CS 558: Homework Assignment 1

Alana Laryssa Seabra A Santos

2/10/2016

1 Gaussian filtering

The next three sets of images bellow show the input images with gaussian filter applied with three different values of σ.

(a) $\sigma = 1$ (b) $\sigma = 2$ (c) $\sigma = 5$

Figure 1: Different values of σ with red.pgm

(a) $\sigma = 1$ (b) $\sigma = 4$ (c) $\sigma = 7$

Figure 2: Different values of σ with plane.pgm

(a) $\sigma = 1$ (b) $\sigma = 2$ (c) $\sigma = 3$

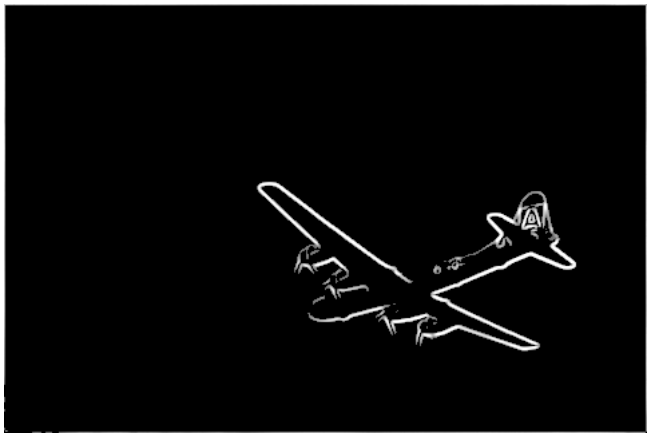
Figure 3: Different values of σ with kangaroo.pgm

2 Gradient computation

Bellow I show the best result I got with each one of the input images. All input images for this step were preprocessed with gaussian filter ($\sigma = 1$).

(a) kangaroo.pgm

(b) red.pgm

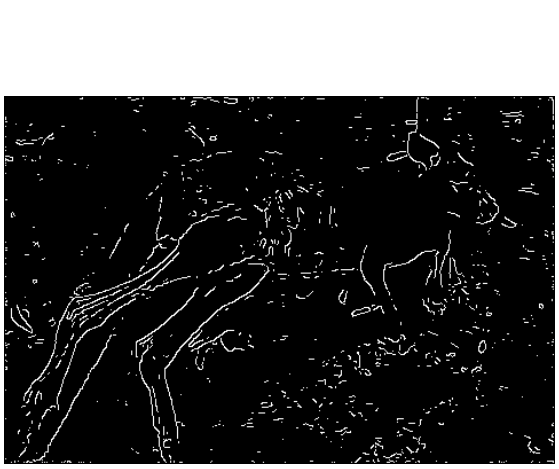


(c) plane.pgm

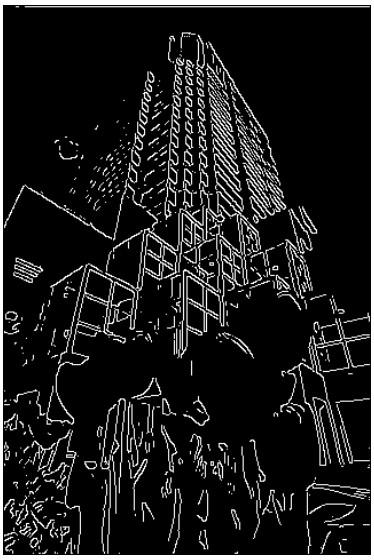
Figure 4: Gradient strength of the input images

3 Non-maximum suppresion

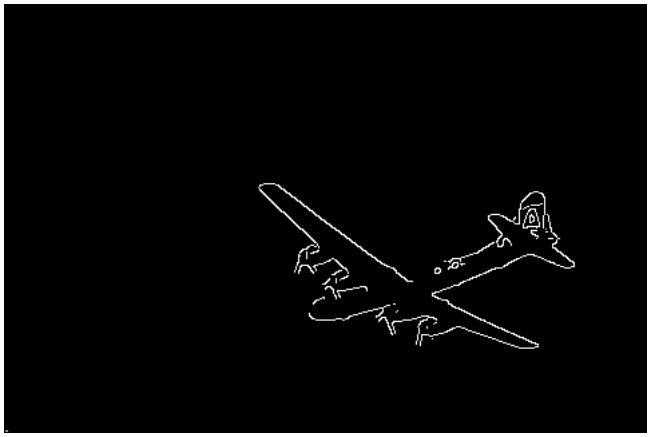
Bellow are the results when non-maximum suppression technique for edge thinning was applied to each of the images in the previous step.



(a) kangaroo.pgm



(b) red.pgm



(c) plane.pgm

Figure 5: Gradient strength of the input images

4 Matlab code

Bellow are the Matlab scripts used to generate the presented results

```
im = imread('cs558s16_hw1/kangaroo.pgm');
% im = imread('cs558s16_hw1/plane.pgm');
% im = imread('cs558s16_hw1/red.pgm');
% im = [zeros(10,10) ones(10,10); ones(10,10) zeros(10,10)];

im = im2double(im);

% gaussian params
sigma = 1;
% after sobel filter
isedge_thresh = 100;


%% 1. gaussian filter
if sigma ~= 0
    % wingauss = 5;
    % halfgauss = (wingauss-1)/2;
    halfgauss = 3*sigma - 1;

    [x,y] = meshgrid(-halfgauss:halfgauss, -halfgauss:halfgauss);
    G = exp(-(x.^2 + y.^2)/(2*sigma^2));  % no need to compute the const part
    G = G./sum(G(:));  % sum has to be 1

    im1 = filtering(im, G);
else
    im1 = im;
end

%% 2. gradient with sobel filter
Sx = [-1 0 1; -2 0 2; -1 0 1];
Sy = [1 2 1; 0 0 0; -1 -2 -1];

im1x = filtering(im1, Sx);  % has neg values
im1y = filtering(im1, Sy);

strength = sqrt(im1x.^2 + im1y.^2);
direction = atand(im1y./im1x);

strength(im2uint8(strength) < isedge_thresh) = 0;  % binary

%% 3. non-maximum suppression
im2 = nonmaxsup(strength, direction);
edges = im2;
edges(edges > 0) = 1;
```

```
function im2 = filtering (im, f)
 % work with images already casted to double
 [s1, s2] = size(f);
 hs1 = (s1-1)/2; hs2 = (s2-1)/2;

 im2 = im;  % copy pixels near the borders
%  im2 = zeros(size(im,1),size(im,2));  % fill borders with black

 for i = hs1+1 : size(im,1) - hs1
     for j = hs2+1 : size(im,2) - hs2
         im2(i,j) = sum(sum(f.*im(i-hs1:i+hs1, j-hs2:j+hs2)));
     end
 end
end


function im2 = nonmaxsup(grad, dir)
 % input: gradient, directions

 [s1, s2] = size(grad);
 im2 = zeros(size(grad,1), size(grad,2));

 for i = 2:s1-1
     for j = 2:s2-1
         if grad(i,j) ~= 0
             x1 = 0; y1 = 0;
             x2 = 0; y2 = 0;
             if dir(i,j) > 67.5 || dir(i,j) <= -67.5 % 90 degrees
                 x1 = -1; y1 = 0;
                 x2 = 1; y2 = 0;
             elseif dir(i,j) <= 67.5 && dir(i,j) > 22.5 % 45 degrees
                 x1 = -1; y1 = 1;
                 x2 = 1; y2 = -1;
             elseif dir(i,j) <= 22.5 && dir(i,j) > -22.5 % 0 degrees
                 x1 = 0; y1 = -1;
                 x2 = 0; y2 = 1;
             elseif dir(i,j) <= -22.5 && dir(i,j) > -67.5 % -45 degrees
                 x1 = -1; y1 = -1;
                 x2 = 1; y2 = 1;
             else
                 disp('wrong');
             end

             temp = [grad(i,j) grad(i+x1,j+y1) grad(i+x2,j+y2)];
             if max(temp) == grad(i,j)
```

```
                im2(i,j) = grad(i,j);
            end
        end
    end
 end

end
```