SUV Color Space & Filtering

Computer Vision I
CSE252A
Lecture 11

Announcement
- HW2 extended to Tuesday

Blob Tracking for Robot Control

Motivation: Lambertian Algorithm Applied to Non-Lambertian Surface: Photometric Stereo

Dichromatic Reflection Model

Color depends on light source color and diffuse color

Transparent Film

Color of light source
**Dichromatic Reflection Model**

- Image formation: Color Channel \( k \)

\[
I_k = (D_k f_d + S_k f_s(\theta)) \mathbf{n} \cdot \mathbf{l}
\]

- Data-dependent SUV Color Space

\[
I_{SUV} = [R] I_{RGB}
\]

- Example

- Properties of SUV

- Image formation: Color Channel \( k \)

Where \( f_d \) and \( f_s \) are the diffuse and specular BRDF.

Image color lies in span of diffuse color \( D \) and specular color \( S \).

First row of \( R \) is specular color \( S \). Other rows are orthogonal to \( S \).

\( U, V \) spans a plane orthogonal to \( S \).

- Shading information is preserved in \( u \) and \( v \) channels.

\[
I_U = r^2 D f_d \mathbf{n} \cdot \mathbf{l}
\]

\[
I_V = r^3 D f_d \mathbf{n} \cdot \mathbf{l}
\]
Multi-channel Photometric Stereo

\[ J = [t_0, t_1] \]

\( J^k \): 2-channel color vector under the \( k \)-th light source.

\( \rho \): 2-channel UV albedo.

\[ J^k = [\rho_0^k, \rho_1^k]^T = (\hat{n} \cdot \hat{P}) \rho. \]

Shading vector \( \hat{P} = \begin{bmatrix} f_1^k & f_2^k & f_3^k \end{bmatrix}^T = \begin{bmatrix} \hat{P}_1^k \hat{P}_2^k \hat{P}_3^k \end{bmatrix} \hat{n} \)

Intensity matrix \( [J] = \begin{bmatrix} I_1^k & I_2^k & I_3^k \\ I_4^k & I_5^k & I_6^k \\ I_7^k & I_8^k & I_9^k \end{bmatrix} = \hat{P} \rho^T. \)

The least squares estimate of the shading vector \( P \) is the principal eigenvector of \( [J] P = \hat{P} \rho^T \hat{n} \).

Qualitative Results

Quantitative Results

What is image filtering?

- Modify the pixels in an image based on some function of a local neighborhood of the pixels.

\[
\begin{array}{c|c|c}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\quad
\begin{array}{c|c|c}
5 & 5 & 5 \\
5 & 5 & 5 \\
5 & 5 & 5 \\
\end{array}
\]

Local image data

Modified image data

(From Bill Freeman)
Smoothing by Averaging

Kernel:

Linear Filters

- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Properties
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e., shift the input image two pixels to the left, the output is shifted two pixels to the left)

Example: smoothing by averaging
- Form the average of pixels in a neighbourhood

Example: smoothing with a Gaussian
- Form a weighted average of pixels in a neighbourhood

Example: finding a derivative
- Form a weighted average of pixels in a neighbourhood

Convolution

Image (I)

Kernel (K)

Note: Typically Kernel is relatively small in vision applications.

Convolution: $R = K \ast I$

$R(i, j) = \sum_{h=-m}^{m} \sum_{k=-m}^{m} K(h, k) I(i-h, j-k)$

Kernel size is $m+1$ by $m+1$
Convolution: $R = K \ast I$

Kernel size is $m+1$ by $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k)I(i-h, j-k)$$
Convolution: $R = K * I$

Kernel size is $m+1$ by $m+1$

$$R(i, j) = \sum_{h=0}^{m} \sum_{k=0}^{m} K(h, k) I(i-h, j-k)$$

Impulse Response

$$0 0 0 0 0 0 0 0$$

Linear filtering (warm-up slide)

original

Pixel offset

Filtered (no change)
Practice with linear filters

![Original Image]

Original

![Filter Image]

Filter

Practice with linear filters

![Original Image]

Original

![Sharpening Filter Image]

Sharpening filter

- Accentuates differences with local average

Smoothing by Averaging

![Kernel Image]

Kernel

Filters are templates

- Applying a filter at some point can be seen as taking a dot-product between the image and some vector.
- Filtering the image is a set of dot products.

Properties of convolution

Let f, g, h be images and * denote convolution.

\[ f \ast g(x, y) = \iint f(x-u, y-v)g(u, v)dudv \]

- Commutative: f \ast g = g \ast f
- Associative: f \ast (g \ast h) = (f \ast g) \ast h
- Linear: for scalars a & b and images f, g, h
  \[ (af+bg) \ast h = af \ast h + bg \ast h \]
- Differentiation rule
  \[ \frac{\partial}{\partial x} (f \ast g) = \frac{\partial f}{\partial x} \ast g = f \ast \frac{\partial g}{\partial x} \]
Filtering to reduce noise

- Noise is what we’re not interested in.
  - We’ll discuss simple, low-level noise today: Light fluctuations; Sensor noise; Quantization effects; Finite precision
  - Not complex: shadows; extraneous objects.
- A pixel’s neighborhood contains information about its intensity.
- Averaging noise reduces its effect.

Additive noise

- \( I = S + N \). Noise doesn’t depend on signal.
- We’ll consider:
  - \( I_i = s_i + n_i \) with \( E(n_i) = 0 \)
  - \( s_i \) deterministic. \( n_i \) a random var.
  - \( n_i, n_j \) independent for \( i \neq j \)
  - \( n_i, n_j \) identically distributed

Averaging Filter

- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

\[
F = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{9}
\]

(Camps)

Does it reduce noise?

- Intuitively, takes out small variations.

\[
E(\hat{N}(i,j)) = 0 \\
E(\hat{N}^2(i,j)) = \frac{1}{m^2} m \sigma^2 \Rightarrow \hat{N}(i,j) - N(0, \frac{\sigma}{\sqrt{m}}) 
\]

(Camps)
An Isotropic Gaussian

- The picture shows a smoothing kernel proportional to

$$e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

(which is a reasonable model of a circularly symmetric fuzzy blob)

Smoothing with a Gaussian

Kernel: [image]

Smoothing by Averaging

Kernel: [image]