y) - R(p,q))^2 + \lambda(p_x^2 + p_y^2 + q_x^2 + q_y^2)$$

Therefore,

$$-2(E-R)\frac{\partial R}{\partial p} - 2\lambda p_{xx} - 2\lambda p_{yy} = 0 \quad \text{and} \quad -2(E-R)\frac{\partial R}{\partial q} - 2\lambda q_{xx} - 2\lambda q_{yy} = 0$$

The discrete form

$$-2(E-R)\frac{\partial R}{\partial p}-2\lambda p_{xx}-2\lambda p_{yy}=0 \quad \text{and} \quad -2(E-R)\frac{\partial R}{\partial q}-2\lambda q_{xx}-2\lambda q_{yy}=0$$

- can be simplified as

$$p_{xx}+p_{yy}=-\frac{1}{\lambda}(E-R)\frac{\partial R}{\partial p} \quad \text{and} \quad q_{xx}+q_{yy}=-\frac{1}{\lambda}(E-R)\frac{\partial R}{\partial q}$$

- The discrete version of these equations are

$$-4p_{i,j}+p_{i,j+1}+p_{i,j-1}+p_{i+1,j}+p_{i-1,j}=-\frac{1}{\lambda}(E(i,j)-R(p_{i,j},q_{i,j}))\left.\frac{\partial R}{\partial p}\right|_{p_{i,j},q_{i,j}}$$

$$-4q_{i,j}+q_{i,j+1}+q_{i,j-1}+q_{i+1,j}+q_{i-1,j}=-\frac{1}{\lambda}(E(i,j)-R(p_{i,j},q_{i,j}))\left.\frac{\partial R}{\partial q}\right|_{p_{i,j},q_{i,j}}$$

An iterative algorithm

- The previous discrete equations can be rewritten as

$$p_{i,j} = \overline{p}_{i,j} + \frac{1}{4\lambda}(E(i,j) - R(p_{i,j}, q_{i,j}))\left.\frac{\partial R}{\partial p}\right|_{p_{i,j}, q_{i,j}}$$

$$q_{i,j} = \overline{q}_{i,j} + \frac{1}{4\lambda}(E(i,j) - R(p_{i,j}, q_{i,j}))\left.\frac{\partial R}{\partial q}\right|_{p_{i,j}, q_{i,j}}$$

where

$$\overline{p}_{i,j} = \frac{p_{i,j+1} + p_{i,j-1} + p_{i+1,j} + p_{i-1,j}}{4}$$

$$\overline{q}_{i,j} = \frac{q_{i,j+1} + q_{i,j-1} + q_{i+1,j} + q_{i-1,j}}{4}$$

An iterative algorithm

- The previous discrete equations can be rewritten as

$$p_{i,j} = \overline{p}_{i,j} + \frac{1}{4\lambda}(E(i,j) - R(p_{i,j}, q_{i,j}))\left.\frac{\partial R}{\partial p}\right|_{p_{i,j}, q_{i,j}}$$

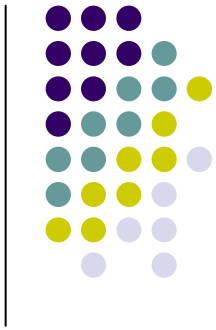
$$q_{i,j} = \overline{q}_{i,j} + \frac{1}{4\lambda}(E(i,j) - R(p_{i,j}, q_{i,j}))\left.\frac{\partial R}{\partial q}\right|_{p_{i,j}, q_{i,j}}$$

- p_{ij} and q_{ij} can be solved by starting from some initial solution at step 0, and advancing from step k to step k+1 using the updating rule

$$p_{i,j}^{k+1} = \overline{p}_{i,j}^{k} + \frac{1}{4\lambda}(E(i,j) - R(p_{i,j}, q_{i,j}))\left.\frac{\partial R}{\partial p}\right|_{p_{i,j}, q_{i,j}}^{k}$$

$$q_{i,j}^{k+1} = \overline{q}_{i,j}^{k} + \frac{1}{4\lambda}(E(i,j) - R(p_{i,j}, q_{i,j}))\left.\frac{\partial R}{\partial q}\right|_{p_{i,j}, q_{i,j}}^{k} \qquad \textbf{(1)}$$

Enforcing integrability



- p_{ij} and q_{ij} are solved independently, there may not exist surface Z so that $Z_x=p$, $Z_y=q$

  Example: Can you find $Z_{i,j}$ so that $p_{i,j}=0$ and $q_{i,j}=(j-100)$?

- Solution:
  - After each iteration, we find a revised solution p' and q', so that they are integrable and are closest to p and q
  - Suppose IFFT of p and q are

$$p=\sum c_p(\omega_x,\omega_y)e^{i(\omega_x x+\omega_y y)}$$

$$q=\sum c_q(\omega_x,\omega_y)e^{i(\omega_x x+\omega_y y)}$$

Enforcing integrability

- Solution:

$$p = \sum c_p(\omega_x, \omega_y) e^{i(\omega_x x + \omega_y y)}$$

$$q = \sum c_q(\omega_x, \omega_y) e^{i(\omega_x x + \omega_y y)}$$

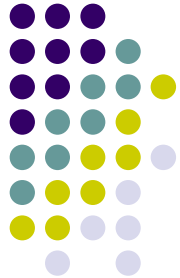
We can construct the surface function as

$$Z = \sum c(\omega_x, \omega_y) e^{i(\omega_x x + \omega_y y)} \quad \textbf{(2)}$$

where

$$c(\omega_x, \omega_y) = \frac{-i\omega_x c_p(\omega_x, \omega_y) - i\omega_y c_q(\omega_x, \omega_y)}{\omega_x^2 + \omega_y^2}$$

Enforcing integrability



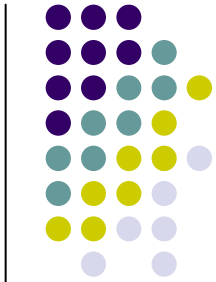
- Then the integrable solution that are closest p and q are

$$p' = \frac{\partial Z}{\partial x} = \sum (i\omega_x c(\omega_x, \omega_y)) e^{i(\omega_x x + \omega_y y)}$$

$$q' = \frac{\partial Z}{\partial y} = \sum (i\omega_y c(\omega_x, \omega_y)) e^{i(\omega_x x + \omega_y y)}$$

- It can be verified by starting from p’ and q’ and using the above equation to obtain the same p’ and q’

Shape from shading - algorithm

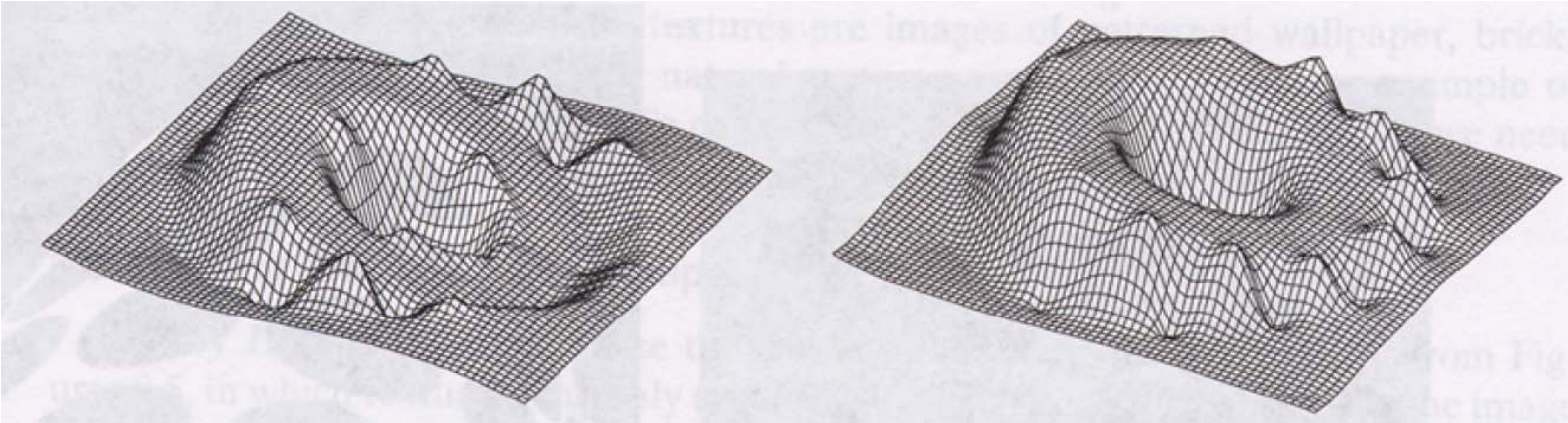


- Given the effective albedo, the illuminant direction, and the image, initialize the surface slopes p and q to 0.

- Until a suitable criterion is met, iterate the following two steps
  - Update *p* and *q* using (1)
  - Compute the FFT of the updated *p* and *q*, estimate Z and p' and q' using (2)

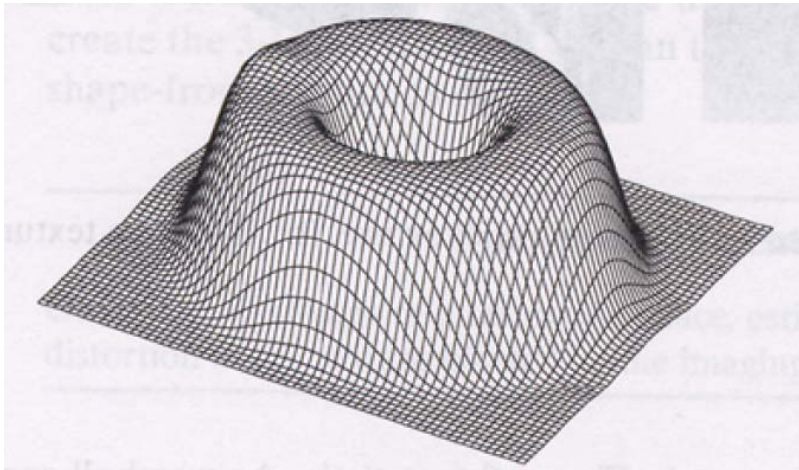
- Output Z, p, q

Experimental results



After 100 iterations

After 1000 iterations

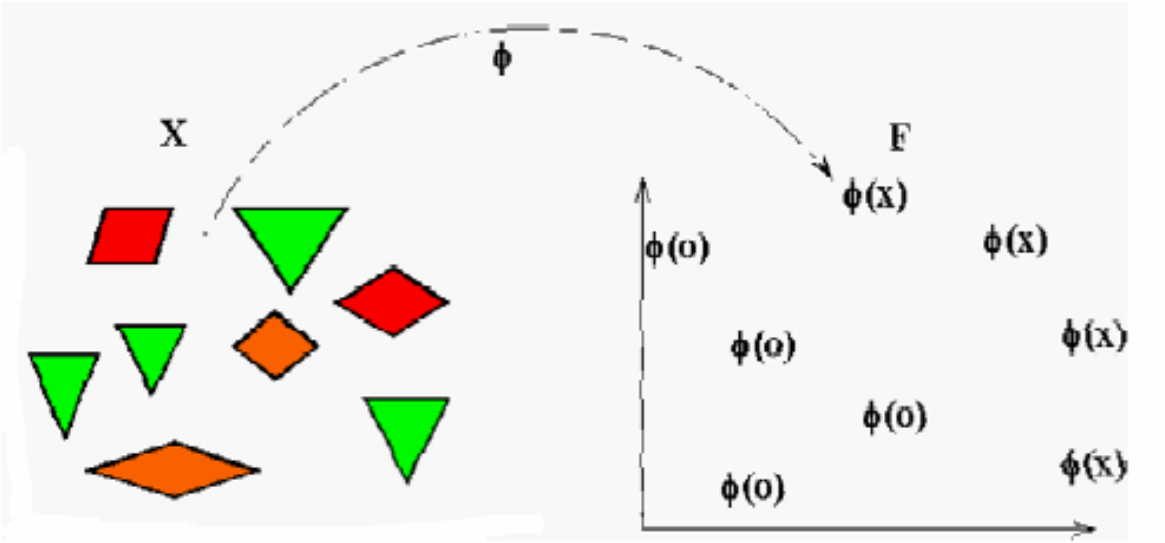


After 2000 iterations

主要内容

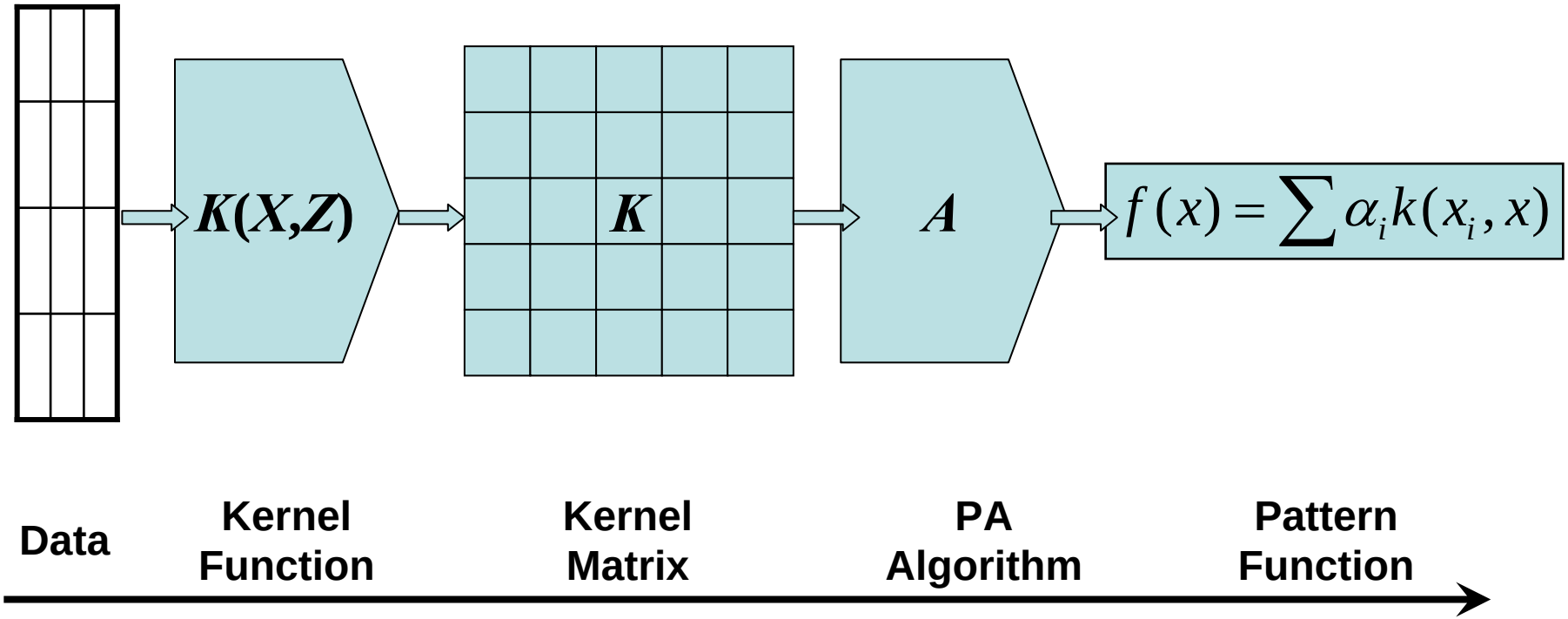
- Shape from shading
- Kernel methods

main idea $\phi : \boldsymbol{x} \mapsto \phi(\boldsymbol{x})$

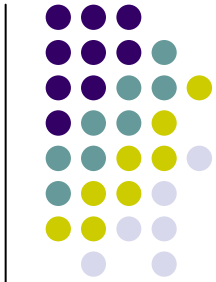


The stages involved in the application of kernel method

Kernel ? Nonlinear ? SVM ?



Linear regression in a feature space



- Primal linear regression:
  - Find a homogeneous real-valued linear function

$$g(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle = \boldsymbol{w}' \boldsymbol{x} = \sum_{i=1}^{n} w_i x_i$$

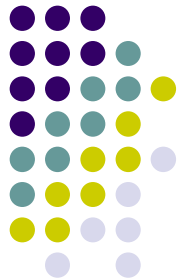
  - That best interpolate a given training set

$$S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots (\boldsymbol{x}_l, y_l)\}$$

  - Or create a pattern function that should be approximately equal to zero

$$f(\boldsymbol{x}, y) = |y - g(\boldsymbol{x})| = |y - \langle \boldsymbol{w}, \boldsymbol{x} \rangle| \approx 0$$

Primal linear regression



- We would like to find a function for which all of these training errors are small

$$L(g,S) = L(\boldsymbol{w},S) = \sum_{i=1}^{l} (y - g(\boldsymbol{x}_i))^2 = \sum_{i=1}^{l} L(g,(\boldsymbol{x}_i, y_i))^2$$

Least squares approximation

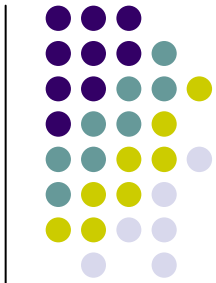
- Hence the lose function can be written as

$$L(g,S) = \|\boldsymbol{\xi}\|_2^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

- We have

$$\frac{\partial L(\boldsymbol{w},S)}{\partial \boldsymbol{w}} = -2\boldsymbol{X}'\boldsymbol{y} + 2\boldsymbol{X}'\boldsymbol{X}\boldsymbol{w}$$

Primal linear regression



- Normal equations

$$X'Xw = X'y \quad \Rightarrow \quad w = (X'X)^{-1}X'y$$

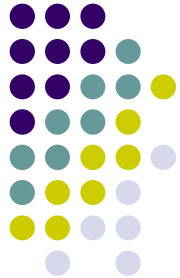
  - The cost is $O(n)^3$

- Dual representation

$$w = (X'X)^{-1}X'y = X'X(X'X)^{-2}X'y = X'\alpha$$

$$g(x) = \langle w, x \rangle = w'x = \alpha'Xx = \sum_{i=1}^{l} \alpha_i \langle x_i, x \rangle$$

Ridge regression



- Ridge regression corresponds to solving the following optimization:

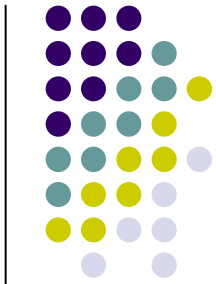
$$\min_{\boldsymbol{w}} L_\lambda(\boldsymbol{w}, S) = \min_{\boldsymbol{w}} \lambda \left\|\boldsymbol{w}\right\|^2 + \sum_{i=1}^{l} (y_i - g(\boldsymbol{x}_i))^2$$

- Primal:

$$\boldsymbol{X}'\boldsymbol{X}\boldsymbol{w} + \lambda \boldsymbol{w} = (\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_n)\boldsymbol{w} = \boldsymbol{X}'\boldsymbol{y}$$

$$\Rightarrow \quad \boldsymbol{w} = (\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{X}'\boldsymbol{y}$$

Ridge regression



- The resulting prediction function is

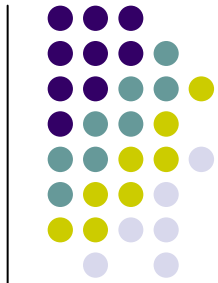
$$g(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle = \boldsymbol{y}\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{x}$$

- Dual: $\boldsymbol{w} = \lambda^{-1}\boldsymbol{X}'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) = \boldsymbol{X}'\boldsymbol{\alpha}$

$$\begin{aligned} \boldsymbol{\alpha} &= \lambda^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) \qquad \boldsymbol{G} = \boldsymbol{X}\boldsymbol{X}' \quad \text{or} \quad \boldsymbol{G}_{ij} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\ &\Rightarrow \quad \lambda\boldsymbol{\alpha} = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{X}'\boldsymbol{\alpha}) \\ &\Rightarrow \quad (\boldsymbol{X}\boldsymbol{X}' + \lambda \boldsymbol{I}_l)\boldsymbol{\alpha} = \boldsymbol{y} \\ &\Rightarrow \quad \boldsymbol{\alpha} = (\boldsymbol{G} + \lambda \boldsymbol{I}_l)^{-1}\boldsymbol{y} \end{aligned}$$

$$g(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle = \left\langle \sum_{i=1}^{l} \alpha_i \boldsymbol{x}_i, \boldsymbol{x} \right\rangle = \sum_{i=1}^{l} \alpha_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle = \boldsymbol{y}'(\boldsymbol{G} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{k}$$

Ridge regression



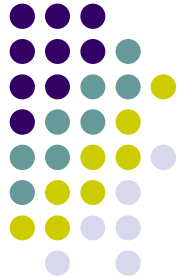
- The dual solution:

$$g(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle = \left\langle \sum_{i=1}^{l} \alpha_i \boldsymbol{x}_i, \boldsymbol{x} \right\rangle = \sum_{i=1}^{l} \alpha_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle = \boldsymbol{y}'(\boldsymbol{G} + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{k}$$

$$k_i = \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle$$

- The ridge regression can be solved in a form that only requires inner products between points

Nonlinear feature mappings



- An embedding map

$$\phi : \boldsymbol{x} \in \mathbb{R}^n \mapsto \phi(\boldsymbol{x}) \in F \subseteq \mathbb{R}^N \qquad N \gg n$$

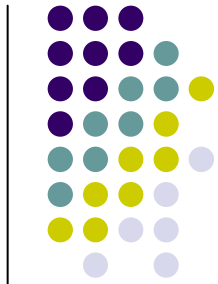
$$S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots (\boldsymbol{x}_l, y_l)\}$$

$$\hat{S} = \{(\phi(\boldsymbol{x}_1), y_1), (\phi(\boldsymbol{x}_2), y_2), \cdots (\phi(\boldsymbol{x}_l), y_l)\}$$

$$f(\boldsymbol{x}, y) = \left| y - g(\boldsymbol{x}) \right| = \left| y - \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle \right| \approx 0$$

$$\boldsymbol{G} = \boldsymbol{X}\boldsymbol{X}' \quad \text{or} \quad \boldsymbol{G}_{ij} = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle \qquad k_i = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle$$

Kernel function

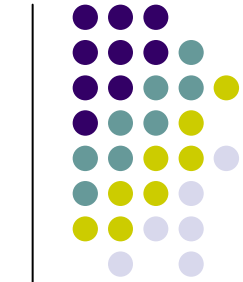


- A *kernel* is a function k that for all $\boldsymbol{x}$, $\boldsymbol{z}$ satisfies

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = \left\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \right\rangle$$

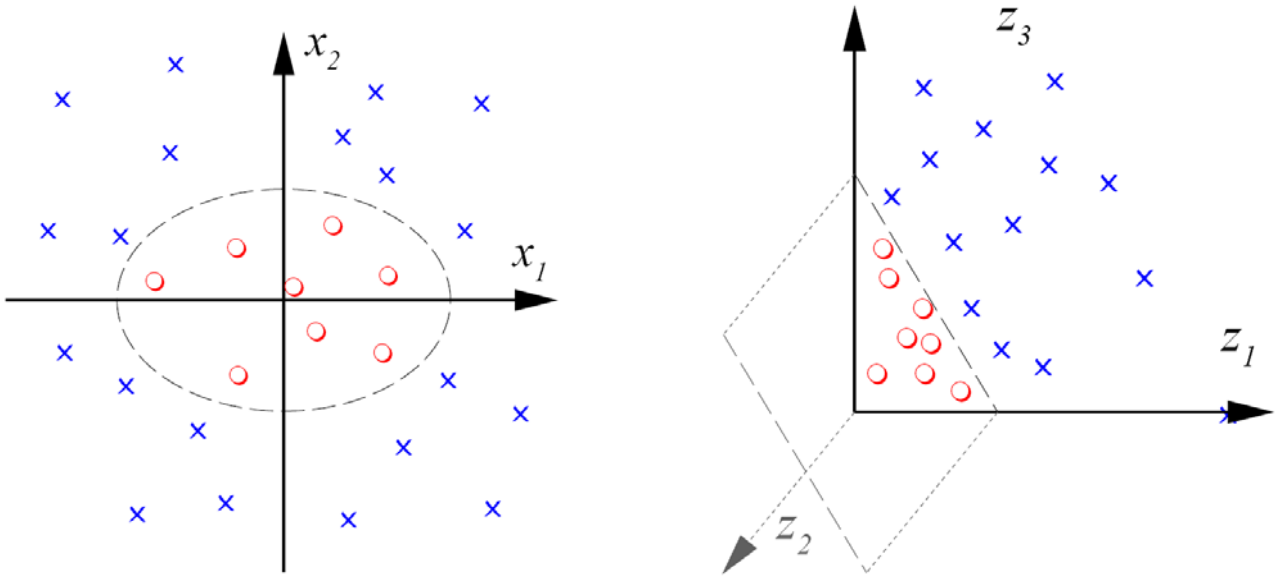
- where ϕ is a mapping from X to an (inner product) feature space F

$$\phi : \boldsymbol{x} \mapsto \phi(\boldsymbol{x}) \in F$$

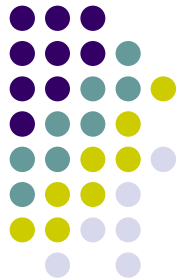


Example: All Degree 2 Monomials

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$



Kernel Example



- The feature map

$$\phi : \boldsymbol{x} = (x_1, x_2) \mapsto \phi(\boldsymbol{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in F = \mathbb{R}^3$$

- The hypothesis space of linear functions in *F*

$$g(\boldsymbol{x}) = w_{11}x_1^2 + w_{22}x_2^2 + w_{12}\sqrt{2}x_1x_2$$

- The inner product

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle = x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1x_2z_1z_2$$

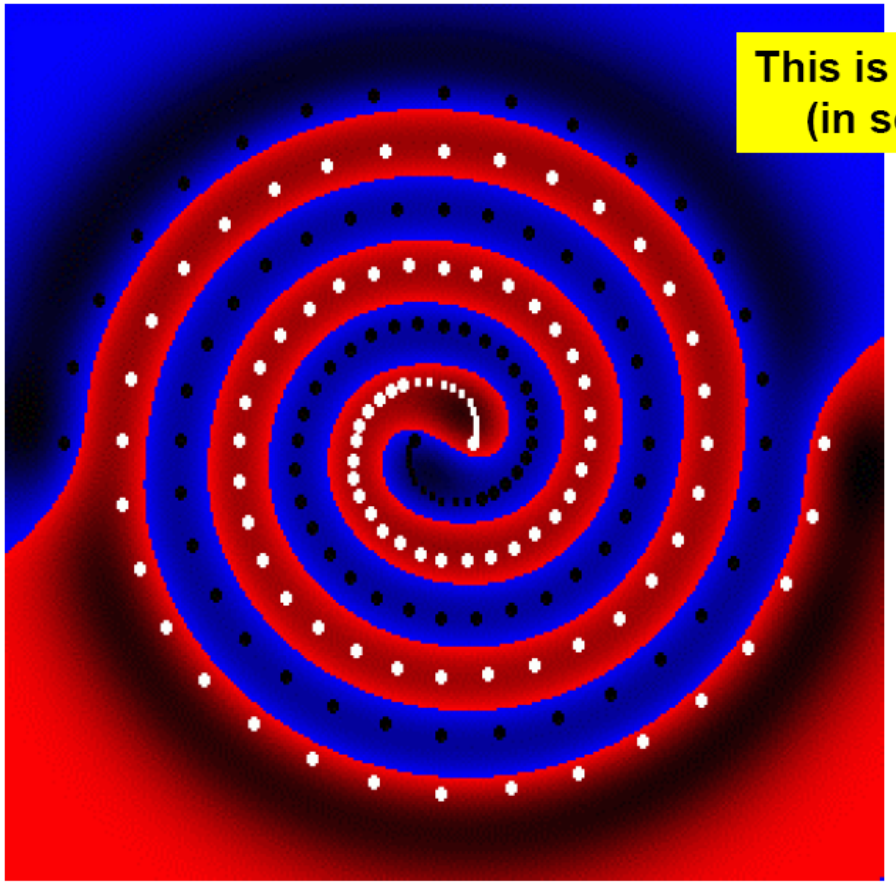
$$= \left(x_1 z_1 + x_2 z_2 \right)^2 = \langle \boldsymbol{x}, \boldsymbol{z} \rangle^2$$

The kernel trick

- More generally: $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^n, d \in \mathbb{N}$

$$\langle \boldsymbol{x}, \boldsymbol{z} \rangle^d = \left(\sum_{j=1} x_i z_i \right)^d$$

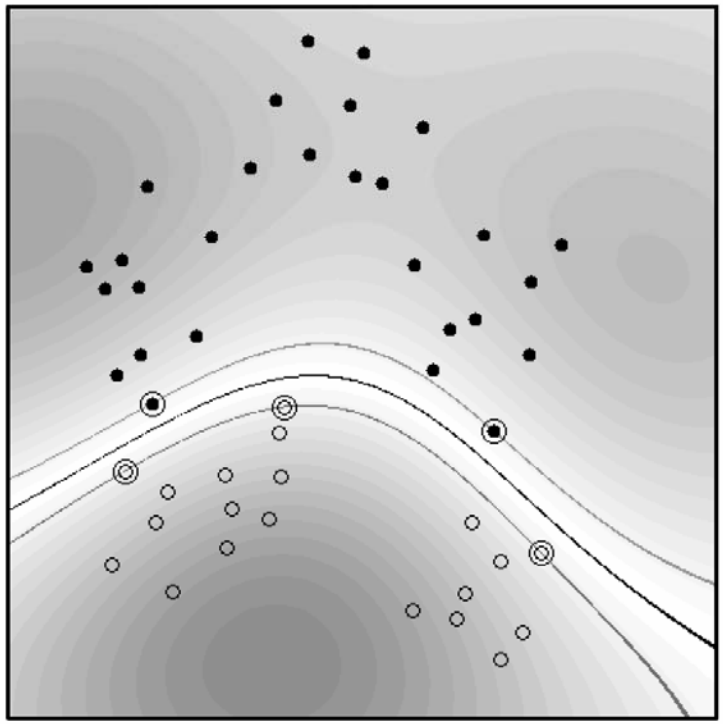
$$= \sum_{j_1, j_2, \cdots j_d = 1}^{l} x_{j_1} x_{j_2} \cdots x_{j_d} z_{j_1} z_{j_2} \cdots z_{j_d} = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle$$

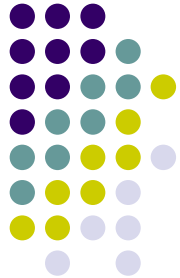


This is a hyperplane!
(in some space)

Toy Example with Gaussian Kernel

$$k(x, x') = \exp\left(-\|x - x'\|^2\right)$$





Mercer's Theorem

If k is a continuous kernel of a positive definite integral operator on $L_2(\mathcal{X})$ (where $\mathcal{X}$ is some compact space),

$$\int_{\mathcal{X}} k(x, x')f(x)f(x')\,dx\,dx' \geq 0,$$

it can be expanded as

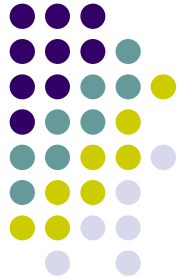
$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(x')$$

using eigenfunctions ψ_i and eigenvalues $\lambda_i \geq 0$ [34].
In that case

$$\Phi(x) := \begin{pmatrix} \sqrt{\lambda_1}\psi_1(x) \\ \sqrt{\lambda_2}\psi_2(x) \\ \vdots \end{pmatrix}$$

satisfies $\left\langle \Phi(x), \Phi(x') \right\rangle = k(x, x')$.

Characterization of kernels



- A function

$$\kappa : X \times X \rightarrow \mathbb{R}$$

which is either continuous or has a countable domain, cab be decomposed

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = \left\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \right\rangle$$

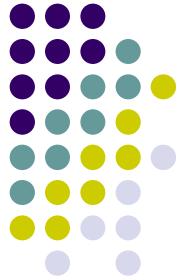
into a feature map into a Hilbert space F applied to both its arguments followed by the evaluation of the inner product in *F if and only if it satisfies the finite positive semi-definite properties*

Some Properties of Kernels

If $k_1, k_2, \ldots$ are pd kernels, then so are

- αk_1, provided $\alpha \geq 0$
- $k_1 + k_2$
- $k_1 \cdot k_2$
- $k(x, x') := \lim_{n \to \infty} k_n(x, x')$, provided it exists

Further operations to construct kernels from kernels: tensor products, direct sums, convolutions [23].



Kernel PCA

- Input: data $S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_l\}$, dimension k

- Process:

$$\boldsymbol{K}_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j), i, j = 1, \ldots l$$

Kernel matrix

$$\boldsymbol{K} \leftarrow \boldsymbol{K} - \tfrac{1}{l}\boldsymbol{jj'K} - \tfrac{1}{l}\boldsymbol{Kjj'} + \tfrac{1}{l^2}(\boldsymbol{jKj'})\boldsymbol{jj'}$$

normalization

$$[\boldsymbol{V}, \boldsymbol{\Lambda}] = eig(\boldsymbol{K})$$

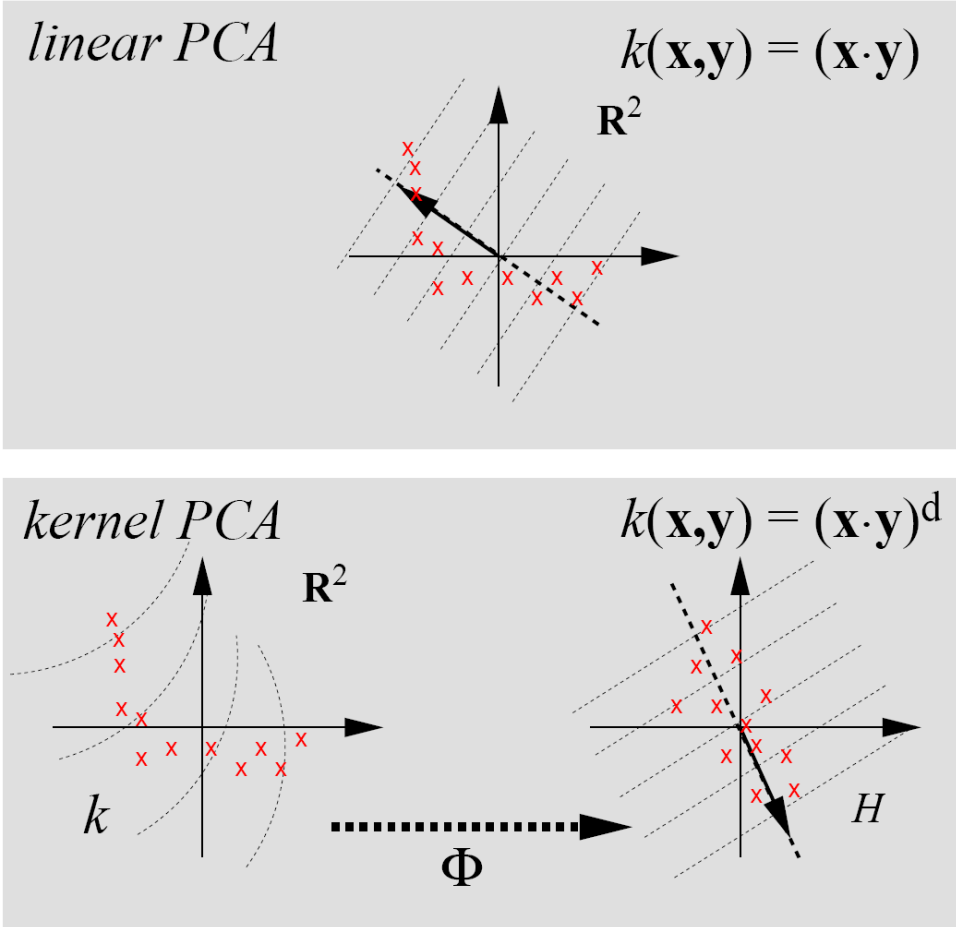
Eigen-analysis

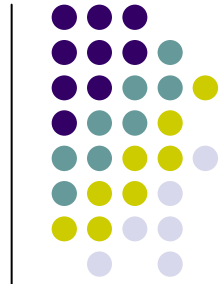
$$\alpha^j = \tfrac{1}{\sqrt{\lambda}}\boldsymbol{v}_j, j = 1, \ldots k$$

$$\tilde{\boldsymbol{x}}_i = \left(\sum_{i=1}^{l} \alpha_i^j \kappa(\boldsymbol{x}_i, \boldsymbol{x}) \right)_{j=1}^{k}$$

- Output: transformed data $\tilde{S} = \{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_l\}$

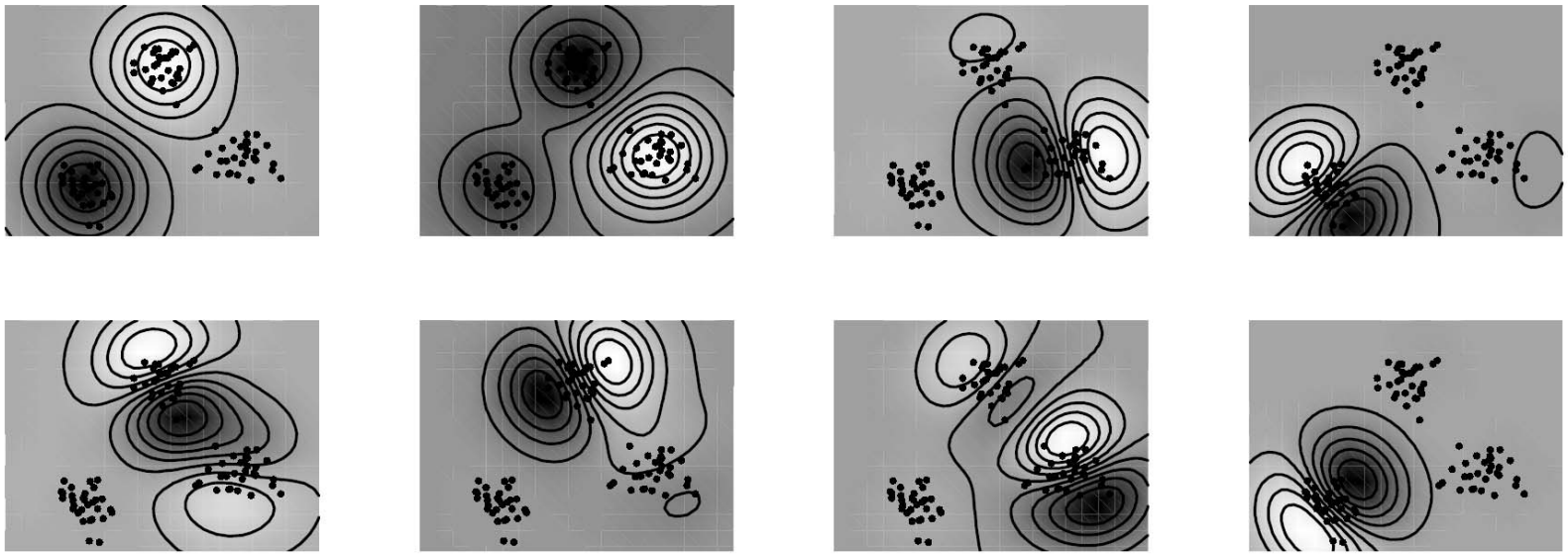
Kernel PCA



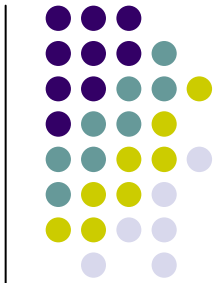


Toy Example with Gaussian Kernel

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2\right)$$



Dual principle component regression



- Input: data $S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_l\}$, dimension k and target output vectors $\boldsymbol{y}^s, s = 1, ..., m$

- Process:

$$\boldsymbol{K}_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j), i, j = 1, ...l$$

$$\boldsymbol{K} \leftarrow \boldsymbol{K} - \frac{1}{l}\boldsymbol{jj'K} - \frac{1}{l}\boldsymbol{Kjj'} + \frac{1}{l^2}(\boldsymbol{jKj'})\boldsymbol{jj'}$$

$$[\boldsymbol{V}, \boldsymbol{\Lambda}] = eig(\boldsymbol{K})$$

$$\alpha^s = \sum_{j=1}^{k} \frac{1}{\sqrt{\lambda}}(\boldsymbol{v}_j' \boldsymbol{y}^s)\boldsymbol{v}_j, s = 1, ...m$$

- Output: regression functions

$$f_s(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i^s \kappa(\boldsymbol{x}_i, \boldsymbol{x}), s = 1, ..., m$$