Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Ηλεκτρονικής και Υπολογιστών

Εργασία 3 στην Ψηφιακή Επεξεργασία Εικόνας

Εργασία του
Φώτη Αλεξανδρίδη, ΑΕΜ: 9953
faalexandr@ece.auth.gr

29 Ιουνίου 2023

Περιεχόμενα

Κεφάλαιο 1

Overview

1.1 Στόχος

Η 3η εργασία στο μάθημα της ψηφιακής επεξεργασίας εικόνας πραγματεύεται την διαδικασία του Image Registration, δηλαδή της αναγνώρισης χαρακτηριστικών σε εικόνες, και αντιστοίχιση των πιθανώς ίδιων χαρακτηριστικών σε 2 ή περισσότερες εικόνες.

1.2 Λογική πορεία

Αρχικά στα πλαίσια της εργασίας υλοποιείται ένας τοπικός περιγραφέας ανεξαρτήτου περιστροφής (rotation invariant descriptor) για τα σημεία μιας εικόνας, καθώς και μια αναβαθμισμένη εκδοχή του. Σαν δεύτερο βήμα, υλοποιείται ο Harris Corner Detector, ένας αλγόριθμος που εντοπίζει σημεία-γωνίες σε μια ασπρόμαυρη εικόνα. Έπειτα, αναπτύσσεται αλγόριθμος αντιστοίχισης σημείων σε δύο εικόνες που προβάλλουν την ίδια πληροφορία με διαφορετική μετατόπιση-περιστροφή, καθώς και αλγόριθμος που επιστρέφει αυτόν τον μετασχηματισμό.

Τέλος, οι προαναφερόμενοι αλγόριθμοι χρησιμοποιούνται για να λύσουν ένα πραγματικό πρόβλημα, αυτό της συνένωσης αεροφωτογραφιών που έχουν ληφθει διαδοχικά σε μια περιοχή.

1.3 Σημείωση

Στην εργασία αυτή, λόγω πολύ μεγάλου φόρτου εργασιών και εξεταστικής, τα αποτελέσματα δεν είναι σε πολλές περιπτώσεις εξ' ολοκλήρου ορθά. Ως ένα βαθμό, αυτό οφείλεται στην φύση του προβλήματος και στους αλγορίθμους-μεθοδολογίες που χρησιμοποιούνται. Σίγουρα με βελτίωση των υπερπαραμέτρων του προβλήματος θα παίρναμε καλύτερα αποτελέσματα, κάτι το οποίο λόγω έλλειψης χρόνου δεν έγινε.

Κεφάλαιο 2

Local Descriptor

2.1 Overview

Σε αυτό το τμήμα της εργασίας καλούμαστε να δημιουργήσουμε έναν τοπικό σημειακό περιγραφέα ανεξαρτήτου περιστροφής.

2.2 Plain Descriptor

Αρχικά, δίνεται η οδηγία για έναν περιγραφέα που βασίζεται στην σάρωση σημείων ομόκεντρων κύκλων γύρω από το ενδιαφερόμενο σημείο, και στην λήψη του μέσου όρου των τιμών για κάθε κύκλο. Η ελάχιστη και μέγιστη ακτίνα των κύκλων, καθώς και το βήμα αύξησης της ακτίνας (κατ΄ επέκταση ο αριθμός των κύκλων), καθώς και ο αριθμός των σημείων που σαρώνονται για κάθε κύκλο, αποτελούν υπερπαραμέτρους του αλγορίθμου.

Ο αλγόριθμος υλοποιείται στο αρχείο `descriptors.py`, στην συνάρτηση `myLocalDescriptor`. Η υλοποίησή του βασίζεται στις πολύ ξεκάθαρες και περιγραφικές οδηγίες που δίνονται στην εκφώνηση [1].

2.3 Upgraded Descriptor

Στον αναβαθμισμένο περιγραφέα, επιστρέφεται η ίδια πληροφορία με τον αρχικό περιγραφέα, καθώς και για κάθε κύκλο η διαφορά της μέγιστης μείον την ελάχιστη φωτεινότητα. Η προσθήκη αυτή μπορεί να συμβάλλει στο να έχουμε παραπάνω πληροφορία για ένα σημείο, καθώς μια σχετικά ομαλή περιοχή της εικόνας και μια περιοχή με μεγάλες φωτεινές και σκοτεινές μεταβολές μπορούν να έχουν τον ίδιο αρχικό περιγραφέα αλλά, κατά κανόνα, δεν αντιστοιχίζονται. Επιπρόσθετα, η αναβαθμισμένη εκδοχή διατηρεί την ανεξάρτητη από την περιστροφή φύση του αρχικού περιγραφέα.

Ο αλγόριθμος υλοποιείται στο ίδιο αρχείο με τον αρχικό περιγραφέα, στην συνάρτηση `myLocalDescriptorUpgrade`. Η υλοποίηση είναι η ίδια με την υλοποίηση του αρχικού περιγραφέα, μόνο που τώρα επιστρέφεται και η διαφορά της μέγιστης μείον την ελάχιστη τιμή της φωτεινότητας στο διάνυσμα.

2.4 Αποτελέσματα

Τα αποτελέσματα παράγονται με την εκτέλεση του αρχείου `demo_descriptors.py`. Τα σημεία που μας ζητούνται είναι τα:

$$p = \begin{bmatrix} 100 \\ 100 \end{bmatrix}, q_0 = \begin{bmatrix} 200 \\ 200 \end{bmatrix}, q_1 = \begin{bmatrix} 202 \\ 202 \end{bmatrix} \tag{2.1}$$

με παραμέτρους:

$$\rho_m = 5, \rho_M = 20, \rho_{step} = 1, N = 8 \tag{2.2}$$

και τα αποτελέσματα που δίνονται από το αρχείο είναι:

$$plain\_descriptor_p = \begin{bmatrix} 153.40625 & 152.5 & 151.625... \\ 167.15625 & 166.90625 & 159.71875... \\ 161.59375 & 158.0625 & 153.21875... \\ 149.96875 & 141.0625 & 130.21875... \\ 120.25 & 122.875 & 125.40625 \end{bmatrix} \tag{2.3}$$

$$plain\_descriptor_{q_0} = \begin{bmatrix} 85.84375 & 85.03125 & 84.03125... \\ 84.96875 & 94.21875 & 97.125... \\ 92.25 & 78.15625 & 80.78125... \\ 82.15625 & 83.1875 & 87.53125... \\ 84.6875 & 86.5625 & 83.15625 \end{bmatrix} \tag{2.4}$$

$$plain\_descriptor_{q_1} = \begin{bmatrix} 115.46875 & 100.8125 & 97.0625... \\ 95.84375 & 93.53125 & 80.1875... \\ 76.4375 & 74.5625 & 76.34375... \\ 78.46875 & 79. & 83.9375... \\ 86.03125 & 84.15625 & 78.53125 \end{bmatrix} \tag{2.5}$$

ενώ για τον αναβαθμισμένο περιγραφέα:

$$upgraded\_descriptor_{p} = \begin{bmatrix} 153.40625 & 111.75 \\ 152.5 & 109.25 \\ 151.625 & 109.25 \\ 167.15625 & 135. \\ 166.90625 & 96.5 \\ 159.71875 & 61. \\ 161.59375 & 67. \\ 158.0625 & 86.75 \\ 153.21875 & 101.75 \\ 149.96875 & 101.75 \\ 141.0625 & 86.5 \\ 130.21875 & 76. \\ 120.25 & 133.5 \\ 122.875 & 133.5 \\ 125.40625 & 160.5 \end{bmatrix} \tag{2.6}$$

$$upgraded\_descriptor_{q_0} = \begin{bmatrix} 85.84375 & 83.75 \\ 85.03125 & 92.5 \\ 84.03125 & 95.5 \\ 84.96875 & 90. \\ 94.21875 & 143. \\ 97.125 & 164. \\ 92.25 & 152. \\ 78.15625 & 110.25 \\ 80.78125 & 160.25 \\ 82.15625 & 163.25 \\ 83.1875 & 149. \\ 87.53125 & 202.25 \\ 84.6875 & 174.75 \\ 86.5625 & 138.75 \\ 83.15625 & 147.5 \end{bmatrix} \tag{2.7}$$

$$upgraded\_descriptor_{q_1} = \begin{bmatrix} 115.46875 & 105.5 \\ 100.8125 & 65.25 \\ 97.0625 & 70.25 \\ 95.84375 & 71. \\ 93.53125 & 101.25 \\ 80.1875 & 83.75 \\ 76.4375 & 86. \\ 74.5625 & 103. \\ 76.34375 & 116. \\ 78.46875 & 112. \\ 79. & 134. \\ 83.9375 & 180. \\ 86.03125 & 203. \\ 84.15625 & 181.5 \\ 78.53125 & 163.5 \end{bmatrix} \tag{2.8}$$

όπου η πρώτη στήλη περιέχει την ίδια πληροφορία με τον απλό περιγραφέα, και η

δεύτερη είναι η μέγιστη διαφορά φωτεινότητας στα σημεία εκείνου του κύκλου.

Κεφάλαιο 3

Corner Detection

3.1 Overview

Σε αυτό το τμήμα της εργασίας, ζητείται η ανάπτυξη αλγορίθμου για τον εντοπισμό εικονοστοιχείων γωνιών σε μια εικόνα.

3.2 Harris Corner Detector

Ο αλγόριθμος που μας προτείνεται-ζητείται είναι ο Harris Corner Detector. Η λειτουργία του και οι εξισώσεις που τον διέπουν περιγράφονται με λεπτομέρεια στην εκφώνηση της εργασίας [1] (παρέχεται επίσης και σχετικό paper). Επομένως, το μόνο που μένει είναι η κατασκευή του αλγορίθμου.

Οι συναρτήσεις που ζητούνται βρίσκονται στο αρχείο `corners.py`. Η σημαντικότερη συνάρτηση είναι η `isCorner`, η οποία εκτελεί έλεγχο για παρεχόμενο σημείο ή πίνακα σημείων σχετικά με το αν αυτά είναι σημεία γωνίας. Οι παράγωγοι υλοποιούνται με την χρήση μασκών, συγκεκριμένα με προσέγγιση `Sobel`:

$$Sobel_x = \begin{bmatrix} -1 & 0 & 1 \\ 2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$Sobel_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.2}$$

Πραγματοποιούμε συνέλιξη της εικόνας με καθεμιά από τις μάσκες για να λάβουμε τα επιθυμητά αποτελέσματα. Έπειτα κατασκευάζουμε την εξίσωση ελέγχου, και ελέγχουμε την τιμή της σχετικά με το παρεχόμενο κατώφλι.

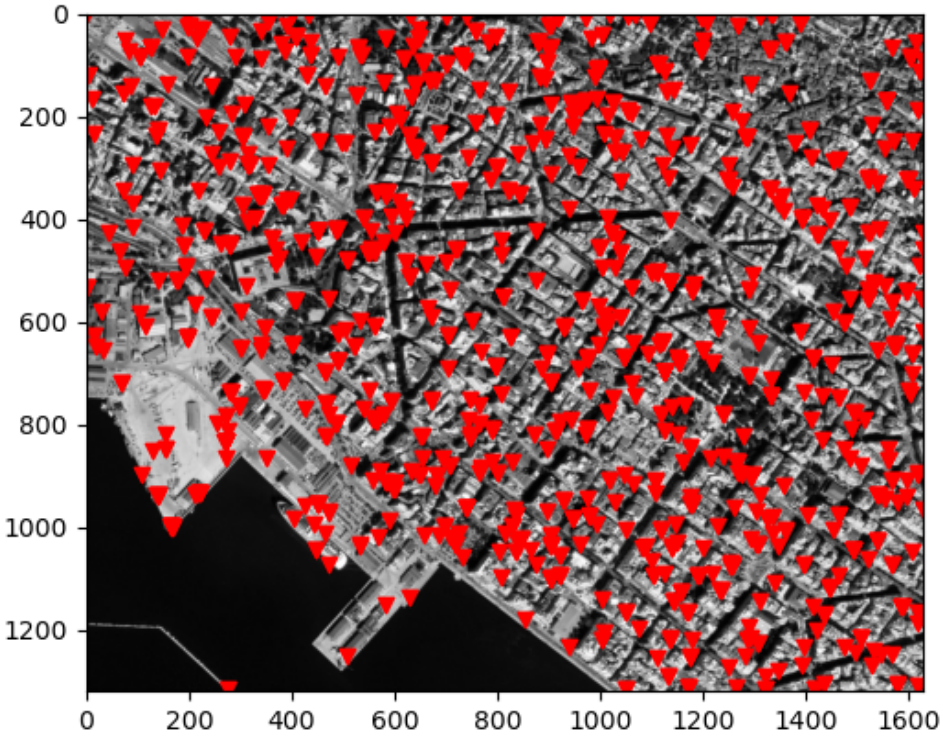
Η συνάρτηση `myDetectHarrisFeatures` πραγματοποιεί τον παραπάνω έλεγχο για κάθε σημείο της εικόνας και επιστρέφει τα σημεία τα οποία είναι γωνίες. Για την κατασκευή του πλέγματος συντεταγμένων όλων των σημείων, χρησιμοποιείται η παρακάτω γραμμή κώδικα:

```
p = np.array([[xt, yt] for xt in x for yt in y])
```

όπου `x, y` οι διαστάσεις της εικόνας.

3.3 Αποτελέσματα

Σαν επείδειξη, το πρόγραμα `demo_corners.py` εκτελεί τον Harris Corner Detector που αναπτύχθηκε στην εικόνα `im1.png`. Το αποτέλεσμα φαίνεται παρακάτω:



παρατηρούμε ότι σε γενικές γραμμές το πρόγραμμα εντοπίζει τις περισσότερες γωνίες. Όμως, κάποιες γωνίες δεν εντοπίζονται, και εντοπίζονται κάποια σημεία με πολύ αχνες γωνίες. Αυτό οφείλεται στην μη βέλτιστη επιλογή των παραμέτρων k, R_{thres} για την συγκεκριμένη εικόνα. Η βέλτιση επιλογή τους θα έπρεπε να γίνει με εκτεταμένη διερεύνηση και μέθοδο trial and error.

Κεφάλαιο 4

Matching Descriptors

4.1 Overview

Σε αυτό το τμήμα της εργασίας αναπτύσσονται αλγόριθμοι ταιριάσματος σημείων σε διαφορετικές εικόνες με βάση την ομοιότητα των τοπικών περιγραφέων τους. Επιπρόσθετα, αναπτύσσεται αλγόριθμος που εξάγει τον γραμμικό μετασχηματισμό μετατροπής από την μια εικόνα στην άλλη, με βάση την βέλτιστη αντιστοίχιση ταιριασμένων σημείων στις δύο εικόνες.

4.2 Point Matching

Αρχικά, δημιουργούμε έναν αλγόριθμο που δημιουργεί ζεύγη σημείων ανάμεσα σε 2 εικόνες, με βάση την ομοιότητα των περιγραφέων τους, και έπειτα διατηρεί τα καλύτερα από αυτά.

Ο αλγόριθμος αυτός υλοποιείται στο πρόγραμμα `matching_descriptors.py`, στην συνάρτηση `descriptorMatching`. Η συνάρτηση αυτή υπολογίζει την ευκλίδεια απόσταση των περιγραφέων όλων των συνδυασμών-ζεύγων σημείων της λίστας των σημείων της πρώτης εικόνας με την λίστα των σημείων της δεύτερης εικόνας. Για λόγους υπολογιστικής πολυπλοκότητας, υπολογίζεται το τετράγωνο της ευκλίδειας απόστασης (αποφεύγουμε την χρήση τετραγωνικής ρίζας). Έπειτα, από όλα αυτά τα ζεύγη, διατηρούνται τα καλύτερα, με κριτήριο ένα ποσοστό των συνολικών ζευγών που παρέχει ο χρήστης στην συνάρτηση.

4.3 Trasnform Derivation

Έπειτα, καλούμαστε με βάση τα ζεύγη των σημείων να εξάγουμε τον βέλτιστο γραμμικό μετασχηματισμό για την αντιστοίχιση των ζεύγων σημείων. Για να το πετύχουμε αυτό, χρησιμοποιούμε τον αλγόριθμο RANSAC, κατόπιν οδηγίας της ασκήσεως.

Για να εξάγουμε για κάθε 2 ζεύγη σημείων τον βέλτιστο γραμμικό μετασχηματισμό τους, χρησιμοποιούμε ανάλυση Προκρούστη [2]. Η μαθηματική αυτή ανάλυση μας επιτρέπει να εξάγουμε τόσο τον πίνακα περιστροφής R, όσο και το διάνυσμα μετατόπισης d ώστε να ισχύει:

$$p_2 = Rp_1 + d \tag{4.1}$$

όπου p_2 είναι οι συντεταγμένες των σημείων στην 2η, μετασχηματισμένη εικόνα και p_1 είναι οι συντεταγμένες των σημείων της αρχικής εικόνας.

Η μεθοδολογία υλοποιείται στην συνάρτηση `myRANSAC`, του ίδιου αρχείου. Πάλι για λόγους υπολογιστικής πολυπλοκότητας, το score για κάθε μετασχηματισμό δεν υπολογίζεται ως το άθροισμα των ευκλίδειων αποστάσεων των μετασχηματισμένων έναντι των ορθών σημείων, άλλα ως το άθροισμα των τετραγώνων των ευκλίδειων αποστάσεων.

Επιπρόσθετα, από τον πίνακα στροφής εξάγεται η γωνία περιστροφής θ με την χρήση της συνάρτησης $atan2$.

4.4 Αποτελέσματα

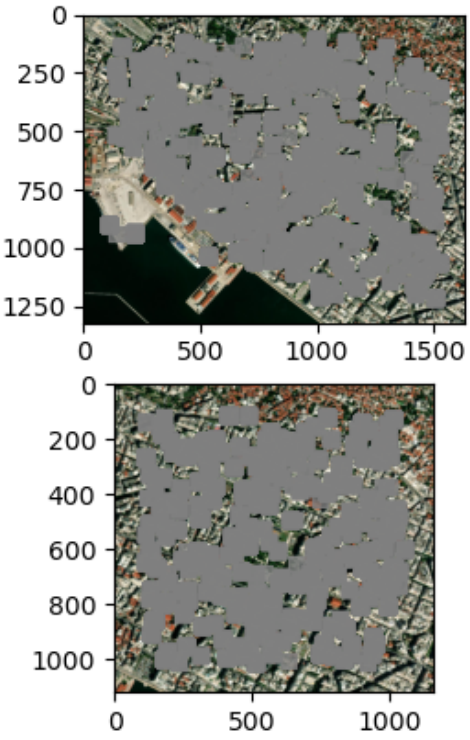
Το πρόγραμμα `demo_matching_descriptors.py` επιδεικνύει την λειτουργικότητα των προαναφερόμενων συναρτήσεων.

Αρχικά, φορτώνονται οι εικόνες της πόλης και εξάγονται οι γωνίες τους, με την χρήση της συνάρτησης `myDetectHarrisFeatures`. Έπειτα, εξάγονται τα καλύτερα 1% ζεύγη σημείων με την χρήση της συνάρτησης `descriptorMatching`. Έπειτα, εξάγεται ο βέλτιστος μετασχηματισμός με την χρήση της συνάρτησης `myRANSAC`. Τέλος, με βάση επιλογή ακτίνας σφάλματος, τα σημεία κατηγοριοποιούνται σε inliers και outliers.

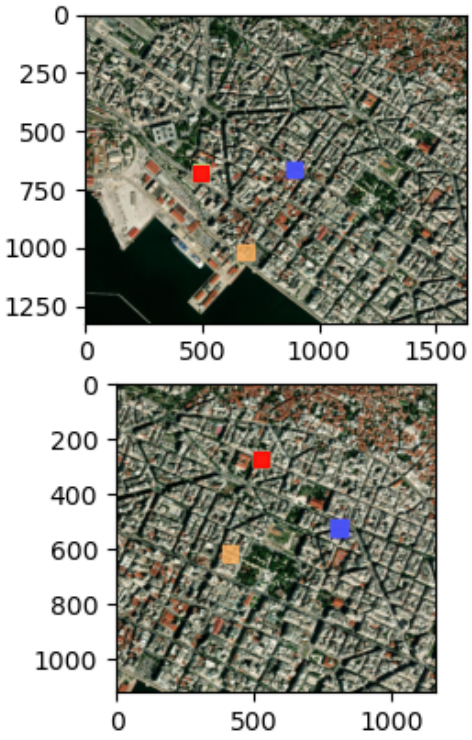
Ο βέλτιστος μετασχηματισμός που βρέθηκε για $N = 100$ επαναλήψεις είναι ο εξής:

$$\theta = -0.8149834188013733 rad, d = \begin{bmatrix} -560.95631879 \\ 683.13930231 \end{bmatrix} \tag{4.2}$$

Για $r = 25$ ακτίνα σφάλματος, τα σημεία outliers φαίνονται στην παρακάτω εικόνα:



ενώ τα inliers, color coded:



Παρατηρούμε πως η αντιστοίχιση δεν είναι βέλτιστη, κάτι που οφείλεται στη μη βέλτιστη επιλογή υπερπαραμέτρων για το πρόβλημα και στην χρήση ενός μη κατάλληλου περιγραφέα για τις εικόνες αυτές (πολύ πυκνή πληροφορία).

Κεφάλαιο 5

Το Πρόβλημα

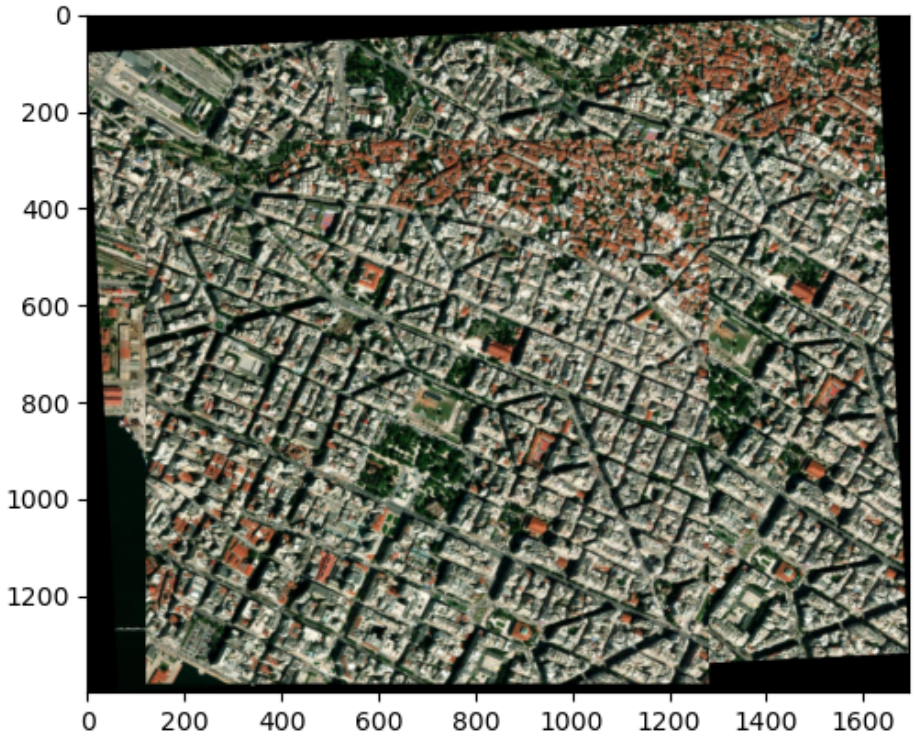
5.1 Περιγραφή

Το πρόβλημα που καλούμαστε να λύσουμε είναι η συρραφή αεροφωτογραφιών που τραβήχτηκαν διαδοχικά σε μια περιοχή, με βάση την αντιστοίχηση των ¨σημαντικών¨ τους σημείων.

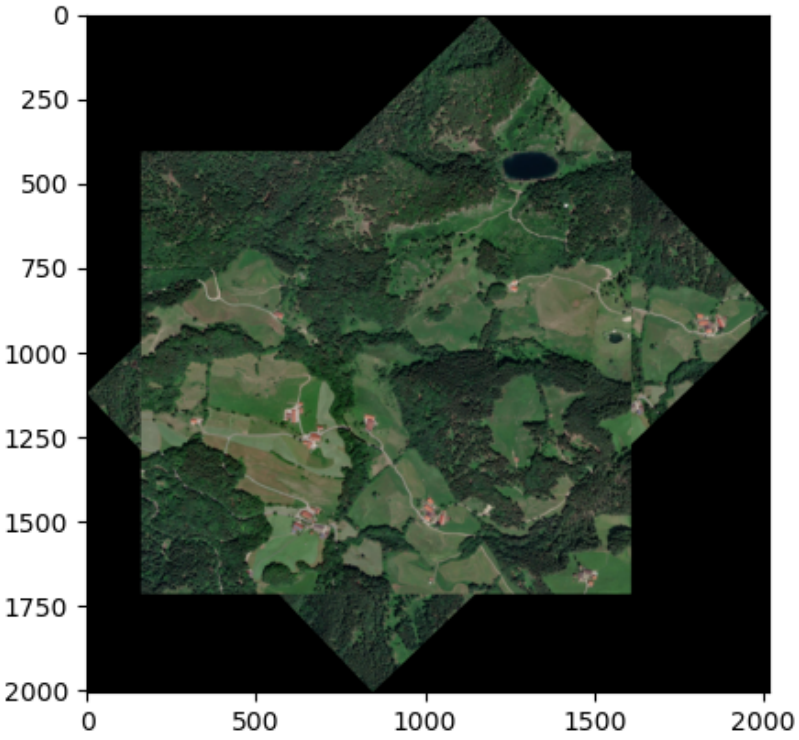
Για να το πετύχουμε αυτό, στο πρόγραμμα `demo.py` έχει υλοποιηθεί η συνάρτηση `imStitch`. Η συνάρτηση αυτή αρχικά εξάγει σημεία-γωνίες από τις εικόνες με την χρήση της συνάρτησης `myDetectHarrisFeatures`. Έπειτα, διατηρείται το 1% των πιο ¨όμοιων¨ ζεύγων σημείων, με την χρήση της συνάρτησης `descriptorMatching`. Από τα ζεύγη αυτά, εξάγεται ο βέλτιστος γραμμικός μετασχηματισμός με την χρήση της συνάρτησης `myRANSAC`. Στη συνέχεια, δημιουργείται ένας μαύρος καμβάς αρκετός για να κρατήσει την μετασχηματισμένη πρώτη εικόνα μαζί με την δεύτερη. Σε αυτόν τον καμβά, αρχικά μεταφέρεται στην κατάλληλη θέση η δεύτερη (μη μετασχηματισμένη) εικόνα. Έπειτα, στα κενά σημεία αντιστοιχίζονται όσα μετασχηματισμένα σημεία της πρώτης (άλλης) εικόνας είναι δυνατόν. Η τελική εικόνα επιστρέφεται στο τέλος της συνάρτησης.

5.2 Αποτελέσματα

Τα αποτελέσματα για τις φωτογραφίες της πόλης φαίνονται παρακάτω (εκτελώντας το πρόγραμμα `demo.py`):



ενώ για τις δασικές φωτογραφίες:



5.3 Σχόλια

Γενικά, παρατηρούμε ότι η συρραφή των εικόνων δεν είναι βέλτιστη, ή καν καλή. Αυτό οφείλεται σε διάφορους παράγοντες. Πέραν λοιπόν από την μη διερεύνηση για την βέλτιστη επιλογή υπερπαραμέτρων, υπάρχουν κι άλλες ατέλειες με την επιλεγμένη μεθοδολογία. Αρχικά, ο περιγραφέας δεν είναι βέλτιστος ειδικά για τις αστικές εικόνες, καθώς εντοπίζονται παντού πυκνά οικοδομικά τετράγωνα, οπότε θα δώσει ίδια ή παραπλήσια τιμή για πολλά διαφορετικά σημεία. Έπειτα, η επιλογή των σημείων-γωνιών ως salient points στην πρώτη φωτογραφία επίσης δεν είναι βέλτιστη, καθώς υπάρχουν πάρα πολλές και όμοιες γωνίες σε εκείνη την εικόνα. Αν αφαιρέσουμε χαρακτηριστικά όπως η θάλασσα του λιμανιού ή τα κόκκινα σπίτια στην πάνω δεξιά μεριά της εικόνας `im1.png`, πολύ πιθανόν να μπερδέψουμε ακόμα και άνθρωπο στην αντιστοίχιση των σημείων. Αυτός είναι και ο λόγος που στην δασική έκταση βλέπουμε οριακά καλύτερα αποτελέσματα (η συρραφή είναι πιο consistent).

Ενδεχομένως η επιλογή τόσο άλλου περιγραφέα που μπορεί να ανταποκριθεί στην περιπλοκότητα των εικόνων του προβλήματός μας, όσο και άλλης επιλογής salient points να έδινε πολύ καλύτερα αποτελέσματα.

Αν αλλάζαμε το ύψος λήψης φωτογραφίας, θα επηρρεάζαμε σημαντικά την λειτουργία του αναπτυγμένου περιγραφέα. Το βήμα και η ακτίνα των κύκλων συνδέεται πολύ στενά με την απόσταση μεταξύ σημείων στην εικόνα. Θεωρητικά αν γνωρίζαμε την διαφορά ύψους, καθώς και τις παραμέτρους της κάμερας, θα μπορούσαμε να βρούμε κατάλληλες παραμέτρους (ρ_m, rho_M, rho_{step}) ώστε οι περιγραφείς να αντιστοιχίζονται ορθά ανεξάρτητα από τα ύψη (τέτοιες μετατροπές εξερευνούμε στο μάθημα της γραφικής),

παρ' όλα αυτά πάλι δεν θα λαμβάναμε κάποιο ικανοποιητικό αποτέλεσμα. Επομένως, αν αλλάζαμε το ύψος λήψης φωτογραφίας ανάμεσα στην λήψη των φωτογραφιών, δεν θα μπορούσαμε να πετύχουμε ακόμα και την ποιότητα που πετυχαμε στις προηγούμενες δοκιμές.

Bibliography

[1] A. Delopoulos, *Digital Image Processing assignment 3: Image Registration*, 2023.

[2] "Procrustes analysis." [Online]. Available: https://en.wikipedia.org/wiki/Procrustes_analysis