

HASH-AIDED MOTION ESTIMATION

Alwin Anbu, Argyris Zymnis

Stanford University
Electrical Engineering Department

{anbu, azymnis}@stanford.edu

ABSTRACT

In this project we apply low-rate visual hash codes to low-complexity video encoding. The visual hash code consists of a short bit-string extracted from an image block (8x8 pixels) by projecting that block over orthonormal basis vectors. We use this hash in conjunction with a distributed video codec that encodes video frames independently but decodes them conditionally, given the previously decoded frames as side-information. Specifically, by transmitting the hash code of an image-block in the current frame as side information, the decoder can coarsely determine motion between the current and previous frames. The decoder can then motion-compensate the side-information frames such that they are good estimates of the current frame. We show through Matlab simulations that DCT is a good orthonormal transform (better than identity and DFT) and a few (10~15) low frequency components need to be quantized (uniformly) and sent over to the decoder to get good motion estimation for low-rate applications.

1. INTRODUCTION

The use of motion compensation (MC) is a standard feature of most state-of-the-art video encoders. It gives improved rate-distortion performance compared to other methods such as conditional replenishment [1]. MC is a computationally intensive task and it is usually performed at the encoder rather than at the decoder. This is because a video sequence is encoded once while it is decoded many times and thus performing MC at the encoder helps reduce decoder complexity and computational requirements.

In some cases however, it might be desirable to reduce the encoder complexity as much as possible, while at the same time making the decoder more complex. This would be the case for example, if we wanted to use a low-power webcam as the encoder, while the decoder consists of a computer, which has a lot more computational power. Another example could be sensor networks where the sensors that collect video data are extremely limited in complexity

and battery life [2]. In these cases we will need to perform MC at the decoder.

Recently developed schemes, under the name of distributed video coding, aim at doing exactly this. Motion estimation is performed by using the previously decoded frame as side information, as well as a low-rate hash sent by the encoder as helper information.

The aim of this project is to investigate the performance of a number of different hashes that could be used to aid motion estimation at the decoder.

A lot of earlier work exists for distributed video coding. More recently, in [3], it was shown that hashes generated by projecting the image blocks onto randomly generated low frequency patterns can give good rate-distortion performance at low rates. Also in [4], it was shown that a Wyner Ziv (distributed video) encoder that protects the video waveform rather than the compressed bit stream achieves graceful degradation under deteriorating channel conditions without a layered signal representation.

2. PROBLEM STATEMENT

Assume that the current frame to be encoded is X and the previous frame, which is available at the decoder, is Y . The current frame is composed by blocks, which we will call X_{ij} . The encoder transmits a hash T_{ij} for every block in the current frame, where $T_{ij} = t(X_{ij})$ and $t(\cdot)$ is some transformation. The decoder then uses the hash of the current block T_{ij} to find the block in the previous frame Z_{ij} which is “closest” to X_{ij} , using some sort of mean squared error criterion. Suppose there are N pixels per block. The distortion for the current frame is then defined as:

$$D = \sum_i \sum_j \frac{\|X_{ij} - Z_{ij}\|^2}{N} \quad (1)$$

The work at the encoder and the decoder is illustrated in Fig. 1

The purpose of this project is to find suitable transformations $t(\cdot)$, together with a set of motion estimation rules

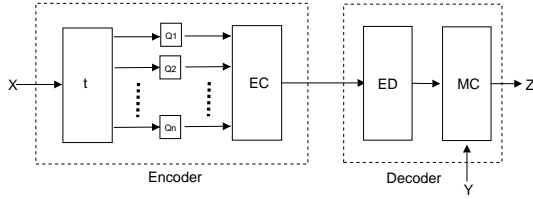


Fig. 1. Block Diagram for Distributed Video Encoding System.

and entropy coding methods, such that the average distortion over all the transmitted frames is minimized. Obviously if the transformation is the identity, the distortion will be minimized, but the rate will be very large. We will show that there are transformations that beat the identity in a rate-distortion sense.

In order to compare the different hash generation schemes, we used the same training set in all cases, which consists of the first 50 frames of the *bus*, *carphone*, *foreman* and *mobile* video sequences. A block size of 8×8 was used for all cases.

3. METHOD

A generic hash-aided motion estimation system consists of some sort of linear transformation, followed by quantisation to generate the hash. The hash is then entropy coded and transmitted to the decoder, which uses some sort of decision rule to perform the motion estimation.

3.1. Linear Transformation

For this part of the system we considered three different transformations: the identity, the DCT and the DFT.

The identity is the simplest linear transformation used and is an obvious initial choice. The hash in this case just consists of the block pixels, which are subsequently quantised and entropy coded. We considered two cases here, namely transmitting the whole block and sub-sampling the block by 2 in each dimension.

The DCT is the most widely used linear block transform, since it has energy concentration properties which are very close to the KLT (the best possible linear block transform). For this reason the DCT is a good linear transform to use as part of this problem. The DCT concentrates most of the block energy in a relatively small amount of low-frequency transform coefficients.

When using the DCT as the linear transformation used to generate the hash, we need to decide on which coefficients to keep and which to discard. We considered two separate possibilities. The first one is to keep a certain number n of low-frequency coefficients and discard the rest. The first coefficient we keep is the DC coefficient and we then

proceed from this coefficient in a zig-zag manner to find the rest low-frequency coefficients. This is motivated by the fact that most of the block energy is contained at the low frequencies and also the findings in [3], where it was shown that two blocks that differ in pixel domain also differ in their low frequency components.

The other possibility we considered was to keep the n highest energy coefficients, which are not necessarily the lowest frequency ones, and set the rest to zero. We thought that this would be a good idea, since the highest energy coefficients are the ones which are going to be most useful for motion estimation. In each case we tried a few different values for n .

We also tried using 2D DFT as the transform $t(x)$. In this case, we chose to use the D.C coefficient and six complex A.C coefficients, giving a hash size of 13. The selection of the A.C coefficients is done from the low frequency region as shown in Fig. 2. Using the conjugate symmetry property of the DFT, six more A.C coefficients are obtained at the decoder. We then do a IDFT at the decoder to get a pixel domain representation of the block. This pixel block is then used for block matching operations.

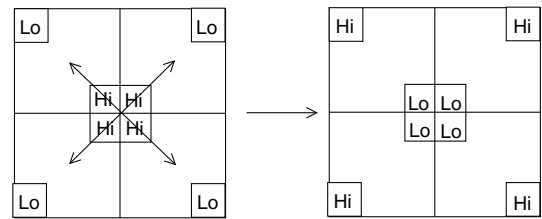


Fig. 2. Coefficient Selection for the DFT.

3.2. Quantisation

The output of the linear transformation is then quantised so as to decrease the rate of the hash.

The simplest option in this case is to use the same uniform quantiser for all the coefficients within the hash. This is actually the only possibility if we are using the identity as our linear transformation. By then varying this single parameter we can obtain points on a rate-distortion curve to evaluate the performance of a specific method. This was the method of choice for most cases in this project.

A refinement that was considered over the previous simple method was to divide the coefficients into a small number of groups and then use a different uniform quantiser for each group of coefficients. The set of different quantisation widths can then be optimized over the training set, such as to minimize a cost function J :

$$J = D + \lambda \times R \quad (2)$$

where D is the average distortion, R is the rate and λ is a Lagrangian multiplier. It is thought that this method will produce better results than just using the same uniform quantiser for all the coefficients. Lack of time however did not permit us to investigate this option. A good method for heuristic local optimization is needed to investigate this approach, such as *taboo search* [5] or *simulated annealing* [6]. We did not however have the MATLAB code to implement such a method, nor did we have the time or the knowledge to write it ourselves.

Another approach that we tried was to use LMQ without any rate constraint. The optimum levels were calculated based on the training set data consisting of all the frames of the four video sequences mentioned before.

3.3. Entropy Coding

In this project, we were more concerned with investigating the properties of different linear transforms, when used for the generation of a hash, rather than actually developing a working motion estimation system. For this reason, we did not design a real entropy coding scheme, but used entropy calculations to find the hash rate instead.

Two different entropy calculation techniques were considered. The first consists of joint encoding of each of the hash coefficients. Thus to calculate the rate, we compute the hash for each block in the training set. We then calculate the probability mass function of each coefficient within the hash and we use this pmf to compute the entropy of each coefficient. The rate of the hash (in bits per pixel) is thus the sum of the individual entropies of each hash coefficient, divided by the number of pixels in the block (in our case 64).

The second method of entropy calculation that we considered was particularly suited to the DCT. It consists of zig-zag scanning of the DCT coefficients, followed by the formation of the zero run-amplitude symbols [r a] for the AC coefficients, using the special symbol [0 0] to denote the end of block (EOB). We then compute the pmf of the run-amplitude symbols and use it to compute the entropy. We then find the average number of [r a] pairs per block and multiply it with the entropy to get the rate of the AC coefficients. Note that in this case the DC coefficient is encoded separately (since the statistics for this coefficient are very different than for the AC coefficients). Thus the rate in bits per pixel is the rate due to the AC coefficients plus the rate due to the DC coefficient.

Run-amplitude coding is the best method we can use if we want to use the highest energy DCT coefficients rather than low-frequency DCT coefficients. This is because otherwise, we would have to also transmit the positions of the coefficients as well.

One extra feature we decided to incorporate into the motion-estimation system was a previous-frame hash store. This was used as follows: If a given block in the current

frame has the same hash as the collocated block on the previous frame, then we should use this collocated block for motion estimation and we should just transmit a single bit to signal this. This places an extra overhead of 1 bit per block, but can potentially reduce the rate for the same distortion.

3.4. Motion Estimation Decision Rule

We decided to use a search window of 16×16 in the neighborhood of each block, in which to calculate the motion vector. We also decided to use integer pixel motion estimation. As for the decision rule, we decided to look at two choices:

First of all, if the hash transformation is invertible (as is the case with the DCT), we can invert it to get an approximation of the current block. We can then use the normal rule of minimizing the sum of square differences to compute the motion estimated block in the previous frame. The advantage of this method is that it is too computationally intensive, since we only need to invert the hash once per block.

On the other hand, instead of inverting the hash transform, we can compute the hash of each block within the search window and try to find the block which minimizes the sum of square differences in the transformed domain. When computing the sum of squares in this case, we should only do so for the relevant coefficients, i.e. those that were originally in the hash. What is meant by this is that if our hash consists of the first 12 low-frequency coefficients, then we should only use these first 12 coefficients to calculate the sum of squared differences.

This method has the advantage that only the relevant coefficients are considered for motion estimation. It is however more computationally intensive than the previous method, as we have to compute the hash for each block within the search window.

4. RESULTS

4.1. Identity transformation

Figure 3, shows the rate-distortion performance of this system when using the actual block as our hash, where the pixels are quantised by a uniform quantiser. The first thing we can see in this case is that we can only use this hash at large rates. At the same time, the PSNR that can be achieved using this hash is very low, substantially lower than what we can get using the DCT as our transformation.

We found out that using a sub-sampled version of the block as our hash, results in even worse rate-distortion performance. For this reason we decided not to include the results of sub-sampling in this report.

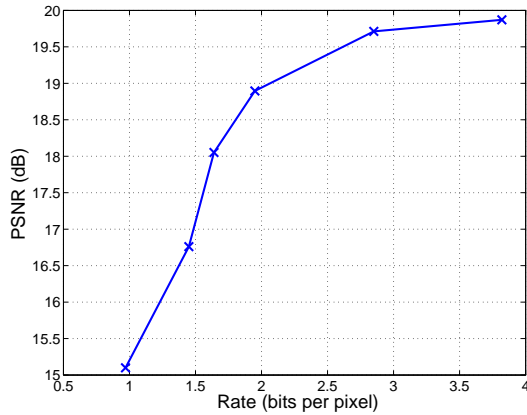


Fig. 3. Rate-distortion performance for identity.

4.2. Use of the DCT

The next linear transform we used was the DCT. We first investigated the performance of the system when using different numbers of quantised low-frequency coefficients as our hash.

Figure 4 shows the rate distortion performance when using the first 5, 10 and 15 low-frequency DCT coefficients as our hash. In this case, we are using the same uniform quantiser for all coefficients, run-amplitude coding for rate calculation. The rule used for motion estimation is to compute the hash for each block within the search window and only consider the relevant low-frequency DCT coefficients when computing the sum of squared differences.

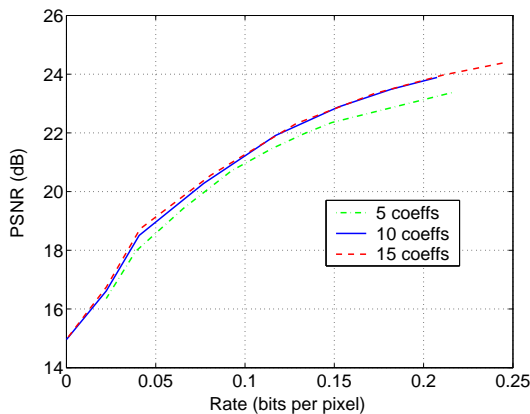


Fig. 4. Rate-distortion performance when using the Low-frequency DCT coefficients.

It can be seen that using more coefficients improves rate-distortion performance. However, we do not get large gains when using more than 10 coefficients. This is because the uniform quantiser used is so coarse that most of the higher frequency coefficients will be quantised to 0 anyway. Us-

ing a larger hash should however improve performance in higher rates. We decided to finally use the first 12 low-frequency coefficients, as this seems to be a good balance between performance and complexity of the hash generation system.

One important thing to notice is that the rate-distortion performance when using the DCT is much better than when using the identity, even at substantially lower rates. This suggests that the DCT is a good transformation to use as the first stage of generating a hash.

We also considered the case of using the 12 highest energy coefficients rather than the 12 lowest frequency ones. This was expected to produce better results. However, the rate-distortion performance of the highest energy hash turned out to be very similar and in fact slightly inferior to the lowest frequency one.

We believe that the main reason this is happening is that if we use the highest energy coefficients, then the pmf of the runlength-amplitude combinations becomes more uniform. Thus any gains in PSNR due to using this scheme are counterbalanced by an increase in rate. For this reason we decided not to make use of this option.

It was also found that the use of LMQ does not improve the performance of the system. For this reason we decided not to incorporate it in our design.

Finally, we also incorporated a hash store in the motion estimation system. This improves the rate-distortion performance at low rates, but does not have that much of an impact at higher rates.

The rate distortion curve for the overall system over the whole training set is shown in figure 5. It can be seen that using the DCT as the linear transform is a much better choice than using the identity. Note that the PSNR flattens out at high rates. This is due to the fact that in this rate the limiting factor is the number of DCT coefficients used.

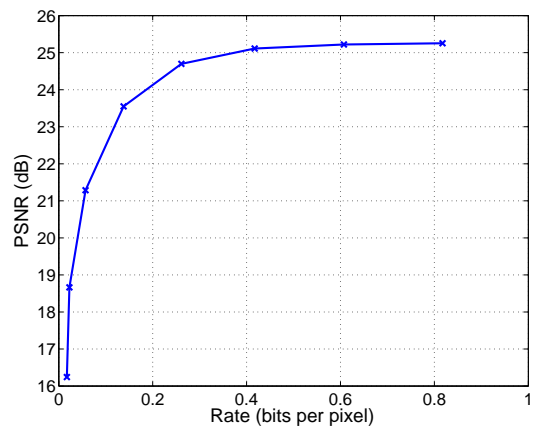


Fig. 5. Rate-distortion performance for proposed system over training set.

4.3. Low Frequency DFT coefficients

We also tried using the DFT as a linear transform. In this scheme, 8 low-frequency DFT coefficients were used as the hash, quantising both their real and imaginary parts by the same uniform quantiser. The IDFT was then performed at the decoder to get an estimate of the current block.

It was found however that the performance of the DFT is inferior to that of the DCT. This could be due to the fact that the DCT has better energy concentration properties, or maybe our choice of coefficients was not the best possible.

4.4. Comparison with different Method

Figure 6 shows the comparison of our proposed method, i.e using the first 12 low-frequency coefficients, with the method suggested in [3], by Shantanu Rane. This was done for the first 50 frames of the *foreman* sequence.

It can be seen that our proposed method performs better in the range of rates from 0.03-0.2 bits per pixel, however it performs worse in lower rates. This could be due to the fact that we are using the same uniform quantisers for all the coefficients. By properly optimizing the quantiser widths, we could probably achieve a better rate-distortion performance at lower rates.

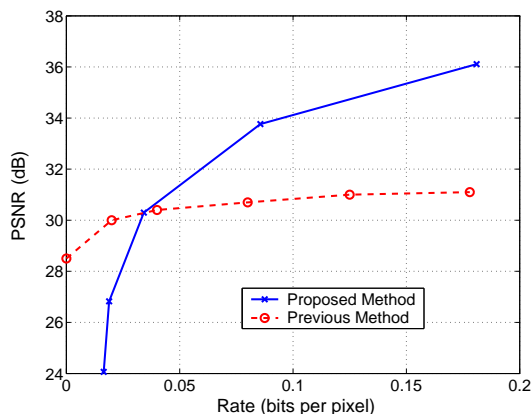


Fig. 6. Comparison of proposed method with previous method, first 50 frames of *foreman*.

5. CONCLUSIONS

Hash-aided motion estimation is an integral part of a distributed coding system. In our investigation we found out that the use of about 12 quantized low-frequency DCT coefficients as a hash, results in better rate-distortion performance than using the actual pixel values. The DCT seems to perform better than other linear transforms such as the DFT, or the method proposed in [3].

Throughout this investigation, we have used the same uniform quantiser for all DCT coefficients. An obvious improvement over this technique would be to optimize the quantiser widths for each coefficient (or group of coefficients) in order to get a better rate-distortion performance.

6. CONTRIBUTIONS OF EACH MEMBER

The work load was split at about half between the two group members.

Alwin Anbu designed the motion estimation code and looked into the use of the DFT as a linear transform. He also ran a large number of simulations once the system was ready. Argyris Zymnis designed the code for rate calculation, including the runlength-amplitude coder. He also modified the system to include a hash store.

We would like to thank Rajiv Agarwal for his valuable help during this project, even though he was not enrolled in axess for the course. Rajiv looked into the use of LMQ and EC-LMQ for quantisation and also offered his advice in terms of organisation and style.

7. REFERENCES

- [1] B. Girod, "EE398 Class Notes," 2005. [Online]. Available: www.stanford.edu/class/ee398/handouts.htm
- [2] T. Yoo, E. Setton, X. Zhu, A. Goldsmith, and B. Girod, "Cross-layer design for video streaming over wireless ad hoc networks," in *IEEE International Workshop on Multimedia Signal Processing*, Siena, Italy, Sep 2003.
- [3] S. Rane, "Hash-aided motion estimation and rate control for distributed video coding," *EE 392J Digital Video Processing Course Project*, 2004.
- [4] B. Girod, A. Aaron, S. Rane., and D. Rebollo-Monedero, "Distributed video coding," in *Proc. IEEE, Special Issue on Advances in Video Coding and Delivery*, 2003.
- [5] F. Glover and M. Laguna, "Tabu search," in *Modern Heuristic Techniques for Combinatorial Problems*. C. R. Reeves (Ed.), John Wiley and Sons, 1996, pp. 70–150.
- [6] D. G. S. Kirkpatrick and M. P. Vecchi, "Optimization by simulated annealing," *Science*, pp. 220–671, 1983.