ECE 634: Digital Video Systems
Spring 2023
Instructor: Prof. Fengqing Maggie Zhu

Project 1

Block Matching Algorithm

Due - February 14, 2023 at 11:59pm Eastern Time, 100 points

In this project, we will implement three Block Marching Algorithms (BMA). The third method is particularly important as it will compute faster than the first two methods.

NOTE: Submit your program source code on GitHub Classroom, and a project report on Gradescope.

Part 1
Write a C/C++ or Python program for implementing an exhaustive block matching algorithm (EBMA) with **integer-pel accuracy**.

- Program inputs: video sequence, block size, search range
- Program outputs: estimated motion field
- Assignment deliverables:
  - Program source code
  - Project report describing your findings. It must include the following items:
    1. Plot of estimated motion field for two images from video sequence
    2. The predicted image and the prediction error image
    3. The PSNR of the predicted frame relative to its original
- Experiments such as:
  - Impact of different search range
  - Impact of different predicted-block size (16×16 as starting point)

Use the YUV videos provided at `https://engineering.purdue.edu/~zhu0/ece634/sample_video.zip`. Since block size is small, motion compensation can be slow to process. Choose two frames that have sufficient motion between them so that it's easy to observe the effect of inaccurate motion estimation. If necessary, choose frame that have several intervening frames.

Part 2
Write a C/C++ or Python program for implementing an exhaustive block matching algorithm (EBMA) with **half-pel accuracy**.

- Program inputs: video sequence, block size, search range
- Program outputs: estimated motion field
- Assignment deliverables:

- Program source code
- Project report describing your findings. It must include the following items:
  1. Plot of estimated motion field for two images from video sequence.
  2. The predicted image, the prediction error image
  3. The PSNR of the predicted frame relative to its original
  4. Compare the predicted image and its PSNR with results from integer-pel accuracy
  5. Compare execution time relative to results from integer-pel accuracy

- Experiments such as:
  1. Impact of different search range
  2. Impact of different predicted-block size (16×16 as starting point)

Part 3
Write a C/C++ or Python program for implementing **hierarchical block matching algorithm (HBMA**).

- Program inputs: video sequence, block size, search range.
- Program outputs: estimated motion field.
- Assignment deliverables:
  - Program source code
  - Project report describing your findings. It must include the following items:
    1. Plot of estimated motion field for two images from video sequence.
    2. The predicted image, the prediction error image
    3. The PSNR of the predicted frame relative to its original
    4. Compare the predicted image and the PSNR of the predicted image with results from integer-pel and half-pel accuracy
    5. Compare execution time relative to results from integer- and half-pel accuracy

- Experiments such as:
  1. Impact of different search range
  2. Impact of different predicted-block size (16×16 as starting point)

Bonus Problem (20 points in addition to the 100 points of the project)
Assuming the motion between two frames can be approximated by an affine mapping. Determine the affine parameters using the indirect method. First, apply the HBMA or EBMA algorithm you implemented, to determine a block-based motion field between the two frames. Then determine the affine parameters using the least-squares method of Equation (6.7.3) in the textbook (Wang etc., *Video Processing and Communications*, see attached). Show the predicted image based on the affine parameters and the associated prediction error (in terms of PSNR) (as above). Compare the results to those obtained with the starting block-based motion estimation.

Compare the results when applied to 2 frames experiencing primarily camera motion, as well as to 2 frames experiencing no camera motion. Discuss the results. Note that for debugging purposes, it may be useful to artificially generate a pair of frames, where the second is generated by applying an affine mapping to the first.

model to the entire frame. These problems can be overcome by a *robust estimation* method [35], if the global motion is dominant over other local motions, in the sense that the pixels that experience the same global motion and only the global motion occupy a significantly larger portion of the underlying image domain than those pixels that do not.

The basic idea in robust estimation is to consider the pixels that are governed by the global motion as *inliers*, and the remaining pixels as *outliers*. Initially, one assume that all the pixels undergo the same global motion, and estimate the motion parameters by minimizing the prediction or fitting error over all the pixels. This will yield an initial set of motion parameters. With this initial solution, one can then calculate the prediction or fitting error over each pixel. The pixels where the errors exceed a certain threshold will be classified as outliers and be eliminated from the next iteration. The above process is then repeated to the remaining inlier pixels. This process iterates until no outlier pixels exist. This approach is called *Hard Threshold Robust Estimator*.

Rather than simply labeling a pixel as either inlier or outlier at the end of each iteration, one can also assign a different weight to each pixel, with a large weight for a pixel with small error, and vice verse. At the next minimization or fitting iteration, a weighted error measure is used, so that the pixels with larger errors in the previous iteration will have less impact than those with smaller errors. This approach is known as *Soft Threshold Robust Estimator*.

6.7.2 Direct Estimation

In either the hard or soft threshold robust estimator, each iteration involves the minimization of an error function. Here we derive the form of the function when the model parameters are directly obtained by minimizing the prediction error. We only consider the soft-threshold case, as the hard-threshold case can be considered as a special case where the weights are either one or zero. Let the mapping function from the anchor frame to the tracked frame be denoted by $\mathbf{w}(\mathbf{x}; \mathbf{a})$, where $\mathbf{a}$ is the vector that contains all the global motion parameters. The prediction error can be written as, following Eq. (6.2.1):

$$E_{\text{DFD}} = \sum_{\mathbf{x} \in \Lambda} w(\mathbf{x}) \, |\psi_2(\mathbf{w}(\mathbf{x}; \mathbf{a})) - \psi_1(\mathbf{x})|^p \tag{6.7.1}$$

where $w(\mathbf{x})$ are the weighting coefficients for pixel $\mathbf{x}$. Within each iteration of the robust estimation process, the parameter vector $\mathbf{a}$ is estimated by minimizing the above error, using either gradient-based method or exhaustive search. The weighting factor at $\mathbf{x}$, $w(\mathbf{x})$, in a new iteration will be adjusted based on the DFD at $\mathbf{x}$ calculated based on the motion parameters estimated in the previous iteration.

6.7.3 Indirect Estimation

In this case, we assume that the motion vectors $\mathbf{d}(\mathbf{x})$ have been estimated at a set of sufficiently dense points $\mathbf{x} \in \Lambda' \subset \Lambda$, where Λ represent the set of all pixels.

This can be accomplished, for example, using either the block-based or mesh-based approaches described before. One can also choose to estimate the motion vectors at only selected feature points, where the estimation accuracy is high. The task here is to determine $\mathbf{a}$ so that the model $\mathbf{d}(\mathbf{x};\mathbf{a})$ can approximate the pre-estimated motion vectors $\mathbf{d}(\mathbf{x}), \mathbf{x} \in \Lambda'$ well. This can be accomplished by minimizing the following fitting error:

$$E_{\text{fitting}} = \sum_{\mathbf{x}\in\Lambda'} w(\mathbf{x}) \left|\mathbf{d}(\mathbf{x};\mathbf{a}) - \mathbf{d}(\mathbf{x})\right|^p \tag{6.7.2}$$

As shown in Sec. 5.5.4, a global motion can usually be described or approximated by a polynomial function. In this case, $\mathbf{a}$ consists of the polynomial coefficients and $\mathbf{d}(\mathbf{x};\mathbf{a})$ is a linear function of $\mathbf{a}$, i.e., $\mathbf{d}(\mathbf{x};\mathbf{a}) = [\mathbf{A}(\mathbf{x})]\mathbf{a}$. If we choose $p = 2$, then the above minimization problem becomes a weighted least squares problem. By setting $\frac{\partial E_{\text{fitting}}}{\partial \mathbf{a}} = 0$, we obtain the following solution

$$\mathbf{a} = \left(\sum_{\mathbf{x}\in\Lambda'} w(\mathbf{x})[\mathbf{A}(\mathbf{x})]^T[\mathbf{A}(\mathbf{x})]\right)^{-1} \left(\sum_{\mathbf{x}\in\Lambda'} w(\mathbf{x})[\mathbf{A}(\mathbf{x})]^T\mathbf{d}(\mathbf{x})\right). \tag{6.7.3}$$

As an example, consider the affine motion model given in Eq. (5.5.16). The motion parameter vector is $\mathbf{a} = [a_0, a_1, a_2, b_0, b_1, b_2]^T$, and the matrix $[\mathbf{A}(\mathbf{x})]$ is

$$[\mathbf{A}(\mathbf{x})] = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix}.$$

In fact, the parameters for the x and y dimensions are not coupled and can be estimated separately, which will reduce the matrix sizes involved. For example, to estimate the x dimensional parameter $\mathbf{a}_x = [a_0, a_1, a_2]$, the associated matrix is $[\mathbf{A}_x(\mathbf{x})] = [1, x, y]$, and the solution is:

$$\mathbf{a}_x = \left(\sum_{\mathbf{x}\in\Lambda'} w(\mathbf{x})[\mathbf{A}_x(\mathbf{x})]^T[\mathbf{A}_x(\mathbf{x})]\right)^{-1} \left(\sum_{\mathbf{x}\in\Lambda'} w(\mathbf{x})[\mathbf{A}_x(\mathbf{x})]^T d_x(\mathbf{x})\right). \tag{6.7.4}$$

6.8 Region-Based Motion Estimation*

As already pointed out in the previous section, there are usually multiple types of motions in the imaged scene, which correspond to motions associated with different objects. By region-based motion estimation, we mean to segment the underlying image frame into multiple regions and estimate the motion parameters of each region. The segmentation should be such that a single parametric motion model can represent well the motion in each region. Obviously, region segmentation is dependent on the motion model used for characterizing each region. The simplest approach is to require each region undergo the same translational motion. This requirement however can result in too many small regions, because the 2-D motion