



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Ψηφιακή Επεξεργασία Εικόνας

Εργασία 3: Image Registration

Ευσεβεία Νεστοροπούλου

AEM: 9703

nestoropo@ece.auth.gr

Εαρινό Εξάμηνο 2023

Περιεχόμενα

1 Εισαγωγή	2
2 Local Descriptor	2
2.1 myLocaDescriptor	2
2.2 myLocaDescriptorUpgrade	3
3 Harris corner detector	4
4 Matching descriptors	6
5 RANSAC	7
6 imForest1 & imForest2	11
7 Παρατηρήσεις	13

Κεφάλαιο 1

Εισαγωγή

Στην παρούσα εργασία, στο μάθημα Ψηφιακή Επεξεργασία Εικόνας, μας δίνονται δύο εικόνες, οι οποίες έχουν κοινά σημεία, και εμείς καλούμαστε να τις ενώσουμε. Πιο συγκεκριμένα, πρέπει να υλοποιήσουμε δύο συναρτήσεις για την εύρεση περιγραφέων σε σημεία της εικόνας [1ο Βήμα]. Έπειτα, να υλοποιήσουμε τον Harris Corner Detector για να εντοπίσουμε τα ιδιάζωντα σημεία στις εικόνες (γωνίες) [2ο Βήμα] και το Matching descriptors, για να αντιστοιχίσουμε τα σημεία στις δύο εικόνες [3ο Βήμα]. Τέλος, υλοποιούμε τον αλγόριθμο Ransac για να ενώσουμε τις εικόνες μεταξύ τους [4ο Βήμα].

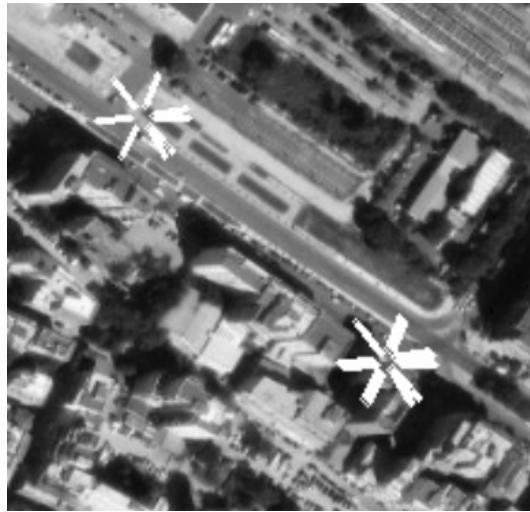
Κεφάλαιο 2

Local Descriptor

Έχουν υλοποιηθεί στο αρχείο descriptors.py

2.1 myLocaDescriptor

Όπως αναφέρεται και στην εκφώνηση, ο αλγόριθμος χρησιμοποιεί διαδοχικούς ομόκεντρους κύκλους με κέντρο το δοσμένο pixel και ακτίνες $r_{min} : r_{step} : r_{max}$, για να σαρώσει την περιοχή γύρω από το pixel. Βρίσκει N σημεία πάνω σε κάθε κύκλο, παίρνει την τιμή της εικόνας σε εκείνα τα σημεία και τέλος, υπολογίζει τον μέσο όρο τους ώστε να έχουμε μία τιμή για κάθε κύκλο. Στην περίπτωση μας, έχουμε 15 τιμές όπως φαίνεται παρακάτω.

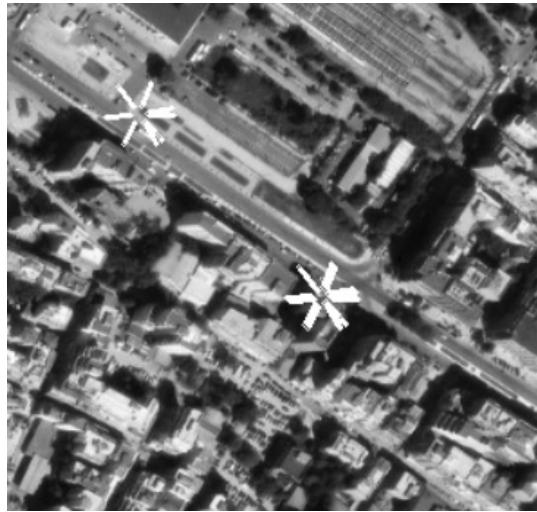


Σχήμα 2.1: Οι περιγραφέις για τα τρία σημεία της εκφώνησης

Ακτίνες	Σημείο [100,100]	Σημείο [200,200]	Σημείο [202,202]
5	146	97	115
6	146	85	102
7	144	86	90
8	147	86	98
9	153	83	90
10	156	74	88
11	158	87	80
12	153	80	78
13	164	85	86
14	153	85	87
15	158	89	87
16	153	86	85
17	141	86	78
18	131	83	73
19	128	83	71

2.2 myLocaDescriptorUpgrade

Στον δικό μου αλγόριθμο για περιγραφέις, πρόσθεσα το εξής: αν για την γωνία ισχύει $\theta \% 20 = 0$, τότε αποθηκεύω την διπλασιασμένη τιμή της εικόνας για εκείνο το σημείο.



Σχήμα 2.2: Οι περιγραφέις για τα τρία σημεία της εκφώνησης

Ακτίνες	Σημείο [100,100]	Σημείο [200,200]	Σημείο [202,202]
5	201	106	131
6	193	102	124
7	189	110	107
8	210	109	125
9	202	126	128
10	198	124	116
11	196	124	95
12	189	86	91
13	194	81	97
14	175	87	90
15	163	107	91
16	163	118	107
17	168	101	111
18	194	100	115
19	201	97	109

Κεφάλαιο 3

Harris corner detector

Καλούμαστε να υλοποιήσουμε τον αλγόριθμο για τον εντοπισμό των ιδιαίζωντων σημείων στις εικόνες (γωνίες). Ακολουθώντας τις οδηγίες της εκφώνησης, πρέπει να υπολο-

γήσουμε τις παραγώγους ως προς τους δύο άξονες, καθώς και τα τετράγωνα τους για να δημιουργήσουμε τους πίνακες:

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

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

με offset = 4. Για τον υπολογισμό των παραγώγων, χρησιμοποίησα την συνάρτηση cv.Sobel η οποία χρησιμοποιείται ειδικά για εικόνες, σε αντίθεση με την np.gradient. Με τις ιδιοτιμές του πίνακα M μπορούμε υπολογίσουμε την τιμή R για κάθε σημείο και να ξεχωρίσουμε αν βρισκεται σε γωνία, πάνω σε ευθεία ή σε ομαλή περιοχή.

$$R(p_1, p_2) = \det(\mathbf{M}(p_1, p_2)) - k \text{Trace}(\mathbf{M}(p_1, p_2))^2$$

Έπειτα, αφού υπολογίσω τον πίνακα R για όλα τα σημεία, τον κανονικοποιώ και με μία τιμή threshold=0.25 αποφασίζω αν το σημείο είναι γωνία ή όχι. Η τιμή του threshold είναι ψηλή αλλά σε άλλη περίπτωση λόγω του μεγάλου πλήθους των εντοπισμένων γωνιών δεν μπορούσα να επεξεργαστώ τα δεδομένα. Ενδεικτικά, εντοπίζονται περίπου 15.000 γωνίες στην 1η εικόνα και 6.000 στην 2η.



Σχήμα 3.1: Οι εντοπισμένες γωνίες στην 1η εικόνα



Σχήμα 3.2: Οι εντοπισμένες γωνίες στην 2η εικόνα

Κεφάλαιο 4

Matching descriptors

Σκοπός μας είναι να αντιστοιχίσουμε τα εντοπισμένα σημεία μεταξύ τους χρησιμοποιώντας τους περιγραφείς τους. Καλώντας την συνάρτηση `calculateDistances`, για κάθε γωνία της πρώτης εικόνας υπολογίζω την απόσταση του περιγραφέα της από όλους τους περιγραφείς των γωνιών της δεύτερης εικόνας. 'Όταν η γωνία προς διερεύνηση είναι κοντά στα άκρα της φωτογραφίας, ορίζω μεγάλο αριθμό ως την απόσταση ώστε να γίνει κακό match και να μην ληθφεί αργότερα υπόψιν.

Με την συνάρτηση `descriptorMatching`, έχοντας τις αποστάσεις των περιγραφέων όλων των σημείων με όλα τα σημεία, για κάθε σημείο της πρώτης εικόνας βρίσκω το βέλτιστο αντιστοιχισμένο σημείο από την δεύτερη εικόνα. Στο τέλος, παίρνω το 10% των καλύτερα αντιστοιχισμένων σημείων και επιστρέφω πίνακα που περιέχει το index του σημείου της πρώτης εικόνας, το index του σημείου της δεύτερης και την απόσταση των

περιγραφέων.

Κεφάλαιο 5

RANSAC

Η εκφώνηση πρότεινε να επιλέγουμε τυχαία δύο ζευγάρια σημείων και να υπολογίζουμε τα d και theta μεταξύ τους. Για κάθε d και theta που θα υπολογίζαμε, θα εφαρμόζαμε σε όλα τα σημεία της δεύτερης εικόνας τον μετασχηματισμό:

$$P_2 = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} P_1 + \vec{d}$$

ώστε να μετακινηθούν και να "κάτσουν" πάνω στα σημεία της πρώτης εικόνας. Ωστόσο, λόγω τις τυχαιότητας του αλγορίθμου, οι παράμετροι d και theta σε κάθε run ήταν πολύ διαφορετικοί και δεν κατάφερα να μειώσω αυτή την τυχαιότητα. Για τον λόγο αυτό, υλοποίησα την συνάρτηση RANSAC χρησιμοποιώντας μόνο ένα ζεύγος σωστά αντιστοιχισμένων σημείων την φορά.

Πιο συγκεκριμένα, έχοντας τις συντεταγμένες (x, y) των δύο σημείων (ένα σημείο για κάθε εικόνα), προσθέτω στο im2_x το πλάτος της πρώτης εικόνας, ώστε να έχω τις συντεταγμένες σαν να ήταν παραταγμένες εικόνες δίπλα δίπλα, όπως φαίνεται στην παρακάτω combined εικόνα. Με αυτόν τον τρόπο, μπορώ να ενώσω τα δύο σημεία και να βρω τα χαρακτηριστικά της μεταξύ τους ευθείας, rho και theta. Τα χαρακτηριστικά αυτά χρησιμοποιούνται στην συνέχεια στην συνάρτηση getTransformedPoints για τον μετασχηματισμό των σημείων της δεύτερης εικόνας.



Σχήμα 5.1: combined image

Για κάθε μετασχηματισμένο σημείο, υπολογίζω την απόσταση του από το αντίστοιχο σημείο στην πρώτη εικόνα και χρησιμοποιώντας κάποιο threshold, αποφασίζω αν είναι inlier ή outlier. Ως κριτήριο για τις βέλτιστες παραμέτρους, χρησιμοποιήσα το ποσοστό των inliers σε σχέση με τα συνολικά αντιστοιχισμένα σημεία. Κάνω update το score έως ότου βρω το βέλτιστο. Οι βέλτιστοι παράμετροι μετά από δοκιμές είναι:

$$d = [760, 0], \theta = 12$$



Σχήμα 5.2: Τα inliers στην πρώτη εικόνα



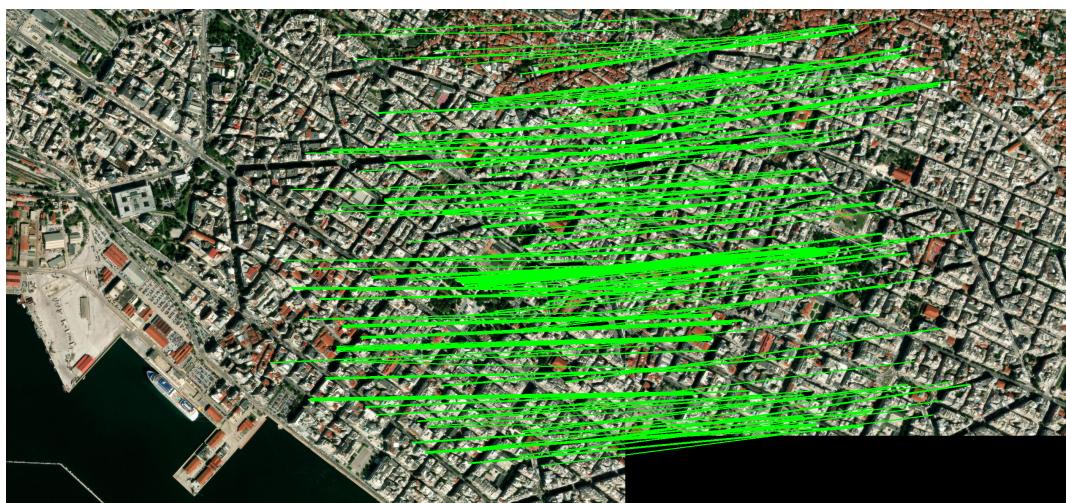
Σχήμα 5.3: Τα inliers στην δεύτερη εικόνα



Σχήμα 5.4: Τα outliers στην πρώτη εικόνα



Σχήμα 5.5: Τα outliers στην δεύτερη εικόνα



Σχήμα 5.6: Τα αντιστοιχισμένα σημεία



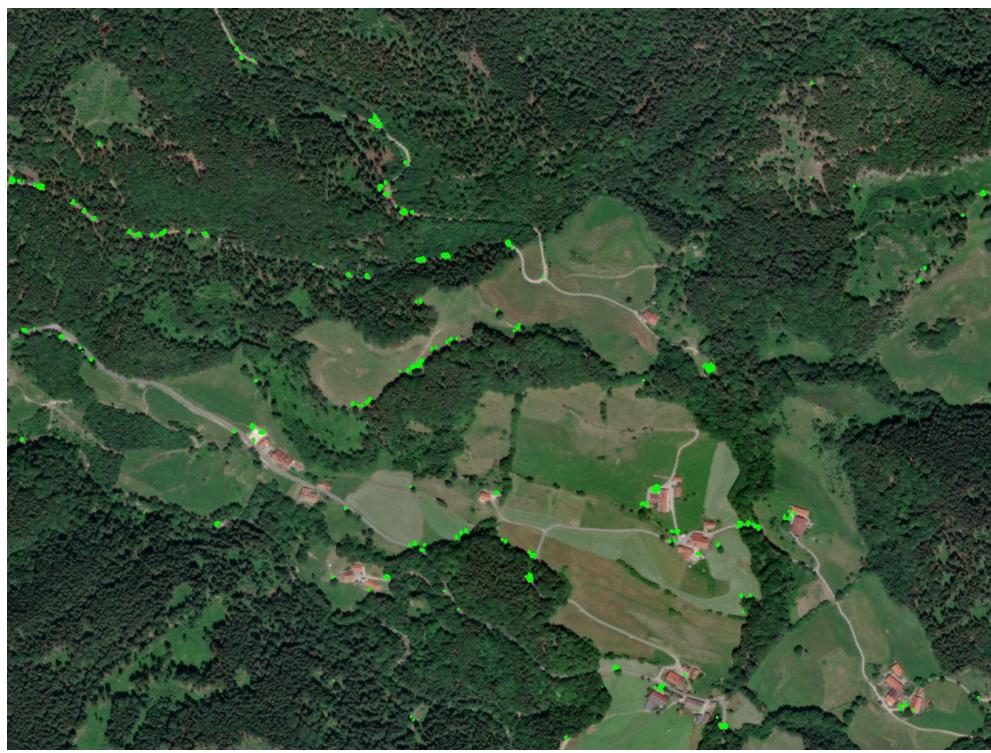
Σχήμα 5.7: Η τελική εικόνα

Κεφάλαιο 6

imForest1 & imForest2

Λόγω του υψηλού threshold που είχα ορίσει για τις φωτογραφίες της πόλης, στις φωτογραφίες του δάσους εντοπίζονται πολύ λιγότερες γωνίες. Πιο συγκεκριμένα, 2070 στην πρώτη και 100 στην δεύτερη.

Οι παράμετροις υπολογίστηκαν: $d=[760, 0]$ και $\theta=5$.



Σχήμα 6.1: Οι εντοπισμένες γωνίες



Σχήμα 6.2: Η τελική εικόνα

Κεφάλαιο 7

Παρατηρήσεις

- Ο κώδικας είναι πλήρως λειτουργικός αλλά λόγω των πολλών σημείων που εντοπίζει και επεξεργάζεται, απαιτούνται τουλάχιστον 20 λεπτά για να τρέξει στον δικό μου υπολογιστή. Για το λόγο αυτό έχω αποθηκεύσει τα δεδομένα για τις γωνίες στην πρώτη εικόνα, τις γωνίες στην δεύτερη και τις αποστάσεις των περιγραφέων στα αρχεία img1.npy, img2.npy, distances.npy αντίστοιχα, ώστε να μπορείτε να μπορείτε να τα φορτώσετε απευθείας. Σε αυτή την περίπτωση, μπορείτε να ορίσετε την μεταβλητή debug = False στην αρχή του αρχείου.
- Στην περίπτωση που οι δύο φωτογραφίες είχαν τραβηχτεί από διαφορετικό ύψος, οι τιμές των περιγραφέων θα ήταν διαφορετικές για τα ίδια σημεία άρα δεν θα μπορούσαμε να αντιστοιχίσουμε σωστά τις δύο φωτογραφίες. Ίσως αν παίρναμε περισσότερα σημεία αν κύκλο για τους περιγραφείς και μεγαλύτερο step για τους κύκλους να λύναμε αυτό το πρόβλημα. Ωστόσο, δεν πρόλαβα να το υλοποιήσω.