Spring 2019: Venu: Haag 315, Time: M/W 4-5:15pm

ECE 5582 Computer Vision  
Lec 04: Filtering and Edge Features  

Zhu Li  
Dept of CSEE, UMKC  
Office: FH560E, Email: lizhu@umkc.edu, Ph: x 2346.  
http://l.web.umkc.edu/lizhu
Outline

• Recap of Lec 03
  • Perspective Projection & Homography
  • SVD and an example
  • HW-1

• Image Filtering – A Quick Review
  • Image Filtering and Correlation
  • Edge Detection

• Edge Descriptors
  • HOG – Histogram of Oriented Gradients
Camera Projection: Intrinsic + Extrinsic Parameters

- Camera Intrinsic + Extrinsic Parameters:

\[
\begin{align*}
\mathbf{x} &= \mathbf{K}[\mathbf{R} \ \mathbf{t}] \mathbf{X} \\
\begin{bmatrix}
    u \\
    v \\
    w \\
\end{bmatrix} &= \\
\begin{bmatrix}
    \alpha & 0 & u_0 \\
    0 & \beta & v_0 \\
    0 & 0 & 1 \\
\end{bmatrix} \\
\begin{bmatrix}
    r_{11} & r_{12} & r_{13} & t_x \\
    r_{21} & r_{22} & r_{23} & t_y \\
    r_{31} & r_{32} & r_{33} & t_z \\
\end{bmatrix} \\
\begin{bmatrix}
    x \\
    y \\
    z \\
    1 \\
\end{bmatrix}
\end{align*}
\]

- Intrinsic: camera center: \([u_0, v_0]\), focus length/aspect ratio: \([\alpha, \beta] = [kf, lf]\)
- Extrinsic: Rotation \(\mathbf{R} = R_x R_y R_z\), translation \(\mathbf{t} = [t_x, t_y, t_z]\)
Homography

• In general, homography $H$ maps 2d points according to, $x' = Hx$

\[
\begin{bmatrix}
    x' \\
    y' \\
    w'
\end{bmatrix} =
\begin{bmatrix}
    h_1 & h_2 & h_3 \\
    h_4 & h_5 & h_6 \\
    h_7 & h_8 & h_9
\end{bmatrix}\begin{bmatrix}
    x \\
    y \\
    w
\end{bmatrix}
\]

• Up to a scale, as $[x, y, w] = [sx, sy, sw]$, so $H$ has 8 DoF

• Affine Transform:
  • 6 DoF: Contains a translation $[t_1, t_2]$, and invertable affine matrix $A_{2x2}$

\[
H = \begin{bmatrix}
    a_1 & a_2 & t_1 \\
    a_3 & a_4 & t_2 \\
    0 & 0 & 1
\end{bmatrix}
\]

• Similarity Transform,
  • 4DoF: a rigid transform that preserves distance if $s=1$:

\[
H = \begin{bmatrix}
    s \cos(\theta) & -s \sin(\theta) & t_1 \\
    s \sin(\theta) & s \cos(\theta) & t_2 \\
    0 & 0 & 1
\end{bmatrix}
\]
Homography Estimation – SVD Pseudo Inv

- In matrix form, we have, \( Ah = 0 \)
- If we have more than 4, then the system is over-determined, we will find a solution by least squares, computationally via SVD

\[
\begin{bmatrix}
-x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1 x'_1 & y_1 x'_1 & x'_1 \\
0 & 0 & 0 & -x_2 & -y_2 & -1 & x_1 y'_1 & y_1 y'_1 & y'_1 \\
-x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2 x'_2 & y_2 x'_2 & x'_2 \\
0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2 y'_2 & y_2 y'_2 & y'_2 \\
& & & & & & & & \\
& & & & & & & & \\
& & & & & & & & \\
& & & & & & & & \\
\end{bmatrix}
\begin{bmatrix}
h_1 \\
h_2 \\
h_3 \\
h_4 \\
h_5 \\
h_6 \\
h_7 \\
h_8 \\
h_9 \\
\end{bmatrix} = 0,
\]

- Pseudo Inverse:
  - SVD: \( A = VSU^T \)
  - Pseudo inverse (minimizing least square error)
    \[
    A^+ = US^+V^T
    \]
  - Details: https://www.ecse.rpi.edu/~qji/CV/svd_review.pdf
HW-1

• Compute HSV color histogram for CIFAR 100 image retrieval
  • rgb2hsv
  • uniform quantize HSV space
  • euclidean distance hist ranking
  • try other metrics, like KL distance as bonus

• Compute Homography
Outline

• Recap of Lec 03
  • SVD and an example

• Image Filtering – A Quick Review
  • Image Filters
  • Edge Detection

• Edge Feature
What is an Image?

- The famous Lenna....

Lenna, 1971

Lenna, 2019

Lena at ICIP 2015, Quebec City
What is an image?

- We can think of a (grayscale) image as a **function**, $f$, from $\mathbb{R}^2$ to $\mathbb{R}$ (or a 2D **signal**):
  - $f(x,y)$ gives the **intensity** at position $(x,y)$

- A **digital** image is a discrete (**sampled**, **quantized**) version of this function
Image as a function: \( I = f(x, y) \);

- A grid (matrix) of intensity values

\[ (\text{common to use one byte per value: } 0 = \text{black}, 255 = \text{white}) \]
Motivation: Why Filtering?

• Some use cases...

- De-noising
- Super-resolution
- Edge detection

Deep Learning: VGG16
Image filtering – Looking at pixel neighbors

- Modify the pixels in an image based on some function of a local neighborhood of each pixel
  - Discrete form:

\[
\begin{align*}
  f[n, m] \rightarrow \text{System } S \rightarrow g[n, m] \\
  g = S[f], \quad g[n, m] = S\{f[n, m]\} \\
  f[n, m] \xrightarrow{S} g[n, m]
\end{align*}
\]

![Image filtering example](image)

Local image data

Modified image data

Some function
Linear filtering

- One simple version: linear filtering (cross-correlation, convolution)
  - Replace each pixel by a linear combination of its neighbors
- The prescription for the linear combination is called the "kernel" (or "mask", "filter")

Source: L. Zhang
Cross-correlation

- Cross-Correlation:
  - No flipping of kernel

Let $F$ be the image, $H$ be the kernel (of size $2k+1 \times 2k+1$), and $G$ be the output image

$$G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]$$

This is called a cross-correlation operation:

$$G = H \otimes F$$
Convolution

- Same as cross-correlation, except that the kernel is “flipped” (horizontally and vertically), called convolution

\[ G = H \ast F \]

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v] \]

- Convolution is **commutative** and **associative**
  - \( F = G \ast h + G \ast f = G \ast (h + f) \)
  - \( F = G \ast h \ast f = G \ast (h \ast f) \)
Mean filtering

- Find the average

\[ \begin{bmatrix}
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
\end{bmatrix} \ast 
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} = 
\begin{bmatrix}
0 & 10 & 20 & 30 & 30 & 30 & 20 & 10 \\
0 & 20 & 40 & 60 & 60 & 60 & 40 & 20 \\
0 & 30 & 60 & 90 & 90 & 90 & 60 & 30 \\
0 & 30 & 50 & 80 & 80 & 90 & 60 & 30 \\
0 & 30 & 50 & 80 & 80 & 90 & 60 & 30 \\
0 & 20 & 30 & 50 & 50 & 60 & 40 & 20 \\
10 & 20 & 30 & 30 & 30 & 20 & 10 \\
10 & 10 & 10 & 0 & 0 & 0 & 0 \\
\end{bmatrix} 
\]
Linear filters: examples

- Identity filter:

Original $\ast$ \[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\] = Identical image

Source: D. Lowe
Linear filters: examples

- Shift Filter

Original  \*  \[
\begin{array}{ccc}
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
\end{array}
\]  =  Shifted left By 1 pixel

Source: D. Lowe
Linear filters: examples

- Average/Blurring

Original

\[ \ast \frac{1}{9} \]

Blur (with a mean filter)

Source: D. Lowe
Linear filters: examples

- Sharpen: remove the average.

Original

\[ \begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix} \begin{array}{c}
\text{Sharpening filter}
\end{array} \]

before

after

Source: D. Lowe
Smoothing with box filter

• Box Filter:

• Box filter in integral image domain:
  • much faster, just 4 ADD operations.
  • More details in CABOX coverage

Source: D. Forsyth
Gaussian Kernel

- Gaussian Kernel with scale $\sigma$

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Matlab: `h=fspecial('gaussian', 5, 1.0);`
Gaussian Filters – A Scale Space Approximation

- Gaussian Blur

Matlab:

```matlab
n=8; s = 1.25.^[1:n]; m=fix(6.*s);
figure(30);
for k=1:8
    subplot(2,4,k); h = fspecial('gaussian', m(k), s(k));
    imagesc(h); title(sprintf('s = %1.1f', s(k)));
end

figure(31);
subplot(3,3,1); imshow(im); title('f_0(1.0)');
for k=1:n
    subplot(3,3,k+1);
    h = fspecial('gaussian', m(k), s(k));
    f = imfilter(im, h);
    imshow(f);
    title(sprintf('f_%d(%1.1f)', k+1, s(k)));
end
```
Gaussian filter properties

- Successive Gaussian filtering
  - convolution of Gaussian is still Gaussian

\begin{equation}
\text{new kernel sigma: } h_1 + h_2
\end{equation}

Matlab:
\begin{align*}
h_1 &= \text{fspecial('gaussian', 11, 1.2)}; \\
h_2 &= \text{fspecial('gaussian', 11, 2.0)}; \\
h_3 &= \text{conv2}(h_1, h_2); \\
h_4 &= \text{fspecial('gaussian', 11, 2.0+1.2)};
\end{align*}
Separable Filter

- Gaussian filter is separable:
  \[ h = u \otimes v \]

- verify via SVD:
  - \( h = \text{fspecial('gaussian', 11, 1.2)} \);
  - \([u,s,v] = \text{svd}(h); \text{plot(diag(s))};\)
Sharpening revisited

• What does blurring take away?

Let's add it back:

\[
\text{original} - \text{smoothed (5x5)} = \text{detail}
\]

\[
\text{original} + \alpha \text{detail} = \text{sharpened}
\]

Source: S. Lazebnik
Sharpen filter - LoG

- Laplacian of Gaussian

\[ F + \alpha (F - F \ast H) \]

image

blurred image

scaled impulse

Gaussian

Laplacian of Gaussian

unit impulse (identity)
Sharpen filter
Convolution in the real world

• Physical world convolutions

Bokeh: Blur in out-of-focus regions of an image.


Source: http://lullaby.homepage.dk/diy-camera/bokeh.html
Filtering in Matlab

- Area of support for the operations

\[
\text{N1} \times \text{M1} \ast \begin{bmatrix} \text{N2} \\ \text{N2} \times \text{M2} \end{bmatrix} = \begin{bmatrix} \text{(N1 + N2 - 1)} \\ \text{(M1 + M2 - 1)} \end{bmatrix}
\]

- To give the same n1 x m1 output, need to padding the edge
  - Default is zero padding
  - Also replicate the last edge pixel
  - Or, mirroring (used in MPEG codec)
Image Filtering, Sweet Deal with Matlab

• It is such a nice tool
  • Main filter operation: \( \text{im2} = \text{imfilter}(\text{im}, \text{h}, \text{‘replicate’}) \)
  • Design your filter: \( \text{h} = \text{fspecial}(\text{‘filter_type’}, \text{kernel_size}, \text{options}) \)

• Filter Design Examples:
  • Sobel
  • Laplacian, Laplacian of Gaussian
  • Gaussian, Difference of Gaussian (SIFT)
Matlab Image Filtering Example

%%%%%% image filters
%%%%%%
im = imread('../pics/Lenna.png');
im = rgb2gray(im);
im1 = double(im(201:320, 201:320));

% edge filters
h{1} = fspecial('sobel');
h{2} = fspecial('laplacian', 0.25);
h{3} = fspecial('log', 7, 0.25);
% gaussians
sigmas = [1.226, 1.554, 1.946];
for k=1:length(sigmas)
    h{3+k} = fspecial('gaussian',11, sigmas(k));
end
% diff of gaussian
h{7} = (h{6} - h{4}); h{7} = h{7}/sum(sum(h{7}));
h{8} = h{5} - h{4}; h{8} = h{8}/sum(sum(h{8}));

for k=1:8
    fprintf('k=%d', k);
    figure(26); subplot(2,4,k); grid on; hold on; colormap('gray'); imagesc(h{k});
    figure(27); subplot(2,4,k); imshow(imfilter(im, h{k}, 'replicate'));
end
Filtering Properties

• Linear operation that is
  • **Shift-Invariant:**
    \[ f(m-k, n-j) * h = g(m-k, n-j), \text{ if } f * h = g \]

• **Associative:**
  \[ f * h_1 * h_2 = f * (h_1 * h_2) \]
  this can save a lot of complexity

• **Distributive:**
  \[ f * h_1 + f * h_2 = f * (h_1 + h_2) \]
  useful in SIFT’s DoG filtering.
Non-Linear Filters

- Linear Filters usually remove details from image in smoothing/denoising
- Non-linear filters like the median filter does better in this

\[ I(u, v) = \text{median}\{I(j, k) \in R(u, v)\} \]

Median Filtering in Matlab:
I = imread('cameraman.tif');
J = imnoise(I,'salt & pepper',0.02);
K = medfilt2(J);
subplot(121);imshow(J);
subplot(122);imshow(K);
Outline

• Recap of Lec 03
• Image Filtering – A Quick Review
  • Image Filtering and Correlation
• Edge Descriptors
  • Edge detection
  • Edge description
Edge Detection

- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels
Edge Mechanism in Images

• Various ways of having edges in images:

- depth discontinuity
- surface normal discontinuity
- surface color discontinuity
- illumination discontinuity
Characterizing edges

• An edge is a place of rapid change in the image intensity function.
How can we differentiate a digital image $F[x,y]$?

- Option 1: reconstruct a continuous image, $f$, then compute the derivative
- Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$$

How would you implement this as a linear filter?

$\frac{\partial f}{\partial x} \cdot H_x$  \hspace{1cm}  $\frac{\partial f}{\partial y} \cdot H_y$

Source: S. Seitz
The gradient points in the direction of most rapid increase in intensity.

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]

The edge strength is given by the gradient magnitude:

\[ ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \]

The gradient direction is given by:

\[ \theta = \tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right) \]

• how does this relate to the direction of the edge?

Source: Steve Seitz
Image Gradient Example

- Compute gradient energy

\[ \| \nabla f \| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} \]
Unreliable Gradient due to Noise

Where is the edge?

Source: S. Seitz
Smoothing before gradient computing

- Instead smooth the image with a Gaussian Filter
- Then do gradient (difference) operation
- Is this good enough?
• Differentiation is convolution, and convolution is associative:

\[
\frac{d}{dx}(f \ast h) = f \ast \frac{d}{dx} h
\]

Source: S. Seitz
2D edge detection filters

- DoG filter

\[
\begin{align*}
    h_\sigma(u, v) &= \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \\
    \frac{\partial}{\partial x} h_\sigma(u, v) &= \frac{\partial}{\partial u} h_\sigma(u, v)
\end{align*}
\]
Directive Gradient Detector

• How to figure out dominant direction of the edge?

(From vector calculus)

\[ \nabla_{\vec{u}} f (\vec{x}) = \nabla f (\vec{x}) \cdot \vec{u} \]

Directional deriv. is a linear combination of partial derivatives

\[ \frac{\partial f}{\partial x} \cdot \vec{u}_x + \frac{\partial f}{\partial y} \cdot \vec{u}_y = \nabla_{\vec{u}} f \]
Derivative of Gaussian filter with a direction

- To detect edge along a particular angle:

\[ \cos(\theta) \quad + \quad \sin(\theta) = \]
Sobel Filter

- An Approximation of DoG Filter

- Matlab Implementation:
  - Use a threshold to throw away edge strength not higher than
  - `bw_im = edge('sobel', thres)`
Sobel operator: example

Before

After

Y-direction Gradients

X-direction Gradients

Example

- original image (Lena)
Finding edges

- Gradient magnitude
- Thresholding (also thinning)

where is the edge?
Finding edges

thinning
Canny edge detector

1. Filter image with derivative of Gaussian
   
   **MATLAB**: `edge(image,'canny')`

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei
Canny edge detector

- Canny Edge Detector

![Original Image](original) ![Canny with σ = 1](canny_1) ![Canny with σ = 2](canny_2)

- The choice of $\sigma$ depends on desired behavior
  - large $\sigma$ detects “large-scale” edges
  - small $\sigma$ detects fine edges

Source: S. Seitz
Outline

• Recap of Lec 03
  • SVD and an example

• Image Filtering – A Quick Review
  • Image Filtering and Correlation
  • Edge Detection

• Edge Descriptors
  • HOG – Histogram of Oriented Gradient
  • SIFT – a taste of it
HOG Pipeline

- HoG Generation:

In practice, effect is very small (about 1%) while some computational time is required*.

## Computing Gradients - various choice of Filters

<table>
<thead>
<tr>
<th>Mask Type</th>
<th>1D centered</th>
<th>1D uncentered</th>
<th>1D cubic-corrected</th>
<th>2x2 diagonal</th>
<th>3x3 Sobel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
<td>[-1, 0, 1]</td>
<td>[-1, 1]</td>
<td>[1, -8, 0, 8, -1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operator 2</td>
<td></td>
<td></td>
<td></td>
<td>[-1 0 1]</td>
<td></td>
</tr>
<tr>
<td>Operator 3</td>
<td></td>
<td></td>
<td></td>
<td>[-1 0 2]</td>
<td>[-1 2 -1]</td>
</tr>
<tr>
<td>Miss rate at 10^-4 FPPW</td>
<td>11%</td>
<td>12.5%</td>
<td>12%</td>
<td>12.5%</td>
<td>14%</td>
</tr>
</tbody>
</table>

**Input image**

- Detecion window
- Normalise gamma & colour
- Compute gradients
- Accumulate weighted votes for gradient orientation over spatial cells
- Normalise contrast within overlapping blocks of cells
- Collect HOGs for all blocks over detection window
Making HoG Features

- Different block structure for Histogram

Variants of HOG descriptors. (a) A rectangular HOG (R-HOG) descriptor with $3 \times 3$ blocks of cells. (b) Circular HOG (C-HOG) descriptor with the central cell divided into angular sectors as in shape contexts. (c) A C-HOG descriptor with a single central cell.
- Generate local histogram per block
  - 2x2 blocks of 4x4 pixels
  - Each block generate gradient magnitude and angle data at each pixel location
  - Histogram is computed for each block which has 9 bins.

\[ f = (h_1^1, ..., h_9^1, \textcolor{red}{h_1^2} ..., \textcolor{red}{h_9^2}, h_1^3, ..., h_9^3, h_1^4, ..., h_9^4) \]

<table>
<thead>
<tr>
<th>Angle</th>
<th>0</th>
<th>15</th>
<th>25</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>15</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>95</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>47</td>
<td>97</td>
<td>10</td>
<td>12</td>
</tr>
</tbody>
</table>

| Magnitude | 5  | 20 | 20 | 10 |
|           | 5  | 10 | 10 | 5  |
|           | 20 | 30 | 30 | 40 |
|           | 50 | 70 | 70 | 80 |

- Binary voting
- Magnitude voting
HOG example

- HoG Properties
  - Fixed vector over image blocks (similar to pooled histogram)
  - Not scale invariant.
  - Not rotation invariant

In each triplet: (1) the input image, (2) the corresponding R-HOG feature vector (only the dominant orientation of each cell is shown), (3) the dominant orientations selected by the SVM (obtained by multiplying the feature vector by the corresponding weights from the linear SVM).
• **VL_FEA T:**

```matlab
cellSize = 8;
hog = vl_hog(im, cellSize, 'verbose');
```

```matlab
imhog = vl_hog('render', hog, 'verbose');
clf; imagesc(imhog); colormap gray;
```
Lec 04 Summary

- Covered basic filtering operation, which is
  - An operation on a neighborhood of pixels (details in DIP/DSP)
  - Are linear operations that can be communicative, distributive and associative
    - Implications: combine filtering operations.

- Edge detection
  - Mostly Gradient operation based detection
  - Smoothing is necessary for robust performance
  - Combine smoothing and differentiation, you get DoG – derivative of Gaussian (not Difference of Gaussian)
  - Many variations of DoGs, Canny, Sobel...etc.

- Edge Description
  - Histogram of Oriented Gradients: very successful feature, simple to compute, spatial partitioning of images blocks, and then compute gradient stats.
  - Think: how about a Histogram of Spatially Pooled Color?