ASSIGNMENT #3 (EC720 A1)
"Digital Video Processing"
Date: October 13, 2017
Due date: October 30, 2017

1. *Small-kernel interpolation: 1-D* (20 points).
Image intensity interpolation is the most computationally-demanding element of motion estimation, especially at sub-pixel accuracy (often up to 80-90% of the overall motion estimation complexity). Therefore, only small-kernel interpolators are considered in motion estimation, such as linear, quadratic, and cubic operators. Learn more about the *optimal cubic interpolator* as proposed by R.G. Keys in "Cubic convolution interpolation for digital image processing", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, pp. 1153–1160, vol. 29, no. 6, Dec. 1981) available in the "Course Material/Motion Estimation" section.

   Derive an equation for the *lowest-complexity implementation* of linear and optimal/Keys cubic interpolator. Starting from the convolution equation: $f_c(x_0) = \sum_{m \in Z} f[m]h(x_0 - m)$, where f_c is the sought continuous-coordinate signal, f is a known discrete-coordinate signal, and h is the impulse response of a small-kernel interpolator (slides #6 and #11 from part 2 of motion estimation), find an equation of the following form:

$$f_c(x_0) = \Phi(..., f[\lfloor x_0 \rfloor - 1], f[\lfloor x_0 \rfloor], f[\lfloor x_0 \rfloor + 1], ...)$$

   where Φ is a polynomial function of $\Delta x = x_0 - \lfloor x_0 \rfloor$ with coefficients depending on the values of f, and $\lfloor \cdot \rfloor$ is a floor operator. Carefully consider various forms of this equation, and find one that requires the lowest number of operations. Assume that the complexity of addition is $O(N)$, where N is the order of the interpolator, the complexity of multiplication of $f[m]$ by an integer is also $O(N)$, while the complexity of its multiplication by a real value is $O(N^2)$. How many operations of each type does your implementation require?

2. *2-D linear interpolation* (20 points).
2-D interpolation of image $f[m,n]$: $f_c(x_0, y_0) = \sum_{m \in Z} \sum_{n \in Z} f[m,n]h_2(x_0 - m, y_0 - n)$ is usually implemented in a separable manner, i.e., $h_2(x,y) = h(x)h(y)$. That is, 1-D interpolation is first applied horizontally to obtain intermediate samples at location x_0 followed by vertical interpolation applied at y_0 to the intermediate samples (or *vice versa*).

   (a) How many operations of each type does a separable 2-D implementation of a *linear* interpolator require?

   (b) How many operations of each type would a *non-separable* 2-D implementation of a *separable linear* interpolator require?

   (c) Is a separable 2-D linear interpolator planar, i.e., does the intensity interpolated at (x_0, y_0) lie on a plane through $f[\lfloor x_0 \rfloor, \lfloor y_0 \rfloor]$, $f[\lfloor x_0 \rfloor + 1, \lfloor y_0 \rfloor]$, $f[\lfloor x_0 \rfloor, \lfloor y_0 \rfloor + 1]$? Justify your answer.

3. *Block-based motion estimation: Matlab* (30 points).

   (a) Implement in *Matlab* exhaustive-search, *full-pixel* precision disjoint-block matching:

$$\min_{(d_1,d_2)} \sum_{(i,j) \in B(m,n)} \psi(I_l[i,j] - I_k[i - d_1, j - d_2]), \quad \forall (m,n)$$

where $(d_1, d_2) \in Z \times Z$ is the sought integer displacement vector for block $B(m,n)$ centered at (m,n), and I_k, I_l $(k < l)$ are two images from an image sequence.

(b) Apply the developed algorithm, using $\psi(\cdot) = (\cdot)^2$ and $\psi(\cdot) = |\cdot|$ as error measures, to *missa_80* and *missa_84*, *coastguard_90* and *coastguard_95*, as well as *container_1* and *container_30* image pairs, available from the course web site. Use 16×16 blocks and as small as possible, but safe, search range to minimize the computational burden (visually estimate the maximum displacement by switching between the two images).

(c) Calculate motion-compensated prediction error defined as follows: $e[i,j] = I_l[i,j] - I_k[i - d_1, j - d_2]$. In a simple video coder this error is transmitted to the receiver in order to encode image I_l. This error is also a good indicator of how well your method works (it should contain mostly small-amplitude values). Examine this error by displaying $e + 128$ (offset allows to visualize negative error values as darker areas and positive error values as brighter areas) and explain what are the non-zero values due to and how their amplitudes can be reduced.

(d) One of the issues in designing a motion estimation algorithm for compression is the selection of error measure $\psi(\cdot)$. Compute both mean-squared error (MSE) and mean-absolute error (MAE) of $e[i,j]$. However, MSE will favor $\psi(\cdot) = (\cdot)^2$ while MAE will favor $\psi(\cdot) = |\cdot|$. A more fair approach is to compute the amount of information contained in $e[i,j]$ by means of entropy. You can use function `entropy` but be **very careful** since $e[i,j]$ can be positive or negative while `entropy` assumes non-negative input and converts any class to `uint8`.

(e) Using `subplot` print a 3×2 array of your results: images I_k and I_l, vector fields for both error measures ψ (use `quiver`), and the prediction error for both measures (3 pages, 1 per image pair). Include MSE, MAE and entropy of the prediction error for the three image pairs in titles of prediction error images or in a separate table.

(f) Comment on the results obtained.

4. *Optical flow estimation: Matlab* (30 points)

(a) Implement in *Matlab* the following iterative update equations for the Horn-Schunck optical flow algorithm derived in class:

$$u^n[i,j] = \bar{u}^n[i,j] - \frac{I_x[i,j]\bar{u}^n[i,j] + I_y[i,j]\bar{v}^n[i,j] + I_t[i,j]}{8\lambda + I_x^2[i,j] + I_y^2[i,j]} I_x[i,j]$$
$$v^n[i,j] = \bar{v}^n[i,j] - \frac{I_x[i,j]\bar{u}^n[i,j] + I_y[i,j]\bar{v}^n[i,j] + I_t[i,j]}{8\lambda + I_x^2[i,j] + I_y^2[i,j]} I_y[i,j]$$

where n is the iteration number and images are assumed to be sampled on an orthonormal lattice (pixel and image spacing equal to 1). *Pre-compute* (to speed-up calculations) partial derivatives I_x, I_y, I_t using first-order difference approximation. Use Gauss-Seidel update (in-place) when calculating velocity averages $\bar{u}^n, \bar{v}^n$ over first-order neighborhood. Make sure to read images into a *double* array of intensities in 0.0-255.0 range (this affects the choice of λ). Use zero values for the initial field u^0, v^0. Make sure to handle boundary conditions by padding/mirroring or modifying the neighborhood structure at the boundaries of vector field (u, v).

(b) Test your equations on two images: I_k = *Fall_trees_0* and I_l = *Fall_trees_0.5* available on the course web site as follows:

i. Make sure that your algorithm converges by calculating after each iteration the value of the cost function under minimization:

$$\begin{aligned} E^n &= \sum_{(i,j)} (I_x[i,j]u^n[i,j] + I_y[i,j]v^n[i,j] + I_t[i,j])^2 + \\ &\quad \lambda((u^n[i,j] - u^n[i-1,j])^2 + (u^n[i,j] - u^n[i,j-1])^2 + \\ &\quad (v^n[i,j] - v^n[i-1,j])^2 + (v^n[i,j] - v^n[i,j-1])^2) \end{aligned}$$

and also the mean-squared error (MSE) of the velocity estimates u^n, v^n with respect to the true velocity is $(0.5, 0.5)$.

ii. Run your algorithm for $\lambda = 50, 100, 200, 400, 800$ over N=200 iterations each to identify optimal λ based on MSE of the estimated velocity.

iii. For the optimal λ, plot:

A. `subplot(2,2,1)`: initial motion-compensated prediction error:

$$E^0[i,j] = I_l[i,j] - I_k[i,j]$$

B. `subplot(3,2,2)`: final motion-compensated prediction error:

$$E^N[i,j] = I_l[i,j] - \widetilde{I}_k[i - u^N[i,j], j - v^N[i,j]],$$

where $\widetilde{I}$ denotes an interpolated intensity value since u^N and v^N will not be integers. Separable linear interpolation from Problem 2 is sufficient.

C. `subplot(2,2,3)`: plot of E^n for $n = 0, ..., N$,

D. `subplot(2,2,4)`: plot of MSE of u^n and v^n for $n = 0, ..., N$ (on one plot)

E. On the second page plot $(u^0, v^0), (u^{10}, v^{10}), (u^{50}, v^{50}), (u^{200}, v^{200})$ using `quiver` and 2-by-2 grid (`subplot(2,2,*)`).

Do the prediction error images and cost function/MSE plots confirm the convergence of your algorithm? How do vector fields change as the iterations proceed? Does the final vector field agree with the motion observed between I_k and I_l?

(c) Using the above optimal λ, produce plots from (iii) for the pair: I_k = *Fall_trees_0* and I_l = *Fall_trees_1* with true velocity (1,1) and also for I_k = *Fall_trees_0* and I_l *Fall_trees_2* with true velocity (2,2). Are your observations the same as for the case of $(0.5, 0.5)$ velocity? Explain.

(d) Apply your algorithm with the optimal λ to the following frame pairs: *missa_80* and *missa_81*, *coastguard_90* and *coastguard_91*, *container_1* and *container_3*. On one page include:

i. `subplot(3,2,1)`: initial prediction error for *missa*,
ii. `subplot(3,2,2)`: final prediction error for *missa*,
iii. `subplot(3,2,3)`: initial prediction error for *coastguard*,
iv. `subplot(3,2,4)`: final prediction error for *coastguard*,
v. `subplot(3,2,5)`: initial prediction error for *container*,
vi. `subplot(3,2,6)`: final prediction error for *container*.

Plot the obtained velocity fields, one per page, using `quiver`; sub-sample each field by 2 for clarity and turn off automatic scaling (S=0). Do the obtained velocity fields agree with motion observed between the corresponding frame pairs?