Image Segmentation

ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ

Αντώνιος Σκούρτης (9142) | askourtis@ece.auth.gr | 29/05/2021

Περιεχόμενα

Εικόνες ως affinity γράφοι

Μία εικόνα $I_{M\times N\times 3}$ μπορεί να αναπαρασταθεί ως affinity πίνακας *(γράφος)* $W_{MN\times MN}$ ως εξής:

$$W_{i,j} = e^{-D_{i,j}} = e^{-\|I_i - I_j\|}, \text{ όπου } I_i = I_{i \bmod M, \frac{i}{M}}$$

Είναι φανερό πώς ο πίνακας W είναι τετραγωνικός και συμμετρικός, επομένως οι ιδιοτιμές του είναι θα πάντα πραγματικές.

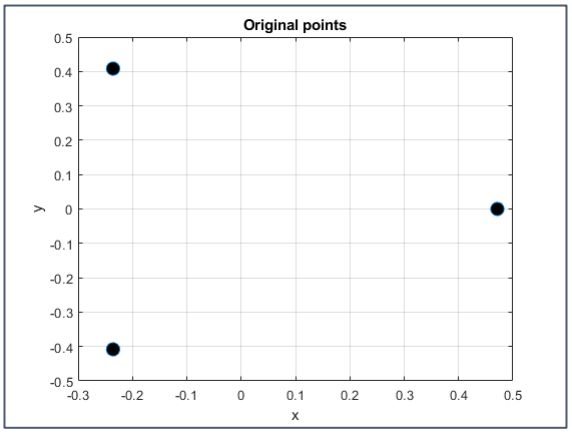
Η μετατροπή της εικόνας σε affinity γράφο είναι χρήσιμη, καθώς επιτρέπει αλγορίθμους όπως ο *kMeans* να ομαδοποιήσουν τα διάφορα pixels της αρχικής εικόνας.

Spectral clustering

Αφού μετατραπεί η εικόνα στον ανάλογο affinity γράφο εκτελείται η διαδικασία του *spectral clustering*, η οποία ομαδοποιεί τα όμοια pixels βάσει μίας παραλλαγής του αλγορίθμου *kMeans*.

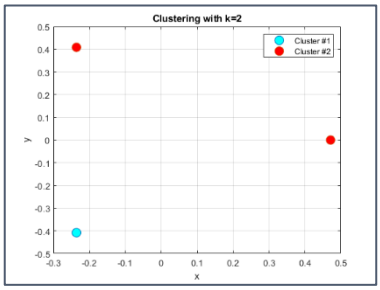
DEMO 1

Στο παρόν demo ο affinity γράφος παράγεται από δώδεκα συνολικά σημεία. Τα σημεία αυτά είναι χωρισμένα σε τρείς περιοχές. Δεδομένων των παραπάνω η πιο λογική ομαδοποίηση είναι η δημιουργία τριών διαφορετικών ομάδων για κάθε περιοχή σημείων.

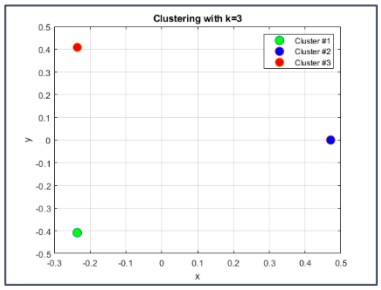


Demo1 - 1: Τα πιθανά αρχικά σημεία του affinity γράφου

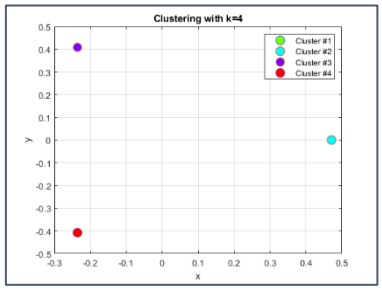
Ο αλγόριθμος επιτρέπει την ομαδοποίηση σε αυθαίρετο αριθμό ομάδων, προφανώς λιγότερων ομάδων από αριθμό σημείων. Επομένως εκτελέστηκαν τρία διαφορετικά πειράματα ομαδοποιώντας τα σημεία σε δύο, τρείς και τέσσερεις ομάδες αναλόγως.



Demo1 - 2: Ομαδοποίηση με 2 ομάδες



Demo1 - 3: Ομαδοποίηση με 3 ομάδες



Demo1 - 4: Ομαδοποίηση με 4 ομάδες

Είναι φανερό πως η ομαδοποίηση με δύο ομάδες έχει ομαδοποιήσει δύο διαφορετικές περιοχές στην ίδια ομάδα, το οποίο είναι υποβέλτιστη περίπτωση. Επιπλέον είναι τυχαίο ποια από τις δύο ομάδες θα έχει την περίσσια περιοχή, καθώς ο *kMeans* αρχικοποιεί τα κέντρα του τυχαία.

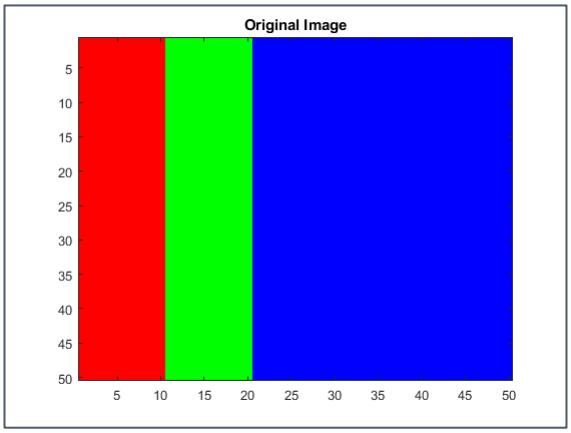
Η περίπτωση της ομαδοποίησης με 3 ομάδες είναι βέλτιστη, καθώς κάθε περιοχή έχει την δική της ομάδα και κάθε ομάδα είναι καλώς χωρισμένη από τους γείτονές της.

Τέλος η ομαδοποίηση με 4 ομάδες είναι εξίσου υποβέλτιστη περίπτωση καθώς σε μία περιοχή βρίσκονται σημεία που ανήκουν σε παραπάνω από μία ομάδες. Ο παραπάνω ισχυρισμός δεν μπορεί να επιβεβαιωθεί με το δοθέν διάγραμμα καθώς τα σημεία της διαφορετικής ομάδας επικαλύπτονται από τα σημεία της αρχικής ομάδας *(ή το ανάποδο)*.

DEMO 2

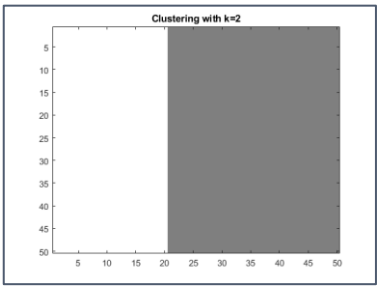
Παράδειγμα 1ο

Στην περίπτωση αυτή αναλύονται δύο εικόνες. Οι εικόνες αυτές αρχικά μετατρέπονται στους ανάλογους affinity γράφους τους και στην συνέχεια εκτελείται ο αλγόριθμος ομαδοποίησης *Specular Clustering*. Ως αποτέλεσμα παράγεται μία νέα εικόνα, όπου περιοχές παρόμοιου χρώματος ανήκουν στην ίδια ομάδα.

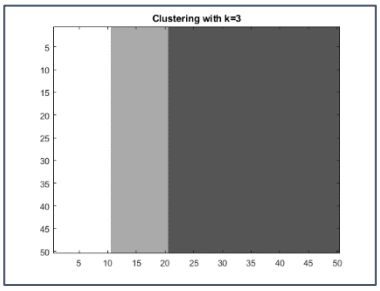


Demo 2 - 1: Αρχική εικόνα

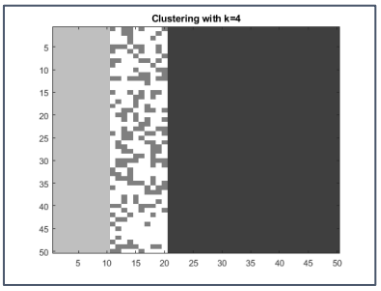
Στο πρώτο πείραμα η εικόνα είναι φανερό πως πρέπει να χωριστεί σε τρείς ομάδες, καθώς υπάρχουν 3 διακριτά χρώματα. Οι εικόνες αποτελέσματος είναι grey-scale και κάθε διαφορετική απόχρωση του γκρι είναι μία ξεχωριστή ομάδα.



Demo 2 - 2: Ομαδοποίηση με 2 ομάδες



Demo 2 - 3: Ομαδοποίηση με 3 ομάδες



Demo 2 - 4: Ομαδοποίηση με 4 ομάδες

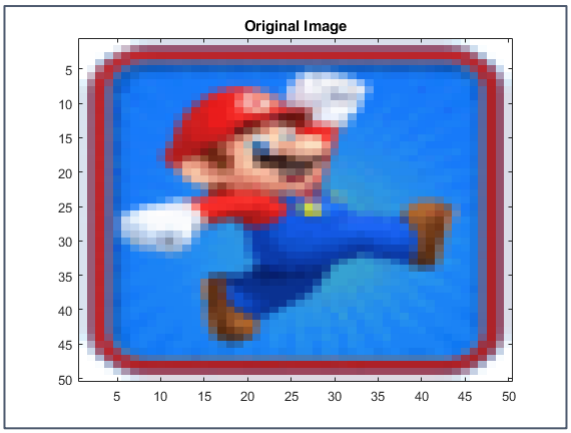
Σε αναλογία με το πρώτο demo στην περίπτωση της ομαδοποίησης 2 ομάδων, δύο διαφορετικά χρώματα *(μπλε και πράσινο)* ομαδοποιήθηκαν μαζί, καθιστώντας την ομαδοποίηση υποβέλτιστη.

Η ομαδοποίηση με 3 ομάδες είναι η βέλτιστη και χωρίζει τέλεια τα διαφορετικά χρώματα.

Ενδιαφέρουσα περίπτωση αποτελεί η ομαδοποίηση με 4 ομάδες καθώς οι δύο ομάδες είναι καλά χωρισμένες αλλά οι άλλες δύο συγχέονται μεταξύ τους. Παρόμοια περίπτωση με το πρώτο demo που τα σημεία ίδια περιοχής ομαδοποιούνται σε διαφορετικές ομάδες.

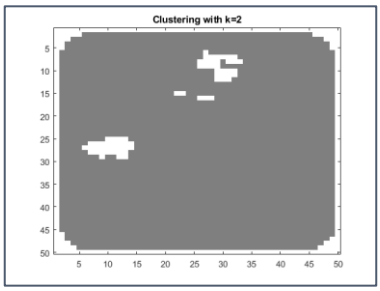
Παράδειγμα 2ο

Η παραπάνω περίπτωση είναι ακραία στην φύση της, καθώς συνήθως οι εικόνες αποτελούνται από παραπάνω χρώματα τα οποία δεν είναι καλώς χωρισμένα. Δεδομένων αυτών, εκτελέστηκε και πείραμα σε μία πιο απτή εικόνα, όπου υπάρχουν πολλά διαφορετικά χρώματα τα οποία βρίσκονται σε διάφορα σημεία στην εικόνα.

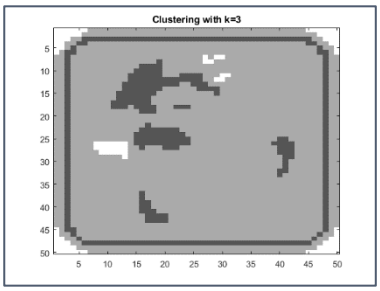


Demo 2 - 5: Αρχική εικόνα

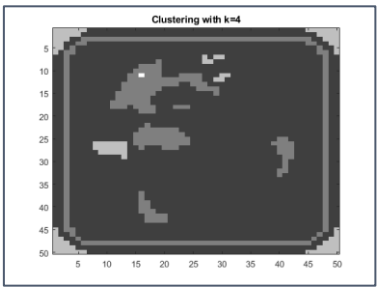
Είναι φανερό πως τα χρώματα που μοιάζουν πρέπει να ομαδοποιηθούν, για παράδειγμα τα κόκκινα όρια της εικόνας με τα κόκκινα ρούχα του χαρακτήρα θα έπρεπε να ανήκουν στην ίδια ομάδα. Όπως επίσης και τα μπλε με τα ανάλογα μπλε.



Demo 2 - 6: Ομαδοποίηση με 2 ομάδες



Demo 2 - 7: Ομαδοποίηση με 3 ομάδες



Demo 2 - 8: Ομαδοποίηση με 4 ομάδες

Τα αποτελέσματα των παραπάνω πειραμάτων προσφέρουν περισσότερη λεπτομέρεια για την λειτουργία του αλγορίθμου. Είναι φανερό από την ομαδοποίηση με 2 ομάδες πως τα άσπρα pixel διαφέρουν περισσότερο από όλα τα έγχρωμα για αυτό και ομαδοποιούνται σε διαφορετικές ομάδες, οι ομάδες αυτές θα μπορούσε να λέγονται «Άσπρα pixel» και «Έγχρωμα pixel» αναλόγως.

Στην δεύτερη περίπτωση, δηλαδή στην ομαδοποίηση με 3 ομάδες, εκτός από τον διαχωρισμό άσπρων και έγχρωμων pixel, υπάρχει και διαχωρισμός μεταξύ κόκκινων και λοιπών χρωμάτων. Το ενδιαφέρον σε αυτό το παράδειγμα είναι, πως τα καφέ υποδήματα είναι εμφανώς διαφορετικού χρώματος από τα κόκκινα ρούχα του χαρακτήρα, αλλά καθώς το κόκκινο με το καφέ είναι σχετικά παρεμφερή χρώματα, ο αλγόριθμος τα ομαδοποιεί στην ίδια ομάδα.

Στο τελευταίο παράδειγμα ο αλγόριθμος φαίνεται πως αποτυγχάνει, καθώς η μόνη νέα ομάδα που εμφανίζεται, εμφανίζεται στο καπέλο του χαρακτήρα και αποτελείται μόνο από ένα pixel, ίσως με ονομασία «Απόλυτο κόκκινο».

Normalized cuts

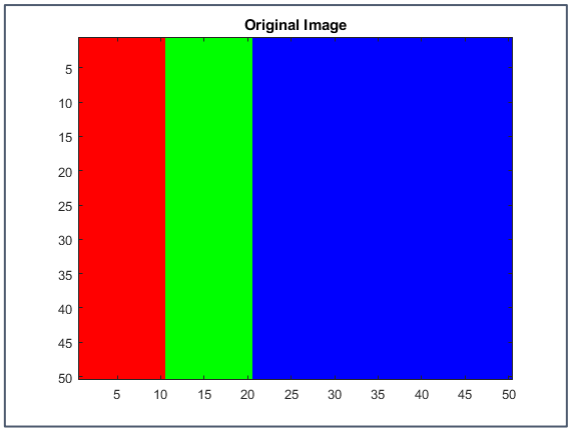
Ο αλγόριθμος *normalized cuts* διαφέρει ελάχιστα από τον *spectral clustering* αλλά παράγει πολύ καλύτερα αποτελέσματα. Η διαδικασία μπορεί να είναι αναδρομική ή μη. Στα πλαίσια της εργασία η μη αναδρομική εκδοχή ελέγχει εκ των προτέρων πόσες ομάδες θα δημιουργηθούν, ενώ η αναδρομική εκδοχή αποφασίζει βάσει δύο μετρικών πόσες ομάδες πρέπει να δημιουργηθούν. Οι μετρικές αυτές είναι ο αριθμός των pixel σε μία ομάδα και η μετρική nC*ut* η οποία δείχνει κατά πόσο όμοια στοιχεία ίδιας ομάδας.

DEMO 3A

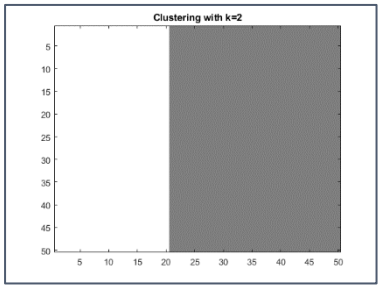
Στο παρόν demo θα συγκριθούν οι δύο μέθοδοι ομαδοποίησης. Πιο συγκεκριμένα, θα συγκριθεί η μέθοδος *spectral clustering* με την μη-αναδρομική μέθοδο *normalized-cuts*. Τα παραδείγματα που θα παρατεθούν, είναι τα ίδια με τα παραδείγματα του demo 2.

Παράδειγμα 1ο

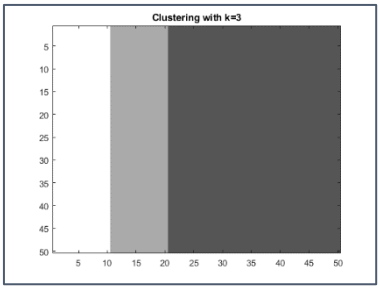
Στο παράδειγμα αυτό χρησιμοποιείται η δοκιμαστική εικόνα με τα 3 διακριτά χρώματα.



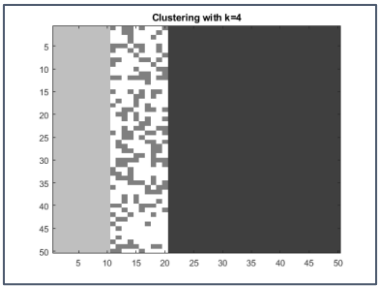
Demo 3 - 1: Αρχική εικόνα



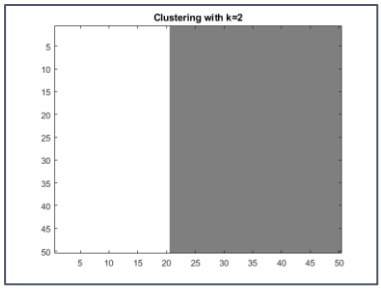
Demo 3 - 2: Spectral Clustering k=2



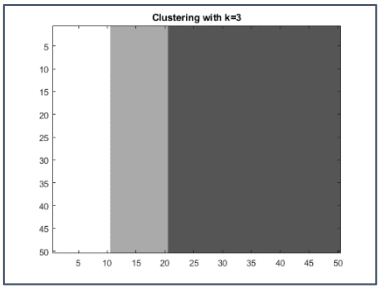
Demo 3 - 3: Spectral Clustering k=3



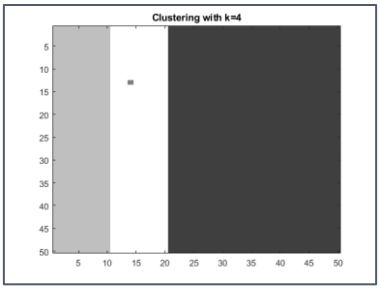
Demo 3 - 4: Spectral Clustering k=4



Demo 3 - 5: nCuts k=2



Demo 3 - 6: nCuts k=3

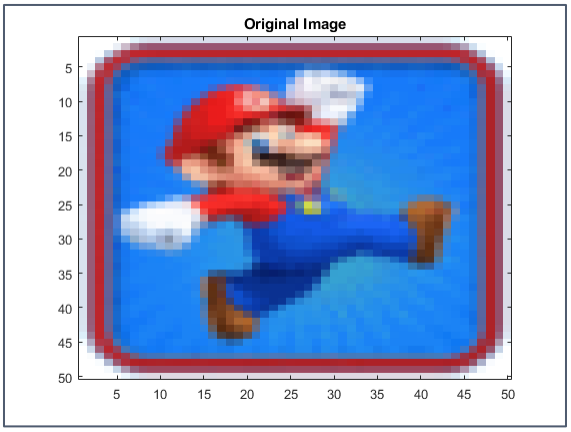


Demo 3 - 7: nCuts k=4

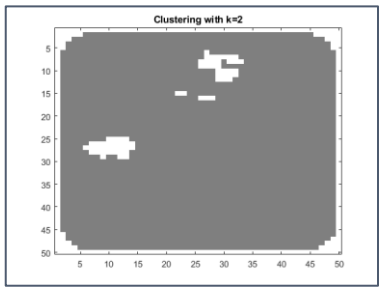
Στις πρώτες δύο περιπτώσεις, δηλαδή στις περιπτώσεις ομαδοποίησης με 2 και 3 ομάδες, οι δύο αλγόριθμοι δεν δείχνουν να έχουν διαφορά. Στην περίπτωση των επιπλέων ομάδων όμως ο αλγόριθμος *nCuts* είναι φανερά ανώτερος από τον *Spectral Clustering*, καθώς ενώ υπάρχει η επιπλέον ομάδα, ανήκει σε αυτήν μόνο ένα pixel αντί για μία συλλογή από pixel.

Παράδειγμα 2ο

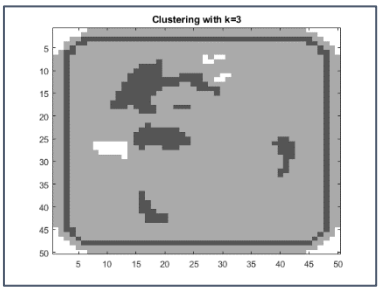
Χρήση πιο απτής εικόνας.



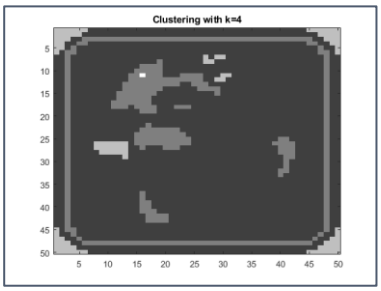
Demo 3 - 8: Αρχική εικόνα



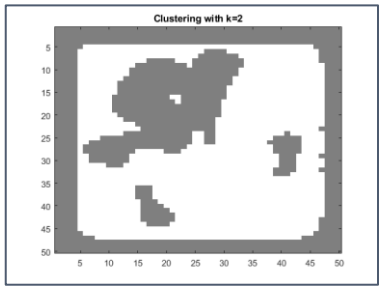
Demo 3 - 9: Spectral Clustering k=2



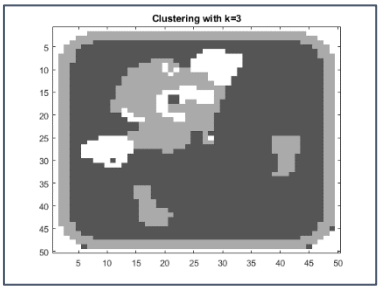
Demo 3 - 10: Spectral Clustering k=3



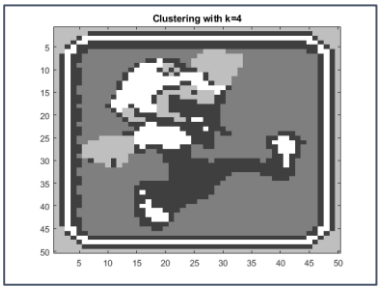
Demo 3 - 11: Spectral Clustering k=4



Demo 3 - 12: nCuts k=2



Demo 3 - 13: nCuts k=3



Demo 3 - 14: nCuts k=4

Είναι φανερό πως η ομαδοποίηση του *nCuts* είναι ανώτερη από τον *Spectral Clustering*. Ο παραπάνω ισχυρισμός είναι πιο φανερός στην ομαδοποίηση 4 ομάδων, όπου σε αντίθεση με τον *Spectral Clustering* μπορεί κανείς να διακρίνει λεπτομέρεια στο αποτέλεσμα του *nCuts*, δηλαδή ο χαρακτήρας μπορεί να αναγνωριστεί μετά τον αλγόριθμο.

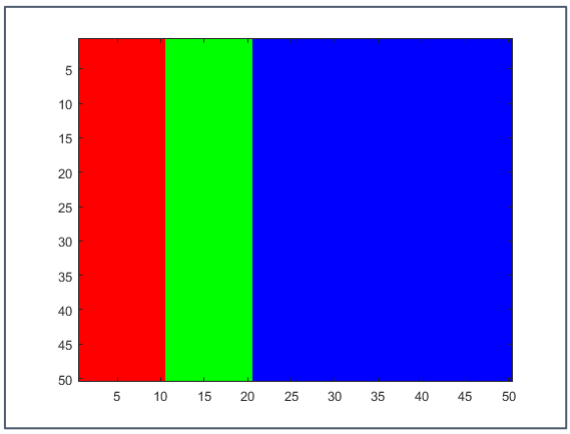
Επιπλέον σε αντίθεση με τον *Spectral Clustering* φαίνεται πως ο αλγόριθμος *nCuts* στην ομαδοποίηση με 2 ομάδες, χώρισε τα pixels σε «Έντονα» και «Σκούρα» αντί για «Άσπρα» και «Έγχρωμα».

DEMO 3B

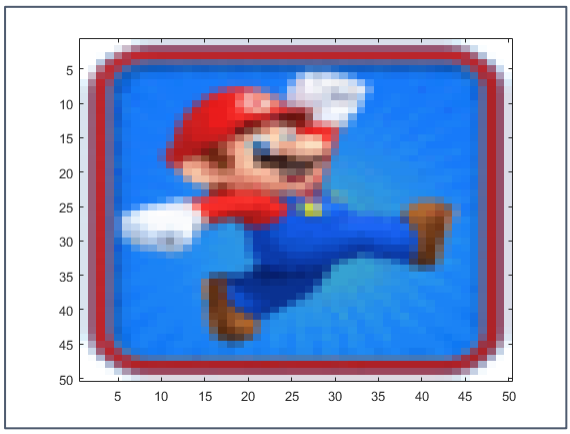
Η αναδρομική εκδοχή του αλγορίθμου *nCuts* αρχικά διχοτομεί την αρχική εικόνα σε δύο ομάδες και στην συνέχεια την κάθε ομάδα σε δύο υποομάδες. Ο αλγόριθμος σταματά όταν τα μέλη μίας ομάδας είναι λιγότερα από ένα επιτρεπτό όριο ή όταν οι δύο ομάδες είναι αρκετά όμοιες μεταξύ τους.

Σε αυτό το demo αναλύεται το πρώτο βήμα του αναδρομικού αλγορίθμου, δηλαδή η διχοτόμηση της εικόνας σε δύο ομάδες, καθώς και ο υπολογισμός της μετρικής *nCut* η οποία δηλώνει κατά πόσο είναι όμοιες οι ομάδες.

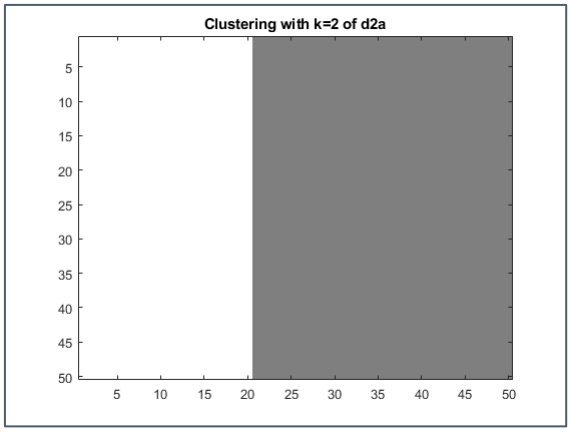
Προφανώς τα αποτελέσματα της διχοτόμησης προαναφέρθηκαν και σε προηγούμενα demo.



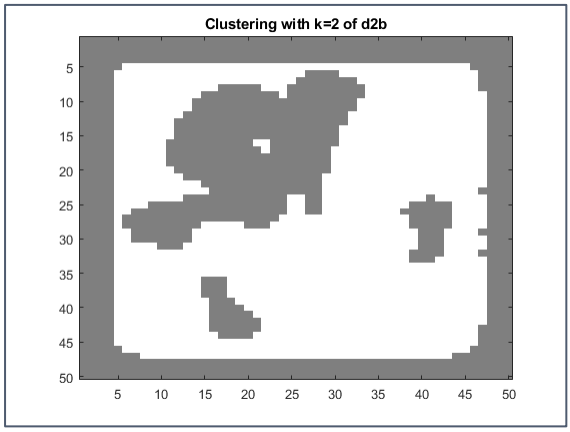
Demo 3 - 15: Αρχική εικόνα



Demo 3 - 16: Αρχική εικόνα



Demo 3 - 17: Διχοτόμηση ακραίας περίπτωσης



Demo 3 - 18: Διχοτόμηση χαρακτήρα

Επιπλέον υπολογίστηκαν και οι μετρικές *nCut* για κάθε περίπτωση.

- $\mathrm{nCut}_1 = 5.092379 \times 10^{-1}$
- $\mathrm{nCut}_2 = 7.852853 \times 10^{-1}$

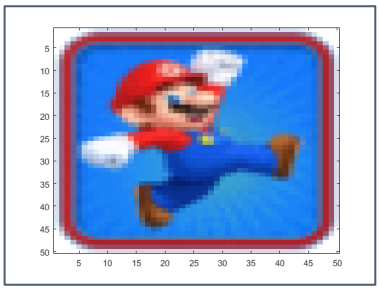
Η μετρική *nCut* είναι το άθροισμα της αποσυσχέτισης των στοιχείων ανά ομάδα, με την μονάδα να σημαίνει τέλεια αποσυσχέτιση και το μηδέν να σημαίνει τέλεια συσχέτιση *(οι ακέραιες περιπτώσεις δεν επιτυγχάνονται καθώς πάντα υπάρχει συσχέτιση, ενώ δεν έχει νόημα η ομαδοποίηση αν όλα είναι ίδια)*. Επομένως η μετρική *nCut* πρέπει να ελαχιστοποιηθεί έτσι ώστε οι ομάδες να εμπεριέχουν στοιχεία όσο το δυνατόν συσχετισμένα.

Τα αποτελέσματα, λοιπόν, είναι λογικά. Στην ακραία περίπτωση τριών χρωμάτων η μετρική *nCut* είναι αρκετά χαμηλή καθώς τα pixel στην «μπλε» ομάδα είναι μεταξύ τους ίδια και αυτό έχει ως αποτέλεσμα την μείωση του *nCut*. Ενώ στην πιο γενική δεύτερη περίπτωση τα pixel μεταξύ τους δεν είναι ίδια αλλά όμοια, γεγονός που αυξάνει την μετρική.

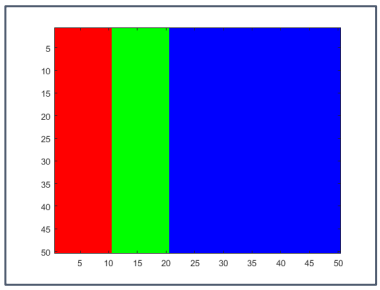
DEMO 3C

Στο τελευταίο demo εκτελείται η αναδρομική εκδοχή του *nCuts* η οποία χωρίζει την εικόνα σε αυθαίρετο αριθμό ομάδων, δεδομένων των κατωφλίων των ελαχίστων στοιχείων σε μία ομάδα και της μετρικής *nCut*.

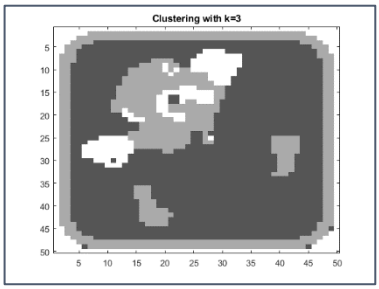
Η εκφώνηση αναφέρει να χρησιμοποιηθεί κατώφλι για την nCut 0.5 *δηλαδή ο αλγόριθμος να σταματά αν* $nCut > 0.5$. *Στα παραδείγματα που αναλύθηκαν η μετρική λαμβάνει τιμές μεγαλύτερες του* 0.5 *επομένως χρησιμοποιήθηκε μεγαλύτερο κατώφλι. Πιο συγκεκριμένα, χρησιμοποιήθηκε κατώφλι* 0.79 *και για τις δύο περιπτώσεις.*



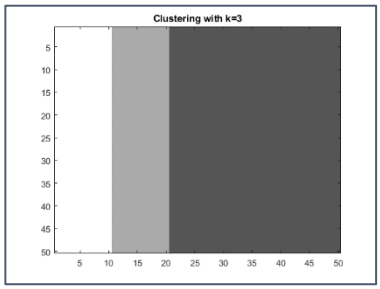
Demo 3 - 19: Αρχική Εικόνα



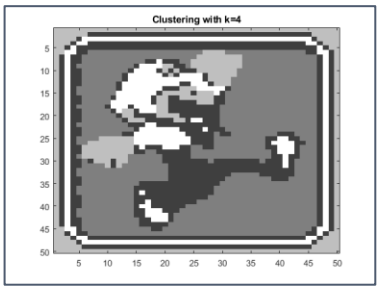
Demo 3 - 20: Αρχική Εικόνα



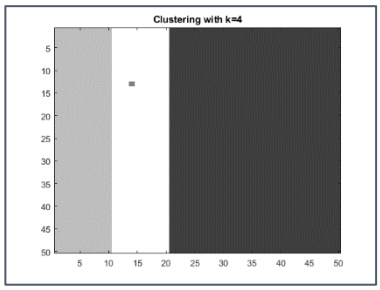
Demo 3 - 21: nCuts k=3



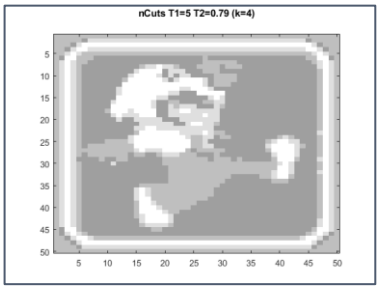
Demo 3 - 22: nCuts k=3



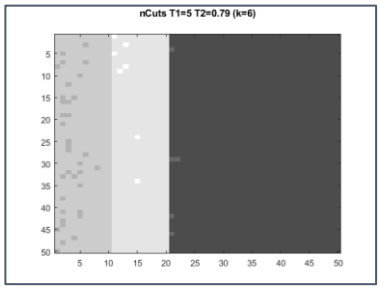
Demo 3 - 23: nCuts k=4



Demo 3 - 24: nCuts k=4



Demo 3 - 25: Recursive nCuts



Demo 3 - 26: Recursive nCuts

Τα αποτελέσματα στις δύο διαφορετικές περιπτώσεις αντικρούονται. Στην ακραία περίπτωση των τριών χρωμάτων η ομαδοποίηση με 3 ομάδες του απλού *nCuts* αλγορίθμου είναι η βέλτιστη ομαδοποίηση, ενώ στην περίπτωση των 4 ομάδων υπάρχει ένα pixel κακώς ομαδοποιημένο, αφού δεν μπορούν τρία χρώματα να χωριστούν σωστά σε 4 ομάδες. Στην περίπτωση του αναδρομικού αλγορίθμου υπάρχουν πολλά pixel κακός ομαδοποιημένα. Τονίζεται πως οι τρείς περιοχές είναι καλώς χωρισμένες, απλώς τα εσωτερικά τους σημεία χωρίζονται σε υποομάδες.

Το αποτέλεσμα είναι λογικό, καθώς ο αλγόριθμος επέλεξε να σταματήσει καθώς η μετρική *nCut* έγινε αρκετά μεγάλη, αλλά η περεταίρω διχοτόμηση είχε ήδη συμβεί. Ίσως ο έλεγχος της *nCut* να πρέπει να γίνεται στο προηγούμενο επίπεδο της αναδρομής ώστε να αποφεύγονται τέτοια σφάλματα. Δηλαδή το αποτέλεσμα να επιστρέφεται από την συνάρτηση και να ελέγχεται από το προηγούμενο επίπεδο.

Στην πιο γενική περίπτωση του χαρακτήρα η αναδρομική ομαδοποίηση φαίνεται να είναι καλύτερη από την μη-αναδρομική εκδοχή. Ο αριθμός ομάδων είναι ο ίδιος και στις δύο περιπτώσεις.

Extra

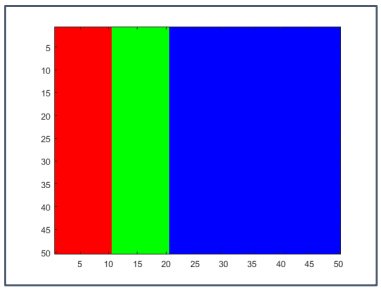
ΑΝΑΔΡΟΜΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ

Αναφέρθηκε πως ο αναδρομικός αλγόριθμος *nCuts* θα έπρεπε να υλοποιηθεί αλλιώς. Η διαφορετική υλοποίηση αυτή ουσιαστικά απορρίπτει την τελευταία αναδρομή καθώς την θεωρεί υποβέλτιστη.

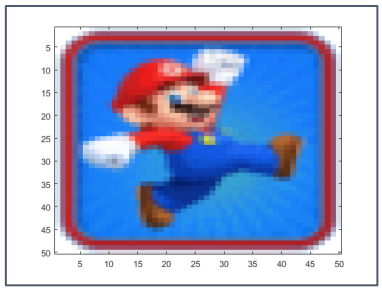
```
function [cI, done] = recursiveNCuts(W, T1, T2)
%% Code

% Binary cluster the affinity matrix
cI = myNCuts(W, 2);

% Find the unique labels of groups
I = unique(cI);

% Split the groups
A = (cI == I(1));
B = ~A;

% Check if clusters have enough elements
M = min(sum(A), sum(B));
if (M < T1)
    done = true;
    return
end

% Check if the nCut metric is small enough
N = calculateNcut(W , cI);
if (N > T2)
    done = true;
    return
end

% Select all entries in the affinity matrix that belong in group A
WA = W(A, A);
% Replace all labels of the A group with the results of the next recursion
% The new labels must be 2*L + L' so that they wont colide with the
% previous labeling, or the labeling of B group
[ncI, done] = recursiveNCuts(WA, T1, T2);
if ~done
    cI(A) = 2*I(1) + ncI;
end

% Select all entries in the affinity matrix that belong in group B
WB = W(B, B);
% Replace all labels of the B group with the results of the next recursion
% The new labels must be 2*L + L' so that they wont colide with the
% previous labeling, or the labeling of A group
[ncI, done] = recursiveNCuts(WB, T1, T2);
if ~done
    cI(B) = 2*I(2) + ncI;
end

done = false;
end
```

Extra 1: Παραλλαγή κώδικα

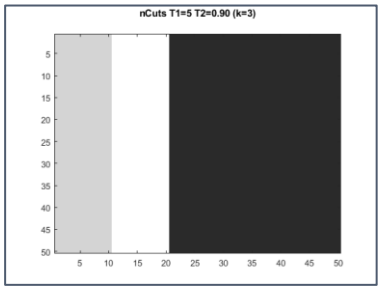
Ο αλγόριθμος αυτός παράγει ουσιαστικά τα ίδια αποτελέσματα χωρίς το τελευταίο βήμα της αναδρομής.



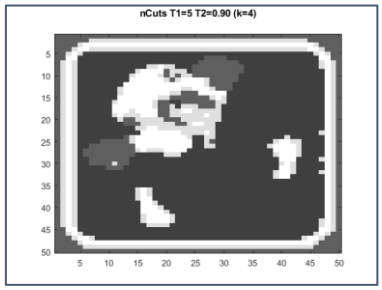
Extra 2: Αρχική εικόνα



Extra 3: Αρχική εικόνα



Extra 4: Ομαδοποίηση



Extra 5: Ομαδοποίηση

Είναι φανερό πως η ομαδοποίηση στην πρώτη περίπτωση δεν έχει παράσιτα και πως η δεύτερη δεν άλλαξε. Για να μην αλλάξει όμως η δεύτερη ομαδοποίηση το κατώφλι της μετρικής *nCut* αυξήθηκε στο 0.9.

ΚΩΔΙΚΑΣ

Το κάθε script της MATLAB είναι επαρκώς σχολιασμένος, επομένως δεν σχολιάζεται περεταίρω στην αναφορά. Επιπλέον μπορεί να βρεθεί αναρτημένος και στο αποθετήριό μου στο GitHub.