**Homework #2: Feature Detection**

Due Wednesday, February 19th, 2008.

Professor Davi Geiger

**CONSIDER THE IMAGES**

![Test Images](image1.png) ![Test+Noise Images](image2.png) ![Butterfly](image3.png)

Fig 1   (a) Test Images                               (b) Test+Noise Images                        (c) Butterfly

The goal is to extract features from an image.

Part I: Edge Detection (read Image.Measurement.ppt)

a. Compute $D_\rho \hat{I}(x, y)$ for the 4 values $\theta = -\pi/2$, $-\pi/4$, $0$, $\pi/4$ and 3 scales, $s = 3$, $5$, $7$. This derivative at orientation $\rho=\theta+\pi/2$ (or $\hat{\theta}(\theta) = (-\sin \theta, \cos \theta)$) is defined as

$$D_\rho \hat{I}(x, y) = D_\rho \hat{I}(x, y, \theta + \frac{\pi}{2}, s) = [(\text{abs}(A) > \text{abs}(B)) \text{? } A : B]$$

where

$$\Delta_x = 1 \quad \text{and} \quad \Delta_y = 1 \quad \text{for} \quad \theta = -\pi/2, 0$$

$$\Delta_x = \sqrt{2} \quad \text{and} \quad \Delta_y = \sqrt{2} \quad \text{for} \quad \theta = -\pi/4, \pi/4$$

Show the results as 4 images per scale. Make sure you compute only within a frame of 7 pixels away from the boundaries (largest scale). Results can be links to a web site or e-mail

b. Compute Edgels. This means for every pixel, compute if the response of $|D_\rho \hat{I}(x, y)|$ is above a threshold and store the best $\theta$ value as well as $D_\rho \hat{I}(x, y)$. More precisely, the pseudo code for Edgels would be
Edgel (xc, yc)

\[ D_{\text{max}} = T_p \]

\[ \text{list} = \text{NIL} \]

\textbf{loop} \textbf{for} \( \theta = -\pi/2, -\pi/4, 0, \pi/4 \)

\[ \mu(x, y, \theta, s) = D_p \hat{I}(x, y, \theta + \frac{\pi}{2}, s) \]

\textbf{if} \( |\mu(x_c, y_c, \theta, s)| > D_{\text{max}} \)

\[ D_{\text{max}} = |\mu(x_c, y_c, \theta, s)| \]

\[ \theta_{\text{max}}(x, y) = \theta \]

\[ \text{list} = (\theta_{\text{max}}(x, y), \mu(x_c, y_c, \theta, s)) \]

\textbf{end}

\textbf{return} \text{list}

Note: run the pseudo code for each pixel (x, y), i.e., at each pixel one returns one edgel, which is a list. The parameter \( T_p(s) \) must be empirically determined. Note that as the scale increases, more smoothing is applied so the threshold parameter should get smaller. We can expect it to be more than 10 and less than 150 (which represent the minimal changes in intensity that are detected as an Edgel). Please, say for each image used and each scale s, what value of \( T_p(s) \) did you use.

How to display the result of Edgels?

Create an image array and assign 255 to every pixel. For each pixel (x, y) where Edgel is not Nil (\( \text{Edgel}(x, y) \neq \text{NIL} \)) store the value \( I(x, y) = \text{floor} \left( \frac{160}{\pi} \left( \theta_{\text{max}} + \frac{\pi}{2} \right) \right) \), where \( \theta_{\text{max}} \) is extracted as the first element of the list in Edgel(x, y). In this way the output will be “white=255” wherever there is no edgel, and wherever there is an edgel, the darkness value will represent the best angle direction at that pixel. Note that the second element of Edgel contains the strength of the derivative, which can be positive or negative, but this term will not be shown in the results (it will be used in the next problem).
c. Compute Corners for scales 5 and 7 only (no need for scale 3). This means for every pixel, compute if the response of \( |D_{p} \hat{I}(x,y)| \) is above a threshold for two (or more) angles, and store the two best \( \theta \)'s value as well as the product of the \( D_{p} \hat{I}(x,y) \)'s. More precisely, the pseudo code for corners would be

\[
\text{Corner (xc,yc)}
\]
\[
D_{\text{max}} = T_{p}
\]
\[
\text{list} = \text{NIL}
\]
\[
\text{if Edgel (xc,yc) } \\
\quad \theta_{\text{max}}(x,y) = \text{first (Edgel(xc,yc))} \\
\text{loop for } \theta_{2} = -\pi/2, -\pi/4, 0, \pi/4 \quad /* look for a second angle "edge" response */ \\
\quad \text{if } \theta_{2} \neq 0 \\
\quad \quad \mu(x_{c}, y_{c}, \theta_{2}, s) = D_{p} \hat{I}(x_{c}, y_{c}, \theta_{2} + \frac{\pi}{2}, s) \\
\quad \quad \text{if } |\mu(x_{c}, y_{c}, \theta_{2}, s)| > D_{\text{max}} \\
\quad \quad \quad \theta_{2,\text{max}}(x_{c}, y_{c}) = \theta_{2} \\
\quad \quad \quad \mu_{c}(x_{c}, y_{c}, \theta, \theta_{\text{max}}, s) = \mu(x_{c}, y_{c}, \theta_{\text{max}}, s) \mu(x_{c}, y_{c}, 2 \theta_{\text{max}}, s) \\
\quad \quad \text{list} = (\theta_{\text{max}}, x_{c}, y_{c}, \theta_{\text{max}}, \theta_{\text{max}, 2 \theta_{\text{max}}, s})
\]
\[
\text{return } \text{list}
\]

How to display the result of corners? Create an image array and assign 255 to every pixel. For each pixel \((x,y)\) where Corner is not Nil \((\text{Corner}(x,y) \neq \text{NIL})\) store the value \(I(x,y)=\)

\[
\text{floor}\left[160 \left( \frac{\theta_{\text{max}}}{\pi} + \frac{\pi}{2} \right)\right], \text{ where } \theta_{2,\text{max}} \text{ is extracted as the second element of the list in Corner(x,y). In this way the output will be "white=255" wherever there is no Corner, and wherever there is a Corner, the darkness value will represent the second best angle direction at that pixel (\( \theta_{2,\text{max}} \)). Note that the first and third element of Corner contains respectively the best angle and the strength of the Corner, which can be positive or negative.}

d. Cleaning up the corner response to select one corner at each location.

Corners are expected to be sparse features, but we observe that near a corner multiple responses to corner will occur. We can group all the responses near a corner and select the largest response as the accurate response. How to identify and remove multiple responses near a corner? We propose to start with a pixel that responds to corners, i.e., that corner \((x,y)\) is not NIL \((\text{Corner}(x,y) \neq \text{NIL})\). Then, in order to clean multiple responses, we loop around its 8
neighbors, and investigate only the ones that Cornel (neighbor(x,y)) is not NIL. Check if the largest magnitude response is larger than the current selected one and if yes, replace the current one by the largest neighbor response. Repeat the procedure until all Cornels have been checked. We find helpful to add a flag that says if the Cornel(x,y) has been visited, so flag=V, or flag=NV (not visited). To speed up the search we store Cornels in a list and add the coordinates explicitly to the Cornel representation. Here is a Pseudo Code for cleaning it.

Cleaning Cornels /* (via depth first search) */
ListCornel=Nil /* list to store cornels */
FinalList=Nil /* list to store final selected isolated corners */
loop for (x,y) in all pixels 
if Cornel (x,y) /* look for a Cornel response */
Cornel(x,y)=add((x,y),Cornel(x,y)) /* add coordinates explicitly to end of the list */
Cornel(x,y)=add(NV, Cornel(x,y)) /* add flag=NV end of the Cornel */
ListCornel =add(Cornel(x,y), ListCornel) /* add Cornel to a list */
end loop
loop for Cornel(x,y) in ListCornel /* loop through the list of Cornels */
if (fifth(Cornel(x,y))==NV) /* check if node has been visited */
Dmax=abs(third(Cornel(x,y))) /* make Dmax a global variable */
(xc , yc)=fourth(Cornel(x,y)) /* set current best location as global variable */
Fifth(Cornel(xx,yy))=V /* set flag=V for this Cornel location */
Expand (x,y)
Finalist=add((xc , yc),Finalist) /* final selected corner locations */
end loop
/* we now describe the subroutine Expand to get the best corner in a region */

Expand (x,y) /* check for neighbors of (x,y) */
if for all neighbors (Cornel(xx,yy)==Nil or fifth(Cornel(xx,yy))==V) Break /* stop expanding if all neighbors have been resolved */
else
loop for (xx,yy) in 8 neighbor of (x,y)
if fifth(Cornel(xx,yy)==NV)
if (μ > Dmax)
Dmax= μ
(xc,yc)=(xx,yy) /* set new best Cornel location */
Fifth(Cornel(xx,yy))=V /* set flag=V for this Cornel location */
Expand (xx,yy)
else
Fifth(Cornel(xx,yy))=V /* set flag=V for this Cornel location */
Expand (xx,yy)
end if
end loop
end