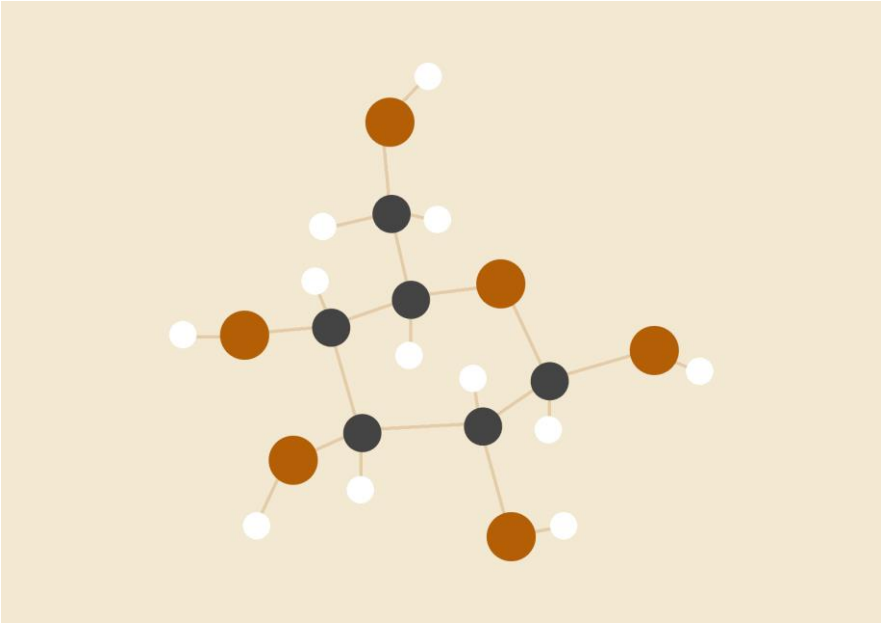
Data compression project

Compressing a video using discrete cosine transformation Algorithm



Names and IDs

Marian Sameh Makram Shendi
20201476313

Rodayna Beshir Hamad
20201503334

Mayar Mohamed Mahmoud Ali
20201382972

Nada Hamdy Fathy Abdelsalam
20201382436

Mayada Elnady Muhammed
20201378363

INTRODUCTION

The Discrete Cosine Transform (DCT) algorithm is a mathematical technique used to convert a spatial domain signal, such as an image or video frame, into a frequency domain representation, it reduces redundancy in video data by analyzing and encoding the frequency components .

CODE OF APPLYING THE DCT-ALGORITHM ON A VIDEO (MATLAB)

this code reads a video file, extracts its frames, applies DCT-based compression to the frames, and writes the compressed frames to an output video file.

```
clear all;
close all;
clc;

% Read the input video file
v = VideoReader('v1.mp4');

% Create a directory to store frames(images)
working = 'Frames';
mkdir(working);
mkdir(fullfile(working,'images'));
%ii will be used in the while loop
ii = 1;
% Write each frame of the input video to an image file in the working directory
while hasFrame(v)
    img = readFrame(v);

    %imgshow(img) and store the frames as images .png
    filename = [sprintf('%04d', ii),'.png'];
    fullname = fullfile(working, 'images', filename);
    imwrite(img,fullname);
    ii = ii+1;
end
% Get the list of image files in the working directory
imageNames = dir(fullfile(working, 'images','*.png'));
imageNames = {imageNames.name};
% Create a video writer object for the compressed output video
writeObj = VideoWriter(fullfile(working, 'output.avi'));
vid = VideoReader('v1.mp4');
% Open the video writer object for writing
open(writeObj);
```

```
% Loop through each image file in the working directory
for t = 1:length(imageNames)
    Frame = imread(fullfile(working, 'images', imageNames{t}));
    % Resize the image to 512x512 pixels
    n = imresize(Frame,[512,512]);
    % Apply DCT to each color channel of the image and store the results in
    % matrix z
    Z(:,:,1) = dct2(n(:,:,1));
    Z(:,:,2) = dct2(n(:,:,2));
    Z(:,:,3) = dct2(n(:,:,3));
% loop through matrix z
    for i = 1:512
        for j = 1:512
            %chick if the rows and cols >60
            if( (i+j) > 60 )
                Z(1,j,1) = 0;
                Z(1,j,2) = 0;
                Z(1,j,3) = 0;
            end
        end
    end
    % Recovering the bixels using the inverse of dct (icdt)for the channels
    % of the frames

    K(:,:,1) = idct2(Z(:,:,1));
    K(:,:,2) = idct2(Z(:,:,2));
    K(:,:,3) = idct2(Z(:,:,3));

    %Resize the frames to 30% and put the new frames in the opened video
    Frame = imresize(uint8(K),0.3);
    writeVideo(writeObj, Frame);

end
%Close the video writer object
close(writeObj);
```

USED FUNCTIONS

1. **VideoReader()** : It reads an input video file
2. **readFrame()** : it reads each frame from the video
3. **imwrite()** : used to save the frames as PNG images
4. **imread() :** It reads an image frame from directory
5. **imresize() :** It resizes the image frame

6. **dct2() :** applies the DCT (Discrete Cosine Transform)
7. **writeVideo() :** it writes the compressed frame to the VideoWriter object

PSEUDO CODE

```
clear all

close all

clc

v <- OUTPUT openreadvideo('video.extension of video)

working <- 'frames'

OUTPUT makedirection(working)

OUTPUT makedirection (OUTPUT fullfile (working,'images'))

ii<-1

while hasframed[v]

  img<- OUTPUT readframe(v)

  filename <- display in next line ('%04d',ii),.          extension of photos)

   fullname <- OUTPUT fullfile(working,'images', filename)

  OUTPUT WriteImagesInFormat(img, fullname)

  ii<-ii+1

Endloop

Imgnames <- directory (OUTPUT fullfile(working,'images','*. extension of photos))

Imgnames <- [imgnames.names]

writeObj <- OUTPUT videowriter(fullfile (working , output.avi)

Vid <- openreadvideo (video.extension of video)

open(writeobj)
```

```
for t=1 to length(imagenames)
  Frame <- OUTPUT WriteImagesInFormat ( OUTPUT fullfile (working,'images',
imagenames (t)))
  n<- OUTPUT imgsize(frame,scale[width,hight])
  Z(:,:,1)= OUTPUT 2d discrete cosine transform(n,(:,:,1))
  Z(:,:,2)= OUTPUT 2d discrete cosine transform(n,(:,:,2))
  Z(:,:,3)= OUTPUT 2d discrete cosine transform(n,(:,:,3))
  for i=1 to 512
     for j=1 to 512
        if( (i+j)>60))
           Z(1,j, 1)=0
            Z(1,j, 3)=0
             Z(1,j, 4)=0
        Endif
    Enfor
Endfor
 K(:,:,1)=OUTPUT inverse of 2d discrete cosine transform z(:,:,1))
 K(:,:,2)=OUTPUT inverse of 2d discrete cosine transform z(:,:,2))
 K(:,:,3)=OUTPUT inverse of 2d discrete cosine transform z(:,:,3))
 frame <- imgsize(unit8(k), scale)
 Write video (writeObj ,frame)
Endfor
Close( writeobj)
```

COMPARISON BETWEEN VIDEO CHARACTERISTICS BEFORE AND AFTER COMPRESSION

POINT OF COMPARISON	BEFORE	AFTER
Bitrate : the amount of data that is required to represent one second of video	1179 kbps	802 kbps
Frame : a still image that represents a single moment in time in the video sequence	179	176
Ratio aspect : relationship between the width and height of the video frame	(1:1.78/1:1.78)	(1:1/1:1)
Resolution: refers to its pixel dimensions, usually expressed as the number of pixels in the height and width of the video frame	576 x 1024	154 x 154
Size	848 KB	605 KB