“Ανακατασκευή Εικόνων με χρήση Τεχνικών Συμπιεσμένης Καταγραφής”

Ορέστης Νικόλας, ΑΜ: 228438, 5ο έτος, ΗΜΤΥ

```
%% Task1
x = load('Fruits.mat');
x = struct2cell(x);
x = cell2mat(x);
X = FDCT2(x);

%small elements replaced by zeroes
X(abs(X) < 0.0001) = 0;

%Nonzero Percentage
Nonzeros = nnz(X);
Nzpercent = Nonzeros/(128*128);
imshow(x);
```

Ερώτημα 1ο :

- Υπολογισμός DCT της δοσμένης εικόνας
- Απόδειξη αραιού σήματος
- Πλήθος και ποσοστό non-zero στοιχείων

Στο παραπάνω code block φαίνεται τι έκανα για να υπολογίσω τα ζητούμενα. Παρακάτω φαίνονται τα αποτελέσματα:

Nonzeros	1558
Nzpercent	0.0951
x	*128x128 double*
X	*128x128 double*

Βλέπουμε ότι από τα 128x128 = 16384 στοιχεία του DCT της εικόνας μόνο τα 1558 είναι μη μηδενικά, άρα για να ανακτήσουμε την εικόνα αρκεί να κρατήσουμε το 9.51% των συντελεστών και προφανώς το σήμα μας είναι αραιό.

Fruits Original Picture

DCT2 of the Original

Ερώτημα 2^{ο} :

- Χρήση OMP
- Γνωρίζω αρχικό dictionary F
- Γνωρίζω αρχική μέτρηση y

```
%% Task2
N = 128;
K = N^2;
k = 6000;
F = normrnd(0,1,[k K]);      %F ~ N(0,1) -> k x N^2
i = reshape(X,K,1);          %i -> N^2 x 1
y = F*i;                     %y -> k x N^2 x N^2 x 1 = k x 1

[i_,residueErr] = OMP(y,F,400,0.05);
Z = reshape(i_,N,N);
z = idct2(Z);
imshow(z);
%[q=400,k=4000/5000/6000]
function [ i_, residueErr] = OMP(y,F,q,err)
    [k,K] = size(F);
    i_= zeros(K,1);
    r=y;
    T=[];
    cntr = 1;
    residueErr = 1;     %for 100%
    while cntr < q && residueErr > err
        [~,jj] = max(abs(transpose(F)* r));
        T = union(T,jj);
        i_(T)=pinv(F(:,T)) * y;  %%coeffecient update
        r=y-F*i_;                %%residu update
        cntr = cntr + 1;
        residueErr = norm(r)/norm(y);
    end
end
```

Στον κώδικα αρχικοποιώ μερικές μεταβλητές όπως k, K, N. Από θεωρία έχουμε y = F*i και εφόσον ξέρω ότι F ~ N(0,1): k x N^2 και i:N^2 x 1 (το reshaped X), υπολογίζω το αρχικό y για χάρη του παραδείγματος. Το y θα έχει διαστάσεις k x N^2 x N^2 x 1 ή αλλιώς y: k x 1 άρα προς στιγμής έχω το διάνυσμα της αρχικής μου μέτρησης.

Τώρα για τη συγκεκριμένη άσκηση υποθέτω ότι δεν ξέρω το i και με τον OMP θα πρέπει να υπολογίσω το i από τη σχέση y = F * i.

Η υλοποίηση του OMP:

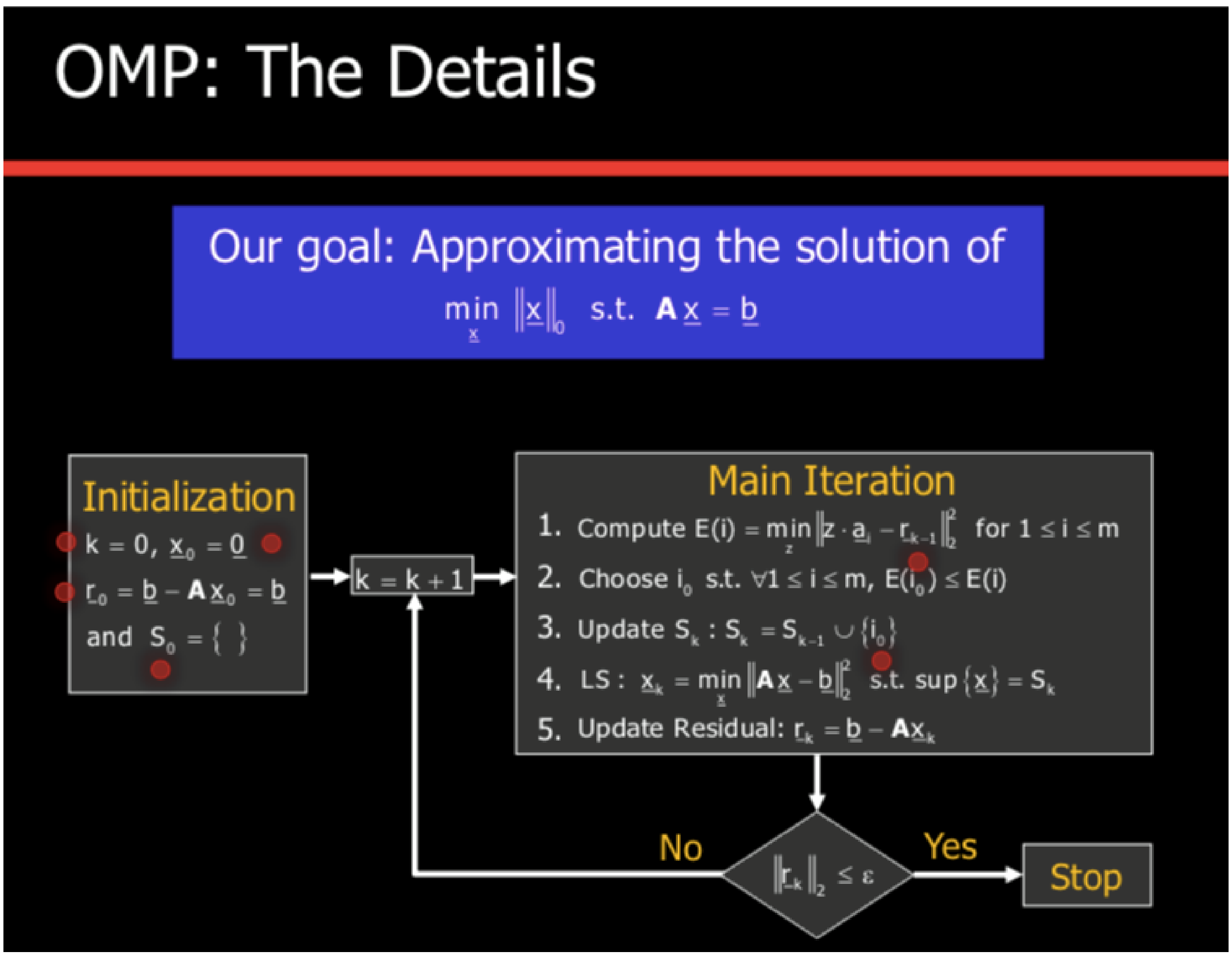
Inputs:

1. y = initial measurement
2. F = dictionary
3. q = #iterations
4. err = preferable residual error

Outputs:

1. i_ = προσέγγιση του i, δηλαδή του reshaped X
2. residueErr = residual error

Το μοντέλο που βασίστηκα για την υλοποίηση του OMP φαίνεται παρακάτω αλλά έκανα μερικές τροποποιήσεις όσον αφορά το exit condition. Η τροποποίηση που έκανα είναι ότι για το exit condition αρκεί είτε να φτάσει το Preferred Residual Error είτε να ξεπεράσει ένα προκαθορισμένο αριθμό iterations γιατί γενικότερα οι υπολογισμοί είναι αρκετά βαριοί.



Εξήγηση αλγορίθμου OMP:

- Αρχικοποιώ τα k, K, i_, r, T, cntr, residueErr όπως φαίνεται στον κώδικα
- Ξεκινάει το loop όπου βγαίνει μόνο εάν περάσει ο προκαθορισμένος αριθμός iterations ή εάν επιτύχω το residual error που θέλω

Σε κάθε iteration κάνω τα εξής:

- Υπολογίζω το jj που είναι πρακτικά τα indices του max(abs(transpose(F) * r))
- Κάνω union κάθε φορά στο T τα indices
- Υπολογίζω i_(T), δηλαδή μερικούς από τους συντελεστές. Εδώ χρησιμοποιώ τη συνάρτηση pinv της matlab όπου υπολογίζει Moore-Penrose pseudoinverse και είναι υπολογιστικά πολύ βαριά πράξη.
- r = y – F*i το residue
- residueErr = norm(r)/norm(y) = μετρική για το πόσο κοντά είμαι στο πραγματικό y

Μετά το τέλος του OMP:

- Έχω την εκτίμηση i_ του i
- Κάνω reshape το i_ σε N x N (όπου πρακτικά είναι το X)
- Κάνω idct2(i_)
- Έχω την εκτίμηση z της εικόνας x

Αποτελέσματα και σχόλια:

Γενικότερα τα max iterations που κάνω είναι 400 γιατί για περισσότερα κάνει πολύ ώρα να τρέξει. Το preferred residual error το έχω στο 5% αν και με τα 400 iterations που κάνω ποτέ δε φτάνει αυτή τη τιμή (βγαίνει από το loop του OMP πάντα επειδή έχει ξεπεράσει τα 400 iteration), συνήθως είναι κάτι ανάμεσα στο 10-20%

Οι δοκιμές που έχω κάνει είναι για err = 0.05 και διάφορους συνδυασμούς q = 50|200|400 και k = 1000|4000|5000|6000, όπου πρακτικά το k είναι πόσες αρχικές μετρήσεις έχουμε και είναι ένα μέτρο συμπίεσης γιατί εν τέλει από k μετρήσεις παράγω ολόκληρη την εικόνα που είναι 128 x 128 = 16384 στοιχεία. Μετρικές που χρησιμοποιώ για να δω πόσο κοντά είναι η εκτίμηση μου είναι το sqrt(immse(x,z)) που είναι η ρίζα του mean-squared error μεταξύ της πραγματικής εικόνας και της εκτίμησης, το πώς φαίνεται η ίδια η εκτίμηση της εικόνας και το residual error.

q=400, k=4000

Fruits Original Picture

My estimation

Χαρακτηριστικά:

- Residual error = 10.53%
- Sqrt(immse(x,z)) = 7.18% (Δεν αποτελεί ποσοστό αυτή η τιμή αλλά τη γράφω έτσι για να φαίνεται καλύτερη)
- Compression Ratio = 1 - 4000/16384 = 75.59%
- Non-zero percentage on i_ = nnz(i_)/(128*128) = 2.44%

q=400, k=5000

Fruits Original Picture

My estimation

Χαρακτηριστικά:

- Residual error = 11.18%
- Sqrt(immse(x,z)) = 6.45%
- Compression Ratio = 1 - 5000/16384 = 69.48%
- Non-zero percentage on i_ = nnz(i_)/(128*128) = 2.44%

q=400, k=6000

Fruits Original Picture

My estimation

Χαρακτηριστικά:

- Residual error = 11.33%
- Sqrt(immse(x,z)) = 6.18%
- Compression Ratio = 1 - 6000/16384 = 63.38%
- Non-zero percentage on i_ = nnz(i_)/(128*128) = 2.44%

q=200, k=6000

Fruits Original Picture

My estimation

Χαρακτηριστικά:

- Residual error = 15.64%
- Sqrt(immse(x,z)) = 8.01%
- Compression Ratio = 1 - 6000/16384 = 63.38%
- Non-zero percentage on i_ = nnz(i_)/(128*128) = 1.21%

q=50, k=6000

Fruits Original Picture

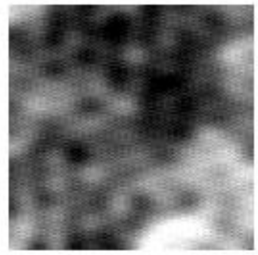
My estimation

Χαρακτηριστικά:

- Residual error = 26.13%
- Sqrt(immse(x,z)) = 12.40%
- Compression Ratio = 1 - 6000/16384 = 63.38%
- Non-zero percentage on i_ = nnz(i_)/(128*128) = 0.3%

q=50, k=1000

Fruits Original Picture



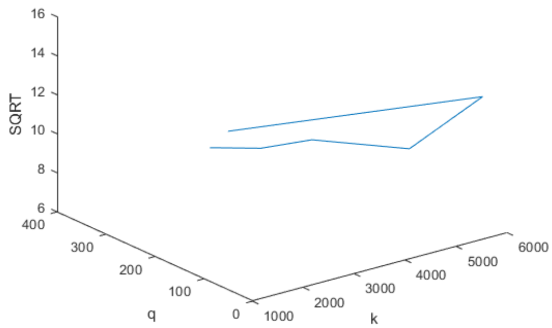
My estimation

Χαρακτηριστικά:

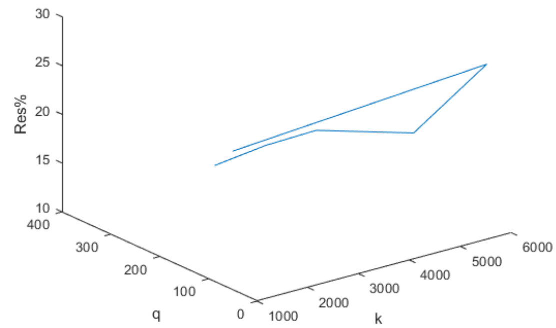
- Residual error = 24.24%
- Sqrt(immse(x,z)) = 14.14%
- Compression Ratio = 1-1000/16384 = 93.90%
- Non-zero percentage on i_ = nnz(i_)/(128*128) = 0.3%

Τελικά έχω:

- Q =   [50   50   200   400   400   400];
- K =   [1000   6000 6000 6000 5000 4000];
- SQRT = [14.14 12.40 8.01 6.18 6.45 7.18];
- Res = [24.24 26.13 15.64 11.33 11.18 10.53];



sqrt(immse(x,z))



Residual Error

Παρατηρούμε ότι τα καλύτερα αποτελέσματα και στο μάτι αλλά και στις μετρικές τα έχουμε όταν ανεβάζουμε ταυτόχρονα και το k αλλά και το q. Με τον OMP αλγόριθμο βλέπουμε ότι μπορούμε να κρατήσουμε 6000 αντί για 128 x 128 = 16384 τιμές (compression ratio 63.38%) και η εικόνα που ανακτούμε είναι αρκετά ικανοποιητική. Είναι προφανές ότι υπάρχει ένα tradeoff μεταξύ της υπολογιστικής ισχύς που διαθέτουμε(μέσω του q) και της ποιότητας όσο και ανάμεσα στη συμπίεση(μέσω του k) που θέλουμε να πετύχουμε και της ποιότητας.

Τέλος, υπάρχουν τροποποιημένοι OMP αλγόριθμοι οι οποίοι είναι πιο γρήγοροι όπως ο OMP-CHOLESKY και ο BATCH-OMP που έχουν κάποια “trick” για να αποφύγουν τον υπολογισμό pseudo-inverse αλλά για τον κλασικό greedy OMP έχουμε αυτά τα θέματα performance.

Ευχαριστώ!