Ψηφιακή Επεξεργασία Εικόνας
-Εργασία 1-

—

salient points

Α. Ντελόπουλος

Άνοιξη 2022

1 Εισαγωγικά

Στην πρώτη ενότητα της εργασίας θα υλοποιήσετε μια σουίτα από ρουτίνες (Ενότητες 1.2, 1.3 και 1.1) τις οποίες θα πρέπει να χρησιμοποιήσετε για να δώσετε λύση στο τελικό, real life, πρόβλημα (Ενότητα 2). Πιο συγκεκριμένα, θα υλοποιήσετε:

1. Την περιστροφή εικόνας
2. Έναν απλό δικό σας τοπικό περιγραφέα (local descriptor)
3. Τον Harris corner detector

1.1 Rotation

Κατασκευάστε την συνάρτηση:

```
function rotImg = myImgRotation(img, angle)
```

η οποία θα λαμβάνει σαν είσοδο μία εικόνα `img` και θα την περιστρέφει αντίστροφα από την φορά του ρολογιού κατά γωνία `angle` σε μοίρες. Η συνάρτηση σας θα πρέπει να λειτουργεί ανεξάρτητα του αριθμού των καναλιών της εικόνας εισόδου (π.χ. RGB ή grayscale). Η εικόνα εξόδου `rotImg` θα πρέπει να έχει τις κατάλληλες διαστάσεις για να χωράει ολόκληρη την εικόνα εισόδου μετά την περιστροφή της. Υποθέστε ότι το background είναι μαύρο.

Η ιδέα πίσω από την περιστροφή της εικόνας είναι να βρείτε την αντιστοίχιση του κάθε pixel στην μετασχηματισμένη εικόνα ως προς την αρχική εικόνα εισόδου. Για παράδειγμα, αντί να πείτε πως το pixel με συντεταγμένες (1,1) στην αρχική εικόνα είναι το pixel με συντεταγμένες (3.4, 6.1) στην μετασχηματισμένη, θα πρέπει να κάνετε το αντίθετο (πως το pixel με συντεταγένες (1,1) στην μετασχηματισμένη εικόνα είναι το pixel (.4, 1.4) της αρχικής. Επιπλέον, θεωρήστε `bilinear interpolation` (Παράδειγμα στον πίνακα 1) για να υπολογίσετε την τιμή του pixel. Οποιαδήποτε στρογγυλοποίηση συντεταγμένων pixel προκύψει να είναι προς τα κάτω.

p_1	p_2	p_3
p_4	$\mathbf{p_5} = ?$	p_6
p_7	p_8	p_9

Πίνακας 1: Παράδειγμα bilinear interpolation. Η τιμή του pixel p_5 είναι το weighted average των γειτονικών pixels, $p_5 = \frac{(p_2+p_4+p_6+p_8)}{4}$

1.1.1 Παραδοτέα

Να δείξετε:

1. Πραγματοποιήστε περιστροφή της εικόνας εισόδου TestIm1.png κατά $\theta_1 = 35°$, δείξτε το αποτέλεσμα.
2. Πραγματοποιήστε περιστροφή της εικόνας εισόδου κατά $\theta_2 = 222°$, δείξτε το αποτέλεσμα.

1.2 Ένας Local Descriptor

Η εργαλειοθήκη ξεκινάει με την κατασκευή μιας ρουτίνας που υλοποιεί τον υπολογισμό ένός απλού rotation invariant περιγραφέα της γειτονιάς ενός σημείου $p = [p_1,\ p_2]$. Ο συγκεκριμένος local descriptor κατασκευάζεται σαρώνοντας διαδοχικούς ομόκεντρους κύκλους με κέντρο το p και ακτίνες $\rho = \rho_m : \rho_s : \rho_M$ όπου τόσο η μικρότερη/μεγαλύτερη ακτίνα όσο και το βήμα ρ_s είναι παράμετροι του αλγορίθμου υπολογισμού. Οι κύκλοι σαρώνονται σε $2\pi/N$ σημεία και έτσι για καθέναν από τους κύκλους υπολογίζεται ένα διάνυσμα $x_\rho = [x_{\rho,0}, \ldots, x_{\rho,N-1}]$. Τα στοιχεία του διανύσματος υπολογίζονται από παρεμβολή των στοιχείων της εικόνας με βάση τη θέση στην οποία αντιστοιχούν. Η βασική έκδοση τους περιγραφέα είναι ένα διάνυσμα d με τόσα στοιχεία όσα οι κύκλοι $((\rho_M - \rho_m)/\rho_s)$ που το καθένα έχει τιμή ίση προς το μέσο όρο του αντίστοιχου x_ρ.

Να κατασκευάσετε τη συνάρτηση:

```
function d = myLocalDescriptor(I,p,rhom,rhoM,rhostep,N)
```

η οποία λαμβάνει ως είσοδο μία grayscale εικόνα `I` και τις τιμές των παραπάνω παραμέτρων `rhom,rhoM,rhostep, N` και επιστρέφει το διάνυσμα του περιγραφέα για τη γειτονιά του σημείου `p`. Για ευκολία είναι αποδεκτό η συνάρτηση να επιστρέφει το κενό διάνυσμα αν το `p` βρίσκεται τόσο κοντά στα όρια της εικόνας που ο μεγαλύτερος κύκλος ακτίνας `rhoM` φτάνει έξω από την εικόνα.

Να κατασκευάσετε τη συνάρτηση:

```
function d = myLocalDescriptorUpgrade(I,p,rhom,rhoM,rhostep,N)
```

στην οποία θα αντικαταστήσετε την βασική εκδοχή του περιγραφέα με άλλη δικής σας έμπνευσης που παραμένει rotation invariant αλλά περιέχει πλουσιότερη πληροφορία. Δεν είναι απαραίτητο ο νέος descriptor να έχει το ίδιο μήκος με αυτόν της βασικής έκδοσης.

1.2.1 Παραδοτέα

Να δείξετε:

1. Την βασική έκδοση του περιγραφέα του pixel $p = [100, 100]$ για την αρχική εικόνα πίνακα testimg1.png, του αντίστοιχου pixel μετά την περιστροφή κατά θ_1 και θ_2 όπως παραπάνω. Οι υπολογισμοί να γίνουν με `rhom=5,rhoM=20,rhostep=1, N=8`
2. Την βασική έκδοση του περιγραφέα των pixels $q = [200, 200]$ και $q = [202, 202]$ για την αρχική εικόνα πίνακα testimg1.png με τις ίδιες τιμές παραμέτρων.
3. Να επαναλάβετε τα παραπάνω για τον δικό σας περιγραφέα

1.3 Harris corner detector

Για τον εντοπισμό σημείων ενδιαφέροντος θα υλοποιήσετε τον αλγόριθμο Harris corner detector που πρωτοπροτάθηκε στο το άρθρο [1]. Ακολουθεί η βασική ιδέα. Ας υποθέσουμε ότι $w(x_1, x_2)$ είναι μία διδιάστατη συνάρτηση που έχει μή μηδενικές τιμές κοντά στην αρχή των αξόνων και "πεθαίνει" καθώς το $(x_1,\ x_2)$ απομακρύνεται από το $(0,\ 0)$. Για παράδειγμα:

$$w(x_1, x_2) \ = \ exp\{-\frac{x_1^2 + x_2^2}{2\sigma^2}\} \tag{1}$$

Αν $I(x_1, x_2)$ είναι η φωτεινότητα μιας gray scale εικόνας, τότε η συνάρτηση

$$E(x_1, x_2; p_1, p_2) \ = \ \sum_{u_1,u_2} w(u_1, u_2)\|I(p_1 + u_1 + x_1, p_2 + u_2 + x_2) - I(p_1 + u_1, p_2 + u_2)\|^2 \tag{2}$$

παρουσιάζει την εξής συμπεριφορά:

- Αν το σημείο (p_1, p_2) βρίσκεται σε μία σχετικά ομαλή περιοχή της εικόνας τότε $E(x_1, x_2; p_1, p_2) \approx 0$.
- Αν το (p_1, p_2) βρίσκεται πάνω ή πολύ κοντά σε μία ακμή η οποία - σε σχέση με το μέγεθος της μη μηδενικής περιοχής του παραθύρου $w(x_1, x_2)$ - είναι ευθύγραμμη, τότε $E(x_1, x_2; p_1, p_2) \approx 0$ όταν η ολίσθηση $(x_1,\ x_2)$ είναι (σχεδόν) παράλληλη με την ακμή ενώ θα παίρνει τιμές $>> 0$ όταν η ολίσθηση $(x_1,\ x_2)$ είναι προς την κάθετη στην ακμή κατεύθυνση.

- Αν το (p_1, p_2) βρίσκεται πάνω ή πολύ κοντά σε μία γωνία τότε $E(x_1, x_2; p_1, p_2) >> 0$ για οποιαδήποτε κατεύθυνση της ολίσθησης $(x_1,\ x_2)$.

Συνεπώς η μελέτη της $E(x_1, x_2; p_1, p_2)$ ως συνάρτηση των $(x_1,\ x_2)$ μας επιτρέπει να αποφασίζουμε αν το σημείο $(p_1,\ p_2)$ βρίσκεται σε περιοχή ομαλής φωτεινότητας, σε ακμή ή σε γωνία. Χρησιμοποιώντας ανάπτυγμα Taylor γύρω από το σημείο $(p_1 + u_1, p_2 + u_2)$ (ως προς τις μεταβλητές (x_1, x_2), η $E(x_1, x_2; p_1, p_2)$ γράφεται

$$E(x_1, x_2; p_1, p_2) \approx \sum_{u_1,u_2} w(u_1, u_2)\|x_1 I_1(p_1 + u_1, p_2 + u_2) + x_2 I_2(p_1 + u_1, p_2 + u_2)\|^2 \tag{3}$$

όπου $I_1(x_1, x_2) = \partial I(x_1, x_2)/\partial x_1$, $I_2(x_1, x_2) = \partial I(x_1, x_2)/\partial x_2$ και έχουμε παραλήψει τους όρους υψηλότερης τάξης. Εναλλακτικά,

$$E(x_1, x_2; p_1, p_2) \approx [x_1,\ x_2]\, \mathbf{M}(p_1, p_2) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4}$$

όπου

$$\mathbf{M}(p_1, p_2) = \sum_{u_1,u_2} w(u_1, u_2)\mathbf{A}(u_1, u_2; p_1, p_2) \tag{5}$$

και

$$\mathbf{A}(u_1, u_2; p_1, p_2) = \begin{bmatrix} I_1(p_1 + u_1, p_2 + u_2)^2 & I_1(p_1 + u_1, p_2 + u_2)I_2(p_1 + u_1, p_2 + u_2) \\ I_1(p_1 + u_1, p_2 + u_2)I_2(p_1 + u_1, p_2 + u_2) & I_2(p_1 + u_1, p_2 + u_2)^2 \end{bmatrix} \tag{6}$$

Πρακτικά οι μερικές παράγωγοι που συμμετέχουν στις παραπάνω εκφράσεις υπολογίζονται με τη χρήση κατάλληλων συνελικτικών μασκών όπως έχουμε εξηγήσει στη θεωρία. Ο υπολογισμός τους μάλιστα γίνεται μια κι έξω για όλη την εικόνα. Οι τρείς εναλλακτικές συμπεριφορές της συνάρτησης $E(x_1, x_2; p_1, p_2)$ που είδαμε παραπάνω αντιστοιχούν σε τρεις αντίστοιχες διαφορετικές εκδοχές για τις ιδιοτιμές του πίνακα $\mathbf{M}(p_1, p_2)$:

- Αν το σημείο (p_1, p_2) βρίσκεται σε μία σχετικά ομαλή περιοχή της εικόνας τότε $\lambda_1 \approx 0$ και $\lambda_1 \approx 0$.
- Αν το (p_1, p_2) βρίσκεται πάνω ή πολύ κοντά σε μία ακμή τότε $\lambda_1 >> \lambda_2 \approx 0$.
- Αν το (p_1, p_2) βρίσκεται πάνω ή πολύ κοντά σε μία γωνία τότε $\lambda_1 | approx \lambda_2 >> 0$.

Συνεπώς η απόφαση για το αν η περιοχή του σημείου (p_1, p_2) είναι ομαλή, βρίσκεται σε ακμή ή σε γωνία μπορεί να ληφθεί από τη μελέτη των δύο ιδιοτιμών του πίνακα $\mathbf{M}(p_1, p_2)$. Για λόγους υπολογιστική πολυπλοκότητας - καθόσον ο υπολογισμός ιδιοτιμών απαιτεί υπολογισμό τετραγωνικών ριζών - χρησιμοποιούμε την παρακάτω μετρική γωνιότητας:

$$R(p_1, p_2) = det(\mathbf{M}(p_1, p_2)) - kTrace(\mathbf{M}(p_1, p_2))^2 \tag{7}$$

όπου $k > 0$ μια οριζόμενη από εμάς παράμετρος. Επειδή

$$R(p_1, p_2) = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{8}$$

η τιμή R θα είναι θετική όταν το σημείο (p_1, p_2) είναι κοντά σε γωνίες, αρνητική στη γειτονιά ακμών και κοντά στο μηδέν σε ομοιόμορφες περιοχές. Κατάλληλα κατώφλια μπορούν να χρησιμοποιηθούν για να επιλεγούν οι πλέον ευδιάκριτες γωνίες.

Κατασκευάστε την συνάρτηση:

```
function c = isCorner(I, p, k, Rthres)
```

η οποία επιστρέφει logical true/false αν η grayscale εικόνα I εμφανίζει γωνία στο pixel $p = [p_1,\ p_2]$. Η παράμετρος εισόδου k αντιστοιχεί στην παράμετρο k της εξ. 7 και η παράμετρος εισόδου $Rthres$ ορίζει το κατώφλι πάνω από το οποίο η τιμή της $R(p_1, p_2)$ θεωρείται αρκετά θετική ώστε να εντοπίζεται γωνία.

Με χρήση της συνάρτησης αυτής να κατασκευαστεί η συνάρτηση

```
function corners = myDetectHarrisFeatures(I)
```

η οποία υλοποιεί τον αλγόριθμο, όπως περιγράφεται παραπάνω. Η μεταβλητή `I` είναι ένας πίνακας 2 διαστάσεων και περιέχει μία εικόνα σε gray scale, με τιμές πραγματικούς αριθμούς στο διάστημα $[0, 1]$. Η μεταβλητή `corners` είναι ένας πίνακας δύο στηλών, και κάθε του γραμμή αντιστοιχεί στις συντεταγμένες μίας γωνίας. Ο αλγόριθμος είναι ήδη υλοποιημένος στο Image Processing toolbox της MATLAB. Οι παρακάτω εντολές επιδεικνύουν τη χρήση του:

```
I=imread('cameraman.tif');
corners=detectHarrisFeatures(I);
figure
hold on
imshow(I);
plot(corners);
hold off
```

Μπορείτε να χρησιμοποιήσετε τη μεταβλητή `corners.Location` για να επαληθεύσετε την υλοποίησή σας συγκρίνοντας τα δικά σας αποτελέσματα με αυτά της έτοιμης συνάρτησης. Η ορθότητα της υλοποίησης δεν εξαρτάται προφανώς από την σειρά με την οποία παρατίθενται οι γωνίες.

1.3.1 Παραδοτέα

Να δείξετε:

1. Τις γωνίες που εντοπίσατε πάνω στην grayscale εικόνα εισόδου δημιουργώντας ένα κόκκινο τετράγωνο 5×5 pixels με κέντρο την κάθε εντοπισμένη κορυφή.

2 Το πρόβλημα

Σε αυτή την ενότητα θα χρησιμοποιήσετε τις συναρτήσεις που έχετε φτιάξει σε αυτή την εργασία για να λύσετε το παρακάτω πρόβλημα.

Οι εικόνες im1 και im2 τραβήχτηκαν διαδοχικά από μία κάμερα που φωτογράφισε μια σκηνή. Στόχος είναι να τις ενώσετε σε μία αφού περιστρέψετε κατάλληλα τη δεύτερη.

Φτιάξτε τη συνάρτηση `Im=myStitch(im1,im2)` η οποία θα ενώνει την εικόνα im2 στην εικόνα im1.

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 στις εικόνες που σας δίνονται.

Ο κώδικας θα πρέπει να είναι σχολιασμένος ώστε να είναι κατανοητό τι ακριβώς λειτουργία επιτελεί (σε θεωρητικό επίπεδο, όχι σε επίπεδο κλίσης 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 τις εικόνες που σας δίνονται.

Απαραίτητες προϋποθέσεις για την βαθμολόγηση της εργασίας σας είναι ο κώδικας να εκτελείται χωρίς σφάλμα, 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 ψηφία του Α.Ε.Μ. του φοιτητή της ομάδας.
- Το προς υποβολή αρχείο πρέπει να περιέχει τα αρχεία κώδικα Maltab 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 `m`.
- 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.

Αναφορές

[1] C. Harris and M. Stephens, *A Combined Corner and Edge Detector*, Proceedings of the 4th Alvey Vision Conference, August 1988, pp. 147-151