Όραση Υπολογιστών

2η Εργασία

Ονοματεπώνυμο: Ιωάννα Σταγωνά

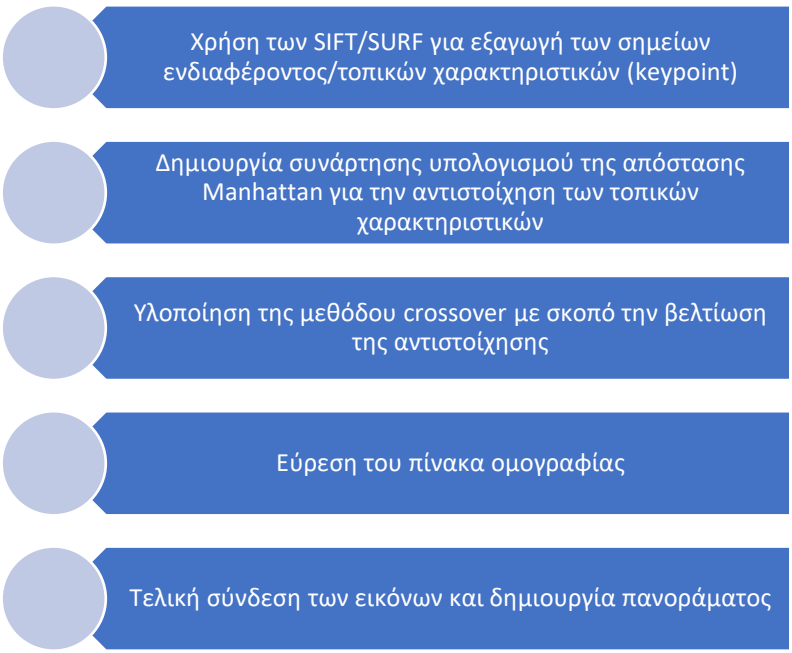
Εξάμηνο φοίτησης:7ο

Αριθμός Μητρώου:58125

Εισαγωγή

Η εργασία αυτή είχε ως στόχο την δημιουργία μιας πανοραμικής φωτογραφίας από τέσσερις αρχικές εικόνες. Χρησιμοποιήθηκαν οι ανιχνευτές και περιγραφείς SIFT(Scale-invariant feature transform) και SURF(Speeded-up robust features) καθώς επίσης το εργαλείο Image composite editor για την σύγκριση των αποτελεσμάτων.

Μπλοκ-διάγραμμα



Οι αρχικές εικόνες που μας δόθηκαν είναι οι εξής:

Εικόνα 1:rio -01

Εικόνα 2:rio -02

Εικόνα 3:rio -03

Εικόνα 4:rio -04

1.Ο περιγραφέας SIFT

Ο SIFT εκφράζει ένα ιστόγραμμα προσανατολισμών για κάθε keypoint. Εφαρμόζεται στην «γειτονιά» η οποία τοποθετείται γύρω από ένα σημείο ενδιαφέροντος και υπολογίζει για κάθε pixel τον προσανατολισμό στο κάθε κελί. Το μήκος κάθε άξονα μας δίνει την συχνότητα παρουσίας του αντίστοιχου προσανατολισμού. Τελικά καταλήγει για κάθε keypoint σε ένα διάνυσμα 128 διαστάσεων.

1.1 Υλοποίηση του SIFT στην Python

Καλούμε την συνάρτηση της open cv «cv.xfeatures2d_SIFT.create» η οποία μας δίνει τα keypoints(kp1,kp2) και τους περιγραφείς (desc1,desc2) για κάθε εικόνα αντίστοιχα. Επιλέγουμε την εξαγωγή 1000 τοπικών χαρακτηριστικών έτσι ώστε να λειτουργήσει με μεγαλύτερη ακρίβεια η συνάρτηση crossover που θα υλοποιηθεί παρακάτω .Επιπλέον χρησιμοποιούμε την συνάρτηση «cv.drawKeypoints» ώστε να εμφανίσουμε τα keypoints στις αρχικές εικόνες.

```
 detector_descriptor = cv.xfeatures2d_SIFT.create(1000)
cv.namedWindow('main1')
img1 = cv.imread(filename1)
grayimg1 = cv.imread(filename1, cv.IMREAD_GRAYSCALE)
cv.namedWindow('main1')
cv.imshow('main1', img1)
cv.waitKey(0)

kp1 = detector_descriptor.detect(img1)
desc1 = detector_descriptor.compute(img1, kp1)
#draw the keypoints in img1
kpimg1 = cv.drawKeypoints(grayimg1,kp1, img1,
flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv.imwrite('keypoints1.jpg', kpimg1)
cv.namedWindow('keypoints1')
cv.imshow('keypoints1', img1)
cv.waitKey(0)

img2 = cv.imread(filename2)
grayimg2 = cv.imread(filename2, cv.IMREAD_GRAYSCALE)
cv.namedWindow('main2')
cv.imshow('main2', img2)
cv.waitKey(0)

kp2 = detector_descriptor.detect(img2)
desc2 = detector_descriptor.compute(img2, kp2)
#draw the keypoints in img2
kpimg2 = cv.drawKeypoints(grayimg2, kp1, img1,
flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv.imwrite('keypoints2.jpg', kpimg2)
cv.namedWindow('keypoints2')
cv.imshow('keypoints2', img2)
cv.waitKey(0)
```

Εικόνα 5: SIFT in python

2.Ο περιγραφέας SURF

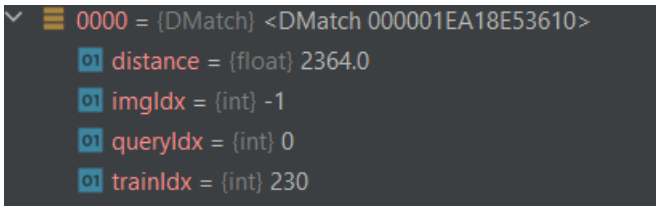
Ο περιγραφέας SURF είναι 6 φορές πιο γρήγορος από τον SIFT . Δεν έχει ιστογράμματα και έχει 4 νούμερα (για 4x4) υπο-περιοχές υπολογίζοντας τα αθροίσματα με integral images.. Ο τελικός περιγραφέας είναι 64 θέσεων αντί για 128 του SIFT .

2.1 Υλοποίηση του SURF στην Python

Η υλοποίηση του SURF στην python είναι όμοια με αυτή που αναλύθηκε παραπάνω καλώντας την συνάρτηση «cv.xfeatures2d_SURF.create». Αυτή την φορά θα επιλέξουμε την εφαρμογή του SURF θέτοντας το Hessian Threshold ίσο με 5000 ειδάλλως θα γίνει η εξαγωγή πάρα πολλών Keypoints άσκοπα. Το όριο αυτό καθορίζει πόσο μεγάλη πρέπει να είναι η έξοδος από το φίλτρο Hessian προκειμένου ένα σημείο να χρησιμοποιηθεί ως σημείο ενδιαφέροντος. Μια μεγαλύτερη τιμή θα έχει ως αποτέλεσμα λιγότερα, αλλά (θεωρητικά) πιο σημαντικά σημεία ενδιαφέροντος, ενώ μια μικρότερη τιμή θα έχει ως αποτέλεσμα περισσότερα αλλά λιγότερο σημαντικά σημεία.

3. Αρχική αντιστοίχιση των keypoints

Για την αρχική αντιστοίχιση των keypoints(εύρεση ομόλογων σημείων) υλοποιήθηκε μια συνάρτηση η οποία υπολογίζει για κάθε ένα χαρακτηριστικό (desci) της μίας εικόνας από κάθε ένα χαρακτηριστικό της άλλης (descj) την απόσταση Manhattan.Έπειτα βρίσκει το «descj» με την ελάχιστη απόσταση έτσι ώστε να γίνει ένα αρχικό «matching» των τοπικών χαρακτηριστικών ανάμεσα στις δύο εικόνες χρησιμοποιώντας την κλάση «cv.DMatch». Η παραπάνω κλάση όπως φαίνεται περιέχει 4 χαρακτηριστικά για κάθε keypoint.



Εικόνα 6:Περιεχόμενα των DMatches

Το «queryIdx» αναφέρεται στο kepoint της αρχικής εικόνας ενώ το «trainIdx» αναφέρεται στο keypoint της δεύτερης εικόνας που έχει γίνει η αντιστοίχιση.

Θα καλέσουμε την συνάρτηση 2 φορές. Από την εικόνα 1 προς την εικόνα 2 και αντίστροφα καθώς θα χρειαστούν και οι δύο αντιστοιχίες για την υλοποίηση της μεθόδου «crossover» παρακάτω.

```
matches_1 = match(desc1[1], desc2[1])
matches_2 = match(desc2[1], desc1[1])
```

Εικόνα 7: Κλήση της συνάρτησης match

4.Υλοποίηση της μεθόδου «crossover»

Όπως φαίνεται και με τα αποτελέσματα που θα παρουσιαστούν παρακάτω, η υλοποίηση μόνο της παραπάνω διαδικασίας για την εύρεση αντιστοιχίσεων δεν είναι αρκετή καθώς αποδίδονται κάποια ζεύγη ομόλογων σημείων λανθασμένα. Συνεπώς ακολουθεί μια διαδικασία η οποία έχει ως στόχο την βελτίωση του «matching» η οποία ονομάζεται crossover. Ουσιαστικά θα περιορίσει τα «DMatches» βασιζόμενη στην παρακάτω λογική:

Για να θεωρηθεί δεκτή η αντιστοίχιση ενός τοπικού χαρακτηριστικού Α της εικόνας 1 με ένα χαρακτηριστικό Β της εικόνας 2 θα πρέπει στη κλήση της «cv.DMatch» από την εικόνα 2 προς την 1 το Β να έχει αντιστοιχηθεί με το Α. Πρακτικά, αυτό σημαίνει ότι θα πρέπει να ισχύουν οι σχέσεις «queryIdx(A)=trainIdx(B)» και «trainIdx(A)=queryIdx(B)» ταυτόχρονα.

Επίσης μπορεί να χρησιμοποιηθεί η συνάρτηση «cv.drawMatches» για να παραχθεί ένα οπτικό αποτέλεσμα των τελικών αντιστοιχίσεων μετά την εφαρμογή της παραπάνω διαδικασίας.

```
#create the cross check
arr1=[]
arr2=[]
i=0
matchesfinal=[]
for x in matches_1:
    arr1.append([x.queryIdx,x.trainIdx])
    for y in matches_2:
        arr2.append([y.queryIdx,y.trainIdx])

        if x.queryIdx==y.trainIdx and x.trainIdx==y.queryIdx:
            matchesfinal.append(matches_1[i])
    i=i+1
matchesfinal=np.array(matchesfinal)
dimg3 = cv.drawMatches(img1, desc1[0], img2, desc2[0], matchesfinal, None)
cv.namedWindow('main5')
cv.imshow('main5', dimg3)
cv.waitKey(0)
```

Εικόνα 8:Υλοποίηση της μεθόδου crossover

5. Εύρεση του πίνακα ομογραφίας

Για την παραγωγή της πανοραμικής εικόνας είναι απαραίτητη η εύρεση του πίνακα ομογραφίας(πίνακας μετασχηματισμού Μ).Η εύρεση των μετασχηματισμών υλοποιείται μέσω της «cv.findHomography» όπου ουσιαστικά μας δείχνει πως θα πρέπει να μετατραπεί η 2η εικόνα ώστε να «ταιριάξει» με την 1η και να δημιουργηθεί το πανοραμικό αποτέλεσμα. Μόλις υπολογιστεί ο πίνακας Μ καλούμε την «cv.warpPerspective» έτσι ώστε να γίνει η εφαρμογή των μετασχηματισμών στην 2η εικόνα. Οι διαστάσεις της τελικής εικόνας θέτονται αρκετά μεγαλύτερες από αυτές της αρχικής έτσι ώστε να είμαστε σίγουροι ότι θα «χωρέσει» η πανοραμική φωτογραφία.

```
img3 = cv.warpPerspective(img2, M, (img1.shape[1]+1000,
img1.shape[0]+1000))
img3[0: img1.shape[0], 0: img1.shape[1]] = img1
```

Εικόνα 9:Δημιουργία πανοραμικής φωτογραφίας.

6. Δημιουργία τελικού πανοράματος από τις 4 φωτογραφίες

Όλες οι παραπάνω διαδικασίες έχουν συμπεριληφθεί σε μία συνάρτηση panorama. Αφού καλέσουμε αυτή την συνάρτηση για τα 2 ζεύγη φωτογραφιών θα προκύψουν 2 ενδιάμεσα πανοράματα τα οποία όταν ενωθούν θα δοθεί το τελικό αποτέλεσμα. Ωστόσο καθώς στην προηγούμενη διεργασία θεωρήσαμε τις διαστάσεις της πανοραμικής εικόνας αρκετά μεγαλύτερες από ότι τελικά χρειάστηκε παρατηρούμε ότι δημιουργούνται μαύρα borders γύρω από την τελική εικόνα. Αυτά θα πρέπει να απομακρυνθούν έτσι ώστε να μπορέσει να εφαρμοστεί σωστά η συνάρτηση με εισόδους τα 2 ενδιάμεσα πανοράματα. Για να επιτευχθεί το παραπάνω επιλέχθηκε η υλοποίηση μιας συνάρτησης crop η οποία δρα ως εξής:

- Μετατροπή της εικόνας εισόδου σε δυαδική με threshold=1 ώστε να διαχωριστούν τα μαύρα περιγράμματα από την εικόνα που μας ενδιαφέρει
- Χρησιμοποίηση της συνάρτησης «cv.connectedComponentsWithStats» που αναλύθηκε στην προηγούμενη εργασία
- Εξαγωγή των στατιστικών για το 2ο συνδεδεμένο αντικείμενο (το 1ο είναι πάντα το background άρα το 2ο θα αφορά την εικόνα που μας ενδιαφέρει)
- Δημιουργία μιας νέας crop εικόνας με βάση των στατιστικών

Να σημειωθεί ότι αφαιρούνται και 15 στήλες pixel καθώς το σχήμα της τελικής εικόνας θα πρέπει να είναι ορθογώνιο χωρίς περίσσια μαύρα περιγράμματα.

```
#make a function for croping the black borders of the panorama images
def cropborders(filename):
    img=cv.imread(filename)
    grayimg = cv.imread(filename, cv.IMREAD_GRAYSCALE)
    # create the binary img
    _, binary = cv.threshold(grayimg, 1, 255, cv.THRESH_BINARY)
    cv.namedWindow('binaryimg')
    cv.imshow('binaryimg', binary)
    cv.waitKey(0)
    num_cc, labeled, stats, centroids = (binary)
    x0 = stats[1, cv.CC_STAT_LEFT]
    y0 = stats[1, cv.CC_STAT_TOP]
    w = stats[1, cv.CC_STAT_WIDTH]
    h = stats[1, cv.CC_STAT_HEIGHT]
    crop=img[y0:y0+h-15,x0:x0+w]
    cv.namedWindow('panoramacroped')
    cv.imshow('panoramacroped', crop)
    cv.waitKey(0)
    return crop
```

Εικόνα 10:Περικοπή μαύρων περιγραμμάτων

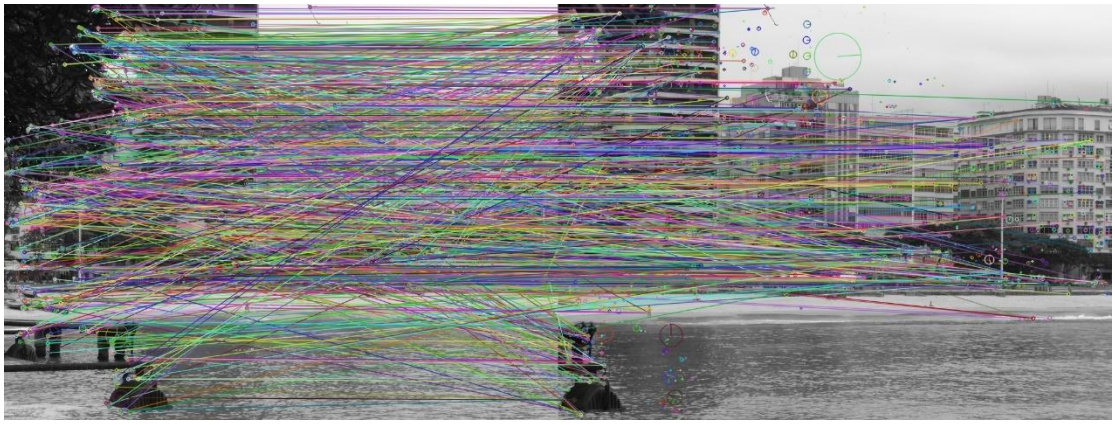
7. Παρουσίαση οπτικών αποτελεσμάτων

Ενδεικτικά θα παρουσιαστούν τα αποτελέσματα για το ζευγάρι εικόνων rio-01,rio-02 ,τα ενδιάμεσα πανοράματα καθώς και το τελικό πανόραμα.

Με την χρήση του περιγραφέα SIFT:

Εικόνα 11:Keypoints της rio-01

Εικόνα 12:Keypoints της rio-02



Εικόνα 13: Αρχικό Matching

Εικόνα 14:Matching έπειτα από την εφαρμογή της crossover μεθόδου

Εικόνα 15:Δημιουργία του 1ου ενδιάμεσου πανοράματος

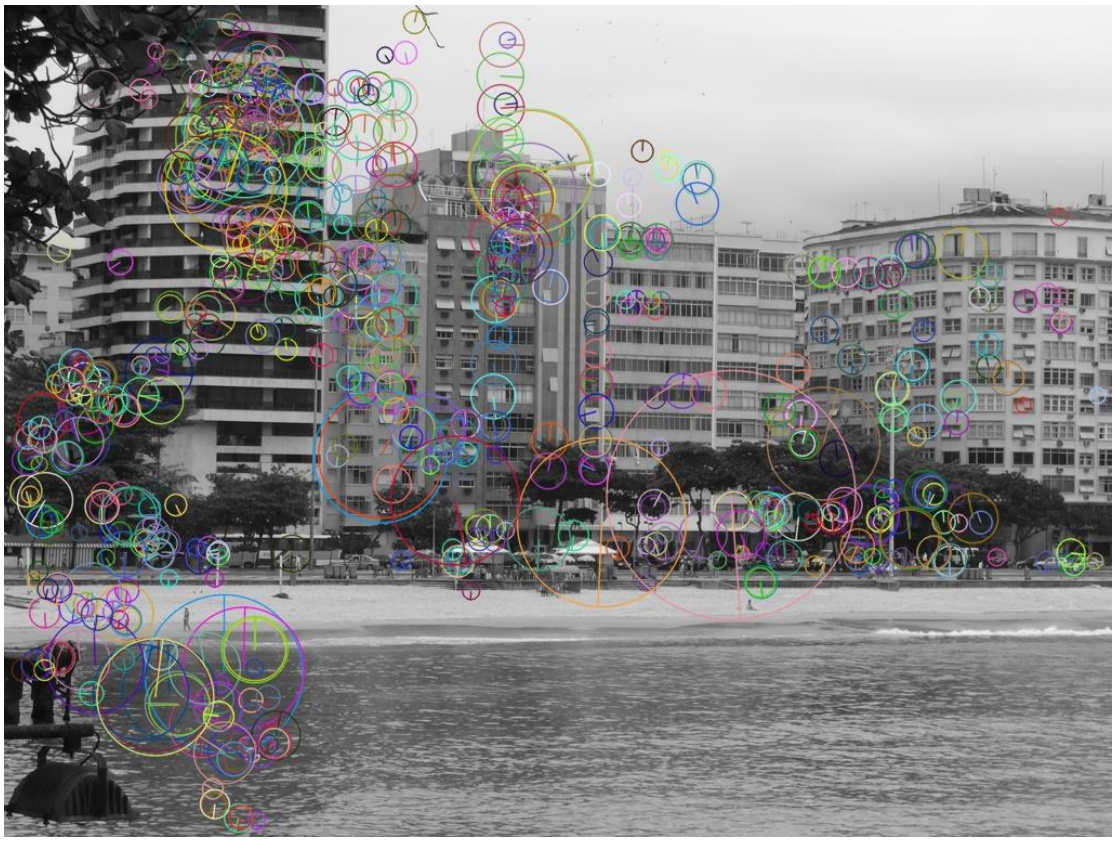
Εικόνα 16:Περικοπή

Εικόνα 17:Δημιουργία και περικοπή του 2ου ενδιάμεσου πανοράματος

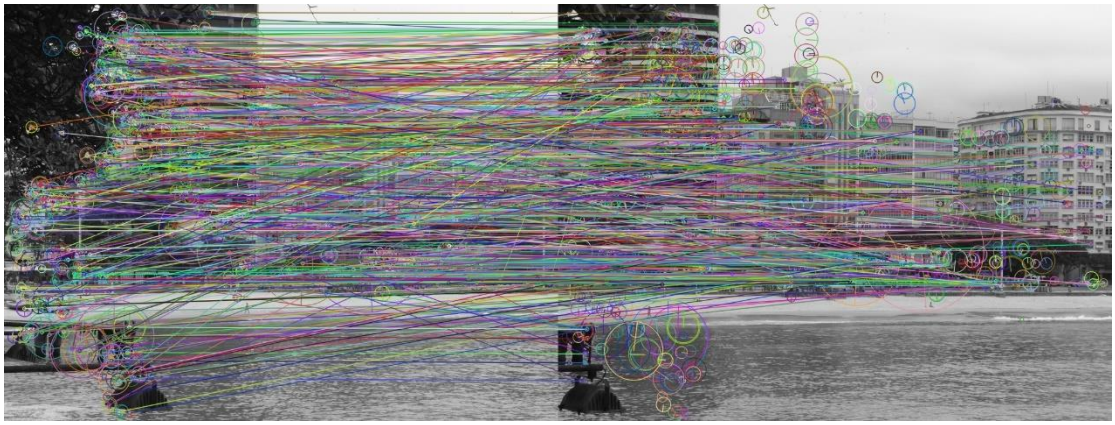
Εικόνα 18:Τελικό αποτέλεσμα

Με την χρήση του περιγραφέα SURF στο τελικό πανόραμα φαίνεται να υπάρχει κάποια ανομοιομορφία στην σύνδεση των 2 ενδιάμεσων πανοραμάτων (η διαφορά είναι εμφανής στα κτήρια),ίσως εάν είχα θέσει ένα μικρότερο threshold και γινόταν ο εντοπισμός παραπάνω keypoints το πρόβλημα να είχε επιλυθεί ωστόσο έπειτα από δοκιμές αντιλήφθηκα ότι θα καθυστερούσε πολύ η διαδικασία. Παρουσιάζονται ενδεικτικά οι εικόνες που φαίνονται τα ζωγραφισμένα keypoints και matches.Όπως φαίνονται οι περιοχές που λαμβάνονται ως τοπικά χαρακτηριστικά είναι μεγαλύτερες γεγονός που κάνει κατανοητές τις διαφορές που αναφέρθηκαν στην αρχή της εργασίας.

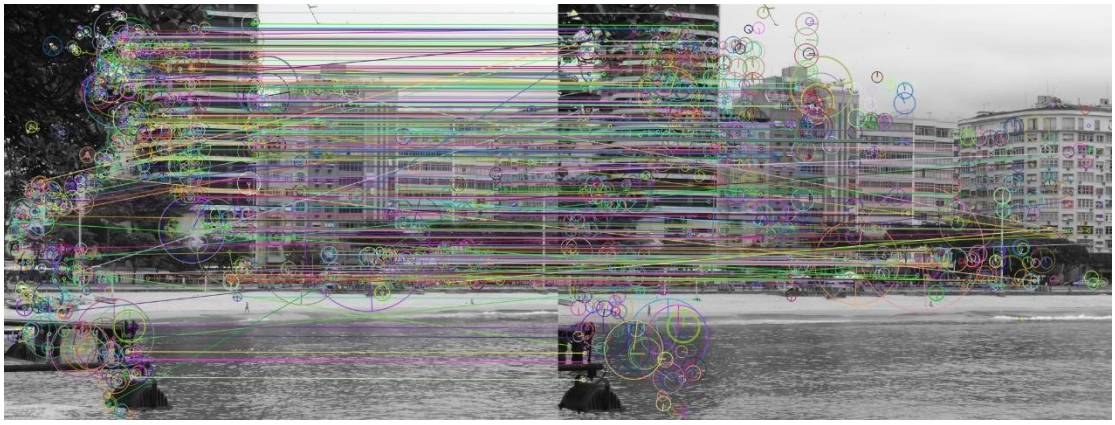
Εικόνα 19:Keypoints της rio-01



Εικόνα 19:Keypoints της rio-02



Εικόνα 20:Αρχικό Matching



Εικόνα 21: Matching έπειτα από την εφαρμογή της crossover μεθόδου

Εικόνα 22: Τελικό πανόραμα με την χρήση του SURF

8. Συγκρίσεις-Συμπεράσματα

Εικόνα 23: Πανοραμική φωτογραφία με την χρήση του Image composite editor

Συγκρίνοντας τις πανοραμικές φωτογραφίες που δημιούργησα με αυτή του image composite editor θεωρώ ότι η διαδικασίες που εφαρμόστηκαν έχουν αποδώσει , με καλύτερη επιλογή περιγραφέα να είναι αυτή του SIFT.Υπάρχουν κάποιες ανομομοιομορφίες στην θάλασσα και στον δρόμο(χάνεται η συνέχεια ενός αμαξιού, ανομοιομορφία στα κύματα) αλλά έπειτα από παρατήρηση των αρχικών εικόνων καταλαβαίνουμε ότι αυτό οφείλεται στο γεγονός ότι οι φωτογραφίες είναι σε διαφορετική στιγμή τραβηγμένες .Για παράδειγμα παρατηρούμε ότι ένα αμάξι που υπάρχει στην εικόνα 1 δεν υπάρχει στην εικόνα 2 και συνεπώς δεν μπορεί να πραγματοποιηθεί η κατάλληλη αντιστοιχία.

9. Δημιουργία πανοράματος με δικές μου φωτογραφίες

Οι φωτογραφίες που επέλεξα είναι οι εξής:

Εικόνα 24: xanthi1

Εικόνα 25: xanthi2

Εικόνα 25:xanthi3

Εικόνα 26:xanthi4

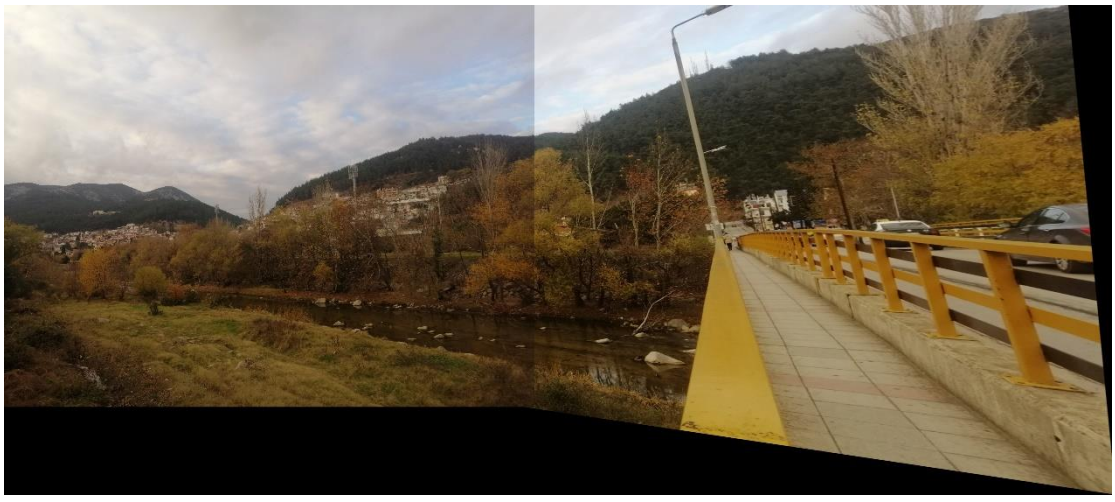
Το ενδιάμεσο πανόραμα των πρώτον 2 φωτογραφιών:

Εικόνα 27:Πανόραμα πρώτου ζεύγους

Το ενδιάμεσο πανόραμα των υπόλοιπων 2 φωτογραφιών:

Εικόνα 28:Πανόραμα δεύτερου ζεύγους

Και το τελικό πανόραμα μετά από την σύνδεση των 2 παραπάνω διαμορφώθηκε ως εξής:



Εικόνα 29:Τελικό πανόραμα

Εικόνα 30:Πανόραμα με την χρήση του Image Compositor

9.1 Παρατηρήσεις

Αρχικά, συγκρίνοντας το πανόραμα που παράχθηκε με αυτό του image compositor παρατηρώ ότι οι εικόνες βρίσκονται σχετικά κοντά. Γενικά η συνέχεια διακρίνεται πιο έντονα κατά την δημιουργία των ενδιάμεσων πανοραμάτων και όχι τόσο του τελικού ,μάλιστα στο τελικό πανόραμα η 2η εικόνα φαίνεται να έχει υποστεί αρκετούς μετασχηματισμούς σε αντίθεση με το αποτέλεσμα του Image Compositor όπου οι διαφορές δεν είναι το ίδιο αντιληπτές.