Edge Detection

Examples
Example – Canny scale

• Read in image “coins.png”
• Vary the scale (sigma of the Gaussian)
  – At what sigma do you just detect the outlines of the coins?
• What thresholds does it use?

```
[E, thresh] = edge(I, 'canny', [], sig);
tlow = thresh(1);
thigh = thresh(2);
fprintf('tlow=%f, thigh=%f
', tlow, thigh);
```
Example – Canny threshold

• Vary the threshold (tHigh)
  – tLow should be set to 0.4*tHigh

```matlab
% Vary thresholds
for tHigh = 0.05:0.05:0.4
    E = edge(I, 'canny', [0.4*tHigh tHigh], 1.5);
    imshow(E);
    title(sprintf('tHigh=%f', tHigh));
    pause;
end
```
Example – Canny hysteresis

• Lower the contrast in half the image
• See effect of changing the threshold

```
I = imread('coins.png');
I = double(I);
imshow(I, []);

% Lower the contrast across the image
X = 1:size(I,2);    % Get a vector of column numbers
X = X / size(I,2);  % Make a vector ranging from 0..1
M = repmat(X,size(I,1),1);  % Replicate across rows
figure, imshow(M, []);

I = I .* M;
figure, imshow(I, []);
pause
```
Example – two vs one threshold

- See effect of using two or one threshold

```matlab
% See effect of using two thresholds
[E,thresh]=edge(I, 'canny', [], 2);
disp(thresh);
figure, imshow(E), title('dual thresh');
pause
E = edge(I, 'canny', [thresh(1) thresh(1)+eps], 2);
figure, imshow(E), title('low thresh only');
pause
E = edge(I, 'canny', [thresh(2) thresh(2)+eps], 2);
figure, imshow(E), title('high thresh only');
```
Example – LoG vs DoG

• The “Laplacian of Gaussian” (LoG) filter can be approximated by a “difference of Gaussians” (DoG).

• LoG:

\[
\nabla^2 G(x, y) = \left[ \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}
\]

• DoG:

\[
DoG(x, y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}
\]

• M&H say for best approximation, \( k = \sigma_1 / \sigma_2 = 1.6 \)

\[
DoG(x, y) = \frac{1}{2\pi k^2\sigma_2^2} e^{-\frac{x^2+y^2}{2k^2\sigma_2^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}
\]
clear all
close all

x = -5:0.1:5;  % Generate values of x

% DoG
s2 = 1;      % Pick a value for sigma2
s1 = 1.6*s2; % Marr-Hildreth say best approximation is when s1/s2=1.6
DoG = (1/(2*pi*s1^2)) * exp(-x.^2/(2*s1^2)) - ...  
      (1/(2*pi*s2^2)) * exp(-x.^2/(2*s2^2));
DoG = DoG/DoG(x==0); % Scale so that DoG=1 at x=0
plot(x,DoG);

% LoG
s = 1;        % Pick a value for sigma
LoG = ( x.^2 - 2*s^2 ) .* exp(-x.^2/(2*s^2));
LoG = LoG/LoG(x==0); % Scale so that LoG=1 at x=0
figure, plot(x,LoG);

% We can put both of them on the same plot.
figure, plot(x,LoG,x,DoG);
Instead of choosing the sigma of LoG arbitrarily, let’s choose it so that LoG and DoG have the same zero crossings – then we have a better comparison.
Example – LoG vs DoG (continued)

- LoG is zero when
  \[ x^2 + y^2 = 2\sigma^2 \]
- DoG is zero when
  \[ \frac{1}{2\pi k^2 \sigma_2^2} e^{\frac{x^2+y^2}{2k^2\sigma_2^2}} = \frac{1}{2\pi \sigma_2^2} e^{\frac{-x^2+y^2}{2\sigma_2^2}} \]
- Find a relationship between \(\sigma\) and \(\sigma_2\); i.e., find \(\sigma\) in terms of \(\sigma_2\).
- Then plot LoG and DoG; they should be very close.
Edge detection in OpenCV

• Use “Canny” function to detect edges using the Canny operator
• Also use “trackbars” to allow user to interactively adjust threshold of edge operator, and sigma for image blurring
/* My OpenCV program. */
#include <iostream>
#include <opencv2/opencv.hpp>

int main(int argc, char* argv[]) {
    printf("Hit ESC key to quit\n");

    // Create image window named "My Image". (You actually don't have to do
    // this step, but this command allows you to set properties of the window,
    // such as its location, or whether you can resize it.)
    cv::namedWindow("My Image");

    // Create trackbar for threshold
    int thresh_pos = 10;
    int thresh_max = 200;
    cv::createTrackbar("threshold", "My Image", &thresh_pos, thresh_max);

    // Create trackbar for sigma
    int sigma_pos = 1;
    int sigma_max = 10;
    cv::createTrackbar("sigma", "My Image", &sigma_pos, sigma_max);

    cv::VideoCapture cap(0); // open the default camera
    cv::waitKey(2000); // wait a bit to let the camera start up
    if (!cap.isOpened()) { // check if we succeeded
        printf("error - can't open the camera\n");
        system("PAUSE");
        return -1;
    }
}
while (true) {
    // Read an image
    //cv::Mat image = cv::imread("C:/Program Files/OpenCV2.4/samples/c/lena.jpg");
    cv::Mat image;
    cap >> image;
    if (!image.data) {
        printf("error - image is empty\n");
        system("PAUSE");
        return -1;
    }

    // Convert to gray
    cv::Mat imageInputGray;
    cvtColor(image, imageInputGray, CV_BGR2GRAY);

    // Smooth the image with a Gaussian filter. If sigma is not provided, it
    // computes it automatically using  sigma = 0.3*((ksize-1)*0.5 - 1) + 0.8
    cv::Mat imageBlur;
    double sigma = sigma_pos;         // sigma of Gaussian
    cv::GaussianBlur(imageInputGray, imageBlur,
                     cv::Size(0, 0),  // kernel size (should be odd numbers; if 0, compute from sigma)
                     sigma, sigma);    // sigmax, sigmay

    cv::Mat imageEdges;
    double lowThreshCanny = thresh_pos;         // lower threshold
    Canny(imageBlur, imageEdges, lowThreshCanny,
           3 * lowThreshCanny,    // upper threshold
           3,    // size of Sobel operator
           true); // use more accurate L2 norm

    cv::imshow("My Image", imageEdges);        // Show the image in the window
    //cv::imshow("My Image", image);

    // Wait for xx ms (0 means wait until a keypress)
    if (cv::waitKey(1) == 27) break; // hit ESC (ascii code 27) to quit
}

return EXIT_SUCCESS;