Ψηφιακή Επεξεργασία Εικόνας
Εργασία 2: Ανίχνευση ακμών και κύκλων

Α. Ντελόπουλος

Άνοιξη 2025

1 Εισαγωγικά

1.1 FIR filters

Τα Γραμμικά χωρικά αναλλοίωτα συστήματα επιδρούν σε μία εικόνα εισόδου $f(n_1, n_2)$ μέσω της διαδικασίας συνέλιξης

$$g(n_1, n_2) = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} h(k_1, k_2) f(n_1 - k_1, n_2 - k_2) \tag{1}$$

παράγοντας την εικόνα εξόδου $g(n_1, n_2)$. Η εικόνα $h(n_1, n_2)$ είναι η κρουστική απόκριση του συστήματος. Αν ειδικότερα η $h(n_1, n_2)$ έχει πεπερασμένη "περιοχή υποστήριξης", δηλαδή έχει μή μηδενικές τιμές σε μια πεπερασμένη περιοχή W τότε το σύστημα είναι ένα Finite Impulse Resonse (FIR) σύστημα. Σε αυτή την περίπτωση η περιοχή W βρίσκεται μέσα σε ένα ορθογώνιο χωρίο $[N_{1,min}, ..., N_{1,max}] \times [N_{2,min}, ..., N_{2,max}]$ και η εξ. 1 γράφεται

$$\begin{aligned} g(n_1, n_2) &= \sum_{k_1=N_{1,min}}^{N_{1,max}} \sum_{k_2=N_{2,min}}^{N_{1,max}} h(k_1, k_2) f(n_1 - k_1, n_2 - k_2) \\ &= \sum_{k_1=N_{1,min}}^{N_{1,min}+N_1-1} \sum_{k_2=N_{2,min}}^{N_{2,min}+N_2-1} h(k_1, k_2) f(n_1 - k_1, n_2 - k_2) \end{aligned} \tag{2}$$

Οι τιμές της FIR κρουστικής απόκρισης συνήθως περιγράφονται από τη μορφή μίας **συνελικτικής μάσκας**, δηλαδή ενός πίνακα διάστασης $N_2 \times N_1$ με διάταξη τέτοια ώστε το n_1 να αυξάνει από τα αριστερά προς τα δεξιά και το n_2 από τα πάνω προς τα κάτω. Η σύμβαση αυτή είναι συνεπής με χρήση πινάκων για την αποθήκευση εικόνων $f(n_1, n_2)$ όπου επίσης η διάταξη των δειγμάτων στον πίνακα ακολουθεί τον ίδιο κανόνα. Η θέση της αρχής $(0, 0)$ των συντεταγμένων, τόσο για την h όσο και για την f, προσδιορίζεται με ένα σημάδι (π.χ., με ένα τετράγωνο γύρω από το στοιχείο που βρίσκεται στην αρχή).

1.2 Τελεστές ανίχνευσης ακμών

Η ανίχνευση ακμών σε μια εικόνα αποτελεί βασική διαδικασία στην επεξεργασία εικόνας και στην υπολογιστική όραση. Ακολουθεί η περιγραφή δύο εναλλακτικών μεθόδων ανίχνευσης ακμών με τη βοήθεια συνέλιξης FIR.

1.3 Τελεστής Sobel

Ο τελεστής **Sobel** είναι ένας από τους πιο διαδεδομένους τελεστές για την εξαγωγή ακμών, καθώς υπολογίζει προσεγγιστικά τις παραγώγους πρώτης τάξης κατά τις κατευθύνσεις x_1 και x_2. Αυτό επιτυγχάνεται μέσω της συνέλιξης της εικόνας με δύο μάσκες 3×3, μία για κάθε διεύθυνση:

$$G_{x1} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_{x2} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3}$$

Οι μάσκες αυτές εφαρμόζονται στην εικόνα με συνέλιξη, και το αποτέλεσμα είναι δύο εικόνες $g_1(n_1, n_2)$ και $g_2(n_1, n_2)$ που περιέχουν διακριτές προσεγγίσεις των παραγώγων της $f(x_1, x_2)$ ως προς x_1 και x_2. Το συνολικό μέτρο της κλίσης σε κάθε σημείο υπολογίζεται με τη σχέση:

$$g(n_1, n_2) = \sqrt{g_1(n_1, n_2)^2 + g_2(n_1, n_2)^2} \tag{4}$$

Οι περιοχές της εικόνας όπου η τιμή της $g(n_1, n_2)$ είναι μεγάλη, αντιστοιχούν σε σημεία με έντονες μεταβολές φωτεινότητας — δηλαδή ακμές. Ο τελεστής Sobel είναι ιδιαίτερα χρήσιμος επειδή συνδυάζει την παράγωγο με κάποια μορφή εξομάλυνσης, περιορίζοντας τον θόρυβο.

1.4 Τελεστής Laplacian of Gaussian (LoG)

Ο τελεστής **Laplacian of Gaussian (LoG)** χρησιμοποιείται για την ανίχνευση ακμών μέσω του εντοπισμού περιοχών όπου η δεύτερη παράγωγος της έντασης της εικόνας μεταβάλλεται απότομα. Η βασική ιδέα είναι να εφαρμοστεί πρώτα εξομάλυνση της εικόνας με έναν διδιάστατο Γκαουσιανό πυρήνα, ώστε να μειωθεί ο θόρυβος, και στη συνέχεια να υπολογιστεί η Λαπλασιανή της εικόνας, δηλαδή το άθροισμα των δεύτερων μερικών παραγώγων:

$$\nabla^2 f(x_1, x_2) = \frac{\partial^2 f}{\partial x_1^2} + \frac{\partial^2 f}{\partial x_2^2} \tag{5}$$

Η εξομάλυνση και η παραγώγιση συνδυάζονται σε έναν ενιαίο τελεστή, που προκύπτει από την εφαρμογή της Λαπλασιανής στον Γκαουσιανό πυρήνα. Ο τελικός πυρήνας συνελίξεως (LoG kernel) έχει τη μορφή:

$$\mathrm{LoG}(x_1, x_2) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x_1^2 + x_2^2}{2\sigma^2}\right) e^{-\frac{x_1^2+x_2^2}{2\sigma^2}} \tag{6}$$

Η συνέλιξη της εικόνας με τον παραπάνω πυρήνα οδηγεί σε μια νέα εικόνα, στην οποία οι ακμές ανιχνεύονται εντοπίζοντας τα μηδενικά σημεία (zero-crossings) του αποτελέσματος. Αυτά τα σημεία αντιστοιχούν σε απότομες αλλαγές φωτεινότητας και συνεπώς σε ακμές. Ο τελεστής LoG προσφέρει καλή ακρίβεια και ταυτόχρονη μείωση της επίδρασης του θορύβου, καθιστώντας τον κατάλληλο για εφαρμογές όπου απαιτείται αυξημένη ευαισθησία στις λεπτομέρειες.

Η διακριτή προσέγγιση γίνεται με την παραγωγή ενός πίνακα συντελεστών (μάσκας) διαστάσεων $L_1 \times L_2$, όπου οι τιμές του LoG υπολογίζονται σε ένα δειγματοληπτημένο πλέγμα γύρω από το κέντρο (π.χ. $x_1, x_2 \in \{-k, \ldots, 0, \ldots, +k\}$, με $L_1 = L_2 = 2k + 1$). Για παράδειγμα, για $L_1 = L_2 = 5$, η διακριτή μάσκα μπορεί να είναι της μορφής:

$$h = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \tag{7}$$

Η συνέλιξη της εικόνας με τη μάσκα αυτή οδηγεί σε ένα αποτέλεσμα όπου οι ακμές εντοπίζονται με την εύρεση των σημείων μηδενικής διέλευσης (zero-crossings). Ο τελεστής LoG συνδυάζει εξομάλυνση και παράγωγο δεύτερης τάξης, προσφέροντας ισορροπία μεταξύ ακρίβειας και ανθεκτικότητας στον θόρυβο.

1.5 Ανίχνευσης κυκλικών περιγραμμάτων με τη μέθοδο Hough

Η μέθοδος **Hough** ανιχνεύει παραμετρικές καμπύλες σε δυαδικές εικόνες, όπου τα σημεία της καμπύλης έχουν άσπρο χρώμα (τιμή 1) και το background είναι μαύρο (τιμή 0). Η μέθοδος είναι ικανή να ανιχνεύσει τις καμπύλες ακόμη κι άν μεγάλο ποσοστό των σημείων της καμπύλης είναι -λόγω λάθους- μαύρα, και ταυτοχρόνως υπάρχουν σημεία του background που είναι λόγω λάθους άσπρα. Στις παραμετρικές καμπύλες που ανιχνεύονται με τη μέθοδο Hough περιλαμβάνονται οι ευθείες (βλ. σημειώσεις), οι κύκλοι, οι ελλείψεις κλπ.

Οι κύκλοι προσδιορίζονται από 3 παραμέτρους: Δύο για το κέντρο $(a, b) \in [0, N_1 - 1] \times [0, N_2 - 1]$ και μία για την ακτίνα $\rho \in [0, R_{max}]$. Συνεπώς ο πίνακας ψηφοφορίας του Hough είναι 3Δ πίνακας διαστάσεων $K \times L \times M$ όπου οι ακέραιοι K, L, M αντιστοιχουν στο πλήθος των υποδιαστημάτων στα οποία διαμερίζονται τα παραπάνω διαστήματα τιμών των a, b, ρ.

2 Παραδοτέα

Για τις ανάγκες της εργασίας θα πρέπει να κατασκευάσετε

1. Μία συνάρτηση Python που θα υλοποιεί τη συνέλιξη FIR
2. Μία συνάρτηση Python που θα υλοποιεί την ανίχνευση ακμών με χρήση του τελεστή Sobel. Υποχρεωτικά θα στηρίζεται στη συνέλιξη FIR
3. Μία συνάρτηση Python που θα υλοποιεί την ανίχνευση ακμών με χρήση του τελεστή Laplacian of Gaussian. Υποχρεωτικά θα στηρίζεται στη συνέλιξη FIR
4. Μία συνάρτηση Python που θα υλοποιεί την ανίχνευση κύκλων σε δυαδικές εικόνες
5. Ένα πρόγραμμα επίδειξης με το όνομα `demo.py` που θα ανιχνεύει κύκλους σε RGB εικόνες με δύο διαφορετικούς τρόπους (α) Sobel και Hough, (β) LoG και Hough
6. Ένα report που θα εξηγείτε τη λειτουργία των συναρτήσεων, θα επιδεικνύετε και θα σχολιάζετε τα αποτελέσματα και θα περιγράφετε πιθανές παραδοχές που κάνατε κατά την υλοποίηση

Διευκρινίζεται ότι μπορείτε να χρησιμοποιήσετε κάθε supported έκδοση της Python (3.8 - 3.12), καθώς και να εισαγάγετε δικές σας επιμέρους βοηθητικές μεθόδους στην υλοποίηση, αρκεί όλα αυτά να καταγράφονται στο report σας.

2.1 `fir_conv.py`

Να κατασκευάσετε συνάρτηση

```
def fir_conv(
    in_img_array: np.ndarray,
    h: np.ndarray,
    in_origin: np.ndarray,
    mask_origin: np.ndarray
) -> np.ndarray, np.ndarray:
```

Ορίσματα:

- `in_img_array`: ένας δισδιάστατος `numpy array` τύπου `dtype=float`, που αναπαριστά μια grayscale εικόνα.
- `h`: ένας δισδιάστατος `numpy array` τύπου `dtype=float`, που αναπαριστά μια συνελικτική μάσκα με πραγματικές τιμές.
- `in_origin`: ένας μονοδιάστατος `numpy array` τύπου `dtype=int`, που περιέχει τη θέση της αρχής των αξόνων στη εικόνα εισόδου (προαιρετικό).
- `mask_origin`: ένας μονοδιάστατος `numpy array` τύπου `dtype=int`, που περιέχει τη θέση της αρχής των αξόνων στη μάσκα (προαιρετικό).

Έξοδοι:

- `out_img_array`: ένας δισδιάστατος `numpy array` τύπου `dtype=float`, που αναπαριστά μια grayscale εικόνα.
- `out_origin`: ένας μονοδιάστατος `numpy array` τύπου `dtype=int`, που περιέχει τη θέση της αρχής των αξόνων στη εικόνα εξόδου (προαιρετικό, μόνο αν έχουν προσδιοριστεί τα αντίστοιχα ορίσματα εισόδου).

Λειτουργία: η συνάρτηση δέχεται την εικόνα εισόδου, και τη συνελικτική μάσκα και υπολογίζει τη συνέλιξη τους.

2.2 `sobel_edge.py`

Να κατασκευάσετε συνάρτηση

```
def sobel_edge(
    in_img_array: np.ndarray,
    thres: float
) -> np.ndarray:
```

Ορίσματα:

- `in_img_array`: ένας δισδιάστατος `numpy array` τύπου `dtype=float`, που αναπαριστά μια grayscale εικόνα με τιμές στο διάστημα $[0, 1]$
- `thres`: μια παράμετρος με θετική πραγματική τιμή, που αντιστοιχεί στο κατώφλι του μέτρου του gradient, πάνω από το οποίο ανιχνεύεται ακμή

Έξοδοι:

- `out_img_array`: ένας δισδιάστατος `numpy array` τύπου `dtype=int`, που αναπαριστά μια binary εικόνα με τιμές στο σύνολο $\{0, 1\}$

Λειτουργία: η συνάρτηση δέχεται την εικόνα εισόδου, και υπολογίζει εικόνα ίδιων διαστάσεων με τιμή 1 σε θέσεις ακμών και 0 αλλού.

2.3 `log_edge.py`

Να κατασκευάσετε συνάρτηση

```
def log_edge(
    in_img_array: np.ndarray
) -> np.ndarray:
```

Ορίσματα:

- `in_img_array`: ένας δισδιάστατος `numpy array` τύπου `dtype=float`, που αναπαριστά μια grayscale εικόνα με τιμές στο διάστημα $[0, 1]$

Έξοδοι:

- `out_img_array`: ένας δισδιάστατος `numpy array` τύπου `dtype=int`, που αναπαριστά μια binary εικόνα με τιμές στο σύνολο $\{1\}$

Λειτουργία: η συνάρτηση δέχεται την εικόνα εισόδου, και υπολογίζει εικόνα ίδιων διαστάσεων με τιμή 1 σε θέσεις ακμών και 0 αλλού μέσω της αναζήτησης zero-crossings στην εικόνα που προκύπτει από την εφαρμογή του τελεστή LoG.

2.4 `circ_hough.py`

Να κατασκευάσετε συνάρτηση

```
def circ_hough(
    in_img_array: np.ndarray,
    R_max: float,
```

```
    dim: np.ndarray,
    V_min: int

) -> np.ndarray, np.ndarray:
```

Ορίσματα:

- `in_img_array`: ένας δισδιάστατος `numpy array` τύπου `dtype=int`, που αναπαριστά μια binary εικόνα με τιμές 0 ή 1.
- `R_max`: μεταβλητή εισόδου τύπου `dtype=float`, που περιέχει τη μέγιστη ενδεχόμενη ακτίνα κύκλου.
- `dim`: ένας μονοδιάστατος `numpy array` μήκους 3 τύπου `dtype=int`, που κατά σειρά περιέχει το πλήθος των διαστημάτων στο οποίο διαμερίζεται η οριζόντια διασταση της εικόνας, η κατακόρυφη διάσταση της εικόνας και η περιοχή $[0, R_{max}]$. Αντιστοιχεί στις διαστάσεις του πίνακα στον αλγόριθμο Hough.
- `V_min`: μεταβλητή εισόδου τύπου `dtype=int`, που περιέχει το ελάχιστο πλήθος ψήφων που μπορεί να λάβει ένα κελί του πίνακα Hough προκειμένου να θεωρηθεί πως αντιστοιχεί σε κύκλο.

Έξοδοι:

- `centers`: ένας δισδιάστατος `numpy array` τύπου `dtype=float`, με διαστάσεις $K \times 2$ που περιέχει τις συντεταγμένες των κέντρων των κύκλων που ανιχνεύθηκαν.
- `radii`: ένας μονοδιάστατος `numpy array` τύπου `dtype=float`, μήκους K, που περιέχει τις ακτίνες των κύκλων που ανιχνεύθηκαν.

Λειτουργία: η συνάρτηση δέχεται την δυαδική εικόνα εισόδου, και υπολογίζει τα κέντρα και τις ακτίνες των κύκλων που αυτή περιέχει. Ενδεικτικά, οι εικόνες εισόδου μπορεί να προέρχονται από προηγούμενη εκτέλεση των συναρτήσεων `sobel_edge()` ή `log_edge()`.

2.5 `demo.py`

Με το script `demo.py` να επιδείξετε τη λειτουργία των ζητούμενων συναρτήσεων, χρησιμοποιώντας την εικόνα εισόδου και αναφοράς, basketball_large.png, που σας δίνεται. Εστιάστε στις εξής πτυχές της μελέτης:

- Παρουσιάστε την εικόνα εξόδου της συνάρτησης `sobel_edge` για διάφορες επιλογές του κατωφλίου (`thres`). Συγκεκριμένα δείξτε τα αποτελέσματα εξόδου καθώς και γραφική παράσταση που δείχνει τον αριθμό ανιχνευόμενων σημείων σε σχέση με την χρησιμοποιούμενη τιμή του κατωφλίου. Σχολιάστε τα αποτελέσματα.
- Παρουσιάστε την εικόνα εξόδου της συνάρτησης `log_edge`. Να συγκρίνετε τα αποτελέσματα με αυτά της μεθόδου Sobel.
- Παρουσιάστε την αρχική εικόνα με χαραγμένους επάνω της του κύκλους που ανιχνεύει η συνάρτηση `circ_hough()`. Δείξτε πέντε διαφορετικές εκδοχές για διαφορετικές τιμές του ορίσματος `V_min`. Ειδικά για τη χάραξη των κύκλων, χρησιμοποιήστε έτοιμες συνάρτησεις της Python (η υλοποίηση αφήνεται στην ευχέρειά σας). Για τη επίδειξη της `circ_hough()` αν εμφανίζονται προβλήματα μνήμης ή υπερβολικά μεγάλου χρόνου εκτέλεσης μπορείτε να σμυκρίνετε την εικόνα πριν τρέξετε τη συνάρτησή σας. Εξηγείστε στο report τις σχετικές ενέργειες που κάνατε.

Οι εικόνες που θα περιλάβετε στην αναφορά της εργασίας θα πρέπει να παραχθούν από το script αυτό.

3 Για την υποβολή της εργασίας

Παραδώστε μία αναφορά με τις περιγραφές και τα συμπεράσματα που σας ζητούνται στην εκφώνηση. Η αναφορά θα πρέπει να επιδεικνύει την ορθή λειτουργία του κώδικά σας στην εικόνα που σας δίνεται και να παρουσιάζει και σχολιάζει τις εικόνες και τα διαγράμματα που παράγονται από το πρόγραμμα επίδειξης.

Ο κώδικας θα πρέπει να είναι σχολιασμένος ώστε να είναι κατανοητό τι ακριβώς λειτουργία επιτελεί (σε θεωρητικό επίπεδο, όχι σε επίπεδο κλήσης συναρτήσεων). Επίσης, ο κώδικας θα πρέπει να εκτελείται και να υπολογίζει τα σωστά αποτελέσματα για *οποιαδήποτε* είσοδο πληροί τις υποθέσεις της εκφώνησης, και όχι μόνο για την εικόνα που σας δίνεται.

Απαραίτητες προϋποθέσεις για την βαθμολόγηση της εργασίας σας είναι ο κώδικας να εκτελείται χωρίς σφάλμα (μόνο demos που εκτελούνται επιτυχώς θα βαθμολογηθούν), καθώς και να τηρούνται τα ακόλουθα:

- Υποβάλετε ένα και μόνο αρχείο, τύπου zip.
- Το όνομα του αρχείου πρέπει να είναι `AEM.zip`, όπου `AEM` είναι τα τέσσερα ή πέντε ψηφία του Α.Ε.Μ. του φοιτητή.
- Το προς υποβολή αρχείο πρέπει να περιέχει τα αρχεία κώδικα Python και το αρχείο `report.pdf` το οποίο θα είναι η αναφορά της εργασίας.
- Η αναφορά πρέπει να είναι ένα αρχείο τύπου PDF, και να έχει όνομα `report.pdf`.
- Όλα τα αρχεία κώδικα πρέπει να είναι αρχεία κειμένου τύπου UTF-8, και να έχουν κατάληξη `.py`.
- Το αρχείο τύπου zip που θα υποβάλετε δεν πρέπει να περιέχει κανέναν φάκελο.
- Μην υποβάλετε τις εικόνες που σας δίνονται για πειραματισμό.
- Μην υποβάλετε αρχεία που δεν χρειάζονται για την λειτουργία του κώδικά σας, ή φακέλους/αρχεία που δημιουργεί το λειτουργικό σας, πχ “Thumbs.db”, “.DS_Store”, “.directory”.
- Για την ονομασία των αρχείων που περιέχονται στο προς υποβολή αρχείο, χρησιμοποιείτε μόνο αγγλικούς χαρακτήρες, και όχι ελληνικούς ή άλλα σύμβολα, πχ “#”, “$”, “%” κλπ.