

2η Εργασία Όραση Υπολογιστών Μπούρχα Ιωάννα 58019

Επιβλέπων καθηγητής: Ιωάννης Πρατικάκης
Επιβλέπων εργαστηρίου: Λάζαρος Τσοχατζίδης

Ακαδημαϊκό έτος: 2022 - 2023



ΠΕΡΙΕΧΟΜΕΝΑ

	σελ.
Πρόλογος	3
Εισαγωγή δεδομένων στον αλγόριθμο	5
Ανίχνευση σημείων ενδιαφέροντος & Προσδιορισμός περιγραφέων	5
Αλγόριθμος SIFT – Scale Invariant Feature Transform	6
Προσδιορισμός σημείων ενδιαφέροντος	6
Προσδιορισμός περιγραφέων	7
Αλγόριθμος SURF – Speeded Up Robust Features	8
Προσδιορισμός σημείων ενδιαφέροντος	8
Προσδιορισμός περιγραφέων	10
Αναγνώριση ομόλογων σημείων ενδιαφέροντος	12
Συνένωση δύο εικόνων	15
Σχολιασμός και παρατηρήσεις	18
Διακριτές γραμμές	18
Αποτέλεσμα με τον κώδικα του εργαστηρίου	18
Αποτέλεσμα με το εργαλείο Image Composite Editor	20

Πρόλογος

Στην παρούσα εργασία ζητείται η δημιουργία πανοράματος από 4 δοθείσες εικόνες.



rio-01



rio-02



rio-03



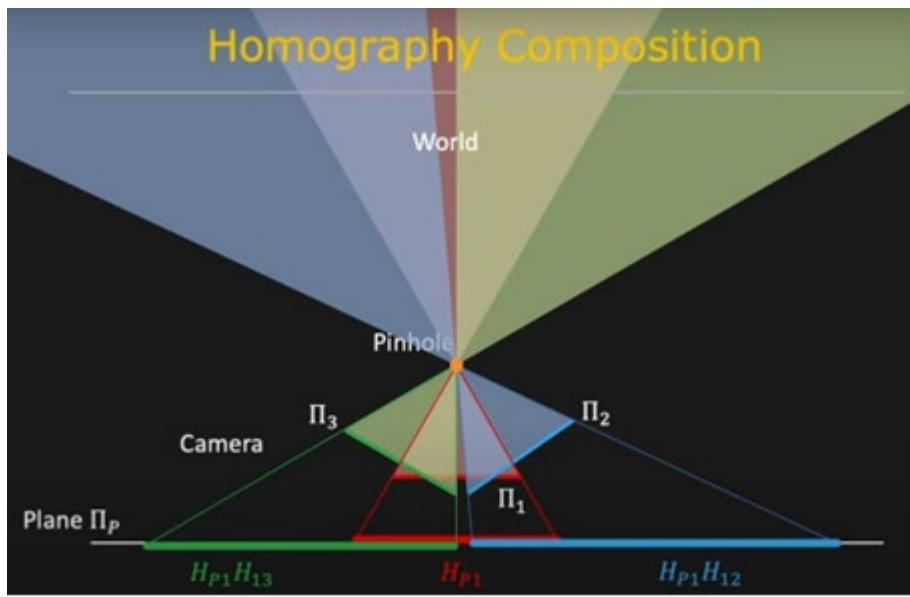
rio-04

Σύμφωνα και την υπόδειξη, επιλέγουμε να εργαστούμε με ζεύγη: πρώτα θα ενώσω τις εικόνες rio-01 και rio-02, έπειτα τις rio-03 και rio-04 και τέλος τα δύο προαναφερόμενα αποτελέσματα.

Για να επιτευχθεί η συνένωση δύο εικόνων, αυτές πρέπει να περιέχουν ορισμένα κοινά στοιχεία, ώστε να προσδιορίσω ποιες περιοχές θα αλληλοκαλυφτούν χωρίς να αλλοιώνεται το τελικό αποτέλεσμα. Ο προσδιορισμός αυτών των στοιχείων γίνεται με την σύγκριση των σημείων ενδιαφέροντος (key points) και των αντίστοιχων περιγραφέων (descriptors) κάθε εικόνας. Αυτά βρίσκονται με τους αλγόριθμους SIFT ή SURF.

Θεωρώντας ότι οι εικόνες έχουν κάποιο κοινό σημείο προβολής (Εικόνα 1, pinhole) προσδιορίζω τον πίνακα ομογραφίας (homography), δηλαδή τί μετασχηματισμούς θα πρέπει να υλοποιήσω στην μία εικόνα (εικόνα πηγής), ώστε να ταιριάζει με την άλλη (εικόνα προορισμού).

Θα συγκριθούν τα αποτελέσματα με την χρήση των αλγορίθμων SIFT και SURF, καθώς και με το εργαλείου Image Composite Editor.



Εικόνα 1: Συνθήκες φωτογράφισης

Η διαδικασία θα επαναληφθεί για 4 δικές μου εικόνες.



mine01



mine02



mine03



mine04

Εισαγωγή δεδομένων στον αλγόριθμο

Αρχικά, ξεκινάμε φορτώνοντας τις βιβλιοθήκες που θα χρησιμοποιήσουμε και τις εικόνες σε πίνακες και μορφή grayscale (ασπρόμαυρες).

```
import numpy as np
import cv2 as cv

" Import images in gray scale "

img1 = cv.imread('rio-01.png', cv.IMREAD_GRAYSCALE)
img2 = cv.imread('rio-02.png', cv.IMREAD_GRAYSCALE)
img3 = cv.imread('rio-03.png', cv.IMREAD_GRAYSCALE)
img4 = cv.imread('rio-04.png', cv.IMREAD_GRAYSCALE)
```

Η βιβλιοθήκη cv2 της OpenCV παρέχει ένα σύνολο εργαλείων για την επεξεργασία εικόνων υπό μορφή πινάκων.

Η βιβλιοθήκη numpy παρέχει σύνολο εργαλείων για επεξεργασία διανυσμάτων και πινάκων.

Η εντολή cv.imread αποθηκεύει στην μεταβλητή τον πίνακα που περιγράφει την αντίστοιχη εικόνα υπό μορφή grayscale λόγω του flag cv. IMREAD_GRAYSCALE.

Ανίχνευση σημείων ενδιαφέροντος & Προσδιορισμός περιγραφέων

Για να αναγνωρίσω τις κοινές περιοχές δύο εικόνων, συγκρίνω τα σημεία ενδιαφέροντος κάθε μίας και αποφασίζω αν υπάρχουν κοινά, ομόλογα ζευγάρια.

Ως σημείο ενδιαφέροντος (key point) θεωρείται ένα χαρακτηριστικό της εικόνας που μπορεί να ανιχνευτεί παρά τις αλλαγές ως προς τον φωτισμό, τον θόρυβο, την προοπτική και την κλίμακα. Το χαρακτηριστικό αυτό πρέπει να διακρίνεται ως προς την επαναληψιμότητα και όχι ως προς την πολλαπλότητα. Για παράδειγμα, η ακμή δεν θεωρείται καλή επιλογή σημείου ενδιαφέροντος καθώς αποτελείται από πολλά εικονοστοιχεία και μπορεί να εντοπιστεί πάρα πολλές φορές σε μία εικόνα. Αντίθετα, η γωνία αποτελείται από λιγότερα εικονοστοιχεία και μπορεί να εντοπιστεί αρκετές φορές σε μία εικόνα. Τόσο η ακμή όσο και η γωνία διακρίνονται από επαναληψιμότητα, αλλά η ακμή διακρίνεται και από έντονη πολλαπλότητα. Συνεπώς, ως σημεία ενδιαφέροντος επιλέγονται οι γωνίες.

Η σύγκριση των σημείων ενδιαφέροντος των δύο εικόνων δεν γίνεται ως προς τις συντεταγμένες αυτών στις δύο εικόνες, καθώς προφανώς δεν είναι ίδιες, αλλά ως προς τα τοπικά χαρακτηριστικά τους. Ως τοπικό χαρακτηριστικό στην παρούσα εργασία θεωρούμε την μεταβολή της διαβάθμισης του γκρι. Το σύνολο των τοπικών χαρακτηριστικών κάθε σημείου ενδιαφέροντος αποθηκεύεται σε ένα διάνυσμα που ονομάζεται περιγραφέας (descriptor).

Για τον προσδιορισμό των σημείων ενδιαφέροντος και των αντίστοιχων περιγραφέων χρησιμοποιώ τον αλγόριθμο SIFT ή SURF.

Αλγόριθμος SIFT – Scale Invariant Feature Transform

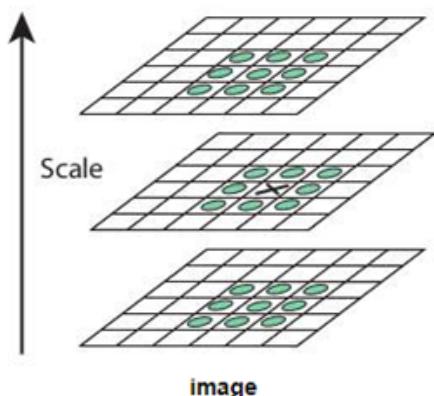
Προσδιορισμός σημείων ενδιαφέροντος

Τα σημεία ενδιαφέροντος λειτουργούν ως ταυτότητα του αντικειμένου, οπότε πρέπει να είναι αναλλοίωτα ως τους μετασχηματισμούς που μπορεί να υποστεί το αντικείμενο. Οι μετασχηματισμοί αυτοί είναι:

- Μετασχηματισμός φωτεινότητας
- Μετασχηματισμός περιστροφής
- Μετασχηματισμός μετατόπισης
- Μετασχηματισμός κλιμάκωσης

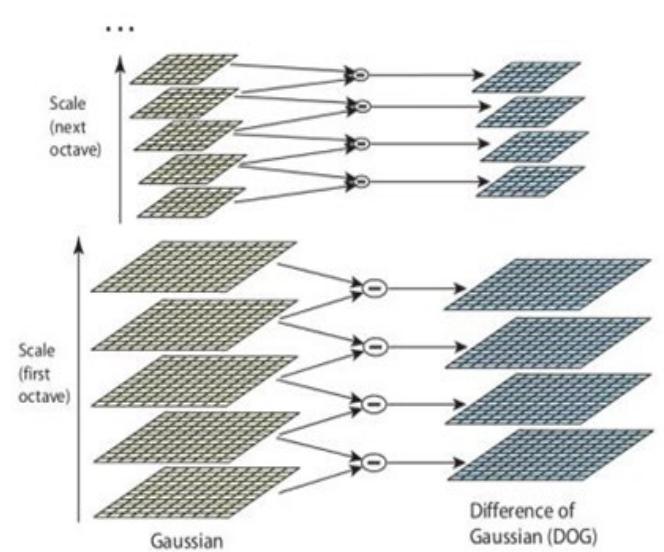
Όπως είδαμε και στο μάθημα, η γωνία παραμένει πρακτικά αναλλοίωτη ως προς τους παραπάνω μετασχηματισμούς με εξαίρεση τον τελευταίο.

Το βασικό, λοιπόν, πρόβλημα στον εντοπισμό των σημείων ενδιαφέροντος είναι ο μετασχηματισμός κλιμάκωσης. Η υλοποίηση αυτού του μετασχηματισμού ουσιαστικά οδηγεί σε πιο θολό αποτέλεσμα. Άρα, ο μετασχηματισμός αποκλιμάκωσης μπορεί να προσεγγιστεί με την με την εφαρμογή ενός φίλτρου Gauss διαφορετικού σ κάθε φορά ανάλογα με τον βαθμό της κλίμακας. Εν ολίγοις, το σ λειτουργεί ως παράμετρος κλιμάκωσης. Έτσι, για διαφορετικά σ δημιουργώ μια πυραμίδα που αποτελείται από την αρχική εικόνα και από εικόνες διαφορετικής κλίμακας (διαφορετικού σ).



Εικόνα 2: Δημιουργία πυραμίδας

Για τον προσδιορισμό των σημείων ενδιαφέροντος, αρχικά υπολογίζεται η διαφορά δύο εικόνων με διαφορετικές παραμέτρους κλιμάκωσης, έστω σ και σ' , (Difference of Gaussian, DoG) και έπειτα αναζητούνται τοπικά ακρότατα ως προς την κλίμακα και τον χώρο.



Εικόνα 3: Υπολογισμός DoG

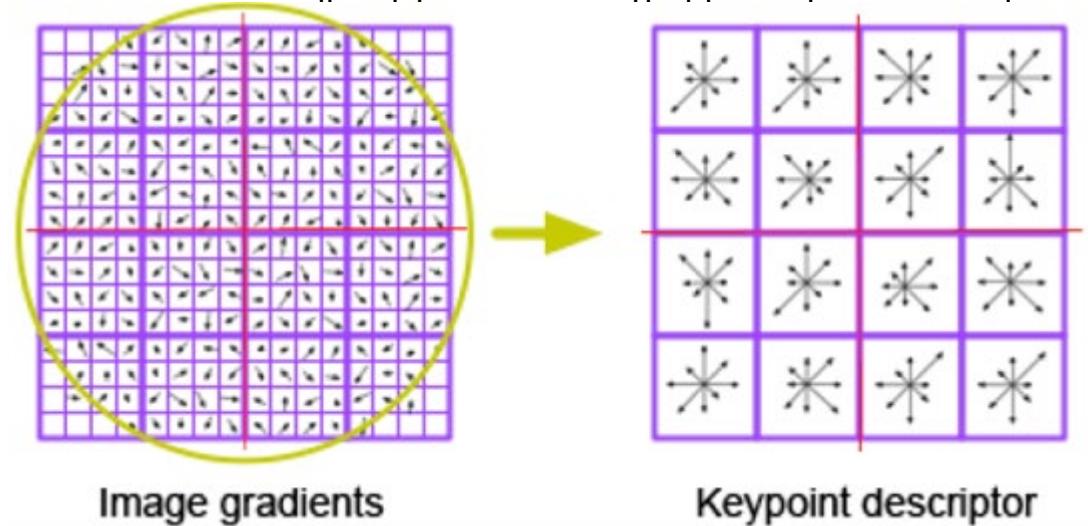
Έστω εικονοστοιχείο σε κλίμακα σ με 8 γείτονες (Εικόνα 2). Αυτό συγκρίνεται με:

- τους γείτονές τους (Harris corner detector)
- 9 εικονοστοιχεία της εικόνας με την αμέσως μεγαλύτερη κλίμακα $\sigma + d\sigma$
- 9 εικονοστοιχεία της εικόνας με την αμέσως μικρότερη κλίμακα $\sigma - d\sigma$

Εάν σε αυτό παρουσιάζεται τοπικό ακρότατο, τότε μπορεί να είναι και σημείο ενδιαφέροντος. Για να αποφανθούμε ότι ένα τοπικό ακρότατο είναι όντως σημείο ενδιαφέροντος χρησιμοποιείται η σειρά Taylor με επέκταση ως προς την κλίμακα. Εάν η τιμή είναι μεγαλύτερη από ένα όριο (edge threshold, 10), τότε είναι όντως σημείο ενδιαφέροντος, διαφορετικά απορρίπτεται.

Προσδιορισμός περιγραφέα

Εφόσον έχουν προσδιοριστεί τα σημεία ενδιαφέροντος, ορίζεται γύρω από κάθε ένα μία περιοχή διαστάσεων 16×16 . Έκαστη διαιρείται σε 16 μη αλληλοκαλυπτόμενα μπλοκ διαστάσεων 4×4 το κάθε ένα. Για κάθε ένα δημιουργείται ένα ιστόγραμμα 8 προσανατολισμών.



Εικόνα 4: Δημιουργία ιστογράμματος

Τελικά, καταλήγω με $8 * 16 = 128$ προσανατολισμούς. Άρα, το διάνυσμα του περιγραφέα κάθε σημείου ενδιαφέροντος αποτελείται από 128 στοιχεία των οποίων οι τιμές είναι αυτές των προσανατολισμών.

Σύμφωνα με μελέτες, η ανάμειξη ακατέργαστων εντοπισμένων πληροφοριών και η κατανομή των χαρακτηριστικών που σχετίζονται με την κλίση αποδίδει καλή διακριτική ισχύ, ενώ αποτρέπει τις επιπτώσεις των σφαλμάτων εντοπισμού αναφορικά με την κλίμακα ή τον χώρο. Επιπλέον, η χρήση σχετικών δυνάμεων και προσανατολισμών των κλίσεων μειώνει την επίδραση των φωτομετρικών αλλαγών. Τις ιδιότητες αυτές αξιοποιεί ο περιγραφέας SIFT.

Ο αλγόριθμος αυτός είναι πολύ σημαντικός καθώς αντιμετωπίζει το πρόβλημα της κλίμακας για τον εντοπισμό των σημείων ενδιαφέροντος, αλλά απαιτεί μεγάλη πολυπλοκότητα για την εκπόνησή του, οπότε είναι ιδιαίτερα αργός.

Ο προσδιορισμός της χρήσης του αλγορίθμου SIFT στον κώδικα γίνεται με την εντολή:

```
type = cv.xfeatures2d_SIFT.create(1000)
```

Τα σημεία ενδιαφέροντος και οι περιγραφέις τους εντοπίζονται με τις εντολές:

```

# KEY POINTS + DESCRIPTOR FOR IMAGE A
kpA = type.detect(imgA)
dA = type.compute(imgA, kpA)

# KEY POINTS + DESCRIPTOR FOR IMAGE B
kpB = type.detect(imgB)
dB = type.compute(imgB, kpB)

```

Αλγόριθμος SURF – Speeded Up Robust Features

Προσδιορισμός σημείων ενδιαφέροντος

Για τον εντοπισμό των σημείων ενδιαφέροντος (γωνίες) επιλέγουμε τον ανιχνευτή γωνιών κατά Harris, ο οποίος όμως επηρεάζεται σημαντικά από την κλίμακα. Στην προσπάθεια αντιμετώπισης του προβλήματος του μετασχηματισμού κλιμάκωσης ο Linberg, εισήγαγε την έννοια της αυτόματης επιλογής κλίμακας. Έτσι, γίνεται εφικτός ο εντοπισμός των σημείων ενδιαφέροντος της εικόνας με τρόπο ώστε να κάθε ένα να έχει και την κλίμακά του.

Για τον προσδιορισμό των σημείων ενδιαφέροντος επιχείρησε να ανιχνεύσει δομές που μοιάζουν με σταγόνες. Ο προσδιορισμός της τοποθεσίας γίνεται με πίνακα Hessian δύο διαστάσεων και της κλίμακας με το φίλτρο Laplace. Επιδιώκοντας την μέγιστη δυνατή ταχύτητα, το φίλτρο Laplace προσεγγίστηκε, όπως και προηγουμένως, με την διαφορά δύο εικόνων όπου έχει εφαρμοστεί το φίλτρο Gauss με διαφορετικές τιμές του παράγοντα σ.

Ο ανιχνευτής χαρακτηριστικών SURF λειτουργεί εφαρμόζοντας σε κάθε εικονοστοιχείο της εικόνας την συνέλιξη του φίλτρου Gauss δύο διαστάσεων σε πολλές κλίμακες με την μερική παράγωγο. Έτσι, ορίζεται ο πίνακας Hessian δύο διαστάσεων (Εικόνα 5). Εφόσον, ο ανιχνευτής χαρακτηριστικών εφαρμόζει μάσκες κατά μήκος κάθε άξονα x και y (ολικές παράγωγοι) και στις 45 μοίρες για κάθε άξονα (μερικές παράγωγοι), είναι πιο ανθεκτικός στον μετασχηματισμό κλίμακας από τον ανιχνευτή γωνιών Harris, όπως ακριβώς έχουν δείξει και πολλές μελέτες.

$$f : R^n \rightarrow R$$

For $f(x,y)$:

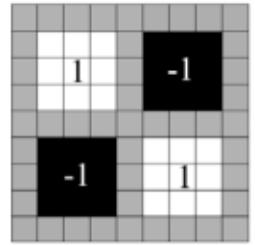
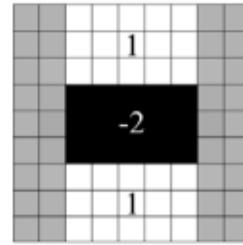
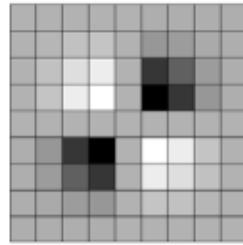
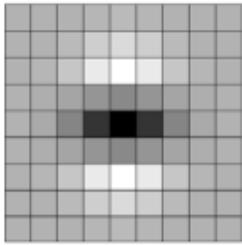
$$H_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & & & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

$$f : R^2 \rightarrow R$$

$$H_{(f(x,y))} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix}$$

Eikόνα 5: Πίνακας Hessian

Μελέτες έχουν υποδείξει ότι η χρήση του φίλτρου Gauss δεν είναι η ιδανικότερη, καθώς στην περίπτωση της υποδειγματοληψίας οι αλλοιώσεις εξακολουθούν να επηρεάζουν το αποτέλεσμα. Για την αντιμετώπιση αυτού του προβλήματος έχει υιοθετηθεί η χρήση του φίλτρου κουτιού (box filter) αντί του φίλτρου Gauss..



Εικόνα 6: Σε συνθήκη υποδειγματοληψίας και μεγάλης αποκλιμάκωσης παρουσιάζονται τα αποτέλεσματα για τον εντοπισμό σημείων ενδιαφέροντος με την χρήση του φίλτρου Gauss (αριστερές εικόνες) και του φίλτρου κουτιού (δεξιές εικόνες).

Επομένως, το φίλτρο Gauss δεύτερης τάξης μπορεί να προσεγγιστεί με το φίλτρο κουτιού. Δεδομένου του πυρήνα ενός τέτοιου φίλτρου (Εικόνα 7), για τον υπολογισμό της τιμής κάθε εικονοστοιχείου στο οποίο εφαρμόζεται μπορούμε να αξιοποιήσουμε την εικόνα ολοκλήρωσης (integral image) που υλοποιήσαμε και στην προηγούμενη εργασία. Η τιμή ενός εικονοστοιχείου (x,y) είναι το άθροισμα όλων των τιμών στο ορθογώνιο που ορίζονται από την αρχή, συνήθως πάνω αριστερά γωνία του πίνακα της εικόνας, μέχρι το εικονοστοιχείο αυτό. Το άθροισμα των εικονοστοιχείων μέσα σε ένα ορθογώνιο οποιουδήποτε μεγέθους προκύπτει από το αποτέλεσμα 4 λειτουργιών. Αυτό επιτρέπει την εφαρμογή μιας ορθογώνιας μάσκας οποιουδήποτε μεγέθους με πολύ μικρό χρόνο υπολογισμού. Άρα, μπορούμε να πετύχουμε υψηλότερη ταχύτητα ανεξάρτητα με το μέγεθος του πυρήνα.

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Εικόνα 7: Ο πυρήνας του φίλτρου κουτιού διαστάσεων 3X3

Συνοψίζοντας, για την αντιμετώπιση του μετασχηματισμού αποκλιμάκωσης επιλέγω την εφαρμογή ενός φίλτρου κουτιού στην αρχική εικόνα και την χρήση της ολοκληρωμένης εικόνας για τον υπολογισμό της τελικής τιμής. Το αποτέλεσμα διαφορετικής κλίμακας προσεγγίζεται με την χρήση μάσκας διαφορετικών διαστάσεων για το φίλτρο. Έτσι, δεν χρειάζεται να εξομαλύνουμε την αρχική εικόνα με χρήση ενός φίλτρου Gauss για διαφορετικές τιμές του σ' όπως στον αλγόριθμο SIFT. Ως αρχικό στρώμα θεωρείται η έξοδος του φίλτρου για μάσκα διαστάσεων 9X9 και αντιστοιχεί στην εφαρμογή φίλτρου Gauss με $\sigma = 1,2$. Τα υπόλοιπα επίπεδα προκύπτουν φιλτράροντας την εικόνα με σταδιακά μεγαλύτερες μάσκες, λαμβάνοντας υπόψη τη διακριτή φύση των ενσωματωμένων εικόνων και τη συγκεκριμένη δομή των φίλτρων μας.

Ο εντοπισμός των σημείων ενδιαφέροντος στα διάφορα επίπεδα της πυραμίδας γίνεται εφαρμόζοντας μη μέγιστη καταστολή σε μία περιοχή διαστάσεων 3X3X3. Τα μέγιστα της ορίζουσας του πίνακα Hessian για την περιοχή αυτή παρεμβάλλονται ως προς την κλίμακα και τον χώρο. Εάν ένα σημείο παρουσιάζει μέγιστο σε πολλές κλίμακες, τότε είναι σημείο ενδιαφέροντος.

Προσδιορισμός περιγραφέα

Ο περιγραφέας SURF βασίζεται σε παρόμοιες ιδιότητες με αυτές του περιγραφέα SIFT, αλλά με πιο προσδιορισμένη πολυπλοκότητα.

Αρχικά, καθορίζεται ένα αναπαραγώγιμο του προσανατολισμού το οποίο βασίζεται στις πληροφορίες μιας κυκλικής περιοχής γύρω από το σημείο ενδιαφέροντος. Στη συνέχεια, κατασκευάζουμε μια τετράγωνη περιοχή ευθυγραμμισμένη με τον επιλεγμένο προσανατολισμό από την οποία και εξάγουμε τον περιγραφέα SURF.

Προκειμένου ο περιγραφέας να είναι αμετάβλητος ως προς τον μετασχηματισμό περιστροφής, προσδιορίζουμε έναν αναπαραγόμενο του προσανατολισμό για τα σημεία ενδιαφέροντος. Για το σκοπό αυτό, υπολογίζουμε πρώτα τις αποκρίσεις κυματιδίων Haar σε κατεύθυνση x και y σε μια κυκλική γειτονιά γύρω από το σημείο ενδιαφέροντος. Η ακτίνα της κυκλικής περιοχής είναι εξαπλάσια της κλίμακας στην οποία εντοπίστηκε το σημείο ενδιαφέροντος.



Εικόνα 8: Αποκρίσεις κυματιδίων Haar

έχει σχεδιαστεί για να είναι αμετάβλητος ως προς τον μετασχηματισμό της κλίμακας και της περιστροφής. Για να αγνοηθεί η κλίμακα, γίνεται δειγματοληψία του περιγραφέα σε ένα παράθυρο ανάλογο με την μάσκα για την οποία εντοπίστηκε το σημείο ενδιαφέροντος και παρουσίασε μέγιστο ως προς τον χώρο. Με αυτόν τον τρόπο, σε μια κλιμακωμένη έκδοση αυτού του σημείου ενδιαφέροντος ο περιγραφέας του θα δειγματιστεί στην ίδια σχετική περιοχή.

Για την εξαγωγή του περιγραφέα κατασκευάζω με κέντρο το σημείο ενδιαφέροντος μια τετράγωνη περιοχή η οποία έχει τον προσανατολισμό που αναλύθηκε προηγουμένως.



Εικόνα 9: Τα τετράγωνα είναι τα παράθυρα που χρησιμοποιούνται για τον προσδιορισμό των περιγραφέων

Έπειτα, η περιοχή χωρίζεται σε 16 υποτετράγωνα και έκαστο χωρίζεται πάλι σε 4 τετράγωνα, όπως ακριβώς και για τον SIFT, όπου υπολογίζονται μερικά απλά χαρακτηριστικά ως προς την "οριζόντια" και την "κάθετη" διεύθυνση. Για λόγους απλότητας, ονομάζουμε dx την απόκριση

κυματιδίου Haar σε οριζόντια κατεύθυνση και δυ την απόκριση κυματιδίου Haar σε κάθετη κατεύθυνση (μέγεθος φίλτρου 2s). Το "οριζόντιο" και το "κάθετο" ορίζονται σε σχέση με τον επιλεγμένο προσανατολισμό του σημείου ενδιαφέροντος. Οι αποκρίσεις των κυματιδίων στις δύο κατευθύνσεις αθροίζονται δημιουργώντας ένα σύνολο τιμών για τον περιγραφέα. Προκειμένου να αξιοποιήσουμε και την πληροφορία από την μεταβολή της έντασης σε αυτές τις κατευθύνσεις στο προαναφερόμενο σύνολο τιμών εισάγεται και το άθροισμα των απόλυτων τιμών των αποκρίσεων.

$$v_{subregion} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy| \right]$$

Εικόνα 10: Μαθηματική σχέση για τον προσδιορισμό του περιγραφέα κατά SURF

Συνεπώς, κάθε υποπεριοχή προσδιορίζεται από ένα τετραδιάστατο διάνυσμα. Οπότε το διάνυσμα του περιγραφέα για ένα σημείο ενδιαφέροντος έχει $16 * 4 = 64$ διαστάσεις.

Ο προσδιορισμός της χρήσης του αλγορίθμου SIFT στον κώδικα γίνεται με την εντολή:

```
type = cv.xfeatures2d_SURF.create(1000)
```

Τα σημεία ενδιαφέροντος και οι περιγραφές τους εντοπίζονται με τις ίδιες εντολές όπως και προηγουμένως:

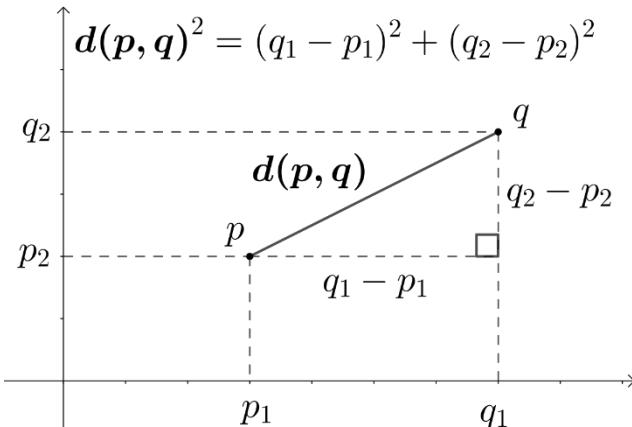
```
print('Finding key points and descriptors ...')

# KEY POINTS + DESCRIPTOR FOR IMAGE A
kpA = type.detect(imgA)
dA = type.compute(imgA, kpA)

# KEY POINTS + DESCRIPTOR FOR IMAGE B
kpB = type.detect(imgB)
dB = type.compute(imgB, kpB)
```

Αναγνώριση ομόλογων σημείων ενδιαφέροντος

Προκειμένου να συνενωθούν δύο εικόνες πρέπει να προσδιοριστούν οι κοινές περιοχές. Αυτό πραγματοποιείται με την σύγκριση των τοπικών χαρακτηριστικών, δηλαδή των περιγραφέων κάθε σημείου ενδιαφέροντος της μία εικόνας με το σύνολο των περιγραφέων της δεύτερης. Η σύγκριση γίνεται αξιοποιώντας την ευκλείδεια απόσταση για τα δύο διανύσματα. Ένα σημείο ενδιαφέροντος της εικόνας πηγής θεωρείται ομόλογο με εκείνο το σημείο ενδιαφέροντος της εικόνας προορισμού για το οποίο η διαφορά των περιγραφέων έδωσε την μικρότερη τιμή (απλό ταίριασμα).



Εικόνα 11: Τύπος Ευκλείδειας απόστασης

Έτσι, κάθε σημείο ενδιαφέροντος της εικόνας πηγής αντιστοιχίζεται υποχρεωτικά με κάποιο σημείο ενδιαφέροντος της εικόνας προορισμού. Αυτό, όμως, δεν οδηγεί πάντα σε σωστά αποτελέσματα, καθώς δεν είναι υποχρεωτικό οι δύο εικόνες να αποτελούνται μόνο από κοινές περιοχές. Για την βελτιστοποίηση της μεθοδολογίας μας, την εφαρμόζουμε δύο φορές:

- μία φορά θεωρώντας ως εικόνα πηγής την imgA & εικόνα προορισμού την imgB
- μία φορά θεωρώντας ως εικόνα πηγής την imgB & εικόνα προορισμού την imgA.

Ουσιαστικά, ανταλλάσσουμε τους όρους του αφαιρέτη και του αφαιρετέου στον υπολογισμό της απόστασης. Ως ομόλογα σημεία ενδιαφέροντος θεωρούμε αυτά των ζευγών που προέκυψαν και από τις δύο περιπτώσεις (αμφίδρομο ταίριασμα.)

Η μεθοδολογία αυτή υλοποιήθηκε καλώντας δύο φορές την συνάρτηση match2 που αναλύθηκε στο εργαστήριο.

Απλό ταίριασμα:

```
def match2(d1, d2):          # LAB CODE
    n1 = d1.shape[0]

    matches = []
    for i in range(n1):
        fv = d1[i, :]
        diff = d2 - fv
        diff = np.abs(diff)
        distances = np.sum(diff, axis=1)

        i2 = np.argmin(distances)
        mindist2 = distances[i2]

        # change the value of the minimum distance to infinity in order to tack the second minimum distance next time
        distances[i2] = np.inf

        i3 = np.argmin(distances)
```

```

mindist3 = distances[i3]

# GOOD MATCHING
if mindist2 / mindist3 < 0.5:
    matches.append(cv.DMatch(i, i2, mindist2))

return matches

```

Αμφίδρομο ταίριασμα:

```

matchesAB = match2(dA[1], dB[1])
matchesBA = match2(dB[1], dA[1])

true_matches = [m for m in matchesAB for n in matchesBA if m.distance == n.distance]

img_ptA = []
img_ptB = []
for x in true_matches:
    img_ptA.append( kpA[x.queryIdx].pt )
    img_ptB.append( kpB[x.trainIdx].pt )

```

Για να ζωγραφίσω τα ομόλογα σημεία χρησιμοποιώ την συνάρτηση drawMatches της OpenCV.

```

print ('Drawing Matches ...')
dimg = cv.drawMatches(imgA, dA[0], imgB, dB[0], true_matches, None)
cv.namedWindow('Crosschecking', cv.WINDOW_NORMAL)
cv.imshow('Crosschecking', dimg)
cv.waitKey(0)

```

Συνολικά, τα σημεία ενδιαφέροντος και οι αντίστοιχοι περιγραφείς ανάλογα με τον αλγόριθμο που επιθυμούμε κάθε φορά, ο προσδιορισμός των ομόλογων σημείων ενδιαφέροντος και η απεικόνιση αυτών πραγματοποιείται με την κλήση της συνάρτησης Keypoints_Detectors.

```

def Keypoints_Detectors (imgA, imgB, algo):
    if algo == "sift":
        type = cv.xfeatures2d_SIFT.create(1000)
    elif algo == "surf":
        type = cv.xfeatures2d_SURF.create(1000)

    print('Finding key points and descriptors ...')

    # KEY POINTS + DESCRIPTOR FOR IMAGE A
    kpA = type.detect(imgA)
    dA = type.compute(imgA, kpA)

    # KEY POINTS + DESCRIPTOR FOR IMAGE B
    kpB = type.detect(imgB)
    dB = type.compute(imgB, kpB)

    print('Finding true matches ...')

    matchesAB = match2(dA[1], dB[1])
    matchesBA = match2(dB[1], dA[1])

    true_matches = [m for m in matchesAB for n in matchesBA if m.distance == n.distance]

    img_ptA = []
    img_ptB = []
    for x in true_matches:
        img_ptA.append( kpA[x.queryIdx].pt )
        img_ptB.append( kpB[x.trainIdx].pt )

    img_ptA = np.array(img_ptA)
    img_ptB = np.array(img_ptB)

```

```

print ('Drawing Matches ...')
dimg = cv.drawMatches(imgA, dA[0], imgB, dB[0], true_matches, None)
cv.namedWindow('Crosschecking', cv.WINDOW_NORMAL)
cv.imshow('Crosschecking', dimg)
cv.waitKey(0)

return img_ptA, img_ptB

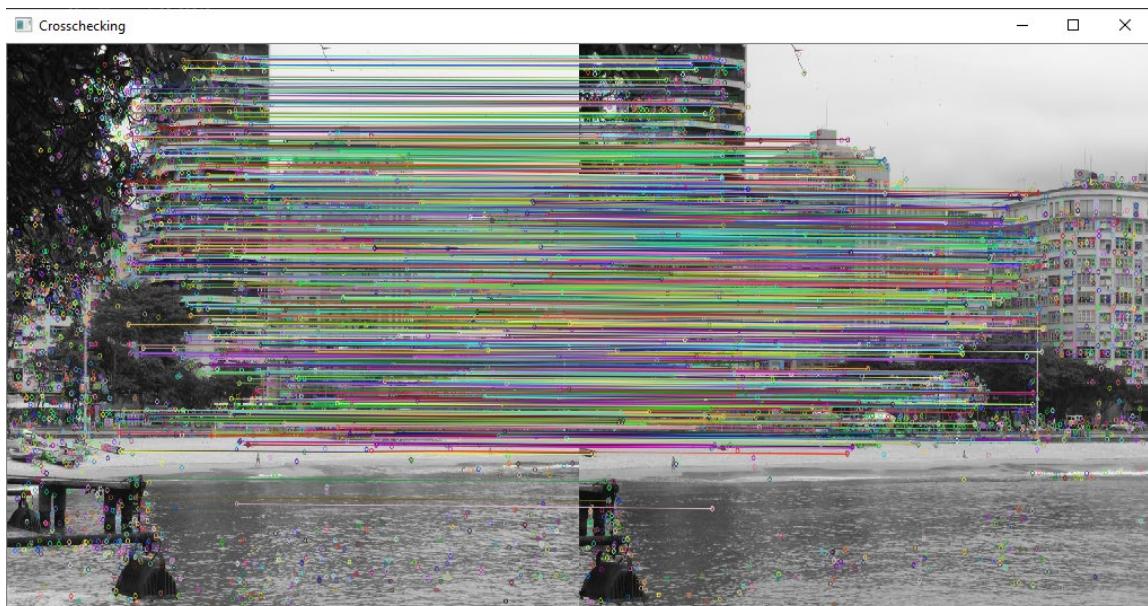
```

Η συνάρτηση αυτή επιστρέφει τα ομόλογα σημεία ενδιαφέροντος υπό μορφή numpy array, καθώς αυτή απαιτείται για τον προσδιορισμό του πίνακα ομογραφίας.

Ως χαρακτηριστικό παράδειγμα παρουσιάζω τον εντοπισμό των ομόλογων σημείων για την συνένωση των εικόνων rio-01 και rio-02 με την χρήση των δύο αλγορίθμων.



Εικόνα 12: Εντοπισμός ομόλογων σημείων με SIFT για την συνένωση των εικόνων rio-01 και rio-02



Εικόνα 13: Εντοπισμός ομόλογων σημείων με SURF για την συνένωση των εικόνων rio-01 και rio-02

Παρατηρώ ότι πράγματι η διαδικασία με την χρήση του αλγόριθμου SURF ολοκληρώνεται αισθητά πιο γρήγορα σε σχέση εκείνη του αλγόριθμου SIFT, ακριβώς όπως υποδεικνύει και η θεωρία.

Συνένωση δύο εικόνων

Εφόσον έχω εντοπιστεί τα ομόλογα σημεία ενδιαφέροντος, δηλαδή τις κοινές περιοχές στις δύο εικόνες, προσδιορίζω το σύνολο των μετασχηματισμών που θα πρέπει να εφαρμόσω στην μία ώστε να ταιριάξει στην άλλη. Η διαδικασία αυτή ονομάζεται ομογραφία και υλοποιείται με την ακόλουθη συνάρτηση:

```
def homography(A, B, algo):
    " HOMOGRAPHY " # Βρίσκει πώς πρέπει να μετασχηματιστεί η μία εικόνα για να "ταιριάξει" με την άλλη
    img_ptA, img_ptB = Keypoints_Detectors(A, B, algo)
    H, mask = cv.findHomography(img_ptB, img_ptA, cv.RANSAC)
    return H
```

Προκειμένου το αποτέλεσμα του κώδικα μου να είναι παρόμοιας μορφής με αυτό που παράγει το κινητό, επιλέγω κατά την συνένωση των δύο πρώτων εικόνων να εφαρμόσω τους απαραίτητους μετασχηματισμούς στην rio-01 έναντι στην rio-02. Η αλλαγή αυτή απαιτεί εφαρμογή επιπλέον μετασχηματισμού μετατόπισης, ώστε κατά την συνένωση του αποτελέσματος με την εικόνα rio-02 να οδηγεί σε σωστό αποτέλεσμα.

```
# Κρατάει σταθερή την εικόνα B και μετασχηματίζει την εικόνα A
H = homography(imgB, imgA, algo) # σταθερό, αλλάζει

translation = int( max(imgA.shape) )
T = np.array([[1, 0, translation], [0, 1, 0], [0, 0, 1]])

# Wrap destination image to source
result = cv.warpPerspective(imgA, np.matmul(T, H), (translation*2, translation*2))
result[0: imgB.shape[0], translation: translation + imgB.shape[1]] = imgB
```



Εικόνα 14: Το αποτέλεσμα μετά την συνένωση των δύο πρώτων εικόνων

Το αποτέλεσμα του κώδικα για την συνένωση των δύο πρώτων εικόνων φαίνεται στην Εικόνα 14. Παρατηρώ ότι λόγω των μετασχηματισμών, το μέγεθος της εικόνας μου έχει αυξηθεί αισθητά με την εισαγωγή μαύρων εικονοστοιχείων, τα οποία δεν εισάγουν κάποια πληροφορία και πρέπει να αφαιρεθούν. Για την αποκοπή τους χρησιμοποιώ την συνάρτηση connectedComponents της OpenCV η οποία χρησιμοποιήθηκε και στην προηγούμενη εργασία.

```
def crop1(img):
    # ----- Binary Image ----- #
    _, img_binary = cv.threshold(img, 1, 255, cv.THRESH_BINARY)
    # cv.imshow('Binary Image', img_binary)
    # cv.waitKey(0)

    eik = img_binary

    # Returns the number of found components in num and an array with the value of each
    # component (e.g. 1,2,...,10) for each white pixel
    num, val_pix = cv.connectedComponents(eik)
    for i in range(1, num):
        counter = 0
        regions = np.zeros(eik.shape, dtype=np.uint8)
        regions[val_pix == i] = 255
        x, y, w, h = cv.boundingRect(regions)

    # Crop the space
    cropped_img = img[y:y + h, x:x + w]

    return cropped_img
```

Το αποτέλεσμα μετά την αποκοπή φαίνεται στην Εικόνα 15.



Εικόνα 15: Αποτέλεσμα της συνένωσης των πρώτων δύο εικόνων μετά την αποκοπή των περιττών γραμμών και στηλών

Για την συνένωση των εικόνων rio-03 και rio-04 θα πρέπει να εφαρμόσω τον πίνακα ομογραφίας στην εικόνα rio-04, οπότε χρησιμοποιώ τον κώδικα του εργαστηρίου. Δεν απαιτείται επιπλέον μετασχηματισμός μετατόπισης. Όπως και πριν, λόγω των μετασχηματισμών για την επίτευξη της ομογραφίας, εισάγονται επιπλέον γραμμές και στήλες που δεν περιέχουν πληροφορία, οπότε και αφαιρούνται με την κλήση της crop1.

```
# Κρατάει σταθερή την εικόνα A και μετασχηματίζει την εικόνα B

H = homography(imgA, imgB, algo) # σταθερό, αλλάζει
# Wrap destination image to source based on homography
result = cv.warpPerspective(imgB, H, (imgA.shape[1] + 1000, imgA.shape[0] + 1000)) #
source, H, (destination)
cv.namedWindow('result', cv.WINDOW_NORMAL) # imgB after homography
```

```

cv.imshow('result', result)
cv.waitKey(0)

result[0: imgA.shape[0], 0: imgA.shape[1]] = imgA
cv.namedWindow('result', cv.WINDOW_NORMAL) # imgB after homography
cv.imshow('result', result)
cv.waitKey(0)

stiched_img = crop1(result)

return stiched_img

```



Εικόνα 16: Αποτέλεσμα της συνένωσης των εικόνων rio-03 και rio-04 μετά την αποκοπή των περιττών γραμμών και στηλών

Ακολουθώντας την ίδια μεθοδολογία για την συνένωση των δύο πρώτων εικόνων, συνενώνω και τα παραπάνω αποτελέσματα. Το αποτέλεσμα φαίνεται στην εικόνα 17.



Εικόνα 17: Αποτέλεσμα της συνένωσης των προηγούμενων αποτελεσμάτων μετά την αποκοπή των περιττών γραμμών και στηλών

Σχολιασμός και παρατηρήσεις

Διακριτές γραμμές συνένωσης

Παρατηρώ ότι για είναι αισθητά τα σημεία όπου γίνεται η συνένωση των εικόνων rio-03 και rio-04 καθώς και των δύο ενδιάμεσων αποτελεσμάτων. Το ανθρώπινο μάτι είναι εξαιρετικά ευαίσθητο στην μεταβολή της φωτεινότητας του αντικειμένου ως προς το περιβάλλον (brightness). Η ύπαρξη αυτών των γραμμών οφείλεται στο γεγονός ότι:

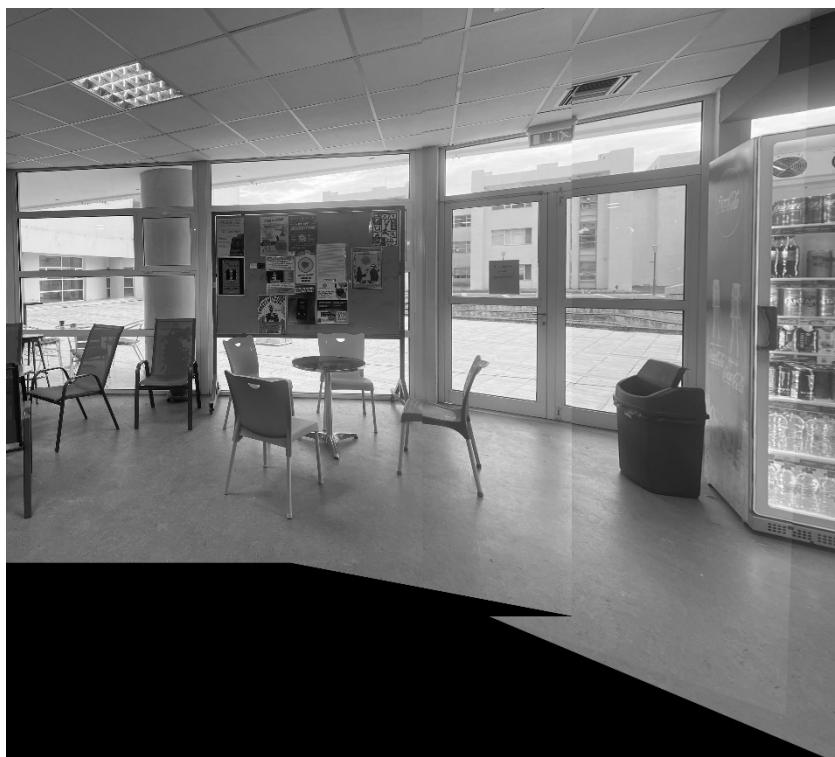
- 1) κατά την λήψη των φωτογραφιών αλλάζει η έκθεση του φακού (exposure) λόγω μεταβολών του φωτός (πχ η πηγή φωτός εμποδίζεται από κάποιο εμπόδιο, όπως στην περίπτωση που ένα σύννεφο περνάει μπροστά από τον ήλιο)
- 2) η απόκριση της φωτεινότητας του φακού της κάμερας περιορίζεται στην περιφέρεια (βινιετάρισμα). Όσο πιο κοντά στην περιφέρεια βρίσκεται ένα εικονοστοιχείο, τόσο λιγότερο φωτεινό είναι σε σχέση με αυτά που βρίσκονται πιο κοντά στο κέντρο της εικόνας.

Για την επίλυση αυτού του ζητήματος σκέφτηκα στην περιοχή όπου γίνεται η επικάλυψη της εικόνας στην οποία εφαρμόζονται οι μετασχηματισμοί (εικόνα πηγή) και της εικόνας που παραμένει αμετάβλητη (εικόνα προορισμού) να ελέγχω για κάθε εικονοστοιχείο την απόστασή του από την περιφέρεια και να επιλέγω για το ενδιάμεσο αποτέλεσμα το εικονοστοιχείο με την μεγαλύτερη απόσταση. Στο εικονοστοιχείο αυτό οι προαναφερόμενες επιπτώσεις είναι οι ελάχιστες δυνατές.

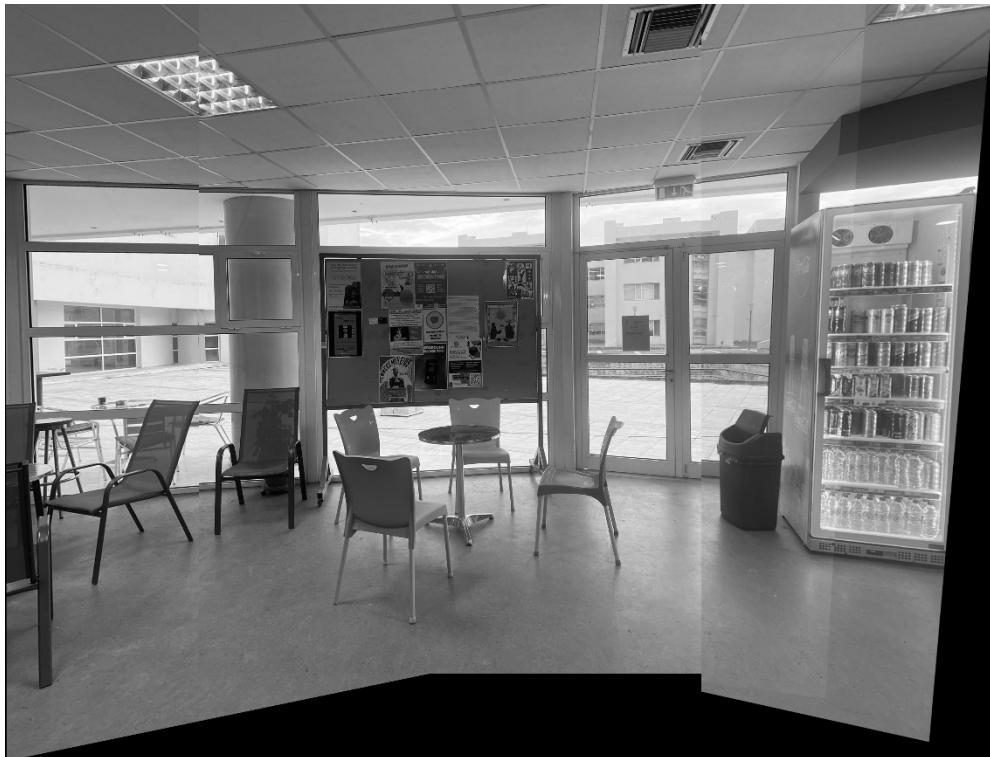
Η υλοποίηση αυτού του κώδικα αποδείχθηκε ιδιαίτερα απαιτητική και σε συνδυασμό με τον χρόνο που είχα διαθέσιμο δεν κατάφερα να τον υλοποιήσω εγκαίρως. Το πρόβλημα μου εντοπίζεται στον υπολογισμό της απόστασης κάθε εικονοστοιχείου από την περιφέρεια.

Αποτέλεσμα με τον κώδικα του εργαστηρίου

Η διαφορά εντοπίζεται καλύτερα για χρησιμοποιώντας τις δικές μου εικόνες



Εικόνα 18: Αποτέλεσμα με τον κώδικα του εργαστηρίου



Εικόνα 19: Αποτέλεσμα με τον δικό μου κώδικα

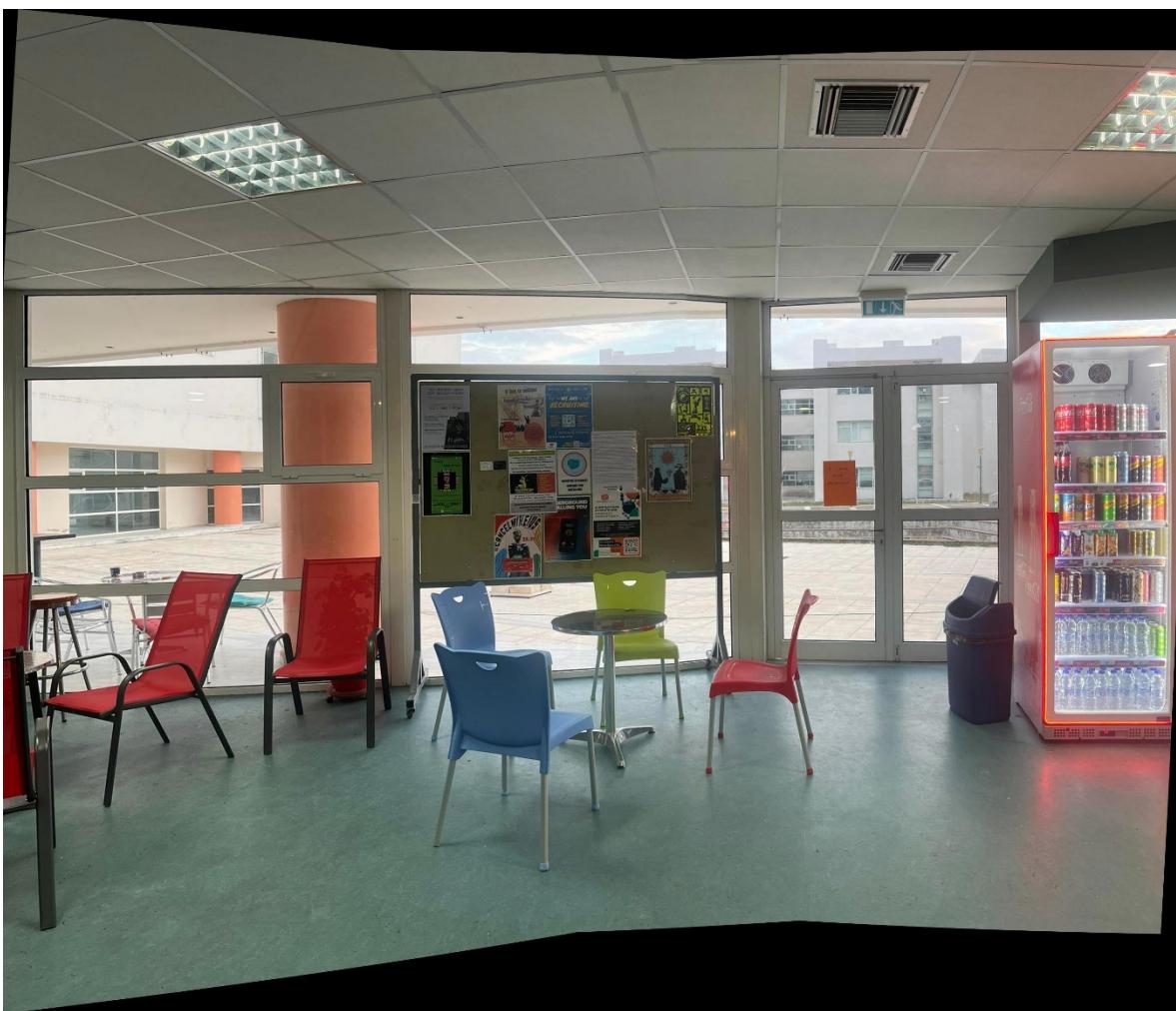
Παρατηρώ ότι η εικόνα 19 δίνει καλύτερη αίσθηση του χώρου και προσεγγίζει το αποτέλεσμα του λογισμικού του κινητού.

Αποτέλεσμα με το εργαλείο Image Composite Editor

Για την προβολή του αποτελέσματος επέλεξα την Perspective



Εικόνα 20: Αποτέλεσμα για τις εικόνες ρίο



Εικόνα 21: Αποτέλεσμα για τις δικές μου εικόνες

Το αποτέλεσμα είναι αρκετά κοντά με αυτό του δικού μου κώδικα τόσο για τις εικόνες που δόθηκαν από την εκφώνηση όσο και για τις δικές μου.