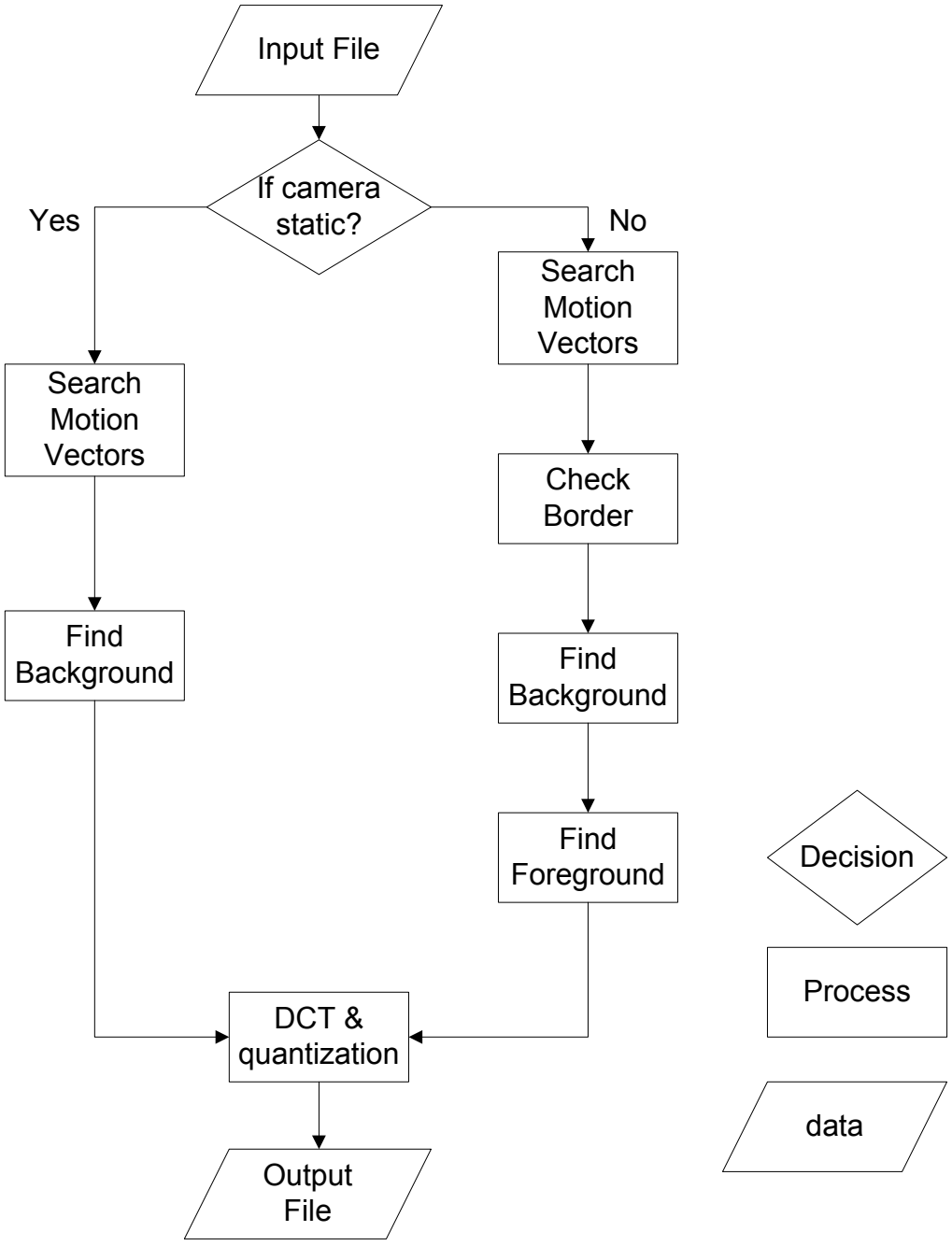
CSCI 576 Project

Wei-Ming Liu

The flow chart of the processes used in the project:



The processes used:

Check if the camera is static:

Do motion vector searching on the last frame, using the first frame as target. If more than half of the blocks have (0,0) as their vectors, then I conclude the camera is static since most blocks don't move from beginning to the end.

Check if there is a border:

Check if there is a border. Check only when the camera is moving. Do motion vector searching on the last frame, using the first frame as target. If all the blocks on a certain row (or column) have (0,0) as their motion vectors, then we can conclude this row is border. Since we know the camera is moving, it's very unlikely all the blocks on a row don't move at all, unless the row is border.

Grouping algorithm:

First I have to explain our grouping algorithm. It uses disjoint sets algorithm. At the beginning I create sets for each block. Each set has only one block. Then I try to union every two adjacent sets, if these two sets have similar motion vectors. (A motion vector of a set is the average of all motion vectors of the blocks in the set) Let's say set 1 has motion vector (x1, y1) and set 2 has (x2, y2). These two sets will be unioned if $|x1-x2| < threshold\_x$ and $|y1-y2| < threshold\_y$. After running this group algorithm, it will end up with several groups. Blocks in the same group will have similar motion vectors, and contiguous.

Find Background:

First group blocks using low threshold. Then I check if there exists a big group which contains many "boundary blocks". If there is, the motion vector of this group is used to represent the motion vector of the whole background. Any blocks which have similar motion vectors to background motion vector are then marked as background block.

Since some foreground blocks may be mistakenly marked as background block, I used some mechanism to solve this problem. First I check if there is any stand-alone background block. (Stand-alone background block is a block which has no other background blocks in its 3x3 neighborhood) Stand-alone background blocks are marked as foreground blocks. I do this check to foreground blocks too.

Then I check if there is any temporally discontinuous foreground block. If a block is foreground, then there must be some foreground blocks at the same area in previous frame, or next frame. If I find a foreground block which violates this rule, I mark it as background.

Find Foreground:

The goal of this process is to make the result even better. At the beginning of this process, I group blocks using high threshold. (The grouping algorithm here ignores blocks which have been marked as background) Here I use high threshold because the motion vectors of a foreground object is not as consistent as background. For example, when a person is moving, his head and body may have slightly different motion vectors, even both are moving at the same direction.

After the grouping, I check if there exist some big groups most blocks of which are in “central area”. There may be several foreground groups. I use these groups as foreground, and mark all the other blocks as background.

Then I use the same process to eliminate stand-alone background blocks and temporally discontinuous foreground blocks.

DCT & quantization:

After I get foreground and background information, I do DCT and quantize the coefficients differently based on which layer they belongs to. Each encoder uses different compression scheme. MyEncoder420 uses 4:2:0 subsampling. MyEncoder420z uses 4:2:0 subsampling and zigzag ordering. MyEncoderMC uses motion compensation, compressed data are kept for I-frames only, P-frames will be reconstructed using motion vectors.

Extra Credit:

I used zip to compress the output of several different encoders. Here is a comparison table of the compressed file size.

	Raw data	Only DCT & quantization	4:2:0 subsampling	4:2:0 + zigzag	Motion Compensation
Moving_camera 2 n1 = 0, n2 = 4	14767KB	6088KB	5135KB	5028KB	1716KB
Moving_camera 3 n1 = 0, n2 = 4	17811KB	9332KB	6522KB	6459KB	2190KB
Static_camera3 n1 = 0, n2 = 4	16088KB	6334KB	4598KB	4379KB	1471KB

Conclusion:

The zigzag ordering doesn't seem to have much improvement. My reason is that I used a uniform quantization table. If I use an uneven quantization table, then high-band coefficients will be quantized more and have a longer run of 0s.

Motion compensation improved the compression very much. Actually we didn't use error image here. If the error image is added, the size of compressed files will be bigger.

From the table, we can see that each compression scheme: 4:2:0 subsampling, zigzag ordering, and motion compensation, does have some improvement to the compression ratio.