Fundamentals of 3D

Lecture 9:

Laplacian Image pyramids

Expectation-Maximization

+ Overview of computational photography

Frank Nielsen
nielsen@lix.polytechnique.fr
Interpreting Fourier spectra

- Stripes of the hat
- Stripes of the hair

Fourier log power spectrum
Laplacian image pyramids

Used also in graphics for texturing (mipmapping)
Gaussian. Blur and sample, and then Laplacian. Interpolate and estimate

\[
G_1 = I \\
G_i = \text{EXPAND}(G_{i+1}) + L_i. \quad \text{Reconstruction}
\]

\[
L_i = G_i - \text{EXPAND}(G_{i+1}) \quad \text{Residual}
\]

\[
\begin{array}{ll}
L_1 = G_1 - \text{EXPAND}(G_2) \\
L_2 = G_2 - \text{EXPAND}(G_3) \\
L_3 = G_3 - \text{EXPAND}(G_4) \\
L_4 = G_4 \\
G_4 = L_4 + \text{EXPAND}(G_5) \\
G_3 = L_3 + \text{EXPAND}(G_4) \\
G_2 = L_2 + \text{EXPAND}(G_3) \\
G_1 = L_1 + \text{EXPAND}(G_2) = I
\end{array}
\]

Precursors of wavelets
Laplacian image pyramids

Blurring is efficient for sampling as it removes high-frequency components. (sample at fewer positions.)

Gaussian kernel and resampling at a \(\text{quarter}\) of the image size. Blurring and resampling is computed using a \(\text{single}\) discrete kernel.

- Central limit theorem:
  (mean of random variables approach Gaussian distribution)

- Infinitely differentiable functions

- Fourier of Gaussians are Gaussians

- Human brain has neuronal regions doing Gaussian filtering
Laplacian image pyramids

\[ G_1 = I \]
\[ G_2 \]
\[ G_3 \]
\[ G_4 \]
\[ G_5 \]
\[ L_1 \]
\[ L_2 \]
\[ L_3 \]
\[ L_4 \]

**FIGURE 4.46**  Gaussian and Laplacian image pyramids: the original image \( I \) can be reconstructed without any error from the smallest image of the Gaussian pyramid \( (G_5) \) and the Laplacian image pyramid \( L = \{L_i\}_i \).
Laplacian image pyramids: Application to blending

**Multiband blending.**

Blending two overlapping images using their pyramids

- Compute Laplacian pyramids $L(I_1)$ and $L(I_2)$ of $I_1$ and $I_2$.
- Generate a hybrid Laplacian pyramid $L_r$ by creating for each image of the pyramid a 50%/50% mix of images, obtained by selecting the leftmost half of $L(I_1)$ with the rightmost half of $L(I_2)$.
- Reconstruct blended images from the Laplacian pyramid $L_r$. 
Laplacian image pyramids: Application to blending

Left pyramid  blend  Right pyramid
Laplacian image pyramids: Application to blending

(d) (h) (l)
Nowadays, we better use **Poisson image editing** and gradient/image reconstruction
Expectation-Maximization (EM)

\[ x; \theta_m \sim \mathcal{N}(x; \mu_m, \Sigma_m) \]

\[ \mathcal{N}(x; \mu, \Sigma) = (2\pi)^{-d/2} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \]

\[ f(Y = y|\theta) = \sum_{j=1}^{k} \alpha_j \frac{1}{\sqrt{(2\pi)^d |\Sigma_j|}} \exp\left\{ -\frac{1}{2} (y - \mu_j)^T \Sigma_j^{-1} (y - \mu_j) \right\} \]

http://www.neurosci.aist.go.jp/~akaho/MixtureEM.html
Indicator variables $z$

\[
\mathbf{z}_i = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 \end{bmatrix}^T
\]

Multinomially distributed $\pi_m$

\[
p(x_i | z_{im} = 1; \theta) = \frac{1}{(2\pi)^{d/2} |\Sigma_m|^{1/2}} \exp \left\{ -\frac{1}{2} (x_i - \mu_m)^T \Sigma_m^{-1} (x_i - \mu_m) \right\}
\]
Generating samples from Gaussian Mixture Models (GMMs)

1: for \( i = 1 \) to \( N \) do
2: \( m \leftarrow \text{index of one of the } M \text{ models randomly selected according to the prior probability vector } \pi \)
3: \( \text{Randomly generate } x_i \text{ according to the distribution } \mathcal{N}(x_i; \mu_m, \Sigma_m) \)
4: end for
Maximize the likelihood (incomplete) \( \mathcal{L}(\theta) = p(X; \theta) \)

\[ \theta = \{\mu_m, \Sigma_m, \pi_m\}_{1}^{M} \]

Maximize the likelihood (complete likelihood) \( p(X, Z; \theta) \)

joint distribution of \( X \) and \( Z = \{z_i\}_{1}^{N} \)
Expectation-Maximization algorithm: Iteration

EM iteration:

- **Expectation step:**

\[
 w_{tj} = p(x_t = j | y_t) = \frac{\alpha_j f(y_t | \mu_j, \Sigma_j)}{\sum_{i=1}^{k} \alpha_i f(y_t | \mu_i, \Sigma_i)}
\]

- **Maximization step:**

\[
 \hat{\alpha}_j \leftarrow \frac{1}{n} \sum_{t=1}^{n} w_{tj}
\]

\[
 \hat{\mu}_j \leftarrow \frac{\sum_{t=1}^{n} w_{tj} y_t}{\sum_{t=1}^{n} w_{tj}}
\]

\[
 \hat{\Sigma}_j \leftarrow \frac{\sum_{t=1}^{n} w_{tj} (y_t - \hat{\mu}_j)(y_t - \hat{\mu}_j)^T}{\sum_{t=1}^{n} w_{tj}}
\]

Initialize with k-means (or k-means++)