Binary Image Compression Algorithm

Prepared by

Registration number	Name	Email
16BCE0098	Nishant Rohan Rodrigues	Rohan.rodrigues2016@vitstudent.ac.in
16BCE0626	Sukrit Bedi	Sukrit.bedi2016@vitstudent.ac.in
16BCE0335	Vibhav Koppula Reddy	Vibhavkoppula.reddy2016@vitstudent.ac.in

Abstract:

Image comperssion by manipulating a binary image and fitting it into integer strings. Using a binary string, we generate a single integer that tabes up lesses space than the entire binary string.

String size > Integer size

To furthur the comperssion, we use the median value to divide all the rest of the integers. Only the quotients and remainders are sotred and this allows us the reduce the size to the integers.

Using the above mentioned techniques, it is now possible to compress the entire binary image without any loss of data.

Keywords: Binary image, Image compression, Lossless comperssion

Main setup:

```
from compress import display, compress, extract
import os

def read_image(filename, splt=False):
    with open(filename, 'r') as f:
        data = f.readlines()
    if splt:
        return [map(int, list(i.strip())) for i in data]
    else:
        return [map(int, i.split()) for i in data[2:]], int(data[1]), int(data[0])

if __name__ == "__main__":

    data = []
    filename = "batman.txt"

    data = read_image("images/" + filename, True)
    # print data
    display(data)

    compress(data, filename)
    data, med, size = read_image("compress/cmp_" + filename)
    # print data, med, size
    data = extract(data, filename, med, size)
    # print data
    display(data)

    original_size = os.path.getsize("images/" + str(filename))
    compressed_size = os.path.getsize("compress/cmp_" + str(filename))
    print "File size"
    print "Original Size : ", original_size
    print "Compressed Size : ", compressed_size
    print "Compression factor : ", float(original_size)/compressed_size
```

Lossless compression and extraction:

```
import matplotlib.pyplot as plt
from numpy import median

def display(data):

    plt.imshow(data, cmap='binary')
    plt.title("Image")
    plt.show()

def compress(data, filename):

    cmp = []
    for each in data:
        lst = map(str, each)
        cmp.append(int(''.join(lst), 2))

    med = int(median(cmp))
    cmp = [[i//med, i%med] for i in cmp]

    f = open("compress/cmp_" + filename, 'w')
    f.write("%s\n" % str(len(data[0])))
    f.write("%s\n" % str(med))
    for item in cmp:
        f.write("%s\n" % (str(item[0]) + " " + str(item[1])))
    f.close()

def extract(data, filename, med, size):

    lst = []
    for i in data:
        val = (med*i[0]) + i[1]
        row = "{0:b}".format(val).zfill(size)
        lst.append(map(int, list(row)))
    return lst
```

Python libraries used:

os – Run operating system commmands through Python for finding file sizes

matplotlib – Python library used to plot and represent data in charts, images and graphs

numpy – numerical python library to make computations fast and efffecient

Testing:

```
11111111111111111111111111
11111100111111111100111111
11110001111100111110001111
11000001111000011110000011
10000000111000011100000001
10000000000000000000000001
00000000000000000000000000
00000000000000000000000000
10000000000000000000000001
10000110001000010001100001
11001111111100111111110011
11100111111100111111100111
11111111111111111111111111
```

Fig: Image in binary uncompressed format

```
26
50825091
1 16283772
1 15497148
1 12601356
1 0
0 33785601
0 33554433
0 0
0 0
0 33554433
0 35161185
1 3688560
1 9980004
1 16283772
```

Fig: Image in compressed format

Output Samples:

Sample test 1:

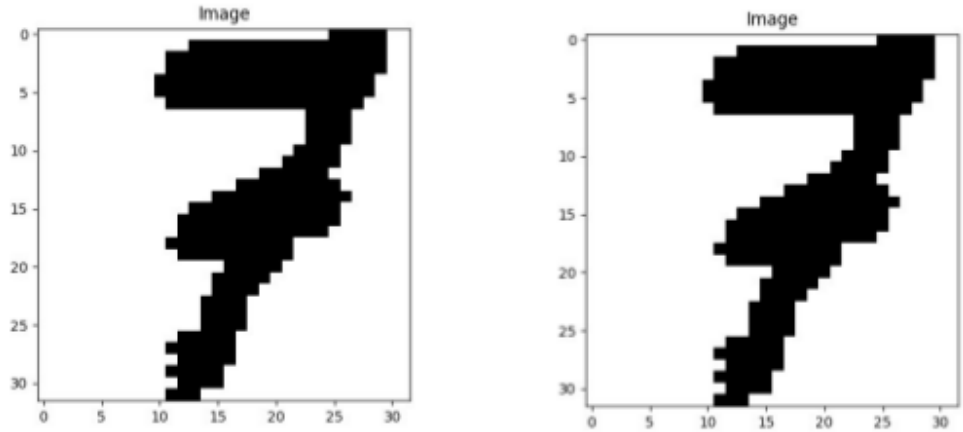

Fig: Initial image, Image after comperssion

```
Python - main.py:27 ✔

File size
Original Size :  1088
Compressed Size :  253
Compression factor :  4.30039525692
[Finished in 113.693s]
```

Fig: Compression factor for above image

Sample test 2:

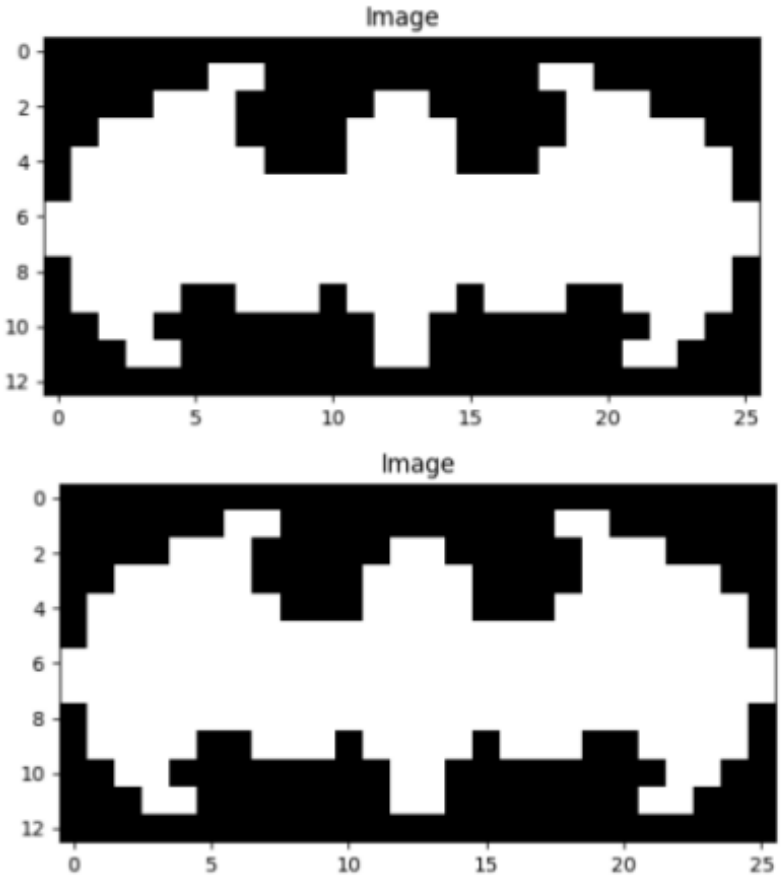

Fig: Initial image, Image after comperssion

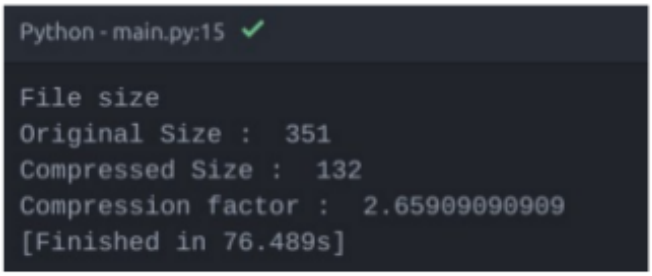

Fig: Compression factor for above image