```
function g = seg_Gradient(image, sigma, alpha, q)
if ~exist('sigma', 'var') sigma = 0.3; end
if ~exist('alpha', 'var') alpha = 0.1; end
if ~exist('q', 'var') q = 1; end
image = double(image);
if size(image, 3) == 1
 disp('have to convert grayscale to color image first!');
 channel = image;
 image = zeros([size(channel), 3]);
 image(:, :, 1) = channel;
 image(:, :, 2) = channel;
 image(:, :, 3) = channel;
end
%if sigma > 0
%    kernel_size = 2 * round(3 * sigma) + 1;
%    image = imfilter(image, fspecial('gaussian', [kernel_size kernel_size],
sigma), 'replicate');
%end
mask = [-1 0 1];
image_c1_x = imfilter(image(:, :, 1), mask, 'replicate');
image_c1_y = imfilter(image(:, :, 1), mask', 'replicate');
%image_c2_x = imfilter(image(:, :, 2), mask, 'replicate');
%image_c2_y = imfilter(image(:, :, 2), mask', 'replicate');
%image_c3_x = imfilter(image(:, :, 3), mask, 'replicate');
%image_c3_y = imfilter(image(:, :, 3), mask', 'replicate');
%grad_mag = (image_c1_x.^2 + image_c2_x.^2 + image_c3_x.^2 + image_c1_y.^2 +
image_c2_y.^2 + image_c3_y.^2).^0.5 / 3;
grad_mag = (image_c1_x.^2 + image_c1_y.^2).^0.5;
if (q == 1)
   g = exp(-alpha * grad_mag);
else
   g = exp(-alpha * grad_mag.^q);
end
% get those pixels in g which are very small
small_gradient_pixels = find(g < 0.1);
g(small_gradient_pixels) = 0.1;
g = g.^-1;
g = g ./ max(g(:));
%min(g(:))
%max(g(:))
```