Πανεπιστήμιο Αιγαίου

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

321 - 6555 Πολυμέσα

Διδάσκων: Καρύμπαλη Ειρήνη

2η Άσκηση

Εργαστηριακοί Συνεργάτες: Καρύμπαλη Ειρήνη

Ευκαρπίδης Κωνσταντίνος

icsd15051

Κωστούλας Δημήτριος

icsd13095

Σάμος, 2/12, 2018

Περιεχόμενα

Ευκαρπίδης Κωνσταντίνος – icsd15051 , Κωστούλας Δημήτριος – icsd13095

1 Run Length Encoder

Στο πρώτο ερώτημα της 2ης άσκησης μας ζητήθηκε να υλοποιήσουμε τον αλγόριθμο run length encoder. Ως είσοδο η συνάρτηση μας παίρνει μια grayscale εικόνα και ως έξοδο επιστρέφει την κωδικοποίηση της. Η κωδικοποίηση γίνεται σε ένα πίνακα ο οποίος γίνεται append με το πέρασμα τιμών κάθε φορά. Σε περίπτωση επανάληψη αριθμών 3 ή περισσότερες φορές τότε θεωρούμε οτι υπάρχει όφελος κωδικοποίησης τους. Έτσι αν δηλαδή έχουμε 5 φορές στην σειρά τον αριθμό 100, τότε στον πίνακα encoded θα περάσουμε το εξής : “5 100 -1”. Αυτό σημαίνει οτι το 100 εμφανίζεται 5 συνεχόμενες φορές. Το -1 το χρησιμοποιήμου ώς διαχωριστικό για το μετέπειτα ερώτημα γιατί δεν αποτελεί καμία τιμή χρωματικής συνιστώσας (0-255).

```
for i = 1 : rows
    for j = 2 : columns
        if image(i,j) == image(i,j-1)
            counter=counter+1;
          %αν τελείωσε η επαναλάψη των ίδιων τιμών και counter<3 (οχι όφελος)
         %βάλε στον πίνακα 1 ή 2 φορές την τιμή
        elseif image(i,j) ~= image(i,j-1) && counter < 3

                for k = 1 : counter
                     encoded = [encoded,image(i,j-1)];
                end
                counter=1;
         %αν τελείωσε η επαναλάψη των ίδιων τιμών και counter>3 (όφελος)
         %βάλε στον πίνακα το πλήθος, την τιμή και το σύμβολο -1 για
         %διαχωριστικό
        elseif image(i,j) ~= image(i,j-1) && counter >= 3
                encoded = [encoded,counter];
                encoded = [encoded,image(i,j-1)];
                encoded = [encoded,-1];
                counter=1;
        end
        %υλοποίηση αλγορίθμου για τα τελευταία κελιά κάθε σειράς
        if j == columns && counter == 1
             encoded = [encoded,image(i,j)];
        elseif j == columns && counter == 2
             encoded = [encoded,image(i,j)];
             encoded = [encoded,image(i,j)];
        elseif j == columns && counter > 2
             encoded = [encoded,counter];
             encoded = [encoded,image(i,j-1)];
             encoded = [encoded,-1];
        end
    end
    counter=1;
end
%υπολογισμός bytes για λόγο συμπίεσης
[~,columns] = size(encoded);
b2 = columns;
l = b1/b2;
end
```

Εικόνα 1.1 : Αλγόριθμος RLE

```
[rows,columns] = size(image);
%μετρητης για πληθος επαναλήψεων, >3 για να υπαρχει οφελος κωδικοποίησης
counter=1;
%πίνακας κωδικοποίησης
encoded = [];
%αρχικό πλήθος bytes
bl=rows*columns;
```

Εικόνα 1.1 : Αρχικοποιήσεις συνάρτησης

1.1 Σχόλια για λόγους συμπίεσης

Οι λόγοι συμπίεσης ααναφέρονται στο κέρδος που έχουμε με την υλοποίηση της κωδικοποίησης RLE. Στην περίπτωση μας έχουμε 3 λόγους με μεγάλες διαφορές μεταξύ τους,όπως άλλωστε είναι αναμενόμενο αφού οι εικόνες μεταξύ τους διαφέρουν.

l1	16.6667
l2	4.8985
l3	1.0020

Εικόνα 1.1.1 : Λόγοι συμπίεσης

Όσο περισσότερες αλλαγές εμφανίζονται μέσα σε μία εικόνα τόσο μικρότερος είναι ο λόγος συμπίεσης αφού δεν θα έχουμε επαναλήψεις τιμών. Γιαυτό ο l1 που αναφέρεται στην εικόνα με το σκάκι βλέπουμε πόσο μεγάλο λόγω έχει, αφού στην ουσία πρόκειται για μία εικόνα με πάρα πολλές επαναλήψεις. Αντίθετο παράδειγμα είναι η εικόνα της Lenna όπου παρουσίαζει συνεχείς διαφορές και βλέπουμε πόσο μικρό όφελος έχουμε από τον συγκεκριμένο αλγόριθμο.

2 Run Length Decoder

Αφού υλοποιήσαμε μία συνάρτηση κωδικοποίησης έπρεπε να υλοποιήσουμε και αντίστοιχη για την αποκωδικοποίηση της.

```
while 1
    %έλεγχος για εύρεση συμβόλου -1
    if encoded(i+2)== -1
        %βάλε για όσο πλήθος βρήκες στον encoded ,τώσα κελιά
        for l = k:k+encoded(i) - 1
            decoded(j,l) = encoded(i+1);
        end
        %έλεγχος για αλλαγή γραμμής και παραμετροποίηση indexes
        if l==dimen
            k=1;
            j=j+1;
        else
            k=k+encoded(i);
        end
        i=i+3;
       %αν δεν βρήκες -1 στο +2 κελί
    else
        %έλεγξε αν χρειάζεται να αλλάξεις γραμμή
        %και παραμετροποίηση εκ νέου τους indexes
        if k==dimen
             decoded(j,k) = encoded(i);
            k=1;
            j=j+1;

            i=i+1;
        else
            %αλλιώς απλά πρόσθεσε το στοιχείο
            %και προχώρα τους indexes
            decoded(j,k) = encoded(i);
             i=i+1;
             k=k + 1;
        end

    end
    %έλεγχος για το αν έφτασε ο encoded στο τέλος
    if i + 2 == columns
        break;
    end

end
```

Εικόνα 2.1 : Βασικό κομμάτι αλγορίθμου αποκωδικοποίησης

```
%αρχικοποίηση πίνακα αποκωδικοποίησης, indexes
decoded = zeros(dimen,dimen);
[~,columns] = size(encoded);
i=1;
j=1;
k=1;
```

Εικόνα 2.2 : Αρχικοποιήσεις πινάκων & μεταβλητών

```
%υλοποίηση αλγορίθμου για τα τελευταία 3 κελιά
if encoded(i+2) == -1
    for l = k : k + encoded(i) - 1
        decoded(j,l) = encoded(i+1);
    end
else
    decoded(j,k) = encoded(i);
    k = k + 1;
    i = i + 1;
    decoded(j,k) = encoded(i);
    k = k + 1;
    i = i + 1;
    decoded(j,k) = encoded(i);
end
```

Εικόνα 2.3 : Τέλος αλγορίθμου

Σαυτή την συνάρτηση πέρνουμε ως παραμέτρους την διάσταση μιας εικόνας και την κωδικοποίηση της και επιστρέφουμε την αρχική εικόνα. Τρέχουμε μία επανάληψη μέχρι τα 3 τελευταία κελιά του πίνακα που περιέχει την κωδικοποίηση. Μέσα της ελέγχουμε το 2ο επόμενο κελί είναι -1 ώστε να περάσει επαναληπτικά τις τιμές ή αν εμφανίζονται 1 ή 2 φορές κάποιες τιμές. Όποτε πρέπει γίνονται έλεγχοι για αλλαγή γραμμής και παραμετροποίησης των indexes. Τέλος τρέχουμε τον αλγόριθμο για τα τελευταία κελιά.

3 Κβαντισμός

Ως παραμέτρους δίνουμε την εικόνα καθώς και των πλήθος των bits/pixel που θέλουμε να κβαντίσουμε. Υπολογίζουμε βάση αυτών σε πόσα τμήματα πρέπει να χωρίσουμε το συνολικό εύρος . Έτσι τρέχουμε επαναληπτικά για όλες τις τιμές της εικόνας και υπολογίζουμε σε ποιό τμήμα ανήκει κάνοντας μία διαίρεση και μία στρογγυλοποίηση. Τέλος η συνάρτηση μας επιστρέφει την κβαντισμένη εικόνα.

```
function [out_image] = quantizer(image,n)

  z=power(2,n); % # τμημάτων των συνιστωσών
  k = 256;      %συνιστώσες
  euros = k/z;  %εύρος

 [rows,columns] = size(image);
  out_image = zeros(rows,columns,'uint8');

    for i = 1 : rows
        for j = 1 : columns

            x = image(i,j)/euros;
            out_image(i,j) = euros * round(x);
            %στρογγυλοποίηση στον πρώτο πάνω ακέραιο
        end
    end


end
```

Εικόνα 3.1 : Σ υνάρτηση κβαντισμού grayscale εικόνων

3.1 Κβάντιση & RLE

Στο ερώτημα αυτό χρησιμοποιούμε την εικόνα της Lenna όπως μας ζητήθηκε για κάνουμε κβάντιση για διάφορες τιμές bits/pixel και να απεικονίσουμε αυτές τις 8 εικόνες σε ένα plot. Επιστρέφουμε τα frames όπου αναφέρονται στις 8 κβαντισμένες εικόνες για να χρησιμοποιηθουν μετέπειτα στην movie.

```
function [frames] =  quantizer2(image)

    %κβάντιση εικόνας για διαφορετικές τιμές bits/pixel
    out1 = quantizer(image,1);
    out2 = quantizer(image,2);
    out3 = quantizer(image,3);
    out4 = quantizer(image,4);
    out5 = quantizer(image,5);
    out6 = quantizer(image,6);
    out7 = quantizer(image,7);
    out8 = quantizer(image,8);
```

Εικόνα 3.1.1 : Χρήση συνάρτησης 3ου ερωτήματος

Κρατάμε σε 8 μεταβλητές τις εικόνες μας.

```
%Συγχώνευση κβαντισμένων εικόνων σε 1 plot
subplot(2,4,1); imshow(out1); subplot(2,4,2); imshow(out2);...
subplot(2,4,3); imshow(out3); subplot(2,4,4); imshow(out4);...
subplot(2,4,5); imshow(out5); subplot(2,4,6); imshow(out6);...
subplot(2,4,7); imshow(out7); subplot(2,4,8); imshow(out8);
```

Εικόνα 3.1.2 : Κάνουμε ένα κοινό plot και τις 8

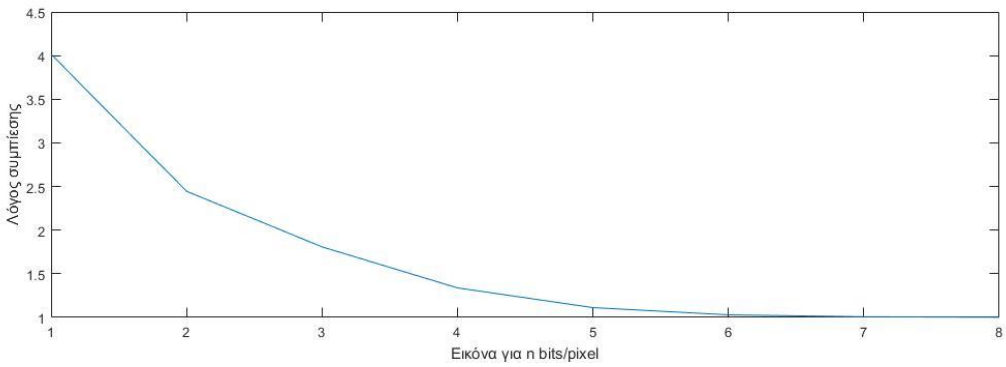
Εικόνα 3.1.3 : Ίδια εικόνα για 8 διαφορετικές τιμές bits/pixel

Η αριστερή επάνω εικόνα ξεκινάει με 1 bits/pixel και η δεξιά κάτω καταλήγει σε 8.

```
%RLE για όλες τις κβαντισμένες εικόνες
[~,l1] = run_length_encoder(out1);
[~,l2] = run_length_encoder(out2);
[~,l3] = run_length_encoder(out3);
[~,l4] = run_length_encoder(out4);
[~,l5] = run_length_encoder(out5);
[~,l6] = run_length_encoder(out6);
[~,l7] = run_length_encoder(out7);
[~,l8] = run_length_encoder(out8);

%λόγοι συμπίεσης και figure αυτών
l = [l1,l2,l3,l4,l5,l6,l7,l8];
figure; plot(l);
```

Εικόνα 3.1.4 : RLE για τις 8 κβαντισμένες εικόνες & figure με λόγους συμπίεσης



Εικπόνα 3.1.5 : Πτώση λόγου συμπίεσης για περισσότερες χρωματικές αποχρώσεις

Όπως είναι αναμμενόμενο οι λόγοι συμπίεσης μειώνονται όσο αυξάνονται τα bits/pixel αφού μειώνονται οι επαναλήψεις τιμών της εικόνας.

4 Αρχείο “workspace.mat”

Το workspace που παραθέτουμε περιέχει συνολικά 10 στοιχεία. Για την ακρίβεια έχουμε

- Encoded_chess: κωδικοποίηση εικόνας σκακιού
- Encoded_geo: κωδικοποίηση γεωμετρικής εικόνας
- Encoded_Lenna: κωδικοποίηση εικόνας Lenna
- Decoded_chess: αποκωδικοποιημένη εικόνα σκακιού
- Decoded_geo: αποκωδικοποιημένη γεωμετρική εικόνα
- Decoded_Lenna: αποκωδικοποιημένη εικονα Lenna
- L1: λόγος συμπίεσης εικόνας σκακιού
- L2: λόγος συμπίεσης γεωμτρικής εικόνας
- L3: λόγος συμπίεσης εικόνας Lenna