Energy Minimization and an Introduction to the Bayesian Framework Problem Set 3 Solutions

Computer Vision 2020
University of Bern

1 Bayesian Framework

1. **Maximum a Posteriori** Suppose that g is a Gaussian random variable such that $g \sim \mathcal{N}(\mu, \Sigma)$ and that we observe m iid samples $g_1, \ldots, g_m$ of g. Let us assume that the parameters μ and Σ are independent random variables with Gaussian distribution (the prior distribution) with an extremely large covariance, so that we can claim that $p(\mu) \simeq \text{const}$ and $p(\Sigma) \simeq \text{const}$. Compute the Maximum a Posteriori solution for μ and Σ.

   Solution Normal distribution:

   Single variable

$$p(x; \mu, \sigma) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

   Multivariable

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{k}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)} \tag{2}$$

   The posterior probability $p(\mu, \Sigma|g)$ can be written as $p(g|\mu, \Sigma)\frac{p(\mu,\Sigma)}{p(g)}$ thanks to Bayes rule. Since $p(\mu, \Sigma) = p(\mu)p(\Sigma)$ is approximately constant we can write

$$\hat{\mu}, \hat{\Sigma} = \arg\max_{\mu,\Sigma} p(\mu, \Sigma|g) \simeq \arg\max_{\mu,\Sigma} p(g|\mu, \Sigma). \tag{3}$$

   Because of the iid assumption we can then write $p(g_1, \ldots, g_m|\mu, \Sigma) = \prod_{i=1}^m p(g_i|\mu, \Sigma)$ and have

$$\hat{\mu}, \hat{\Sigma} = \arg\max_{\mu,\Sigma} \sum_{i=1}^m \log p(g_i|\mu, \Sigma) \tag{4}$$

$$= \arg\max_{\mu,\Sigma} \sum_{i=1}^m -\log\left((2\pi)^n |\Sigma|\right) - (g_i - \mu)^T \Sigma^{-1} (g_i - \mu). \tag{5}$$

   To find $\hat{\mu}$, we need to take the partial derivative of the above expression w.r.t $\hat{\mu}$ and set it to zero and then find $\hat{\mu}$.

$$\frac{\partial}{\partial \mu}\left(\sum_{i=1}^{m} -\log\left((2\pi)^n|\Sigma|\right) - (g_i-\mu)^T\Sigma^{-1}(g_i-\mu)\right) = \frac{\partial}{\partial \mu}\left(\sum_{i=1}^{m} -(g_i-\mu)^T\Sigma^{-1}(g_i-\mu)\right) \tag{6}$$

$$= \sum_{i=1}^{m} 2\Sigma^{-1}(g_i-\mu) = 0 \tag{7}$$

The above expression is zero when $\sum_{i=1}^{m}(g_i-\mu)^T = 0$. From here we find

$$\mu = \frac{1}{m}\sum_{i=1}^{m} g_i \tag{8}$$

Let us find Σ in the folowwing way.

$$\frac{\partial}{\partial \Sigma^{-1}}\left(\sum_{i=1}^{m} -\log\left((2\pi)^n|\Sigma|\right) - (g_i-\mu)^T\Sigma^{-1}(g_i-\mu)\right) = \frac{\partial}{\partial \Sigma^{-1}}\left(\sum_{i=1}^{m} -\log\left((2\pi)^n|\Sigma|\right)\right) - \frac{\partial}{\partial \Sigma^{-1}}\sum_{i=1}^{m}(g_i-\mu)^T\Sigma^{-1}(g_i-\mu) \tag{9}$$

Let us find the derivate of the first term:

$$\frac{\partial}{\partial \Sigma^{-1}}\left(\sum_{i=1}^{m} -\log\left((2\pi)^n|\Sigma|\right)\right) = \frac{\partial}{\partial \Sigma^{-1}}\left(\sum_{i=1}^{m} -\log\left((2\pi)^n\right) - log(|\Sigma|)\right) = \frac{\partial}{\partial \Sigma^{-1}}\left(-mlog(|\Sigma|)\right) = m\Sigma \tag{10}$$

Let us find the derivate of the second term:

$$\frac{\partial}{\partial \Sigma^{-1}}\sum_{i=1}^{m}(g_i-\mu)^T\Sigma^{-1}(g_i-\mu) = \sum_{i=1}^{m}(g_i-\mu)(g_i-\mu)^T \tag{11}$$

Let us gather the derivatives of the first and second term together:

$$m\Sigma^{-1} - \sum_{i=1}^{m}(g_i-\mu)(g_i-\mu)^T = 0 \tag{12}$$

From the above expression we find:

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(g_i-\mu)(g_i-\mu)^T \tag{13}$$

Useful Identities:

$$|\Sigma| = \frac{1}{|\Sigma^{-1}|} \tag{14}$$

$$\frac{\partial|\Sigma|}{\partial\Sigma} = |\Sigma|(\Sigma^{-1})^T \tag{15}$$

$$\frac{\partial x^T A x}{\partial A} = xx^T \tag{16}$$

2. **Conditional mean (challenging)** Suppose that g is a Gaussian random variable such that $g \sim \mathcal{N}(\mu, \Sigma)$ and that we observe m iid samples $g_1, \ldots, g_m$ of g. Assume that Σ is a fixed parameter and that μ is a random variable with a Gaussian distribution (the prior distribution) with an extremely large covariance, so that we can claim that $p(\mu) \simeq \text{const}$. Compute the Conditional Mean estimate of the parameters μ.

Solution The conditional mean estimate is

$$\hat{\mu} = \int \mu p(\mu|g) d\mu = \int \mu p(g|\mu) \frac{p(\mu)}{p(g)} d\mu \tag{17}$$

By using the same assumptions as in the previous exercise, we have

$$\hat{\mu} = Z \int \mu \sum_{i=1}^{m} \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(g_i-\mu)^T \Sigma^{-1}(g_i-\mu)} d\mu \tag{18}$$

$$= Z \sum_{i=1}^{m} g_i = \frac{1}{m} \sum_{i=1}^{m} g_i \tag{19}$$

where $Z = \frac{p(\mu)}{p(g)} = \frac{1}{m}$ is a constant (use marginalization to compute $p(g) = \int p(g|\mu)p(\mu)d\mu$).

2 Optimization and Regularization

1. **Energy minimization**

   Show that the solution to the following energy minimization problem

$$\arg\min_u |Au - f|^2 \tag{20}$$

   where $A \in \mathbf{R}^{n \times n}$ and $u, f \in \mathbf{R}^n$ is

$$u = \left(A^\top A\right)^{-1} A^\top f. \tag{21}$$

   Solution To find the solution we need to compute the gradient and set it to zero

$$E = (Au - f)^T (Au - f) \tag{22}$$

$$= (u^T A^T - f^T)(Au - f) \tag{23}$$

$$= u^T A^T Au - u^T A^T f - f^T Au - f^T f \tag{24}$$

$$= u^T A^T Au - 2u^T A^T f - f^T f \tag{25}$$

$$\nabla E = 2A^T Au - 2A^T f = 2A^\top (Au - f) = 0. \tag{26}$$

   By rearranging the equation we obtain

$$A^\top Au - A^\top f = 0 \tag{27}$$

$$A^\top Au = A^\top f \tag{28}$$

$$u = \left(A^\top A\right)^{-1} A^\top f. \tag{29}$$

2. **Energy minimization with regularization**

   Show that the solution to the following energy minimization problem

$$\arg\min_u |Au - f|^2 + \lambda |u|^2 \tag{30}$$

   where $A \in \mathbf{R}^{n \times n}$, $u, f \in \mathbf{R}^n$ is

$$u = \left(A^\top A + \lambda I\right)^{-1} A^\top f. \tag{31}$$

Solution To find the solution we need to compute the gradient and set it to zero

$$\nabla E = 2A^\top (Au - f) + 2\lambda u = 0. \tag{32}$$

By rearranging the equation we obtain

$$A^\top Au - A^\top f + \lambda u = 0 \tag{33}$$

$$\left(A^\top A + \lambda I\right) u = A^\top f \tag{34}$$

$$u = \left(A^\top A + \lambda I\right)^{-1} A^\top f. \tag{35}$$

3. **Energy minimization 2**

Find the solution to the following energy minimization problem

$$\arg\min_{u,\sigma} \frac{|Au - f|^2 + \epsilon}{\sigma} + \lambda \log \sigma. \tag{36}$$

Solution To find the solution we need to compute the gradient with respect to both σ and u and set it to zero. We start by working with σ. Define $\Delta = |Au - f|^2 + \epsilon$. Then, the gradient of the energy with respect to σ is

$$-\frac{\Delta}{\sigma^2} + \frac{\lambda}{\sigma} = 0. \tag{37}$$

This yields immediately the first solution

$$\sigma = \frac{\Delta}{\lambda}. \tag{38}$$

We can then plug this in the energy and compute the gradient with respect to u. The new optimization problem is then

$$\arg\min_{u} 1 + \lambda \log \frac{|Au - f|^2 + \epsilon}{\lambda}. \tag{39}$$

The gradient of this new energy can then be equated to zero

$$\frac{\lambda A^\top (Au - f)}{|Au - f|^2 + \epsilon} = 0 \tag{40}$$

and the solution is as in the previous exercises

$$u = \left(A^\top A\right)^{-1} A^\top f. \tag{41}$$

4. **The role of regularization (challenging)**

Suppose that the following energy is provided

$$\hat{x} = \arg\min_{x} |Ax - b|^2 + \lambda |x|^2 \tag{42}$$

with a given matrix A, a vector b and regularization parameter λ. Analyze how the solution of this minimization problem varies with λ by using the singular value decomposition of A.

Solution By taking the gradient with respect to x of the energy and by setting the gradient to zero, we obtain

$$A^T(Ax - b) + \lambda x = 0. \tag{43}$$

Finally, we obtain

$$x = (A^T A + \lambda I_d)^{-1} A^T b. \tag{44}$$

Let us decompose A via the singular value decomposition. Then, we obtain

$$A = USV^T \tag{45}$$

where U and V are orthonormal matrices and S is diagonal and with nonnegative entries. Then, we obtain

$$x = V(S^2 + \lambda I_d)^{-1} SU^T b = V \text{diag}\left\{\frac{s_i}{s_i^2 + \lambda}\right\}_{i=1,\ldots,n} U^T b. \tag{46}$$

The diagonal elements $\frac{s_i}{s_i^2+\lambda}$ tend to $1/s_i$ when $\lambda = 0$ and to 0 when $\lambda \mapsto \infty$. When $\lambda = 0$ the solution is indeed $x = A^\dagger b$ where $A^\dagger$ is the Moore-Penrose pseudoinverse. Thus, a large regularization parameter λ projects singular values of the inverse to 0, and these then lower the norm of x (as expected when minimizing the regularization term $|x|^2$).

5. **Discretization of the energy [Python implementation available]**

Consider the following discretized energy for a 1D denoising problem

$$\min_u \sum_{i=1}^{N-1} |u[i+1] - u[i]| + \frac{\lambda}{2} \sum_{i=1}^{N} (u[i] - f[i])^2 \tag{47}$$

where f is a given discrete signal and $\lambda > 0$ is some regularization parameter. Compute the gradient and write the gradient descent algorithm. Implement it in Python and test it with some signals f.

Solution The gradient of the energy with respect to $u[i]$ gives

$$\nabla E[i] = \text{sign}(u[i] - u[i-1]) - \text{sign}(u[i+1] - u[i]) + \lambda(u[i] - f[i]) \quad \forall 1 < i \leq N-1 \tag{48}$$

$$\nabla E[1] = -\text{sign}(u[2] - u[1]) + \lambda(u[1] - f[1]) \tag{49}$$

$$\nabla E[N] = \text{sign}(u[N] - u[N-1]) + \lambda(u[N] - f[N]). \tag{50}$$

The gradient descent is then

$$u^{t+1} = u^t - \epsilon \nabla E(u^t). \tag{51}$$

Fig. 1 shows an example of what one should obtain. See Python code `denoising1Dmethod1.py`.

6. **Gauss/Gauss-Seidel/SOR [Python implementation available]**

Consider the linear system corresponding to the gradient of this energy

$$\min_u \sum_i \frac{1}{2} |\nabla u[i]|^2 + \frac{\lambda}{2} \sum_i (u[i] - f[i])^2 \tag{52}$$

where f is a given 1D signal and $\lambda > 0$ is a regularization parameter. Compute the inverse of the linear system matrix by using a direct inverse, Gauss' algorithm, Gauss-Seidel's algorithm and the Successive Over-Relaxations algorithm. Implement these methods in Python and test with different f signals.

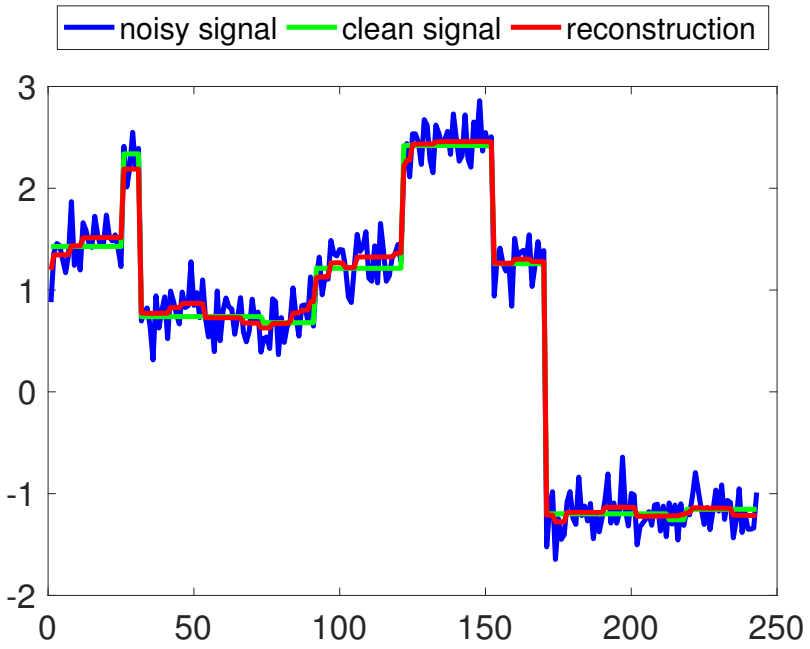

Fig. 1: 1D Total Variation denoising.

Solution The derivation of the matrix A and the vector b is available in the slides. Firstly, we notice that the matrix A of the linear system is diagonally dominant. Then, for Gauss we compute the solution of $Au = b$ by using the iteration

$$u^{t+1}[i] = \frac{b[i] - \sum_{j \neq i} A[i,j] u^t[j]}{A[i,i]}. \tag{53}$$

For Gauss-Seidel we compute the solution of $Au = b$ by using the iteration

$$u^{t+1}[i] = \frac{b[i] - \sum_{j<i} A[i,j] u^{t+1}[j] - \sum_{j>i} A[i,j] u^t[j]}{A[i,i]}. \tag{54}$$

For Successive Over-Relaxations we define $w \in [1,2)$ and compute the solution of $Au = b$ by using the iteration

$$u^{t+1}[i] = (1-w)u^t[i] + w\frac{b[i] - \sum_{j<i} A[i,j] u^{t+1}[j] - \sum_{j>i} A[i,j] u^t[j]}{A[i,i]}. \tag{55}$$

See Python code `compareMatrixInverseMethods.ipynb`.

7. **Iterative linearization [Python implementation available]**

Consider the energy

$$\min_u \sum_i \frac{1}{2} |\nabla u[i]|^2 + \frac{\lambda}{2} \sum_i \log\left[(u[i] - f[i])^2 + \epsilon\right] \tag{56}$$

where f is a given 1D signal and $\lambda, \epsilon > 0$ are regularization parameters. Discretize the derivatives via forward differences, compute the gradient equations, linearize them around some solution u^t at iteration time t, and solve the corresponding linear system by using any of the solver seen in class (Gauss, Gauss-Seidel, or SOR). Write all as an iterative algorithm. Implement it in Python and test it with some signals f.

Solution The forward difference gives

$$\nabla u[i] = u[i+1] - u[i] \quad \forall i = 1, \dots, N-1 \tag{57}$$

so we rewrite the energy as

$$\min_u \frac{1}{2}\sum_{i=1}^{N-1}(u[i+1]-u[i])^2+\frac{\lambda}{2}\sum_{i=1}^{N}\log\left[(u[i]-f[i])^2+\epsilon\right]. \tag{58}$$

The gradient of the energy with respect to $u[i]$ with $i=2,\dots,N-1$ gives

$$\nabla E[i]=(2u[i]-u[i+1]+u[i-1])+\lambda\frac{u[i]-f[i]}{(u[i]-f[i])^2+\epsilon} \tag{59}$$

$$=\sum_{j=1}^{N}\Delta[i,j]u[j]+\lambda\frac{u[i]-f[i]}{(u[i]-f[i])^2+\epsilon} \tag{60}$$

where we have introduced the matrix Δ since the gradient of the regularization term is linear. The definition of Δ comes directly from the definition of the gradient and it corresponds to a discrete Laplacian operator.

The iterative linearization algorithm uses the update rule $u^{t+1}=u^t+\delta u$, where δu solves the linearized gradient equations (first order Taylor expansion around u^t). By plugging $u=u^{t+1}$ in eq. (60) and linearizing with respect to δu we obtain for $i=2,\dots,N-1$

$$0=\sum_{j=1}^{N}\Delta[i,j]u^t[j]+\lambda\frac{u^t[i]-f[i]}{(u^t[i]-f[i])^2+\epsilon}+\sum_{j=1}^{N}\Delta[i,j]\delta u[j]+\lambda\frac{\epsilon-(u^t[i]-f[i])^2}{\left((u^t[i]-f[i])^2+\epsilon\right)^2}\delta u[i]. \tag{61}$$

Let us now define the given term b as

$$b[i]=-\sum_{j=1}^{N}\Delta[i,j]u^t[j]-\lambda\frac{u^t[i]-f[i]}{(u^t[i]-f[i])^2+\epsilon} \tag{62}$$

and the matrix A as

$$A[i,j]=\Delta[i,j]+\lambda\frac{\epsilon-(u^t[i]-f[i])^2}{\left((u^t[i]-f[i])^2+\epsilon\right)^2}I_d[i,j] \tag{63}$$

where I_d denotes the identity matrix.

We then have to solve

$$A\delta u=b \tag{64}$$

via Gauss, Gauss-Seidel or SOR and, finally, the iterative algorithm is

$$u^{t+1}=u^t+\delta u=u^t+A^{-1}b. \tag{65}$$

See Python code `denoising1DIterativeLinearization.ipynb`.

3 Shading

1. **Normal integration [Python implementation available]**

   Suppose that the normal map $\mathbf{n}$ of a depth map d is given. Write the equation that relates the depth map to the normal map and then write an algorithm to reconstruct the depth map from the normal map.

   Solution If we write the 3D point on the surface corresponding to a pixel (x,y) as

$$P=\begin{bmatrix}x\\y\\d(x,y)\end{bmatrix} \tag{66}$$

then we can write the tangent vectors by taking the first order derivatives with respect to x and y

$$T_x = \begin{bmatrix} 1 \\ 0 \\ d_x(x,y) \end{bmatrix} \qquad T_y = \begin{bmatrix} 0 \\ 1 \\ d_y(x,y) \end{bmatrix}. \tag{67}$$

Since the normal to the surface must be orthogonal to both tangent vectors, we have

$$\mathbf{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \propto \begin{bmatrix} \nabla d \\ -1 \end{bmatrix}. \tag{68}$$

Then we can write

$$d_x = D_x d = -\frac{n_1}{n_3} \tag{69}$$

$$d_y = D_y d = -\frac{n_2}{n_3} \tag{70}$$

where D_x and D_y are the matrices denoting derivatives with respect to x and y. Thus we have a linear system

$$\begin{bmatrix} D_x \\ D_y \end{bmatrix} d = -\begin{bmatrix} \frac{n_1}{n_3} \\ \frac{n_2}{n_3} \end{bmatrix}. \tag{71}$$

Now we can invert this linear system by using any of the linear system solvers.

To ensure smoothness of the solution (the normals might be noisy or not satisfy the integrability constraint) one could also add another set of equations. For example, one could impose that the Laplacian of d be zero

$$\Delta d = 0. \tag{72}$$

When combining all these equations together, the solution will be a tradeoff (that is, it will not satisfy exactly all the equations). One can determine which set of equations should be favored by introducing a parameter $\lambda > 0$. For example, we could solve

$$\begin{bmatrix} D_x \\ D_y \\ \lambda\Delta \end{bmatrix} d = -\begin{bmatrix} \frac{n_1}{n_3} \\ \frac{n_2}{n_3} \\ 0 \end{bmatrix}. \tag{73}$$

See Python code `Normal Integration.ipynb`.

2. **Integrability and shading [Python implementation available]**

Suppose that the image of a Lambertian object is made available. The illumination of the scene is due to a far point light source of known orientation $s = [s_1 \ s_2 \ s_3]^T \in S^2$ (a vector in the unit sphere) and intensity $L = 1$. We also assume that the albedo ρ of the object is constant and equal to 1. Given the shading model $I(x,y) =< n(x,y), s >$, where $n(x,y)$ is the normal vector at the pixel (x,y) on the surface, reconstruct the normal map n. Make use of the integrability constraint and use the simple parametrization in p and q. Write Python code to perform the reconstruction.

Solution The integrability in p and q can be written as

$$p_y(x,y) = q_x(x,y). \tag{74}$$

Then, we can aim to minimize the following discretized energy in the unknowns p and q

$$\begin{aligned} E(p,q) = &\sum_{i,j} \left(I[i,j] - \frac{p[i,j]s_1 + q[i,j]s_2 - s_3}{\sqrt{1 + p^2[i,j] + q^2[i,j]}} \right)^2 \\ &+ \lambda \sum_{i,j} |\nabla p[i,j]|^2 + |\nabla q[i,j]|^2 \\ &+ \mu \sum_{i,j} |p_y[i,j] - q_x[i,j]|^2 \end{aligned} \tag{75}$$

with regularization parameters $\lambda > 0$ and $\mu > 0$. The first term is the data term. The second term is the regularization (smoothness in x and y of the normal components). Finally, the third term is the integrability. The gradient equations give

$$\begin{aligned}\nabla_p E(p,q)[i,j] = &\left(\frac{p[i,j]s_1 + q[i,j]s_2 - s_3}{\sqrt{1+p^2[i,j]+q^2[i,j]}} - I[i,j]\right)\frac{s_1 + s_1 q^2[i,j] - s_2 p[i,j]q[i,j] + s_3 p[i,j]}{(1+p^2[i,j]+q^2[i,j])^{3/2}} \\ &- 2\lambda \sum_{m,n} \Delta[i,j,m,n]p[m,n] \\ &+ 2\mu(2p[i,j] - p[i,j+1] - p[i,j-1] - q[i,j] + q[i+1,j] - q[i+1,j-1] + q[i,j-1]) = 0 \end{aligned} \tag{76}$$

$$\begin{aligned}\nabla_q E(p,q)[i,j] = &\left(\frac{p[i,j]s_1 + q[i,j]s_2 - s_3}{\sqrt{1+p^2[i,j]+q^2[i,j]}} - I[i,j]\right)\frac{s_2 + s_2 p^2[i,j] - s_1 p[i,j]q[i,j] + s_3 q[i,j]}{(1+p^2[i,j]+q^2[i,j])^{3/2}} \\ &- 2\lambda \sum_{m,n} \Delta[i,j,m,n]q[m,n] \\ &+ 2\mu(2q[i,j] - q[i+1,j] - q[i-1,j] - p[i-1,j+1] + p[i-1,j] + p[i,j+1] - p[i,j]) = 0 \end{aligned} \tag{77}$$

where Δ denotes the 2D Laplacian matrix. Then, we use gradient descent to minimize E. See Python code `Shading Reconstruction.ipynb`.

4 Photometric Stereo

1. Under certain simplifying assumptions, you can write that the intensity of a particular point P on an object illuminated by a single point light source as $\mathbf{I}_P = \rho_P \mathbf{N}_P \cdot \mathbf{S}$ where ρ_P is the albedo at a point P on the object surface, $\mathbf{N}_P$ is the normal vector at P and $\mathbf{S}$ is the light source direction expressed as a unit vector. We are interested in computing the albedo and the normal vector at P from the image intensities. You can change the direction of the light source and acquire multiple intensity measurements.

   (a) What is the minimum number of images you need to compute ρ and $\mathbf{N}$? Are there any restrictions that the measurement process must satisfy?

   Solution

   The equation $I_P = \rho_P \mathbf{N}_P \cdot \mathbf{S}$ provides one equation on ρ_P, $\mathbf{N}_P$ and $\mathbf{S}$. Since we know the direction of the point source $\mathbf{S}$, we have four unknown variables ρ_P, n_1, n_2, n_3 where $\mathbf{N}_P = [n_1\ n_2\ n_3]^\top$. If we constrain $\mathbf{N}_P$ to be a normal vector, i.e. $\|\mathbf{N}_P\| = 1$, we have three independent variables per pixel (corresponding to the 3D point P): ρ_P, n_1, n_2 and we know that $n_3 = \sqrt{1 - n_1^2 - n_2^2}$. Hence we require at least 3 images to compute ρ and $\mathbf{N}$. The placement of the light sources in the three images must be such that they are linearly independent (otherwise we do not have enough equations in our linear system).

   (b) Show the steps involved in solving for ρ and $\mathbf{N}$ provided that you have exactly the required minimum number of images. Write down the equations involved.

   Solution

   With 3 images and source directions $\mathbf{s_1}$, $\mathbf{s_2}$, and $\mathbf{s_3}$ respectively, we formulate the following system of equations

$$I_P^1 = \rho_P(s_{11}n_1 + s_{12}n_2 + s_{13}n_3) \tag{78}$$

$$I_P^2 = \rho_P(s_{21}n_1 + s_{22}n_2 + s_{23}n_3) \tag{79}$$

$$I_P^3 = \rho_P(s_{31}n_1 + s_{32}n_2 + s_{33}n_3) \tag{80}$$

$$\mathbf{I} = \begin{bmatrix} I_P^1 \\ I_P^2 \\ I_P^3 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ s_{31} & s_{32} & s_{33} \end{bmatrix} \begin{bmatrix} \rho_P n_1 \\ \rho_P n_2 \\ \rho_P n_3 \end{bmatrix} = S\mathbf{n} \tag{81}$$

$$|\mathbf{N}_P| = \sqrt{n_1^2 + n_2^2 + n_3^2} = 1 \tag{82}$$

Thus we get a system like $\mathbf{I} = S\mathbf{n}$ where $\mathbf{n} = \rho_P \mathbf{N}_P$ is not normalized. We have then $\mathbf{n} = S^{-1}\mathbf{I}$, and $\rho_P = \|\mathbf{n}\| \Rightarrow \mathbf{N}_P = \frac{\mathbf{n}}{\rho_P}$.

(c) Is it advantageous to have more than the minimum number of images? If yes, why? How would your solution change?

Solution

Yes, then we can have a least squares solution which will be more robust to noise and other inconsistencies of measurements. The solution in this case can be computed as:

$$\mathbf{n} = (S^T S)^{-1} S^T \mathbf{I}$$

where S and $\mathbf{n}$ are defined in the previous answer.

2. The relationship between a 3D point at world coordinates (X, Y, Z) and its corresponding 2D pixel at image coordinates (u, v) can be defined as a projective transformation, i.e. a 3×4 camera projection matrix P.

- How many degrees of freedom does the projection matrix P have? Briefly justify your answer.

  Solution. 11 degrees of freedom.

  Any multiplication by a non-zero scalar results in an equivalent camera matrix. Since P maps vectors that are represented in homogeneous coordinates, so $P(aX)$ (a is a non-zero scalar) represents the same point as PX.

3. Consider a plane with a Lambertian surface defined by the equation $z = 0$. The surface has constant albedo $\rho = 1$ and is illuminated by a point light source. The plane is imaged by an orthographic camera. The light source is positioned at the 3D point $\begin{bmatrix}5 & 5 & 10\end{bmatrix}^\top$. Derive equations to prove that the brightest point on the surface is at the 3D point $\begin{bmatrix}5 & 5 & 0\end{bmatrix}^\top$.
Hint: Use the Lambertian shading model. However, this case is different from the classic photometric stereo setting, where light sources are located at infinity. In this case the light source is close to the surface, therefore the light direction changes at each point on the plane.

Solution.
The brightness of a lambertian surface is given by the equation

$$B(\mathbf{x}) = \rho \mathbf{N}(\mathbf{x}) \cdot \mathbf{L}(\mathbf{x}) I_L. \tag{83}$$

since $N(\mathbf{x}) \cdot \mathbf{L}(\mathbf{x}) = |N(\mathbf{x})||\mathbf{L}(\mathbf{x})| \cos\alpha$ where α is the angle between the normal of the surface and the direction of the light, given constant N (the surface is a place) and I_L, the maximum $B(\mathbf{x})$ is obtained when $\cos\alpha = 1$, that is when the vectors are parallel. Therefore the maximum brightness is obtained at the point $\begin{bmatrix}5 & 5 & 0\end{bmatrix}$. This corresponds to a spotlight on a flat surface.