Facial Recognition Using Motion Vectors: H.264

Project by: Yan Meng, Sriharsha Madala.

I. Introduction:

Autonomous control of Pan-Tilt-Zoom (PTZ) camera is an interesting field of research with numerous applications such as recording lectures, seminars, live coverage of sports, video surveillance, etc. The work done in this project is preliminary in that regard. This work targets a more simplistic scenario of recording a live lecture, where the subject has constrained motion. A PTZ camera records the subject live and the h.264 encoder generates the encoded bit stream in real-time. Traditional facial recognition algorithms can be used to track the subject but they fail in some scenarios such as the subject facing away from the camera or a poster in the background. The objective of this project is to assist traditional Facial recognition algorithms to improve their accuracy by simply using the motion vectors available in the bit stream. We proposed an algorithm to track multiple subjects with motion in each frame. Assuming the scenario of recoding a lecture, this subject is the face of the lecturer. This algorithm is generic in the sense that this can easily be extended to other applications suggested earlier.

In section II, we give an overview of one of the successful video coding standards- MPEG-2. In section III, we give an overview of H.264. In this section, we mainly describe the improvements suggested in H.264 over MPEG-2 and how they contribute to improvement in compression efficiency. In section IV, we describe our algorithm for facial tracking using motion vectors and demonstrate its working using some sample frames. In section V, we conclude by suggesting other applications for our technique.

II. Overview of MPEG-2:

Video is essentially a series of pictures captured at a constant frame rate. Video compression is a means to exploit spatial and temporal redundancies to minimize the number of bits required to store the video. Compression can be lossy or lossless, where the former is more preferred in practice. Several video compression standards have been proposed, MPEG-2, H.264, HEVC to name a few.

Let us first understand each of the components of a video. Video consists of a series of pictures. A fixed number of pictures are grouped together called Group Of Pictures (GOP). Each 16x16 pixel segment in a picture forms the basic coding unit of MPEG. This unit is called “Macroblock”. A contiguous group of macroblocks is called a “Slice”. Slices are useful for the decoder to handle errors. In case of errors, the decoder skips the current slice and jumps to the

next slice to proceed with the decoding. “Block” is consists of 8X8 pixels and is the smallest coding unit. It can be a Luminance (Y) or Chrominance (Cr, Cb) block. Depending upon the type of YUV format, we have varied number of blocks per each macroblock. For example 4:2:0 has 4 luminance and 2 chrominance blocks per each macroblock.

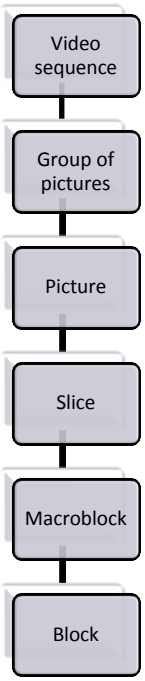

Fig.1: Hierarchy of components of a video sequence.

MPEG-2 standard defines three types of pictures Intra (I), Predicted (P) and Bidirectional (B). I-pictures are coded only using the information in that picture. P-pictures are coded by taking the nearest (forward direction) I or P- picture as a reference picture. P-pictures use motion compensation and hence provide higher compression efficiency. B-pictures use both forward and backward prediction and hence have the highest compression efficiency. Each GOP contains exactly I-picture. The number of B and P-pictures is a design choice, which has several consequences. For example $\mathrm{M} = 3, \mathrm{N} = 12$ gives a GOP structure, IBBPBBPBBPBB. M is the distance between two anchor pictures (I or P) and N is the distance between two I-pictures. Having more B-pictures increases the time to encode the video, while also improving the compression efficiency. Having a larger N, decreases the ability to randomly access the video (I-pictures provide random access points). Important thing to note is that the display order of frames is not the same as the sequence stored in the video.

Intra coding does not involve motion compensation. It relies on spatial redundancies to achieve good compression efficiencies. The basic compression paradigm is described by the following block diagram. MPEG-2 uses Discrete Cosine Transform (DCT) on each 8X8 block. The resulting coefficients are quantized according to a predefined quantization matrix that sets quantization parameter for each frequency coefficient. Higher frequencies are coarser than lower frequencies as human perception is less sensitive to high frequencies. Variable length codes are used to compress these quantized DCT coefficients. Run length coding, followed by Huffman coding is the most common choice. A zigzag scan is performed to generate long runs of zeros and hence fewer bits to represent each DCT coefficient.

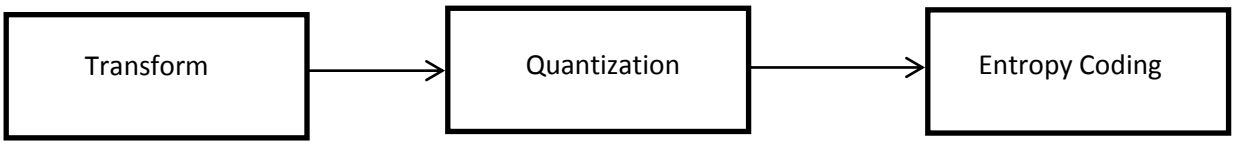

Fig.2: Fundamental Transform coding paradigm.

A P-picture is encoded using a reference picture (I or P). Much of the compression is due to the temporal redundancy. Each 16x16 macroblock in a P-picture is predicted from the reference picture and is represented by a motion vector(mv_x, mv_y). This implies that the 16x16 block from the current macroblock is the closest match in terms of distortion, which is the mean square error (MSE) calculation. Exhaustive search restricted by search_width, search_height is used to calculate motion vectors. More advanced searching algorithms are available in H.264. The difference between original macroblock and the predicted macroblock is call prediction error which is then encoded using DCT-quantization-entropy coding, similar to intra coding.

Bidirectional prediction varies slightly as each macroblock can use either forward prediction or backward prediction or both. When it uses both, it takes the average of past and future predicted macroblocks. Advanced concepts such as Rate control [7] and Adaptive quantization require extensive study, which is beyond the objective of this project.

III. Overview of H.264

In this section, we study the main improvements in H.264 over MPEG-2. Broadly speaking improvements were made in Transform coding, Entropy coding, Motion compensation and Profiles.

DCT over 8X8 blocks is chosen as the transform tool for both intra coding and coding the motion prediction error. Although DCT has the advantages of being very close to KKT and has fast implementation algorithms (FFT) [5], it has some undesirable properties. DCT transforms integer-valued images into real-valued coefficients which require floating point arithmetic. Hence a new integer transform is introduced in H.264 and it operates on 4x4 blocks [3] [4]. This does not require floating point operations and exact reconstruction at the decoder is possible.

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

Another improvement in intra coding of H.264 is that it has 9 directions for spatial prediction [4]. This increases the coding efficiency at the small cost of increased complexity. Two modes of entropy coding are used in H.264, Context Adaptive Variable Length Coding (CAVLC) and Context Adaptive Binary Arithmetic Coding (CABAC). CAVLC is very similar to MPEG-2 except out of the 4 VLC tables available, depending upon the statistics of the adjacent blocks, one is chosen. On the other hand, CABAC achieves significant compression efficiencies and has high computational complexity [3]. The adaptive arithmetic code changes as the input

changes. It is also possible to allocate non-integer number of bits to symbols, thus minimizing the bit rate adaptively.

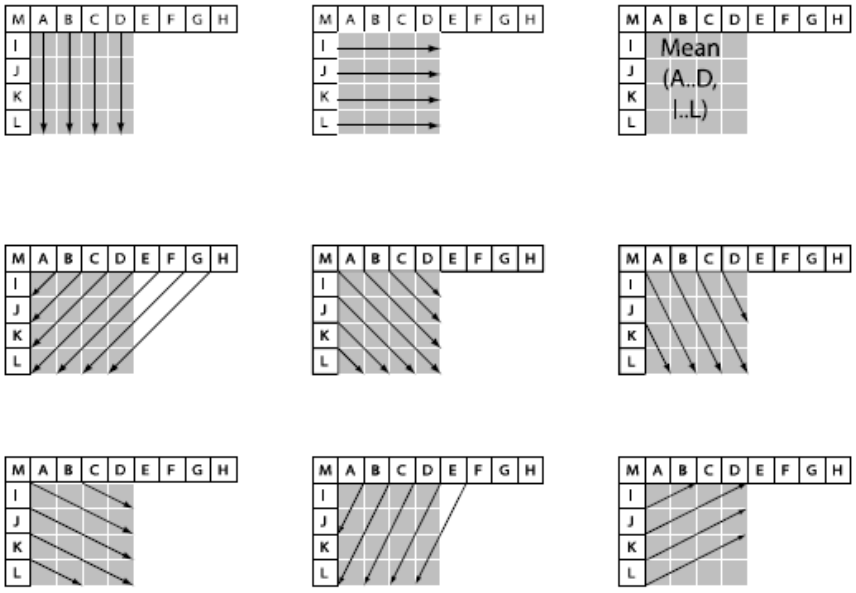

Fig.3: The nine modes of spatial prediction available in H.264

Improvements made in the motion compensation can be briefly summarized into few aspects:

- Increased number of macroblock partition types

  Allowing the choice of different macroblock partition sizes increases the compression efficiency. For a monotonous background, a 16x16 macroblock with no partitions and for more detailed foreground 4x4 partitions allows minimum distortion for the same number of bits. H.264 offers macroblock partitions of sizes 8x8, 16x8, 8x16 and 16x16 luma samples and further sub-macroblock partitions of sizes 8x8, 8x4, 4x8 or 4×4 luma samples.
- Multiple picture prediction

  H.264 allows a maximum of 15 pictures to be used as reference pictures for prediction [3]. This is particularly useful in situations similar to scene cuts.
- Quarter pixel resolution of motion vectors

  Quarter pixel resolution provides better PSNR than the half pixel resolution used in the MPEG-2 [4]. The Quarter pixel interpolated motion compensation is achieved in two stages, 1) half pixel predictions generated from a 6-tap filter, followed by, 2) linear averaging with adjacent pixels.
- Deblocking filter

Block based Intra coding and motion estimation typically result in "blocking artifacts" [4]. Hence H.264 defines an adaptive in-loop deblocking filter. For a detailed study please refer to [6].

MPEG-2 has defines 6 profiles (Simple, Main, 4:2:2 profile, SNR, Spatial and High) and 4 levels (High, High 1440, Main and low). Profile gives the type of coding tools used (B-pictures, entropy coding type). Level gives the range of coding parameters (bit rate, resolution). H.264 defines three profiles, Baseline, Main and Extended that can be chosen to suit different applications.

Motion estimation contributes significantly to the encoding complexity of any video compression standard. The following are a subset of various searching algorithms available [2]:

- ***Full search***

  All the macroblocks within the search range of $\pm W$ are searched to minimize the prediction error. The criterion of error can be Sum of Absolute Differences (SAD) or Sum of Squared Differences (SSD). If we have N reference pictures and M block types then a total of $M * N * (2W + 1)^2$ search candidates. This is computationally intensive and the following search algorithms were studied to decrease its complexity.

- ***Fast Full Search (FFS) (default)***

  A lower bound is derived for SAD in terms of Successive Elimination Algorithm (SEA) as follows:

$$SAD(f_c, f_r) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}|f_c(i,j) - f_r(i,j)| \geq \left|\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} f_c(i,j) - \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} f_r(i,j)\right| = SEA(f_c, f_r)$$

  Computation of SEA is much simpler than SAD and its parallelizable. If SEA is greater than the current minimum, we can skip that block and proceed to the next one. It is also possible to use the SEA of each 4x4 block to compute the SEA of larger blocks. Thus in essence, we have a FFS algorithm with lower complexity without any loss in performance.

- ***Simplified Hexagon Search***

  Similar to a diamond search, it has a large hexagon pattern of radius 2 and a smaller hexagon pattern of radius 1. Compute the distortion at all points. If the center point has the minimum distortion, repeat the process for the smaller hexagon pattern. Otherwise, move the center of the large hexagon pattern to the point that has the minimum distortion and continue.

- ***Enhanced Predictive Zonal Search (EPZS)***

  The EPZS, similar to many predictive algorithms has three main steps [1].

  1. Predictor selection: This is the most important step of any predictive algorithm. The accuracy of this directly affects both the performance and the computational complexity of this algorithm. The most commonly considered set of predictors are the "Median

Predictor", the three spatially adjacent MVs (left, top and top-right) and the collocated MV in the reference frame. It is not necessary to consider each predictor as will be described in the early termination step.

2. Adaptive early termination: A threshold $T_k$is chosen based on the SAD of the three adjacent blocks, the collocated blocks and a predetermined a_k and b_k as follows: $T_k = a_k * \min\{MSAD_1, MSAD_2, \dots, MSAD_n\} + b_k$.
   If the any of the predictors from the chosen set has a SAD smaller than the threshold, we can stop the searching and choose that predictor as the motion vector. Else, we continue to step-3 to refine our best predictor.
3. Prediction refinement: If the early termination criteria are not met, motion estimation is further refined by using Diamond/Hexagonal search located around the center of the best predictor among the set defined in step-1.

IV. Algorithm for Facial Recognition

The main application we are targeting is the PTZ camera recording a live person speaking, tracking his/her face in real-time. Hence the inherent assumptions here are that the majority of the motion in any frame is contributed by the face and there are no other kinds of motion in the foreground or background.

We use the motion vector information of each partition of a macroblock available in the encoded bit stream to identify the moving face. The trivial way to do this is to identify all the non-zero motion vectors in a particular frame. If our assumption is in fact true that only the face moves in any frame, we have successfully identified it. But there are two deviations commonly observed in video sequences.

- Firstly, there may be stray motion in the frame, which means some macro blocks may have non-zero motion vectors although that part of the frame does not have significant motion.
- Secondly, there can be other parts of the frame that have more motion than the face, for example hands, legs etc.

Hence we proposed an algorithm to handle both these conflicts and identify the face of the subject with reasonable accuracy. The algorithm is as follows:

I. We create a hash table such that 4x4 sub-macroblock partitions, whose norm is in a specified range, will fall in the same bucket. This represents a histogram of motion in the frame.

II. Next we identify the buckets with highest (N_1) and second highest (N_2) number of 4x4 sub-macroblock partitions. If $N_1 - N_2 < 15 * \frac{N_1}{100}$ and the norm of the second

highest bucket is smaller than that of the highest bucket, we choose the lower norm value of the second highest bucket as our threshold. Otherwise we choose the lower norm value of the highest bucket.

III. From the motion vectors of each 4x4 block, we identify how the 16x16 macroblock is partitioned. This is done by defining a forest for the macroblock where each 4x4 block is a node and an edge exists between adjacent (spatially) nodes if they have the same motion vector. By running a Breadth First Search (BFS) algorithm on each disconnected graph of this forest, we get the top-left and the bottom-right pixel of each macroblock partition.

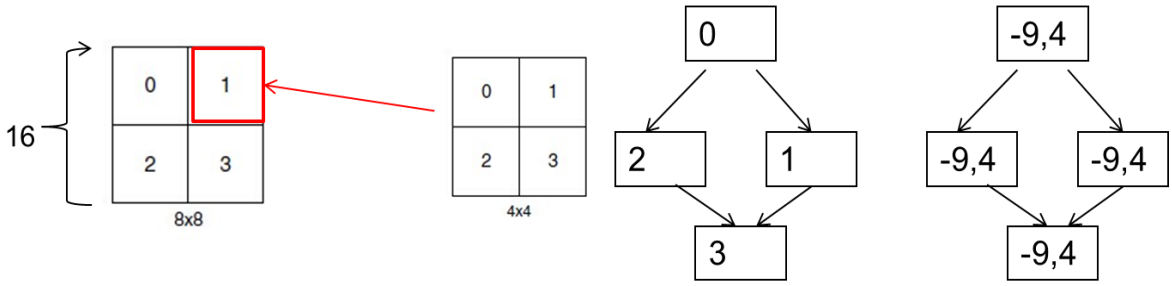

Fig.4: Identifying macroblock partitions using BFS on each 4x4 block.

IV. We define a connection between two macroblock partitions as having at least one pixel overlap. We then run the strongly connected component algorithm on all the partitions exceeding the threshold calculated in the previous step. This results in multiple disjoint groups of partitions, where each group has spatially connected partitions (at least one pixel). The largest such group represents the dominant object with motion in the frame, around which we draw a box.

In the encoded bit stream, we have motion vectors of each 4x4 block of each macroblock of each frame. Each macroblock can be partitioned into several types as described in section III. One of the challenges is to identify the type of partition, which is handled in the step-III of our algorithm. Steps-I, II calculate a dynamic threshold for the norm of motion vectors. Step-III identifies macroblock partitions that are greater than the threshold using BFS. Step IV identifies the largest group of partitions that are connected with motion greater than the threshold from step II.

To see how this algorithm minimizes the above two deviations, we first analyze step-II. Implementing a static threshold is not feasible since the magnitudes of the norm of motion vectors vary from frame to frame. Hence the dynamic threshold is chosen such that a large number of partitions within the same bucket must belong to the object of our interest (the

face). But due to the fixed hash table we are using, it could happen the partitions are slit across two different buckets. Hence the 15% check. This minimizes stray motion in the frame.

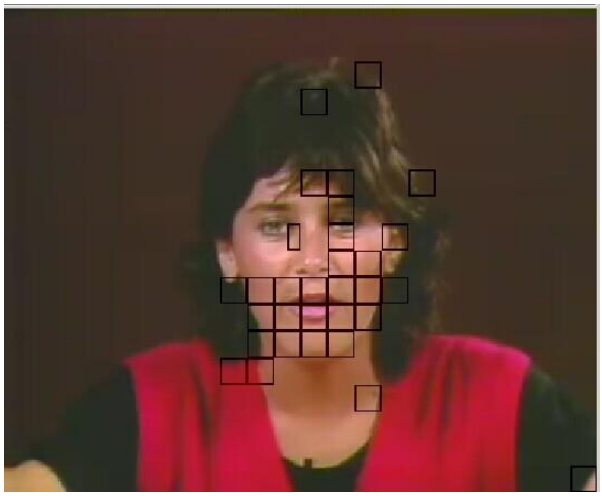

Fig.5: All the macroblock partitions in the sample frame, greater than the dynamic threshold.

How do we handle multiple objects moving in the same frame? We iterate that our algorithm only works when the face is the prominent object (occupying significant portion of the frame). We define an edge/connection exists between two macroblock partitions if they have at least one common pixel between them. This generates a forest of partitions. By running our version of strongest connected components algorithm on this forest, we get the disconnected sets of partitions. The largest such set represents the “prominent object” in the frame whose motion is larger than our dynamic threshold.

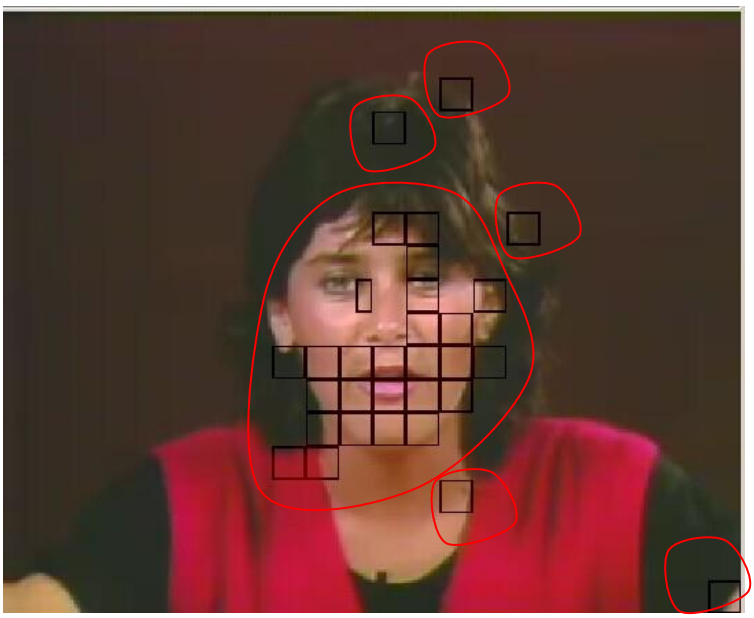

Fig.6: Each set of partitions on the same sample frame is marked by a red boundary after running SCC.

Fig.7: The result of drawing big box around the largest set of partitions.

V. Future Work

In this project, we gave an overview of the MPEG-2 and H.264 video compression standards. Then we analyzed the particular problem of facial recognition in the context of real-time video recording of a subject. We proposed an algorithm that takes into account the non-idealities of general video sequences, which have stray motion and multiple motions in the frame. This algorithm can easily be extended to more complicated case of tracking the motion of multiple subjects.

As stated this algorithm is mainly designed to complement the performance of traditional facial recognition algorithms. Further work needs to be done to create such a hybrid scheme. Another interesting area of application for our algorithm is the autonomous live sports coverage. This is a particularly challenging task as there are many motions in each frame and separating the motion of the target subject from all other motion is hard. Video surveillance is also another interesting application. It is to be noted that none of these applications require any fundamental changes to the proposed algorithm but simply some refinements in thresholds and the number of subjects to track. Extending this work to a HEVC encoded bit stream is another avenue for future research.

VI. References

[1] A.M.Tourapis, "Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation", *in proceedings of Visual Communications and Image Processing 2002 (VCIP-2002),* pp. 1069-79, San Jose, CA, Jan'02.

[2] Liyin,Xie, Su Xiuqin, and Zhang Shun, "A review of motion estimation algorithms for video compression". *Computer Application and System Modeling (ICCASM), 2010 International Conference on*.Vol. 2.IEEE, 2010.

[3] Michael Igarta, *"A study of MPEG-2 and H.264 video coding*", MS. Thesis, Purdue University, U.S.A, (2004).

[4] Wiegand, Thomas, et al. "Overview of the H. 264/AVC video coding standard". *Circuits and Systems for Video Technology, IEEE Transactions on* 13.7 (2003): 560-576.

[5] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform", IEEE Trans. Signal Process., Vol. 40, September 1992.

[6] P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol.13, pp. 614–619, July 2003.

[7] Westerink, Peter H., Rajesh Rajagopalan, and Cesar A. Gonzales. "Two-pass MPEG-2 variable-bit-rate encoding." *IBM Journal of Research and Development*43.4 (1999): 471-488.