Elec4622 Laboratory Project 3, 2018 S2

David Taubman

October 15, 2018

1 Introduction

This is the second project, to be demonstrated and assessed within the regular scheduled laboratory session in Week 13. The project is worth a nominal 10 marks, but optional bonus marks are available.

In this project, you explore various aspects of block-based motion estimation and global motion estimation, including sub-pixel precision motion compensation and estimation of the reliability of block motion vectors. You should review the lecture notes on motion estimation and also download and study the materials entitled "Lab 5" on the class web-site, since these provide you with an introduction to block-based motion estimation.

The last hour of your scheduled lab/tut session in Week 12 has been set aside to allow you to get help from the lab demonstrators in understanding this project's objectives, and also understanding "Lab 5," but you should be sure to take a look at these ahead of time to make the most out of this opportunity.

2 Global Motion Estimation

In this project, you will be estimating a global translational motion vector $\mathbf{v}$ from individual block motion vectors $\mathbf{v}_b$, the subscript b here identifies individual blocks and runs from 1 to B, where B is the total number of blocks in the image. The estimation process involves to images (video frames) denoted here as $x[\mathbf{n}]$ and $y[\mathbf{n}]$, where $x[\mathbf{n}]$ is a source image and $y[\mathbf{n}]$ is the target image that will ultimately be approximated by motion compensation based on $x[\mathbf{n}]$. The target frame $y[\mathbf{n}]$ is partitioned into B blocks, denoted $y_b[\mathbf{n}]$, and a motion vector $\mathbf{v}_b$ is estimated from

$$\mathbf{v}_b = \underset{\mathbf{u}}{\operatorname{argmin}}\, J_b(\mathbf{u})$$

where

$$J_b(\mathbf{u}) = \sum_{\mathbf{n}\in b} |y[\mathbf{n}] - x[\mathbf{n}-\mathbf{u}]|^p$$

where $p = 1$ for the SAD metric and $p = 2$ for the MSE metric.

Based on the estimated block motion vectors $\mathbf{v}_b$, a global motion vector $\mathbf{v}$ can be found using a weighted least squares fitting procedure, as follows:

$$\mathbf{v} = \underset{\mathbf{u}}{\operatorname{argmin}}\, J_{\text{fit}}(\mathbf{u})$$

where

$$J_{\mathrm{fit}}\left(\mathbf{u}\right)=\sum_{b=1}^{B}w_b\cdot\left\|\mathbf{u}-\mathbf{v}_b\right\|^2=\sum_{b=1}^{B}w_b\left(u_x-v_{b,x}\right)^2+\sum_{b=1}^{B}w_b\left(u_y-v_{b,y}\right)^2$$

Here, the subscripts x and y denote the horizontal and vertical components of a motion vector. The solution to this minimization problem is easily found (e.g., by setting derivatives with respect to u_x and u_y to 0) to be

$$\mathbf{v}=\frac{\sum_{b=1}^{B}w_b\cdot\mathbf{v}_b}{\sum_{b=1}^{B}w_b}$$

That is, one only needs to take the weighted average of the block motion vectors.

An important question is how the weights should be selected. To that end, in the last task of this project you will use a reliability metric inspired by the Harris corner detector's figure of merit. Specifically, you will set

$$w_b=\frac{\det\left(\bar{\Gamma}_{\sigma,b}\right)}{\operatorname{trace}\left(\bar{\Gamma}_{\sigma,b}\right)}$$

where $\bar{\Gamma}_{\sigma,b}$ is a 2×2 matrix formed from aggregating local matrices $\Gamma_\sigma\left[\mathbf{n}\right]$ over the domain of block b, according to

$$\bar{\Gamma}_{\sigma,b}=\sum_{\mathbf{n}\in b}\Gamma_\sigma\left[\mathbf{n}\right]$$

and

$$\Gamma_\sigma\left[\mathbf{n}\right]=\nabla_\sigma\left[\mathbf{n}\right]\cdot\nabla_\sigma^t\left[\mathbf{n}\right]$$

Here, $\nabla_\sigma\left[\mathbf{n}\right]$ is a scale dependent gradient vector that could be obtained by applying derivative of Gaussian (DOG) filters to the target image $y\left[\mathbf{n}\right]$. However, for simplicity, in this project you will use a difference of Gaussian approach, setting

$$\nabla_\sigma\left[n_1,n_2\right]=\frac{1}{2}\begin{pmatrix}y_\sigma\left[n_1+1,n_2\right]-y_\sigma\left[n_1-1,n_2\right]\\ y_\sigma\left[n_1,n_2+1\right]-y_\sigma\left[n_1,n_2-1\right]\end{pmatrix}$$

where $y_\sigma\left[\mathbf{n}\right]$ is obtained by subjecting $y\left[\mathbf{n}\right]$ to a Gaussian low-pass filter with scale-parameter (standard deviation) σ.

You should make sure you understand why the weighting procedure described here should help in estimating more reliable global motion.

Global motion is only applicable to certain types of video content. In general, one should use a richer global motion model than pure translation – common model are the affine or perspective model, as described in your lecture notes. To simplify things for this project, we limit our attention to global translational motion, as described above, and test content suitable for exploring the solution will be uploaded to the class web-site for you to use in this project.

3 Tasks

Task 1: (4 marks) In your own time, complete the exercises in "Lab 5" – see the class web-site. In particular, you must complete the modifications to the "motion_example" workspace, which are required to produce a colour output image which simultaneously shows the target frame and the motion vectors. You should also be prepared to comment on (and demonstrate) the impact of different block sizes and search ranges, on motion compensated MSE, the motion vector field, **and the visual quality of the motion compensated output image**.

Task 2: (2 marks) Modify the code to fit a global translational motion vector $\mathbf{v}$ to the block motion vectors, as explained in Section 2, taking the weights w_b to be equal for all blocks b. The global vector $\mathbf{v}$ should have real-valued coordinates, not integer values, and your program should print the vector $\mathbf{v}$.

Task 3: (3 marks) Further modify the code to perform motion compensation using the global motion vector $\mathbf{v}$, rather than the individual estimated block motion vectors, reporting MSE and writing the motion compensated result out as an image. To do this, you will need to interpolate the original image samples, since $\mathbf{v}$ is not generally integer-valued, and for this purpose you should simply use **bilinear interpolation**. You can, if you like, continue to use the approach in "Lab 5," where individual blocks are separately motion compensated, except that you must use the global motion vector in every block, and use bi-linear interpolation. Equivalently, you can simply shift the original image by $\mathbf{v}$.

- Boundary extension is very important here. The code from "Lab 5" does not select motion vectors that will map a block beyond the boundaries of the source frame, but since a single global vector is being selected for all blocks, this constraint would force $\mathbf{v}$ to be $\mathbf{0}$, which is not interesting. It follows that your modified code will need to extend the original source image by the maximum value of any block motion vector.
- Use the **point-symmetric boundary extension** method covered in Problem Set 1, problem 9, rather than the symmetric extension method.
- In addition to reporting MSE associated with the global motion vector $\mathbf{v}$, your program should report the MSE associated with the originally estimated block motion vectors, so it will print out two MSE values, report the global motion vector $\mathbf{v}$ and write out the global motion compensated image for you to look at.
- Run the program on the two test sets provided on the web-site for evaluating global motion estimation.
- You will need to explore the use of different block sizes and be able to explain what happens as block size is changed.

Task 4: (1 mark) Modify the search criterion used in Task 3 to find motion vectors, so that the best vector is considered to be that which minimizes MSE over the block, rather than SAD. What impact does this have upon the two MSE values computed in Task 3 and the motion vector field? Can you find good explanations for your observations?

Task 5: (up to 4 bonus marks) As it stands, the global motion estimation strategy described above has a major weakness: it treats all block motion vectors as equally reliable. To address this, **modify your program from Task 3** to compute a reliability weight w_b for each block b following the method proposed in Section 2, which is based on the figure of merit that is used in the Harris corner detector. This method requires you to perform Gaussian filtering, with scale parameter σ, take local horizontal and vertical differences, compute the 2×2 matrices $\Gamma_\sigma[\mathbf{n}]$ and aggregate these over each block to form $\bar{\Gamma}_{\sigma,b}$, after which the ratio of determinant to rank can be computed and used as the weight w_b. Your program should then find the weighted average of the block motion vectors $\mathbf{v}_b$, using this as the global vector $\mathbf{v}$.

- If your implementation does not involve any Gaussian filtering, the maximum number of bonus marks you can receive is limited to 2.
- If your program does not allow σ to be supplied on the command-line (at least covering the range $1 \leq \sigma \leq 4$), the maximum number of bonus marks you can receive is limited to 3.
- Explore the improvements obtained in global motion compensated MSE when using this weighted scheme, in comparison to the unweighted approach of Task 3, for various block sizes and various values of σ. This is easiest to do, if your program accepts a command-line argument that can be used to disable the weighting process. This might be done, if you like, by specifying a meaningless value for σ, such as 0 or a negative value.

4 Assessment

You should not rely upon implementing this project within scheduled laboratory sessions. Also, remember that you cannot expect to get marked in the last hour of the laboratory session; you should arrive at the laboratory in Week 13 with a working implementation to demonstrate, so that you have time, if necessary to make modifications to your work and still get it assessed.

In order to obtain the full marks for any given task, you must have a working program to demonstrate and you must be able to explain how it works and answer questions on demand.

You may feel free to re-use code from previous laboratory sessions, so long as you understand it. You may also discuss the project with other students in the class, but your programs must otherwise be your own original work.

You are required to **create a zip file containing the source code for all of your programs in this project and submit this source code electronically before the end of the laboratory session in Week 13**. Instructions on how to submit your code will be provided to you in Week 13.