CS4551 Multimedia Software Systems (Spring 2018)

Homework4 (10%) - Block-based Motion Compensation

- Due: Electronic submission via CSNS by Saturday, 05/05/2018.
- What to turn in:
  - Submit ***source code only.*** You must submit all necessary files (except data files) for compile and run.
  - Do NOT submit data files.
  - You **MUST** provide a ***readme.txt*** file containing all information to help with the grading process.
- If your program produces any compile errors, you will receive 0 automatically no matter how close your program is to the solution.
- **Do not use any Java built-in image class methods, library, or tools to complete this homework.**
- You will receive penalty if you do not follow any of the given instructions.

0. What your program should do

Name your main application as *CS4551_[Your_Last_Name].java*.

<eg> On Command Prompt

```
java CS4551_Doe
```

Display the following main menu to the user.

```
Main Menu----------------------------------
1. Block-Based Motion Compensation
2. Removing Moving Objects
3. Quit

Please enter the task number [1-3]:
```

After performing the selected task, display the menu again.

Download the HW4 test dataset, IDB. IDB includes 200 PPM images (Walk_000.ppm ~ Walk_200.ppm). Before programming, check the images using Irfanview.

Download Sample Result.

1. Task1 - Block-based Motion Compensation (50pts)

Write a ***routine(method)*** that compensates motions between two images (one is the target image and the other one is the reference image). Receive two images as parameters. Make the routine perform the followings:

- Receive an integer **n** for the macro block size from the user. **n** must be 8, 16, or 24.
- Receive an integer **p** for search window from the user. **p** must be 4, 8, 12, or 16.
- Divide the target image into a set of **n**x**n** macro blocks.
- For each target macro block:
  - Estimate a motion vector by finding the best matched macro block (a.k.a. predicted block) in the reference image using **p** for the search area size and MSD (Mean Square Difference) for the matching criteria. When you compute MSD, use *Gray* values.

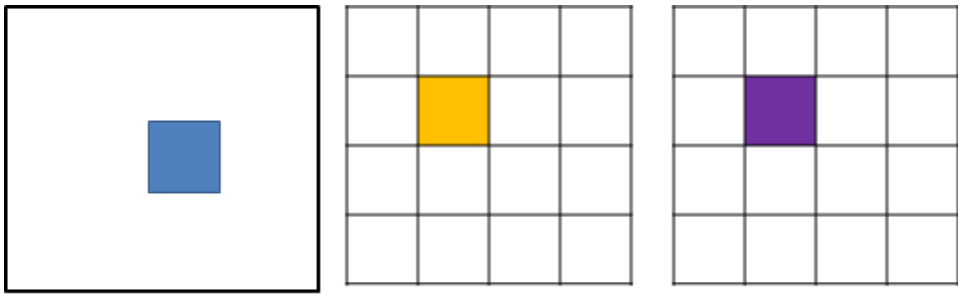
    *Gray = round(0.299 * R + 0.587 * G + 0.114 * B)*
  - After finding the best matched block, compute the motion vector using

d=(dx, dy)=(tx,ty)-(rx,ry)
where (tx,ty) is the target block location and
(rx,ry) is the best matched block location in the reference frame

Also, compute the error block (a.k.a. residual block) consisting of pixel differences (absolute values) between the target block and the best matched block (a.k.a predicted block).

error_pixel_value = | pixel_in_target_block – corresponding_pixel_in_the matched_block |

- Outputs
  - Display and Save your error (or residual) image. Your error image is a composition of all error blocks. Scale error values to range [0, 255]. In order to do so, compute MIN*error* and MAX*error* and map values from [MIN*error*, MAX*error*] to [0, 255].
  - Save the estimated motion vectors to the *mv.txt*. It should have motion vectors of all target blocks. Refer to the, refer sample *mv.txt* file.



Reference frame with the best matched block indicated in blue

Target frame with a target block indicated in yellow

Error frame with an error block indicated in purple

Extracredit (15pts): Implement matching in the half-pixel accuracy. Provide an additional menu option for this. Display and save motion vectors and a residual image.

Extracredit (30pts): Implement logarithmic search for matching. Provide an additional menu option for this. Display and save motion vectors and a residual image. Also, display the number of matches performed given **p**.

2. Task2 - Application - Moving Object Detection and Removal (50pts)

The block-based motion compensation can be used in applications other than compression. One example is to remove moving objects in the scene.

When you have a video sequence where the camera has not moved and the background is relatively static (like our dataset in IDB) but a few objects (e.g. person in our dataset) are moving, you can estimate which blocks were moved (*dynamic block*) and which blocks were not moved (*static block*) based on the estimated motion vector by the first task routine. By replacing the dynamic blocks with other static blocks, you can create a scene as though the moving object was never there.

Use IDB and implement the following steps to achieve this task:

- Select the target image.

For the target image, receive from the user a frame number *n* between 19 and 179. 19~179 frames in the data set include moving people.

- Identify dynamic blocks.
  Given the input *n*, use $(n-2)^{th}$ as the reference image automatically. Identify dynamic blocks in the target image by performing the block-based motion estimation and checking motion vector as you have done in the task1.
- Mark dynamic blocks:
  Mark dynamic blocks. For example, draw boundaries of dynamic blocks or composite dynamic blocks with red color. Display and save the result.
- Remove moving object from the target image. Apply two different methods and display two different result images:
  1. For each dynamic block *d*, find the closest static block *s* in the n^{th} frame and replace *d* with *s*. After processing all dynamic blocks, display and save the result.
  2. The 5^{th} frame of test data set does not include any moving objects. Utilize the 5^{th} frame to get static block *s*. For each dynamic block *d*, find the static block *s* from 5^{th} frame and replace *d* with *s*. After processing all dynamic blocks, display and save the result.

  See the sample outputs for the result format.