Homework #9

Due Date: 11:59pm Tuesday 27 May 2014

In this homework you will write a C++ program that can compress images using an H.261-like compression algorithm. Your program will read the image on `stdin` and outputs the resulting image on `stdout`, and get in the required operation on the command line.

Please present a report containing your answers as well as a zip file containing all your code.

1. [3 points] Implement the functions `compute_sad()` and `compute_motion_vector()` to compute the motion vector for the best matching block for a 16×16 macroblock (MB) in a target frame. The distance is the Sum of Absolute Differences (SAD), and the best motion vector is the pair (i, j) that minimize:

$$SAD(i,j)=\sum_{k=0}^{15}\sum_{l=0}^{15}\left|C(x+k,y+l)-R(x+i+k,y+j+l)\right|$$

   where R is the MB in the reference frame, C is the MB in the target frame, (x, y) is the top-left corner of the target block and i and j go from $\{-15 \dots 15\}$ i.e. we try all MBs in a window of $2p + 1 \times 2p + 1$ to find the one that minimizes the SAD distance. The default value for $\boldsymbol{p = 15}$.

   Try the function on the small images `msg1.pgm` and `msg2.pgm` to make sure it is working correctly.

2. [5 points] Implement the functions `compress_i_frame()` and `compress_p_frame()` to a frame using either Intra-frame or Inter-frame compression. The functions will compress and reconstruct the compressed frame, and will need to compute the number of bits required. You don't need to explicitly represent the *compressed* frame (e.g. represent the zig-zag coded coefficients or their run-length coding), just keep track of how many bits will be used after the entropy coding (Huffman Coding) of the DCT quantized coefficients. The I-frame compression is similar to the DCT compression homework, so feel free to use all your code from that.

   Notes:

   - The quantization used here is simpler than the JPEG or DCT homeworks. The step for the DC coefficient is 8:

$$QDCT=round\left(\frac{DCT}{8}\right)$$

     while the AC coefficients are quantized with a step depending on a *scale* factor:

$$QDCT=\left\lfloor\frac{DCT}{2\times scale}\right\rfloor$$

where *scale* is an integer in the range [1, 31]. Use a default value of ***scale* = 4**.

- The quantized values for each 8×8 block are run-length coded (RLC) in the same zig-zag order of the JPEG compression. Although the H.261 performs entropy coding with a fixed Huffman Tree, you will not need to perform entropy coding for this homework. Just use a **14-bit** fixed code for every (run, level) pair, where *run* is the number of consecutive zeros followed by a value of *level*. *run* uses 6 bits for values [0, 63] and *level* uses 8 bits to represent values [–128, 127].
- In the P-frame coding, compute the best motion vector in the reference frame. You will have three possibilities depending on the value of the SAD:
  - The SAD is too large, e.g. larger than 500, in which case the MB is coded as in I-frame coding i.e. no motion compensation is applied.
  - The SAD is too small, e.g. below 50, in which case the MB is assumed to equal its best matching MB from the reference frame i.e. just send the motion vector.
  - The SAD is between 50 and 500, in which case you will compute the difference between the target MB and its best match, pass this through the DCT, quantization, and entropy coding, and send the motion vector and the coded coefficients to the decoder.
- You will need to think about what information you need to encode for each MB e.g. its type (motion-comprensated inter-block with data, motion compensated inter-block without data, intra-block), DCT coefficients, … etc. See the code for more details.
- Although the H.261 standard uses Huffman coding with a fixed table for the motion vector (and also uses motion vector prediction), you will not need to perform prediction or entropy coding for this homework. Assume the motion vectors are coded with a **10-bit** fixed code, where 5 bits are used for each of the horizontal and vertical components, which take values in the range [–15, 15].

Try the function on the small images `msg1.pgm` and `msg2.pgm` to make sure it reconstructs a good approximation for them and correctly decodes them their compressed versions.

3. [4 points] Implement the function `compress_movie()` to compress a set of frames using an H.261-like compression algorithm. Try the function on the included *Suzie* movie movie, using a standard setting of $p = 15$, $N = 16$, $M = 3$, and *scale* = 4. Compute the number of bits required for the compressed movie, and form a movie of the reconstructed movie (using the reconstructed frames) using the `ffmpeg` program:

```
# install ffmpeg
sudo apt-get install ffmpeg

# put the image files image-*.pgm into a movie with frame
# rate 12 fps
ffmpeg -r 12 -i suzie-hat-%03d.pgm -vcodec png suzie-hat.avi
```

4. [2 points] Try your movie compression algorithm with different parameter settings for:
    1. The number of I- and P-frames e.g. M = 0 (no inter-frame compression i.e. JPEG-like compression for each block), 1 (1 P-frame between each two I-frames), 2, 3, 4, and 5.
    2. The *scale* factor for the quantization e.g. *scale* = 1, 2, 4, 8, 16.

    Comment on the output file size (number of bits required) and the subjective quality of the reconstructed movie for each setting.

Command Line

You need to modify the main file `hw09.cpp` to include the required functionality. Your program should be named `hw09`, and should be called as follows:

- To compute the motion vector for a given MB in the target frame on stdout:

  ```
  cat r.pgm t.pgm | ./hw09 -compute_mv p N x y
  ```

  where *p* is search window size, *N* is the size of the macroblock, (*x*, *y*) is the top-left corner of the MB in the target frame, and the reference (r.pgm) and target (t.pgm) frames are passed to stdin. For example, to compute the motion vector for a MB of size 2×2 in msg2.pgm starting at (4, 5) to the best matching in msg1.pgm with $p = 2$, you could run:

  ```
  cat msg1.pgm msg2.pgm | ./hw09 -compute_mv 2 2 4 5
  ```

- To compress a frame using I- or P-frame compression and output the reconstructed frame on stdout:

  ```
  cat r.pgm t.pgm | ./hw09 -compress t scale
  ```

  where *t* is the type (*I* or *P*) is search window size, and *scale* is the quantization scale. For example, to compress the frame suzie-003.pgm using I-frame compression, you could run:

  ```
  cat suzie-003.pgm | ./hw09 -compress I
  ```

  and to compress the frame suzie-003.pgm using P-frame compression with the reference frame suzie-001.pgm and scale = 4, you could run:

  ```
  cat suzie-001.pgm suzie-003.pgm | ./hw09 -compress P 4
  ```

- To compress a movie and output the reconstructed frames, you could run:

  ```
  ./hw09 -compress_movie inframes outframes NF M
  ```

  where *inframes* is the format for the input frames, *inframes* is the format for the output frames, *NF* is the number of frames, and *M* is the number of P-frames between each two I-frames. The output reconstructed frames are written in the sameFor example, to compress the *suzie* movie with $M = 3$, you could run:

  ```
  ./hw09 -compress_movie suzie-%03d.pgm suzie-hat-%03d.pgm 150 3
  ```

Instructions

- All code should be implemented in C++ under Linux.
- Please submit your homework in one zip file named as follows: *CMPN206.HW##.FirstName.LastName.zip*, so for example if your name is Mohamed Aly and this is homework #1, then the file name should be *CMPN206.HW01.Mohamed.Aly.zip*.
- Please include all your code and sample output in the zip file, with a README file to explain what you did. Failure to follow these instructions will cause deductions from your grade.
- You are allowed to discuss the problems among yourselves. However, **copying** any part of the code will result a grade of **ZERO**. No exceptions.

Grading

- 14 points: requirements above
- 1 point: submission instructions