Optical Flow: Estimate Flow

The perception of motion and the subsequent formation of an interpretation guides our everyday lives. The ability to determine if an object is moving, judge its speed and direction, and react accordingly is fundamental to our survival. The apparent motion which guides our actions is called optical flow. Since optical flow is determined by time varying image intensities, it is not always consistent with the true motion of objects and surfaces called the motion field. Motion estimation also plays a critical role in a variety of computer vision tasks. While applications such as object tracking, scene reconstruction and image alignment have very different objectives, they all rely to some degree on low-level motion cues.

In this lab you will estimate the optical flow between a pair of images via the implementation sketch in video .... In this section y you will combine your previous solutions to estimate the optical flow estimate over the entire image.

Your Script

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
[I1, I2, I3, I4] = test_images();

[u1,v1] = estimate_flow(I1,I2,2);
[u2,v2] = estimate_flow(I3,I4,2);

figure()
subplot(221)
imagesc(u1)
subplot(222)
imagesc(u2)
subplot(223)
imagesc(v1)
subplot(224)
imagesc(v2)







%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
I1_ = im2double(rgb2gray(imread('parkinglot_left.png')));
I2_ = im2double(rgb2gray(imread('parkinglot_right.png')));

I1_ = imresize(I1_,.25);
I2_ = imresize(I2_,.25);

[u,v] = estimate_flow(I1_,I2_,2);

figure()
subplot(221)
imshow(I1_)
subplot(222)
imshow(I2_)
subplot(223)
imagesc(u)
subplot(224)
imagesc(v)

function [u,v] = estimate_flow(I1,I2,wsize)
    %% INPUT:
    %% I1, I2: nxm sequential frames of a video
    %% wsize: (wsize*2 + 1)^2 is the size of the neighborhood used for displacement estimation
    %% OUTPUT:
    %% u,v: nxm flow estimates in the x and y directions respectively

    % Compute the image gradients for the second image
```

```matlab
    % Compute the image gradients for the second image
    [I2_x,I2_y] = grad2d(I2);
    % The temporal gradient is the smoothed difference image
    I2_t = gauss_blur(I1-I2);

    u = zeros(size(I2));
    v = zeros(size(I2));


    % loop over all pixels in the allowable range
    for i = wsize+1:size(I2_x,1)-wsize
       for j = wsize+1:size(I2_x,2)-wsize

          % Select the appropriate window
          Ix = I2_x(i-wsize:i+wsize, j-wsize:j+wsize);
          Iy = I2_y(i-wsize:i+wsize, j-wsize:j+wsize);
          It = I2_t(i-wsize:i+wsize, j-wsize:j+wsize);

          d = estimate_displacement(Ix,Iy,It);

          u(i,j) = d(1);
          v(i,j) = d(2);
       end
    end
    % use medifilt2 with a 5x5 filter to reduce outliers in the flow estimate
    u = medfilt2(u,[5 5]);
    v = medfilt2(v,[5 5]);

end

function d = estimate_displacement(Ix,Iy,It)
    %% INPUT:
    %% Ix, Iy, It: m x m matrices, gradient in the x, y and t directions
    %% Note: gradient in the t direction is the image difference
    %% OUTPUT:
    %% d: least squares solution

    % b is the 2x1 matrix [Ix*It'; Iy*It']
    % sum over all pixels to get a 2x2 matrix
    % means sum over matrix elements


    b = [ Ix(:) Iy(:) ]' * It(:);
    A = [ Ix(:) Iy(:) ]' * [ Ix(:) Iy(:) ];

    % to help mitigate effects of degenerate solutions add eye(2)*eps to the 2x2 matrix A
    % add eps value
    A = A + eye(2)*eps;

    % use pinv(A)*b to compute the least squares solution
    d = pinv(A)*b;
end

function [I_x,I_y] = grad2d(img)
        %% compute image gradients in the x direction
        %% convolve the image with the derivative filter from the lecture
        %% using the conv2 function and the 'same' option
        dx_filter = [1/2 0 -1/2];
        I_x = conv2(img, dx_filter, 'same');

        %% compute image gradients in the y direction
        %% convolve the image with the derivative filter from the lecture
        %% using the conv2 function and the 'same' option
        dy_filter = dx_filter';
        I_y = conv2(img, dy_filter, 'same');
end

function smooth = gauss_blur(img)
```

```
    %% Since the Gaussian filter is separable in x and y we can perform Gaussian smoothing by
    %% convolving the input image with a 1D Gaussian filter in the x direction then
    %% convolving the output of this operation with the same 1D Gaussian filter in the y direction.

    %% Gaussian filter of size 5
    %% the Gaussian function is defined f(x) = 1/(sqrt(2*pi)*sigma)*exp(-x.^2/(2*sigma))
    x = -2:2;
    sigma = 1;
    gauss_size = [1 5];

    % my soln: I use fspecial('gaussian', hsize = [1 5], sigma)
    %gauss_filter_x = fspecial('gaussian', gauss_size, sigma);
    %gauss_filter_y = fspecial('gaussian', gauss_size', sigma);
    %smooth_x = imfilter(img, gauss_filter_x);
    %% convolve smooth_x with the transpose of the Gaussian filter
    %smooth = imfilter(smooth_x, gauss_filter_y);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % for edX class:
    gauss_filter = 1/(sqrt(2*pi)*sigma)*exp(-x.^2/(2*sigma^2));

    %% using the conv2 function and the 'same' option
    %% convolve the input image with the Gaussian filter in the x
    smooth_x = conv2(img, gauss_filter, 'same');
    %% convolve smooth_x with the transpose of the Gaussian filter
    smooth = conv2(smooth_x, gauss_filter', 'same');
end

```

Run Script

Previous Assessment: All Tests Passed

Submit

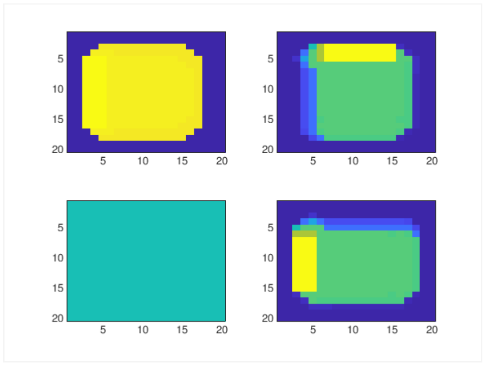
✓ **Is the optical flow estimate is correct: u1?**

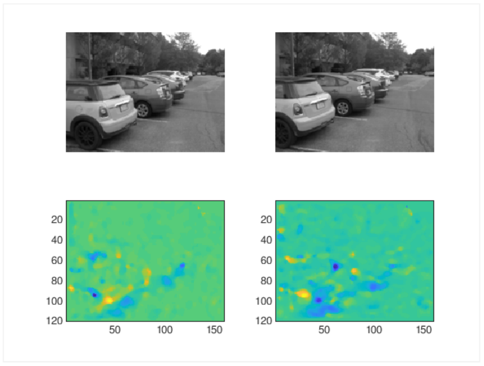
✓ **Is the optical flow estimate is correct: v1?**

✓ **Is the optical flow estimate is correct: u2?**

✓ **Is the optical flow estimate is correct: v2?**

Output


5
10
15
20
5
10
15
20


20
40
60
80
100
120
50
100
150