Visvesvaraya Technological University
Belagavi-590 018, Karnataka

A MINI PROJECT REPORT ON

"Image Compression using SVD Algorithm"

Mini Project Report submitted in partial fulfillment of the requirement for The VI Semester File Structures Laboratory [17ISL68]

Bachelor of Engineering
In
Information Science and Engineering

Submitted by
SUJITH UP [1JT17IS045]

Under the Guidance of
Mr. Vadiraja A
Asst. Professor, Department of ISE

Department of Information Science and Engineering
Jyothy Institute of Technology
Tataguni, Bengaluru-560082

Jyothy Institute of Technology
Tataguni, Bengaluru-560082
Department of Information Science and Engineering

CERTIFICATE

Certified that the mini project work entitled **“Image Compression Using SVD Algorithm”** carried out by **SUJITH UP [1JT17IS045]** bonafide student of Jyothy Institute of Technology, in partial fulfillment for the award of **Bachelor of Engineering** in **Information Science and Engineering** department of the **Visvesvaraya Technological University, Belagavi** during the year **2020-2021**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Prof. Vadiraja A	**Dr. Harshvardhan Tiwari**
Guide,Asst, Professor	Associate Professor and HOD
Dept. Of ISE	Dept. OF ISE

External Viva Examiner

Signature with Date :

-
-

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of my project would be incomplete without mentioning the people who made it possible, whose constant guidance and encouragement crowns all the efforts with success. We would greatly mention enthusiastic influence provided by **Mr. Vadiraja A,** Asst. Professor, Dept. of ISE for his ideas and co-operation showed on us during our venture and making this project a great success. I am thankful to **Mr. Harshvardhan Tiwari**, HOD, Department of ISE for his co-operation and encouragement at all moments of our approach. I am also thankful to **Dr. Gopalakrishna K**, Principal, JIT, Bangalore for being kind enough to provide us an opportunity to work on a project in this Institution.

Finally,it’s pleasure and happiness to the friendly co-operation showed by all the staff members of Information Science Department, JIT.

SUJITH UP

1JT17IS045

ABSTRACT

This project has been created using PyCharm, with the platform Linux and language Python. The project title-“DATA COMPRESSION” is a process of encoding information using fewer bits than the original representation. Any compression can be lossy or lossless. No information is lost in lossless compression. Lossy compression reduces bits by removing unwanted information.

There are many many algorithms that can be used for compressing images. The algorithm I used is SVD algorithm. Singular Value Decomposition(SVD) is a lossy compression technique. It uses linear matrix manipulation for compressing images. The purpose of this project is to reduce the size of images thus saving cost of storage. Simple User Interface (UI) is designed to select a photo and compress it. It also display compression ratio and other useful information.

SVD algorithm gives good compression with less computational complexity when compared to other compression algorithms. Besides image compression, SVD has application in noise reduction, face recognition, water marking, etc.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

INTRODUCTION

1.1 Introduction to File Structures

In simple terms, a file is a collection of data stored on mass storage (e.g., disk or tape).But there is one important distinction that must be made at the outset when discussing file structures. And that is the difference between the logical and physical organization of the data.

On the whole a file structure will specify the logical structure of the data, that is the relationships that will exist between data items independently of the way in which these relationships may actually be realized within any computer. It is this logical aspect that we will concentrate on. The physical organization is much more concerned with optimizing the use of the storage medium when a particular logical structure is stored on, or in it. Typically for every unit of physical store there will be a number of units of the logical structure (probably records) to be stored in it.

For example, if we were to store a tree structure on a magnetic disk, the physical organization would be concerned with the best way of packing the nodes of the tree on the disk given the access characteristics of the disk.

Like all subjects in computer science the terminology of file structures has evolved higgledy-piggledy without much concern for consistency, ambiguity, or whether it was possible to make the kind of distinctions that were important.

It was only much later that the need for a well-defined, unambiguous language to describe file structures became apparent. In particular, there arose a need to communicate ideas about file structures without getting bogged down by hardware considerations.

1.2 Introduction to File System

In computing, a file system or file system controls how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with noway to tell where one piece of information stops and the next begins. By separating the data into pieces and giving each piece a name, the information is easily isolated and identified. Taking its name from the way paper-based information systems are named, each groups of data is called a “file”. The structure and logic rules used to manage the groups of information and their names is called a “file system”.

There are many different kinds of file systems. Each one has different structure and logic, properties of speed, flexibility, security, size and more. Some file systems have been designed to be used for specific applications. For example, the ISO 9660 file system is designed specifically for optical discs.
File systems can be used on numerous different types of storage devices that use different kinds of media. The most common storage device in use today is a hard disk drive. Other kinds of media that are used include flash memory, magnetic tapes, and optical discs. In some cases, such as with tmpfs, the computer's main memory (random-access memory, RAM) is used to create a temporary file system for short-term use.

Some file systems are used on local data storage devices; others provide file access via a network protocol(for example, NFS, SMB, or 9P clients). Some file systems are “virtual”. Meaning that the supplied “files” (called virtual files) are computed on request (e.g. procfs) or are merely a mapping into a different file system used as a blacking store. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning with regard to the physical storage medium are important design considerations.

1.3 Data Compression

With the advancement in technology, multimedia content of digital information is increasing day by day, which mainly comprises of images. Hence storage and transmission of these images require a large memory space and bandwidth. The solution is to reduce the storage space required for these images while maintaining acceptable image quality.

Many methods are available for compression of images. The compression technique that we used is SVD (Singular Valued Decomposition). SVD is a linear matrix transformation used for compressing images. It consist of 3 component matrices L, D and R such that

$$G = LDR^T$$

where G is a m * n matrix

D is a m * n diagonal matrix with singular values of G

L is a m * n matrix containing left singular vectors of G

R is a n * n matrix containing right singular vector of G.

To compress an image, after applying SVD, only a few singular values are retained while other singular values are discarded. This follows from the fact that singular values are arranged in descending order on the diagonal of D and that first singular value contains the greatest amount of information and subsequent singular values contain decreasing amounts of image information.

CHAPTER 2
REQUIREMENT ANALYSIS AND DESIGN

REQUIREMENT ANALYSIS AND DESIGN

2.1 Domain Understanding

Lossy compression is used a needy tool in many practical applications in real world such as whatsapp image transfer, snapchat dm’s and etc. Further, we focus on developing a tool to compress the given image in a standard format. The outcome of this project is to learn and implement standard image compression algorithm and ease the user for compressing images with an interactive GUI.

2.2 Classification of Requirements

Project requirements and expected outcomes:

Project requirements lay out the scope of the project and thereby providing a direct correlation with its usability. Hence in this section, we describe critical requirements assumed and their expected outcomes. On a higher level, we have categories such as functional, software and hardware requirements.

User requirements:

- Image file format and Compression type: In here, we expect the tool (image compressor) to handle standard JPG/JPEG files as input and to provide a lossy compressed version having more than 50% compression of the same as output.
- Image type and resolution: The tool is expected to handle both gray scale and color images with resolution up to 640x480 pixels. We have restricted the size of the image to above resolution because to provide an optimal execution time or to provide a feel of in-line compression.

- Interactive GUI: A simple and intuitive interface, able to provide an image file upload option and view the compression details in the window.

Software and Hardware Requirements

- Any Linux system with minimum 2 GB RAM with python and supporting libraries such as Tkinter, NumPy, Pillow installed.

2.3 System Analysis

When the program is executed, it loads a simple Gui which primarily asks the user to press the Plus button. If the user chooses the Plus button, then it asks to input the photo to compress. After the Photo is obtained, the entire photo is viewed for verification of photo with name and location.

When user press Compress button it starts to compress the photo. After the completion of compressing, the user is given information about compression such as time taken, original image size, compressed image size, compression ratio.

CHAPTER 3
IMPLEMENTATION

IMPLEMENTATION

3.1 Opening and Reading the Image

PIL is the Python Imaging Library (PIL) which provides the python interpreter with image editing capabilities. The Image module provides a class with same name which is used to represent PIL image. The module also provides a number of factory functions to load images from files, and to create new images.

PIL.Image.open() opens and identifies the given Image file. This function identifies the file, but the file remains open and the actual image data is not read from the file until you try to process the data. Parameters can also be passed to this function. Filename, path or file object can be passed to this function. The mode if given, this argument must be ‘r’.

3.2 Processing the Image

By storing the images read by Pillow (PIL) as a NumPy array, various Image processing can be performed using NumPy functions. By the operation of array, acquisition and rewriting of pixel values, trimming, concatenating can be done. Pass the image read by PIL to array. Then RGB (color) images become 3D array (row [height] * column [width] * color [3]).

Generate single-color images by setting other color values to 0. Pass array to Image.fromarray() to obtain PIL.Image. It can be saved as an image file with save().

3.3 Algorithm

In Linear algebra, the Singular Value Decomposition (SVD) is a factorization of a real or complex matrix that generalizes the Eigen decomposition of a square normal matrix to any m * n matrix via an extension of the polar decomposition.

The SVD is calculated via iterative numerical method. In Python, we can achieve SVD algorithm using NumPy function. NumPy also has built in function called linalg.svd() for calculating SVD. The function takes a matrix and returns U, Sigma, V and T. The Sigma diagonal matrix is returned as a vector of singular values. The V matrix is returned in a transposed form V^T.

Color images are represented in python as three dimension to represent the color values (red, blue, green). However SVD method is applicable to two dimensional array. So we need to convert the three dimensional array to two dimensional array. I did this by flattening the third dimension of the image array into second dimension using numpy’s reshape method. The function numpy.matmul() is used to return the product of two arrays. It is then converted into uint-8 format which is unsigned integer number which is expressed in hexadecimal or decimal notation Compress RGB colors separately and merge them to get overall compressed image.

3.4 Reason for choosing SVD Algorithm

- SVD algorithm is very efficient, it can be applied to real big matrices.
- We can implement PCA algorithm using SVD algorithm.
- SVD has application in noise reduction, face recognition, water marking, etc.
- It tends to perform quite well for most of the data sets.
- SVD is a powerful technique widely used in lot of models.
- Less computational complexity involved in achieving the image compression

3.5 User Interface

I have used Tkinter to create User Interface (UI) for my project. Tkinter is a python binding to TK GUI toolkit. It is the standard Python interface to Tk GUI toolkit. It provides fast and easy way to create GUI application.

There are two main methods used which the user needs to remember while creating the python GUI:

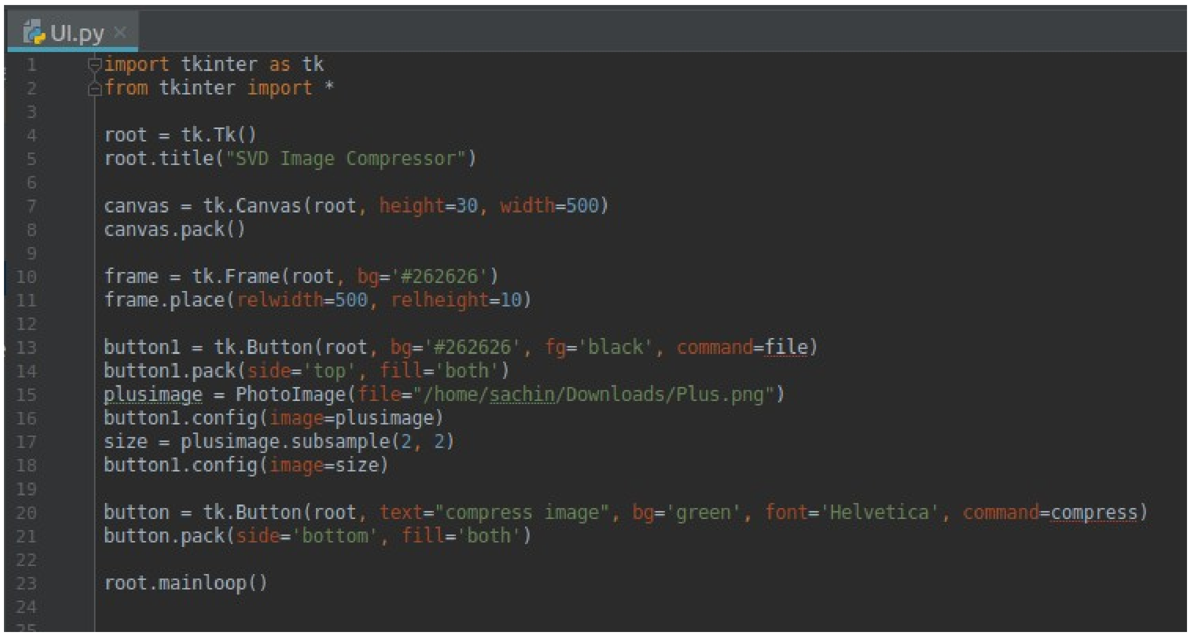
- Tk() - Used to create new main window. It has various parameters such has screen name, base name, class name. To change the name of the window, you can change the class name to the desired one.
- mainloop() - Used when your application is ready to run. It is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

Tkinter provides various controls, these are commonly called widgets. Some of the widgets that I used are:

- Button – It displays buttons in your application.
- Canvas – It is used to draw shapes, such as lines, ovals in your application.
- Frame – It is used as a container widget to organize other widgets.
- Label – It can hold images, can also be used to provide caption.

Tkinter also offers access to the geometric configuration of the widgets which can organize them in the parent windows. There are mainly three geometric manager classes:

- pack() method – It organizes the widgets in the blocks before placing in the parent widget.
- Place() method – It organizes the widgets by placing them on specific position directed by the programmer.
- grid() method – It organizes the widgets in grid before placing in the parent widget.



UI.py

```
import tkinter as tk
from tkinter import *

root = tk.Tk()
root.title("SVD Image Compressor")

canvas = tk.Canvas(root, height=30, width=500)
canvas.pack()

frame = tk.Frame(root, bg='#262626')
frame.place(relwidth=500, relheight=10)

button1 = tk.Button(root, bg='#262626', fg='black', command=file)
button1.pack(side='top', fill='both')
plusimage = PhotoImage(file="/home/sachin/Downloads/Plus.png")
button1.config(image=plusimage)
size = plusimage.subsample(2, 2)
button1.config(image=size)

button = tk.Button(root, text="compress image", bg='green', font='Helvetica', command=compress)
button.pack(side='bottom', fill='both')

root.mainloop()
```

Fig. 3.5 Here we can see the code where the user interface is created.

CHAPTER 4

RESULT AND SCREENSHOTS

Result And Screenshots

Screen Shot 1 (Main Screen)

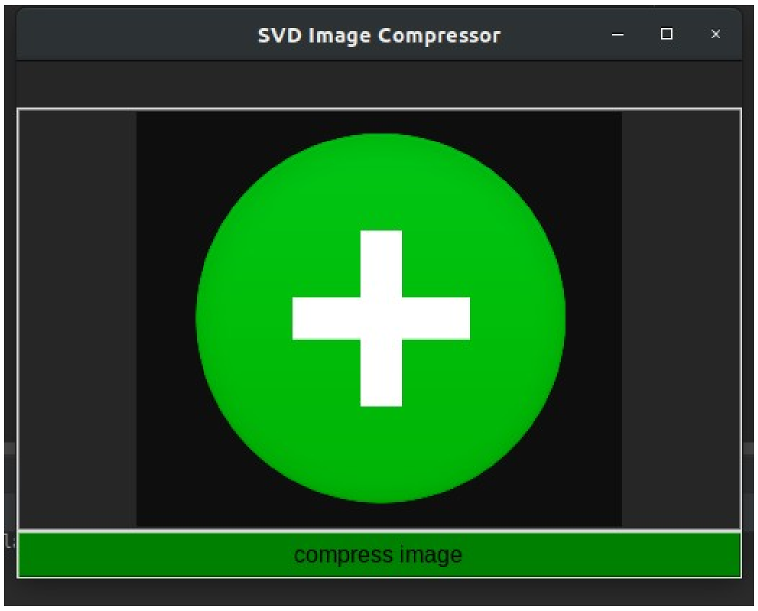


Fig 4.1 Main Screen

This is the main screen of the program. The user can press the plus button to choose the desired image.

Screen Shot 2

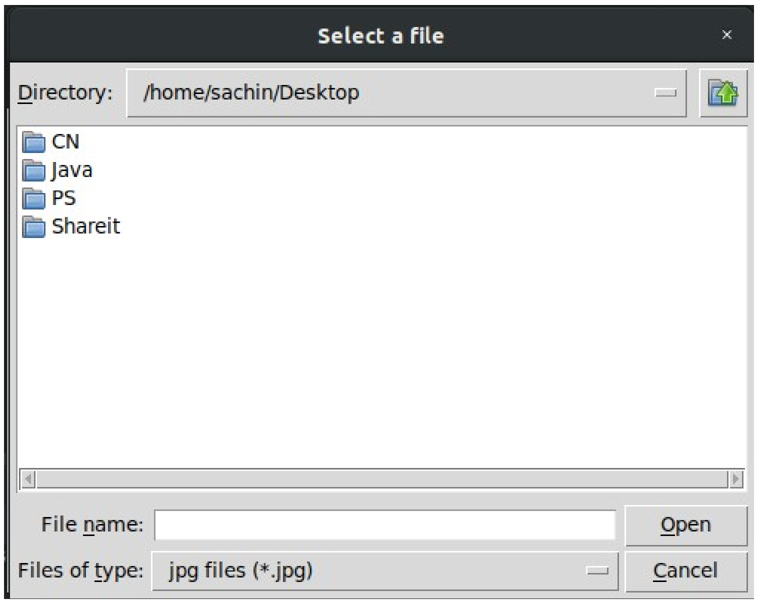


Fig 4.2 File Directory

This is the screen that the user is taken to when he opts for the “Plus” operation. The user has to just search the file to be compressed through the given browser.

Screen Shot 3

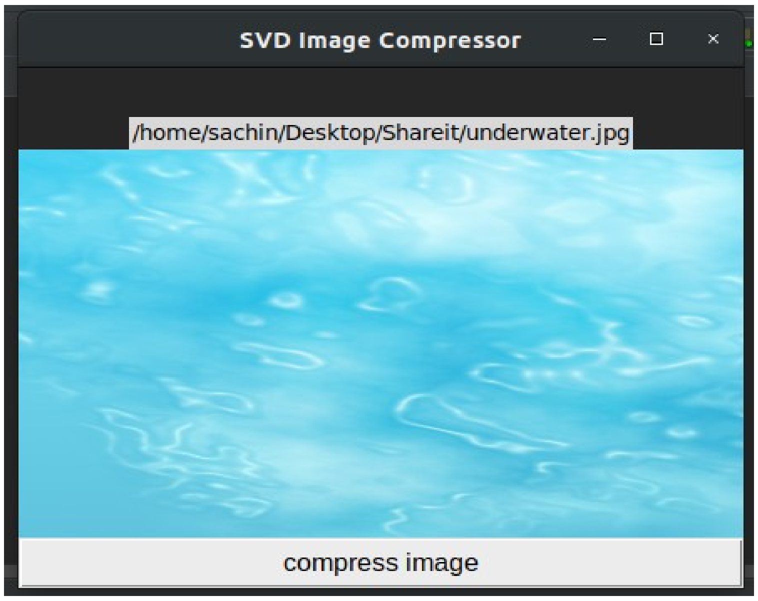


Fig 4.3 Selected Image

This is where the user can verify the Image. It also displays path and name of the selected picture.

Screen Shot 4

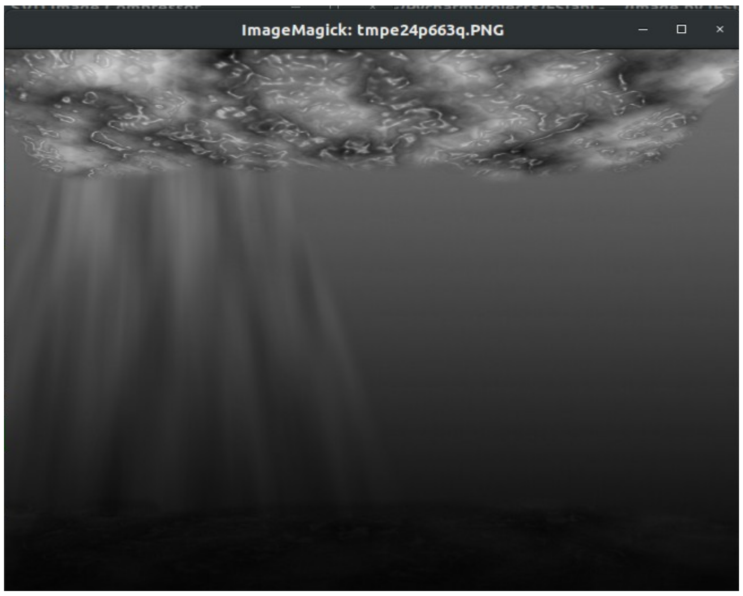


Fig 4.4 Red Channel Compression

The Selected image is separated into three channels, that is RGB (Red, Blue, Green). The above image displays color representation of Red channel image.

Screen Shot 5

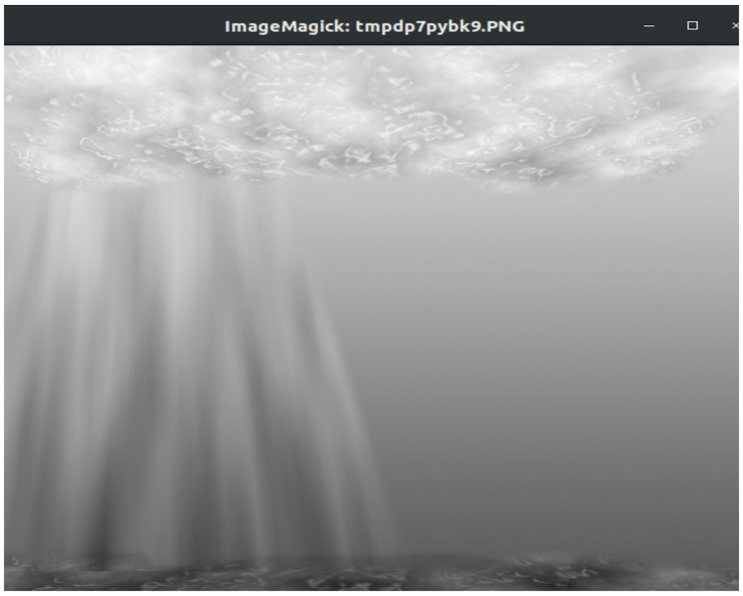


Fig 4.5 Blue Channel Compression

The above image displays color representation of Blue channel image.

Screen Shot 6

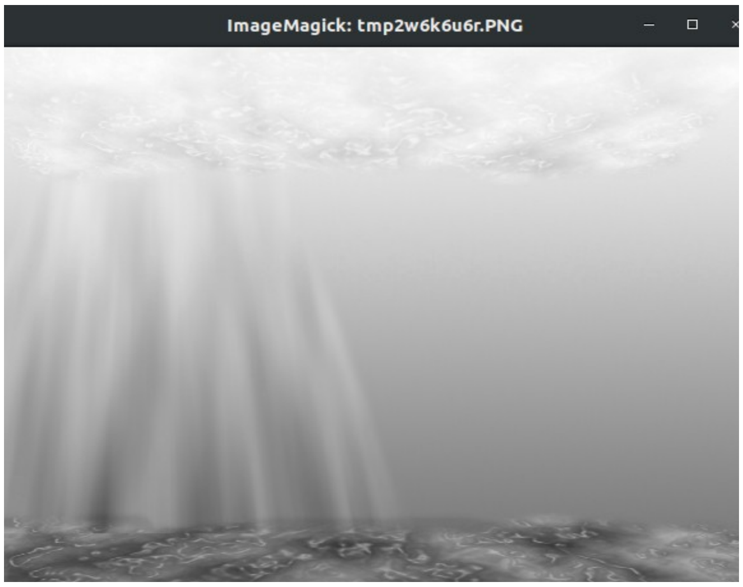


Fig 4.6 Green Channel Compression

The above image displays color representation of Green channel image.

Screen Shot 7

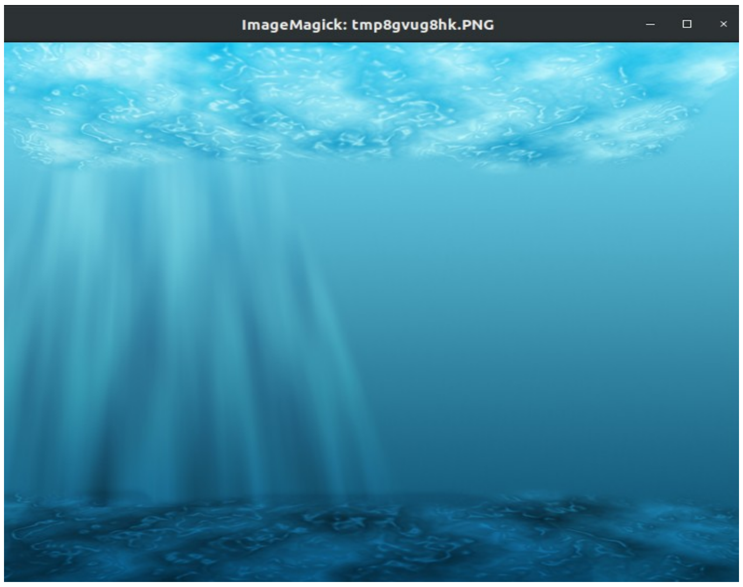


Fig 4.7 Merged Channels

This is the image we get by merging compressed RGB channels. The picture has lost its details and cannot be reverted back.

Screen Shot 8

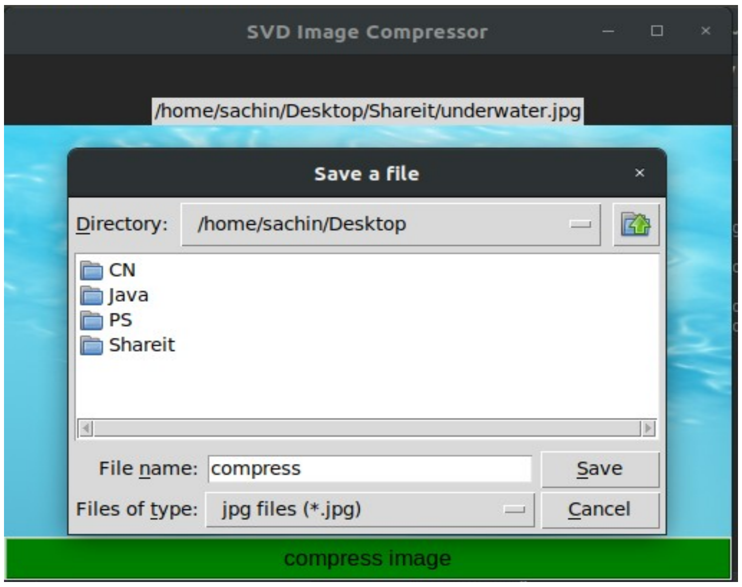
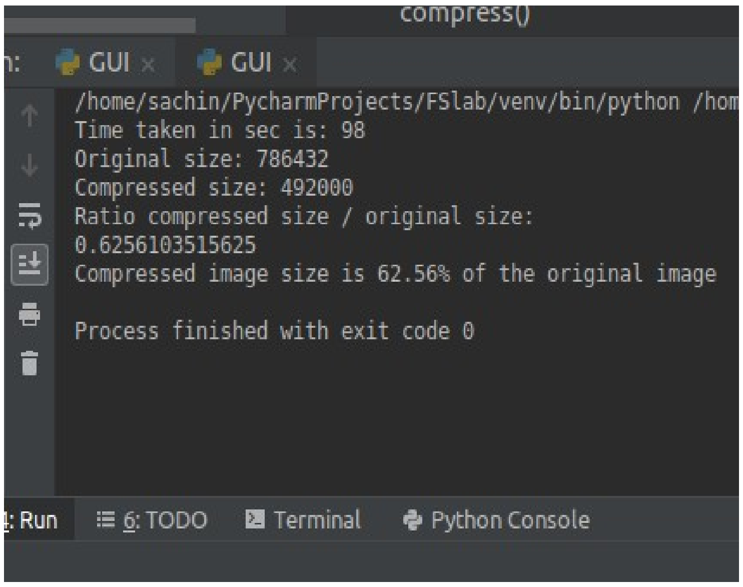


Fig 4.8 File Directory

User can choose the folder to save the compressed pic. File name for the compressed image should also be specified.

Screen Shot 9



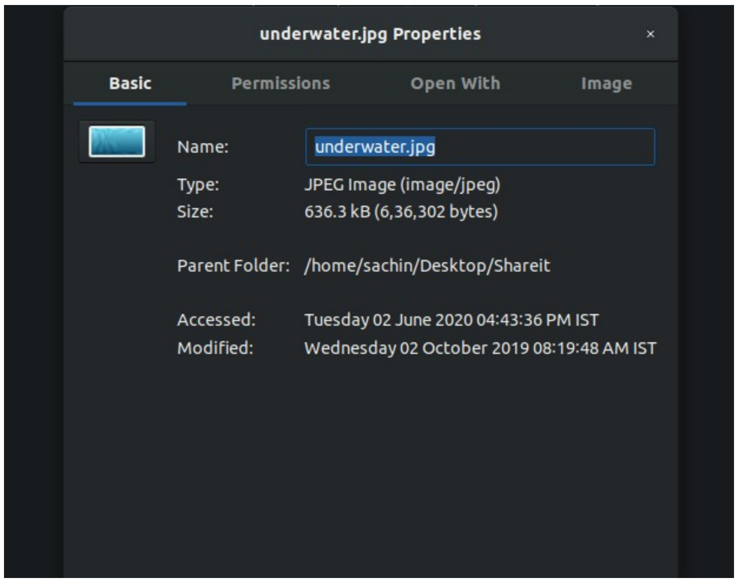
```
/home/sachin/PycharmProjects/FSlab/venv/bin/python /hom
Time taken in sec is: 98
Original size: 786432
Compressed size: 492000
Ratio compressed size / original size:
0.6256103515625
Compressed image size is 62.56% of the original image

Process finished with exit code 0
```

Fig 4.9 Console

The useful information such as time taken, compressed size, ratio are provided in the console

Screen Shot 10



Name: underwater.jpg
Type: JPEG Image (image/jpeg)
Size: 636.3 kB (6,36,302 bytes)

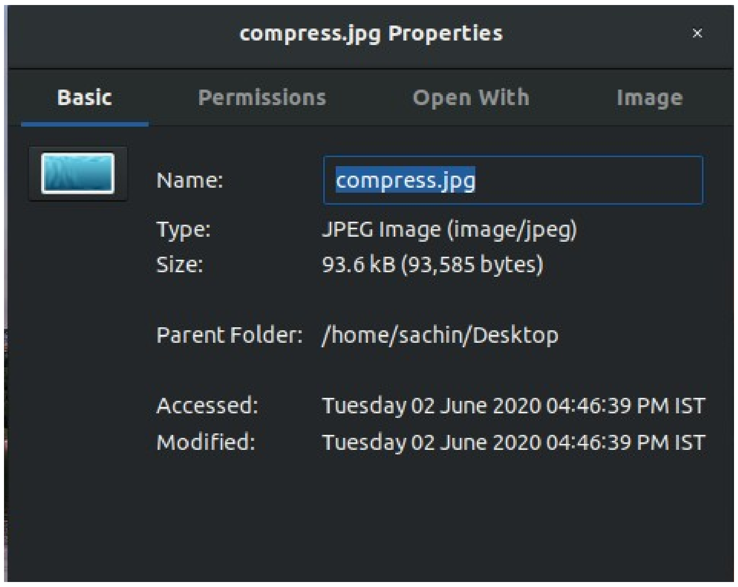
Parent Folder: /home/sachin/Desktop/Shareit

Accessed: Tuesday 02 June 2020 04:43:36 PM IST
Modified: Wednesday 02 October 2019 08:19:48 AM IST

Fig 4.10 Before compression

The properties of the image before image compression is shown for comparison

Screen Shot 11



Name: compress.jpg
Type: JPEG Image (image/jpeg)
Size: 93.6 kB (93,585 bytes)

Parent Folder: /home/sachin/Desktop

Accessed: Tuesday 02 June 2020 04:46:39 PM IST
Modified: Tuesday 02 June 2020 04:46:39 PM IST

Fig 4.11 After compression

The properties of the image after image compression is shown for comparison

Graphs -

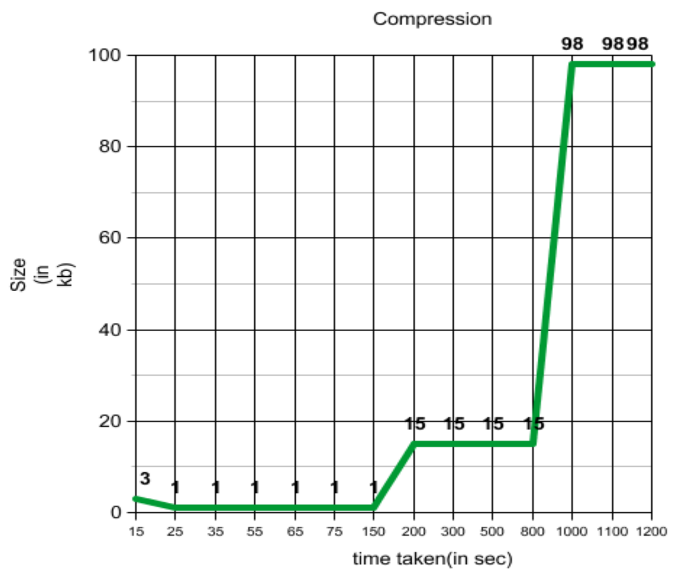


Fig 4.12 Graph which shows the duration of compression versus size of the file

SVD algorithm performs extremely well when image size in the range 1 Kb to 150 Kb. It takes duration of maximum 3 seconds to compress the image. When image size is in range 150 Kb to 1 Mb, SVD algorithm takes about 15 seconds to compress the image. When image size is taken more than 1 Mb, SVD algorithm’s performance becomes sluggish. Hence we can conclude that SVD algorithm is extremely good in handling small sized images, decently good in handling medium sized images and sluggish while handling large sized images.

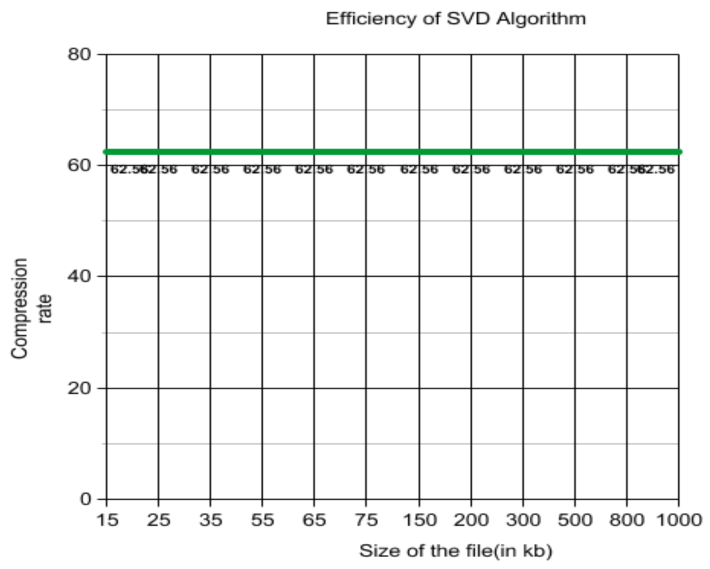


Fig 4.12 Graph which shows the efficiency of the algorithm

SVD algorithm is highly efficient in compressing images when compared to other similar compression algorithms. By the above graph we can conclude that compression rate is not dependent on the size of the image. SVD algorithm yields 62.2% compression rate for all the images. The compressed image looks decently good. The reason for the sharper contrast is because SVD is very pattern oriented, so natural patterns in the image may show up little stronger than the original image

Conclusion

I have successfully implemented image compression using SVD algorithm. From the above results we can conclude that SVD gives good compression results with less computational complexity. SVD algorithm is extremely good in handling small sized images, decently good in handling medium sized images and sluggish while handling large sized images. SVD algorithm yields 62.2% compression rate for all the images. The compressed image looks decently good. To achieve high value of compression ratio, image quality is to be sacrificed. Therefore it is required to select proper number for adding singular value to the matrices. Beside image compression, SVD also has application in noise reduction, face recognition, water marking, etc.

References

- Using SVD for image compression in python

  https://www.youtube.com/watch?v=SU851ljMIZ8.
- Github

  https://github.com/playandlearntocode/using-svd-for-image-compression-in-python-in-python.
- Image compression using SVD

  http://www.ijoart.org/docs/Image-Compression-using-Singular-Value-Decomposition.pdf
- Image compression using SVD

  http://www.math.utah.edu/~goller/F15_M2270/BradyMathews_SVDImage.pdf
- Docs.python.org – tkinter

  https://docs.python.org/3/library/tkinter.html
- Tutorialspoint - tkinter

  https://www.tutorialspoint.com/python/python_gui_programming.html
- How to program a GUI application

  https://www.youtube.com/watch?v=D8-snVfekto
- Creating input fields with Tkinter – python

  https://www.youtube.com/watch?v=7A_csP9drJw