Παρατήρηση σχετικά με το Α' μέρος της 1ης Εργασίας

Χρειάζεται προσοχή στην διαίρεση των τιμών του πίνακα με την αντίστοιχη ακέραιη τιμή ώστε να βγει το σωστό αποτέλεσμα, δεδομένου ότι οι τιμές του πίνακας I της εικόνας είναι τύπου `uint8`.

Καταρχήν, όταν έχουμε την αρχική εικόνα, όπως γνωρίζουμε αυτή έχει για κάθε pixel 256 διαφορετικές τιμές του γκρι [0..255].

Εάν θέλουμε να κάνουμε την εικόνα να έχει 128 διαφορετικές τιμές του γκρι τότε το σκεπτικό είναι να διαιρέσουμε τις τιμές του πίνακα I δια 2 (και φυσικά να χρησιμοποιήσουμε και το κατάλληλο `colormap` με 128 τόνους του γκρι, ώστε να εμφανιστεί σωστά η εικόνα).

Στο άλλο άκρο, εάν θέλουμε να μετατρέψουμε την εικόνα σε bitmap με δύο τόνους του γκρι τότε η σκέψη είναι να διαιρέσουμε τις τιμές του πίνακα I δια 128 (με το αντίστοιχο πάλι `colormap`).

Η παρατήρηση αφορά στο ΠΩΣ θα κάνουμε αυτή την ΑΚΕΡΑΙΑ διαίρεση.

Ας πάρουμε ένα πιο απλό παράδειγμα μιας «εικόνας» I με 16 διαφορετικά επίπεδα του γκρι που έχει τις εξής μόνο τιμές

```
>> I=uint8(0:15)
I =
  1×16 uint8 row vector
    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
>>
```

Πρακτικά δηλαδή, ένα «pixel» για κάθε τιμή graylevel. Εάν θέλουμε να μετατρέψουμε την εικόνα σε bitmap διαιρώντας με 8 (με το ίδιο σκεπτικό που παραπάνω την 256 graylevel θα την διαιρούσαμε με 128) και γράψουμε

```
>> B=I/8
B =
  1×16 uint8 row vector
   0   0   0   0   1   1   1   1   1   1   1   1   2   2   2   2
>>
```

τότε το αποτέλεσμα είναι λάθος διότι εμφανίζονται 3 διαφορετικά graylevels 0, 1 και 2!

Ο λόγος είναι ότι για να υπολογιστεί το ΑΚΕΡΑΙΟ αποτέλεσμα (δεδομένου ότι το I είναι τύπου `uint8` οπότε και το αποτέλεσμα B θα είναι τύπου `uint8`) γίνεται στρογγυλοποίηση στον ΠΛΗΣΙΕΣΤΕΡΟ ακέραιο, (πρακτικά η εντολή `round`).

Πράγματι, αυτό μπορούμε να το δούμε πιο αναλυτικά εάν μετατρέψουμε τα δεδομένα σε double για να δούμε τι ακριβώς συμβαίνει:

```
>> A=double(I)
A =
     0     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15

>> A/8
ans =
         0    0.1250    0.2500    0.3750    0.5000    0.6250    0.7500    0.8750    1.0000    1.1250
1.2500    1.3750    1.5000    1.6250    1.7500    1.8750

>> round(A/8)
ans =
     0     0     0     0     1     1     1     1     1     1     1     1     2     2     2     2
>>
```

Αυτό είναι ίδιο με το λάθος αποτέλεσμα `B=I/8` που είδαμε παραπάνω. Για να πάρουμε το σωστό αποτέλεσμα θα πρέπει να μην κάνουμε στρογγυλοποίηση στον πλησιέστερο ακέραιο (όπως γίνεται με την `B=I/8`) αλλά στρογγυλοποίηση στον ΜΙΚΡΟΤΕΡΟ ακέραιο (εντολή `fix`):

```
>> fix(A/8)
ans =
     0     0     0     0     0     0     0     0     1     1     1     1     1     1     1     1
>>
```

που δίνει και το σωστό αποτέλεσμα. Αυτή την λειτουργία μπορείτε να την κάνετε

- είτε απευθείας, ενσωματώνοντας όλα τα παραπάνω σε μια εντολή:

```
>> B=uint8(fix(double(I)/8))
B =
  1×16 uint8 row vector
   0   0   0   0   0   0   0   0   1   1   1   1   1   1   1   1
>>
```

- είτε χρησιμοποιώντας την έτοιμη συνάρτηση `idivide`:

```
>> B=idivide(I,8)
B =
  1×16 uint8 row vector
   0   0   0   0   0   0   0   0   1   1   1   1   1   1   1   1
>>
```

Τάσος Κεσίδης