VIDEO COMPRESSION

Mentor : **Dr. Niteesh Sahni**

Submitted By:

Mayank Roy

Sarvagya Mukhi

Afroz Alam

Akshit Gupta

Riya Aggarwal

Shreya Gupta

Faisal Ahmed

Sunny Kr. Maurya

Sanchit Kaushal

AIM:

The aim of the project is to develop a software for image reconstruction and use it for video compression using the “Frame Theory”.

WHY IS THERE A NEED FOR VIDEO COMPRESSION?

We all have felt annoyed at not being able to e-mail our favourite videos to our friends. Also at the time when it takes quite a lot of time and money for data storage the need of effective video compression software is felt to compress the data effectively and also in such a way that it can be retrieved in the future.

Raw video now days contains an immense amount of data which creates a lot of issues while it’s storage and transmission for various purposes and hence need to be compressed.

Communication and storage capabilities are limited and expensive and hence video compression is required to resolve the issue to a certain extent

EARLIER DEVOLOPMENTS

Video compression was earlier done using the “Wavelet Transform”, “Discrete Cosine Transform”, “Single Value Decomposition” and “Fourier Transform” which the basis of all the available Video Compression Softwares. However the technique employed here is the use of the “Frame Theory” (which was devised just few years back) and not much work has been done in this technique.

SOFTWARES USED

Mathworks Matlab R2011a

REFERENCE BOOK

Frames for Undergraduates by Deguang Han, Keri Kornelson, David Larson, Eric Weber

Mathematical Concepts Used:

- Inner product spaces
- Orthonormal basis
- Cauchy-Schwartz inequality
- Linear transformation
- Matrix of a linear transformation
- Eigen values and Eigen vectors
- Moore-Penrose Inverse using Single Value Decomposition
- Discrete Fourier transformation
- Theory of frames

VECTOR SPACES

- A vector space V is a non empty set with two operations
  - ✓ Addition "+"
  - ✓ Multiplication "." by scalars such that the following conditions are satisfied for any x,y,z ϵ V and α , β in F

i. $x + y = y + x$

ii. $(x + y) + z = x + (y + z)$

iii. $x+z = y$ has unique solution z for each pair(x,y).

iv. $\alpha(\beta x) = (\alpha\beta)x$

v. $(\alpha+\beta)x = \alpha x+\beta x$

vi. $\alpha(x + y) = \alpha x + \alpha y$

vii. $1x = x$

BASIS AND DIMENSIONS

Basis: Suppose $S=\{v_1,v_2,\ldots,v_n\}$ is a set of vectors from the vector space *V*. Then *S* is called a **basis** (plural is **bases**) for *V* if both of the following conditions hold.

(a) *S* spans the vector space *V*.

(b) *S* is a linearly independent set of vectors.

Dimension: The dimension of vector space V is the "cardinality of a basis" which we denote by dim V

INNER PRODUCT SPACE

Let H be a vector space over F. An **inner product** is a map $\langle \cdot , \cdot \rangle$:H X H → F which satisfies the following properties for every x ,y , z ϵ H and α ϵ F:

- ✓ $\langle x , x\rangle \geq 0$ for all x ϵ H with $\langle x , x\rangle = 0$ if and only if x=0
- ✓ $\langle x+z,y\rangle = \langle x,y\rangle+ \langle z,y\rangle$ and $\langle \alpha x,y\rangle = \alpha\langle x,y \rangle$
- ✓ $\overline{< x, y >} = \langle y , x \rangle$ where $\overline{< x, y >}$denotes the complex conjugate of $\langle x , y >$.

- ✓ A vector space H equipped with an inner product is called an **inner product space** .

LINEAR OPERATOR

A mapping T from a vector space V to another vector space W such that :

$$T(au + bv)= aT(u) + bT(v) \quad [\, u,v \in V,\ \ a,b \text{ are scalars}]$$

ANALYSIS OPERATOR

Let $\{x_i\}^{k}_{i=1} \subset H$.The analysis Operator $Q : H \rightarrow C^k$ defined by

$$Qx= \begin{bmatrix} < x , xi > \\ . \\ < x , xk > \end{bmatrix} = \sum^{k}_{i=1} \langle x , x_i \rangle e_i$$

is called the analysis operator of $\{x_i\}^{k}_{i=1}$ $\{e_i\}^{k}_{i=1}$ is the standard orthonormal basis for C^k

ADJOINT OPERATOR

For any linear operator T from a finite dimensional inner product space H to K ,there exist a unique linear operator S from K to H ,such that for all $x \in H$ and $y \in K$,

$$\langle Tx , y \rangle = \langle x , Sy \rangle$$

We call this operator the "Adjoint of T" and will denote it T^*

ORTHONORMAL BASES

Orthogonal: Dot product is zero.

Normal: Magnitude is one.

Orthonormal = Orthogonal + Normal

Two special orthonormal bases :

1) Standard orthonormal basis
2) Fourier basis

- ➔ Standard orthonormal basis $\{e_0,e_1,\ldots e_{N-1}\}$ where $e_n[k]=0$ when $n \neq k$ and $e_n[n]=1$.
- ➔ Fourier basis $\{f_0,f_1,\ldots f_{N-1}\}$ for $l^2(Z_N)$ is defined as follows :

$$f_n[k]=\exp(2\Pi ikn/N) / \sqrt{N}$$

SOME IMPORTANT THEOREMS

- ✓ If T , F $\subset$ { 0,1,….N-1 } with $|T|>N+2\sqrt{N}+|F|$, the { $f_n\chi$ F|n ϵT} is a frame for $l^2(F)$.

- ✓ Suppose F $\subset$ {$f_0,f_1,\ldots f_{N-1}$} ,we define subspace S(F) $\subset l^2(Z_N)$ by

  $$S(F) = \{ v \subset l^2(Z_N) : v= \sum c_k f_k \}$$

- ✓ Suppose F $\subset$ { 0,1,….N-1 } and consider the vector space S(F) $\subset l^2(Z_n)$ .If T $\subset$ { 0,1,….N-1 } is any subset such that $|T|> N + 2\sqrt{N} +|F|$, then T is a set of uniqueness for S(F) .

Fourier Uncertainty Principle

Suppose that V is an image of size M X N which is band limited to set F $\subset$ Zm X Zn with cardinality|F|. If the pixel value of the image V are known on pixels in the set T $\subset$ Zm X Zn with

$$\underline{|T| - |F|>MN-2\sqrt{MN}}$$

then the remaining MN- |T| unknown pixel value can be uniquely reconstructed.

The Moore-Penrose Inverse

Let T: H → K be a linear transformation which is one to one . We define Moore –Penrose inverse(pseudo inverse) of T , T^+ : K →H by :

$$T^+ = (T^*T)^{-1} T^*$$

Singular Value Decomposition

Let A ϵ R^{mxn} . Then there exist orthogonal matrices U ϵ R^{mxm} and V ϵ R^{nxn} such that matrix A can be decomposed as follows :

$$A = U \sum V^T$$

where $\sum$ is and m x n diagonal matrix of form :

$$\begin{bmatrix} a1\,0\,0\,\ldots 0\,0 \\ 0\,a2\,0\,\ldots 0\,0 \\ 0\,0\,a3\ldots\,0\,0 \\ \ldots\quad\ldots\,0 \\ 0\,0\,0\ldots\,ap\,0 \end{bmatrix}$$

and $a_1 \geq a_2 \geq \ldots.. \geq a_p \geq 0$

The { a_i } are termed as the singular values of matrix A.

- ✓ The column of U are termed as *left singular vectors* while the column of V are termed as *right singular vectors.*

Using SVD , The pseudo–inverse (moore-penrose inverse) can be easily computed as follows .Let A be decomposed as :

$$A^{+} = V \sum^{+} U$$

where the matrix $\sum^{+}$ takes the form :

$$\begin{bmatrix} 1/a1 & 0 & \dots & 0 & 0 \\ 0 & 1/a2 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1/a3 & \dots & 0 & 0 \\ . & . & . & \dots & . & . \\ 0 & 0 & 0 & \dots 1/ap & 0 \end{bmatrix}$$

For all non- zero singular values. If any a_i are zero, then a zero is placed in corresponding entry in $\sum^{+}$. If the matrix is deficient ,then one or more of its singular values will be zero.

Hence , SVD provides a means to compute the pseudo- inverse of a singular matrix.

Fourier Transform

- ✓ If we express a vector v as a span of its Fourier basis *viz.*

$$v= a_0f_0+ a_1f_1+\dots\dots+a_nf_n$$

Then the set of scalars $[a_0\ a_1 \dots\dots a_n]$ forms the Fourier transform F(v) of v.

- ✓ In digital image compression, the fourier basis is used.

Image

- ✖ A digital image consists of pixels with color intensity values for red, green and blue(rgb format). These values together make up a 3-D matrix consisting of three planes.

- ✖ An LUVMAP image is an M-by-3 matrix that contains the NTSC(**National Television System Committee**) luminance (L) and chrominance (U and V) colour components as columns that are equivalent to the colours in the RGB colour map.

The image on the right shows the L plane of the original image (left).

The L plane consisting of the luminance values returns a grayscale version of the image.

A grayscale image is really just a 2-D matrix whose entries correspond to pixel intensity value.

Image As A Vector Space

Figures 1 & 2 represent addition operation acted upon the above image.
Figure 1 → image + 80
Figure 2 → image – 20
Thus the image can be treated as a vector space where the specified magnitude gets added/subtracted from the original intensity values.

Image Compression Algorithm

a. Define the analysis operator of the frame $\{f_{m,n}\chi F \ : (m,n) \in T \}$

$$Q : l^2\,(F) \rightarrow l^2\,(T) : v \leftrightarrow (\langle\, v,\, f_{m,n}\chi F \,\rangle)_{(m,n) \in T}$$

b. Compute the Moore-Penrose inverse : $Q^{+} : l^2\,(T) \rightarrow l^2\,(F)$ of Q

c. Form the vector $A \in l^2\,(T)$ by $A[m,n] = V[m,n]$ for $(m,n) \in T$

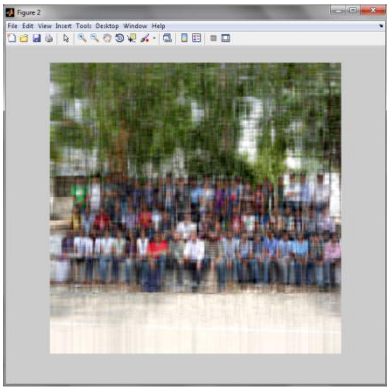
d. Compute $B \in l^2\,(F)$ defined by $B = Q^{+}\,A$

e. Form the vector $B \in l^2\,(Zm \times Zn)$ by $B = B\,\chi\,F$

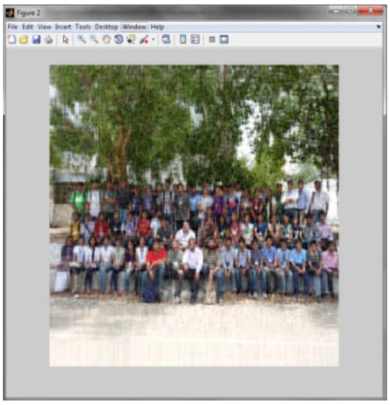
f. Compute the vector $W \in l^2\,(Zm \times Zn)$ by $W = F^{-1}\,B$

- ✓ The images shown below are approximated using different number of fourier basis .
- ✓ **IMAGE RECONSTRUCTION AFTER ADDITION OF FOURIER COEFFICIENTS**

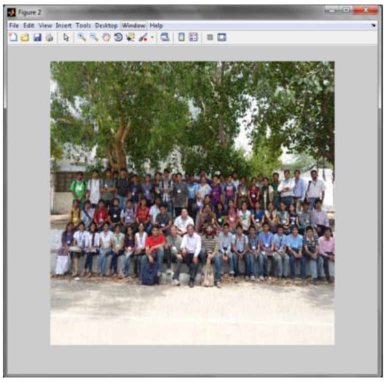
The images shown below are approximated using different number of fourier coefficients.



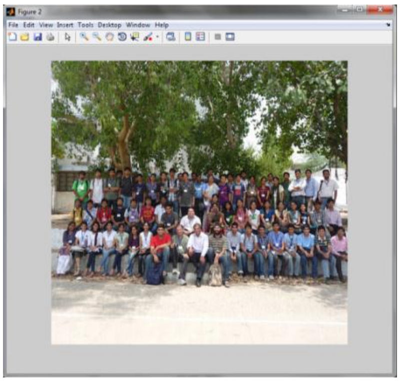
15 Fourier Coeff



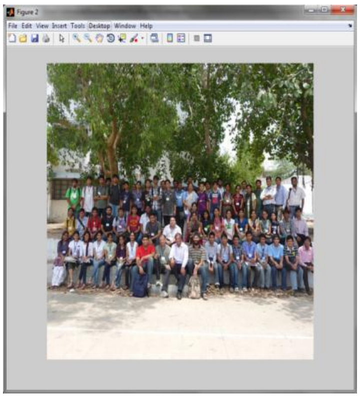
40 Fourier Coeff



100 Fourier Coeff



150 Fourier Coeff

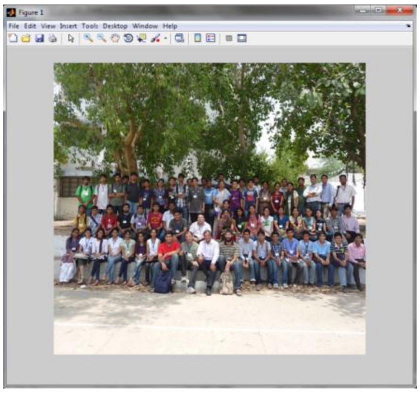


200 Fourier Coeff

250 Fourier Coeff

300 Fourier Coeff

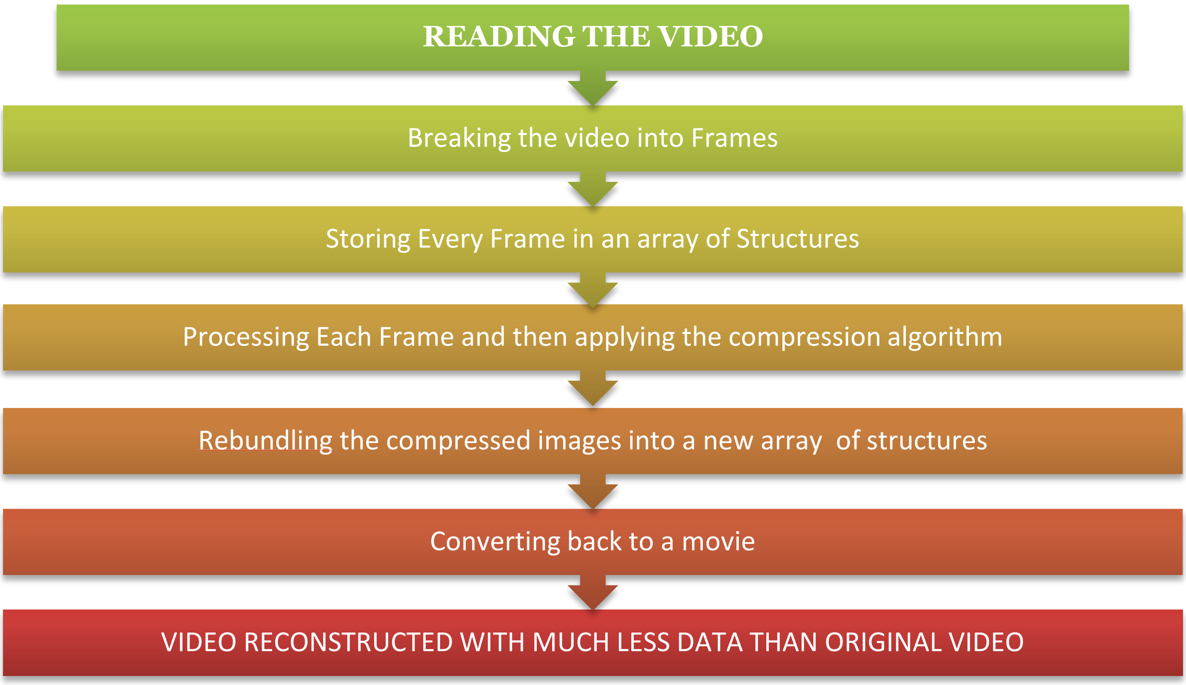
350 Fourier Coeff



Original image(540 Fourier Coeff)

Obviously, the smaller the number of fourier coefficient used, smaller is the approximation of the image with the original image.

ALGORITHM FOR VIDEO COMPRESSION



Source Code:

```
%Reading The Video
xyloObj = mmreader('xylophone.mpg');

%Reading Number of Frames in Video
nFrames=xyloObj.NumberOfFrames

%Reading width and height of the original video
vidHeight=xyloObj.Height;
vidWidth=xyloObj.Width;

% Defining a movie structure to store original video

mov(1:nFrames)=struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'),'colormap
',[ ]);


% Defining a movie structure to store compressed video
mov2(1:nFrames)=struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'),'colorma
p',[ ])

for k = 1:nFrames

%Storing each frame in the cdata of each element of the array of structures
mov(k).cdata = read(xyloObj,k);
     z=mov(k).cdata;

%Compressing Each Frame and displaying it
     X=compress_img(z);
imshow(X);

% Converting Double type compressed image into uint8 encoding
     X=uint8(150*X);
```

```
%Storing the compressed image in the compressed video structure
mov2(k).cdata=X;

end

hf1=figure(1)
set(hf1,'position',[150 150 vidHeightvidWidth])
hf2=figure(2)
set(hf2,'position',[150 150 vidHeightvidWidth])

%Playing Original Video
movie(hf1,mov,1,xyloObj.FrameRate);

% Playing Compressed Video
movie(hf2,mov2,1,xyloObj.FrameRate);
```

FUNCTIONS USED:

```
function X=compress_img(z)
%Converting image into ntsc mode dividing the image into luminosity and
%chrominance planes
luv=rgb2ntsc(z);

%Extracting the luminous plane
    L=luv(:,:,1);

%Taking single value decomposition of the plane
    [U D V]=svd(L);

    A=zeros(size(L));
    [M N]=size(L)
temp=M/4;
%Using temp Fourier coefficients to reconstruct the image
for j=1:temp

%Applying the Moore-Penrose inverse to reconstruct
    A=A+D(j,j)*U(:,j)*V(:,j)';
end

%Storing the new luminous plane in the compressed ntsc planes
luv2(:,:,1)=A;
luv2(:,:,2)=luv(:,:,2);
luv2(:,:,3)=luv(:,:,3);

%Converting back to rgb
    X=ntsc2rgb(luv2);
end
```

```
function [U,S,V] = svd(A)
```

```matlab
%SVD    Symbolic singular value decomposition.
%
%   With three output arguments, both [U,S,V] = SVD(A) and
%   [U,S,V] = SVD(VPA(A)) return numeric unitary matrices U and V
%   whose columns are the singular vectors and a diagonal matrix S
%   containing the singular values.  Together, they satisfy
%   A = U*S*V'.

if all(size(A) == 1)

   % Monoelemental matrix

   if nargout < 2
      U = A;
   else
      U = sym(1);
      S = A;
      V = sym(1);
   end

elseif nargout < 2

    if isscalar(struct(A)) && strcmp(class(A.s),'maplesym')
        U = sym(svd(A.s));
    else
        % Singular values only
        [U,stat] = mupadfeval('mlsvdvals',A);
        if stat ~= 0
            U = localsingularvals(A);
        end
    end

elseif isscalar(struct(A)) && strcmp(class(A.s),'maplesym')
    [U,S,V] = svd(A.s);
    U = sym(U); S = sym(S); V = sym(V);
else


   % Numeric singular values and vectors.

   A = vpa(A);
   out = mupadfeval('mlsvdvecs',A);
   U = fullshape(out(1));
   S = fullshape(out(2));
   V = fullshape(out(3));
end

function U = localsingularvals(A)
[m,n] = size(A);
if (m<n)
    B = A*A';
else
    B = A'*A;
end
ev = eig(B);
U = sqrt(ev);
```