VIDEO COMPRESSION USING SPARSE PROJECTIONS AND PRE-TRAINED ORTHONORMAL BASES

A Project Report

submitted by

RAHUL NOOLU

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

AND

MASTER OF TECHNOLOGY

under the guidance of

Dr. Shankar Balachandran

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

JUNE 2013

THESIS CERTIFICATE

This is to certify that the thesis titled **"VIDEO COMPRESSION USING SPARSE PROJECTIONS AND PRE-TRAINED ORTHONORMAL BASES",** submitted by **RAHUL NOOLU**, to the Indian Institute of Technology Madras, Chennai for the award of the degree of **Master of Technology and and Bachelor of Technology**, is bonafide record of research work done by him under my supervision. The contents of the this thesis, in full or parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Shankar Balachandran

Project Guide

Assistant Professor

Dept. of Computer Science and Engineering

IIT-Madras, Chennai-600036

Place: Chennai

Date: 22nd June, 2013

ACKNOWLEDGEMENTS

I would first like to express my great appreciation to my project guide Dr. Shankar Balachandran, from the Dept. of CSE, for his guidance and mentoring throughout the course of my project. I am deeply indebted to him for the completion of this project.

I also owe thanks to my wing-mates and friends, who have made my insti life much more fun and enjoyable. I would also like to thank my classmates for helping me academically, allowing me to complete all my courses successfully. Special mention to my project-mate Tushar, who has helped me a lot during the course of this project.

Last but not the least, I owe a huge chunk of thanks to my parents, grand-parents and both my uncles, for their constant show of support during my whole life. Specially my parents, who have sacrificed so much for me at every stage. For it's the faith that they showed in me, I could successfully clear JEE and get into IITM.

ABSTRACT

Video compression is a process of efficiently coding digital video to reduce the number of bits required in representing video frames. Its purpose is to reduce the storage space and transmission cost while maintaining good quality. This project primarily deals with video compression based on a patch-wise, SVD-like algorithm. Unlike the usual SVD method to compress data (using low-rank approximations), we test a new method which uses sparse projections and exemplar bases to code individual frames of a video. We have also discussed about how changing various parameters affect the resulting video quality and storage.

Contents

List of Tables

List of Figures

ABBREVIATIONS

SVD	Singular Value Decomposition
PSNR	Peak signal-to-noise ratio
RPP	Average number of bits per pixel

Chapter 1

Introduction

1.1 Scope of the Project

There have been many techniques to compress images using Singular Value Decomposition in the past, similarly for the case of video compression. In this project we have tried to approach video compression in a new way. Instead of using low-rank approximations of the SVD, the new method uses a fixed number of full-rank orthonormal bases but with sparse projection matrices. This has its own advantages over low-rank approximations. We have taken video data, separated them into a set of images, and then tested them using the new method, also observing the storage versus the quality.

1.2 Overview and Organization of the report

Chapter 2 summarizes the related previous work done in this area, explanation of various terms in the project.

Chapter 3 explains in detail how the video data is processed, the different

parameters that are used, and how the new frames are tested.

Chapter 4 presents the results of the tests we have done, and the conclusions based on the result.

Chapter 2

Background and Related Work

2.1 Singular Value Decomposition

2.1.1 What is SVD?

The singular value decomposition (SVD) is a popular matrix factorization that has been used widely in applications ever since an efficient algorithm for its computation was developed in the 1970s. In recent years, the SVD has become even more prominent due to a surge in applications and increased computational memory and speed.

Nowadays, SVD is of great significance, and is used in many applications such as low rank approximation, image compression, estimation & inversion, pseudo inverse, principal component analysis etc.

The theory behind SVD is that a given matrix A of size $m \times n$ and rank r can be written as,

$$A = U \times S \times V^T \tag{2.1}$$

where U is a $m \times m$ orthogonal matrix, S is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and V^T (the conjugate transpose of V) is an $n \times n$ orthogonal matrix

Matrix S is a diagonal matrix having only r nonzero entries, which makes the effective dimensions of these three matrices $m \times r$, $r \times r$ and $r \times n$ respectively.

U and V are two orthogonal matrices and S is a diagonal matrix, called the singular matrix.

The diagonal entries s_i of S are known as the singular values of A and have the following properties:

- $s_i > 0$
- $s_1 \geq s_2 \geq s_3 \geq \ldots \geq s_r$

The m columns of U and the n columns of V are called the left-singular vectors and right-singular vectors of A, respectively.

2.1.2 Low-rank matrix approximation.

SVD has an important property that makes it particularly interesting for our application. SVD provides the best low-rank linear approximation of the origi-

nal matrix A.

It is possible to retain only $k << r$ singular values by discarding other entries. We term this reduced matrix S_k. Since the entries in S are sorted i.e., $s_1 \geq s_2 \geq s_3 \geq \ldots \geq s_r$, the reduction process is performed by retaining the first k singular values.

The matrices U and V are also reduced to produce matrices U_k and V_k , respectively. The matrix U_k is produced by removing $(r-k)$ columns from the matrix U and matrix V_k is produced by removing $(r-k)$ rows from the matrix V .

When we multiply these three reduced matrices, we obtain a matrix A_k. The reconstructed matrix:

$$A_k = U_k \times S_k \times V_k^T \tag{2.2}$$

is a rank-k matrix that is the closest approximation to the original matrix A.

2.1.3 Application to Image Compression

An image can be defined as a two-dimension function $f(x, y)$ (2D image), where x and y are spatial coordinates, and the amplitude of f at any pair of (x, y) is gray level of the image at that point. For example, a gray level image can be represented as : f_{ij} where $f_{ij} \equiv f(x_i, y_i)$

We know that "the rank of matrix A is equal to the number of its non-zero singular values". In many applications, the singular values of a matrix decrease quickly with increasing rank. This property allows us to reduce the noise or compress the matrix data by eliminating the small singular values or the higher

ranks.

When an image is SVD transformed, it is not compressed, but the data take a form in which the first singular value has higher amount of the image information. With this, we can use only a few singular values to represent the image with little differences from the original.

To illustrate the SVD image compression process :

$$A = USV^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{2.3}$$

That is, A can be represented by the outer product expansion :

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \ldots + \sigma_r u_r v_r^T \tag{2.4}$$

When compressing the image, the sum is not performed to the very last singular values, the singular values with small enough values are dropped. (Remember that the singular values are ordered on the diagonal.) The closet matrix of rank k is obtained by truncating those sums after the first k terms:

$$A_k = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \ldots + \sigma_k u_k v_k^T \tag{2.5}$$

The total storage for A_k will be $k(m + n + 1)$.

The integer k can be chosen confidently less than n, and the image corresponding to A_k will still be very close the original image. However, choosing

different k will have a different corresponding image and storage for it.

2.2 Image Compression for Set of Images : Sparse projections onto Orthonormal Exemplar bases

2.2.1 Introduction

In the paper [1] by Ajit Rajwade *et al,* they have presented a new technique for compact image representation. This forms an integral part of our encoding system.

If we consider the earlier techniques for matrix-based representation for a set of images, a single pair of orthonormal bases (U, V) is learned from a set of some N images, and each image I_j, $1 < j < N$ is represented by means of a projection S_j onto these bases, i.e. in the form $I_j = US_jV^T$.

It is quite intuitive to observe that, as compared to an entire image, a small image patch is a simpler, more local entity, and hence can be more accurately represented by means of a smaller number of bases. Following this intuition, regarding an image as a set of matrices (one per image patch) instead of using a single matrix for the entire image is much accurate.

Furthermore, there usually exists a great deal of similarity between a large number of patches in one image or across several images of a similar kind. We can exploit this fact to learn a small number of full-sized orthonormal bases (as opposed to a single set of low-rank bases) to reconstruct a set of patches from a training set by means of sparse projections with least possible error.

This change of focus from learning lower-rank bases to learning full-rank bases but with sparse projection matrices, brings with itself several significant theoretical and practical advantages. This is because it is much easier to adjust the sparsity of a projection matrix in order to meet a certain reconstruction error threshold than to adjust for its rank.

2.2.2 Theory

Let $P \in R^{m1 \times m2}$ be an image patch. Using singular value decomposition (SVD), we can represent P as a combination of orthonormal bases $U \in R^{m1 \times m1}$ and $V \in R^{m2 \times m2}$ in the form $P = USV^T$, where $S \in R^{m1 \times m2}$ is a diagonal matrix of singular values.

However P can also be represented as a combination of any set of orthonormal bases $\bar{U}$ and $\bar{V}$, different from those obtained from the SVD of P. In this case, we have $P = \bar{U}S\bar{V^T}$ where S turns out to be a non-diagonal matrix. Contemporary SVD-based compression methods leverage the fact that the SVD provides the best low rank approximation to a matrix. We choose to depart from this notion, and instead answer the following question:

What sparse matrix $W \in R^{m1 \times m2}$ will reconstruct P from a pair of orthonormal bases $\bar{U}$ and $\bar{V}$ with the least error $\left\|P - \bar{U}W\bar{V^T}\right\|^2$? Sparsity is quantified by an upper bound T , i.e. the number of non-zero elements in W (denoted as $\|W\|_0$). The optimal W with this sparsity constraint is obtained by nullifying the least (in absolute value) $m_1 m_2 - T$ elements of the estimated projection matrix $P = \bar{U}W\bar{V^T}$ as proven in [1].

2.2.3 Learning the Bases

The essence of this method lies in a learning method to produce K exemplar orthonormal bases $\{(U_a, V_a)\}$, $1 \leq a \leq K$, to encode a training set of N image patches $P_i \in R^{m_1 \times m_2} (1 \leq i \leq N)$ with least possible error (in the sense of root mean square error between original and reconstructed patches). Note that $K \ll N$. In addition, a sparsity constraint is imposed that every S_{ia} (the matrix used to reconstruct P_i from (U_a, V_a)) has at most T non-zero elements. The main objective function to be minimized is:

$$E(\{U_a, V_a, S_{ia}, M_{ia}\}) = \sum_{i=1}^{N} \sum_{a=1}^{M} M_{ia} \left\| P_i - U_a S_{ia} V_a^T \right\|^2 \tag{2.6}$$

subject to the constraints that $U_a^T U_a = V_a^T V_a = I$, $\forall a$, $\|Sia\|_0 \leq T$, $\forall (i, a)$ and $\sum_a M_{ia} = 1$, $\forall i$. Here M_{ia} is a binary matrix of size $N \times K$ which indicates whether the i^{th} patch belongs to the space defined by (U_a, V_a).

2.2.4 Application

Our framework is geared towards compact but low error patch reconstruction. We are not concerned with the discriminating assignment of a specific kind of patches to a specific exemplar, quite unlike in a clustering or classification application. In our method, after the optimization, each training patch Pi $(1 \leq i \leq N)$ gets represented as a projection onto one out of the K exemplar orthonormal bases, which produces the least reconstruction error, i.e. the k^{th} exemplar is chosen if $\left\| P_i - U_k S_{ik} V_k^T \right\|^2 \leq \left\| P_i - U_a S_{ia} V_a^T \right\|^2$, $\forall a \in \{1, 2, ..., K\}$, $1 \leq k \leq K$. For patch P_i, we denote the corresponding 'optimal' projection matrix as $S_i^* = S_{ik}$, and the corresponding exemplar as $(U_i^*, V_i^*) = (U_k, V_k)$.

Thus the entire training set is approximated by:

- The common set of basis-pairs $\{(Ua, Va)\}$, $1 \leq a \leq K$ ($K \ll N$), and
- The optimal sparse projection matrices S_i^* for each patch, with at most T non-zero elements each.

The overall storage per image is thus greatly reduced . Furthermore, these bases $\{(Ua, Va)\}$ can now be used to encode patches from a new set of images that are somewhat similar to the ones existing in the training set.

Chapter 3

Video Compression

3.1 Application to video compression

In the paper [1], they have used the method of exemplar bases and sparse projections to compress face databases successfully. The idea behind the method is to use reference images to train the exemplar bases, and then use these bases to represent new similar images.

Now, if we consider a the individual frames in a video, there is a lot of similarity among the frames. Using this redundancy, we can use the first frame as the reference image to train the orthonormal bases. Using these bases we encode the rest of the frames (using the projections generated from the bases).

Our project is based on this idea to compress a video, by using the initial frames as the reference, training the orthonormal bases and compressing the rest of the frames using these bases.

The results for this part of the project are discussed in Chapter 4.

3.2 Working of the encoder

The following flowchart summarizes the working of our encoder :

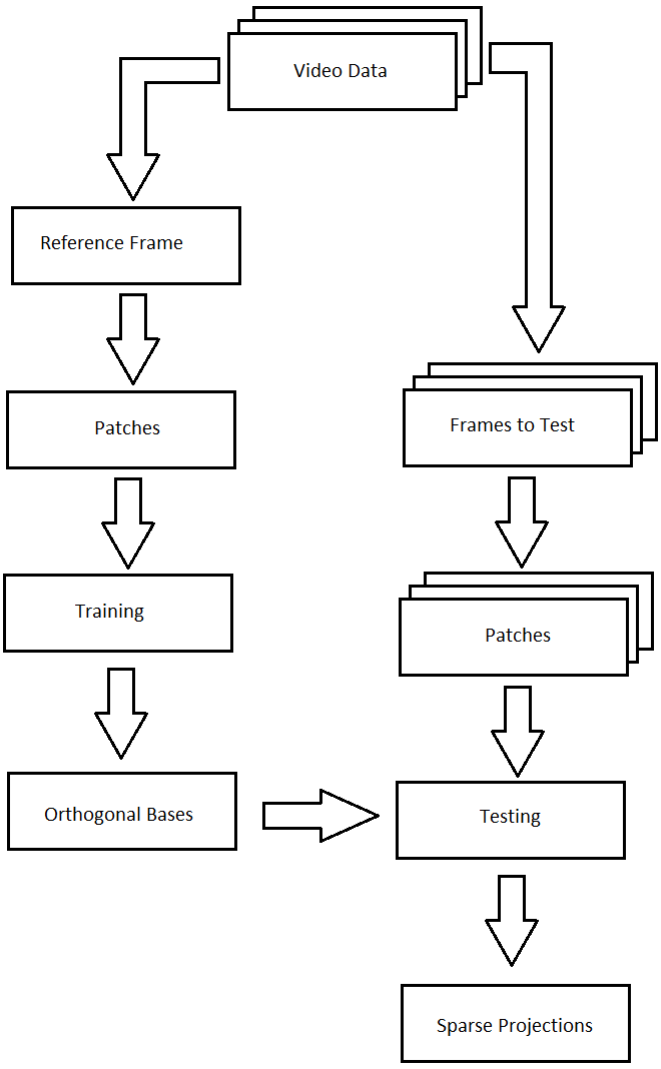

Figure 3.1: Working of the encoder

3.2.1 Splitting the Video into frames.

The video to be coded is taken and the frames were generated by taking an image at every tick. For each frame, we divide the entries by the maximum pixel value such that the image intensity values range between [0,1].

3.2.2 Training the bases

Next step is to train the exemplar bases using a training image. This was done using the first frame of the video as the training image.

The training image is taken and is divided into N equal sized patches of size $m_1 \times m_2$. All of these N patches are now sent to train K orthonormal bases, where each patch can be matched to one of the K bases. The other parameter to supply during training is T, the number of non-zero elements.

The different parameters to set are:

- The patch size ($m_1 \times m_2$). If the size of the patch is too large, it becomes more and more unlikely that a fixed number of bases will serve as to generate a low-error sparse representation for the rest of the testing frames. Furthermore, if the patch size is too large, the algorithm will lose out on the advantages of locality and simplicity. However, if the patch size is too small, there is very little a compression algorithm can do in terms of lowering the number of bits required to store these patches.

- The number of orthonormal bases (K). We have observed that the choice of the value K did not matter much to accuracy of the results, since each patch is matched to one of the K bases. The only downside of of a higher K value is that the computation time increases.

- Number of non-zero elements in each patch (T). A very high value of T generates bases which over-fit the training frame, and a very low value of T gives a larger error.

The K bases are generated using the method given in [1], using equation (6). Each base is pair of orthonormal matrix $\{U_a, V_a\}$,$1 \leq a \leq K$.

3.2.3 Testing the new frames

Once the bases are generated (using parameters K, T and $m_1 \times m_2$), we can now use them to test the rest of the frames.

Now we divide the frames into patches of size $m_1 \times m_2$. Let us denote each test patch as P_i, using the exemplar bases $\{U_a, V_a\}$, we generate the projection matrices S_{ia}.

$$S_{ia} = U_a^T P_i V_a \tag{3.1}$$

The corresponding base$\{U_i^*, V_i^*\}$for patch P_i is chosen such that it produces the sparsest projection S_i.

The sparsity of S_i is further increased by nullifying the smallest elements, such that maximum average per-pixel reconstruction error won't go over δ, giving us S_i^*.

Where δ is given by:

$$\frac{\left\| P_i - U_i^* S_i^* V_i^{*T} \right\|^2}{m_1 m_2} \tag{3.2}$$

3.2.4 Storage

Once we have generated the optimal sparse projection, the next step is storage of the information.

The following information must be stored for each test patch of a frame :

- The values of each non-zero element in the $\{S_i^*\}$ matrices over the whole patch. The integer part requires an average of some Q_1 bits per entry. The fractional part is quantized with Q_2 bits per entry.

- The location of each non-zero element is encoded using a_1 bits.

- The index of the matched base $\{U_i^*, V_i^*\}$ is encoded using a_2 bits.

- The number of non-zero elements encoded using $a_3$bits.

So the average storage(no. of bits) per-pixel for an image is given by:

$$RPP = \frac{N(a_2 + a_3) + NT_{eff}(a_1 + Q_1 + Q_2)}{M_1 M_2} \tag{3.3}$$

where N is the number of patches per image,
T_{eff} is the average number of non-zero elements per patch,
and $M_1 M_2$ is the total number of pixels in the image.

Chapter 4

Results and Conclusions

4.1 Results

4.1.1 Introduction

The quality of the output was measured using Peak signal-to-noise ratio of individual frames and the storage is given by the RPP.

PSNR is calculated using the following expression:

$$10 \log_{10} \frac{N m_1 m_2}{\sum_{i=1}^{N} \left\| P_i - U_i^* S_i^* V_i^{*T} \right\|^2} \tag{4.1}$$

RPP is calculated using the expression given in (13).

We have tested our method on two gray scale videos:

- FOREMAN Sequence (4:3 | 300 frames | 176 × 144).
- MOTHER AND DAUGHTER Sequence (4:3 | 300 frames | 176 × 144).

We have tested both the videos on the basis of :

- PSNR vs RPP for different patch sizes ($m_1 \times m_2$).
- PSNR vs RPP for the number of fractional bits (Q_2).

Finally, we have tabulated the compression ratio (compared to original size) compared with the video quality.

4.1.2 Patch Sizes and T values

The following graphs depicts the variation of PSNR vs RPP values for different patch sizes. We have tested for the following patch sizes:

- 8 x 8
- 12 x 12
- 15 x 15
- 20 x 20

The aim in this test is to determine the best patch-size for various videos. A better PSNR for same bits per-pixel (RPP) generally points us towards the best value for patch size.

4.1.2.1 FOREMAN : PSNR vs RPP, for different patch sizes

Following are the plots for PSNR vs RPP for different patch sizes for the FOREMAN video.

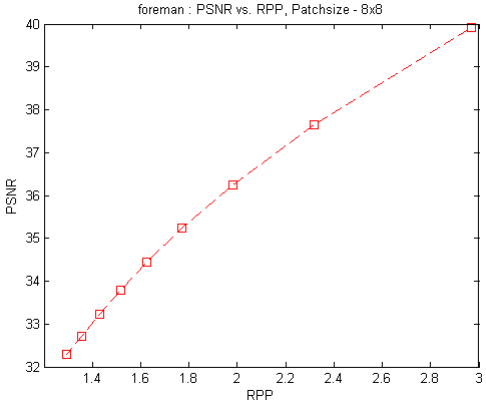

Figure 4.1: Foreman:PSNR vs RPP, Patchsize-8x8

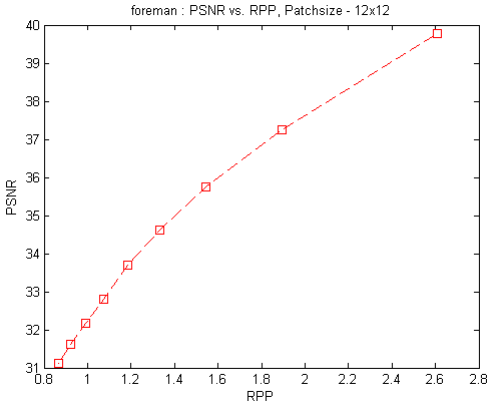

Figure 4.2: Foreman:PSNR vs RPP, Patchsize-12x12

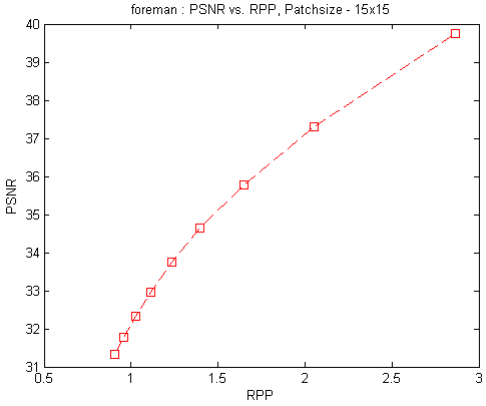

Figure 4.3: Foreman:PSNR vs RPP, Patchsize-15x15

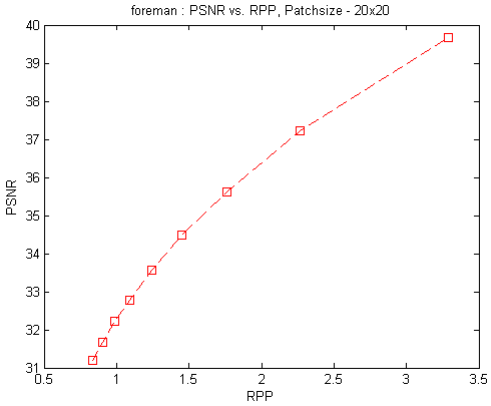

Figure 4.4: Foreman:PSNR vs RPP, Patchsize-20x20

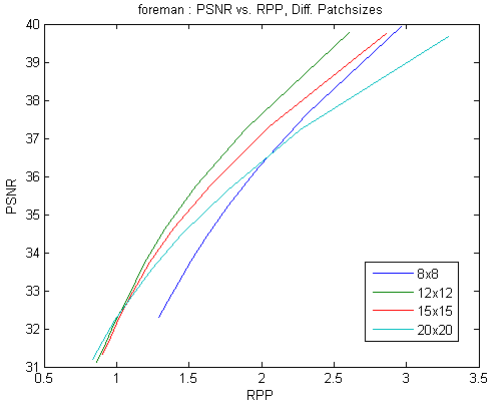

Figure 4.5: Foreman:PSNR vs RPP, different patch sizes

4.1.2.2 MOTHER AND DAUGHTER : PSNR vs RPP, for different patch sizes

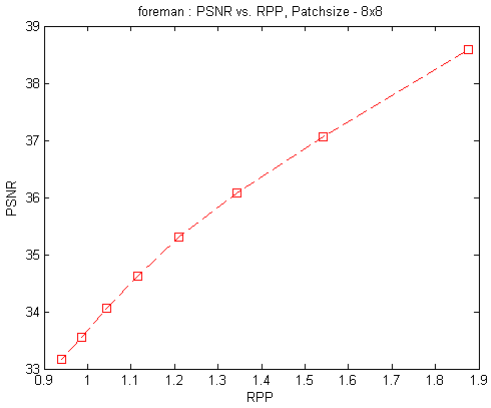

Figure 4.6: Mother:PSNR vs RPP, Patchsize-8x8

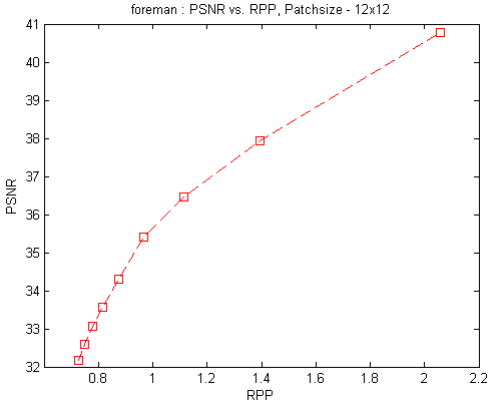

Figure 4.7: Mother:PSNR vs RPP, Patchsize-12x12

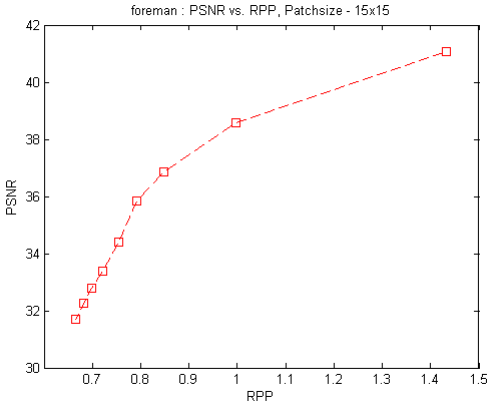

Figure 4.8: Mother:PSNR vs RPP, Patchsize-15x15

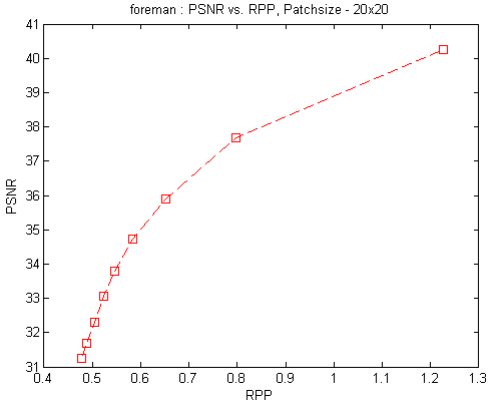

Figure 4.9: Mother:PSNR vs RPP, Patchsize-20x20

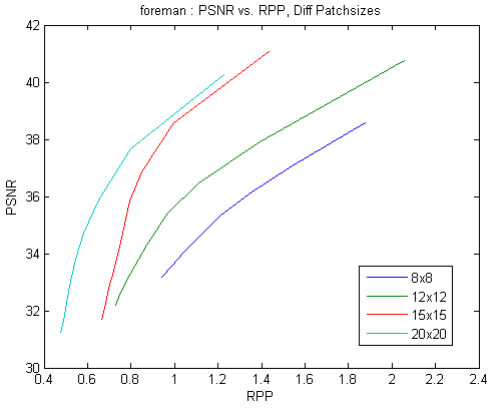

Figure 4.10: Mother:PSNR vs RPP, Diff. Patch sizes

4.1.2.3 Conclusion

For the FOREMAN video we have observed that using a patch-size of 12x12 is the optimum value. As we keep increasing the patch-size to 15x15 and 20x20 the efficiency is dropping. The size of 8x8 is also less efficient compared to 12x12. As with any algorithm, if the patch size is too small, there is very little a compression algorithm can do in terms of lowering the number of bits required to store these patches.

For the MOTHER AND DAUGHTER video however we observed that 20x20 is the best value for the patch-size. The efficiency keeps dropping for decreasing patch sizes. Compared to the FOREMAN video, MOTHER AND DAUGHTER has less detail in the background and the objects, so even a larger patch sizes like 20x20 don't tend to distort the detail in the video.

4.1.3 No. of Fractional bits (Q_2)

We have also examined the affect of the parameter Q_2(no. of bits in the fractional part) on the PSNR for both the videos.

4.1.3.1 FOREMAN : PSNR vs Q_2

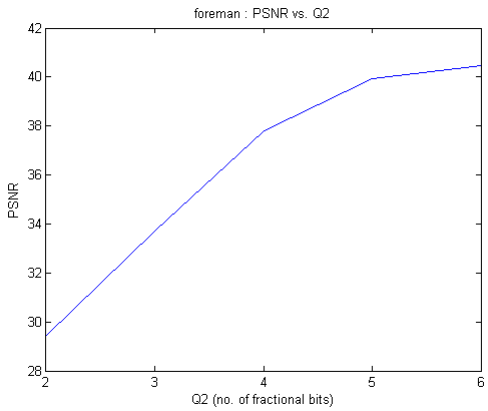

Figure 4.11: Foreman:PSNR vs Q_2

4.1.3.2 MOTHER AND DAUGHTER : PSNR vs Q_2

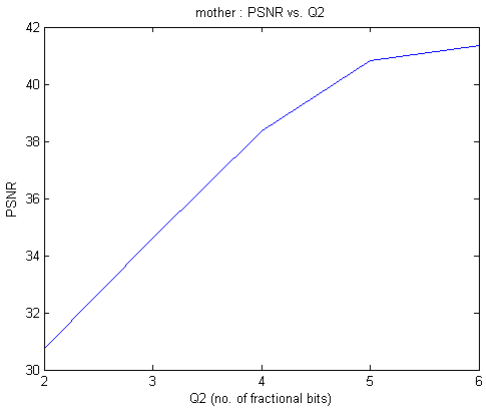

Figure 4.12: Mother and Daughter:PSNR vs Q_2

4.1.3.3 Conclusions

As expected the PSNR increases with increase in the no. of bits for the fractional part (the error is less if the value is more accurately coded). But beyond a point there is no significant improvement of PSNR (flattening), this just means that we have reached the limit of accuracy of the values.

4.1.4 Preview of reconstructed frames of using different δ

4.1.5 Compression

The following table summarizes the result of the encoder. Compression ratio is calculated by comparing the storage of the encoded video data to original video size.

Compression Ratio	Quality(PSNR)
3.06	39.77 dB
5.17	35.74 dB
8.66	31.61 dB

Table 4.1: Foreman (12x12) : Compression Ratio vs Quality

Compression Ratio	Quality(PSNR)
6.52	40.26 dB
12.25	35.89 dB
16.37	31.67 dB

Table 4.2: Mother and Daughter (20x20) : Compression Ratio vs Quality

4.2 Conclusion and Future Work

From the results of the above tests, one can observe that it is possible to encode video data by the SVD-like compression algorithm, using pre-trained orthonormal bases and sparse projections. We have achieved a considerable amount of compression compared to the raw video data. But this level of compression is much less than the compression achieved by the currently used video encoders. The next extension to this project is to add Motion Estimation Algorithms like Block-based matching. Among nearby frames, we can save storage of data of similar patches, by storing only the the motion vectors instead of the whole patch.

Bibliography

[1] K. Gurumoorthy, A. Rajwade, A. Banerjee, and A. Rangarajan, "Beyond SVD - sparse projections onto exemplar orthonormal bases for compact image representation", in *International Conference on Pattern Recognition (ICPR)*, 2008.

[2] Lijie Cao, "Singular Value Decomposition Applied To Digital Image Processing", *Division of Computing Studies Arizona State University Polytechnic Campus.*

[3] Xiph.org Video Test Media, URL : http://media.xiph.org/video/derf/

[4] Colin Malling : Digital Video Compression, URL : http://www.newmediarepublic.com/dvideo/compression/

[5] M. Anto Bennet, R. Nithyadevi, Dr. I. Jacob Reglend, Dr. C. Nagarajan, "Performance and Analysis of Video Compression Using Block Based Singular Value Decomposition Algorithm", in *International Journal of Modern Engineering Research (IJMER), Vol. 3, Issue. 3, May.-June. 2013 pp-1482-1486*

[6] G. W. Stewart, On the Early History of the Singular Value Decomposition, in *SIAM Review, Volume 35, Issue 4 (Dec., 1993), 551-566*