



ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ



**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΕΦΑΡΜΟΣΜΕΝΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

ΕΠΛ445: Ψηφιακή Επεξεργασία Εικόνας

Εργαστήριο

5^η Εργαστηριακή Άσκηση

Ομάδα:

Βάκη Νικόλας 1025806
Δημητρίου Καρολίνα 1035986
Ηλιάδη Έλλη 1018368

Ημερομηνία Παράδοσης
01 Σεπτεμβρίου 2022

1. Εισαγωγή - Στόχος

Στόχος της άσκησης είναι η εφαρμογή των βημάτων Βασικού Συστήματος JPEG και η σύγκριση των εικόνων με multiplying factors 0.5, 1.0 και 2.0. Η μέθοδος αυτή αποτελεί διαδικασία συμπίεσης εικόνων με απώλεια.

2. Μεθοδολογία

Εφαρμογή αλγορίθμου συμπίεσης όπου σπάζουμε την εικόνα σε πολλαπλά 8x8 blocks, εφαρμόζουμε DCT σε κάθε block ξεχωριστά και στη συνέχεια κβαντοποιούμε:

```
def imgCompression(self, factor):
    dpcm = []
    jpeg = []
    prevDc = 0
    iHeight, iWidth = img.shape[:2]
    factor = float(factor)
    q_table_factor = np.multiply(q_table, factor)
    for startY in range(0, iHeight, 8):
        for startX in range(0, iWidth, 8):
            # Blocking 8x8 and DCT
            block = img[startY:startY + 8, startX:startX + 8]
            # apply DCT for a block
            block_f = np.float32(block)
            dst = cv2.dct(block_f)
            # Quantization of DCT coefficients
            block_q = np.floor(np.divide(dst, q_table_factor) + 0.5)
```

Στη συνέχεια εφαρμόζουμε zigzag σε κάθε block και dcmp, παίρνουμε το dc στοιχείο και το αφαιρούμε από το zigzag table:

```
# Αναδιάταξη δεδομένων Zig-Zag
zigzag_table = (zigzag(block_q))
# DPCM for DC values
dpcm.append((zigzag_table[0] - prevDc))
prevDc = zigzag_table[0]
#RLC for AC values
zigzag_table.pop(0)
```

Ακολουθεί εφαρμογή RLC υπολογίζοντας πόσα μηδενικά υπάρχουν στον πίνακα zigzag και εύρεση του αριθμού άνισου του μηδενός ώστε να προστεθεί μαζί με τα μηδενικά στον πίνακα RLC. Τοποθετούνται ακόμα δύο μηδενικά ώστε να δείξουμε το τέλος του πίνακα. Τέλος εφαρμόζουμε την συμπιεσμένη εικόνα RLC με τον αλγόριθμο Huffman:

```
#RLC for AC values
zigzag_table.pop(0)
rlc = []
count = 0
# Apply the RLC for all the elements from the zigzag table
for i in zigzag_table:
    if i == 0:
        count += 1
        continue
    else:
        rlc.append(count)
        rlc.append(i)
        count = 0
rlc.append(0)
rlc.append(0)
# huffman encoding
jpeg.append(huffmanList(rlc))
# getting compressed img
jpeg.append(huffmanList(dpcm))
```

Παίρνουμε τα στοιχεία της συμπιεσμένης εικόνας εκτός από τη τελευταία στήλη του πίνακα και εφαρμόζουμε decode με τον αλγόριθμο Huffman για να πάρουμε όλες τις τιμές. Εάν το decode επιστρέφει 0 τότε προσθέτουμε στον πίνακα του block 64 μηδενικά.

```
# Decompression of img
dComp = []
for i in jpeg[:-1]:
    current = [float(j) for j in decode(i[0], i[1])]
    array = []
    if len(current) == 0:
        for l in range(63):
            array.append(0)
```

Για να πάρουμε τις υπόλοιπες AC τιμές του block για τα στοιχεία που επιστρέφει ο Huffman αλγόριθμος στο decode εφαρμόζουμε το εξής:

```
for k in range(0, len(current), 2):
    # if this is the last element
    if int(current[k]) == 0 and int(current[k + 1]) == 0:
        # apply the rest with 0 until 64 is completed.
        for l in range(63 - len(array)):
            array.append(0.0)
        break
```

Εάν το πρώτο στοιχείο που παίρνουμε δεν είναι 0 τότε για όλο το μήκος του στοιχείου αυτού προσθέτω 0. Και στην συνέχεια προσθέτω το δεύτερο στοιχείο που παίρνουμε από τον πίνακα όπου δεν είναι 0. Τέλος βάζω το block μέσα στην αποσυμπιεσμένη εικόνα.

```
break
if int(current[k]) != 0:
    # append the number of zeros we had before the number
    for l in range(int(current[k])):
        array.append(0.0)
    # append the number after the zeros
    array.append(current[k + 1])
dComp.append(array)
c = 0
```

Μόλις τελειώσουμε από όλα τα blocks παίρνουμε πίσω το dc πίνακα κάνοντας decode με τον εξής τρόπο:

```
c = 0
# decode the DC table
dc = decode(jpeg[-1][0], jpeg[-1][1])
for i in dc:
    if c == 0:
        dComp[0].insert(0, float(i))
    else:
        dComp[c].insert(0, float(i) + float(dComp[c - 1][0][0]))
    dComp[c] = np.array(dComp[c])
    # inverse zig zag
    dComp[c] = inverse_zigzag(dComp[c].astype(int))
    c += 1
```

Για όλα τα blocks που πήρα πίσω στην πρώτη φάση της συμπίεσης κάνω το βήμα του quantization και τον αντίστροφο του DCT. Για κάθε block κάνουμε αναδιοργάνωση για να βρεθούμε πίσω στην αρχική του μορφή.

```
# our decompressed image table
img_comp = np.empty(shape=(iHeight, iWidth))
temp = []
for i in dComp:
    # inverse the quantization for each block
    iblock_q = np.floor(np.multiply(i, q_table_factor) + 0.5)
    # inverse the dct
    inv_dct = cv2.idct(iblock_q)
    np.place(inv_dct, inv_dct > 255.0, 255.0)
    np.place(inv_dct, inv_dct < 0.0, 0.0)
    temp.append(inv_dct)
```

```
k = 0
# reorganise the block to create the image in its original structure.
for startY in range(0, iHeight, 8):
    for startX in range(0, iWidth, 8):
        img_comp[startY:startY + 8, startX:startX + 8] = temp[k]
        k += 1
```

Τέλος εφαρμόζω τους τύπους:

```
if(path[-3:]=="tif"):
    cv2.imwrite('original.tif', img)
    img_size = os.path.getsize('original.tif')

elif(path[-3:]=="png"):
    cv2.imwrite('original.png', img)
    img_size = os.path.getsize('original.png')

else:
    print("wrong image format")

#getting metrics for the compressed image
cv2.imwrite('compressed.jpg', img_comp)
comp_size = os.path.getsize('compressed.jpg')
sumMSE = 0
for i in range(0, iHeight):
    for j in range(0, iWidth):
        sumMSE = sumMSE + math.pow(img[i][j] - img_comp[i][j], 2)
sumMSE = sumMSE/(iHeight*iWidth)
pnsr = 10 * np.log10(math.pow(255, 2) / sumMSE)
ssim_value = ssim(img, img_comp, data_range=img.max() - img.min())
cr = img_size / comp_size
```

3. Αποτελέσματα

1η εικόνα:

A) factor: 0.5

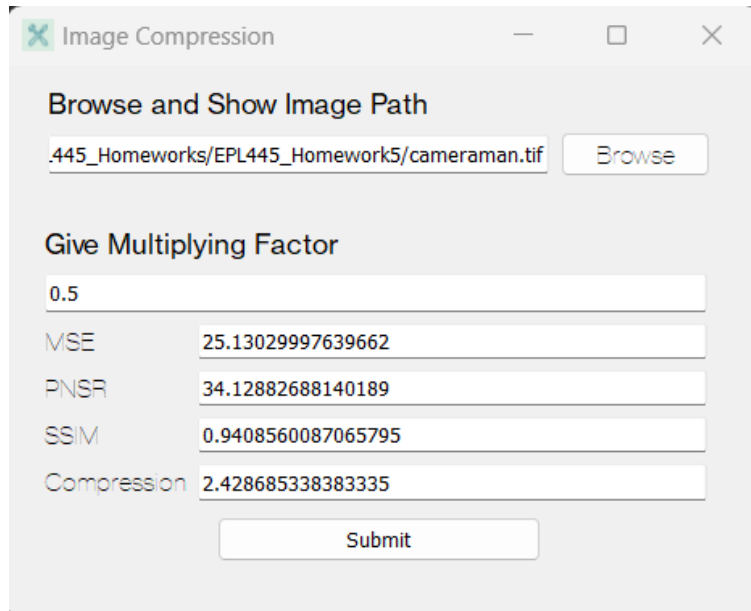


Image Compression

Browse and Show Image Path

.445_Homeworks/EPL445_Homework5/cameraman.tif Browse

Give Multiplying Factor

0.5

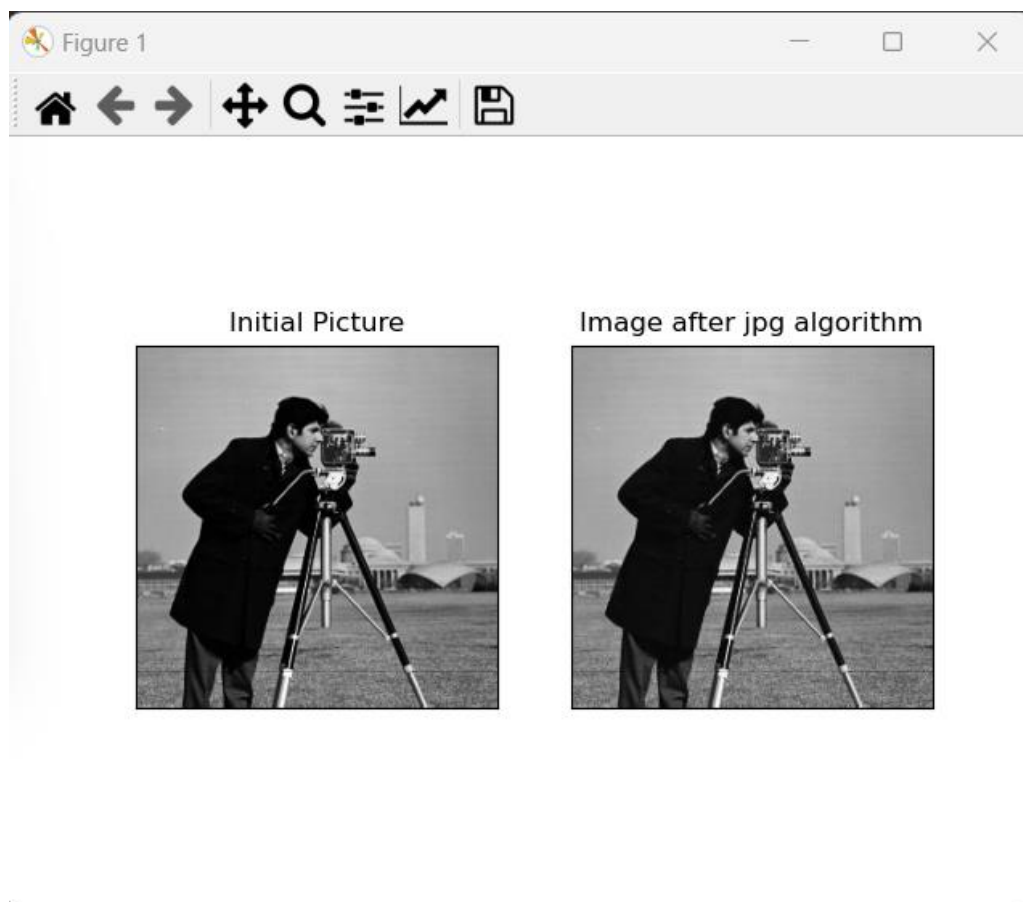
MSE 25.13029997639662

PNSR 34.12882688140189

SSIM 0.9408560087065795

Compression 2.428685338383335

Submit



Αρκετά καλή ποιότητα στην συμπιεσμένη εικόνα. Η διαφορά με την αρχική εικόνα δεν είναι αισθητή οπτικά.

B) factor: 1.0

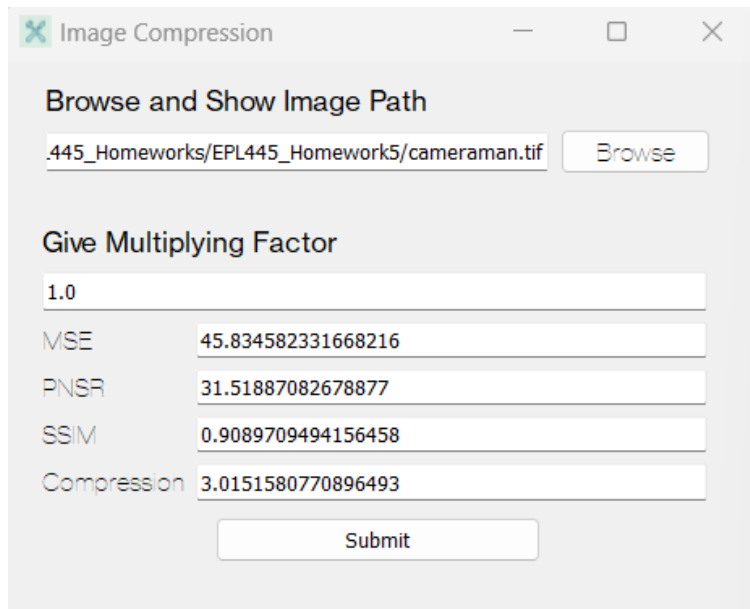


Image Compression

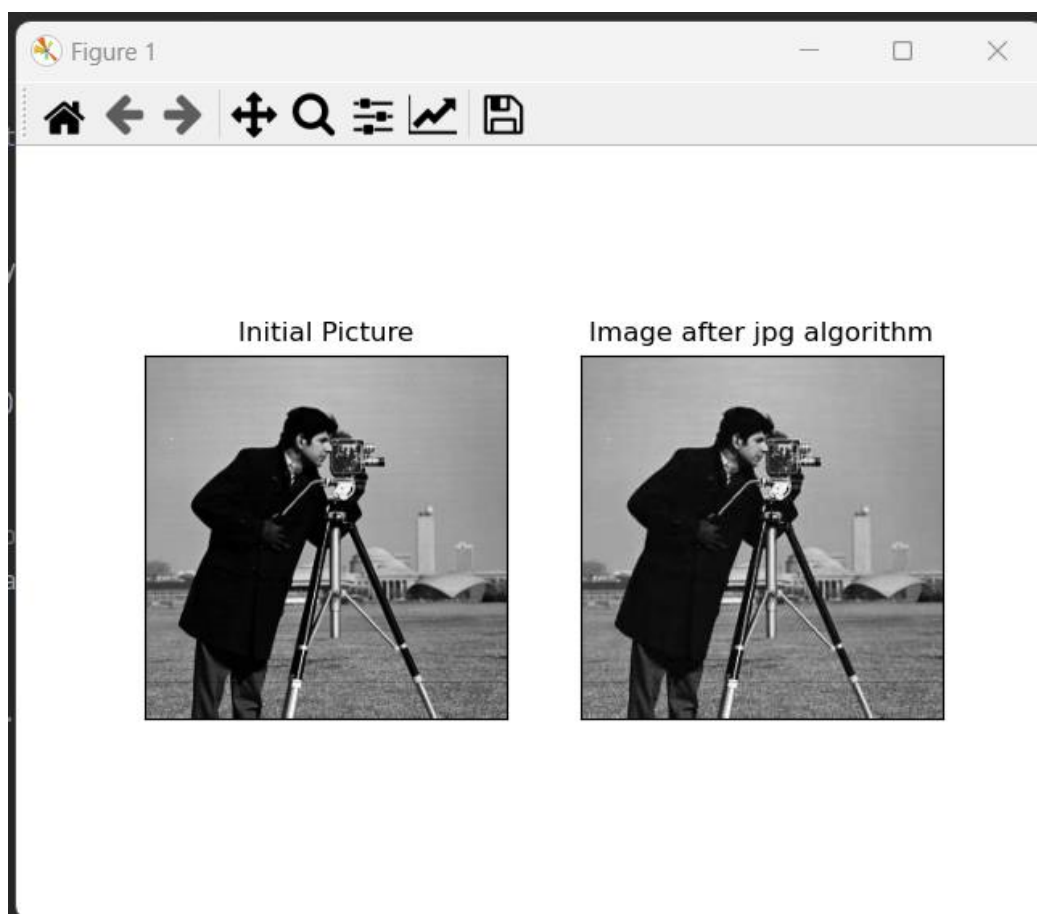
Browse and Show Image Path

.445_Homeworks/EPL445_Homework5/cameraman.tif

Give Multiplying Factor

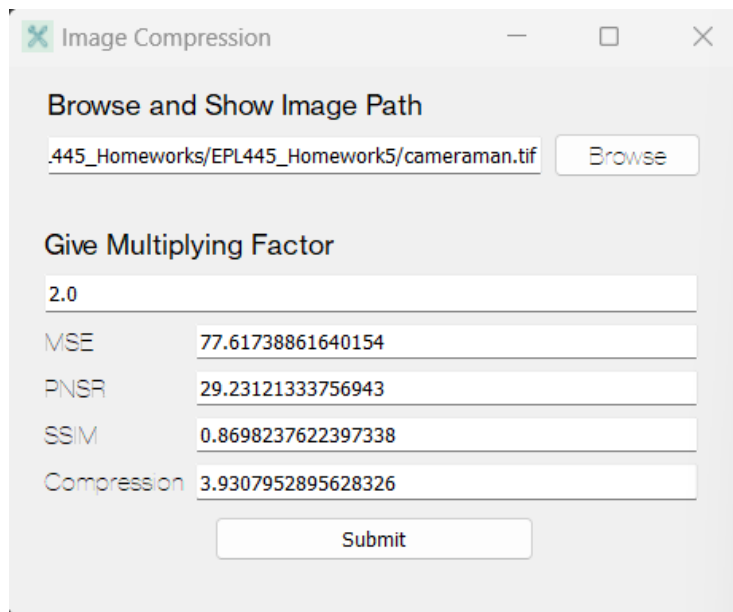
1.0

MSE	45.834582331668216
PNSR	31.51887082678877
SSIM	0.9089709494156458
Compression	3.0151580770896493

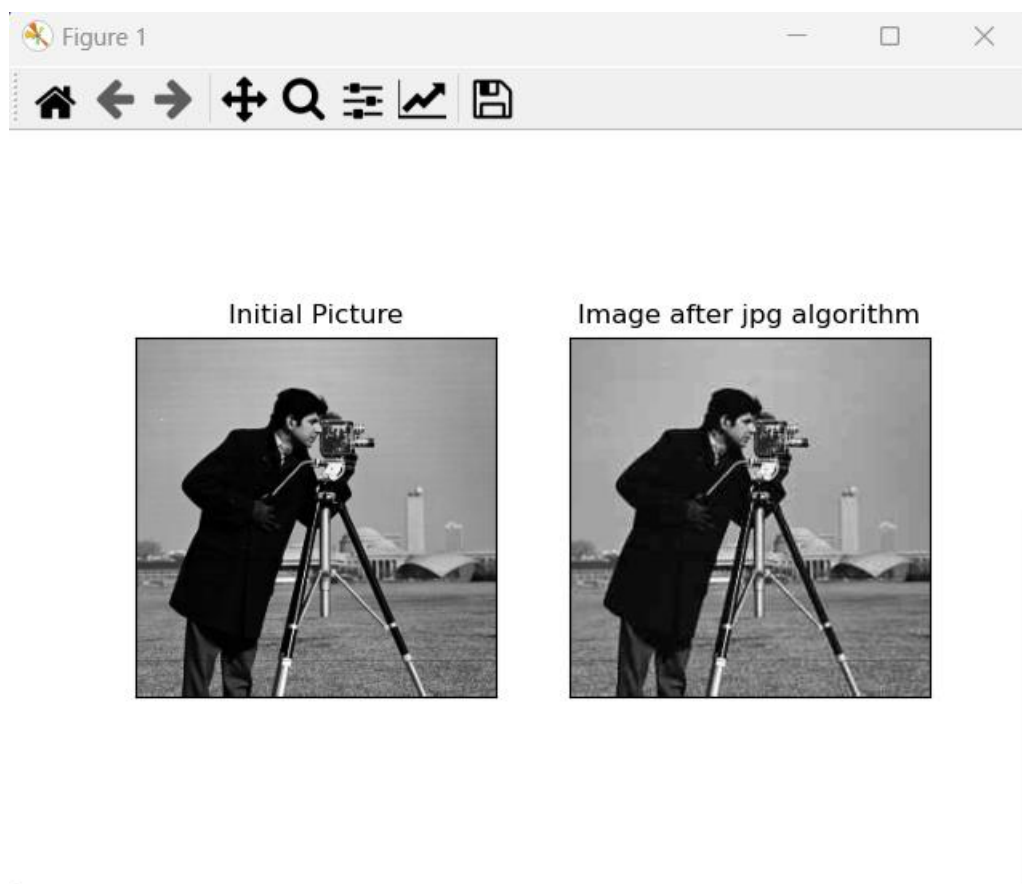


Οπτικά δεν φαίνεται κάποια διαφορά μεταξύ multiplying factor 0.5 και 1.0.

C) factor: 2.0



Give Multiplying Factor	
MSE	77.61738861640154
PNSR	29.23121333756943
SSIM	0.8698237622397338
Compression	3.9307952895628326



Παρατηρούμε διακριτά blocks στην εικόνα, σε σύγκριση με multiplying factor 0.5 και 1.0.

2η εικόνα:

A) factor: 0.5

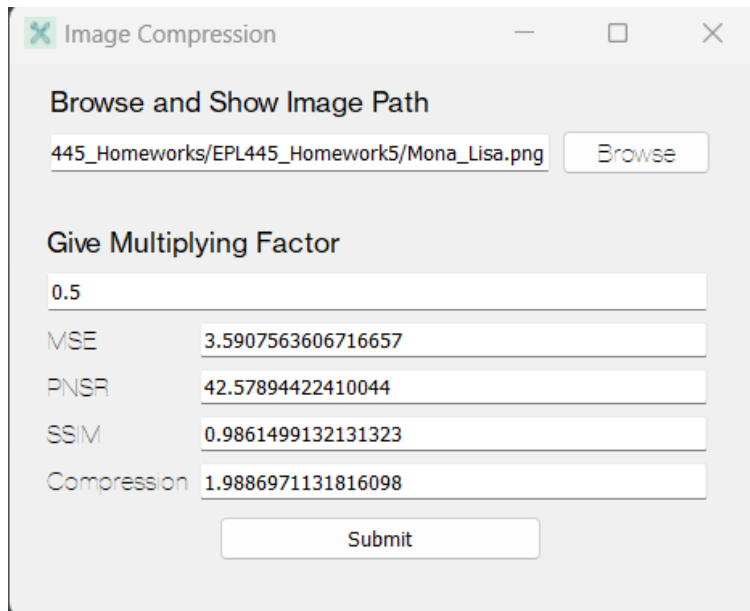


Image Compression

Browse and Show Image Path

445_Homeworks/EPL445_Homework5/Mona_Lisa.png Browse

Give Multiplying Factor

0.5

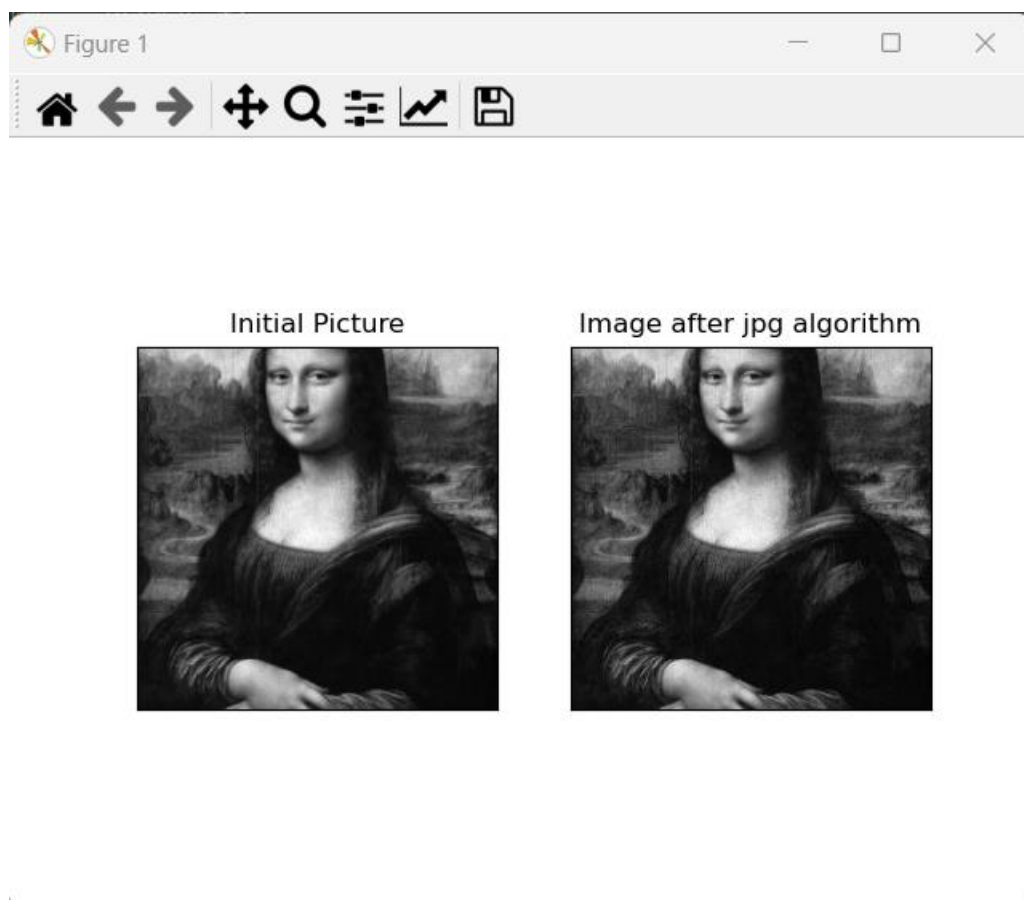
MSE 3.5907563606716657

PSNR 42.57894422410044

SSIM 0.9861499132131323

Compression 1.9886971131816098

Submit



Αρκετά καλή ποιότητα στην συμπιεσμένη εικόνα. Η διαφορά με την αρχική εικόνα δεν είναι αισθητή οπτικά.

B) Factor: 1.0

Image Compression

Browse and Show Image Path

445_Homeworks/EPL445_Homework5/Mona_Lisa.png

Give Multiplying Factor

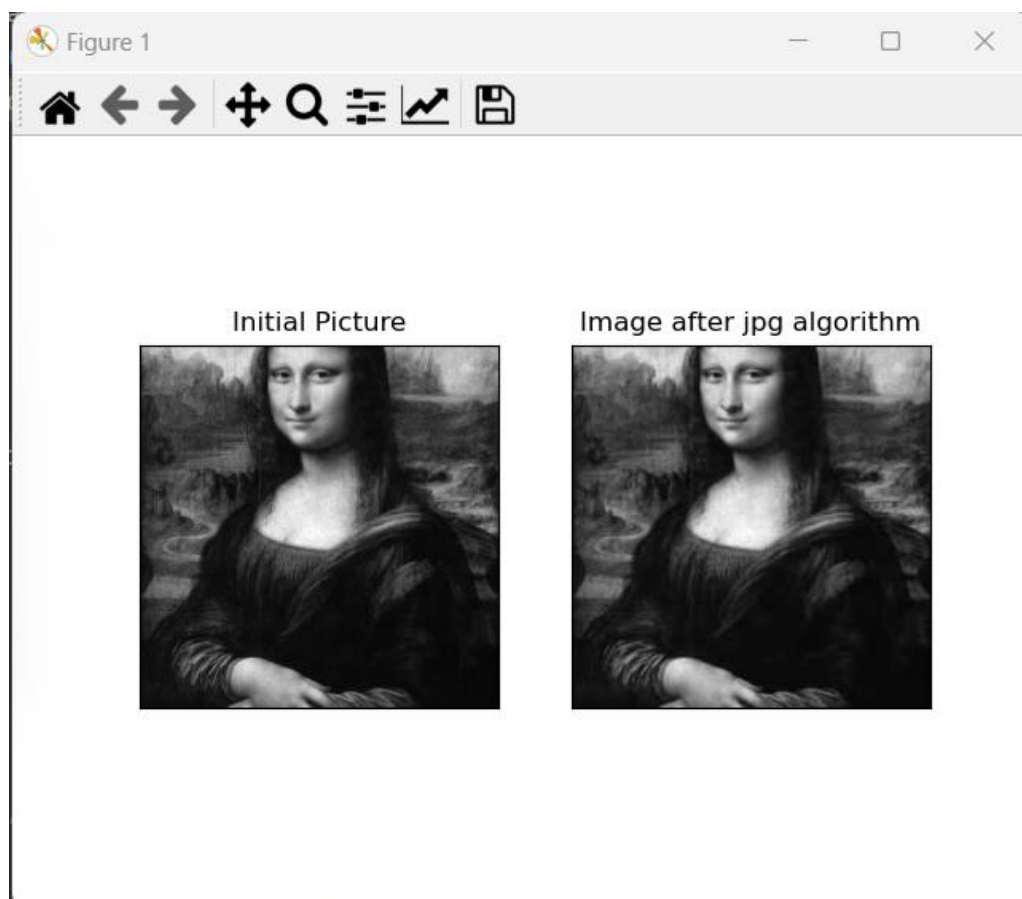
1.0

MSE 17.223768990199257

PNSR 35.76952168887276

SSIM 0.9269067762406913

Compression 3.0475150191152376



Οπτικά δεν φαίνεται κάποια διαφορά μεταξύ multiplying factor 0.5 και 1.0.

C) Factor: 2.0

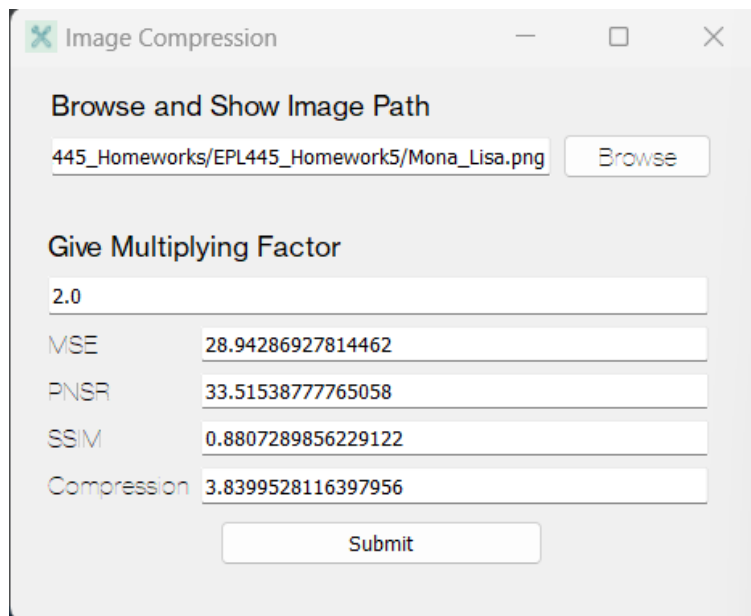
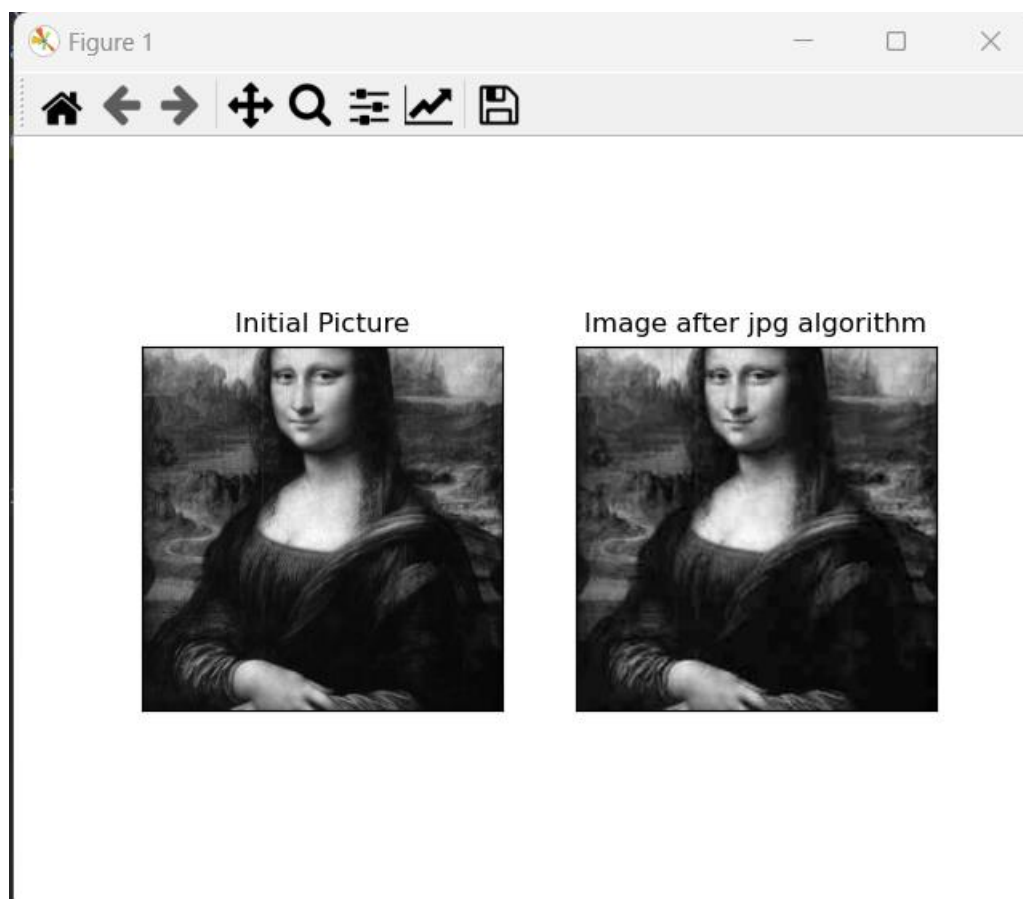


Image Compression window showing the process of compressing an image. The window has a title bar with a close button and standard window controls. The main content area is divided into sections. The first section, 'Browse and Show Image Path', contains a text field with the path '445_Homeworks/EPL445_Homework5/Mona_Lisa.png' and a 'Browse' button. The second section, 'Give Multiplying Factor', contains a text field with the value '2.0'. Below this, there are four rows of metrics, each with a label and a corresponding value in a text field: MSE (28.94286927814462), PNSR (33.51538777765058), SSIM (0.8807289856229122), and Compression (3.8399528116397956). A 'Submit' button is located at the bottom of the window.

Metric	Value
MSE	28.94286927814462
PNSR	33.51538777765058
SSIM	0.8807289856229122
Compression	3.8399528116397956



Παρατηρούμε διακριτά blocks στην εικόνα, σε σύγκριση με multiplying factor 0.5 και 1.0.

4. Συμπεράσματα - Σχολιασμός Αποτελεσμάτων

Από τα πιο πάνω πειράματα παρατηρούμε ότι όσο αυξάνεται το multiplying factor, το MSE και το Compression Ratio αυξάνονται ενώ τα PSNR και SSIM μειώνονται. Αυτά τα αποτελέσματα ήταν αναμενόμενα αφού από την θεωρία ξέρουμε ότι όσο αλλοιώνετε η φωτογραφία, η διαφορά με την αρχική εικόνα μεγαλώνει, αρά αυξάνεται και το MSE. Επίσης από την θεωρία γνωρίζουμε ότι όσο πιο μεγάλο είναι το PSNR τόσο καλύτερη και η ποιότητα της εικόνας, κάτι που φαίνεται και στα πειράματα. Το SSIM μειώνεται αφού η ποιότητα της συμπιεσμένης εικόνας είναι χειρότερη από την αρχική. Το compression rate αυξάνεται επειδή μειώνεται το μέγεθος της εικόνας.

Ο σκοπός της συμπίεσης μιας εικόνας είναι η ελαχιστοποίηση στο χώρο που χρειάζεται για να αποθηκευτεί και παράλληλα να διατηρεί μια καλή οπτική στην ποιότητα της. Παρόλο που η συμπίεση εικόνας με απώλεια είναι πιο αποτελεσματική και διαδεδομένη, σε μερικές εικόνες που δοκιμάσαμε όπως logos, η εικόνα επηρεάστηκε αρκετά από την συμπίεση. Στην φωτογραφία που χρησιμοποιήθηκε πιο πάνω παρατηρούμε πως δεν διακρίνονται οπτικά οι απώλειες πληροφορίας της, επομένως είναι μία επιτυχής συμπίεση εικόνας.