

**Ψηφιακή Επεξεργασία Εικόνας**  
**Εργασία 1**

Κουτρουμπής Γεώργιος, ΑΕΜ: 9668  
2022

## Περιεχόμενα

<b>1 Περιστροφή Εικόνας</b>	<b>3</b>
<b>2 Τοπικός Περιγραφέας</b>	<b>5</b>
2.1 Βασική Έκδοση . . . . .	5
2.2 Upgrade Έκδοση . . . . .	7
2.3 Naive SIFT . . . . .	9
<b>3 Harris Corner Detector</b>	<b>10</b>
<b>4 Συνένωση Εικόνων</b>	<b>12</b>

## 1 Περιστροφή Εικόνας

Για την περιστροφή εικόνας ακολουθείτε η λογική για κάθε πίξελ της τελικής περιστραμένης εικόνας, πραγματοποιείται η αντίστροφη περιστροφή, για να αντιστοιχηθεί σε πίξελ της αρχικής εικόνας. Με αυτό τον τρόπο αποφεύγονται τα κενά στην περιστραμένη εικόνα, καθώς θα μπορούσε να μην υπάρχει πίξελ της αρχικής εικόνας που να αντιστοιχεί σε κάποιο πίξελ της περιστραμένης, μετά την περιστροφή.

Αρχικά, περιστρέφουμε το πιο ακραίο πίξελ της αρχικής εικόνας με τον εξής τρόπο:

$$Y = height|\cos(\theta)| + width|\sin(\theta)|$$

$$X = height|\sin(\theta)| + width|\cos(\theta)|$$

ώστε να βρούμε το μέγεθος που θα χρειάζεται να έχει η περιστραμένη εικόνα. Γίνεται χρήση αυτού του τύπου, καθώς το σχήμα της εικόνας είναι παραλληλόγραμμο, το οποίο περιστρέφεται γύρω από το κέντρο του, δύο από τις γωνίες του θα είναι απλά mirrored σε σχέση με το κέντρο του, οπότε έτσι βρίσκουμε τις μέγιστες συντεταγμένες που θα χρειαστούν, για να χωρέσει όλη η περιστραμένη εικόνα.

Έχουν υλοποιηθεί 2 τρόποι περιστροφής:

- Ο απλός τρόπος κατά τον οποίο για κάθε πίξελ της περιστραμένης εικόνας, υπολογίζεται η αντίστροφη περιστροφή ως εξής:

$$yrot = [-(x - centerX)\sin(\theta) + (y - centerY)\cos(\theta)] + ogCenterY$$

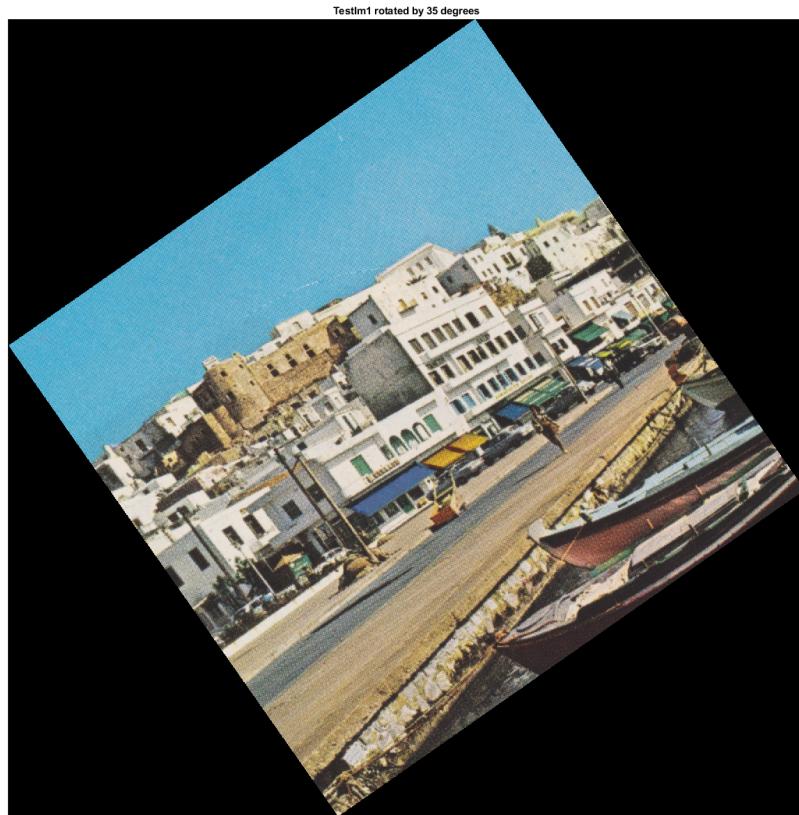
$$xrot = [(x - centerX)\cos(\theta) + (y - centerY)\sin(\theta)] + ogCenterX$$

όπου τα x,y αντιστοιχούν στο σημείο της περιστραμένης εικόνας, και το xrot, yrot, στο αντίστοιχο στοιχείο της αρχικής. Εφόσον περιστρέφουμε με κέντρο την περιστραμένη εικόνα, αφαιρούμε το κέντρο της περιστραμένης εικόνας, και αφού γίνει η περιστροφή, προσθέτουμε τις συντεταγμένες του κέντρου της αρχικής εικόνας, για να είναι τα yrot, xrot σύμφωνα με το σύστηματα συντεταγμένων της αρχικής.

Έπειτα, εφόσον το σημείο p=[xrot, yrot] είναι εντός της αρχικής εικόνας, υπολογίζεται η τιμή του πίξελ p'=[x, y] με χρήση του bilinear interpolation της εκφώνησης, δηλαδή ως ο μέσος όρος των ορθοκανονικών γειτόνων του.

- Μία vectorized έκδοση της περιστροφής κατά την οποία μαζικά τα πίξελ της περιστραμένης εικόνας περιστρέφονται (με την ανάποδη περιστροφή), και δίνεται στα πίξελ της περιστραμένης εικόνας πάλι η bilinear interpolated τιμή του αρχικού πίξελ, από μια νέα εικόνα που έχει ήδη δημιουργηθεί με όλα τα πίξελ να έχουν την bilinear interpolated τιμή τους.

Ως προεπιλεγμένος έχει επιλεχθεί ο 2ος τρόπος, καθώς ο πρώτος είναι χρονοβόρος. Παρακάτω παρουσιάζονται αποτελέσμα της στροφής της TestIm1.png κατά  $\theta_1 = 35^\circ$ ,  $\theta_2 = 222^\circ$ .



Εικόνα 1. Η εικόνα *TestIm1* περιστραμένη κατά 35 μοίρες



Εικόνα 2. Η εικόνα TestIm1 περιστραμένη κατά 222 μοίρες

## 2 Τοπικός Περιγραφέας

### 2.1 Βασική Έκδοση

Στη βασική έκδοση του περιγραφέα, για ένα πίζελ παίρνουμε έναν αριθμό από ακτίνες και σχεδιάζουμε ομόκεντρους κύκλους, παίρνοντας  $N$  σημεία γύρω από το σημείο για κάθε ακτίνα. Ένα πίζελ ή από τα  $N$  σημεία βρίσκεται σε γωνία  $\frac{2pi}{N} * n$  και σε ακτίνα  $r$  από το πίζελ που μας ενδιαιφέρει. Παίρνοντας τον μέσο όρο των σημείων πάνω στον εκάστοτε κύκλο, το σημείο χαρακτηρίζεται από ένα διάνυσμα με τόσα στοιχεία όσες οι ακτίνες που πήραμε. Όλες οι τιμές που ακολουθούν είναι αφού η εικόνα μετατράπηκε σε double.

Για το πίζελ  $p = [100 \ 100]$  στην TestIm1, καθώς και του αντίστοιχου σημείου στις εικόνες περιστραμμένες κατά  $\theta_1, \theta_2$ :

P	P ( $\theta=35$ )	P ( $\theta=222$ )
0.5890931	0.5852941	0.5909314
0.5800245	0.5849265	0.5953431
0.5867647	0.5895833	0.5963235
0.6044118	0.5957108	0.5936275
0.6012255	0.5977941	0.5894608
0.5883578	0.5938725	0.5873775
0.5772059	0.591299	0.5884804
0.5884804	0.5957108	0.5914216
0.5979167	0.5973039	0.5901961
0.5981618	0.5996324	0.5952206
0.5784314	0.5925245	0.5895833
0.586152	0.5958333	0.5894608
0.6006127	0.5982843	0.5926471
0.6196078	0.6031863	0.6131127
0.6088235	0.5973039	0.6156863
0.6051471	0.6029412	0.6170343

Εικόνα 3. Τοπικός περιγραφέας για το πίξελ ρ στην αρχική εικόνα και των αντίστοιχων πίξελ στις περιστραμμένες εικόνες

Για τα πίξελ  $q1 = [200, 200]$  και  $q2 = [202, 202]$ :

q1	q2
0.619	0.617
0.62	0.616
0.623	0.61
0.623	0.609
0.622	0.611
0.613	0.614
0.607	0.618
0.608	0.616
0.612	0.623
0.62	0.618
0.625	0.621
0.622	0.618
0.62	0.622
0.62	0.621
0.616	0.618
0.615	0.626

Εικόνα 4. Τοπικός περιγραφέας για τα πίξελ q1, q1

## 2.2 Upgrade Έκδοση

Για την upgrade έκδοση του περιγραφέα ακολουθήθηκε η λογική του βασικού τοπικού περιγραφέα, με μερικές αλλαγές. Αρχικά, ως όρισμα του περιγραφέα ζητείται πλέον εικόνα που έχει περαστεί από Gaussian Blur Filter (στις δοκιμές, για  $\sigma = 1$  είχαμε καλά αποτελέσματα) και το διάνυσμα του περιγραφέα κανονικοποιείται πριν επιστραφεί. Αυτό έχει ως αποτέλεσμα ο περιγραφέας να μην επηρεάζεται πολύ από τις αλλαγές στην φωτεινότητα και στην αντίθεση της εικόνας.

Για το πίξελ  $p = [100 \ 100]$  στην TestIm1, καθώς και του αντίστοιχου σημείου στις εικόνες περιστραμμένες κατά  $\theta_1, \theta_2$ :

P	P ( $\theta=35$ )	P ( $\theta=222$ )
0.2477238	0.2475118	0.2482972
0.2472804	0.2475619	0.2488272
0.2487559	0.2486557	0.249997
0.2512705	0.2499462	0.2492365
0.2502791	0.2507682	0.2492849
0.248291	0.2504015	0.2486582
0.2468629	0.2506361	0.24859
0.2483953	0.2508091	0.2493433
0.2506335	0.250812	0.2489855
0.2500436	0.2497204	0.2488473
0.2468759	0.2483571	0.2484119
0.2481497	0.2486477	0.2479324
0.2511895	0.2498732	0.2499715
0.2555273	0.251483	0.2535692
0.254471	0.2518006	0.254516
0.2540248	0.2529452	0.2553706

Εικόνα 5. Upgraded τοπικός περιγραφέας για το πίξελ ρ στην αρχική εικόνα και των αντίστοιχων πίξελ στις περιστραμμένες εικόνες

Για τα πίξελ q1 = [200, 200] και q2 = [202, 202]:

q1	q2
0.251	0.25
0.25	0.249
0.251	0.248
0.251	0.248
0.251	0.248
0.249	0.249
0.248	0.25
0.247	0.25
0.249	0.251
0.251	0.25
0.252	0.251
0.251	0.251
0.251	0.251
0.25	0.251
0.249	0.251
0.248	0.251

Εικόνα 6. Upgraded τοπικός περιγραφέας για τα πίξελ q1, q1

### 2.3 Naive SIFT

Έγινε επίσης προσπάθεια για μια απλή έκδοση του SIFT περιγραφέα. Για το πίξελ ενδιαφέροντος, αρχικά υπολογίζεται σε μια γειτονία (π.χ. 5x5) γύρω από το σημείο το ιστόγραμμα των παραγώγων των πίξελ (History of Gradients, HoG). Δηλαδή, για κάθε πίξελ παίρνεται η διαφορά του intensity των δεξιά - αριστερά πίξελ για την παράγωγο στον x, και των πάνω-κάτω για την παράγωγο στον y, και υπολογίζεται το μέτρο (ως η τετραγωνική ρίζα του αυθοίσματος των τετραγώνων των παραγώγων) και η γωνία (ως η atan της παραγώγου στο y ως προς την παράγωγο στο x).

$$\begin{aligned} I_x &= I(x+1, y) - I(x-1, y) \\ I_y &= I(x, y+1) - I(x, y-1) \\ magnitude &= \sqrt{I_x^2 + I_y^2} \\ theta &= \text{atan2}(I_y / I_x) \end{aligned}$$

Δημιουργείται έπειτα ένα ιστόγραμμα 36 θέσεων, όπου κάθε bin έχει 10 μοίρες. Για κάθε πίξελ, προστίθεται το μέτρο της παραγώγου του στο αντίστοιχο bin (πχ αν ένα πίξελ έχει παράγωγο με  $theta = 120^\circ$ ,  $magnitude = 0.3$ , θα προστιθόταν στο 2o bin το 0.3. Αφού γίνει η διαδικασία αυτή για όλη τη γειτονία του πίξελ, θεωρούμε ως κυρίαρχη φορά (dominant orientation) αυτή που στο ιστόγραμμα

έχει την υψηλότερη τιμή. Πχ αν το 3x bin είχε την υψηλότερη τιμή θα θεωρούσαμε τη γωνία 30° ως κυρίαρχη φορά.

Στη συγκεριμένη υλοποίηση, σχηματίζεται ένα 9x9 grid γύρω από το πίξελ, και αυτό διαρείται σε 3x3 grids των 3x3. Ακολουθείτε η ίδια διαδικασία για κάθε 3x3 grid όπως παραπάνω, με τη διαφόρα πως αντί να πάρουμε την πληροφορία για το πίξελ στο grid, πάρουμε την πληροφορία αφού περιστρέψουμε αυτό και τους ορθοκανονικούς του γείτονες, σύμφωνα με την κυρίαρχη φορά. Έτσι ο περιγραφέας είναι rotation invariant. Τελικά, ο περιγραφέας χαρακτηρίζεται από 9x36 = 323 στοιχεία.

Παρόλα αυτά, ενώ σε μεμονομένα σημεία τα αποτελέσματα είναι ικανοποιητικά, η χρήση του περιγραφέα για την συνένοση εικόνων δεν επέφερα καλά αποτελέσματα, οπότε δεν έγινε χρήση του.

Λόγω του μεγάλου μήκους του διανύσματος του περιγραφέα, παραδείγματα τιμών του μπορούν να βρεθούν τρέχοντας τον κώδικα στο main.m, και δεν παρατίθονται εδώ.

### 3 Harris Corner Detector

Υλοποιήθηκε αναγνώριση γωνιών σύμφωνα με τον αλγόριθμο του Harris. Για να βρεθεί εάν ένα πίξελ ανήκει σε γωνία, πρέπει να υπολογιστεί η τιμή R για το συγκεκριμένο πίξελ. Αυτό γίνεται σχηματίζοντας έναν πίνακα M, που έχει ως στοιχεία στοιχεία τα ανθροίσματα των πινάκων του γινομένου της παραγώγου κατά τον x άξονα με τον εαυτό της σε γειτονιά γύρω από το σημείο, αφού περαστεί από Gaussian φίλτρο, και αντίστοιχα για την παραγώγο στον y αλλά και για το γινόμενο της παραγώγου στον x και στον y. Δηλαδή, πρώτα βρίσκονται οι παράγωγοι

κατά x και y, μέσω convolution της εικόνας με  $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$  και  $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

αντίστοιχα, και προκύπτουν οι παράγωγοι  $I_x, I_y$ . Μετά υπολογίζονται τα element-wise γινόμενα  $I_x * I_x, I_y * I_y, I_x * I_y$ , τα οποία μετά γίνονται convolve με ένα Gaussian φίλτρο, και προκύπτουν τα  $S_{xx}, S_{yy}, S_{xy}$ . Έχοντας ως κέντρο το πίξελ που μας ενδιαφέρει, και παίρνοντας μια γειτονιά του πίξελ (πχ 3x3) με κέντρο αυτό, και ανθροίζοντας τις τιμές των γειτονικών πίξελ για τα  $S_{xx}, S_{yy}, S_{xy}$ , ορίζεται ο πίνακας M ως εξής:

$$M = \begin{bmatrix} \sum S_{xx} & \sum S_{xy} \\ \sum S_{xy} & \sum S_{yy} \end{bmatrix}$$

$S_{xx}, S_{yy}, S_{xy}$  να αναφέρονται στους υποπίνακες, που περιέχουν τις τιμές για την γειτονιά.

Τελικά η τιμή R υπολογίζεται:

$$R = \det(M) - k * \text{Trace}(M)$$

όπου  $\det$  η ορίζουσα του M,  $\text{Trace}$  το ίχνος του M, και k μία σταθερά, συνήθως από 0.04 έως 0.06.

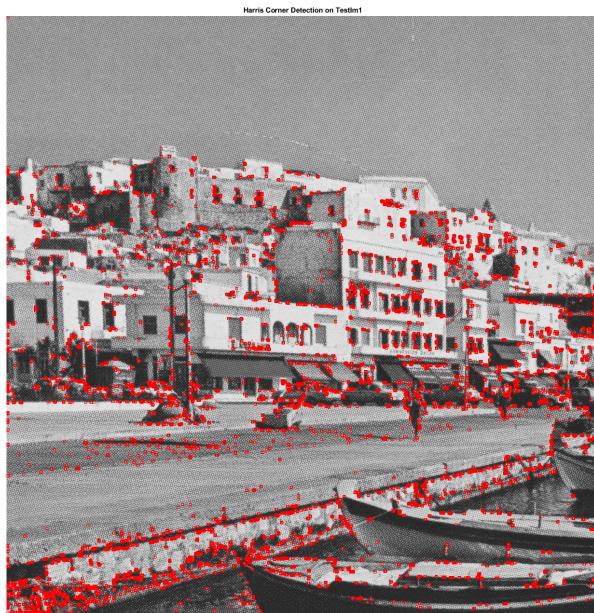
Έχει υλοποιηθεί συνάρτηση `isCorner`, η οποία παίρνει σχηματίζει μία υπο-εικόνα, γύρω από τη γειτονιά του πίξελ, και πραγματοποιεί τους παραπάνω υπολογισμούς.

Αν η τιμή του  $R$  είναι πάνω από ένα όριο, επιστρέφει ότι το πίξελ είναι γωνία.

Τλοποιήθηκε και μια 2η συνάρτηση `myIsCorner`, η οποία παίρνει ως όρισμα τους πίνακες  $S_{xx}$ ,  $S_{yy}$ ,  $S_{xy}$  για όλη την εικόνα, οπότε τελικά στην συνάρτηση `myIsCorner` γίνονται μόνο οι υπολογισμοί που αφορούν τον πίνακα  $M$ , αφού παρθούν υποπίνακες 3x3 των  $S_{xx}$ ,  $S_{yy}$ ,  $S_{xy}$  γύρω από το πίξελ.

Η συνάρτηση `myDetectHarrisFeatures` μπορεί να λειτουργήσει και με τις 2, όμως γίνεται χρήση της `myIsCorner`, λόγω μειωμένης υπολογιστικής πολυπλοκότητας. Εάν χρειαστεί να γίνει χρήση της `isCorner`, θα πρέπει να αλλάξει και το όριο του  $R$ , ώστε να δίνει τα επιθυμητά αποτελέσματα. Η λειτουργία της `myDetectHarrisFeatures` είναι να εκτελέσῃ την `myIsCorner` για όλα τα πίξελ της εικόνας.

Παρακάτω παρουσιάζεται το αποτέλεσμα της `myDetectHarrisFeatures` στην `TestIm1`:



Εικόνα 7. Αποτέλεσμα της `myDetectHarrisFeatures`

**Σημείωση:** Επειδή το bilinear interpolation που ζητήθηκε στα πλαίσια της εργασίας λειτουργεί ως smoothing φίλτρο, για να τρέξει σωστά ο αλγόριθμος σε εικόνα που έχει περιστραφεί με την `myImgRotation`, πρέπει να αλλάξουν οι παράμετροι του αλγορίθμου

## 4 Συνένωση Εικόνων

Έχοντας δύο εικόνες Im1 και Im2, υλοποιήθηκε αλγόριθμος συνένωσης των δύο εικόνων, οι οποίες διαφέρουν κατά στροφή και μετατόπιση. Οι εικόνας αναφοράς χρησιμοποιήθηκε η εικόνα Im1 (όποια δηλαδή μπαίνει ως πρώτο όρισμα). Αρχικά, εντοπίζονται οι γωνίες στην εικόνα μέσω του myDetectHarrisFeatures στις 2 εικόνες. Τα σημεία που βρίσκονται σε γωνία, τα keypoints (σημεία ενδιαφέροντος) των 2 εικόνων, περιγράφονται μέσω του περιγραφέα μας. Στη συνέχεια, με έναν brute-force αλγόριθμο για ταίριασμα σημείων βάσει της περιγραφής τους, δηλαδή ελέγχοντας την ελάχιστη ευκλείδια απόσταση ενός keypoint στην εικόνα 1 με όλα τα keypoints στην εικόνα 2, εντοπίζονται όλα τα σημεία κοινά keypoints των 2 εικόνων. Για να υπολογιστεί η γωνία στροφής και η μετατόπιση της 2ης εικόνας σε σχέση με την 1η χρησιμοποιήθηκε η εξής λογική:

Από τα σημεία που ταιριάζουν αρκεί να πάρουμε 3 ζεύγη, και να φτιάξουμε έναν

$$\text{πίνακα } 3 \times 3 \text{ με τα σημεία της 1ης εικόνας ως εξής: } \begin{bmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ C_x & C_y & 1 \end{bmatrix} \text{ και έναν με τα}$$

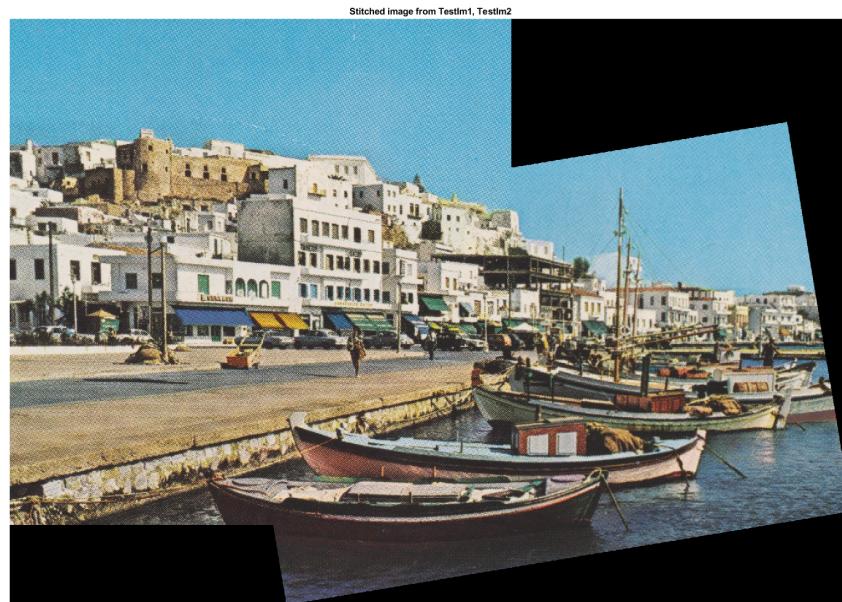
σημεία της 2ης:

$$\begin{bmatrix} P_x & P_y & 1 \\ Q_x & Q_y & 1 \\ R_x & R_y & 1 \end{bmatrix} \text{ θεωρώντας ως A,B,C τα σημεία της 1ης και ως P,Q,R τα σημεία της δεύτερης.}$$

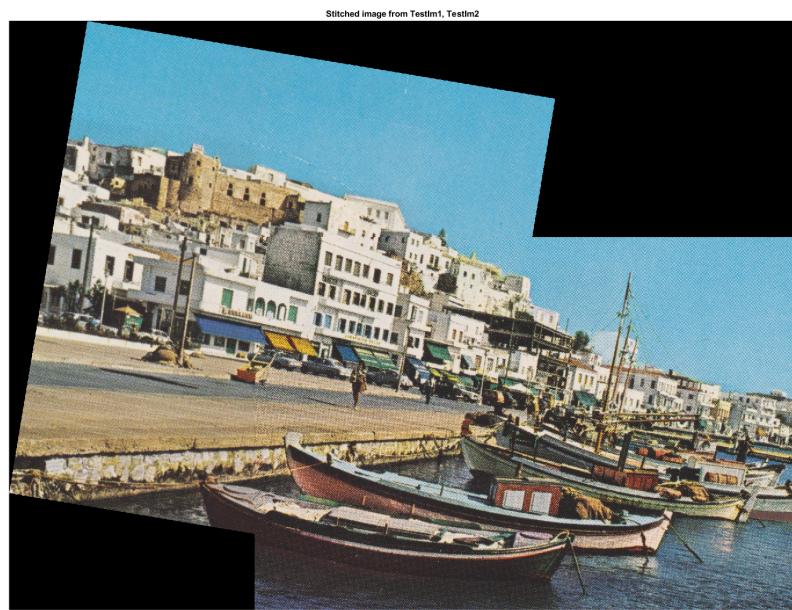
ΕΦόσον τα σημεία αυτά ταιριάζουν, είναι σαν τα ABC μέσω ενός πίνακα transform να έγιναν τα σημεία PQR, δηλαδή  $ABC * T = PQR$ , και για να βρούμε τον T αρκεί:  $T = ABC^{-1} * PQR$ . Επειδή όμως εμείς θέλουμε το αντίστροφο transform, δηλαδή από τα PQR να πάμε στα ABC αρκεί να πάρουμε το inverse του T. Ο T περιέχει τις πληροφορίες που χρειάζονται για την στροφή και μετατόπιση της 2ης εικόνας, σε σχέση με την 1η.

Σχηματίζουμε την ενωμένη εικόνα, τοποθετώντας την εικόνα που πάει αριστερά τέρμα αριστερά στην τελική εικόνα, και τοποθετώντας μετατοπισμένα την δεξιά εικόνα.

Παρακάτω παρουσιάζεται το αποτέλεσμα του myStitch έχοντας ως εικόνα αναφοράς την TestIm1 και μετά την TestIm2:



Εικόνα 8. Αποτέλεσμα της *myStitch* με εικόνα αναφοράς την *TestIm1*



Εικόνα 9. Αποτέλεσμα της *myStitch* με εικόνα αναφοράς την *TestIm2*

Παρακάτω παρουσιάζονται επίσης 2 εικόνες που άνηκαν στην ίδια, κόπηκαν και

περιστράφηκε η μία:



(a) Δοκιμαστική εικόνα a



(b) Δοκιμαστική εικόνα b

και παραχάτω παρουσιάζεται το αποτέλεσμα του myStitch, έχοντας ως εικόνα αναφοράς την δοκιμαστική εικόνα (a) και μετά την (b)



Εικόνα 11. Αποτέλεσμα της myStitch με εικόνα αναφοράς την a



Εικόνα 12. Αποτέλεσμα της *myStitch* με εικόνα αναφοράς την *b*

Παρατηρούμε μία αδυναμία του myImgRotation αλγορίθμου εδώ, που λόγω ότι στρογγυλοποιούμε προς τα κάτω, αλλά και με το bilinear interpolation που χρησιμοποιείται, όταν έχουμε στροφές κατά 180°ην μοίρες, υπάρχει περίπτωση να εμφανιστούν γραμμές μαύρων πίξελ στην άκρη. Επίσης, κατά τις δοκιμές, παρατηρήθηκε μεγαλύτερη ακρίβεια κατά την "συγκόλληση" των δύο εικόνων, με τη χρήση του upgraded descriptor έναντι του απλού.