

**Visvesvaraya Technological University
Belagavi-590 018, Karnataka**

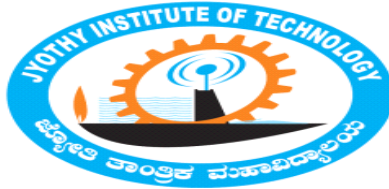


**A MINI PROJECT REPORT ON
“Image Compression using SVD Algorithm”**

**Mini Project Report submitted in partial fulfillment of the requirement for
The VI Semester File Structures Laboratory
[17CSL68]**

**Bachelor of Engineering
In
Information Science and Engineering**

**Submitted by
SUJITH UP [1JT17IS045]**



**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**”
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
Guide, Asst, Professor
Dept. Of ISE

External Viva Examiner

-
-

Dr. Harshvardhan Tiwari
Associate Professor and HOD
Dept. OF ISE

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 informations.

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

SERIAL NO.	DESCRIPTION	PAGE NO.
	Chapter 1	
1	Introduction	1 - 3
1.1	Introduction to File Structures	1
1.2	Introduction to File System	2
1.3	Introduction to File System	3
	Chapter 2	
2	Requirement analysis and design	4 - 5
2.1	Domain Understanding	4
2.2	Classification of Requirements	4
2.3	System Analysis	5
	Chapter 3	
3	Implementation	6 - 7
	Chapter 4	
4	Result and Analysis	8 - 9

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 backing 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 $m * n$ diagonol 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

The main object is to compress the given image. The outcome of this project is to ease the user for compressing images with a friendly, understandable GUI.

2.2 Classification of Requirements

User Requirements

OS : Windows / Linux

IDE : PyCharm

Software and Hardware Requirements

Programming Languages: Front End – Python Tkinter

Back End – Python

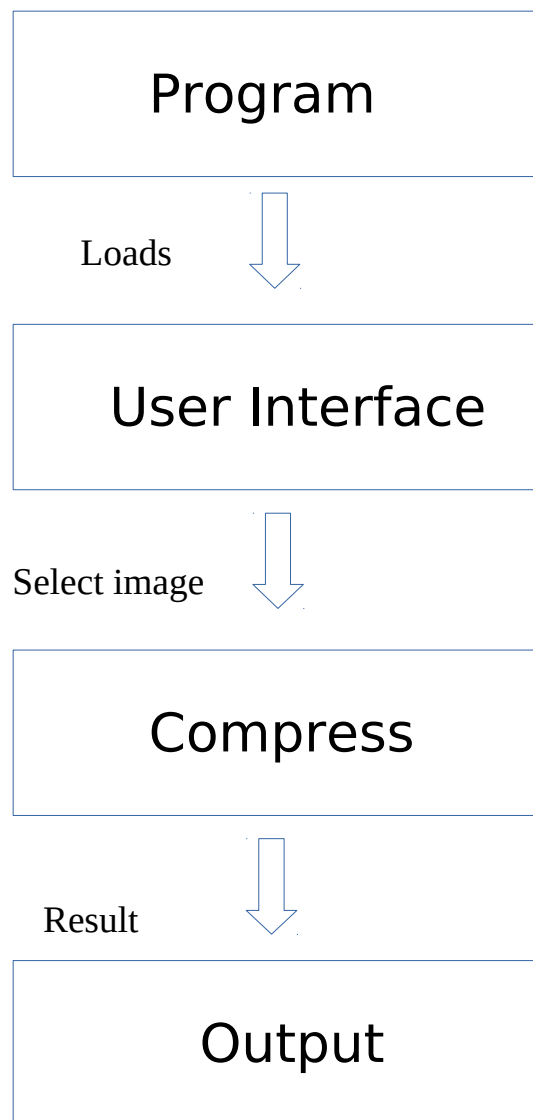
Hard Disk: 1GB Hard Disk

RAM: 1GB or more

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.

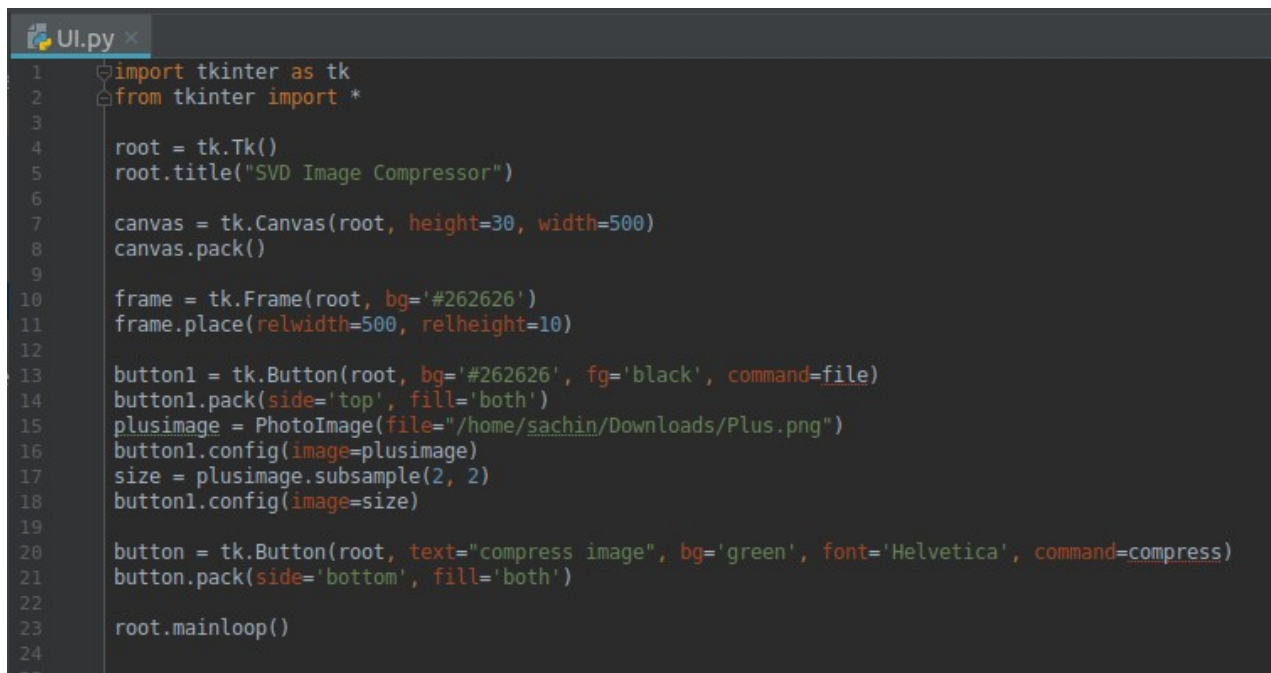


CHAPTER 3

IMPLEMENTATION

IMPLEMENTATION

User Interface

A screenshot of a code editor window titled 'Ui.py'. The code is a Python script using Tkinter to create a user interface for an 'SVD Image Compressor'. The script includes imports for Tkinter, initializes a root window, sets its title, and creates a canvas and a frame. It also defines two buttons: one with a plus icon to open a file and another labeled 'compress image' to perform the compression. The script ends with a call to the mainloop function.

```
1 import tkinter as tk
2 from tkinter import *
3
4 root = tk.Tk()
5 root.title("SVD Image Compressor")
6
7 canvas = tk.Canvas(root, height=30, width=500)
8 canvas.pack()
9
10 frame = tk.Frame(root, bg='#262626')
11 frame.place(relwidth=500, relheight=10)
12
13 button1 = tk.Button(root, bg='#262626', fg='black', command=file)
14 button1.pack(side='top', fill='both')
15 plusimage = PhotoImage(file="/home/sachin/Downloads/Plus.png")
16 button1.config(image=plusimage)
17 size = plusimage.subsample(2, 2)
18 button1.config(image=size)
19
20 button = tk.Button(root, text="compress image", bg='green', font='Helvetica', command=compress)
21 button.pack(side='bottom', fill='both')
22
23 root.mainloop()
24
25
```

Fig. 3.1 Here we can see the code where the user interface is been created.

The code we used here is in Python where we have used Tkinter.

Main Program

```
import numpy
from PIL import Image

def openImage(imagePath):
    imOrig = Image.open(imagePath)
    im = numpy.array(imOrig)
    aRed = im[:, :, 0]
    aGreen = im[:, :, 1]
    aBlue = im[:, :, 2]
    return [aRed, aGreen, aBlue, imOrig]

def compressSingleChannel(channelDataMatrix, singularValuesLimit):
    uChannel, sChannel, vhChannel = numpy.linalg.svd(channelDataMatrix)
    aChannelCompressed = numpy.zeros((channelDataMatrix.shape[0], channelDataMatrix.shape[1]))
    k = singularValuesLimit
    leftSide = numpy.matmul(uChannel[:, 0:k], numpy.diag(sChannel)[0:k, 0:k])
    aChannelCompressedInner = numpy.matmul(leftSide, vhChannel[0:k, :])
    aChannelCompressed = aChannelCompressedInner.astype('uint8')
    return aChannelCompressed

print('Image Compression using SVD :')
start = timeit.default_timer()
aRed, aGreen, aBlue, originalImage = openImage('/home/sachin/Downloads/me.jpeg')
imageWidth = 512
imageHeight = 512
singularValuesLimit = 160
aRedCompressed = compressSingleChannel(aRed, singularValuesLimit)
aGreenCompressed = compressSingleChannel(aGreen, singularValuesLimit)
aBlueCompressed = compressSingleChannel(aBlue, singularValuesLimit)
imr = Image.fromarray(aRedCompressed, mode=None)
img = Image.fromarray(aGreenCompressed, mode=None)
imb = Image.fromarray(aBlueCompressed, mode=None)
newImage = Image.merge("RGB", (imr, img, imb))
mr = imageHeight
mc = imageWidth
originalSize = mr * mc * 3
compressedSize = singularValuesLimit * (1 + mr + mc) * 3
stop = timeit.default_timer()
print('Original size: %d' % originalSize)
print('Compressed size: %d' % compressedSize)
print('Ratio compressed size / original size:')
ratio = compressedSize * 1.0 / originalSize
print(ratio)
print('Compressed image size is ' + str(round(ratio * 100, 2)) + '% of the original image ')
print('Time: ', stop - start)
print('DONE!!')
```

CHAPTER 4

RESULT AND ANALYSIS

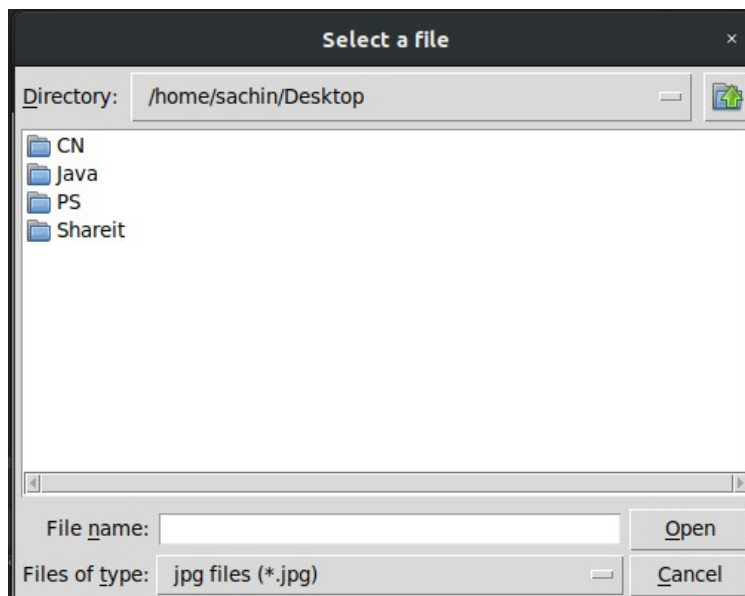
Result And Analysis

Screen Shot 1 (Main Screen)



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

Screen Shot 2



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 opened through browse.

Screen Shot 3



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

Screen Shot4 (Result)

```
/home/sachin/PycharmProjects/FSlab/venv/bin/python /home
Original size: 786432
Compressed size: 492000
Ratio compressed size / original size:
0.6256103515625
Compressed image size is 62.56% of the original image
Time: 0.6490841240010923
```