ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Ψηφιακή Επεξεργασία Εικόνας Αναφορά

Εργασία 3

Διακολουκάς Δημήτριος
ΑΕΜ 10642

Email: ddiakolou@ece.auth.gr

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή και ανάλυση προβλήματος

Η παρούσα εργασία πραγματοποιήθηκε στα πλαίσια του μαθήματος **Ψηφιακή Επεξεργασία Εικόνας** και έχει ως βασικό αντικείμενο την υλοποίηση και αξιολόγηση αλγορίθμων για image segmentation μέσω τεχνικών Spectral Clustering και Normalized Cuts.

Πιο συγκεκριμένα, εξετάζονται και υλοποιούνται οι εξής βασικές τεχνικές:

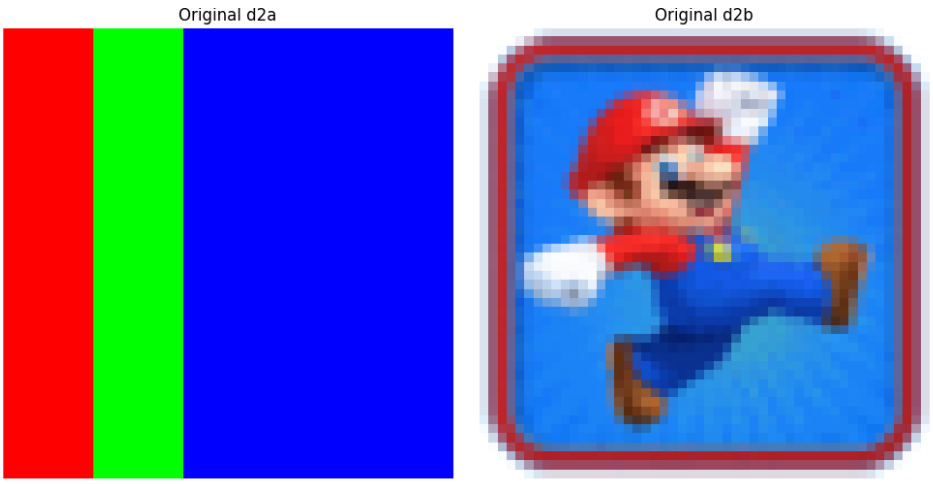
- *Αναπαράσταση εικόνας ως γράφος:* Κάθε pixel της εικόνας θεωρείται κορυφή (node) ενός πλήρως συνδεδεμένου μη-κατευθυντικού γράφου, με τα βάρη των ακμών να προκύπτουν από την εκθετική απόσβεση της Ευκλείδειας απόστασης των τιμών φωτεινότητας μεταξύ των pixel.
- *Spectral Clustering:* Μέθοδος ομαδοποίησης που βασίζεται στην επίλυση του προβλήματος ιδιοτιμών του Λαπλασιανού πίνακα του γράφου. Τα πρώτα k ιδιοδιανύσματα εισάγονται σε k-meansγια την παραγωγή k clusters.
- *Normalized Cuts (n-cuts):* Εναλλακτική προσέγγιση του spectral clustering όπου επιλύεται το γενικευμένο πρόβλημα ιδιοτιμών $Lx = \lambda Dx$. Περιλαμβάνονται τόσο η μη-αναδρομική εκδοχή όσο και η αναδρομική, με κριτήρια τερματισμού βασισμένα στο μέγεθος του cluster και τη μετρική Ncut.
- *Υπολογισμός Ncut Metric:* Μετρά την ποιότητα του διαχωρισμού σε δύο υποσύνολα, υπολογίζοντας την εσωτερική συνοχή σε κάθε ομάδα και την μεταξύ τους συνδεσιμότητα.

Για την πειραματική αξιολόγηση χρησιμοποιήθηκαν οι εικόνες d2a και d2b που παρέχονται μέσω του αρχείου `dip_hw_3.mat`. Για κάθε τεχνική κατασκευάστηκαν αντίστοιχα Python scripts που υλοποιούν τις συναρτήσεις και παρουσιάζουν αποτελέσματα με μεταβλητές τιμές του k ή των κατωφλίων T_1 και T_2.

Η υλοποίηση ακολουθεί πιστά τις προδιαγραφές της εκφώνησης και χρησιμοποιεί αποκλειστικά βασικές βιβλιοθήκες της Python, όπως NumPy, SciPy και scikit-learn, αποφεύγοντας έτοιμες συναρτήσεις segmentation ή clusteringπέρα από το KMeans.

Ο στόχος της εργασίας είναι αφενός η κατανόηση της θεωρίας πίσω από τις τεχνικές spectral και graph-based clusteringκαι αφετέρου η ορθή και αποδοτική υλοποίησή τους με δυνατότητα πειραματισμού και αξιολόγησης αποτελεσμάτων σε εικόνες RGB.

Σε αυτό το σημείο αξίζει να επισυνάψουμε και την αρχική εικόνα η οποία περιγράφεται από το `dip_hw_3.mat` αρχείο και έγινε extract για reference σε custom `original_dotmat.py` αρχείο.



Σχήμα 1.1: Οπτικοποίηση των εικόνων d2a και d2b από το αρχείο `dip_hw_3.mat`.

Κεφάλαιο 2

Εικόνες ως Γράφοι και Spectral Clustering

Σε αυτή την ενότητα περιγράφεται η θεμελιώδης ιδέα της μετατροπής μιας εικόνας σε γράφο, μια τεχνική απαραίτητη για την εφαρμογή αλγορίθμων φασματικής τμηματοποίησης όπως οι Spectral Clustering και Normalized Cuts αλλά και της τεχνικής Spectral Clustering. Η υλοποίηση βασίζεται στην κατασκευή ενός πίνακα γειτνίασης (affinity matrix), όπου κάθε pixel αποτελεί έναν κόμβο και οι ακμές μεταξύ τους αντανακλούν μετρικές ομοιότητας.

2.1 Συνάρτηση για Εικόνα σε Γράφο

Η μετάβαση από μια αναπαράσταση εικόνας σε γράφο αποτελεί απαραίτητο βήμα για την εφαρμογή φασματικών μεθόδων ομαδοποίησης, όπως Spectral Clustering και Normalized Cuts. Η συνάρτηση `image_to_graph()` αναλαμβάνει να κατασκευάσει τον πίνακα γειτνίασης (affinity matrix) που περιγράφει έναν μη-κατευθυντικό πλήρως συνδεδεμένο γράφο, όπου κάθε κόμβος αντιστοιχεί σε ένα pixel της εικόνας.

2.1.1 Θεωρητική Ανάλυση

Έστω ότι η είσοδος είναι μια εικόνα `img_array` διαστάσεων $M \times N \times C$, όπου C ο αριθμός των καναλιών (συνήθως $C = 3$ για έγχρωμες εικόνες RGB). Η μετατροπή της εικόνας σε γράφο περιλαμβάνει τα εξής στάδια:

1. *Μετασχηματισμός των pixels σε διανύσματα:* Κάθε pixel θεωρείται σημείο $x_i \in \mathbb{R}^C$ (με C χαρακτηριστικά). Η εικόνα επίπεδο-επιπέδου μετατρέπεται σε έναν πίνακα διαστάσεων $(MN) \times C$.

2. *Υπολογισμός αποστάσεων μεταξύ των pixels:* Για κάθε ζεύγος κόμβων (i, j) υπολογίζεται η Ευκλείδεια απόσταση μεταξύ των διανυσμάτων x_i, x_j ως:

$$d(i, j) = \|x_i - x_j\| = \sqrt{\|x_i\|^2 + \|x_j\|^2 - 2x_i \cdot x_j}$$

3. *Κατασκευή του affinity matrix:* Η ομοιότητα μεταξύ δύο κόμβων ορίζεται ως:

$$A(i, j) = e^{-d(i,j)}$$

Το αποτέλεσμα είναι ένας πλήρως συμμετρικός πίνακας $A \in \mathbb{R}^{MN \times MN}$ όπου κάθε τιμή εκφράζει την ομοιότητα μεταξύ δύο pixels.

4. *Ιδιότητες πίνακα:* Ο πίνακας που κατασκευάζεται είναι:

   - *Πυκνός* (fully-connected)
   - *Συμμετρικός* ($A = A^\top$)
   - *Θετικά ορισμένος* (καθώς $e^{-d(i,j)} > 0$ για κάθε i, j)

2.1.2 Ανάλυση Κώδικα και Ψευδοκώδικας

Η υλοποίηση βασίζεται στο παρακάτω διάγραμμα ψευδοκώδικα, που περιγράφει βήμα προς βήμα τη λογική της `image_to_graph()`:

Ψευδοκώδικας `image_to_graph()`

```
1. [M, N, C] ← shape(img_array)
2. pixels ← reshape(img_array) to shape (MN, C)

3. sq_norms ← row squared L2 norms of pixels (shape MN×1)

4. D_sq ← pairwise squared Euclidean distances:
          D_sq = sq_norms + sq_norms.T - 2 * pixels.dot(pixels.T)

5. D ← sqrt(max(D_sq, 0))   # numerical stability

6. affinity_mat ← exp(-D)   # similarity matrix

7. return affinity_mat
```

Η παραπάνω συνάρτηση κατασκευάζει επιτυχώς τον γράφο, ο οποίος στη συνέχεια χρησιμοποιείται από τις συναρτήσεις `spectral_clustering()` και `n_cuts_recursive()` για την υλοποίηση της τμηματοποίησης εικόνας.

Ο αντίστοιχος κώδικας Python βρίσκεται στο αρχείο `image_to_graph.py` .

2.2 Συνάρτηση για Spectral Clustering

Η φασματική ομαδοποίηση (Spectral Clustering) βασίζεται στην ιδιοφασματική ανάλυση του πίνακα γειτνίασης (affinity matrix) και επιτρέπει την εύρεση μη γραμμικά διαχωρίσιμων ομάδων. Η συνάρτηση `spectral_clustering()` υλοποιεί τα βασικά στάδια του αλγορίθμου που περιγράφονται στη θεωρία, με είσοδο έναν συμμετρικό affinity πίνακα και έναν ακέραιο αριθμό clusters k.

2.2.1 Θεωρητική Ανάλυση

Η μέθοδος Spectral Clustering βασίζεται στην ιδιοφασματική ανάλυση του γράφου που περιγράφεται από τον πίνακα γειτνίασης (ή affinity matrix) $W \in \mathbb{R}^{n \times n}$. Ο γράφος υποτίθεται ότι είναι μη-κατευθυντικός, πλήρως συνδεδεμένος και με θετικές ακμές.
Η διαδικασία περιλαμβάνει τα παρακάτω βήματα:

1. *Κατασκευή του διαγώνιου πίνακα βαθμού D:* Ορίζεται ως:

$$D(i,i) = \sum_j W(i,j), \quad D(i,j) = 0 \text{ για } i \neq j$$

   Ο πίνακας D περιέχει στον διαγώνιο την τιμή του βαθμού κάθε κόμβου (το άθροισμα των βαρών όλων των εισερχόμενων ακμών).

2. *Υπολογισμός Λαπλασιανού πίνακα L:* Ο μη-κανονικοποιημένος Laplacian δίνεται από:

$$L = D - W$$

   Ο πίνακας L είναι συμμετρικός και θετικά ημι-ορισμένος. Οι ιδιοτιμές του είναι μη αρνητικές, και η μικρότερη ιδιοτιμή είναι πάντα 0 (με αντίστοιχο ιδιοδιάνυσμα το $\mathbf{1}$).

3. *Επίλυση του ιδιοτιμικού προβλήματος:* Λύνουμε το πρόβλημα:

$$Lx = \lambda x$$

   και επιλέγουμε τα k ιδιοδιανύσματα που αντιστοιχούν στις k μικρότερες ιδιοτιμές (εκτός της μηδενικής, εφόσον είναι απομονωμένη). Αυτά περιέχουν τις πιο σημαντικές πληροφορίες σχετικά με τη δομή του γράφου.

4. *Κατασκευή πίνακα χαρακτηριστικών U:* Ορίζουμε τον πίνακα:

$$U = [u_1\, u_2\, \ldots\, u_k] \in \mathbb{R}^{n \times k}$$

   όπου κάθε στήλη είναι ένα από τα επιλεγμένα ιδιοδιανύσματα. Ο πίνακας U θεωρείται ως νέα αναπαράσταση (embedding) των κόμβων σε k-διάστατο χώρο.

5. *Ομαδοποίηση με k-means:* Κάθε γραμμή του U (δηλαδή κάθε κόμβος του γράφου) θεωρείται ως ένα k-διάστατο διάνυσμα $y_i \in \mathbb{R}^k$. Εφαρμόζεται ο αλγόριθμος k-means στα n σημεία $\{y_i\}_{i=1}^n$ ώστε να εκχωρηθεί κάθε κόμβος σε ένα από τα k clusters:

$$C_1, C_2, \ldots, C_k$$

2.2.2 Ανάλυση Κώδικα και Ψευδοκώδικας

Η λογική του αλγορίθμου φαίνεται συγκεντρωτικά στον παρακάτω ψευδοκώδικα:

Ψευδοκώδικας `spectral_clustering()`

```
1. [n, _] ← shape(affinity_mat)
2. assert square matrix and k < n

3. W ← sparse CSR matrix from affinity_mat
4. W.setdiag(0)  # zero self-loops
5. D ← diagonal matrix of degrees: D(i) = sum_j W(i,j)

6. L ← D - W     # Laplacian

7. [vals, vecs] ← eigs(L, M=D, k=k, which='SM')
   U ← real(vecs)  # take real part if complex

8. kmeans ← KMeans(n_clusters=k, random_state=1)
9. kmeans.fit(U)
10. return kmeans.labels_.astype(float)
```

Σημειώσεις Υλοποίησης:

- Η χρήση του παραμέτρου `which='SM'` στην `scipy.sparse.linalg.eigs()` ζητά τις k ιδιοτιμές με το μικρότερο μέτρο.
- Ο Laplacian μπορεί να είναι είτε ο κανονικός είτε ο κανονικοποιημένος. Εδώ επιλέγεται ο μη κανονικοποιημένος.
- Για λόγους αναπαραγωγιμότητας, το `random_state` του `KMeans` ορίζεται ίσο με 1.

Η παραπάνω συνάρτηση αποτελεί βασικό δομικό στοιχείο τόσο για την εφαρμογή Spectral Clustering όσο και για την συγκριση που μας ζητείται αργότερα με την επαναληπτική εφαρμογή της μεθόδου για τμηματοποίηση εικόνας.

Ο αντίστοιχος κώδικας Python βρίσκεται στο αρχείο `spectral_clustering.py` .

2.3 Ανάλυση Αποτελεσμάτων και Οπτικοποίησης

Σε αυτό το section πρόκειται να αναδείξουμε τα αποτελέσματα που προέκυψαν από την εφαρμογή τον προεναφερθέντων υλοποιημένων συναρτήσεων στα αρχεία `demo1.py` και `demo2.py`.

2.3.1 Περιγραφή και λειτουργικότητα demo1.py

Το αρχείο `demo1.py` χρησιμοποιείται για την πειραματική επαλήθευση της λειτουργίας της συνάρτησης `spectral_clustering()`, όπως ορίζεται στην προηγούμενη section. Ειδικότερα, φορτώνει τον affinity matrix από το αρχείο `dip_hw_3.mat` (μεταβλητή `d1a`), εφαρμόζει φασματική ομαδοποίηση για τρεις διαφορετικές τιμές $k = 2, 3, 4$, και παρουσιάζει τα εξής:

- Τις πρώτες 20 ετικέτες που παράγονται από την κλήση της `spectral_clustering()` για κάθε τιμή του k.

- Το μέγεθος κάθε cluster, δηλαδή πόσα σημεία περιλαμβάνει.
- Δύο διαγράμματα για κάθε k:
  - *Διάγραμμα ετικετών*: Παρουσιάζει γραφικά τις πρώτες 20 ετικέτες cluster που εκχωρήθηκαν.
  - *Διάγραμμα μεγεθών*: Παρουσιάζει το μέγεθος κάθε cluster υπό μορφή bar plot.

Η χρήση της παραμέτρου `random_state = 1` στη συνάρτηση `KMeans()` εξασφαλίζει την αναπαραγωγή των αποτελεσμάτων σε κάθε εκτέλεση. Η επόμενη υποενότητα παραθέτει και σχολιάζει τα αποτελέσματα που προέκυψαν.

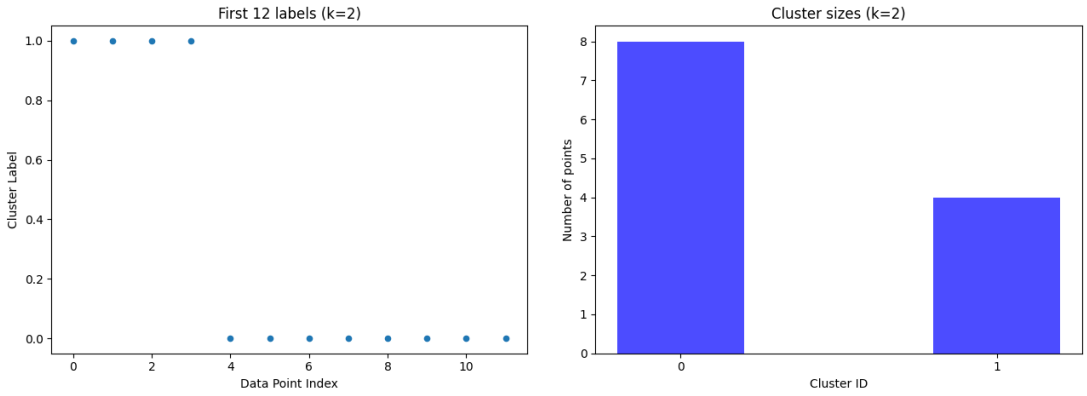
Παράθεση Αποτελεσμάτων

Η εκτέλεση του `demo1.py` για τιμές $k = 2, 3, 4$ παρήγαγε τις παρακάτω ετικέτες και μεγέθη clusters:

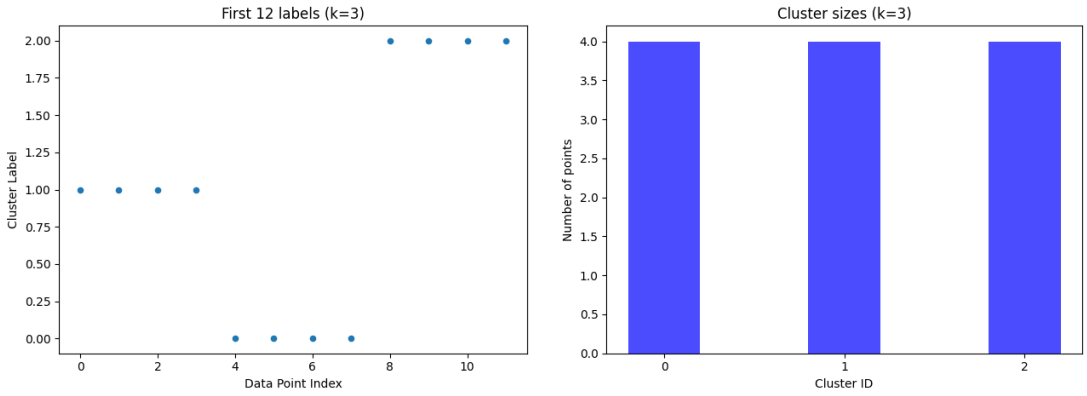
- $k = 2$*:*
  - Πρώτες 20 ετικέτες: `[1 1 1 1 0 0 0 0 0 0 0 0]`
  - Μέγεθος clusters: `[8 4]`
  - *Παρατήρηση:* Το σύνολο των 12 σημείων διαχωρίστηκε σε 2 ομάδες με σχετικά ανισοβαρή κατανομή.
- $k = 3$*:*
  - Πρώτες 20 ετικέτες: `[1 1 1 1 0 0 0 0 2 2 2 2]`
  - Μέγεθος clusters: `[4 4 4]`
  - *Παρατήρηση:* Η ομαδοποίηση είναι αρκετά ισορροπημένη, υποδεικνύοντας φυσικό διαχωρισμό των δεδομένων σε 3 ίσες κατηγορίες.
- $k = 4$*:*
  - Πρώτες 20 ετικέτες: `[2 3 3 3 0 0 0 0 1 1 1 1]`
  - Μέγεθος clusters: `[4 4 1 3]`
  - *Παρατήρηση:* Ο διαχωρισμός είναι πιο ανισοβαρής, με ένα cluster που περιλαμβάνει μόνο ένα σημείο. Αυτό μπορεί να υποδεικνύει υπερδιάσπαση για $k = 4$.

Τα αποτελέσματα δείχνουν πως η επιλογή του αριθμού των clusters k επηρεάζει σημαντικά τη σταθερότητα και την ισορροπία της ομαδοποίησης. Η περίπτωση $k = 3$ φαίνεται να προσφέρει την πιο συνεκτική κατανομή.

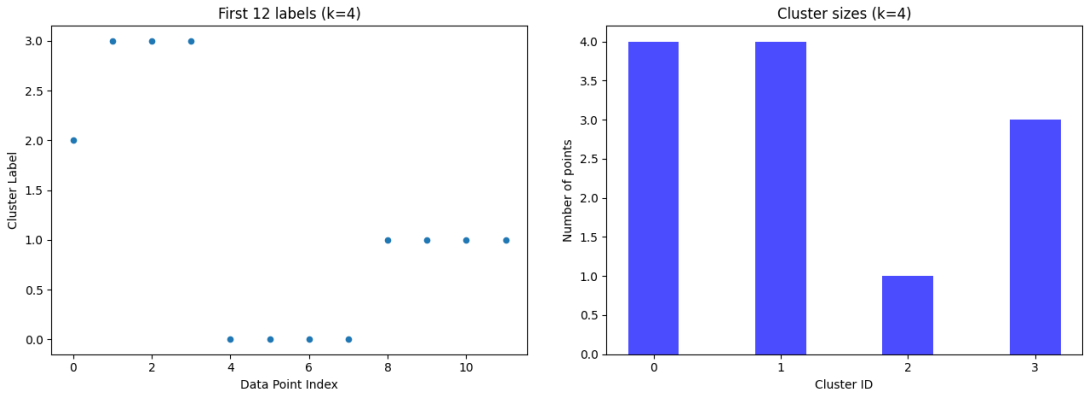
Παρακάτω παρουσιάζονται ενδεικτικά γραφήματα των πρώτων ετικετών καθώς και τα μεγέθη των clusters:



Σχήμα 2.1: Ετικέτες και μεγέθη clusters για $\mathbf{k = 2}$



Σχήμα 2.2: Ετικέτες και μεγέθη clusters για $\mathbf{k = 3}$



Σχήμα 2.3: Ετικέτες και μεγέθη clusters για $\mathbf{k = 4}$

2.3.2 Περιγραφή και λειτουργικότητα demo2.py

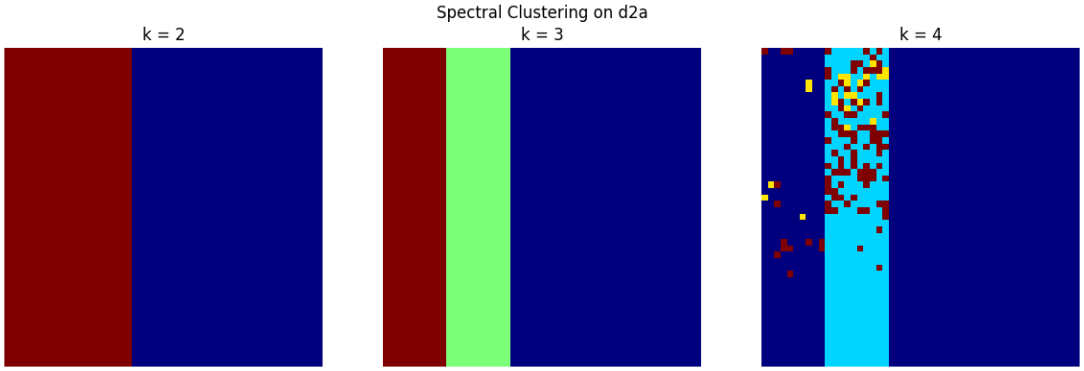
Το αρχείο `demo2.py` αξιοποιεί τη συνάρτηση `spectral_clustering()` σε συνδυασμό με τη `image_to_graph()` για την τμηματοποίηση πραγματικών εικόνων. Συγκεκριμένα:

- Φορτώνονται από το `dip_hw_3.mat` οι δύο έγχρωμες εικόνες `d2a` και `d2b` .
- Κάθε εικόνα μετατρέπεται σε γράφο μέσω της `image_to_graph()` (παράγοντας τον affinity matrix).
- Εκτελείται φασματική ομαδοποίηση για τρεις διαφορετικές τιμές $k = 2, 3, 4$.
- Οι τελικές ετικέτες cluster αναδιαμορφώνονται σε εικόνες, και απεικονίζονται με χρωματική κωδικοποίηση (χάρτης χρωμάτων jet).

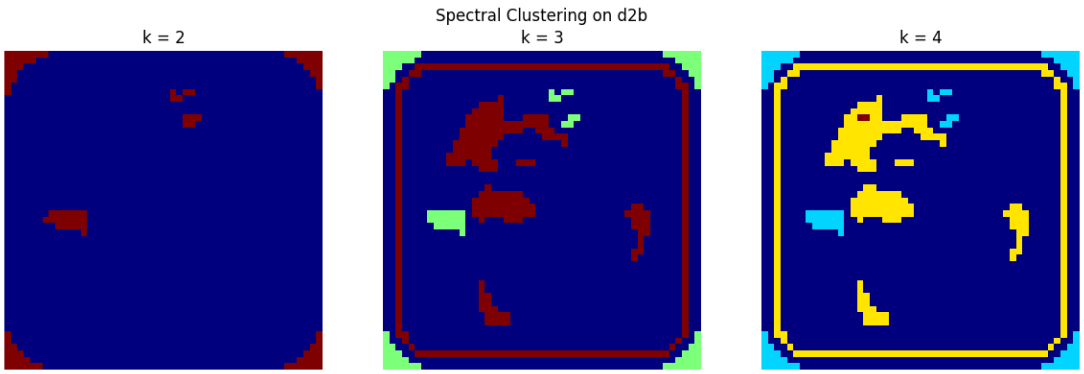
Για κάθε εικόνα, αποθηκεύεται ένα αρχείο μορφής PNG που περιέχει τις τρεις τμηματοποιήσεις με βάση τις τιμές του k. Οι εικόνες αποθηκεύονται με ονόματα `d2a_segmentation.png` και `d2b_segmentation.png` .

Παράθεση Αποτελεσμάτων

Οι τελικές εικόνες που προέκυψαν από τη διαδικασία spectral clustering παρουσιάζονται παρακάτω. Κάθε εικόνα περιλαμβάνει τις τρεις τμηματοποιήσεις για $k = 2, 3, 4$:



Σχήμα 2.4: Τμηματοποίηση εικόνας d2a για $k = 2, 3, 4$



Σχήμα 2.5: Τμηματοποίηση εικόνας d2b για $k = 2, 3, 4$

Η επιλογή διαφορετικών τιμών για k οδηγεί σε εμφανώς διαφορετική κατανομή των περιοχών της εικόνας. Για παράδειγμα, μικρές τιμές όπως $k = 2$ δίνουν γενικές κατηγορίες, ενώ μεγαλύτερες τιμές $k = 4$ οδηγούν σε πιο λεπτομερή διαχωρισμό, αλλά ενδέχεται να προκύψουν υπερ-τμηματοποιήσεις.

Η παράμετρος `random_state = 1` εξασφαλίζει ότι η τυχαία αρχικοποίηση του k-means δίνει αναπαραγώγιμα αποτελέσματα ανά εκτέλεση.

Παρατηρήσεις και Συμπεράσματα

- *Εικόνα* ***d2a****:* Η φασματική ομαδοποίηση αποδίδει άριστα για $k = 2$ και $k = 3$, καταφέρνοντας να εντοπίσει τις μεγάλες χρωματικές περιοχές. Για $k = 4$, προκύπτουν ανεπιθύμητα μικρά θορυβώδη clusters, φανερώνοντας ότι η τιμή αυτή μπορεί να είναι υπερβολική για τη συγκεκριμένη εικόνα.

- *Εικόνα* ***d2b****:* Πρόκειται για πολύπλοκη εικόνα, με υψηλή παραμόρφωση και πολλά αντικείμενα. Παρατηρείται:

  - Για $k = 2$: περιορισμένη τμηματοποίηση.
  - Για $k = 3$: σημαντική βελτίωση με ανίχνευση περιγραμμάτων.
  - Για $k = 4$: εμφανίζονται επιμέρους λεπτομέρειες, ωστόσο κάποια clusters είναι υπερβολικά τοπικά.

- Η μέθοδος `spectral_clustering` είναι ιδιαίτερα αποτελεσματική για εικόνες με καθαρή δομή (π.χ. `d2a`). Αντίθετα, σε περιπτώσεις σύνθετων και θορυβωδών εικόνων (όπως η `d2b`), η επιλογή της παραμέτρου k παίζει κρίσιμο ρόλο στην ποιότητα του αποτελέσματος.

- Η αλλαγή πίνακα γειτνίασης (affinity matrix) θα έπαιζε καθοριστικό ρόλο προκειμένου να μπορούσαμε να διακρίνουμε αισθητές βελτιώσεις.

Οι αντίστοιχοι κώδικες Python βρίσκονται στα αρχεία `demo1.py` και `demo2.py` .

Κεφάλαιο 3

Μέθοδοι Normalized-cuts

Η μέθοδος Normalized-cuts (n-cuts) αποτελεί μια προχωρημένη τεχνική τμηματοποίησης εικόνων βασισμένη στη θεωρία γράφων. Σε αντίθεση με απλούστερες μεθόδους όπως το spectral clustering, λαμβάνει υπόψη όχι μόνο τη συνοχή εντός των clusters αλλά και τη σχετική ανεξαρτησία μεταξύ τους. Στο κεφάλαιο αυτό θα παρουσιαστούν τόσο η μη-αναδρομική όσο και η αναδρομική υλοποίηση της μεθόδου, συνοδευόμενες από αριθμητικά παραδείγματα και πειραματικά αποτελέσματα.

3.1 Μέθοδος n-cuts

Οπως προαναφέρθηκε η μέθοδος n-cuts αποτελεί επέκταση του spectral clustering και επιδιώκει την εύρεση μιας βέλτιστης διχοτόμησης ενός γράφου ελαχιστοποιώντας μια σχετική μετρική Ncut. Η παρακάτω ενότητα περιγράφει αναλυτικά τόσο την υπολογιστική διαδικασία εξαγωγής των ομάδων (μέσω του ιδιοπροβλήματος), όσο και τον ακριβή υπολογισμό της τιμής της μετρικής Ncut για δεδομένη διχοτόμηση.

3.1.1 Υλοποίηση ομαδοποίησης με τη `n_cuts()`

Η συνάρτηση `n_cuts()` αποτελεί την υλοποίηση της μη αναδρομικής εκδοχής της μεθόδου normalized cuts, βασισμένης στη φασματική ανάλυση του Λαπλασιανού πίνακα του γράφου. Πιο συγκεκριμένα, δέχεται ως είσοδο έναν πίνακα γειτνίασης (affinity matrix) και έναν αριθμό ομάδων k και επιστρέφει διανύσματα ετικετών για τα n σημεία.

Η μέθοδος πραγματοποιεί τα εξής στάδια:

1. Αρχικά μετατρέπει τον πίνακα `affinity_mat` σε αραιό πίνακα (sparse format), και μηδενίζει τη διαγώνιο ώστε να εξαλειφθούν οι αυτοσυνδέσεις (self-loops).

2. Υπολογίζει το διαγώνιο πίνακα βαθμών D, όπου κάθε τιμή $D_{ii} = \sum_j W_{ij}$, και κατασκευάζει τον μη-κανονικοποιημένο Λαπλασιανό $L = D - W$.

3. Λύνει το γενικευμένο ιδιοπρόβλημα $L\mathbf{v} = \lambda D\mathbf{v}$ για τις k ιδιοτιμές με τη μικρότερη απόλυτη τιμή, μέσω της συνάρτησης `eigs()` του πακέτου `scipy`.

4. Χρησιμοποιεί τα k ιδιοδιανύσματα που προκύπτουν (συναποτελούν τον πίνακα $U \in \mathbb{R}^{n\times k}$) ως νέα χαρακτηριστικά σημεία, και εκτελεί k-means clustering σε αυτά, ώστε να παραχθούν οι τελικές ομάδες.

Παρατήρηση: Στην προεπιλεγμένη του λειτουργία, ο υπολογισμός των ιδιοδιανυσμάτων μέσω της `eigs()` εμπεριέχει εσωτερική στοχαστικότητα (random initialization μέσω Lanczos iteration). Για την εξασφάλιση αναπαραγώγιμων αποτελεσμάτων, μπορεί να δοθεί συγκεκριμένο αρχικό διάνυσμα $v_0 \neq 0$, όπως στο παρακάτω (σχολιασμένο) απόσπασμα του κώδικα:

```
# --> For deterministic results, use a fixed nonzero vector as v0
n = L.shape[0]
v0 = np.ones(n)
vals, vecs = eigs(L, M=D, k=k, which='SM', v0=v0)
```

Η πρακτική αυτή θα μπορούσε να είναι ιδιαίτερα σημαντική για συνεπή συγκριτική αξιολόγηση των αποτελεσμάτων σε πειραματικές μελέτες ωστόσο δεν χρησιμοποιήθηκε για την εξαγωγή αποτελεσμάτων στην γενικότερη υλοποίηση της εργασίας.
Ακολουθεί ψευδοκώδικας που αναπαριστά τον συλλογισμό μου στην δόμηση του κώδικα:

Ψευδοκώδικας `n_cuts()`

```
1. W ← csr_matrix(affinity_mat); set W[i,i] ← 0
2. D ← diagonal matrix with row sums of W
3. L ← D - W
4. [vals, vecs] ← eigs(L, M=D, k=k)
5. U ← real(vecs)
6. Cluster rows of U with KMeans(n_clusters=k)
7. Return cluster labels
```

3.1.2 Υπολογισμός της μετρικής `Ncut(A,B)`

Η πραγματική τιμή της μετρικής `Ncut(A,B)` για δεδομένο διαχωρισμό σε δύο ομάδες υπολογίζεται με τη συνάρτηση `calculate_n_cut_value()`. Η μετρική δίνεται από:

$$\mathrm{Ncut}(A,B) = 2 - \left(\frac{\mathrm{assoc}(A,A)}{\mathrm{assoc}(A,V)} + \frac{\mathrm{assoc}(B,B)}{\mathrm{assoc}(B,V)} \right)$$

- $\mathrm{assoc}(A,V) = \sum_{i \in A, j \in V} W(i,j)$
- $\mathrm{assoc}(A,A) = \sum_{i,j \in A} W(i,j)$
- Ομοίως για B

Αυτός ο υπολογισμός είναι χρήσιμος ειδικά σε ιεραρχικές ή αναδρομικές στρατηγικές (π.χ. `n_cuts_recursive`) που θα αναδειχθεί ως υλοποίηση σε επόμενο section, ώστε να ελέγχεται αν μια διχοτόμηση είναι αρκετά καλή με βάση κάποιο κατώφλι T_2.

Ακολουθεί ψευδοκώδικας που αναπαριστά τον συλλογισμό μου στην δόμηση του κώδικα για την συνάρτηση `calculate_n_cut_value()`:

Ψευδοκώδικας `calculate_n_cut_value()`

```
1. A ← indices of cluster 0
   B ← indices of cluster 1

2. assoc_A_V ← sum of W[i,:] for i in A
   assoc_B_V ← sum of W[i,:] for i in B

3. assoc_A_A ← sum of W[i,j] for i,j in A
   assoc_B_B ← sum of W[i,j] for i,j in B

4. nassoc ← (assoc_A_A / assoc_A_V) + (assoc_B_B / assoc_B_V)
5. Return ncut_value ← 2 - nassoc
```

3.2 Μέθοδος n-cuts Recursive

Η μέθοδος recursive n-cuts αποτελεί μια αναδρομική παραλλαγή της φασματικής τεχνικής normalized cuts, και στοχεύει στην επαναληπτική διάσπαση του γράφου σε υποσύνολα, με βάση ένα φασματικό κριτήριο ποιότητας (το normalized cut value).

Η βασική ιδέα είναι να εφαρμόζεται το `n_cuts()` για $k = 2$ και, ανάλογα με την ποιότητα του διαχωρισμού, να αποφασίζεται εάν η ομάδα πρέπει να διασπαστεί περαιτέρω ή να παραμείνει ως έχει.

Η συνάρτηση `n_cuts_recursive()` βασίζεται στις συναρτήσεις `calculate_n_cut_value()` και `n_cuts()` που παρουσιάστηκαν στην προηγούμενη ενότητα. Η λογική της αναλυτικά:

- Κρατά έναν πίνακα υποομάδων (`queue`) που πρόκειται να εξεταστούν.
- Για κάθε ομάδα:
  - Αν το μέγεθός της είναι μικρότερο από ένα κατώφλι T_1, δεν διασπάται περαιτέρω.
  - Αλλιώς, εφαρμόζεται η `n_cuts()` για $k = 2$ ώστε να παραχθεί ένας δυαδικός διαχωρισμός.
  - Υπολογίζεται η τιμή `Ncut` με χρήση της `calculate_n_cut_value()`.
  - Αν η τιμή αυτή ξεπερνά το κατώφλι T_2, η ομάδα δεν διασπάται περαιτέρω.
  - Διαφορετικά, διαχωρίζεται στις δύο υποομάδες και αυτές επιστρέφουν στην `queue`.
- Η διαδικασία συνεχίζεται έως ότου καμία υποομάδα να μην μπορεί ή να μην πρέπει να διασπαστεί περαιτέρω.

Ο τελικός πίνακας ετικετών (`cluster_idx`) επιστρέφει τις ομαδοποιήσεις όλων των κόμβων, όπως προέκυψαν από τη διαδικασία αναδρομικής διάσπασης.

Σχόλια και Πλεονεκτήματα:

- Η αναδρομική έκδοση επιτρέπει δυναμικό προσδιορισμό του αριθμού των τελικών ομάδων, ανάλογα με την τοπική δομή του γράφου.
- Ο συνδυασμός των κατωφλίων T_1 (ελάχιστο πλήθος κόμβων) και T_2 (ποιότητα διαχωρισμού) προσφέρει έλεγχο στο βάθος και στην ποιότητα της τμηματοποίησης.
- Ο αλγόριθμος είναι ιδιαίτερα χρήσιμος για προβλήματα image segmentation, όπου δεν είναι εκ των προτέρων γνωστός ο αριθμός των αντικειμένων.

Σημείωση: Η απόδοση της συνάρτησης εξαρτάται άμεσα από την ποιότητα των affinity matrices και την ορθή ρύθμιση των παραμέτρων T_1 και T_2. Μικρό T_2 οδηγεί σε υπερδιάσπαση, ενώ μεγάλο μπορεί να αποτρέψει χρήσιμους διαχωρισμούς.

Παρακάτω παρατίθεται και ο αντίστοιχος Ψευδοκώδικας για την συνάρτηση `n_cuts_recursive()`:

Ψευδοκώδικας `n_cuts_recursive()`

```
1. Initialize queue ← [all node indices]
2. final_clusters ← []

3. While queue not empty:
    a. idx ← queue.pop(0)
    b. If len(idx) ≤ T1:
           final_clusters.append(idx)
           continue
    c. W_sub ← affinity_mat[idx, idx]
    d. labels ← n_cuts(W_sub, k=2)
    e. ncut_val ← calculate_n_cut_value(W_sub, labels)
    f. If ncut_val ≥ T2:
           final_clusters.append(idx)
       Else:
           part0 ← idx[labels == 0]
           part1 ← idx[labels == 1]
           queue.append(part0)
           queue.append(part1)

4. Assign final label to each group in final_clusters
5. Return cluster_idx
```

3.3 Ανάλυση Αποτελεσμάτων και Οπτικοποίησης

Σε αυτό το section παρουσιάζονται και σχολιάζονται τα αποτελέσματα που προέκυψαν από την εφαρμογή των συναρτήσεων `n_cuts()`, `calculate_n_cut_value()` και `n_cuts_recursive()` στα σκριπτς `demo3a.py`, `demo3b.py` και `demo3c.py`.

Η κάθε υποενότητα αναλύει τη λειτουργία της αντίστοιχης ρουτίνας, καθώς και τα αποτελέσματα που παράγονται για τις εικόνες `d2a` και `d2b` από το dataset. Η σύγκριση με τη μέθοδο spectral clustering δίνει καλύτερη εικόνα της συμπεριφοράς των υλοποιήσεων n-cuts.

3.3.1 Περιγραφή και λειτουργικότητα demo3a.py

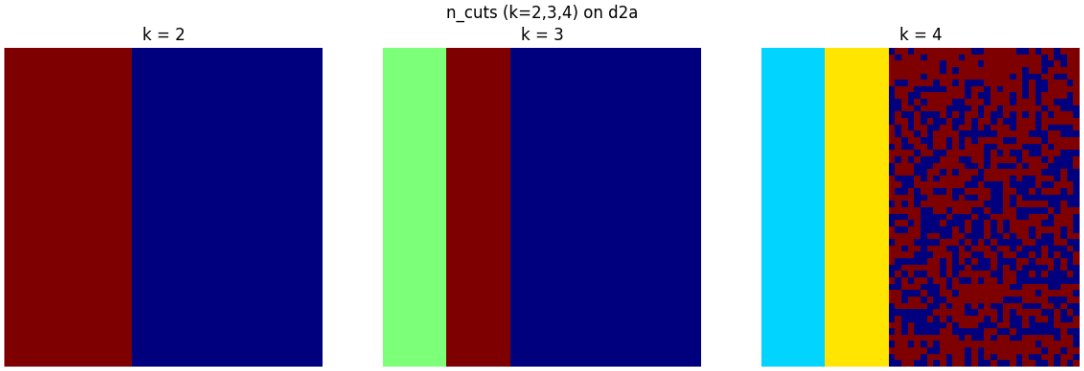
Το αρχείο `demo3a.py` αξιοποιεί τη μη-αναδρομική εκδοχή της μεθόδου n-cutsγια να ομαδοποιήσει τις εικόνες `d2a` και `d2b`, οι οποίες φορτώνονται από το αρχείο `dip_hw_3.mat`. Για κάθε εικόνα, ακολουθούνται τα εξής βήματα:

- Μετατροπή της εικόνας σε γράφο μέσω της συνάρτησης `image_to_graph()` , ώστε να προκύψει ο affinity matrix W.
- Εκτέλεση της συνάρτησης `n_cuts()` για τρεις διαφορετικές τιμές αριθμού ομάδων: $k = 2, 3, 4$.
- Ανακατασκευή των ετικετών σε μορφή εικόνας (πίνακας $M \times N$) και απεικόνιση των αποτελεσμάτων.
- Εκτύπωση του πλήθους των κόμβων σε κάθε cluster για αξιολόγηση της ισορροπίας των αποτελεσμάτων.

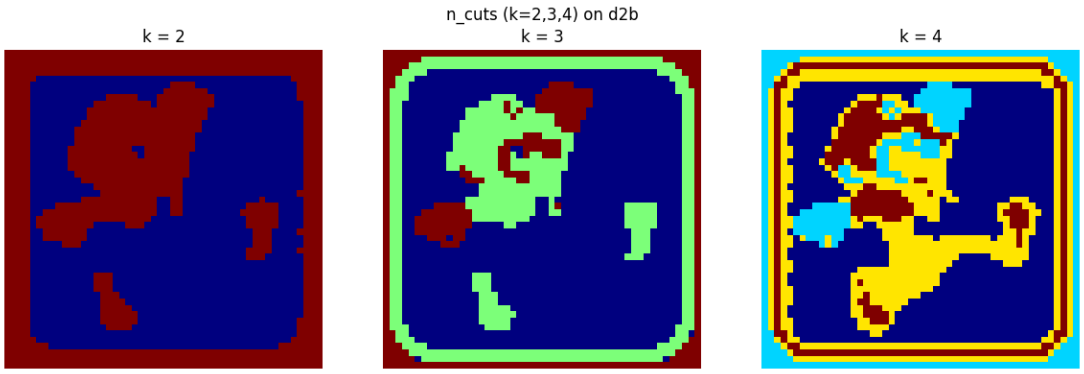
Για κάθε μία από τις δύο εικόνες, παράγονται τρία διαγράμματα που αντιστοιχούν στις τιμές του k. Τα αποτελέσματα αποθηκεύονται σε εικόνες με ονόματα τύπου `d2a_n_cuts.png` και `d2b_n_cuts.png`.

Παράθεση Αποτελεσμάτων

Παρακάτω παρατίθενται και τα αποτελέσματα του `demo3a.py`.



Σχήμα 3.1: Αποτελέσματα n-cuts για `d2a` με $k = 2, 3, 4$



Σχήμα 3.2: Αποτελέσματα n-cuts για `d2b` με $k = 2, 3, 4$

Τα αντίστοιχα μεγέθη των clusters που προέκυψαν από την εκτέλεση είναι:

- Εικόνα `d2a`:
  - $k = 2$: `[1500, 1000]`
  - $k = 3$: `[1500, 500, 500]`
  - $k = 4$: `[624, 500, 500, 876]`
- Εικόνα `d2b`:
  - $k = 2$: `[1360, 1140]`
  - $k = 3$: `[1456, 659, 385]`
  - $k = 4$: `[1028, 381, 752, 339]`

Σχολιασμός Αποτελεσμάτων

Οι παρατηρήσεις που προκύπτουν είναι οι εξής:
Για την εικόνα d2a:

- Με $k = 2$: Δύο καθαρές, μεγάλες περιοχές (1500 και 1000 pixels). Η τμηματοποίηση είναι ισχυρή και αναπαριστά το απλό μοτίβο της εικόνας.
- Με $k = 3$: Το μεγάλο cluster παραμένει σταθερό (1500 pixels), ενώ εμφανίζονται δύο μικρότερα, συμμετρικά (500+500).
- Με $k = 4$: Οι δύο περιοχές των 500 pixels παραμένουν και εμφανίζονται δύο νέες (624 και 876), δείχνοντας ότι η μέθοδος διαχωρίζει επιπλέον λεπτομέρειες — με ενδεχόμενη υπερδιάσπαση.

Για την εικόνα d2b:

- Με $k = 2$: Σχετικά ισορροπημένος διαχωρισμός (1360–1140). Η τμηματοποίηση είναι απλοϊκή για την πολυπλοκότητα της εικόνας.
- Με $k = 3$: Μεγαλύτερη διαφοροποίηση, με το κυρίαρχο cluster να έχει 1456 pixels. Ενδείξεις για καλύτερη αναγνώριση διαφορετικών περιοχών του αντικειμένου.

- Με $k = 4$: Σημαντικά περισσότερη λεπτομέρεια. Το μέγεθος των clusters είναι πιο ανισοβαρές, αλλά αποτυπώνουν διάφορες λειτουργικές περιοχές του αντικειμένου.

Συμπεράσματα:

- Οι αριθμητικές τιμές των clusters επιβεβαιώνουν τις παρατηρήσεις από τις οπτικοποιήσεις.
- Η εικόνα `d2a` δείχνει σταθερότητα στις ομαδοποιήσεις καθώς αυξάνει το k, ενώ η `d2b` επωφελείται από μεγαλύτερο k για καλύτερη αποτύπωση της πολυπλοκότητας.
- Τέλος, για τη σύγκριση με την υλοποίηση της μεθόδου Spectral Clustering, παρατηρούμε ότι η μέθοδος n-cutsπαρέχει σαφώς πιο καθαρά και λειτουργικά αποτελέσματα στην εικόνα δ2β, ειδικά για μεγαλύτερες τιμές του k. Οι δομές του αντικειμένου εντοπίζονται με μεγαλύτερη ακρίβεια και οι τμηματοποιήσεις διαχωρίζουν με σαφήνεια τα λειτουργικά μέρη του σχήματος. Αντίθετα, στην περίπτωση της εικόνας `d2a`, για μικρές τιμές του k (π.χ. $k = 2, 3$), και οι δύο μέθοδοι εμφανίζουν σχεδόν ταυτόσημα και οπτικά καθαρά αποτελέσματα. Ωστόσο, για $k = 4$, η μέθοδος Spectral Clustering οδηγεί σε περισσότερη τυχαία κατανομή και θόρυβο, σε αντίθεση με την n-cutsπου, παρότι παρουσιάζει υπερδιάσπαση, διατηρεί πιο συνεπή και συστηματική τμηματοποίηση. Συνολικά, η n-cuts εμφανίζεται πιο σταθερή και πιο αποτελεσματική, ιδίως σε περιπτώσεις σύνθετης δομής όπως στην `d2b`.

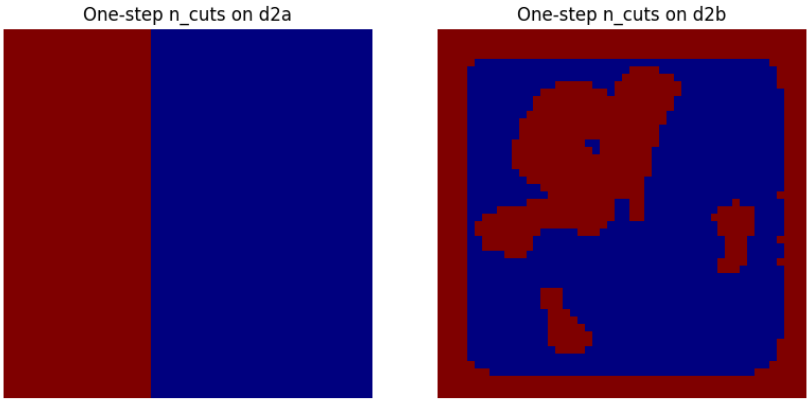
3.3.2 Περιγραφή και λειτουργικότητα demo3b.py

Το αρχείο `demo3b.py` εξετάζει τη λειτουργία της μη-αναδρομικής μεθόδου n-cuts για $k = 2$, εφαρμόζοντας μόνο ένα βήμα διάσπασης στον αρχικό γράφο της εικόνας. Δηλαδή, η εικόνα διαχωρίζεται σε δύο υποσύνολα χωρίς περαιτέρω αναδρομή.
Η διαδικασία έχει ως εξής:

- Για κάθε εικόνα `d2a`, `d2b`, δημιουργείται ο affinity matrix μέσω της `image_to_graph()`.
- Εκτελείται η `n_cuts()` για $k = 2$.
- Υπολογίζεται η τιμή του `normalized cut` μέσω της `calculate_n_cut_value()`.
- Τα αποτελέσματα αποθηκεύονται ως εικόνες και αναδεικνύονται παρακάτω.

Παράθεση Αποτελεσμάτων

Τα παρακάτω διαγράμματα παρουσιάζουν την οπτικοποίηση των διαχωρισμών:



Σχήμα 3.3: Αποτελέσματα one-step n-cuts για `d2a` και `d2b` με $k = 2$

Οι αριθμοί και οι τιμές `Ncut` για κάθε εικόνα είναι:

- *Εικόνα d2a*:
  - Μέγεθος clusters: `[1500, 1000]`
  - Τιμή `Ncut`: `0.5092`
- *Εικόνα d2b*:
  - Μέγεθος clusters: `[1360, 1140]`
  - Τιμή `Ncut`: `0.7853`

Σχολιασμός και Σύγκριση με Spectral Clustering για $k = 2$

Για την εικόνα d2a:

- Το αποτέλεσμα είναι σχεδόν ταυτόσημο με εκείνο της spectral clustering για $k = 2$, αφού η εικόνα έχει μια απλή δομή και ο φυσικός διαχωρισμός εντοπίζεται εύκολα.
- Η τιμή `Ncut` είναι σχετικά χαμηλή (≈ 0.5092), γεγονός που υποδεικνύει ικανοποιητική διάσπαση.

Για την εικόνα d2b:

- Η τιμή `Ncut` είναι υψηλότερη (≈ 0.7853), υποδεικνύοντας ότι η απλή διάσπαση σε 2 ομάδες δεν περιγράφει επαρκώς την πολυπλοκότητα της εικόνας.
- Συγκριτικά με τη Spectral Clustering για $k = 2$, η ομαδοποίηση είναι πιο απλοϊκή και δεν εντοπίζει μικρότερες περιοχές ή λεπτομέρειες.
- Η μέθοδος τείνει να διαχωρίζει μόνο τις μεγαλύτερες, μακροσκοπικές δομές.

Γενικό Συμπέρασμα: Η one-step n-cuts λειτουργεί ικανοποιητικά όταν υπάρχει καθαρός, φυσικός διαχωρισμός (π.χ. `d2a`), αλλά υπολείπεται σε πιο σύνθετες περιπτώσεις (`d2b`), σε σχέση με τη spectral clustering που έχει μεγαλύτερη ευελιξία και καταγράφει λεπτομερέστερες δομές.

3.3.3 Περιγραφή και λειτουργικότητα demo3c.py

Το αρχείο `demo3c.py` εκτελεί τη πλήρη (αναδρομική) υλοποίηση της μεθόδου n-cuts για τις εικόνες `d2a` και `d2b`, με στόχο την αυτόματη διάσπαση των δεδομένων σε υποσύνολα, χωρίς τον εκ των προτέρων καθορισμό του πλήθους των ομάδων.

Οι παράμετροι που καθορίζουν τη διαδικασία είναι:

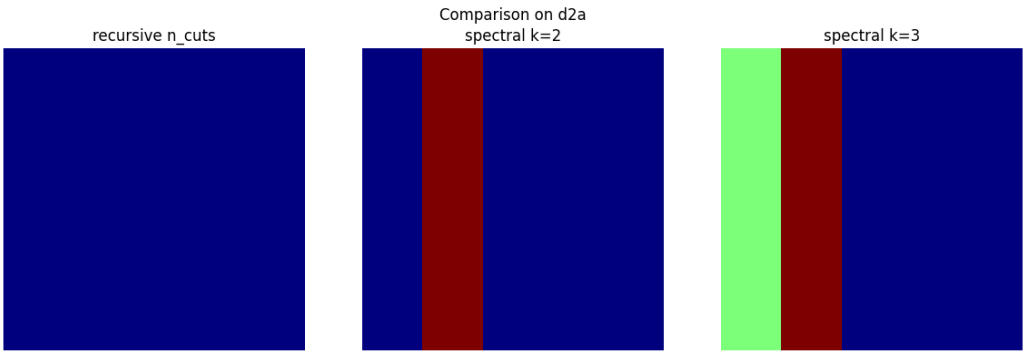
- $T_1 = 5$: Ελάχιστο πλήθος κόμβων ώστε μια ομάδα να μπορεί να διασπαστεί.
- $T_2 = 0.20$: Μέγιστη αποδεκτή τιμή του `Ncut` ώστε να συνεχιστεί η διάσπαση.

Τα βήματα που εκτελούνται για κάθε εικόνα είναι:

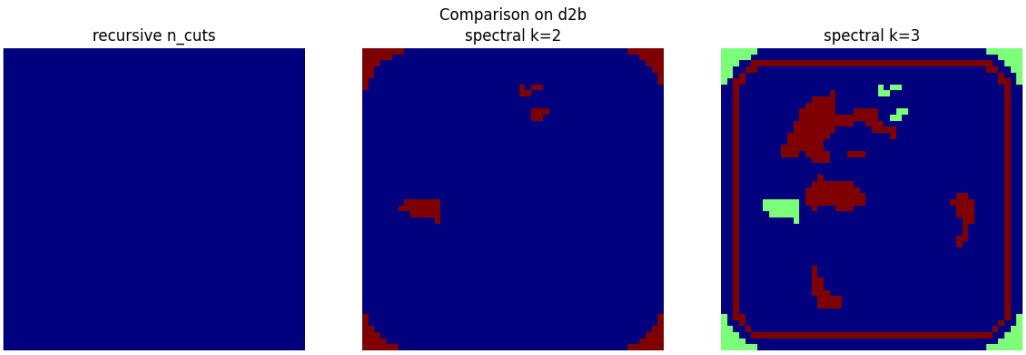
- Κατασκευή affinity matrix μέσω της `image_to_graph()`.
- Εκτέλεση της `n_cuts_recursive()` για την αναδρομική τμηματοποίηση.
- Εκτέλεση της `spectral_clustering()` για $k = 2$ και $k = 3$, ως σύγκριση.
- Οπτικοποίηση των τριών αποτελεσμάτων σε κοινό διάγραμμα.
- Εκτύπωση μεγεθών ομάδων για την αξιολόγηση της ισορροπίας και της λεπτομέρειας.

Τα αποτελέσματα αποθηκεύονται σε εικόνες με όνομα `d2a_comparison.png`, `d2b_comparison.png`, κ.λπ.

Παράθεση Αποτελεσμάτων



Σχήμα 3.4: Σύγκριση Recursive n-cuts και Spectral Clustering για την εικόνα `d2a`



Σχήμα 3.5: Σύγκριση Recursive n-cuts και Spectral Clustering για την εικόνα `d2b`

Τα πλήθη κόμβων σε κάθε cluster που προέκυψαν είναι τα εξής:

- `d2a`, *recursive*: `[2500]` (δεν έγινε περαιτέρω διάσπαση λόγω υψηλής τιμής `Ncut`)
- `d2a`, *spectral k=2*: `[2000, 500]`
- `d2a`, *spectral k=3*: `[1500, 500, 500]`
- `d2b`, *recursive*: `[2500]` (δεν έγινε περαιτέρω διάσπαση)
- `d2b`, *spectral k=2*: `[2399, 101]`
- `d2b`, *spectral k=3*: `[2062, 97, 341]`

Σχολιασμός και Σύγκριση

Για την εικόνα d2a:

- Η `recursive n-cuts` επέστρεψε ένα μόνο cluster (ολόκληρη η εικόνα ως μία ομάδα), γεγονός που φαίνεται αρχικά αντιφατικό με τα αναμενόμενα αποτελέσματα.
- Αυτό όμως εξηγείται πλήρως αν αναλυθεί η ροή της συνάρτησης `n_cuts_recursive()`:
  - Ο αρχικός διαχωρισμός του γράφου γίνεται μέσω `n_cuts()`με $k = 2$.
  - Στη συνέχεια υπολογίζεται η τιμή `Ncut` του διαχωρισμού.
  - Αν η τιμή `Ncut` είναι μεγαλύτερη ή ίση από το κατώφλι `T2`, η ομάδα δεν διασπάται περαιτέρω και καταχωρείται ως έχει.
- Στην περίπτωση του `d2a`, η αρχική τιμή `Ncut` είναι αρκετά υψηλή (πάνω από 0.2), συνεπώς η διάσπαση απορρίφθηκε και το σύνολο των 2500 κόμβων παρέμεινε ως ένα ενιαίο cluster.
- Αντιθέτως, η μέθοδος spectral clustering δεν έχει κάποιο φραγμό ποιότητας, αλλά επιβάλλει αυστηρά την παραγωγή k ομάδων — ανεξάρτητα από το αν αυτές έχουν ουσιαστική σημασία ή όχι.
- Για $k = 3$ και $k = 2$ στην εικόνα `d2a`, οι ομάδες που προκύπτουν φαίνονται εύλογες και αποδίδουν καλύτερη διαχωριστική πληροφορία από την αναδρομική `n-cuts`, για τις συγκεκριμένες παραμέτρους.

Για την εικόνα d2b:

- Αντίστοιχα με το `d2a`, η `recursive n-cuts` δεν προχώρησε σε περαιτέρω διαχωρισμό επειδή η αρχική τιμή `Ncut` υπερέβη το κατώφλι `T2 = 0.2`.
- Επομένως, η εικόνα θεωρήθηκε μη διασπάσιμη και επιστράφηκε ως ένα ενιαίο σύνολο 2500 κόμβων.
- Στην πράξη όμως, η εικόνα `d2b` έχει μεγαλύτερη πολυπλοκότητα και κρύβει περισσότερες τοπικές δομές.
- Η `spectral clustering` με $k = 3$ καταφέρνει να εντοπίσει κάποιες από αυτές, αποδίδοντας διαφορετικές περιοχές, παρότι τα όρια μεταξύ τους δεν είναι τόσο καθαρά.

- Αυτό καταδεικνύει πως η `n_cuts_recursive` είναι πολύ ευαίσθητη στην επιλογή του `T2`. Αν είναι πολύ αυστηρό, τότε δεν γίνονται καν βασικές διασπάσεις.

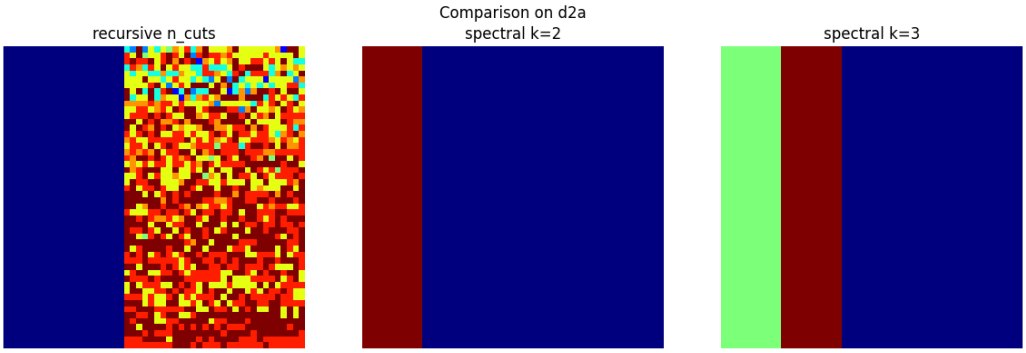
Συμπερασματικά: Η `n_cuts_recursive` προσφέρει προσαρμοστικότητα και αποφυγή υπερδιάσπασης, αλλά εξαρτάται κρίσιμα από τη ρύθμιση του κατωφλιού `T2`, που δρα ως φίλτρο ποιότητας. Αντίθετα, η `spectral clustering` επιβάλλει k ομάδες και αποδίδει πάντα αποτέλεσμα, ανεξαρτήτως ποιότητας. Στο συγκεκριμένο πείραμα, η `T2 = 0.2` αποδείχθηκε υπερβολικά αυστηρή και δεν επέτρεψε την αποκάλυψη της υποκείμενης δομής, σε αντίθεση με τη `spectral clustering` που εμφάνισε περισσότερα διαχωριστικά χαρακτηριστικά.

Πείραμα με παραμετροποίηση $T_1 = 1000$, $T_2 = \infty$

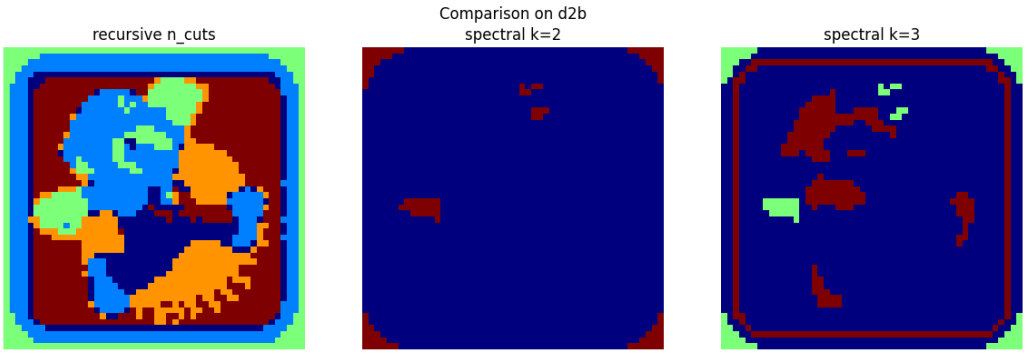
Στο πείραμα αυτό, επιλέχθηκε υψηλή τιμή για το κατώφλι ελάχιστου μεγέθους ομάδας ($T_1 = 1000$) και απενεργοποιήθηκε ο έλεγχος της τιμής Ncut ($T_2 = \infty$). Με αυτό τον τρόπο, η συνάρτηση `n_cuts_recursive` αναγκάζεται να διαιρεί οποιαδήποτε ομάδα μεγαλύτερη των 1000 κόμβων, ανεξαρτήτως της ποιότητας του διαχωρισμού, με σκοπό να εξεταστεί η συμπεριφορά της μεθόδου υπό καθαρά δομικά κριτήρια μεγέθους.

Αποτελέσματα Ομαδοποίησης:

- `d2a`, *recursive*: `[1000, 6, 11, 47, 8, 340, 87, 461, 540]` (σύνολο 9 ομάδες)
- `d2a`, *spectral k=2*: `[2000, 500]`
- `d2a`, *spectral k=3*: `[1500, 500, 500]`
- `d2b`, *recursive*: `[359, 758, 382, 332, 669]` (σύνολο 5 ομάδες)
- `d2b`, *spectral k=2*: `[2399, 101]`
- `d2b`, *spectral k=3*: `[2062, 97, 341]`



Σχήμα 3.6: Σύγκριση Recursive n-cuts με Spectral Clustering για την εικόνα `d2a` με $T_1 = 1000$, $T_2 = \infty$



Σχήμα 3.7: Σύγκριση Recursive n-cuts με Spectral Clustering για την εικόνα `d2b` με $T_1 = 1000$, $T_2 = \infty$

Σχολιασμός:

Για την εικόνα d2a:

- Η μέθοδος `recursive n-cuts` οδηγεί σε υπερδιάσπαση, όπως είναι αναμενόμενο λόγω του χαμηλού T_1 και της απουσίας ελέγχου ποιότητας διαχωρισμού (αφού $T_2 = \infty$).
- Παρατηρείται έντονος θόρυβος στις μικρές περιοχές — δημιουργούνται πολύ μικρές ομάδες (π.χ. 6, 8, 11 κόμβοι) χωρίς ουσιαστικό νόημα.
- Η μέθοδος spectral clustering , ιδιαίτερα για $k = 3$, καταφέρνει καλύτερα να διατηρήσει τη λογική χωρική κατανομή των περιοχών.

Για την εικόνα d2b:

- Η recursive n-cuts παρουσιάζει μια πολύ πιο πλούσια και λεπτομερή τμηματοποίηση, αποκαλύπτοντας λειτουργικά ξεχωριστές περιοχές (π.χ. πρόσωπο, σώμα, καπέλο).
- Η spectral clustering περιορίζεται σε μακροσκοπικά σχήματα (ιδιαίτερα για $k = 2$) και αποτυγχάνει να εντοπίσει μικρότερες δομές.

Συμπεράσματα:

- Η απενεργοποίηση του ελέγχου ποιότητας ($T_2 = \infty$) οδηγεί σε υπερκατάτμηση της εικόνας, ειδικά όταν επιβάλλεται μεγάλος αριθμός διαχωρισμών λόγω μικρού T_1.
- Αν και αυτό είναι επιθυμητό σε σύνθετες εικόνες όπως η `d2b`, καθώς αποδίδει περισσότερη λεπτομέρεια, είναι επιβλαβές για απλές περιπτώσεις όπως η `d2a`.
- Το πείραμα αναδεικνύει τη σημασία της κατάλληλης ρύθμισης των παραμέτρων T1 και T2 για τον έλεγχο της λεπτομέρειας και της συνοχής της τμηματοποίησης.

Οι αντίστοιχοι κώδικες για αυτό το Chapter βρίσκονται στα αρχεία `n_cuts.py`, `demo3a.py`, `demo3b.py` και `demo3c.py` αντίστοιχα.

Βιβλιογραφία

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.eigs.html

[2] https://elearning.auth.gr/course/view.php?id=16312