ΕΡΓΑΣΙΑ 2

ΟΡΑΣΗ ΥΠΟΛΟΓΙΣΤΩΝ

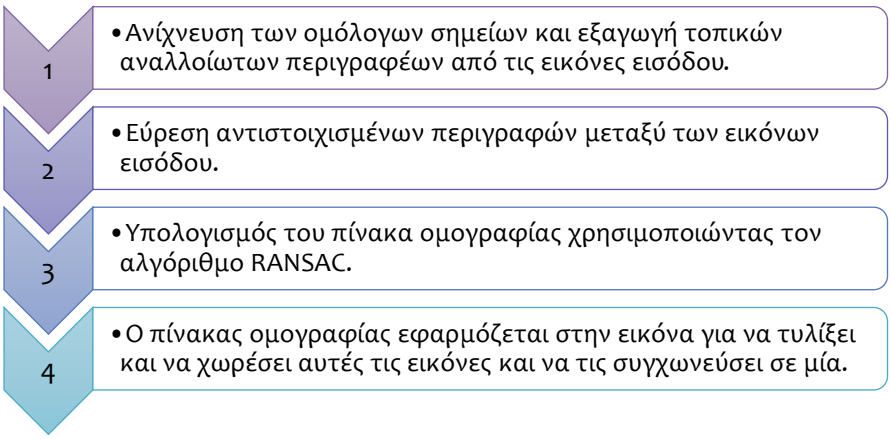
ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Ζαχάρη Βάια

ΑΜ: 58161

Εισαγωγή:

Η συγκεκριμένη εργασία αφορά στην υλοποίηση αλγορίθμου που παράγει πανοράματα από πολλαπλές επιμέρους εικόνες. Σε πρώτο στάδιο δίνονται από την εκφώνηση τέσσερις εικόνες και παράγεται το πανόραμα των εικόνων με τον αλγόριθμο sift και στη συνέχεια με βάσει τον αλγόριθμο surf. Σε δεύτερο στάδιο είσαγονται τέσσερις νέες εικόνες που τραβήχτηκαν από το κινητό και εφαρμόζουμε την ίδια διαδικασία. Επιπλέον αφού συγκριθεί το αποτέλεσμα με το ιδανικό που εξάγει η εφαρμογή Image Composite Editor, θα αναλυθούν παρατηρήσεις ,αστοχίες και συμπεράσματα σχετικά με τον τρόπο υλοποίησης.

ΜΕΘΟΔΟΛΟΓΙΑ



SIFT

1ο Στάδιο:

Αφού εισάγουμε τις εικόνες και τις μεταφέρουμε σε γκρι κλίμακα, δημιουργούμε τη συνάρτηση panorama, η οποία έχει ως ορίσματα δύο εικόνες. Στη συνέχεια στόχος μας είναι να υπολογίσουμε τα σημεία ενδιαφέροντος, keypoints, της κάθε εικόνας.

```
kp1 = sift.detect(img1)
kp2 = sift.detect(img2)
```

Τα σημεία ενδιαφέροντος είναι σημεία αναγνωριστικά για την εικόνα και μοναδικά ,ακόμα και αν αυτή περιστραφεί .Με βάση αυτά θα γίνει η αντιστοίχιση των δύο εικόνων.

Για κάθε keypoint παίρνουμε ένα διάνυσμα με τα τοπικά χαρακτηριστικά της εικόνας,γύρω από το σημείο ενδιαφέροντος, τον περιγραφέα.Οι περιγραφείς υπολογίζονται εξετάζοντας τη γειτονιά του σημείου κλειδιού, αναλύοντας την τοπική γειτονιά σε μικρές περιοχές και στη συνέχεια υπολογίζοντας τη διαβάθμιση σε αυτές τις μικρές περιοχές. Αυτές οι διαβαθμίσεις συλλέγονται με τη μορφή ιστογραμμάτων. Πιο συγκεκριμένα πόσο συχνά εμφανίζονται αυτές οι διαβαθμίσεις και το μέγεθός τους μετατρέπονται σε ιστογράμματα για μικρές τοπικές

περιοχές. Τέλος, η συνένωση όλων των τιμών που υπολογίζονται από το ιστόγραμμα θα έχει ως αποτέλεσμα το διάνυσμα του περιγραφέα.

```
desc1 = sift.compute(img1, kp1)
desc2 = sift.compute(img2, kp2)
```

Με την εντολή sift.compute() προσκομίζουμε τους 128 descriptors του κάθε keypoint.

Matching

Η εύρεση των ομόλογων σημείων που θα αντιστοιχηθούν επιτυγχάνεται μέσω της συνάρτησης match(). Η συνάρτηση αυτή για κάθε keypoint της εικόνας 1 υπολογίζει την Manhattan απόσταση των περιγραφέων τους με τους περιγραφείς των keypoint της εικόνας 2. Τα αντιπροσωπευτικά keypoints θα είναι αυτά που η παραπάνω απόσταση χαρακτηρίζεται ως η ελάχιστη. Στο τέλος η συνάρτηση επιστρέφει μία σειρά από ζευγάρια τύπου DMatch.Το καθένα περιέχει δύο βασικές μεταβλητές την query και την train , οι οποίες δίνουν το keypoint της εικόνας που συγκρίνω και το keypoint της εικόνας με την οποία συγκρίνω και θα κάνει match, αντίστοιχα.

Έχοντας υπόψιν αυτή τη πληροφορία για να κάνω τελικά την αντιστοίχιση δημιουργώ μία συνάρτηση την best_matches(), η οποία θα διαχωρίσει τα καλά matches. Συγκεκριμένα για να επιβεβαιωθεί ένα ζευγάρι χρειάζεται τα ζευγάρια που θα παράξει η συνάρτηση match() ,όταν την εφαρμόζουμε στην εικόνα 1 για την εικόνα 2 και το αντίστροφο,να είναι τα ίδια. Αυτό μπορεί να γίνει αν ελέγχουμε τις δύο παραπάνω μεταβλητές χιαστί, δηλαδή το query της 1 να είναι ίδιο με το train της εικόνας 2 και το αντίστροφο.

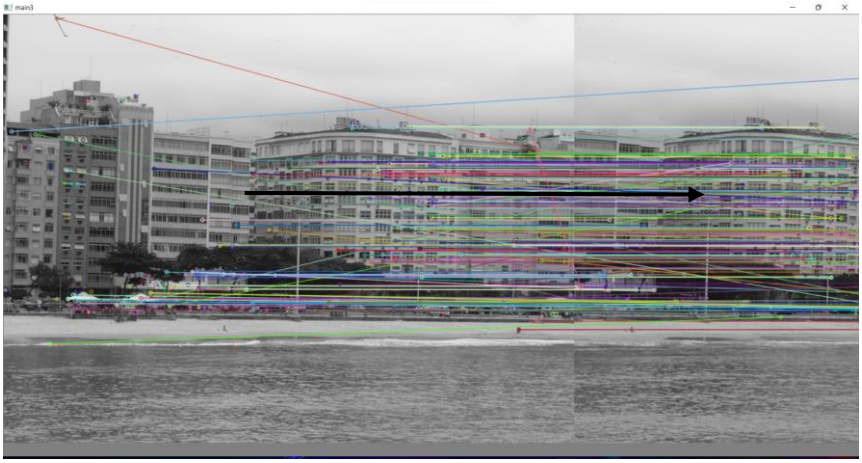
```
for i in matches1_2:
    for j in matches2_1:
        t1 = i.trainIdx
        q1 = i.queryIdx
        q2 = j.queryIdx
        t2 = j.trainIdx
        if (q1 == t2) and (q2 == t1):
            good.append(i)
```

Σχεδιασμός ένωσης των keypoints

Υπάρχει η δυνατότητα να σχεδιάσουμε την ένωση των keypoints της κάθε εικόνας με έγχρωμες γραμμές με την παρακάτω κλήση συνάρτησης:

```
dimg = cv2.drawMatches(img1, desc1[0], img2, desc2[0], good_matches, None)
```

Έτσι μπορούμε να παρατηρήσουμε ότι οι γραμμές αυτές ακολουθούν ένα βασικό προσανατολισμό και να επιβεβαιώσουμε την αντιστοίχηση των keypoints.



Ομογραφία

Η ομογραφία συσχετίζει τα ομόλογα σημεία δύο θέσεων προβολής με βάση ένα παρατηρητή.

Μέχρι στιγμής δεν ασχοληθήκαμε με συντεταγμένες των σημείων ενδιαφέροντος, διότι η ελάχστη απόσταση αναφερόταν στους περιγραφείς. Τώρα όμως που βρέθηκαν τα ζευγάρια βρίσκουμε τις συντεταγμένες των keypoints που θα αντιστοιχηθούν,καθώς χρειάζονται ως ορίσματα στην συνάρτηση cv2.findHomography().

```
img_pt1 = np.array([kp1[x.queryIdx].pt for x in good_matches])
img_pt2 = np.array([kp2[x.trainIdx].pt for x in good_matches])
M, mask = cv2.findHomography(img_pt2, img_pt1, cv2.RANSAC)
```

Για τον υπολογισμό της ομογραφίας χρησιμοποιείται ο αλγόριθμος RANSAC, ο οποίος μας δίνει την επιβεβαίωση όταν τα ομόλογα σημεία που επιλέχθηκαν ακολουθούν ένα κοινό προσανατολισμό, όπως αναφέρθηκε προηγουμένως.

Με βάση τον πίνακα μετασχηματισμού M βρίσκουμε την θέση της μίας εικόνας μέσα στην άλλη. Ιδιαίτερη προσοχή χρειάζεται το μέγεθος της τελικής εικόνας ,καθώς αν το κομμάτι που ενώνεται δεν χωράει στην αρχική εικόνα,η τελική εικόνα πρέπει να είναι μεγαλύτερη.Έτσι αυξάνουμε το μέγεθος κατα 1000 pixel και στους δύο άξονες. Έπιπλεον η παρακάτω συνάρτηση εφαρμόζει στην εικόνα 2 τον σύνθετο μετασχηματισμό.

```
img5 = cv2.warpPerspective(img2, M, (img1.shape[1]+1000, img1.shape[0]+1000))
```

Πανόραμα

Όταν καλούμε τη συνάρτηση panorama() λαμβάνουμε ως αποτέλεσμα το πανόραμα των δύο εικόνων που είχαν εισαχθεί ως ορίσματα.

Όπως αναφέρθηκε στην εισαγωγή το τελικό πανόραμα θέλουμε να αποτελείται όχι μόνο από δύο, αλλά από τέσσερις εικόνες. Για να κατατμήσουμε τη δυσκολία ταυτόχρονης διαχείρησης πολυάριθμων εικόνων, χωρίζουμε τις εικόνες σε δύο ζευγάρια.

Άρα διακρίνουμε 3 περιπτώσεις πανοράματος:

1)πανόραμα των εικόνων 1 και 2

2)πανόραμα των εικόνων 3 και 4

3)πανόραμα των πανοραμάτων των περιπτώσεων 1) και 2)

ΠΕΡΙΠΤΩΣΗ 1:

Image_1: rio-01.png

Image_2: rio-02.png

Πανόραμα: panorama1

ΠΕΡΙΠΤΩΣΗ 2:

Image_1: rio-03.png

Image_2: rio-04.png

Πανόραμα: panorama2

ΠΕΡΙΠΤΩΣΗ 3:

Image_1: panorama1.png

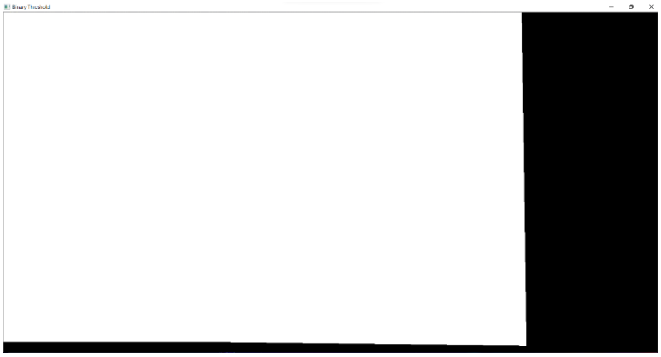
Image_2: panorama2.png

Πανόραμα: panorama3

Αν εφαρμόσουμε κατευθείαν την συνάρτηση panorama() για τα δύο πανοράματα θα παρατηρήσουμε αστοχίες.

Για να μπορέσουμε να ενώσουμε τις εικόνες πανοράματος πρέπει να απομακρύνουμε τα μαύρα πίξελ. Αυτά εμφανίζονται όταν αυξάνουμε το μεγέθος της εικόνας κι έτσι όταν προσπαθούμε να προσθέσουμε την δεύτερη εικόνα τα μάυρα πίξελ μένουν και παραμορφώνουν τα αποτελέσματα.Για το λόγο αυτό δημιουργούμε την συνάρτηση separate() που λαμβάνει ως ορίσματα δύο εικόνες και επιστρέφει μία ενοποιημένη εικόνα χωρίς τα μαύρα πίξελ ενδιάμεσα των εικόνων.

Για να καταφέρουμε να απομονώσουμε τη πληροφορία της εικόνας από τα μαύρα πίξελ, πρέπει πρώτα να μετατρέψουμε την εικόνα σε binary με βάση ένα threshold. Το αποτέλεσμα είναι η περιοχή που αντιστοιχεί στην καθαρή εικόνα να γεμίσει τα εικονοστοιχεία με τιμή 255, ενώ τα μαύρα πίξελ να πάρουν τη τιμή 0. Άρα παίρνουμε το εξής αποτέλεσμα.



Binary image of image: Panorama1

Binary image of image: Panorama2

Στη συνέχεια εντοπίζουμε τα σημεία που παρατηρούνται οι εναλλαγές από 255 σε 0 στο κατακόρυφο και στον οριζόντιο άξονα.

Για τον οριζίοντιο άξονα τρεις συνθήκες ελέγχονται:

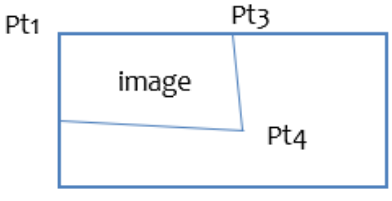
```
if (binary[row, col] == 255) and (binary[row - 1, col] == 0) and (row != 0):

elif (binary[row, col] == 255) and (binary[row - 1, col] == 0) and (row != 0) and (col == 0):

elif (binary[row, col] == 255) and (binary[row , col + 1 ] == 0):
```

Για τον κατακόρυφο άξονα:

```
if (binary[row, col] == 255) and (binary[row , col - 1] == 0) and (col != 0) :
```

α τελικά σημεία του οριζόντιου είναι ένα array pt3 με 2 στήλες για τις δύο συντεταγμένες x και y του κάθε σημείου του κατακόρυφου pt4 αντίστοιχα:

```
pt3 = pt3[0:r,0:cnt_col-cnt_col0-1]
pt4 = pt4[0:r2,0:cnt2_col-cnt2_col0-1]
```



Με βάση τα σημεία αυτά, ένα για τον κατακόρυφο και ένα για το οριζόντιο , οριοθετούμε το δεξί και κάτω μέρος της παραγόμενης εικόνας.Σύμφωνα με αυτά τα σημεία επιτρέπουμε τα μαύρα πίξελ να περάσουν ή όχι, ανάλογα με πόση πληροφορία από την αρχική θέλουμε να πετάξουμε.

Με δοκιμές μπορούμε να παράξουμε εικόνες που θα βασίζονται στα σημεία pt3 ή pt4.

Η παράμετρος l της separate() καθορίζει ποιες περιπτώσεις παίρνουμε.

ΠΕΡΙΠΤΩΣΗ 1:

Image: pan1.png

Πανόραμα: pan3

Με βάση μόνο το τελευταίο σημείο του pt4, l = 2

Ευστοχία

Με βάση μόνο το τελευταίο σημείο του pt3, l = 1

Αστοχία

ΠΕΡΙΠΤΩΣΗ 2:

Image: pan2.png

Πανόραμα: pan3

Με βάση μόνο το τελευταίο σημείο του pt4, l = 2

Ευστοχία

Με βάση μόνο το τελευταίο σημείο του pt3, l = 1

Αστοχία

ΠΕΡΙΠΤΩΣΗ 3:

Image_1: pan1.png

Image_2: pan2.png

Πανόραμα: panorama3

Ευστοχία

Αστοχία

Πανόραμα εφαρμογής:

ΠΑΡΑΤΗΡΗΣΕΙΣ:

Μία επιπλέον αστοχία είναι ότι η το τελικό πανόραμα δεν είναι ίδιο με αυτό της εφαρμογής.Παρουσιάζει θολά σημεία κυριώς στην θάλασσα.Γενικά γνωρίζουμε ότι τα keypoints βρίσκονται υπολογίζοντας τη διαφορά του Gaussian blur της εικόνας σε διαφορετικά επίπεδα. Αυτό σημαίνει ότι η εικόνα είναι θολή με γκαουσιανό θάμπωμα σε διαφορετικά μεγέθη, από ελαφρώς θολή έως πιο θολή κτλ. Στη συνέχεια, αυτές οι εικόνες αφαιρούνται η μία από την άλλη με αποτέλεσμα τη διαφορά των εικόνων με διαφορετικά επίπεδα Gaussian θαμπώματος. Οι εικόνες που προκύπτουν στοιβάζονται η μία πάνω στην άλλη για να αναζητηθούν ακραία σημεία που είναι τοπικά διακριτά, αυτά είναι ομόλογα σημεία. Οπότε μία πιθανή λύση για το θάμπωμα οφείλεται σε αυτή την διαδικασία.

Επιπρόσθετα παρατηρούμε ασυνέχειες μεταξύ των εικόνων στη τελική εικόνα.Οι ασυνέχειες οφείλονται σε αστοχίες συνένωσης των εικόνων μετά τον μετασχηματισμό. Πιθανόν ο πίνακας ομογραφίας να χρειάζεται παραπάνω ζευγάρια keypoints για να κάνει την αντιστοίχιση και να πετύχει καλύτερο προσανατολισμό.

2ο Στάδιο

Εισαγωγή νέων τεσσάρων εικόνων:

1η περίπτωση: pt4 πρώτο και τελευταίο σημείο,l=4

2η περίπτωση:pt4 τελευταίο σημείο,l=2

Πανόραμα

Σύγκριση με το πανόραμα της εφαρμογής (έγχρωμη)

Το αποτέλεσμα της τελικής εικόνας παρουσιάζει αστοχίες. Ένας λόγος είναι ότι η αρχική ανάλυση της εικόνας απο 4000 επι 3000 της μετέτρεψα σε 1024 επι 768.

Επιπλέον για να πετύχουμε τη σωστή περικοπή των μαύρων εικονοστοιχείων χρειάζεται να κάνουμε δοκιμές με το ποια σημεία πρεπει να κρατήσουμε για να οριοθετήσουμε τις εικόνες που θα παράξουν το τελικό πανόραμα. Όπως αναφέρθηκε ανάλογα με το ποια σημεία θα πλαισιώσουν την κάθε εικόνα θα χάσουμε αντίστοιχη πληροφορία των αρχικών εικόνων. Συνεπώς είναι λογικό να χάνουμε μέρος της αρχικής εικόνας στο πανόραμα.Ακόμα όταν βάζουμε πολύ παραπάνω πίξελ για επέκταση της αρχικής τότε η εικόνα παραμορφώνεται.

Τέλος, παρατηρούμε ότι το αριστερό μέρος της εικόνας δεν επηρεάζεται από τα μαύρα πίξελ.Αυτό συμβαίνει διότι όταν κολλάμε την εικόνα ένα στην περιοχή για την εικόνα ένα ξεκινάμε από το σημείο (0,0). Αν αλλάζαμε το σημείο αυτό σε ένα πιο εσωτερικό ,εφαρμόζαμε μετασχηματισμό μετατόπισης,θα βλέπαμε και μαύρα πίξελ στο αριστερό και πάνω μέρος του πανοράματος όπως φαίνεται και στην εφαρμογή. Ωστόσο αν θέλαμε να φαίνονται θα έπρεπε να προσαρμοστεί ο κώδικας που βρίσκει τα σημεία που γίνονται οι εναλλαγές 0 και 255 στην binary εικόνα ώστε να τις δείχνει και σε εκείνες τις μεριές. Έχοντας στη κατοχή μας επιπλέον τους πίνακες αυτών των δυο σημείων θα οριοθετούσαμε καλύτερα την αρχική εικόνα και θα παίρναμε ένα αποτελέσμα κόντα σε αυτό της εφαρμογής.

SURF

Η διαφορά του SURF detector με του SIFT είναι ότι ο SURF εξάγει για κάθε keypoint 64 descriptors κι όχι 128,με τη βοήθεια της συνάρτησης integral.

Αυτό σημαίνει ότι είναι πιο 6 φορές πιο γρήγορος. Διαφορετικά δεν υπάρχουν ευδιάκριτες διαφορές μεταξύ των δύο περιγραφέων.

Οι αλλαγές στο κώδικα βρίσκονται σε σχόλια.Επίσης χρειάζονται αλλαγές στα σημεία οριοθέτησης.

Παράδειγμα πανόραμα SURF με l=3 και l = 1 Με μικρές αστοχίεςλόγω επιλογής σημείων.

Συμπεράσματα

Καταληκτικά, έμμεσα παρατηρούμε ότι όλη η διαδικασία είναι σειριακή. Στη περίπτωση που δεν γίνει καλή εύρεση ομόλογων σημείων και άρα σωστά ζευγάρια, δεν θα παραχθεί σωστά ο πίνακας ομογραφίας, και άρα η αντιστοίχιση θα έχει σφάλματα. Συνεπώς χρειάζεται εξονυχιστική εύρεση των σημείων ενδιαφέροντος ,καθώς αυτά θα είναι υπεύθυνα για τη ένωση των εικόνων. Στη περίπτωση των παραπάνω εικόνων το ταίριασμα ήταν σχετικά εύκολο καθώς οι φωτογραφίες ήταν σε γκρι κλίμακα, είχαν παρόμοιο φωτισμό και γενικά η οπτική του παρατηρητή δεν άλλαζε πολύ. Όλα αυτα βοήθησαν στην εύρεση αντιπροσωπευτικών ομόλογων σημείων, που σε διαφορετική περίπτωση μπορεί να προκαλούντουσαν προβλήματα.Επίσης πολύ σημαντική είναι η απομάκρυνση των μαύρων εικονοστοιχίων,η οποία οδηγεί σε απώλεια της αρχικής εικόνας. Τεχνικές όπως ένωση πινάκων διαφορετικού μεγέθους θα βοηθούσαν έτσι ώστε μετα την απομάκρυνση των μαύρων εικονοστοιχείων να συνενώναμε τους υποπίνακες της αρχικής εικόνας που αποβλήθηκαν και έτσι να μην χάναμε πιθανά keypoints και να βγάζαμε ένα καλύτερο αποτέλεσμα.