ΤΕΧΝΙΚΗ ΑΝΑΦΟΡΑ

`Για την Δεύτερη Εργασία στο μάθημα

«Όραση Υπολογιστών»

Καθηγητής: Ιωάννης Πρατικάκης

Στέλιος Μούσλεχ

ΑΜ:57382

31/11/2019,Ξάνθη

ΕΙΣΑΓΩΓΗ

Στην παρακάτω τεχνική αναφορά θα περγραφεί η λύση της δεύτερης εργασίας στο μάθημα «Όραση Υπολογιστών» αναλύοντας σε περιληπτικό επίπεδο το απαραίτητο θεωρητικό υπόβαθρο της επίλυσης καθώς θα γίνει και η ανάλυση μερικών ενδιάμεσων αποτελεσμάτων και προβλημάτων που αντιμετώπισα. Υπάρχουν συνολικά 5 αρχεία που αντιστοιχούν στην χρήση του SIFT για τις φωτογραφίες της εκφώνησης, στην χρήση του SURF για τις ίδιες φωτογραφίες, καθώς και άλλα 2 αρχεία για χρήση SIFT και SURF για δικές μου φωτογραφίες. Τέλος υπάρχει ένα αρχείο test2.py που δοκίμασα μια διαφορετική προσσέγγιση για να επιλύσω μερικά από τα προβλήματα που προέκυψαν

Τα ζητούμενα της παρούσας εργασίας είναι τα εξής :

- 1. Να παράξετε το πανόραμα που προέρχεται από τη σύνθεση τουλάχιστον τεσσάρων εικόνων χρησιμοποιώντας τους παρακάτω ανιχνευτές και περιγραφείς:
  - a. SIFT
  - b. SURF
- 2. Να προβληθούν τα πανοράματα που προέκυψαν με τις παραπάνω μεθοδολογίες, να συγκριθούν τόσο μεταξύ τους όσο και με το πανόραμα που θα παράξετε με τη χρήση του Image Composite Editor.
- 3. Να επαναλάβετε την παραπάνω διαδικασία χρησιμοποιώντας τουλάχιστον τέσσερις (4) δικές σας εικόνες

<u>ΥΠΟΔΕΙΞΗ</u>

1. Για το συνταιριασμό (matching) των σημείων ενδιαφέροντος θα πρέπει να υλοποιηθεί η μέθοδος “cross checking”. Κατά τη μέθοδο αυτή το «απλό» ταίριασμα εφαρμόζεται αμφίδρομα. Δηλαδή, βρίσκονται ταιριάσματα από την Εικόνα_1 προς την Εικόνα_2 κι έπειτα από την Εικόνα_2 προς την Εικόνα_1
   Τέλος διατηρούνται μόνο τα ταιριάσματα τα οποία προέκυψαν και στα 2 περάσματα, δηλαδή το ίδιο ζευγάρι να εμφανίζεται και στις 2 περιπτώσεις. Το ταίριασμα αυτό παρέχεται στην κλάση ‘BFMatcher’ της OpenCV αλλά δεν μπορείτε να το χρησιμοποιήσετε. Θα πρέπει να κάνετε δική σας υλοποίηση.

2. Για κάθε ζευγάρι εικόνων για το οποίο ελέγχετε το συνταιριασμό, να δείξετε τα ζευγάρια ομόλογων σημείων κλειδιών που προέκυψαν.

ΠΕΡΙΕΧΟΜΕΝΑ

1.**Θεωρητική ανάλυση και περιγραφή των μεθόδων υλοποίησης της εργασίας**

1.1Περιγραφή της γενικής μεθοδολογία – Προβλήματος

Θα δημιουργήσουμε ένα πανόραμα με βάση τις παρακάτω φωτογραφίες,

Εικόνα 1 yard-05.png

Εικόνα 2 yard-04.png

Εικόνα 3 yard-03.png

Εικόνα 4 yard-02.png

Αρχικά η μεθοδολογία για να ενώσουμε δύο φωτογραφίες (stitching) σωστά θα ακολουθηθεί η παρακάτω μεθοδολογία **(σε λεπτομέρεια θα αναλυθεί το κάθε βήμα ξεχωριστά παρακάτω)**

1. Για κάθε εικόνα θα εξάγουμε τα τοπικά χαρακτηριστικά και τους περιγραφείς τους (μέσω SIFT ή SURF)
2. Θα συγκρίνουμε τους περιγραφείς της μίας εικόνας με την άλλη για κάθε χαρακτηριστικό για να αντιστοιχήσουμε (match) ίδια χαρακτηριστικά
3. Με βάση τα matched features θα εξάγουμε ένα μετασχηματισμό προοπτικής υπολογίζοντας το H (Homography) όπου είναι ένας πίνακας 3χ3

4. Με βάση το H θα μετασχηματίσουμε την δεύτερη εικόνα για να ταιριάξει και να ενωθεί με την πρώτη

Στη συνέχεια αν θέλουμε να ενώσουμε και παραπάνω από 2 φωτογραφίες θα πρέπει να αντιμετωπίσουμε μερικά προβλήματα που προκύπτουν, κάτι που θα δούμε καλύτερα παρακάτω διαφορετικές προσεγγίσεις που δοκίμασα.

1.2 Ανίχνευση και περιγραφή τοπικών χαρακτηριστικών

Τα χαρακτηριστικά μιας εικόνας γνωστά και ώς σημεία ενδιαφέροντος (interest points) είναι δύσκολο να ορισθούν νοηματικά , ουσιαστικά αποτελούν σημεία κάποιο αντικειμένου σε μια εικόνα που θα μπορούσαμε να αναγνωρίσουμε (μερικές φορές και με το μάτι) αν παραμορφώσουμε την εικόνα μας. Αυτά τα σημεία είναι σημαντικά καθώς έτσι μπορούμε να ορίσουμε-αναγνωρίσουμε αντικείμενα να τα ακολουθήσουμε σε ένα frame εικόνων , να κάνουμε τριδιάστατή ανακατασκευή εικόνων, κ.α. καθώς και όπως στην δικιά μας περίπτωση να ενώσουμε εικόνες και να δημιουργήσουμε ένα πανόραμα . Ένα παράδειγμα τέτοιων σημείων είναι οι γωνίες

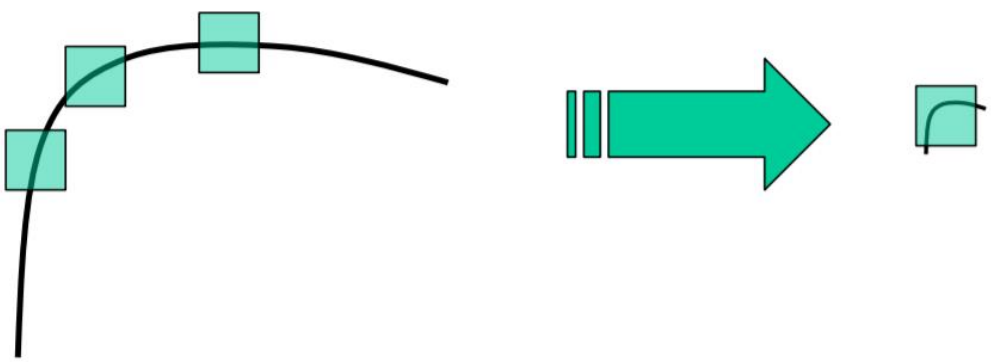
Τα κύρια μέρη των τοπικών χαρακτηριστικών είναι

1) Η ανίχνευση
    a. Η αναγνώριση της θέσης του σημείου ενδιαφέροντος
2) Η περιγραφή
    B. Ένα διάνυσμα που μας δίνει πληροφορίες για το σημεί αυτό

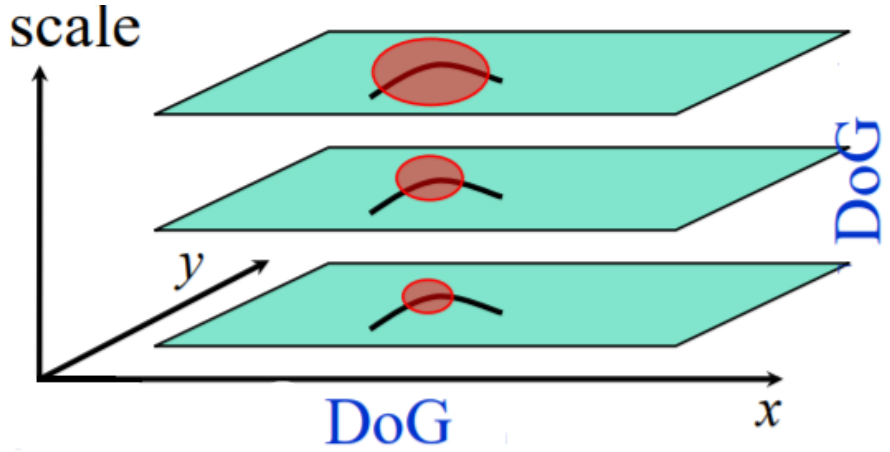
Στην δικιά μας περίπτωση για να γίνει η ανίχνευση και η περιγραφή των σημείων αυτών θα χρησιμοποιηθεί ο αλγόριθμος SIFT και SURF

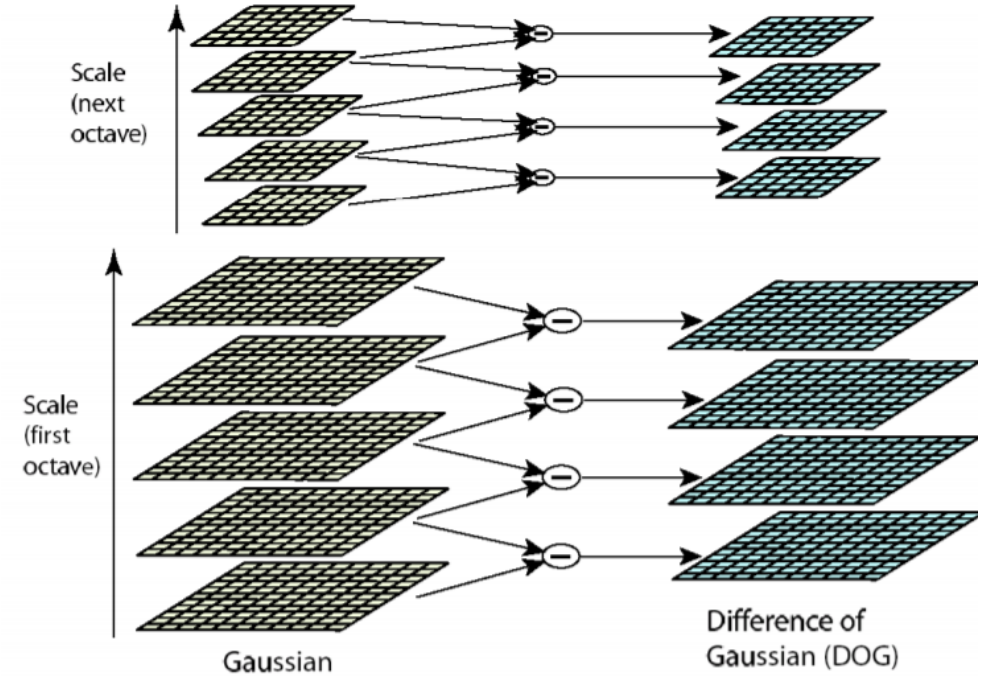
1.2.1 Scale-invariant feature transform (SIFT)

Ο SIFT είναι ένας αλγόριθμος για αναγνώριση και περιγραφή τοπικών χαρακτηριστικών , και έχει την πολύ σημαντική ιδιότητα ότι αναγνωρίζει χαρακτηριστικά ανεξαρτήτως κλίμακας κάτι σημαντικό καθώς σε άλλους όπως πχ ο Harris Detector η κλίμακα επηρεάζει την αναγνώριση (εικόνα)



Αυτήν η ανεξαρτησία από τη κλίμακα πραγματοποιείται βρίσκοντας τοπικά μέγιστα(ή ελάχιστα) από την διαφορά των Γκαουσιανών στο χώρο και στη κλίμακα





Μετά δημιουργείται ο περιγραφέας του Keypoint. Μια γειτονιά 16x16 γύρω από το βασικό σημείο έχει ληφθεί. Είναι χωρισμένο σε 16 υπο-τετράγωνα μεγέθους 4x4. Για κάθε υπο-ομάδα δημιουργείται ιστόγραμμα προσανατολισμού 8 bins. Επομένως, είναι διαθέσιμες συνολικά 128 τιμές. Παρουσιάζεται ως φορέας για τον σχηματισμό του βασικού περιγραφικού σημείου.

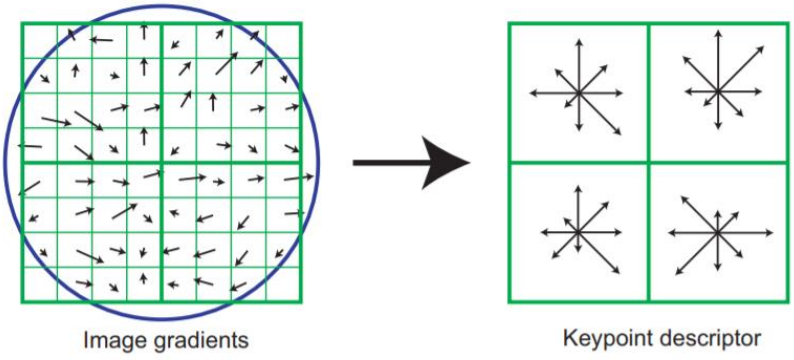


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

1.2.2 Speeded up robust features (SURF)

Καθώς το SIFT διαπιστώνεται ότι είναι αρκετά αργό, δημιουργήθηκε η ανάγκη για κάτι νέο πιο γρήγορο. Έτσι, κατασκευάστηκε το SURF το οποίο όπως υποδηλώνει και το όνομά του είναι ένα πιο γρήγορο SIFT (speeded-up version of SIFT). Χρησιμοποιείται κι αυτό, όπως και το SIFT, για εργασίες, όπως είναι η αναγνώριση αντικειμένων, η εγγραφή εικόνας, η ταξινόμηση, η 3D ανακατασκευή, κλπ. Εν μέρει εμπνέεται από τον περιγραφικό μετασχηματιστή μεταβλητής κλίμακας (SIFT) και σύμφωνα με τους κατασκευαστές του, η τυποποιημένη έκδοση του SURF είναι αρκετές φορές πιο γρήγορη και ισχυρή έναντι διαφορετικών μετασχηματισμών εικόνας σε σχέση με το SIFT. Το SURF ανιχνεύει κι αυτό σημεία ενδιαφέροντος, χρησιμοποιώντας μια ακέραια προσέγγιση του προσδιοριστή του ανιχνευτή Hessian blob, ο οποίος μπορεί να υπολογιστεί με 3 ακέραιες λειτουργίες χρησιμοποιώντας μια προ αναμιγμένη ολοκληρωμένη εικόνα (integral image). Η εικόνα μετατρέπεται σε συντεταγμένες, χρησιμοποιώντας την τεχνική πυραμίδων πολλαπλών αναλύσεων, για να αντιγράψει την αρχική εικόνα με σχήμα πυραμιδικού Gauss ή Laplacian Pyramid για να αποκτήσει μια εικόνα με το ίδιο μέγεθος αλλά με μειωμένο εύρος ζώνης. Αυτό επιτυγχάνεται με ειδικό εφέ θολώματος στην αρχική εικόνα, που ονομάζεται κλίμακα-χώρος και εξασφαλίζει ότι τα σημεία ενδιαφέροντος έχουν αμετάβλητη κλίμακα. Το SURF προσεγγίζει το Laplacian of Gaussian με το Box Filter. Ένα μεγάλο πλεονέκτημα αυτής της προσέγγισης είναι ότι

η συνέλιξη με το Box Filter μπορεί εύκολα να υπολογιστεί με τη βοήθεια των integral images και μπορεί να γίνει παράλληλα για διαφορετικές κλίμακες

1.2.3 Matching

Για να γίνει η αντιστοίχιση των εικόνων πρέπει να αρχικά να ταιριάξουμε (match) τα KeyPoints μιας εικόνας με μια άλλη.

Στην βασική περίπτωση για κάθε keypoint της μίας εικόνας συγκρίνουμε την απόσταση του descriptor του με όλους τους descriptor της άλλης εικόνας και τον αντιστοιχούμε με αυτόν με την μικρότερη απόσταση. Η απόσταση μπορεί να είναι είτε η απόσταση Manhattan(L1):

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

where $(\mathbf{p}, \mathbf{q})$ are vectors

$$\mathbf{p} = (p_1, p_2, \ldots, p_n) \text{ and } \mathbf{q} = (q_1, q_2, \ldots, q_n)$$

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Εφόσον έχουμε τα matches θα πρέπει να μετασχηματίσουμε την δεύτερη εικόνα ώστε να ταιριάξει στην πρώτη. Όπως βλέπουμε παρακάτω μια απλή μετακίνηση της δεύτερης εικόνας δεν είναι αρκετή για να ταιριάξουν οι εικόνες.

Translations are not enough to align the images

Χρειάζεται να κάνουμε projective μετασχηματισμό.

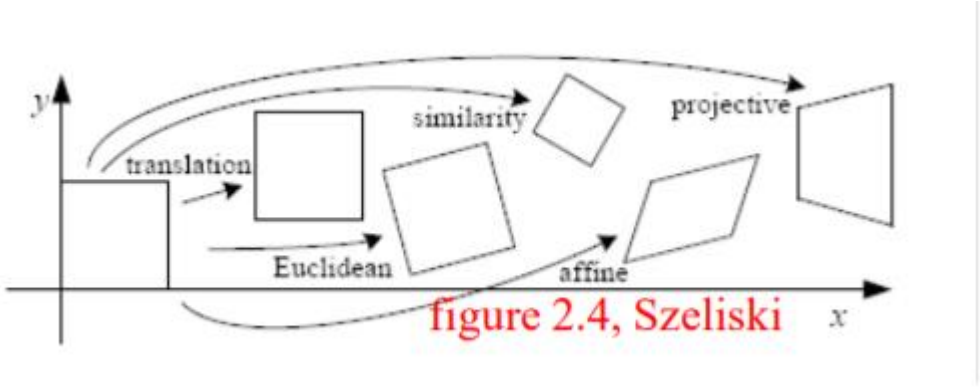


figure 2.4, Szeliski

Για να υπολογίσουμε το μετασχηματισμό προοπτικής πρέπει να υπολογίσουμε την ομογραφία Η.Η χαρτογράφηση μεταξύ οποιωνδήποτε δύο επιφανειών προβολών με το ίδιο κέντρο προβολής ονομάζεται Ομογραφία και αντιπροσωπεύεται ως πίνακας 3χ3 σε ομογενή συντεταγμένες

Επαναλαμβάνοντας την ίδια διαδικασία για παραπάνω φωτογραφίες μπορούμε να φτιάξουμε ένα πανόραμα.

2.Ανάλυση του κώδικα- εμφάνιση ενδιάμεσων αποτελεσμάτων

Αρχικά φορτώνουμε τις φωτογραφίες μας στην OpenCV της οποίες θα κάνουμε και scale down για να μπορούμε να χωρέσουμε καλύτερα το αποτέλεσμα σε μια εικόνα και γιατί με τις δικές μας μεθόδους η υλοποίηση με μεγάλες εικόνες διαρκεί πολύ χρόνο στην εκτέλεση (περίπου 4 λεπτά στο λάπτοπ μου) Οπότε κάνω scale down στο 50%. Μέσω της εντολής cv2.resize(src, dsize[, dst[, fx[, fy[, interpolation]]]])

Π.Χ για την 1η εικόνα

```
img1 = cv.imread('yard-05.png', cv.IMREAD_GRAYSCALE)
img1 = cv.resize(img1,None,fx=0.5,fy=0.5)
```

Στην συνέχεια δημιουργούμε ένα αντικείμενο του detector μας

Για τον sift:

```
sift = cv.xfeatures2d_SIFT.create()
```

Για τον SURF:

```
surf = cv.xfeatures2d_SURF.create()
```

και έπειτα για κάθε εικόνα υπολογίζω τα keypoints και τους περιγραφείς τους. Ο πίνακας των οποίων έχει μέγεθος (αριθμός των keypoints x 128) (το 128 λόγω των bins που περιγράψαμε παραπάνω (16x8).

Π.χ. για την πρώτη εικόνα με sift.

```
kp1 = sift.detect(img1)
desc1 = sift.compute(img1, kp1)
```

Τώρα για να κάνω το matching keypoints έφτιαξα δύο συναρτήσεις την match(d1,d2) όπου βασισμένη στον κώδικα του εργαστηρίου όρισα και μια συνάρτηση για να υπολογίζω την απόσταση L2 από ένα keypoint της μίας εικόνας στην άλλη εικόνα και επιστρέφει το index του kp με την μικρότερη απόσταση.

Για να γίνει το crossCheck ελέγχω για το index που επιστρέφει η πρώτη κλήση δηλαδή για κάθε kp της 1 εικόνας αν το kp 2 με την ελάχιστη απόσταση της δεύτερης εικόνας έχει για ελάχιστη απόσταση το kp της εικόνας 1.

Παρακάτω ο κώδικας για τις παρακάτω συναρτήσεις:

```
def match(d1, d2):
    n1 = d1.shape[0]
```

```
    n2 = d2.shape[0]

    matches = []
    for i in range(n1):
        fv1 = d1[i, :]
        # L2 distance
        i1,mindist2=closestDistanceIndex(fv1,d2)
        fv2= d2[i1, :]
        i2,_=closestDistanceIndex(fv2,d1)

        if i2==i:
            matches.append(cv.DMatch(i, i1, mindist2))

    return matches


def closestDistanceIndex(one_desc1,all_desc2):
    diff = all_desc2 - one_desc1
    diff=np.square(diff)
    distances = np.sum(diff, axis=1)
    distances = np.sqrt(distances)
    index = np.argmin(distances)
    mindistance = distances[index]
    return index , mindistance
```

Στη συνέχεια πρέπει να αρχίσω να υπολογίζω τους πίνακες ομογραφίας για να κάνω τις ενώσεις.

Εδώ να σημειωθεί ότι επέλεξα να ενώσω την 1 με την 2 την 3 με την 4 και την 1-2 με την 3-4 μετασχηματίζοντας πάντα την δεξιά με σταθερή την αριστερά για να μπορέσω να βάλω σωστά τις 2 εικόνες μαζί.

Η επιλογή αυτή μου έφερε λίγο λιγότερη παραμόρφωση στην τέρμα δεξιά λόγω του τρόποι υπολογισμού της ομογραφίας με αποτέλεσμα να την προτιμήσω στο τέλος καταλαβαίνοντας βέβαια ότι αν είχα παραπάνω εικόνες θα ήταν πιο “modular” να κάνω stitch συνέχεια την πιο δεξιά εικόνα στις προηγούμενες

Για την 1-2 και και 3-4 αρκεί να υπολογίσουμε την ομογραφία , με βάση τον κώδικα του εργαστηρίου

```
matches1 = match(desc1[1], desc2[1]) #Καλούμε τη συνάρτηση match
img_pt1 = np.array([kp1[x.queryIdx].pt for x in matches1])
img_pt2 = np.array([kp2[x.trainIdx].pt for x in matches1])
M, mask = cv.findHomography(img_pt2, img_pt1, cv.RANSAC)
# Βρίσκει πώς πρέπει να μετατραπεί η πρώτη για να "ταιριάξει" με τη δευτερη
img5= cv.warpPerspective(img2, M, (img1.shape[1]+500, img1.shape[0]+500))
```

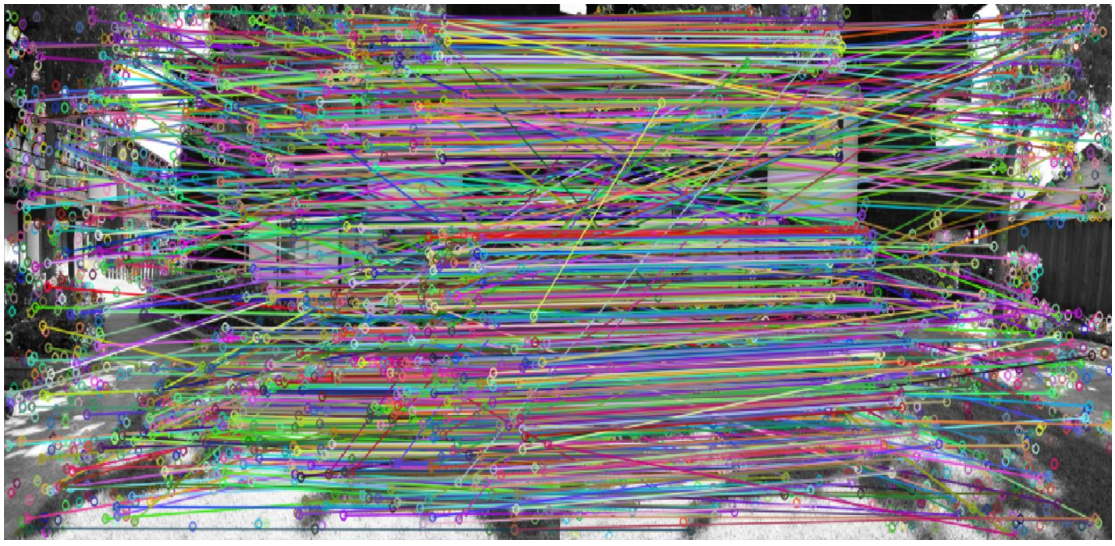
και για να μπει η αριστερά ίδια πάνω αριστερα στην νέα εικόνα μας

```
img5[0: img2.shape[0], 0: img2.shape[1]] = img1
```

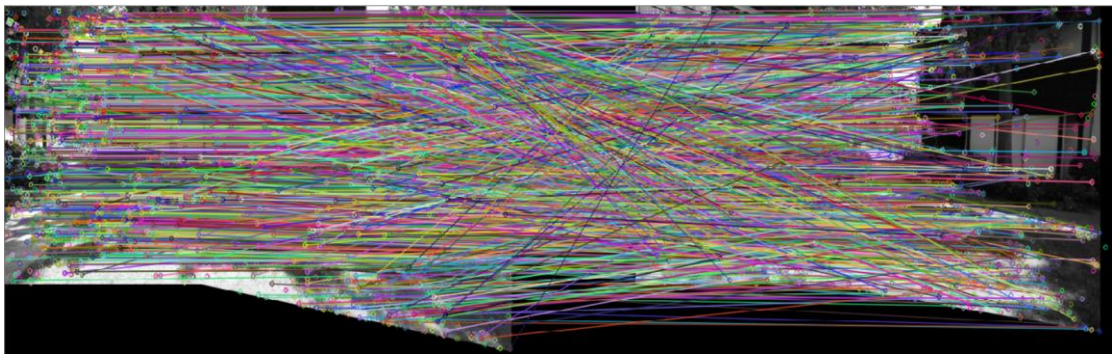
Τέλος μπορούμε να δόυμε για κάθε ζεύγος εικόνων τα ζεύγη των matches.

Για την εικόνα 1 με την εικόνα 2

Για την εικόνα 3 με την εικόνα 4



Για τις εικόνες 1-2 με 3-4:



Τώρα πηγαίνοντας στο stitching παραπάνω εικόνων(1-2 με 3-4) πρέπει να λύσω ένα πρόβλημα που προκύπτει αν απλά έκανα το ίδιο αν το έκανα όπως πριν δημιουργείται ένα πλαίσιο

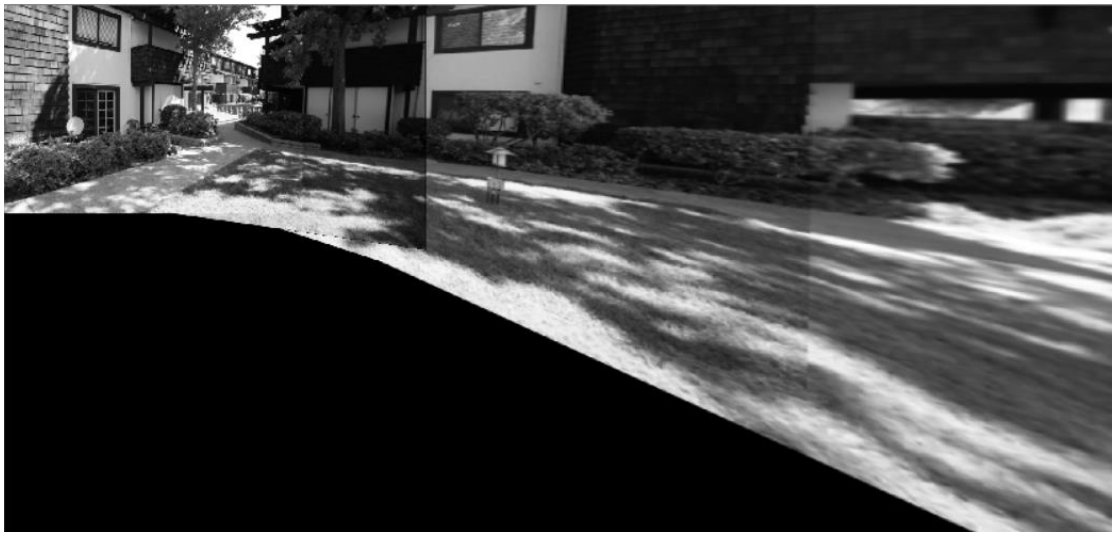
Οπότε αποφάσισα να τοποθετήσω μόνο το κομμάτι της εικόνας που δεν είναι μαύρο δηλαδή:

```
for x in range (img5.shape[0]):
    for y in range (img5.shape[1]):
        if(img5[x,y]!=0):
            img7[x,y]=img5[x,y]
```

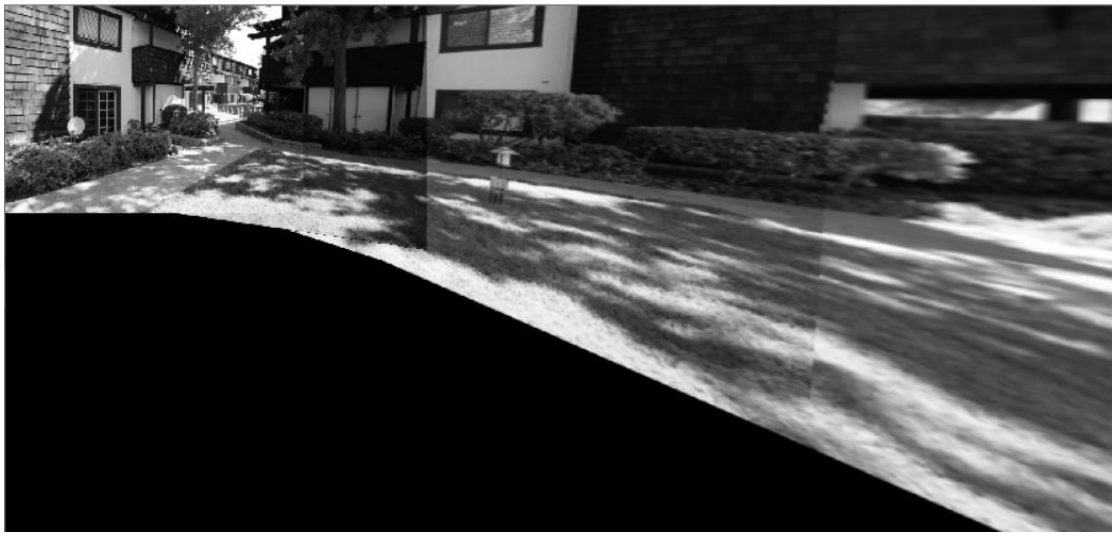
και το αποτέλεσμα είναι πολύ καλύτερο (υπάρχει παρακάτω) στην ανάλυση αποτελεσμάτων

3.Σύγκριση τελικών πανοραμάτων με τα αντίστοιχα του προγράμματος -Σχολιασμός Προβλημάτων επιλύσεων

Με την χρήση SIFT και με τον τρόπο που έδειξα παραπάνω:



Με την χρήση SURF και με τον τρόπο που έδειξα παραπάνω:



Ανάμεσα σε SIFT και SURF δεν έχω κάποια εμφανή διαφορά

Με το Image Composite Editor:

Παρατηρώ το αποτέλεσμα εδώ είναι πολύ καλύτερο δεν έχουμε αυτήν την προοπτική παραμόρφωση όπου τα δεξιά φαίνονται πιο κοντά και τα αριστερά πιο μακριά με αποτέλεσμα το δεξί κομμάτι να είναι και πιο stretched και zoomed.

Θεωρώ αυτό προκύπτει εξαιτίας του τρόπου που υπολογίζω το homography κάθε φορά αφού κοιτάω προς τα αριστερά μια ήδη παραμορφωμένη εικόνα και υπολογίζω με βάση μόνο αυτή. Το φαινόμενο αυτό ίσως φαίνεται και χειρότερο αν οι εικόνες μου παρουσιάζουν αντικείμενα με πολλές διαφορετικές αποστάσεις (πχ πιο κοντά στα πλάγια).

Βλέποντας αυτά τα αποτελέσματα αποφάσισα να προσπαθήσω να κάνω warp προς τα αριστερά την 1-2 και να κολλήσω εκεί την 3-4 δημιουργώντας μια μάσκα (με erosion για να μην έχω μαύρα) για να τοποθετήσω . Το αποτέλεσμα εμφανισιακά ήταν καλύτερο

από θέμα προοπτικής αλλά έχανα κάποια πληροφορία από τα αριστερά . Η προσπάθεια μου αυτή για τις εικόνες της εργασίας είναι στο test2.py και το αποτέλεσμα:

Με μια διαφορετική μου προσέγγιση

Εδώ σκέφτηκα να πολλαπλασιάσω τον πίνακα ομογραφίας με έναν πίνακα για να κάνει translate η εικόνα η αριστερή προς τα μέσα και να φαίνεται όλη αλλά μετά πρέπει να αλλάξω τον τρόπο που κολλάει η δεξιά εικόνα και στον χρονικό πλαίσιο της εργασίας δεν μπόρεσα να το κάνω να εμφανίζεται σωστά με αυτόν τον τρόπο

Αντίστοιχα τα αποτελέσματα για δικές μου φωτογραφίες (σε grayscale)

Με την χρήση SIFT

Με την χρήση της SURF

Με το Image Composite Editor:

Εδώ και πάλι βλέπουμε πολύ καλύτερο αποτέλεσμα με την χρήση του λογισμικού και αντιμετωπίζουμε ο ίδιο πρόβλημα σε μικρότερο βαθμό βέβαια λόγω του γεγονότος ότι η επιφάνεια της εικόνας είναι πιο μακριά από

4.ΠΗΓΕΣ-ΑΝΑΦΟΡΕΣ

[1] οι εικόνες και το περισσότερο κείμενο είναι βασισμένο στις διαφάνειες του μαθήματος :

- Μάθημα 6
- Μάθημα 7

[2]Οι διαφάνειες και ο κώδικας του μαθήματος από τα εργαστήρια

- Μάθημα 6
- Μάθημα 7

[3]Opencv tutorial SIFT- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html

[4] Opencv tutorial SURF - https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html