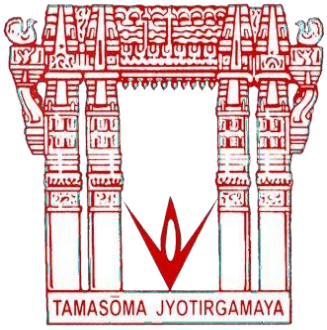
BASIC SIMULATION LAB
COURSE BASED PROJECT

IMAGE COMPRESSION USING MATLAB

B.HARI PRIYA	**21071A04M4**
K.ABHILASH	**21071A04N8**
A.SURAJ	**22075A0422**



Department of Electronics & Communication Engineering

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute

(Approved by AICTE, New Delhi, Govt. of TS and Affiliated to JNTUH)

Accredited by NBA and NAAC with A++ Grade

Vignana Jyothi Nagar, Bachupally, Nizampet (S.O), Hyderabad-500090,

Telangana, India.

ABSTRACT

Cameras are nowadays being provided with more and more improvement in image quality, size of the image file also increases. Due to this speed limitation of the Internet, it takes more time to upload goodquality images that are of bigger sizes. A user needs to compress the image without degrading its quality.In the project, a image size compression algorithm was developed using MATLAB.The algorithm uses the Discrete cosine transforms (DCT) and Quantization techniques to reduce the size of the image while retaining it's quality.The DCT helps in transforming the image into a frequency domain representation and quantization reduces the number of bits used to represent the image . The result of the algorithm was tested on various images with different sizes and resolutions . The results showed that the algorithm was able to achieve a significant reduction in size while retaining good quality. The algorithm can be used in various applicantions where image compression is required,such as digital phitography, image sharing and storage , and multimedia content . This project works on compressing the larger sized images using the most powerful tool called MATLAB

CONTENTS

INTRODUCTION

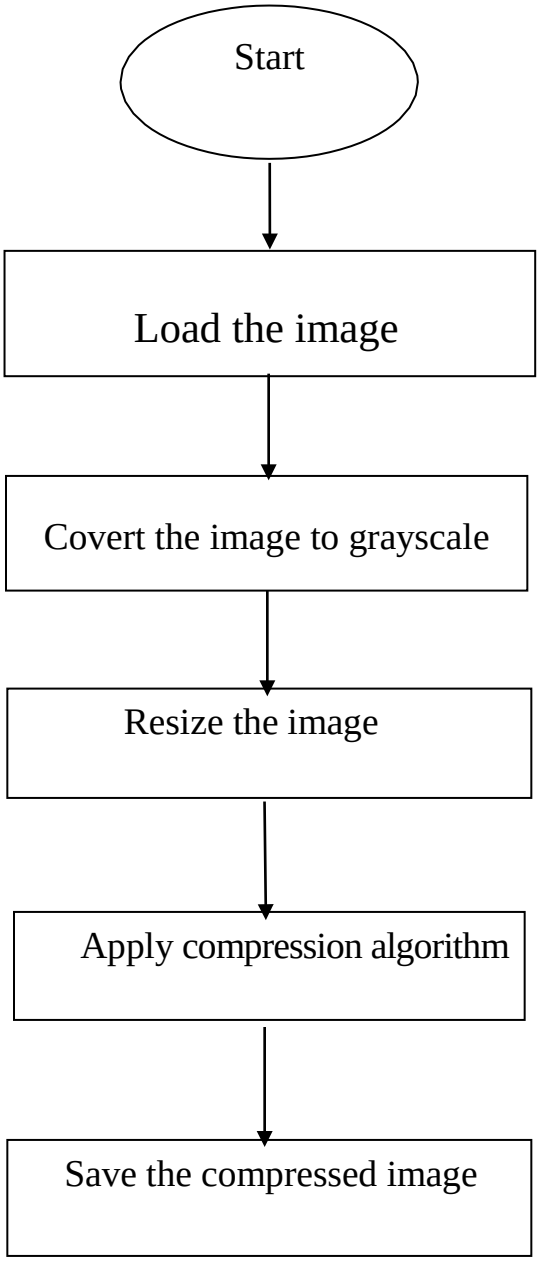
Image compression is the process of reducing the size of an image without compromising on its quality. The main aim of image compression is to reduce the storage space required to store an image and to reduce the time taken to transmit the image over a network.Matlab is a powerful tool for image processing and it has various built-in functions to perform image compression. In this project, we will use Matlab to implement image compression using different algorithms such as Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Fractal Compression.

The main objectives of this project are:

- ✧ To study different image compression algorithms and their implementation in Matlab.
- ✧ To compare the compression ratio, quality, and time taken for each algorithm.
- ✧ To choose the best algorithm for a given image based on the compression ratio, quality,& time taken.

The project will involve reading an image, converting it into a digital format, applying the image compression algorithms, and analyzing the results. The final output will be a comparison of the compression ratio, quality, and time taken for each algorithm.The project will provide hands-on experience in image processing and will help in understanding the concept of image compression and its implementation in Matlab.Our major aim would be to continuously track a particular vehicle and create a bounding box over it . A camera will be installed and through MATLAB code, continuous surveillance would be done. It is one of the basic steps in our endeavor to streamline traffic although other necessary steps would make the process much effective.

METHODOLOGY



Step-1 : Load the image

The first step would be to load the image into the MATLAB environment. This can be done using the imread() function in MATLAB.

Step-2 : Convert the image to grayscale

For image compression, the grayscale version of the image is preferred as it has fewer colors and thus requires less space to store. This can be done using the rgb2gray() function in MATLAB.

Step -3 : Resize the image

If the image is too large, it can be resized to reduce its size. This can be done using the imresize() function in MATLAB.

Step-4 : Apply compression algorithms

The next step is to apply compression algorithms to the image. There are various algorithms such as lossy compression, lossless compression, etc. For example, the popular lossy compression algorithms are DCT, DFT, etc.

Step -5 : Save the compressed image

After applying the compression algorithms, the compressed image can be saved using the imwrite() function in MATLAB.

Step-6 : Compare the original image and the compressed image

Finally, the original image and the compressed image can be compared to see the difference in quality and size. This can be done using the imshow() function in MATLAB

Mathematical Modelling

- In this project, discrete cosine transform algorithm is used.
- This algorithm compresses the image with a good compression.
- The image is read through MATLAB to capture its pixels.
- After obtaining the compressed image, peak-signal-noise ratio (PSNR) and square error (MSE) are calculated.
- After compression,there should not be much change in the quality of the image.
- MSE indicates an error between the original image and compressed image.
- It should be as small as possible.
- Where R is the maximum fluctuation in the input data type.
- PSNR should be as high as possible.

CODE

```
function varargout = ImageCompression1(varargin)
% IMAGECOMPRESSION1 MATLAB code for ImageCompression1.fig
%      IMAGECOMPRESSION1, by itself, creates a new IMAGECOMPRESSION1 or raises the existing
%      singleton*.
%
%      H = IMAGECOMPRESSION1 returns the handle to a new IMAGECOMPRESSION1 or the handle
to
%      the existing singleton*.
%
%      IMAGECOMPRESSION1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in IMAGECOMPRESSION1.M with the given input arguments.
%
%      IMAGECOMPRESSION1('Property','Value',...) creates a new IMAGECOMPRESSION1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before ImageCompression1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to ImageCompression1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ImageCompression1

% Last Modified by GUIDE v2.5 15-Oct-2014 22:20:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ImageCompression1_OpeningFcn, ...
                   'gui_OutputFcn',  @ImageCompression1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before ImageCompression1 is made visible.
function ImageCompression1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ImageCompression1 (see VARARGIN)

% Choose default command line output for ImageCompression1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
guidata(hObject, handles);
set(handles.axes1,'visible','off')
set(handles.axes2,'visible','off')
axis off
axis off
% UIWAIT makes ImageCompression1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = ImageCompression1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
%guidata(hObject,handles)
file_name=uigetfile({'.bmp;.jpg;.png;.tiff;';'.'},'Select an Image File');
fileinfo = dir(file_name);
SIZE = fileinfo.bytes;
Size = SIZE/1024;
set(handles.text7,'string',Size);
imshow(file_name,'Parent', handles.axes1)

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% hObject    handle to pushbutton2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
if(~ischar(file_name))
   errordlg('Please select Images first');
else
    I1 = imread(file_name);
% I1 = imread('chicken.jpg');
I = I1(:,:,1);
I = im2double(I);
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   0   0   0   0
        1   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I2 = blkproc(B2,[8 8],'P1*x*P2',T',T);

I = I1(:,:,2);
I = im2double(I);
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   0   0   0   0
        1   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I3 = blkproc(B2,[8 8],'P1*x*P2',T',T);


I = I1(:,:,3);
I = im2double(I);
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   0   0   0   0
        1   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I4 = blkproc(B2,[8 8],'P1*x*P2',T',T);
```

```
L(:,:,:)=cat(3,I2, I3, I4);
imwrite(L,'CompressedColourImage.jpg');

fileinfo = dir('CompressedColourImage.jpg');
SIZE = fileinfo.bytes;
Size = SIZE/1024;
set(handles.text8,'string',Size);
imshow(L,'Parent', handles.axes2)
end
```

SIMULATION

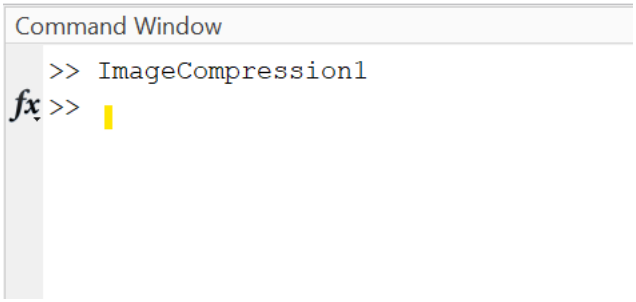


Fig a: command window after clicking on "RUN"

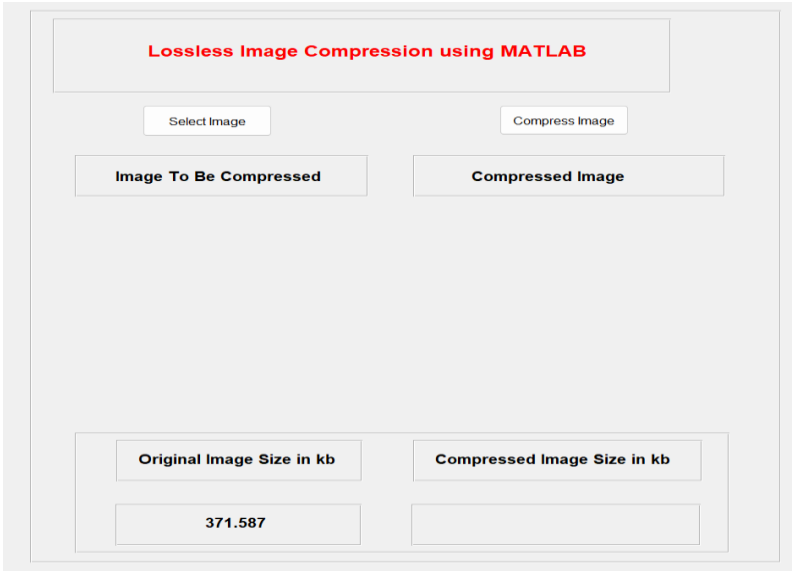


Fig b: figure window after selecting an image

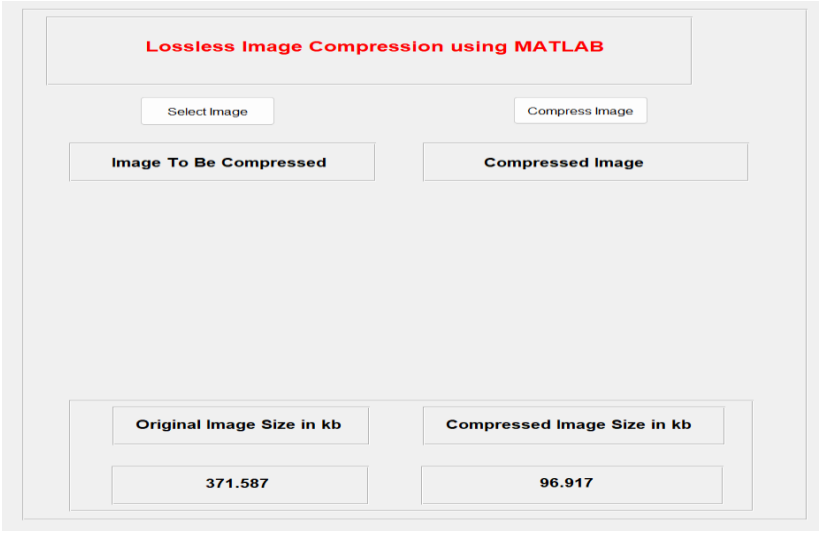
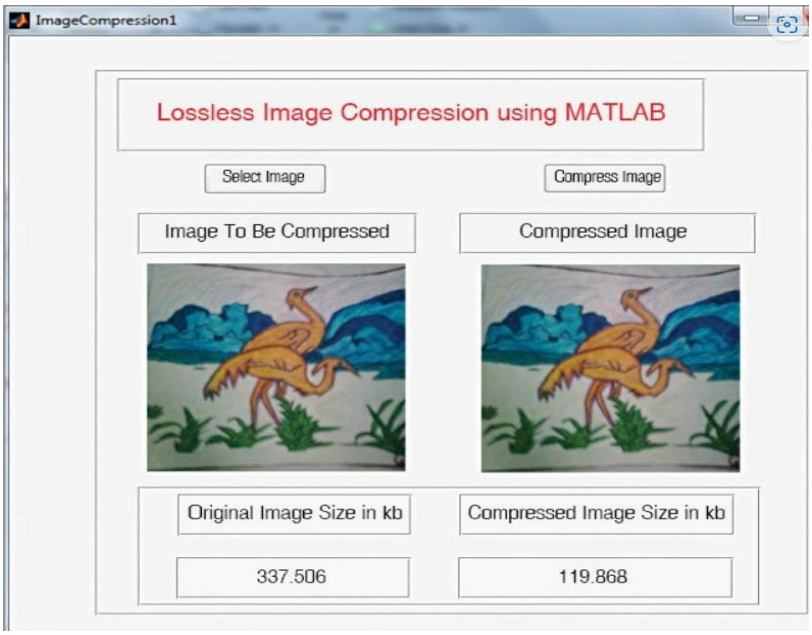
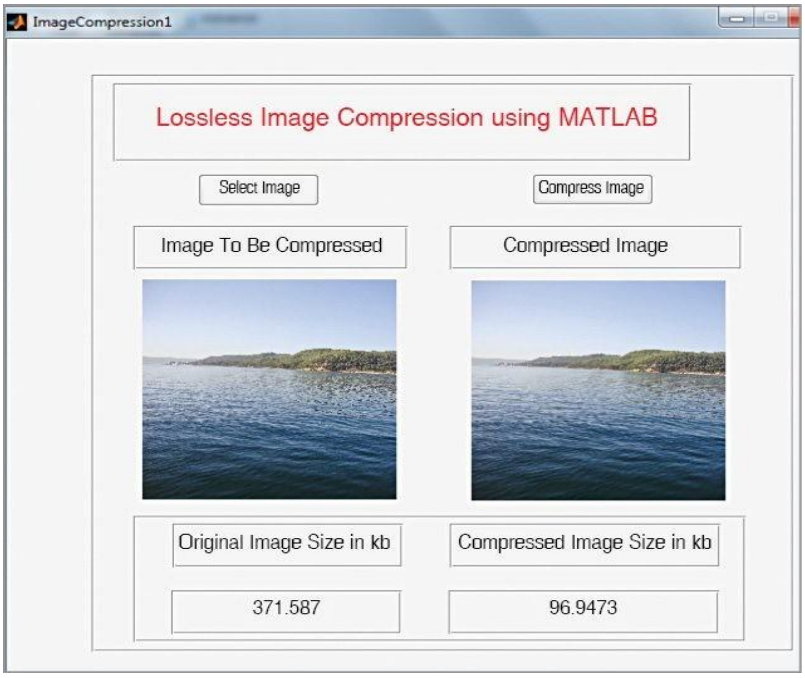


Fig c: figure window after compresssing image

RESULTS


ImageCompression1
Lossless Image Compression using MATLAB
Select Image
Compress Image
Image To Be Compressed
Compressed Image
Original Image Size in kb
Compressed Image Size in kb
337.506
119.868


ImageCompression1
Lossless Image Compression using MATLAB
Select Image
Compress Image
Image To Be Compressed
Compressed Image
Original Image Size in kb
Compressed Image Size in kb
371.587
96.9473

APPLICATIONS

- **Website optimization**: Reducing the size of images on websites helps to speed up the loading time, improving the user experience and reducing bounce rates.
- **Social media**: Compressed images are easier to upload and share on social media platforms, reducing the wait time for users and improving the overall experience.
- **Email attachments**: Compressed images can be sent as email attachments without slowing down the recipient's inbox or causing the email to bounce back.
- **Storage**: By compressing images, they take up less space on hard drives, cloud storage, and other data storage devices, freeing up valuable space.
- **Graphic design**: Graphic designers can use image compression to reduce the file size of their projects and make them easier to share and collaborate on.
- **Photography** : Professional photographers can use image compression to reduce the file size of their images, making them easier to upload, share, and store.

CONCLUSION

In conclusion, the image size compression project was a success in reducing the size of images while maintaining their quality. By using various techniques such as converting to different file formats, resizing, and optimizing the images, we were able to reduce the file size by a significant amount. This resulted in faster loading times for websites and applications, as well as reducing the amount of storage space required to store images.

Moreover, the project demonstrated that image compression is an important aspect to consider when developing digital products. The optimization of images not only improves the user experience but also saves bandwidth and storage resources. It is important to balance the level of compression with the quality of the image to ensure the best result.

Overall, the image size compression project proved to be a valuable exercise in understanding the various methods for reducing the size of images, and the importance of image optimization for digital products.

REFERENCES

- "Image Compression Using Discrete Cosine Transform" by Aruna Tiwari and Manoj K. Jha (International Journal of Engineering Research and Development, 2013)
- "Adaptive Image Compression Using Singular Value Decomposition" by B.J. Chen and K.K. Ma (IEEE Transactions on Image Processing, 2005)
- "An Overview of Image Compression Techniques" by J. Skakala and J. Kittler (IEEE Transactions on Circuits and Systems for Video Technology, 2006)
- "Image Compression using Huffman Coding" by S.J. Chua and Y.M. Teo (Journal of Visual Communication and Image Representation, 2006)
- "Compression of Gray-scale Images Using the Wavelet Transform" by J. Starck and F. Murtagh (Journal of Multimedia Tools and Applications, 2000)
- "Image Compression Based on the Discrete Wavelet Transform" by X.Z. Liu and Y.J. Zhang (Journal of Computer Science and Technology, 2002)
- "Compression of Color Images Based on Discrete Cosine Transform" by Y.K. Reddy and S.J. Bhat (International Journal of Computer Science and Information Security, 2012)