```matlab
function [Ix, Iy] = ImageDerivatives(im)
    % x derivation
    Kx = [0.5 -0.5];
    Ix = conv2(im, Kx, 'full');
    Ix = Ix(:, 1:end-1);
    Ix = BordersToZero(Ix);
    %show(Ix)


    % y derivation
    Ky = [-0.5; 0.5];
    Iy = conv2(im, Ky, 'full');
    Iy = Iy(1:end-1, :);
    Iy = BordersToZero(Iy);
    %show(Iy)

end


function [im] = BordersToZero(im)
    im(:, [1 end]) = 0;
    im([1 end], :) = 0;
end


function [L] = Deriv2Laplace(Ix, Iy)
    Kx = [0.5 -0.5];
    Ky = [-0.5; 0.5];
    L = conv2(Ix, Kx, 'same') + conv2(Iy, Ky, 'same');
end

function [R, L] = do_retinex(I,T)

    % Take log and calculate the derivatives of the log image.
    [Ix, Iy] = ImageDerivatives(log(I));

    % Calculate the norm of the derivative at each point
    norm = sqrt(Ix.^2 + Iy.^2);

    % Set all derivatives whose norm is less than T to zero.
    Ix(norm < T) = 0;
    Iy(norm < T) = 0;

    % Use the function Deriv2Laplace to compute the Laplacian.
    Laplas = Deriv2Laplace(Ix, Iy);

    % Compute the inverse Laplacian kernel K using the provided invDel2.
    [invK] = invDel2(size(I));

    % Convolve the Laplacian with K to obtain the log reflectance
    logRef = conv2(Laplas, invK, 'same');

    % Exponentiate the log reflectance to obtain a reflectance image.
    R = exp(logRef);
    L = I./R;

end


function q3
    im = double(imread('simul_cont_squares.tif'));
    [Ix, Iy] = ImageDerivatives(im);
    [L] = Deriv2Laplace(Ix, Iy);
    imshow(L);

    % after threshold
    threshold = 10;
    L = abs(L);
    L_threshold = L >= threshold;
    imshow(L_threshold);

end

function q4
    im = double(imread('cross.tif'));
    [Ix, Iy] = ImageDerivatives(im);
    [L] = Deriv2Laplace(Ix, Iy);
    imshow(L);

    % after threshold
    threshold = 5;
    L = abs(L);
    L_threshold = L >= threshold;
    imshow(L_threshold);

end

function q5
    im = double(imread('kofka_ring.tif'));
    [Ix, Iy] = ImageDerivatives(im);
    [L] = Deriv2Laplace(Ix, Iy);
    %imshow(L);

    % after threshold
    threshold = 20;
    L = abs(L);
    L_threshold = L >= threshold;
    imshow(L_threshold);

end

function  q8

    [im] = twoSquares(1);
    show(im,[0 2])

    T = 0.07;
    [R1, L1] = do_retinex(im,T);
    d1 = diag(R1);
    imshow(R1)
    imshow(L1)
    plot(d1)


    [im] = twoSquares(2);
    show(im,[0 2])

    T = 0.07;
    [R2, L2] = do_retinex(im,T);
    d2 = diag(R2);
    plot(d2)
    imshow(R2)
    imshow(L2)

end

function q9

    load('checkerShadow.mat');
    figure;
    imshow(im1,[0 1])
    title('org image')

    % Verify that indeed the two checks have exactly the same intensity
    disp('Intensities at (y1,x1), (y2,x2):');
    disp([im1(y1,x1) im1(y2,x2)])


    T_lst = [0.02, 0.07, 0.2];
    for i = 1:length(T_lst)
        T = T_lst(i);
        %T =  0.07;
        [R, L] = do_retinex(im1,T);
        disp('calculated reflectance of the two checks, T is:');
        disp(T)
        disp([R(y1,x1) R(y2,x2)])
        figure;
        imshow(R)
        title_str = sprintf('R, T= %.2f' ,T);
        title(title_str);
    end

end

function q10

    load('runner.mat');
    figure;
    imshow(im1, [0 1])
    title('org image')

    figure;
    T = 0.05:0.02:0.15;
    for i = 1:6
        [R, L] = do_retinex(im1, T(i));
        subplot(2,3,i);
        str = sprintf('R, T=%.2f', T(i));
        imshow(R, [0 2]);
        title(str);
    end

end

function q11

    load('couch.mat');
    figure;
    imshow(im1, [0 1])
    title('org image')

    figure;
    T = 0.01:0.01:0.04;
    for i = 1:4
        [R, L] = do_retinex(im1, T(i));
        subplot(2,2,i);
        str = sprintf('R, T=%.2f', T(i));
        imshow(R, [0 1]);
        title(str);
    end

end
```