

CSE 398-012 Advanced Topics in Mobile Robotics  
The DARPA Urban Challenge

Homework 2: Optical Flow

Report Due Date: Friday, 4 May 07 by 5PM via blackboard

A. **Objective:** Implement the Lukas-Kanade variant of optical flow for estimating motion across an image sequence.



Figure 1: Sample Program Output

B. **Requirements:**

1. You *must* use Matlab to complete this assignment.
2. This is an individual assignment. Each student is required to submit his/her own work in order to receive credit.
3. This assignment is worth ~10% of your grade. Please treat it as such.

C. **Instructions:**

1. Convolution of an image with a Gaussian kernel is a traditional means for image smoothing, and is often used as a first step to subsequent filtering. Early computers would repeatedly convolve box filters to generate approximation for a Gaussian kernel. Write a function

```
function kernel = MakeGaussian( kernel_size )
```

that uses such an approach. Starting with a [1 1] box filter, the function should repeatedly convolve with such a filter to return a discrete Gaussian kernel of `kernel_size` width. You can assume that width is  $\geq 3$ . You should also ensure that `kernel` is normalized to have a total weight of 1.

2. An important property of convolution is associativity. Rather than convolving an image with 2 successive kernels, the same result can be achieved by first convolving the 2 kernels, and then convolving the image with the resulting kernel. This can improve run time dramatically. To demonstrate this, write a Matlab function

```
function [time1, time2] = ConvolutionTest( image )
```

that takes as an argument a grayscale image (you can convert one of your test images or generate a “pseudo-image” using the `rand` function) and returns 2 times as output:

- a. `time1`, which is the elapsed time to convolve the image with a 5x1 Gaussian kernel `G5` and then our derivative kernel `D1 = [-1 0 1]` - *i.e.*  
`im1=(image*G5)*D1`
- b. `time2`, which is the elapsed time to convolve the same Gaussian and derivative kernels together, and then the image with the resulting kernel - *i.e.* `im2=image*(G5*D1)`

You can obtain the elapsed time using the `tic` and `toc` functions from Matlab, along with `conv2` for convolution. Run this function more than once, as the first function call will be interpreted rather than compiled and will not turn a representative time.

3. Verify that the order of the two convolutions in Problem 2 does not affect the value of the resulting image. You can do this by summing over the absolute difference of `im1` and `im2`.
4. The resulting kernel from 2.b `DG7 = G5*D1` has the effect of both smoothing the image and taking the derivative of the image intensities. You can achieve the same result by taking the derivative of the Gaussian directly. To demonstrate this, take the derivative of the Gaussian function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

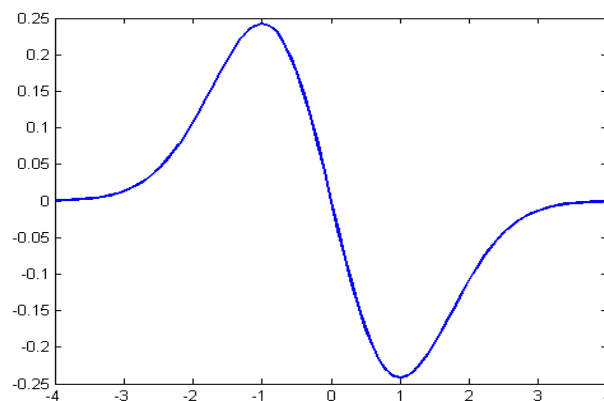
Then write a functions

```
function kernel = DGauss( kernel_width, sigma )
```

which takes as input the kernel width and standard deviation  $\sigma$ , and returns a corresponding  $1 \times (2 * \text{kernel\_width} + 1)$  kernel where the discrete values for  $f'(x)$  are sampled at integer values of  $\sigma$ . Again, ensure that the *absolute* sum of you kernel is equal to 1.

How do the values for a  $1 \times 7$  kernel compare with those obtained from 2.a? Explain the difference.

Verify the correctness of your derivation of  $f'(x)$  by plotting the resulting function for  $x = -4.0 : 0.1 : 4.0$ . You can use the `plot` command for this. Your result should look something like



5. You are now ready to begin the assignment. Download the Matlab shell program and the test 7 image sequence (provided by Ohio State U. and CV Online) from the course webpage. You *must* use this shell program. NOTE: The shell program will not work properly if you change the names of the test images.
6. Write a function `MyOpticalFlow.m` that takes as an input the number of images (in our case, this will always be 7) and a neighborhood size to be used (in our case, we will look at  $5 \times 5$  or  $7 \times 7$  neighborhoods around each pixel). The function will then display the center image with the estimated flow field superimposed. In doing this, you will generate 3 additional image sets  $I_x$ ,  $I_y$ ,  $I_t$  where:
  - a.  $I_x$  is obtained by convolving each image in the x-direction with a  $1 \times 7$  derivative of Gaussian kernel (obtained from calling `DGauss(3, 1)`), in the y-direction with a  $1 \times 7$  Gaussian kernel (obtained from calling `MakeGaussian(7)`), and in the time direction with a  $1 \times 7$  Gaussian image.
  - b.  $I_y$  is obtained by convolving each image in the x-direction with a  $1 \times 7$  Gaussian kernel, in the y-direction with a  $1 \times 7$  derivative of Gaussian kernel, and in the time direction with a  $1 \times 7$  Gaussian image.

- c. *It* is obtained by convolving each image in the x-direction with a 1x7 Gaussian kernel, in the y-direction with a 1x7 Gaussian kernel, and in the time direction with a 1x7 derivative of Gaussian image.
7. Use the flow values from the center image for display purposes.
8. Note convolving in the third dimension (*i.e.* time) can be accomplished using the `convn` function where the kernel will be defined as a 1x1x7 dimension vector. Also note that if you pass `convn` a 'valid' argument when convolving in the time direction, it will only return values that have a "valid" convolution - in our case conveniently just the center image.
9. When setting up and solving the system of linear equations at each pixel ( $i,j$ ) to solve for  $u(i,j)$  and  $v(i,j)$ , you will find the `reshape` function and pseudo-inverse operator "`\`" useful.
10. To plot arrows on the images, you can use the super-handly Matlab `quiver` function. Note that you will not want to plot arrows at every pixel, as the image will be too cluttered. To avoid this, you should subsample the flow components at each pixel (I would look at every 8 pixels or so). You can also use the super-handly Matlab function `meshgrid` as well to generate the x and y coordinates for `quiver`.
11. When you get everything working, look at different values for your neighborhood sizes (e.g., 5, 7, 9...). Note any significant differences and include representative images in your write-up.
12. Matlab has excellent documentation. Use the online help!!!

**D. Turn in:** In addition to your Matlab source code (your code should execute properly, as I *will* run it), your report must include Matlab workspace output demonstrating each of the functions above, the derivative of Gaussian plot, flow-field images, and answers to all questions mentioned above.