Μυλωνάκης Αλέξανδρος
ΑΜ:3045
Εργασία 1η Πολυμέσα

Λογισμικό: GNU Octave

Μέρος 1ο

Αρχικά φορτώνω την εικόνα με τη εντολή που μου δίνεται και δημιουργώ 5 λίστες για την αποθήκευση δεδομένων ώστε να τα προβάλω μετά(1 για όλες τις εικόνες, 1 για τους τίτλους, 1 για τις τιμές της μέγιστης αναλογίας σήματος προς θόρυβο (PSNR), 1 για τις τιμές του συνόλου των συντελεστών που μηδενίστηκαν και 1 για τις τιμές των κατωφλίων.

Στην συνέχεια μέσα σε μια επαναληπτική διαδικασία για να διατρέξω των πίνακα thresholds, ο οποίος περιέχει τις τιμές κατωφλίων, υπολογίζω την αναδημιουργημένη εικόνα και τους συντελεστές που μηδενίστηκαν χρησιμοποιώντας συναρτήσεις από την κλάση reconstructImage που δημιούργησα. Τέλος για την εκτύπωση των αποτελεσμάτων και την προβολή των τελικών εικόνων δημιούργησα μια κλάση displayResults.

```
f = imread('C:\Users\Alekos\Desktop\Πολυμεσα\erg1\cameraman.tif');
images ={f};
titles={'Original Image'};
psnr_values={Inf};
zeroedCoefficientsValues = {0};



thresholds = [5, 10, 15];

for i = 1:length(thresholds)
  threshold = thresholds(i);
  recon = reconstructImage(f);
  [reconstructedImage, zeroedCoefficients] = recon.reconstructByThreshold(threshold);
  images{end+1} = reconstructedImage;
  titles{end+1} = sprintf('Altered Image %d', i);
  zeroedCoefficientsValues{end+1} = zeroedCoefficients;
  psnrCalculator = findPeakSignalToNoiseRatio(f, reconstructedImage);
  psnrValue = psnrCalculator.calculatePSNR();
  psnr_values(end+1) = psnrValue;
end

result = displayResults(images,titles,psnr_values,'zeroed Coefficients');
result.displayPart1(zeroedCoefficientsValues);
```

```
classdef reconstructImage
  properties
    image;
  end
  methods
    function obj = reconstructImage(image)
      obj.image=image;
    end
```

Η κλάση reconstructImage παίρνει ως όρισμα μια εικόνα και μέσω συναρτήσεων ανακατασκευάζει μια εικόνα χρησιμοποιώντας τιμές κατωφλίου και τις συναρτήσεις dct2 και idct2 μηδενίζω τις τιμές που είναι κάτω από το κατώφλι υπολογίζω των αριθμό των συντελεστών που μηδενίστηκαν και ανακατασκευάζω την εικόνα.

```
    function [reconstructedImage,zeroedCoefficients] = reconstructByThreshold(obj,threshold)
      F_dct = dct2(obj.image);
      zeroedCoefficients =0;
      for u = 1:size(F_dct, 1)
        for v = 1:size(F_dct, 2)
          if abs(F_dct(u,v))< threshold
            F_dct(u,v)=0;
            zeroedCoefficients+=1;
          end
        end
      end
      reconstructedImage=idct2(F_dct);
      reconstructedImage = obj.makeImageUInt8(reconstructedImage);
    end
```

```
function reconstructedImage = makeImageUInt8(obj,reconstructedImage
  for u = 1:size(reconstructedImage, 1)
    for v = 1:size(reconstructedImage, 2)
      if reconstructedImage(u,v) <0
        reconstructedImage(u,v)=0;
      elseif reconstructedImage(u,v)>255
        reconstructedImage(u,v)=255;
      endif
    end
  end
  reconstructedImage = uint8(reconstructedImage);
end
```

Με την συνάρτηση makeImageUInt8 ελέγχω τις τιμές μεγαλύτερες από 255 και μικρότερές από 0 και μετατρέπω την εικόνα σε uint8.

```
classdef displayResults
  properties
    images;
    titles;
    psnr_values;
    displayValuesId;
  end
  methods
    function obj = displayResults(images,titles,psnr_values,displayValuesId)
      obj.images = images;
      obj.titles = titles;
      obj.psnr_values = psnr_values;
      obj.displayValuesId = displayValuesId;
    end
```

Η κλάση displayResult παίρνει ως ορίσματα τις 3 λίστες που δημιουργήσαμε με τα αποτελέσματα από την επανάληψη με τις διαφορετικές τιμές κατωφλίων και τα τυπώνει σε ένα

```
  function displayPart1(obj,displayValues)
    num_images = length(obj.images);
    num_cols = ceil(sqrt(num_images));
    num_rows = ceil(num_images / num_cols);

    figure;
    colormap(gray);
    for i = 1:num_images
      ax = subplot(num_rows, num_cols, i);
      subplot(num_rows, num_cols, i);
      imagesc(obj.images{i});
      axis image;
      title(obj.titles{i});
      annotationText = sprintf('%s: %0.2f, PSNR: %0.2f', obj.displayValuesId, displayValues{i}, obj.psnr_values{i});
      if num_rows > 1
         textYPosition = -0.1 - (0.1 * (num_rows - 1));  % Adjust Y position based on rows
      else
        textYPosition = -0.25;  % Default position for single row
      end
      text(0.5, textYPosition, annotationText, 'Units', 'normalized', ...
       'HorizontalAlignment', 'center', 'FontSize', 10, 'Color', 'black');;
    end
  end
end
```

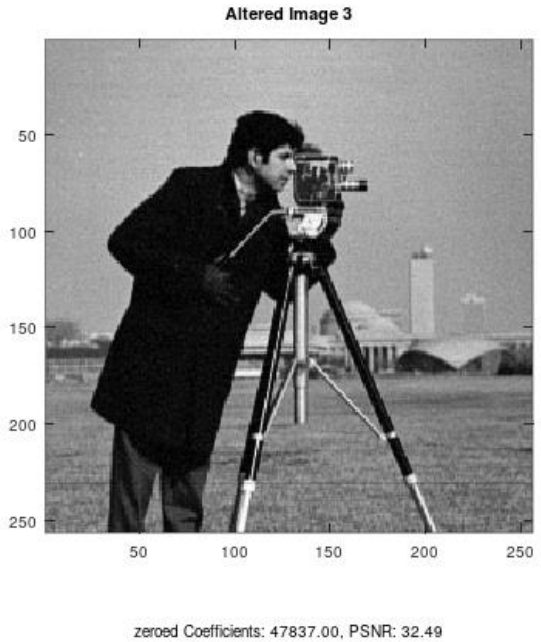
Τα αποτελέσματα:

Μηδενισμένοι συντελεστές: 0.00
PSNR: inf
(αρχική εικόνα)

Μηδενισμένοι συντελεστές: 23240
PSNR: 43.59

Μηδενισμένοι συντελεστές: 38919
PSNR: 36.09

Μηδενισμένοι συντελεστές: 47837
PSNR: 32.49



Μέρος 2ο

Αρχικά φορτώνω την εικόνα με τη εντολή που μου δίνεται και δημιουργώ τον πίνακα Q και 5 λίστες για την αποθήκευση δεδομένων ώστε να τα προβάλω μετά(1 για όλες τις εικόνες, 1 για τους τίτλους, 1 για τις τιμές της μέγιστης αναλογίας σήματος προς θόρυβο (PSNR), 1 για τις τιμές της εντροπίας που υπολογίζω και 1 για τις τιμές των πολλαπλάσιων του Q. Στην συνέχεια μέσα σε μια επαναληπτική διαδικασία για να διατρέξω των πίνακα qValues, ο οποίος περιέχει τις τιμές με τις οποίες πολλαπλασιάζεται το Q , υπολογίζω την αναδημιουργημένη εικόνα και τις τιμές της εντροπίας με την βοήθεια των κλάσεων reconstructImage (του πρώτου ερωτήματος) και quantization που δημιούργησα. Τέλος για την εκτύπωση των αποτελεσμάτων και την προβολή των τελικών εικόνων χρησιμοποιώ την ιδιά κλάση displayResults που είχα και στο 1ο μέρος .

```
function [reconstructedImage,myEntropy] =reconstructByQuantization(obj,Q)
  myQuantization = quantization(obj.image,Q);
  [reconstructedImage,myEntropy] = myQuantization.computeBlocks();
  reconstructedImage = obj.makeImageUInt8(reconstructedImage);
end
```

Πρόσθεσα αυτή τη συνάρτηση στην κλάση reconstructImage για να δημιουργήσω αντικείμενα τύπου quantization,

```
f = imread('C:\Users\Alekos\Desktop\Πολυμεσα\ergl\cameraman.tif');
image_entropy = entropy(f);
Q = [
    16 11 10 16 24 40 51 61;
    12 12 14 19 26 58 60 55;
    14 13 16 24 40 57 69 56;
    14 17 22 29 51 87 80 62;
    18 22 37 56 68 109 103 77;
    24 35 55 64 81 104 113 92;
    49 64 78 87 103 121 120 101;
    72 92 95 98 112 100 103 99
];

images ={f};
titles={'Original Image'};
psnr_values={Inf};
entropyValues = {image_entropy};

qValues = [1, 2, 4];

for i=1:length(qValues)
  Q=Q*qValues(i);
  recon = reconstructImage(f);
  [reconstructedImage, entropyValue] = recon.reconstructByQuantization(Q);
  images{end+1} = reconstructedImage;
  entropyValues{end+1} = entropyValue;
  titles{end+1} = sprintf('Altered Image %d', i);
  psnrCalculator = findPeakSignalToNoiseRatio(f, reconstructedImage);
  psnrValue = psnrCalculator.calculatePSNR();
  psnr_values(end+1) = psnrValue;
end

result = displayResults(images,titles,psnr_values,'entropy');
result.displayPart1(entropyValues);
```

```
function quantizedBlock= quantize(obj,block)
  block = double(block);
  dctBlock = dct2(block);
  quantizedBlock = round(dctBlock ./ obj.Q);
end

function dequantizedBlock = dequantize(obj,block)
  block = double(block);
  dequantizedBlock = block .* obj.Q;
  dequantizedBlock =idct2(dequantizedBlock);
end
```

Η κλάση quantization περιέχει 3 συναρτήσεις μια computeBlocks για την δημιουργία και διαχείριση του κάθε block τις εικόνας και τον υπολογισμό της εντροπίας. Οι άλλες 2 συναρτήσεις είναι η quantize & dequantize για τον κβαντισμό και την αποκβάντωση της εικόνας.

```
function [dequantizedBlocks,myEntropy] = computeBlocks(obj)
  blockSize = size(obj.Q);
  [rows,cols] = size(obj.image);
  quantizedBlocks = [];
  dequantizedBlocks = [];
  for u = 1:blockSize(1):rows
    for v = 1:blockSize(2):cols
      rowEnd = u + blockSize(1) - 1;
      colEnd = v + blockSize(2) - 1;
      block = obj.image(u:rowEnd, v:colEnd);
      quantizedBlock = obj.quantize(block);
      quantizedBlocks(u:rowEnd , v:colEnd) = quantizedBlock;
      dequantizedBlock = obj.dequantize(quantizedBlock);
      dequantizedBlocks(u:rowEnd , v:colEnd) = dequantizedBlock;
    end
  end
  quantizedBlocks = abs(quantizedBlocks);
  myEntropy= entropy(quantizedBlocks);
end
```

Μέρος 3ο

Για το 3ο μέρος δημιούργησα μια συνάρτηση που υπολογίζει τα μηδενικά πριν από κάθε μη μηδενικό συντελεστή και τα αποθηκεύει σε ζευγάρια (Level- Run). Παίρνω ως είσοδο την κβαντισμένη λίστα που μου δίνεται στην εκφώνηση

```
]function level_run_pairs = level_run_toEncrypt(ac_Coefficients)
    level_run_pairs = [];
    run = 0;
]   for i = 1:length(ac_Coefficients)
      currentCoefficient = ac_Coefficients(i);
]     if currentCoefficient==0
        run =run +1;
      else
        level_run_pairs = [level_run_pairs; currentCoefficient , run];
        run = 0;
      end
    end
end
```

```
AC_coefficients = [7, 0, 4, 0, 2, 3, 0, 0, 2, 4, 0, 1, 0, 0, 0, 1, 0, 0, -1, -1, 0, 1, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
level_run_pairs=level_run_toEncrypt(AC_coefficients);
disp(level_run_pairs);
```

τα αποτελέσματα του 3ου μέρους είναι:

```
    7    0
    4    1
    2    1
    3    0
    2    2
    4    0
    1    1
    1    3
   -1    2
   -1    0
    1    1
    1    3
    2    2
    1    0
```

Τα αποτελέσματα του 2ου μέρους:

Entropy : 7.01
PSNR: inf
(Είναι η ίδια εικόνα)

Entropy :0 .60
PSNR: 31.74
Q = 1*Q

Entropy : 0.45
PSNR: 29.31
Q = 2Q1

Entropy : 0.23
PSNR: 14.98
Q = 4*Q