ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

ΜΕΛΕΤΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΟΥ ΔΗΜΙΟΥΡΓΙΑΣ ΠΑΝΟΡΑΜΙΚΩΝ ΕΙΚΟΝΩΝ

Πτυχιακή εργασία της

Αικατερίνης Σαλιάρη (1975)

Επιβλέπων: Νικολαΐδης Αθανάσιος, Ε. Καθηγητής

ΣΕΡΡΕΣ, ΑΥΓΟΥΣΤΟΣ 2014

Υπεύθυνη δήλωση

Υπεύθυνη Δήλωση: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής ΤΕ του Τ.Ε.Ι. Κεντρικής Μακεδονίας.

Σύνοψη

Η δημιουργία πανοραμικών εικόνων είναι η διαδικασία κατά την οποία συνδυάζονται πολλαπλές εικόνες με μερική επικάλυψη θέας προκειμένου να παραχθεί μια εικόνα υψηλής ανάλυσης. Στη βιβλιογραφία έχουν προταθεί διάφορες τεχνικές δημιουργίας πανοραμάτων, εκ των οποίων άλλες βασίζονται στη φωτεινότητα, άλλες στο συχνοτικό πεδίο και άλλες σε χαρακτηριστικά. Στα πλαίσια της παρούσας πτυχιακής θα υλοποιηθεί τεχνική η οποία αρχικά εφαρμόζει αλγόριθμο ανίχνευσης γωνιών στην κάθε εικόνα. Στη συνέχεια, δημιουργείται ένας πίνακας ομοιότητας γωνιών δύο διαδοχικών εικόνων, βάσει του οποίου σχηματίζεται ένα αρχικό σύνολο από ζεύγη αντίστοιχων γωνιών. Τέλος, εφαρμόζονται πολλαπλοί περιορισμοί πάνω στα ζεύγη αυτά, με αποτέλεσμα να περιοριστεί το πλήθος των ζευγών. Τελικά, εφαρμόζεται ο αλγόριθμος RANSAC προκειμένου να βρεθεί ο γεωμετρικός μετασχηματισμός των σημείων διαδοχικών εικόνων και να δημιουργηθεί η πανοραμική εικόνα. Η απόδοση της τεχνικής θα ελεγθεί πάνω σε κατάλληλα σύνολα εικόνων που ανα δύο θα έχουν μερική επικάλυψη.

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος διαγραμμάτων

Ευχαριστίες

Ευχαριστώ τους γονείς μου για την υπομονή τους όλα αυτα τα χρόνια. Ευχαριστώ όλους τους καθηγητές μου για τις υπέροχες γνώσεις που μου μετάδωσαν (αν και αργά, πλέον κατάλαβα πόσο σημαντικές είναι). Τέλος ευχαριστώ τον επιβλέποντα καθηγητή μου, κ.Αθανάσιο Νικολαΐδη που με τον δικό του τρόπο με έκανε να εκτιμήσω την σημασία της προσωπικής προσπάθειας και έρευνας.

Ορισμοί

Grayscale Image Πρόκειται για μια εικόνα που περιέχει ως μοναδικό χρώμα διαφορετικές διαβαθμίσεις του γκρί. Στην επεξεργασία εικόνας χρησιμοποιούνται συχνά γιατί χρειάζονται λιγότερες πληροφορίες για τον προσδιορισμό ενός pixel. Οι εντάσεις των βασικών χρωμάτων RGB ενός pixel που χρειάζονται για να προσδιοριστεί, είναι ίσες μεταξύ τους και αυτο έχει ως αποτέλεσμα να χρειάζεται μόνο μια τιμή για προσδιορίσει ένα pixel και όχι τρεις.

Intensity Ένταση ή φωτεινότητα εικόνας. Κάθε εμφάνιση εικονοστοιχείου ελέγχεται από την ένταση τριών δεσμών φωτός (RGB). Στην επεξεργασία εικόνας και συγκεκριμένα στην παρούσα εργασία ταυτίζεται με τον όρο grayscale, δηλαδή πρόκειται για μια εικόνα που αναπαρίσταται ως ένας πίνακας δεδομένων με συγκεκριμένο εύρος τιμών και η κάθε μια τιμή του προσδιορίζει ένα pixel (και οχι τρείς).

Length Στην παρούσα εργασία εννοείται η ευκλείδια απόσταση μεταξύ δύο σημείων εικόνας.

Mean Έχει πολλαπλές σημασίες, στην επεξεργασία εικόνας εννοείται η μέση τιμή φωτεινότητας ή έντασης του σημείου ενδιαφέροντος της εικόνας.

Pixel Εικονοστοιχείο. Προέρχεται από τις λέξεις ”picture element” και είναι το μικρότερο στοιχείο απο το οποίο αποτελείται μια εικόνα. Πολλά χαρτογραφημένα bits (τετραγωνάκια) αποτελούν μια ψηφιακή εικόνα. Κάθε εικονοστοιχείο μιας εικόνας στην οθόνη του υπολογιστή έχει ένα μοναδικό χρώμα. Το χρώμα κάθε pixel αναπαρίσταται με πολλά bits και παράγεται από το συνδυασμό διαφορετικών τόνων των βασικών χρωμάτων: του κόκκινου (Red), του πράσινου (Green) και του μπλε (Blue).Για κάθε χρωματική απόχρωση RGB, ορίζεται μια τιμή για τα τρία

βασικά χρώματα, από 0 έως 255, και έτσι έχουμε $256 * 256 * 256 = 2^24 = 16,7$ εκατομμύρια δυνατούς συνδυασμούς χρωμάτων.

Resolution Ανάλυση εικόνας. Είναι ο αριθμός των εικονοστοιχείων της εικόνας σε κάθε διάστασή της και μετριέται σε κουκκίδες ανά ίντσα, dots per inch (dpi) ή pixels per inch (ppi).

Size Το μέγεθος της εικόνας. Είναι το πλήθος των pixels που περιέχονται σε μια εικόνα στην οριζόντια και κάθετη διεύθυνσή της π.χ 640x480 pixels και υπολογίζεται ως εξής: Μέγεθος εικόνας (σε Byte) = (Οριζόντιος αριθμός εικονοστοιχείων * Κάθετος αριθμός εικονοστοιχείων * Βάθος χρώματος)/8

Slope Η κλίση μιας γραμμικής συνάρτησης. ($slope = \frac{y2-y1}{x2-x1}$)

Standard Deviation Τυπική απόκλιση. Είναι το μέτρο της διασποράς (δηλαδή πόσο διασκορπισμένες είναι οι τιμές μεταξύ τους).

Stitching Η διαδικασία συνένωσης πολλαπλών εικόνων με μερική επικάλυψη για τη παραγωγή μιας υψηλης ανάλυσης εικόνας (πανόραμα).

Texture Η Υφή εικόνας μας δίνει πληροφορίες σχετικά με τη χωρική διάταξη των χρωμάτων ή εντάσεις σε μια εικόνα ή επιλεγόμενη περιοχή μιας εικόνας. Μπορεί να χρησιμοποιηθεί στην επεξεργασία εικόνας. Οι μέθοδοι που συγκρίνουν δύο εικόνες βάσει της υφής, αναζητούν οπτικά πρότυπα καθώς και το που βρίσκονται στο χώρο.

Threshold Τιμή κατωφλίου.

Variance Το τετράγωνο της τυπικής απόκλισης.

Κεφάλαιο 1

Εισαγωγή

1.1 Γενικά

Στον χώρο της ψηφιακής φωτογραφίας σήμερα, παρά την εξέλιξη της τεχνολογίας, προκύπτει ένα σημαντικό ζήτημα. Πώς μπορεί κάποιος να «αιχμαλωτίσει» το εύρος της εικόνας που βλέπει, παραδείγματος χάρη ένα τοπίο, χωρίς να χρειαστεί να βγάλει μια μακρινή φωτογραφία που χάνει σε λεπτομέρειες. Στο ερώτημα αυτό, η απάντηση είναι η δημιουργία πανοραμικών εικόνων. Η διαδικασία έχει ως εξής:

Ο χρήστης «τραβάει» διαδοχικές φωτογραφίες που έχουν κάποια συνέχεια μεταξύ τους και με ένα κατάλληλο εργαλείο/λογισμικό οι εικόνες ενώνονται και προκύπτει ένα πανοραμικό αποτέλεσμα, μια φωτογραφία του χώρου όπως τον βλέπουμε. Τα τελευταία χρόνια έχουν αναπτυχθεί πολλοί μηχανισμοί με αποστολή τη συνένωση πολλών διαδοχικών φωτογραφιών και τελικά τη δημιουργία ενός πανοράματος. Ο αγγλικός όρος που αποδίδει τη λειτουργία αυτή είναι stitching (*διάγραμμα 1.1*) και πλέον με ειδικές εφαρμογές μπορεί να γίνει εύκολα και γρήγορα.

1.2 Στόχοι της εργασίας

Η εργασία έχει σαν κεντρικό θέμα τη μελέτη και την υλοποίηση ένος εργαλείου δημιουργίας πανοραμικών εικόνων. Συγκεκριμένα οι στόχοι της παρούσας εργασίας είναι να γίνει κατανοητός ο μηχανισμός που λειτουργεί πίσω τον αλγόριθμο της δημιουργίας πανοράματος και η υλοποίηση ενός τέτοιου. Έχει επιλεχθεί να χρησιμοποιηθεί ένας συγκεκριμένος αλγόριθμος για το σκοπό αυτό, όμως η φιλοσοφία του stitching είναι

Διάγραμμα 1.1: Εικόνα πανόραμα

παρόμοια για όλους τους αλγορίθμους που έχουν αναπτυχθεί κατα καιρούς. Τελικά ο αναγνώστης θα έχει την ευκαιρία να μελετήσει τον τρόπο που λειτουργούν τέτοιες εφαρμογές βήμα-βήμα και επιπλέον να γνωρίσει μια διαφοροποιημένη μέθοδο, την προτεινόμενη, που σκοπό έχει να ελαχιστοποιήσει κάποια απο τα μειονεκτήματα των μέχρι σήμερα υλοποιημένων αλγορίθμων παρόμοιας φύσης.

1.3 Παρόμοιες εφαρμογές

Στον πίνακα 1.1 αναφέρονται μερικές εφαρμογές δημιουργίας πανοράματος που κυκλοφορούν (δωρεάν και μη) πηγή: wikipedia.

Πίνακας 1.1: Λογισμικό Πανόραμα

Όνομα	Λειτουργικό σύστημα	Κόστος	WWW
Panorama Factory v5	Windows,MacOS	$40	panoramafactory
Autostitch	Windows,MacOS, Linux,Android	Time limited demo only	Autostitch
hugin	Windows,MacOS, Linux,FreeBSD	Free	hugin.sf.net
PanoramaMaker	Windows,MacOS	$39	STOIK.com
PhotoStitcher	Windows,MacOS	$19.99	photostitcher.com
SharpStitch	Windows	$995.00	imagingshop.com

1.4 Δομή της εργασίας

Συμπεριλαμβανομένου και του τρέχοντος κεφαλαίου το οποίο αποτελεί την εισαγωγή της πτυχιακής εργασίας, ακολουθούν τα εξής κεφάλαια:

Κεφάλαιο 2 Σύντομη ανάλυση των αλγόριθμων που χρησιμοποιούνται για τη δημιουργία πανοραμικών εικόνων σε μαθηματικό επίπεδο.

Κεφάλαιο 3 Περιεκτική περιγραφή της τεχνολογίας και του περιβάλλοντος που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής.

Κεφάλαιο 4 Η μεθοδολογία του αλγορίθμου που επιλέχθηκε για υλοποίηση.

Κεφάλαιο 5 Οδηγός και ανάλυση του γραφικού περιβάλλοντος της εφαρμογής.

Κεφάλαιο 6 Περιγραφή σχεδίασης/υλοποίησης του συστήματος.

Κεφάλαιο 7 Πειραματικά αποτελέσματα.

Κεφάλαιο 8 Συμπεράσματα και προτάσεις για επεκτάσεις της εφαρμογής.

Κεφάλαιο 2

Μια μαθηματική προσέγγιση των αλγορίθμων του stitching

2.1 Η φιλοσοφία του stitching

Έστω δύο εικόνες με συνέχεια μεταξύ τους. Είναι προφανές οτι για να ενωθούν πρέπει να βρεθεί η συσχέτιση μεταξύ τους. Χρειάζεται κάποιο κριτήριο ομοιότητας, που θα βοηθήσει τον αλγόριθμο να ξεχωρίσει που πρέπει να «συρράψει» τις εικόνες. Οι τεχνικές που χρησιμοποιούνται για το σκοπό αυτό (συχνά αναφέρεται ως image registration ή image alignment) βασίζονται στη φωτεινότητα (intensity based), άλλες στο συχνοτικό πεδίο (frequency domain) και άλλες σε χαρακτηριστικά (features based).

Οι αλγόριθμοι που βασίζονται στη φωτεινότητα συχνά περιλαμβάνουν ένα μεγάλο μέρος υπολογισμών και ως εκ τούτου δεν είναι κατάλληλοι για ευθυγράμμιση εικόνων όταν έχουμε να κάνουμε με εικόνες που έχουν υποστεί κλιμάκωση (scaling) *διάγραμμα 2.2* και περιστροφή (rotation) *διάγραμμα 2.3* .

Διάγραμμα 2.1: Αρχική Εικόνα

Διάγραμμα 2.2: Εικόνα με κλιμάκωση

Διάγραμμα 2.3: Εικόνα με περιστροφή

Απο την άλλη πλευρά, οι αλγόριθμοι που βασίζονται στη συχνότητα πεδίου είναι γενικά πιο γρήγοροι και μπορούν να χειριστούν μικρή μετατόπιση, περιστροφή (rotation) και κλιμάκωση (scaling) της εικόνας. Όταν όμως πρόκειται για εικόνες που έχουν μικρές περιοχές μερικής επικάλυψης *διάγραμμα 2.4*, οι αλγόριθμοι αυτοί δεν προτιμώνται.

Διάγραμμα 2.4: Εικόνες με μικρή περιοχή επικάλυψης

Οι αλγόριθμοι με βάση τα χαρακτηριστικά της εικόνας χρησιμοποιούν ένα μικρό αριθμό απο αμετάβλητα σημεία, γραμμές ή ακμές για να πετύχουν την ευθυγράμμιση εικόνας. Ένα σημαντικό πλεονέκτημα αυτών των αλγορίθμων είναι ότι η υπολογιστική πολυπλοκότητα μειώνεται λόγω της μικρότερης πληροφορίας που χρειάζεται να υποβληθεί σε επεξεργασία. Επιπλέον οι αλγόριθμοι που βασίζονται στα χαρακτηριστικά είναι ανθεκτικοί σε αλλαγές στην ένταση της εικόνας. *διάγραμμα 2.5*.

Παρόλα αυτά, εντοπίζεται ένα σοβαρό ζήτημα στους ήδη υπάρχοντες αλγόριθμους. Η πλειοψηφία αυτών χρησιμοποιεί μια εξαντλητική μέθοδο αναζήτησης που βασίζεται στο ταίριασμα ενώς προτύπου (template matching). Ως αποτέλεσμα, οι υπολογισμοί, αν και έχουν ήδη μειωθεί σε κάποιο βαθμό, συνεχίζουν να είναι χρονοβόροι και δεν πληρούν τις απαιτήσεις σε πραγματικό χρόνο, στον οποίο βασίζονται συνήθως οι

Διάγραμμα 2.5: Εικόνες με διαφορετική ένταση

εφαρμογές για το πανόραμα. Στην παρούσα πτυχιακή θα παρουσιαστεί ένας αλγόριθμος που χειρίζεται τα παραπάνω μειονεκτήματα αποτελεσματικά [**1**] .

2.2 Συνοπτικά βήματα του stitching

1. Εντοπισμός σημείων ενδιαφέροντος ανα δύο διαδοχικές εικόνες με μερική επικάλυψη (detect keypoints)
2. Αντιστοίχιση υποψήφιων σημείων μεταξύ των διαδοχικών εικόνων με κάποιο κριτήριο ομοιότητας (match keypoints)
3. Γεωμετρικός μετασχηματισμός των σημείων διαδοχικών εικόνων (estimate homography)
4. Ένωση εικόνων/δημιουργία πανοράματος (stitching)

2.3 Ο προτεινόμενος αλγόριθμος

Όσον αφορά την δημιουργία πανοραμικών εικόνων, έχει παρατηρηθεί οτι οι γειτονικές εικόνες που χρησιμοποιούνται για την ευθυγράμμιση, έχουν συνήθως μικρή μερική επικάλυψη, περιστροφή, κλιμάκωση και μετατόπιση. Ο προτεινόμενος αλγόριθμος βασίζεται σε μια καινοτομία που εφαρμόζει πολλαπλούς περιορισμούς μέχρι να καταλήξει στο τελικό ταίριασμα των εικόνων. Αρχικά φιλτράται ένας μεγάλος αριθμός απο υποψήφιες γωνίες με βάση τη θέση, έπειτα παράγεται ένα αρχικό σύνολο απο

ζευγάρια γωνιών που βασίζεται στις grayscale τιμές κάθε γωνίας μεταξύ των γειτονικών εικόνων. Τέλος, εφαρμόζονται πολλαπλοί περιορισμοί, (π.χ σε σχέση με την μέση τιμή, την απόσταση και την κλίση) σε όλα τα υποψήφια ζευγάρια για να αφαιρεθούν τα ασυσχέτιστα μεταξύ τους. Ως εκ τούτου μειώνεται σημαντικά ο αριθμός επαναλήψεων που χρειάζεται ο τελικός αλγόριθμος RANSAC (random sample consensus). Το αποτέλεσμα είναι η δημιουργία πανοράματος να πραγματοποιείται πιο αποδοτικά.

2.4 Επιλογή γωνιών

Με τον όρο «γωνία» εννοούνται τα σημεία των εικόνων για τα οποία μια μετακίνηση προς οποιαδήποτε διεύθυνση στο επίπεδο της εικόνας, οδηγεί σε μεγάλες μεταβολές της συνάρτησης φωτεινότητας. Η συνάρτηση φωτεινότητας της εικόνας περιέχει δύο μεταβλήτές, μια για την οριζόντια και μια για τη κάθετη διεύθυνση. Έτσι, ο όρος «γωνία» αφορά σε σημεία της εικόνας στα οποία η καμπυλότητα της συνάρτησης φωτεινότητας αποκτά ένα τοπικό μέγιστο. Τα σημεία αυτά είναι δυνατόν να αποτελούν προβολές πραγματικών γωνιών αντικειμένων, όμως γενικά αναφέρονται σε σημεία περιοχών με έντονη υφή (texture). Για την ανίχνευση γωνιών σε εικόνες έχουν αναπτυχθεί πολλοί αλγόριθμοι. Στην παρούσα πτυχιακή θα χρησιμοποιηθεί ο αλγόριθμος των Harris & Stephens.[**2**][**3**]

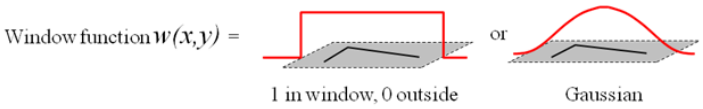
2.4.1 Περιγραφή του αλγορίθμου εύρεσης γωνιών του Harris

Ο αλγόριθμος Harris αποτελεί μια βελτιωμένη εκδοχή ενός παλαιότερου αλγορίθμου του Moravec. Η βασική λογική είναι ότι για κάθε σημείο ελέγχεται η συνάρτηση φωτεινότητας ή αλλιώς έντασης (intensity) για μικρές μετακινήσεις γύρω από αυτό και ως γωνίες χαρακτηρίζονται τα σημεία εκείνα στα οποία οι μεταβολές της τιμής της συνάρτησης φωτεινότητας είναι μεγάλες. Η μεταβολή αυτή εκφράζεται μέσω της συνάρτησης E που δίνεται απο τη σχέση:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

Ο όρος $w(x,y)$ εκφράζει το παράθυρο εντός του οποίου ελέγχεται η μεταβολή αυτή, ενώ u και v είναι η οριζόντια και κάθετη μετακίνηση αντίστοιχα γύρω απο το σημείο

x, y. Οι μετακινήσεις (u, v) λαμβάνουν τιμές $(1,0),(1,1),(0,1),),(-1,1)$. Η απόκριση θα περιέχει έντονα την παρουσία θορύβου εξαιτίας της δυαδικής, ορθογώνιας μορφής του παραθύρου $w(x, y)$ έτσι για λόγους εξομάλυνσης χρησιμοποιείται και ένα Gaussian κυκλικό παράθυρο.[**4**]



Διάγραμμα 2.6: Gaussian φίλτρο

Για μικρές μεταβολές (shifts) $[u, v]$ χρησιμοποιείται η παρακάτω προσέγγιση:

$$E(u, v) \equiv [u, v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

Όπου M είναι ένας 2×2 πίνακας που υπολογίζεται απο τα παράγωγα της εικόνας (image derivatives):

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

I_x^2 είναι το γινόμενο Kronecker του τετραγώνου της πρώτης παραγώγου της συνάρτησης I ως προς x με το παράθυρο $w(x, y)$, I_y^2 είναι το γινόμενο Kronecker του τετραγώνου της πρώτης παραγώγου της συνάρτησης I ως προς y με το παράθυρο $w(x, y)$ και $I_x I_y$ είναι το γινόμενο Kronecker του γινομένου της πρώτης παραγώγου της συνάρτησης I ως προς x με την πρώτη παράγωγο της συνάρτησης I ως προς y, με το παράθυρο $w(x, y)$ [**5**].

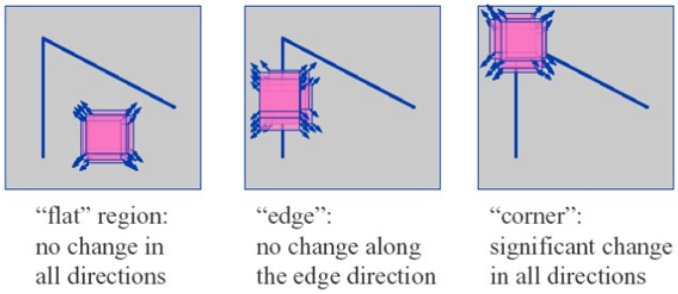
Αποδεικνύεται ότι οι ιδιοτιμές (eigenvalues) του παραπάνω πίνακα είναι ανάλογες με τις κύριες καμπυλότητες της επιφάνειας της εικόνας I. Η τελική εκτίμηση γίνεται λοιπόν μέσω των ιδιοτιμών του πίνακα . Αν $\lambda 1$, $\lambda 2$ είναι οι ιδιοτιμές του πίνακα , διακρίνονται 3 περιπτώσεις:

1. Αν $\lambda 1$ και $\lambda 2$ είναι μικρές, τότε η συνάρτηση φωτεινότητας της εικόνας έχει σχεδόν σταθερή ένταση (οι μετακινήσεις γύρω από το σημείο που ελέγχεται έχουν σχεδόν μηδαμινή επίδραση στην συνάρτηση I).

2. Αν μία από τις ιδιοτιμές $\lambda 1$ ή $\lambda 2$ είναι μεγάλη και η άλλη μικρή, τότε η τοπική αυτοσυσχέτιση παρουσιάζει κορυφή με αποτέλεσμα μια μικρή μετακίνηση κατά

μήκος της κορυφής να προκαλεί μικρή αλλαγή στην συνάρτηση I στη μία διεύθυνση και σημαντική στην άλλη. Αυτό υποδεικνύει την ύπαρξη ακμής.

3. Αν και οι δύο ιδιοτιμές είναι μεγάλες, τότε ακόμα και μικρή μετακίνηση σε οποιαδήποτε διεύθυνση έχει σαν αποτέλεσμα μεγάλη αλλαγή στην συνάρτηση . Αυτό υποδεικνύει την ύπαρξη γωνίας.



Διάγραμμα 2.7: Πιθανές θέσεις σημείων και η σημασία τους

Η σχέση που εκφράζει την ανταπόκριση (corner response) είναι:

$$R = detM - k(traceM)_2$$

Όπου,

$$detM = \lambda_1\lambda_2$$

$$traceM = \lambda_1\lambda_2$$

και k μια σταθερά που κυμαίνεται μεταξύ $0.04 - 0.06$.

- R εξαρτάται μόνο απο τις ιδιοτιμές του M.
- R είναι μεγάλο όταν πρόκειται για γωνία.
- R είναι αρνητικό με μεγάλο μέγεθος όταν πρόκειται για ακμή.
- $|R|$ είναι μικρό όταν πρόκειται για "flat region" .

Διάγραμμα 2.8: Σχετικά με R values

2.5 Αντιστοίχιση γωνιών

Η αντιστοίχιση δύο εικόνων, ή γενικότερα σημείων είναι ένα θέμα που προκύπτει στην επεξεργασία εικόνας. Ουσιαστικά πρέπει να βρεθεί κάποιο μοτίβο που επαναλαμβάνεται και στις δύο εικόνες, μια συσχέτιση μεταξύ των σημείων για να προσδιοριστεί ένα πρότυπο ομοιότητας το οποίο θα οδηγήσει σε κάποια σύγκριση των σημείων και θα καθορίσει τελικά το «κατα πόσο μοιάζουν» μεταξύ τους. Μια βασική προσέγγιση που χρησιμοποιείται συχνά σε τέτοιες περιπτώσεις είναι η κανονικοποιημένη συσχέτιση. Στην επεξεργασία σήματος με τον όρο συσχέτιση (cross correlation) εννοείται το μέτρο της ομοιότητας μεταξύ δύο κυματομορφών ως μια συνάρτηση time-lag που εφαρμόζεται σε ένα απο τα δύο. Είναι παρόμοιας φύσης με την συνέλιξη δυο συναρτήσεων [**6**]. Για την επεξεργασία εικόνας όπου η ανάλυση, η φωτεινότητα και άλλα χαρακτηριστικά μεταξύ δύο εικόνων μπορεί να διαφέρουν, πρέπει πρώτα οι εικόνες να κανονικοποιηθούν εξού και ο όρος normalized cross correlation. Στην παρούσα εργασία λοιπόν, θα χρησιμοποιηθεί ο αλγόριθμος της κανονικοποιημενης συσχέτισης NCC (normalized cross correlation) ως πρότυπο ομοιότητας μεταξύ των εικόνων, ώστε να γίνει η αντιστοίχιση των γωνιών που βρέθηκαν.

2.5.1 Περιγραφή αλγορίθμου κανονικοποιημένης συσχέτισης

Η κανονικοποιημένη συσχέτιση υπολογίζεται μεταξύ ενός παραθύρου (template) ή αλλιώς μιας ορισμένης περιοχής εικόνας και μιας αρχικής εικόνας. Συγκεκριμένα, αν θεωρηθεί οτι η μέση τιμή φωτεινότητας (mean) ενός παραθύρου δίνεται απο τη σχέση:

$$\bar{I} = \frac{1}{N} \sum_{x \in w(x)} I(x)$$

όπου $N = |w(x)|$ ο αριθμός των σημείων του παραθύρου, ενώ η διασπορά απο την σχέση:

$$\sigma_I = \sqrt{\sum_{x \in w(x)} (I(x) - \bar{I})^2}$$

τότε η κανονικοποιημένη συσχέτιση για δύο εικόνες $I1, I2$ δίνεται απο την σχέση:

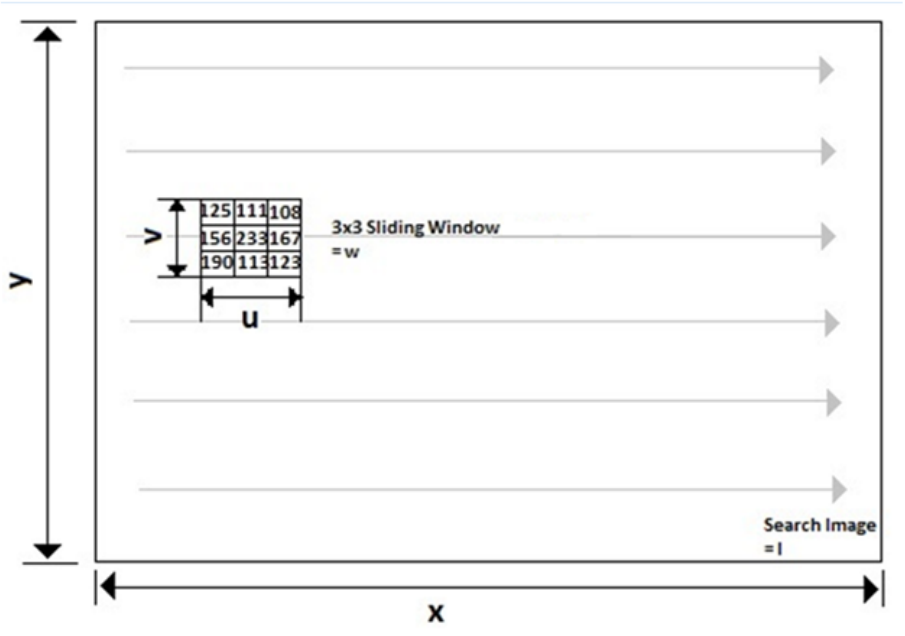
$$NCC = \sum_{x \in w(x)} \frac{(I_1(x) - \bar{I_1})(I_2(x) - \bar{I_2})}{\sigma_1 \sigma_2}$$

Για κάθε σημείο της πρώτης εικόνας, οι τιμές των φωτεινοτήτων των σημείων του παραθύρου κανονικοποιούνται ως προς τη μέση τιμή και τη διασπορά και υπολογίζονται

οι διαφορές τους από τις κανονικοποιημένες τιμές φωτεινότητας των σημείων παραθύρου ίδιου μεγέθους γύρω απο κάθε σημείο της δεύτερης εικόνας [**7**]. Πρακτικά, για κάθε σημείο της εικόνας I_1, I_2 αφαιρώντας την μέση τιμή φωτεινότητας (mean) ενός παραθύρου και διαιρώντας με την τυπική απόκλιση (standard deviation) προκύπτει η κανονικοποιημένη συσχέτιση NCC.

Οι τιμές της κανονικοποιημένης συσχέτισης κυμαίνονται απο -1 έως 1 όπου 1 θεωρείται οτι υπάρχει τέλεια συσχέτιση στις τιμές των σημείων των δύο παραθύρων, ενώ -1 θεωρείται οτι τα δύο παράθυρα είναι πλήρως ασυσχέτιστα. Η διαδικασία υπολογισμού της κανονικοποιημένης συσχέτισης έχει πολυπλοκότητα $O(N1 * N2)$ όπου $N1$ είναι ο αριθμός των σημείων της πρώτης εικόνας και $N2$ ο αριθμός των σημείων της δεύτερης εικόνας καθώς η έξοδος είναι ένας πίνακας τιμών κανονικοποιημένης συσχέτισης διαστάσεων $N1 \times N2$. Όπως εύκολα προκύπτει, αυτή η διαδικασία μπορεί να είναι πολύ χρονοβόρα ειδικά όταν ο αριθμός των σημείων είναι πολύ μεγάλος. Υπάρχει όμως τρόπος να μειωθεί ο χρόνος εκτέλεσης της διεργασίας αν οριστεί μια μέγιστη επιτρεπτή απόσταση αναζήτησης αντιστοιχιών. Σε αυτή τη περίπτωση δεν θα χρειάζεται να ερευνηθούν όλα τα σημεία της άλλης εικόνας για αντιστοίχιση αλλά όσα ανήκουν σε μια συγκεκριμένη περιοχή αναζήτησης.

Ο τρόπος που λειτουργέι η αναζήτηση συσχέτισης με τον αλγόριθμο NCC γίνεται πιο κατανοητός στο παρακάτω διάγραμμα [**8**]: Το w συμβολίζει το sliding window



Διάγραμμα 2.9: Αναζήτηση συσχέτισης μεταξύ εικόνων με τη χρήση παραθύρου

(το μέγεθος του παραθύρου πρέπει να είναι περιττός αριθμός ώστε να είναι κεντραρισμένο στο σημείο ενδιαφέροντος). Έστω ότι I είναι ολόκληρη η εικόνα, τότε για κάθε γωνία που έχει βρεθεί (στο γράφημα μια γωνία έχει τιμή 233) δημιουργείται ένα παράθυρο αναζήτησης γύρω απο την επιλεγμένη γωνία και οι τιμές του συγκρίνονται με το παράθυρο (template) της δεύτερης εικόνας. Αφού το template της δεύτερης εικόνας έχει συγκριθεί με όλες τις γωνίες της πρώτης, ως template επιλέγεται ένα παράθυρο γύρω απο τη δεύτερη γωνία της δεύτερης εικόνας και η διαδικασία επαναλαμβάνεται μέχρι όλες οι γωνίες της πρώτης εικόνας να συγκριθούν με όλες της γωνίες της δεύτερης. Η μέση τιμή φωτεινότητας (mean) του template παραμένει ίδια, ενώ η μέση τιμή φωτεινότητας της εικόνας αλλάζει καθώς μετακινείται το sliding window μέσα στην εικόνα.

Στην παρούσα εργασία η κανονικοποιημένη συσχέτιση υπολογίζεται σε ένα τετραγωνικό παράθυρο $N \times N$ pixels (template) γύρω απο τα υπολογισμένα μέσω του αλγορίθμου Harris σημεία (corners) των εικόνων. Όσο μεγαλύτερο είναι το μέγεθος του παραθύρου τόσο πιο αποτελεσματική είναι η αντιστοίχιση των σημείων των εικόνων αλλά και τόσο πιο μεγάλος είναι ο χρόνος εκτέλεσης του προγράμματος.

2.6 Ο αλγόριθμος RANSAC

Ο αλγόριθμος RANSAC είναι συντομογραφία του «Random Sample Consensus» και δημοσιεύτηκε αρχικά από τους Fischler και Bolles το 1981 [**7**][**9**]. Πρόκειται για μια επαναληπτική μέθοδο για τον υπολογισμό παραμέτρων ενός μαθηματικού μοντέλου από ένα σύνολο δεδομένων. Ο RANSAC είναι ένας μη-ντετερμινιστικός αλγόριθμος με την έννοια ότι παράγει ένα λογικό αποτέλεσμα μόνο με μια συγκεκριμένη πιθανότητα, η οποία αυξάνει με τον αριθμό των επαναλήψεων. Παρόλα αυτά, είναι πολύ δημοφιλής γιατί είναι ένας απλός αλγόριθμος που δουλεύει πολύ καλά στη πράξη χωρίς να χρειάζεται να γίνουν υποθέσεις για τα δεδομένα εισόδου του.

Η βασική ιδέα του αλγορίθμου είναι ότι τα δεδομένα εισόδου του αποτελούνται από «inliers», δηλαδή από δεδομένα που η κατανομή τους μπορεί να εξηγηθεί βάσει κάποιων παραμέτρων ενός μοντέλου και από «outliers», δηλαδή από δεδομένα που δεν ταιριάζουν σε ένα συγκεκριμένο μοντέλο. Επιπλέον, θεωρείται ότι τα δεδομένα μπορεί να έχουν πειραχθεί από θόρυβο. Τα outliers μπορεί να έχουν προέλθει από υψηλές

τιμές θορύβου ή από λανθασμένες μετρήσεις ή ακόμα και από λανθασμένες υποθέσεις σχετικά με την ερμηνεία των δεδομένων. Ο RANSAC επίσης υποθέτει ότι δοσμένου ενός μικρού αριθμού inliers, υπάρχει διαδικασία υπολογισμού των παραμέτρων ενός μοντέλου στο οποίο αυτά ταιριάζουν. Δηλαδή, ακόμα και αν ο θόρυβος έχει επηρεάσει μεγάλο ποσοστό των δεδομένων εισόδου, ο αλγόριθμος RANSAC είναι ικανός να βρει λύση.

2.6.1 Περιγραφή του αλγορίθμου Ransac

Οι κύριες παράμετροι του αλγορίθμου RANSAC είναι: η πιθανότητα p σε μια επανάληψη να επιλέξει ο αλγόριθμος δεδομένα που είναι όλα inliers, ο αριθμός των επαναλήψεων του αλγορίθμου και ο αριθμός των επαναλήψεων του αλγορίθμου για την εύρεση ενός μη-εκφυλισμένου σετ δεδομένων. Εκφυλισμένο σετ δεδομένων ορίζεται ένα σύνολο δεδομένων που δεν μπορεί να ταιριάξει σε κανένα μοντέλο. Κάτι τέτοιο μπορεί να οφείλεται στο γεγονός ότι κάποια δεδομένα μπορεί να μην τηρούν τους περιορισμούς που απαιτούνται για τον υπολογισμό του μοντέλου. Παραδείγματος χάριν, για τον υπολογισμό του πίνακα F (fundamental matrix: θεμελιώδης πίνακας 3×3 [**10**]) θα πρέπει τα δεδομένα να είναι ανεξάρτητα, δηλαδή να μην υπάρχουν κοινά σημεία στις αντιστοιχίες. Σε αντίθετη περίπτωση, είναι αδύνατος ο υπολογισμός του πίνακα F. Αν ο αλγόριθμος βρει ένα τέτοιο σετ δεδομένων τότε επιλέγει άλλο, τόσες φορές όσες χρειάζεται για να βρει ένα μη-εκφυλισμένο σετ που μπορεί να ταιριάξει σε ένα μοντέλο. Η τελευταία λοιπόν παράμετρος καθορίζει τον αριθμό προσπαθειών του αλγορίθμου και τερματίζει την διαδικασία αν ο αλγόριθμος ύστερα από τόσες προσπάθειες δεν καταλήξει σε μη-εκφυλισμένο σετ δεδομένων. Έτσι αποφεύγεται η περίπτωση να κολλήσει ο αλγόριθμος σε ατέρμων βρόγχο.

Σε αυτό το σημείο πρέπει να γίνει μια παρατήρηση. Ο αλγόριθμος RANSAC δουλεύει καλύτερα σε θεωρητικό επίπεδο από ότι σε πρακτικό. Αυτό οφείλεται στη πιθανότητα p εύρεσης ενός σετ δεδομένων που αποτελείται μόνο από inliers. Κάτι τέτοιο συμβαίνει γιατί δεν μπορεί να οριστεί με βεβαιότητα μια τέτοια πιθανότητα καθώς δεν είναι γνωστός ο αριθμός των inliers στο αρχικό σετ δεδομένων εισόδου του αλγορίθμου RANSAC. Η πιθανότητα p εκφράζει κυρίως την επιθυμία να βρεθεί σετ δεδομένων χωρίς outliers σε μια επανάληψη και επομένως πρόκειται για την πιθανότητα η λύση του αλγορίθμου να είναι η βέλτιστη. Η πιθανότητα p και ο αριθμός των επαναλήψεων

είναι άρρηκτα συνδεδεμένα μεταξύ τους σύμφωνα με τα άρθρα των D. Capel [**11**] και O. Chuma, J. Mattas [**12**]. Η πιθανότητα ο αλγόριθμος RANSAC να αποτύχει να βρει ένα σετ δεδομένων μεγέθους H που αποτελείται μόνο από inliers ύστερα από k επαναλήψεις δίνεται από την σχέση:

$$\eta = 1 - p = (1 - P_H)^k$$

Η μεταβλητή P_H ορίζεται ως η πιθανότητα να επιλεγεί ένα δείγμα m inliers απο ένα σετ N αρχικών δεδομένων εισόδου και υπολογίζεται: $P_H = \prod_{j=0}^{m-1} \frac{H-j}{N-j} \approx \epsilon^m$ όπου $\epsilon = \frac{1}{N}$ είναι το ποσοστό των inliers Έτσι προκύπτει ότι ο απαραίτητος αριθμός επαναλήψεων για να διασφαλιστεί η πιθανότητα η είναι: $k = \frac{log(\eta)}{log(1-P_H)}$ Η ικανοποίηση της συνθήκης της παραπάνω εξίσωσης οδηγεί στον πρόωρο τερματισμό του αλγορίθμου. Έτσι αυξάνεται η ταχύτητα εκτέλεσής του αφού θεωρείται ότι έχει βρεθεί η βέλτιστη λύση.

Συνοπτικά, η γενική διαδικασία για την περίπτωση αντιστοίχισης σημείων με τον αλγόριθμο RANSAC είναι:

1. Επιλέγει σε κάθε επανάληψη ένα τυχαίο σετ 8 ζευγών σημείων απο το αρχικό σύνολο ζευγών τα οποία θεωρεί ως υποθετικά inliers. Η επιλογή 8 σημείων δεν είναι τυχαία αλλα είναι η ελάχιστη δυνατή για τον υπολογισμό του πίνακα F.

2. Προσπαθεί να υπολογίσει τον πίνακα F διαστάσεων 3×3 απο τα υποθετικά inliers μέσω του αλγορίθμου 8-σημείων [**22**].

3. Όλα τα υπόλοιπα ζεύγη αντιστοιχιών που δεν έχουν επιλεγεί ως υποθετικά inliers για τον σχηματισμό του πίνακα F εξετάζονται αν ταιριάζουν μέσω της απόστασης Sampson's που εκφράζει την απόσταση των πραγματικών σημείων απο τις προβολές τους μέσω του πίνακα F. Αν για κάποιο ζεύγος σημείων η απόσταση αυτή είναι πολύ μικρή και κάτω απο ένα όριο, τότε θεωρείται ότι και αυτό το ζεύγος ανήκει στα υποθετικά inliers.

4. Ο πίνακας μεταφοράς και περιστροφής F θεωρείται αρκετά καλός αν σχετικά μεγάλος αριθμός αντιστοιχιών σημείων αποτελούν υποθετικά inliers.

5. Τελικά ο πίνακας F αξιολογείται αθροίζοντας το σφάλμα όλων των ζευγών σημείων, δηλαδή υπολογίζοντας την απόσταση των πραγματικών σημείων απο τις

προβολές τους, που προκύπτουν απο την λύση της γεωμετρίας επιπόλων (ισχύει $x_2^T \cdot F \cdot x_1 = 0$) με τον υπολογισμένο πίνακα F.

6. Ο πίνακας F με την μικρότερη τιμή σφάλματος επιλέγεται ως βέλτιστος πίνακας και στη συνέχεια επαναπροσδιορίζεται απο όλα τα υποθετικά inliers αφού είχε υπολογιστεί μόνο για το αρχικό σε 8 σημείων.

Κεφάλαιο 3

Τεχνολογίες που χρησιμοποιήθηκαν

3.1 Εισαγωγή

Το κεφάλαιο αυτό έχει ως αντικείμενο τις τεχνολογίες που εφαρμόστηκαν για την υλοποίηση της παρούσας εργασίας. Ποιό συγκεκριμένα, περιγράφεται συνοπτικά το λογισμικό Matlab με το οποίο αναπτύχθηκε η εφαρμογή, η εργαλειοθήκη Image Processing Toolbox που χρησιμοποιήθηκε για την επεξεργασία εικόνας καθώς και η πλατφόρμα GUIDE στην οποία βασίζεται το γραφικό περιβάλλον της εφαρμογής.

3.2 Λογισμικό Matlab

Το όνομα MATLAB προέρχεται απο τις λέξεις MAtrix και LABoratory. Πρόκειται για μια υψηλού επιπέδου γλώσσα προγραμματισμού που ειδικεύεται κυρίως σε μαθηματικούς υπολογισμούς,ανάλυση και προγραμματισμό. Παρέχει στον χρήστη ένα διαδραστικό περιβάλλον με χιλιάδες ενσωματωμένες συναρτήσεις κατάλληλες για την υλοποίηση απαιτητικών υπολογιστικών αναλύσεων, γραφημάτων καθώς επίσης και την παραγωγή διάφορων animations [**13**].

3.2.1 Το περιβάλλον του Matlab

Το Matlab υποστηρίζει σχεδόν όλα τα διαθέσιμα λειτουργικά συστήματα (windows,unix,mac OS,sun solaris,linux) και λειτουργέι μέσω τριών βασικών παραθύρων *διάγραμμα 3.1* [**14**]:

1. *Επιφάνεια εργασίας του Matlab (Matlab Desktop):* είναι το εξ ορισμού παράθυρο το οποίο συναντά ο χρήστης με την εκκίνηση του προγράμματος και αποτελείται απο τα εξής επιμέρους υπο-παράθυρα:

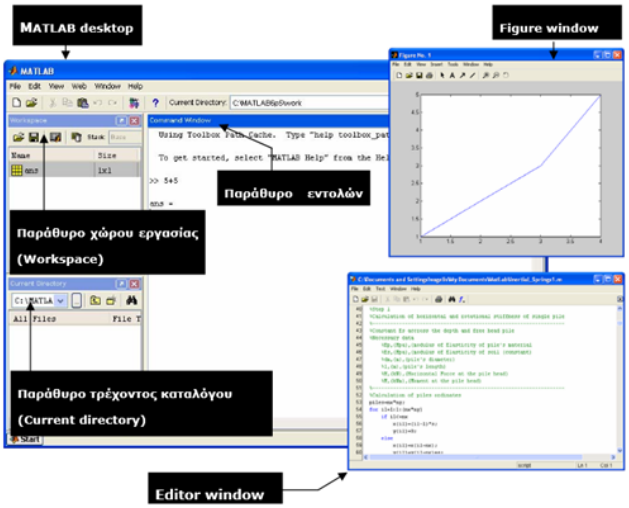
    - *Παράθυρο εντολών (Command Window):* είναι το βασικό παράθυρο στο οποίο πληκτρολογεί ο χρήστης το σύνολο των εντολών για την εξαγωγή αποτελεσμάτων.
    - *Παράθυρο τρέχοντος καταλόγου (Current Directory):* είναι το σημείο της επιφάνειας εργασίας του Matlab όπου αναγράφονται το σύνολο των αρχείων που είναι αποθηκευμένα στον τρέχον κατάλογο του συστήματος και παρέχει τη δυνατότητα πλοήγησης και διαφόρων επιλογών.
    - *Παράθυρο χώρου εργασίας (Workspace):* σε αυτό το παράθυρο απεικονίζονται όλες οι μεταβλητές που εισάγονται και χρησιμοποιούνται στο παράθυρο εντολών και παρέχει πληροφορίες για την εκάστοτε μεταβλήτη (τύπος,μέγεθος,περιεχόμενο).
    - *Παράθυρο ιστορικού εντολών (Command History):* το σύνολο εντολών που πληκτρολογούνται στο παράθυρο εντολών καταγράφεται στο παράθυρο αυτό και παρέχει στον χρήστη τη δυνατότητα ανάκλησης.

2. *Παράθυρο σύνταξης (Editor Window):* είναι το παράθυρο στο οποίο ο χρήστης μπορεί να αναπτύξει,επεξεργαστεί και αποθηκεύσει τα δικά του αρχεία εντολών,τα οποία απαρτίζονται κυρίως απο τα M-files.

3. *Παράθυρο γραφημάτων (Figure Window):* το αποτέλεσμα απο όλες τις σχετικές με γραφήματα εντολές, οι οποίες έχουν εκτελεστεί στο παράθυρο εντολων στο ξεχωριστό αυτό παράθυρο το οποίο εκτός απο την εμφάνιση γραφημάτων, δίνει στο χρήστη και τη δυνατότητα επεξεργασίας και χειρισμου αυτών (τελευταίες εκδόσεις).

3.3 Image Processing Toolbox

Πρόκειται για μια βιβλιοθήκη-εργαλειοθήκη του Matlab που προσφέρει μια ολοκληρωμένη σειρά απο τυπικούς αλγορίθμους, συναρτήσεις και εφαρμογές για την επε-



Διάγραμμα 3.1: Επιφάνεια εργασίας Matlab

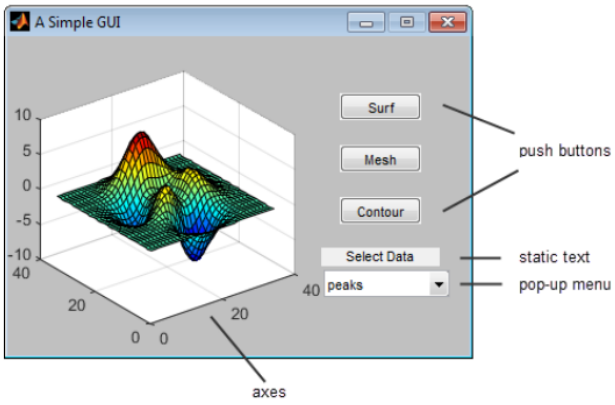
ξεργασία, ανάλυση, απεικόνιση εικόνας και ανάπτυξη αλγορίθμων. Ο χρήστης μπορεί να εκτελέσει λειτουργίες που αφορούν την ανάλυση, κατάτμηση, βελτίωση εικόνας, μείωση θορύβου, γεωμετρικούς μετασχηματισμούς κ.α.

Υποστηρίζει ένα ευρύ σύνολο τύπων εικόνας, συμπεριλαμβανόμενων και high dynamic range, gigapixel resolution, embedded ICC profile, and tomographic. Οι συναρτήσεις απεικόνισης και οι εφαρμογές παρέχουν τη δυνατότητα στο χρήστη για πειραματισμούς σε διάφορες περιοχές των pixel της εικόνας, ρυθμίσεις στο χρώμα, την αντίθεση, δημιουργία περιγραμμάτων ή ιστογραμματων και διαχείρισμός περιοχών ενδιαφέροντος (ROI) μιας εικόνας. Η εργαλειοθήκη υποστηρίζει την επεξεργασία, προβολή και την πλοήγηση μεγάλων εικόνων [**15**].

3.4 GUIDE

Σε πολλές περιπτώσεις (παραδείγματος χάρη για χρήστες που δεν είναι εξικοιωμένοι με τη γραμμή εντολών του MATLAB) προκύπτει η ανάγκη για δημιουργία ενός διαδραστικού εργαλείου απεικόνισης που εκτελεί τα δεδομένα του προγραμματιστή αλλά τα παρουσιάζει σε ένα φιλικό για τον χρήστη γραφικο περιβάλλον. Το MATLAB διαθέτει ένα εργαλείο δημιουργιάς GUI που ονομάζεται ΟΔΗΓΟΣ (GUIDE) και προσφέρει στο χρήστη έναν point-and-click control της εφαρμογής. Πρόκειται για μια διεπαφή χρήστη σε γραφικό περιβάλλον που περιέχει components (κουμπιά ελέγχου, μενού, λίστες, γραφήματα, toolbars κτλ), και επιτρέπει στο χρήστη να εκτελεί διαδραστικές ερ-

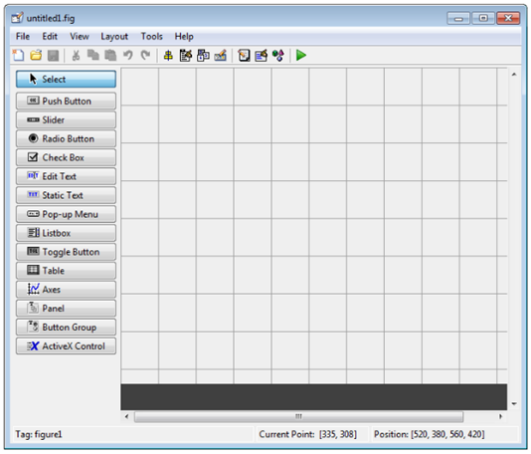
γασίες. Το βασικό πλεονέκτημά του είναι ότι ο χρήστης δεν χρειάζεται να καταλαβαίνει τις λεπτομέρειες για το πώς λειτουργεί η κάθε διεργασία. Μια εφαρμογή σε MATLAB που χρησιμοποιεί το GUIDE μπορεί να εκτελέσει κάθε είδος υπολογισμού, εγγραφή και διάβασμα δεδομένων, επικοινωνία με άλλα GUI's και απεικόνιση δεδομένων ως γραφήματα ή πίνακες [**16**].



Διάγραμμα 3.2: Ένα απλό interface με το GUIDE

3.4.1 Το περιβάλλον του GUIDE

Για να ξεκινήσει το GUIDE, ο χρήστης πρέπει να γράψει guide στη γραμμή εντολών του MATLAB και να πατήσει enter. Τότε εμφανίζεται ένα μενου που προτρέπει το χρήστη να επιλέξει μεταξύ διάφορων επιλογών. Η default επιλογή είναι να δημιουργηθεί ένα Blank GUI και το αποτέλεσμα φαίνεται στο διάγραμμα.



Διάγραμμα 3.3: Περιβάλλον εργασίας του GUIDE

Κεφάλαιο 4

Μεθοδολογία Αλγορίθμου

4.1 Εισαγωγή

Παρακάτω θα αναλυθεί η μεθοδολογία του αλγορίθμου που έχει επιλεχτεί προς υλοποίηση σύμφωνα με το Research Article:A Fast Image Stitching Algorithm via Multiple-Constraint Corner Matching [**1**]. Στο σημείο αυτό πρέπει να αναφερθεί ότι στην συγκεκριμένη υλοποίηση, όσον αφορά την επιλογή γωνιών με τον αλγόριθμο Harris, χρησιμοποιείται ο παραδοσιακός αλγόριθμος και όχι αυτός που αναφέρεται στο Article. Γενικά έχει γίνει μια προσπάθεια το τελικό αποτέλεσμα να ακολουθεί τη μεθοδολογία που επιλέχτηκε, όμως λόγω ελλειπών πληροφοριών μπορεί κάποια σημεία να μην έχουν ακριβώς το επιθυμητό βέλτιστο αποτέλεσμα.

4.2 Επιλογή γωνιών

Όπως έχει αναφερθεί, το πρώτο βήμα για τη δημιουργία πανοράματος είναι ο εντοπισμός σημείων ενδιαφέροντος ανα δύο διαδοχικές εικόνες. Στην παρούσα εργασία τα σημεία ενδιαφέροντος θα είναι οι γωνίες.

Για τον εντοπισμό γωνιών χρησιμοποιείται ο αλγόριθμος Harris & Stephens. Έστω ότι η ένταση φωτεινότητας μιας εικόνας συμβολίζεται με $I(x, y)$. Η περιοχή μεταβολής για ένα pixel (x, y) συμβολίζεται με $w(x, y)$ και (u, v) είναι η μετατόπιση γύρω απο το pixel. Η αλλαγή έντασης στο pixel αυτό συμβολίζεται με $E(u, v)$ και μπορεί να υπολογιστεί απο την *εξίσωση* 4.1 όπου $M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = w(x, y) \otimes \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

($I(x)$ και $I(y)$ είναι η μερική παράγωγος του pixel αντίστοιχα, και w το φίλτρο του Gauss για την εξάλειψη του θορύβου) [**1**].

$$E(u,v) \equiv [u,v]M \begin{bmatrix} u \\ v \end{bmatrix} \tag{4.1}$$

Η συνάρτηση απόκρισης γωνίας ορίζεται στην εξίσωση *εξίσωση 4.2* όπου k είναι μια σταθερά που κυμαίνεται μεταξύ $[0.04, 0.06]$. Όσα pixel έχουν μια R τιμή που είναι μεγαλύτερη μιας τιμής κατωφλίου λ, επιλέγονται ως υποψήφιες γωνίες.

$$R = det(M) - k(trace(M))^2 \tag{4.2}$$

4.3 Αντιστοίχιση γωνιών βάσει πολλαπλών περιορισμών

Το δεύτερο βήμα είναι η αντιστοίχιση των σημείων ενδιαφέροντος (γωνιών) με κάποιο κριτήριο ομοιότητας. Ο παραδοσιακός αλγόριθμος RANSAC στην δημιουργία πανοράματος εικόνων δεν είναι τόσο αποτελεσματικός όταν πρόκειται για πολλές εικόνες με παρόμοια χαρακτηριστικά. Συνήθως όταν πρόκειται για εφαρμογή σε πραγματικό χρόνο οι υποκείμενες εικόνες έχουν υψηλή ομοιότητα ως προς τα χαρακτηριστικά τους η μια με την άλλη, αυτό σημαίνει μικρές διαφορές περιστροφής, κλιμάκωσης ή απόκλισης η μια με την άλλη. Εφαρμόζοντας λοιπόν πολλαπλούς περιορισμούς στα υποψήφια ζευγαρια για το ταίριασμα, αφαιρούνται τα λανθασμένα ζευγάρια και μειώνεται ο αριθμός επαναλήψεων που απαιτεί ο αλγόριθμος RANSAC. Η διαδικασία που ακολουθείται είναι η εξής:

4.3.1 Δημιουργία πίνακα ομοιότητας γωνιών μεταξύ γειτονικών εικόνων

Ας θεωρηθεί ότι μια εικόνα I έχει ανάλυση $W \times H$ $(Width \times Height)$ και η νιοστή γωνία k συμβολίζεται με I_k (με συντεταγμένες $I_k \cdot x$ και $I_k \cdot y$, και ένταση $I(x_k, y_k)$). Μια γωνία απο την αριστερή εικόνα I_i^l και μια απο την δεξιά I_i^r μπορούν να αντιστοιχούν μεταξύ τους εαν ικανοποιούνται οι παρακάτω προϋποθέσεις [**1**]:

i. Η απόλυτη διαφορά μεταξύ των y συντεταγμένων των δύο επιλεγμένων γωνιών δεν είναι μεγαλύτερη απο $H/3$.

ii. Η συντεταγμένη του x της αριστερής γωνίας είναι μεγαλύτερη ή ίση απο αυτήν της δεξιάς γωνίας.

iii. Υπάρχει υψηλή συσχέτιση της έντασης φωτεινότητας μεταξύ των δύο γωνιών.

Σύμφωνα με τα παραπάνω χρησιμοποιείται η *σχέση* 4.3 για τον υπολογισμό της ομοιότητας των αντίστοιχων γωνιών και απο το αποτέλεσμα παράγεται ένας πίνακας ομοιότητας μεταξύ των γειτονικών εικόνων I^l και I^r:

$$sim(i,j) = \begin{cases} |NCC(I_i^l, I_j^r)| & if\ |I_i^l \cdot y - I_j^r \cdot y| < \lambda_h, \\ & I_i^l \cdot x \geq I_j^r \cdot x \\ 0 & else. \end{cases} \tag{4.3}$$

Το λ_h είναι μια τιμή κατωφλίου (threshold) για την οποία η διαφορά μεταξύ των y συντεταγμένων δύο γωνιών πρέπει να είναι μικρότερη, και η κανονικοποιημένη συσχέτιση (NCC) είναι η εξίσωση που αναλύθηκε στη *παράγραφο* 2.5.1. Αν θεωρηθεί λοιπόν, ότι το παράθυρο ομοιότητας (similarity window) είναι μεγέθους $(2w+1) \times (2w+1)$; η κανονικοποιημένη συσχέτιση υπολογίζεται ως εξής:

$$NCC(I_i^l, I_j^r) = \frac{\sum_{u=-w}^{w} \sum_{v=-w}^{w} D_l(i,u,v) \cdot D_r(j,u,v)}{\sqrt{\sum_{u=-w}^{w} \sum_{v=-w}^{w} D_l(i,u,v)^2 \cdot \sum_{u=-w}^{w} \sum_{v=-w}^{w} D_r(j,u,v)^2}} \tag{4.4}$$

όπου

$$\begin{aligned} D_l(i,u,v) &= I^l(x_i + u, y_i + v) - \bar{I}_i^l, \\ D_r(j,u,v) &= I^r(x_j + u, y_j + v) - \bar{I}_j^r \end{aligned} \tag{4.5}$$

και $\bar{I}_i^l$,$\bar{I}_j^r$ είναι η μέση τιμή φωτεινότητας (mean) των παραθύρων γύρω απο τις γωνίες I_i^l και I_j^r αντίστοιχα. Έπειτα, όσα ζευγάρια γωνιών έχουν χαμηλή ομοιότητα μεταξύ τους φιλτράρονται και σύμφωνα με την *σχέση* 4.6, όπου το λ_n είναι η τιμή κατωφλίου (ένας πραγματικός αριθμός μεγαλύτερος απο 0.5) για την οποία θα γίνει η σύγκριση (similarity threshold).

$$sim(i,j) = \begin{cases} sim(i,j) & if\ sim(i,j) > \lambda_n, \\ 0 & else. \end{cases} \tag{4.6}$$

Δηλαδή, κάθε ζεύγος σημείων του πίνακα ομοιότητας που ικανοποιεί τον περιορισμό $sim(i,j) > \lambda_n$ παραμένει ως έχει, αλλίως για το ζεύγος αυτό η τιμή ομοιότητας θα είναι ίση με 0. Συνοπτικά, απο τις σχέσεις 4.3 και 4.6 υπολογίζεται ανα

ζεύγη (μια γωνία απο την αριστερή και μια την απο δεξιά εικόνα) η ομοιότητα γωνιών, $sim(i,j)$, και τελικά προκύπτει ένας πίνακας ομοιότητας μεταξύ των δύο εικόνων μεγέθους $Sum_l \times Sum_r$ όπου Sum_l είναι το πλήθος των γωνιών της αριστερής εικόνας και Sum_r της δεξιάς.

4.3.2 Δημιουργία συνόλου απο αντίστοιχα ζευγάρια γωνιών

Αφού έχει υπολογιστεί ο πίνακας ομοιότητας όλων των γωνιών δύο διαδοχικών εικόνων, με την παρακάτω διαδικασία θα παραχθεί ένα αρχικό σύνολο με τους δείκτες/τις θέσεις των ζευγών γωνιών (matching corners) που έχουν μεγάλη συσχέτιση μεταξύ τους:

Για κάθε γραμμή του πίνακα ομοιότητας $sim(i,j)$, επιλέγεται ο δείκτης της στήλης για τον οποίο η τιμή ομοιότητας σε αυτή τη θέση, είναι η μεγαλύτερη για αυτή τη γραμμή, και έτσι το ζευγάρι δείκτης/θέση γραμμή, δείκτης/θέση στήλη (row index,column index) προστίθεται στο αρχικό σύνολο. Αφού η διαδικασία επαναληφθεί για όλες τις γραμμές του πίνακα sim, προκύπτει ένα αρχικό σύνολο ζευγών δεικτών L^l . Η *σχέση 4.7* περιγράφει επίσημα αυτή τη διαδικασία, όπου Sum_l είναι το πλήθος γωνιών της αριστερής εικόνας I^l [**1**]:

$$L^l = \{(i,j) | \forall i \in [1, Sum_l], s(i,j) = max(s(i,:)), s(i,j) \neq 0\} \tag{4.7}$$

Αντίστοιχα, παράγεται ένα ακόμα αρχικό σύνολο ζευγών L^r, επιλέγοντας τον δείκτη γραμμής με τη μεγαλύτερη τιμή ομοιότητας για κάθε στήλη. Η *σχέση 4.8* περιγράφει επίσημα αυτή τη διαδικασία, όπου Sum_r είναι το πλήθος γωνιών της δεξιάς εικόνας I^r:

$$L^r = \{(i,j) | \forall j \in [1, Sum_r], s(i,j) = max(s(:,j)), s(i,j) \neq 0\} \tag{4.8}$$

Γενικά, τα Sum_l, Sum_r μπορούν να είναι διαφορετικού πλήθους. Στον συγκεκριμένο αλγόριθμο όμως, έχουν το ίδιο πλήθος. Αφού λοιπόν υπολογιστούν τα δύο αυτά σύνολα L^l, L^r ακολουθεί μια σύγκριση μεταξύ αυτών: Εαν ένας δείκτης γραμμής της αριστερής εικόνας και ένας δείκτης στήλης της δεξιάς, τυχαίνει να έχουν το ένα το άλλο σαν ”δεύτερο συστατικό” στο ζευγάρι, η ομοιότητα τους (στον πίνακα ομοιότητας sim) θα μεταβληθεί σε 1. Η *σχέση 4.9* περιγράφει αυτή τη διαδικασία:

$$sim_{updated}(i,j) = \begin{cases} 1 & if\ (i,j) \in L^l \cap L^r, \\ sim(i,j) & else. \end{cases} \tag{4.9}$$

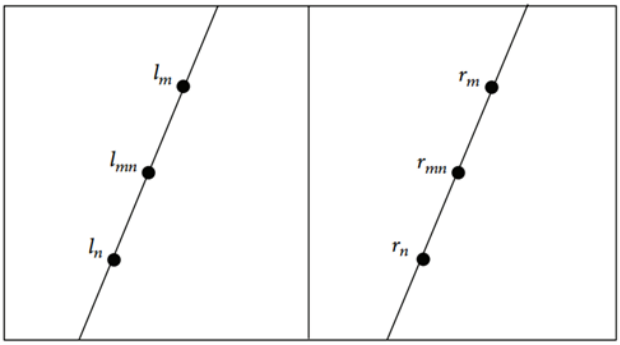
Το τελικό ενιαίο σύνολο απο ζεύγη γωνιών L προκύπτει απο την ένωση των L^l και L^r και ορίζεται στη *σχέση* 4.10. Να σημειωθεί ότι αυτό το σύνολο αυτό έχει ήδη μειωθεί σε αριθμό, όπως αναθέρθηκε στην *σχέση* 4.3 όπου κάποιες γωνίες φιλτραρίστηκαν με βάση τις θέσεις των συντεταγμένων.

$$L = L^l \cup L^r = \left\{(l,r)|(l,r) \in L^l \, or \, (l,r) \in L^r\right\} \tag{4.10}$$

4.3.3 Εφαρμογή πολλαπλών περιορισμών στα αντίστοιχα ζευγάρια γωνιών

Θεωρούνται δύο αρχικά ζευγάρια γωνιών στο *διάγραμμα* 4.1, $(l_m, r_m), (l_n, r_n)$ με τα αντίστοιχα μεσοσημεία τους, έστω l_{mn} μεταξύ των l_m, l_n και r_{mn} μεταξύ των r_m, r_n. Ορίζεται δ_m και $\bar{m}$ ως η κλίση και η απόσταση του τμήματος που σχηματίζεται μεταξύ του l_m, r_m αντίστοιχα, και $\delta_n, \bar{n}$ η κλίση και η απόσταση του τμήματος που σχηματίζεται μεταξύ του l_n, r_n. Σ'αυτά τα δύο αρχικά ζευγάρια θα εφαρμοστούν οι παρακάτω περιορισμοί [**1**]:

$$\begin{aligned} constraint1: \quad & |\delta_m - \delta_n| && < \lambda_\delta \\ constraint2: \quad & |\bar{m} - \bar{n}| && < \lambda_d \\ constraint3: \quad & |NCC(I^l_{l_{mn}}, I^r_{r_{mn}}| && > \lambda_n \end{aligned} \tag{4.11}$$



Διάγραμμα 4.1: Δύο αρχικά αντίστοιχα ζεύγη γωνιών

Η *σχέση* 4.11 δηλώνει πως μεταξύ δύο ζευγαριών, πρέπει όχι μόνο η τιμή της έντασης των αντίστοιχων μεσοσημείων αυτων, να είναι συσχετισμένη (constraint 3), αλλα και η κλίση (constraint 1) και απόσταση (constraint 2) των τμημάτων που σχηματίζονται μεταξύ των δύο ζευγαριών πρέπει να έχουν κάποια ορισμένη ομοιότητα. Σύμφωνα

λοιπόν με τους πολλαπλούς περιορισμούς που ορίσταν στην παραπάνω *σχέση* 4.11, υπολογίζεται η ομοιότητα μεταξύ όλων των αρχικών ζευγαριών γωνιών, με χρήση της *σχέσης* 4.12 και τελικά παράγεται ένας πίνακας D με μέγεθος $K \times K$, με K να είναι πληθικότητας L που προκύπτει απο την *σχέση* 4.10.

$$D(m,n) = \begin{cases} \frac{sim_{updated}(l_m,r_m)+sim_{updated}(l_n,r_n)+|NCC(I^l_{mn},I^r_{mn})|}{3}, \\ \qquad if\ satisfying\ all\ 3\ constraints \\ 0 \qquad\qquad else \end{cases} \tag{4.12}$$

4.3.4 Δημιουργία τελικού περιορισμένου συνόλου απο αντίστοιχα ζευγάρια γωνιών

Σε αυτό το σημείο θα υπολογιστεί το τελικό σύνολο ζευγαριών γωνιών. Απο το σύνολο αρχικών ζευγαριών γωνιών πλήθους K, σύμφωνα με την *σχέση* 4.13, βρίσκεται ένα ιδιαίτερο ζευγάρι t, το οποίο έχει την μεγαλύτερη συσχέτιση μεταξύ όλων των υπόλοιπων ζευγαριών [**1**]:

$$t = argmax_{i\in[1,K]} \left(\sum_{j=1}^{K} D(i,j) \right) \tag{4.13}$$

Έπειτα απο τον πίνακα D που προέκυψε προηγουμένως βρίσκονται όλα τα αρχικά ζευγάρια που έχουν κάποια συσχέτιση με το ζευγάρι t. Ένα ζευγάρι γωνιών θα ανήκει στο τελικό σύνολο, εφόσον η αντίστοιχη τιμή του D και το t είναι διάφορα του μηδενός. Η *σχέση* 4.14 περιγάφει επίσημα τη διαδικασία, όπου το είναι το τελικό μειωμένο σύνολο ζευγαριών γωνιών.

$$T = \{l | \forall l \in L, D(t,l) \neq 0\} \tag{4.14}$$

4.4 Δημιουργία Πανοράματος

Αφού παραχθεί το τελικό σύνολο απο ζεύγη μεταξύ των δύο εικόνων, επιλέγεται μια εικόνα ως εικόνα αναφοράς και υπολογίζεται ο γεωμετρικός μετασχηματισμός τους με χρήση του αλγόριθμου RANSAC. Βάση των παραμέτρων που προκύπτουν γίνεται το ταίριασμα των εικόνων, χαρτογραφώντας τις συντεταγμένες των pixel της άλλης εικόνας στο σύστημα συντεταγμένων της εικόνας αναφοράς [**1**].

Κεφάλαιο 5

Το γραφικό περιβάλλον της εφαρμογής

5.1 Εισαγωγή

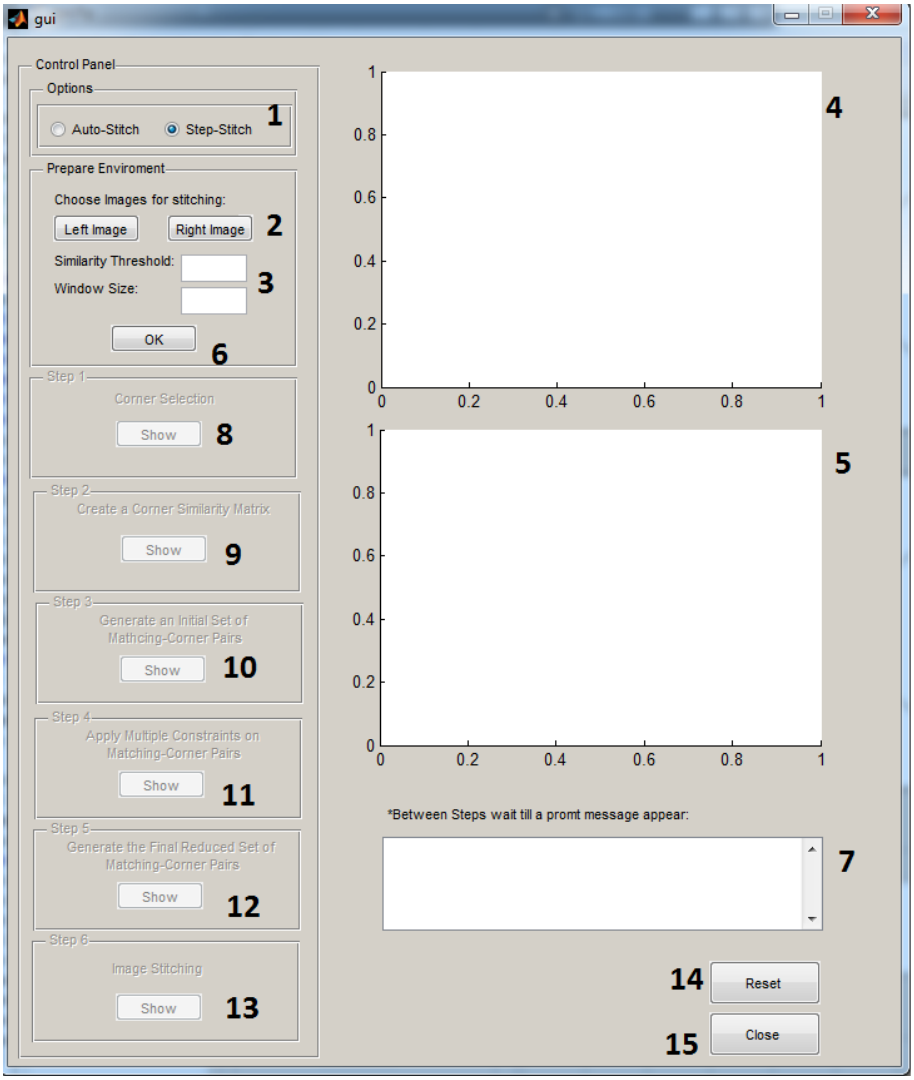
Στο τρέχον κεφάλαιο θα παρουσιαστεί το γραφικό περιβάλλον της εφαρμογής, οι λειτουργίες του και ο τρόπος χρήσης.

5.2 Δυνατότητες χρήσης

- Εισαγωγή εικόνων για stitching ανα δύο
- Εισαγωγή τιμής κατωφλίου που επιλέγει ο χρήστης
- Επιλογή αυτοματοποιημένης ή ανα βήμα εμφάνισης της διαδικασίας
- Αποθήκευση αποτελέσματος (εξ ορισμού του matlab)
- Επανεκκίνηση διαδικασίας
- Κλείσιμο εφαρμογής

5.3 Περιγραφή Λειτουργιών

Στο *διάγραμμα 5.1* φαίνεται το γραφικό περιβάλλον της εφαρμογής. Αποτελείται απο το κεντρικό παράθυρο που περιέχει ένα πάνελ με τις διάφορες λειτουργίες, δύο διαγράμματα, ένα πλαίσιο όπου εμφανίζονται μηνύματα και τα κουμπία επανεκκίνησης και εξόδου. Πιο συγκεκριμένα:



Διάγραμμα 5.1: Το gui της εφαρμογής

1. Ο χρήστης έχει δύο επιλογές: Auto-Stitch όπου αφού εισαχθούν οι εικόνες και η τιμή κατωφλίου, εμφανίζεται το τελικό αποτέλεσμα χωρίς να δίνεται στον χρήστη η δυνατότητα παρακολούθησης. Step-Stitch, όπου ο χρήστης έχει τον έλεγχο και παρακολουθεί βήμα βήμα την διαδικασία και τα αποτελέσματα.

2. Εισαγωγή εικόνων απο τον υπολογιστή: Left Image η αριστερή εικόνα και Right Image η δεξιά εικόνα για stitch.

3. Similarity Threshold: πρόκειται για μια τιμή κατωφλίου που επηρεάζει όλο το αποτέλεσμα και τον χρόνο εκτέλεσης. Όσο μικρότερο είναι τόσο περισσότερες γωνίες θα εντοπιστούν. Εμπειρικά η τιμή του δεν θα είναι μικρότερη απο 100000.

4. Εμφανίζεται η αριστερή εικόνα.

5. Εμφανίζεται η δεξιά εικόνα.

6. Με το πάτημα του button ξεκινάει η διαδικασία. Τα κουμπία εισαγωγής εικόνας και αλλαγής κατωφλίου απενεργοποιούνται.

   Τα παρακάτω (8 με 13) ισχύουν εφόσον έχει επιλεχθεί η επιλογή Step-Stitch, στην περίπτωση του Auto-Stitch θα είναι απενεργοποιημένα:

7. Για κάθε ένα απο τα παρακάτω βήματα εμφανίζεται ένα μήνυμα σε αυτό το πλαίσιο που ενημερώνει τον χρήστη οτι η διαδικασία ολοκληρώθηκε πρίν προχωρήσει στο επόμενο βήμα.

8. Step 1: Με το πάτημα του button εμφανίζονται σε διαγράμματα οι γωνίες που εντοπίστηκαν και ένας πίνακας με τις τιμές αυτών.

9. Step 2: Με το πάτημα του button υπολογίζεται και εμφανίζεται ο πίνακας ομοιότητας.

10. Step 3: Με το πάτημα του button υπολογίζεται και εμφανίζεται ένας πίνακας που περιέχει το αρχικό σύνολο αντίστοιχων ζευγαριών γωνιών.

11. Step 4: Με το πάτημα του button υπολογίζεται και εμφανίζεται ένας πίνακας μετά την εφαρμογή πολλαπλών περιορισμών.

12. Step 5: Με το πάτημα του button υπολογίζεται και εμφανίζεται ένας πίνακας με το τελικό σύνολο των αντίστοιχων ζευγαριών γωνιών και ένα διάγραμμα με τις αντιστοιχίες.

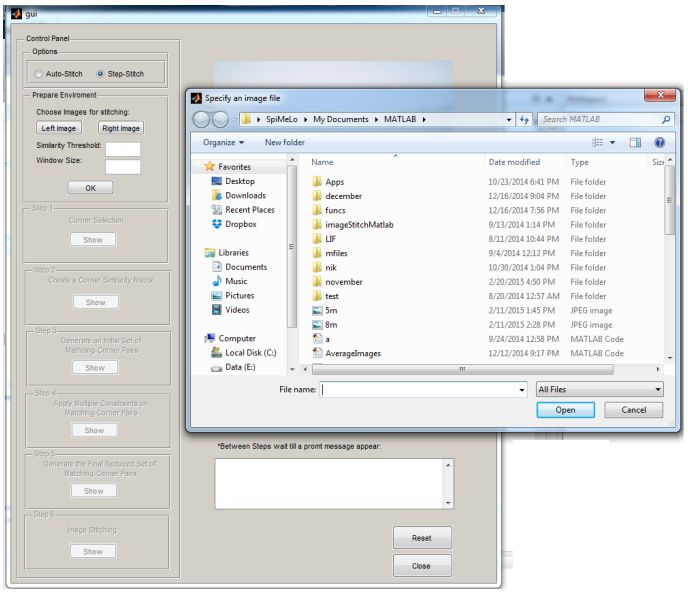
13. Step 6: Το τελικό αποτέλεσμα. Ένα διάγραμμα με τις αντιστοιχίες, ένα διάγραμμα με το πανόραμα σε grayscale και ένα σε rgb.

14. Κάνει reset η εφαρμογή (κλείνουν ό,τι παράθυρα είναι ανοιχτά).

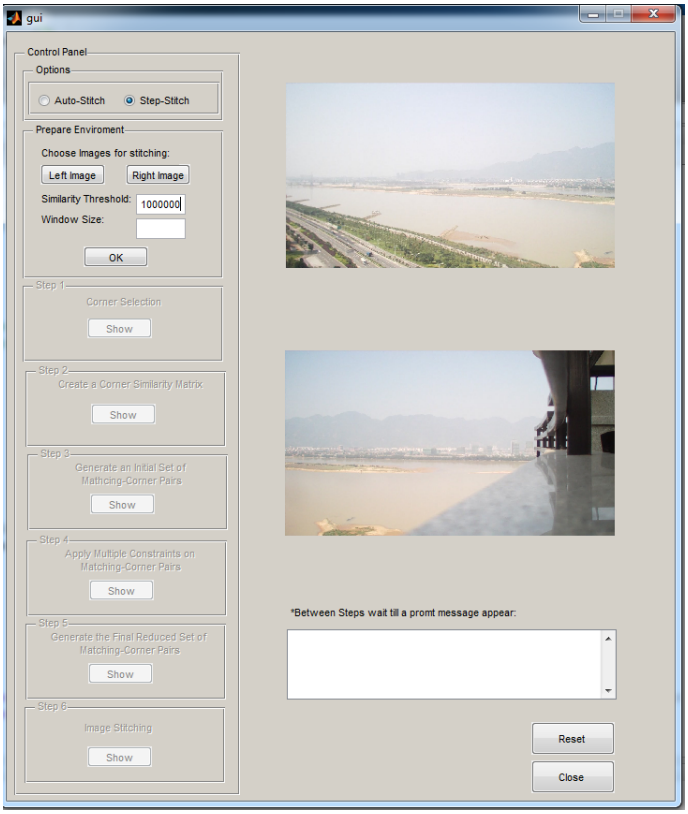
15. Κλείσιμο εφαρμογής.

5.4 Παράδειγμα τρόπου χρήσης

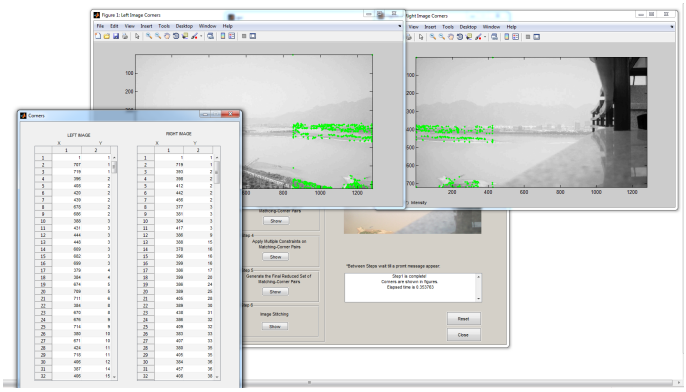
Παρακάτω εμφανίζεται σε διαγράμματα η περίπτωση χρήσης με το Step-Stitch επιλεγμένο.



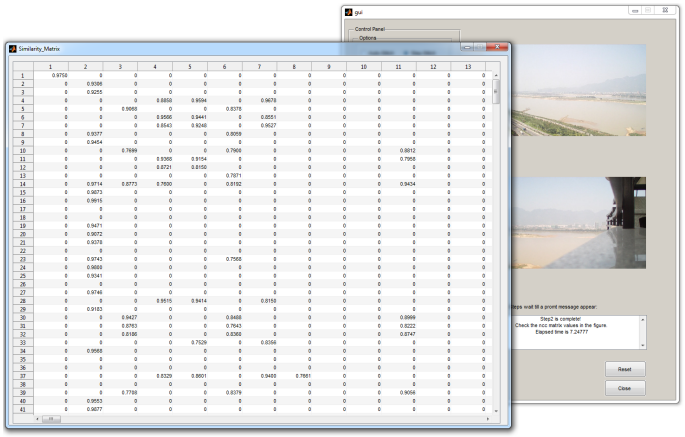
Διάγραμμα 5.2: Ο χρήστης εισάγει δύο εικόνες



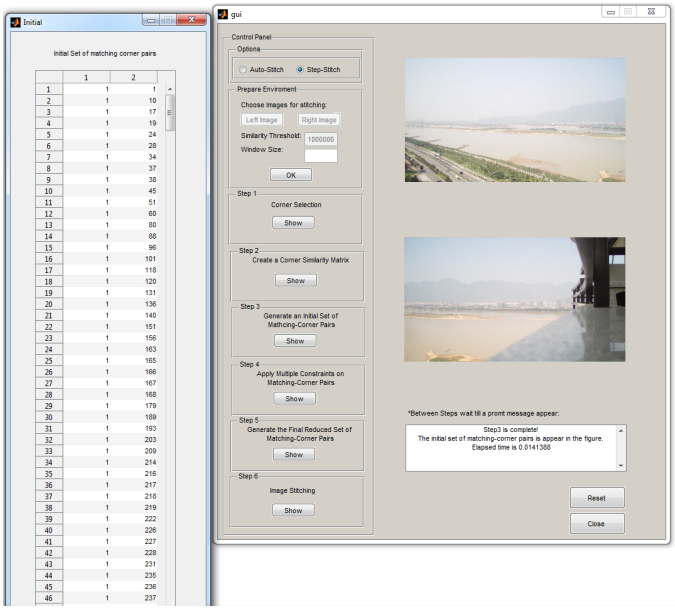
Διάγραμμα 5.3: Ο χρήστης ορίζει τιμή κατωφλίου



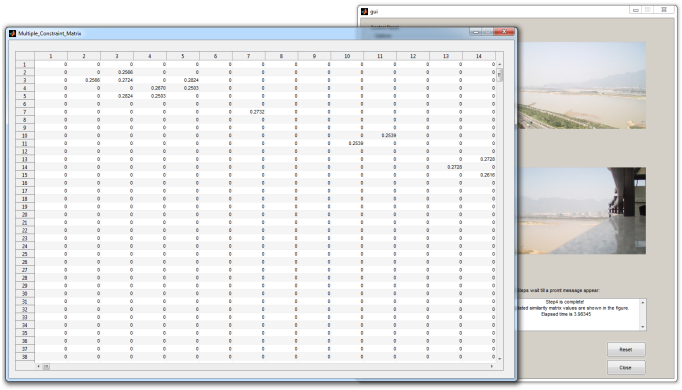
Διάγραμμα 5.4: Step 1 Τί εμφανίζεται



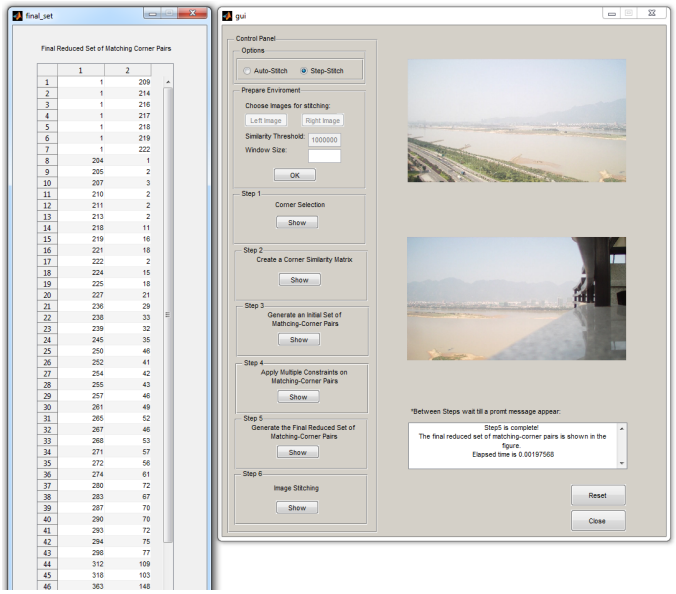
Διάγραμμα 5.5: Step 2 Τί εμφανίζεται



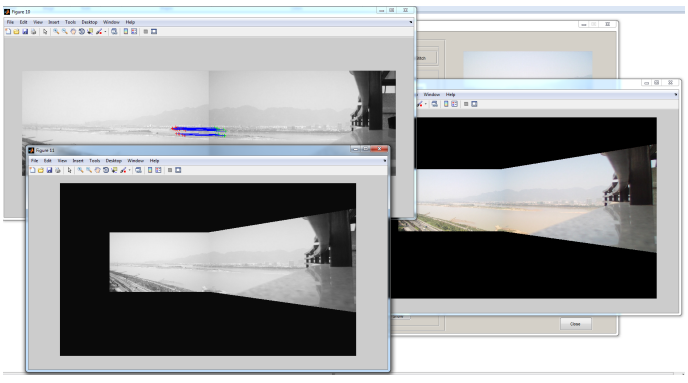
Διάγραμμα 5.6: Step 3 Τί εμφανίζεται



Διάγραμμα 5.7: Step 4 Τί εμφανίζεται



Διάγραμμα 5.8: Step 5 Τί εμφανίζεται



Διάγραμμα 5.9: Step 6 Τί εμφανίζεται

Κεφάλαιο 6

Περιγραφή σχεδίασης του συστήματος

Σε αυτό το κεφάλαιο θα αναφερθούν οι κύριες μέθοδοι υλοποίησης του αλγορίθμου όπως αναλύθηκε στο *κεφάλαιο 4*. Συγκεκριμένα θα περιγραφούν οι συναρτήσεις που χρησιμοποιήθηκαν και θα δωθεί έμφαση σε όσα σημεία κρίνεται αναγκαίο για την καλύτερη κατανόηση της λειτουργίας του αλγορίθμου.

Για να υπάρχει συνοχή και ροή στην ακόλουθη περιγραφή, το κεφάλαιο αυτό έχει χωριστεί σε υπο-ενότητες ίδιες με τα βήματα του *κεφαλαίου 4* όπου για κάθε μια απο αυτές θα ακολουθεί μια μικρή ανάλυση στα MFiles με τις συναρτήσεις που τα αποτελούν.

6.1 Προετοιμασία Περιβάλλοντος

Αρχικά ο χρήστης εισάγει τις δύο εικόνες με συνέχεια μεταξύ τους για τη δημιουργία πανοράματος. Πρόκειται για το αρχείο *readImage.m* που περιέχει την ακόλουθη συνάρτηση:

```
1 function [img1,img2,img1Region,img2Region,img1rgb,img2rgb] = ...
      readImage(img1,img2)
```

Η συνάρτηση διαβάζει τις δύο εικόνες, τις μετατρέπει σε *grayscale* και με την εντολή *resize* φροντίζει να έχουν κ οι δύο το επιθυμητό ίδιο μέγεθος που είναι 1280×720. Έπειτα επιλέγει/απομονώνει το δεξί-ένα τρίτο της αριστερής εικόνας και το αριστερό-ένα τρίτο της δεξιάς γιατί είναι η περιοχή της έκαστης εικόνας που χρειάζεται για τη

δημιουργία πανοράματος (*περιορισμός i, παράγραφος 4.3.1*). Στη συνέχεια ο τύπος των νέων εικόνων αλλάζει σε *double*. Η συνάρτηση δεν δέχεται ορίσματα, παρα μόνο επιστρέφει τα ακόλουθα:

- *img1*: η αρχική αριστερή εικόνα σε *grayscale*
- *img2*: η αρχική δεξιά εικόνα σε *grayscale*
- *img1Region*: η επιλεγμένη περιοχή της αριστερής εικόνας σε *grayscale* και τύπο *double*
- *img2Region*: η επιλεγμένη περιοχή της δεξιάς εικόνας σε *grayscale* και τύπο *double*
- *img1rgb*: η αρχική αριστερή εικόνα σε *rgb*
- *img2rgb*: η αρχική δεξιά εικόνα σε *rgb*

Για λόγους συντομίας, απο δώ και πέρα αριστερή εικόνα (imgLeftRegion) θα εννοείται η επιλεγμένη περιοχή της αριστερής εικόνας και αρχική αριστερη εικόνα (imgLeft), ολόκληρη η εικόνα. Αντίστοιχα και δεξιά εικόνα (imgRightRegion), αρχική δεξιά εικόνα (imgRight).

6.2 Επιλογή γωνιών

Σύμφωνα με την *παράγραφο 4.2* το πρώτο βήμα για τη δημιουργία πανόραμα είναι η επιλογή γωνιών για κάθε εικόνα. Ο αλγόριθμος που επιλέχθηκε για αυτό το σκοπό είναι των Harris & Stephens [**2**]. Το αρχείο που υλοποιεί τον αλγόριθμο είναι το *harris.m*, που περιέχει την συνάρτηση:

```
1 function [cim,r,c] = harris(im,sigma,thresh,radius,displ)
```

Η συνάρτηση είναι υλοποιημένη απο τον Peter Covesi [**17**] και έχει συμπεριληφθεί αυτούσια. Η συνάρτηση *harris* καλείται δύο φορές, μια για κάθε εικόνα. Τα ορίσματα που παίρνει ως είσοδο είναι:

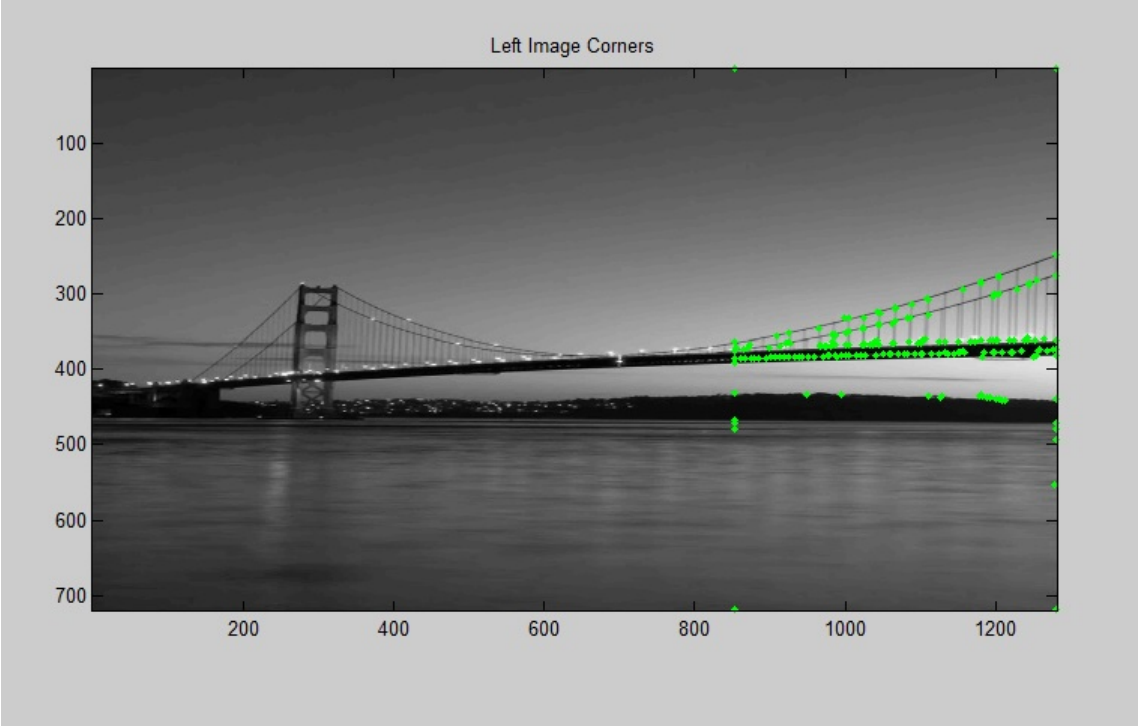
- *im*: η αριστερή ή δεξιά εικόνα για επεξεργασία

- *sigma*: η τυπική απόκλιση, με τιμή 1, που χρησιμοποιείται για το φίλτρο του Gauss
- *threshold*: η τιμή κατωφλίου που επηρεάζει την ποσότητα των γωνιών (όσο πιο μεγάλο τόσο λιγότερες γωνιές), ενδεικτικά παίρνει την τιμή 1000000
- *radius*: είναι η ακτίνα της περιοχής μέσα στην οποία θα κρατήσει μόνο το σημείο με την μέγιστη απόκριση. Χρησιμεύει στο να περιορίζεται το πλήθος των σημείων που επιστρέφει ο αλγόριθμος harris. Αντίστοιχα περιορίζει το πλήθος και το threshold. Τιμή μεταξύ 1-3 (optional).

Η συνάρτηση επιστρέφει στην έξοδο τα εξής:

- *r*: η γραμμή ή x συντεταγμένη της κάθε γωνίας (πίνακας $N \times 1$ όπου N το πλήθος των γωνιών που βρέθηκαν)
- *c*: η στήλη ή y συντεταγμένη της κάθε γωνίας (πίνακας $N \times 1$ όπου N το πλήθος των γωνιών που βρέθηκαν)

Το *διάγραμμα 6.1* εμφανίζει τις γωνίες που βρέθηκαν σε μια εικόνα μετά την εκτέλεση του *harris*.



Διάγραμμα 6.1: Harris Corner Detector

6.3 Δημιουργία πίνακα ομοιότητας γωνιών μεταξύ γειτονικών εικόνων

Αφού έχουν βρεθεί οι γωνίες, το επόμενο βήμα είναι να γίνει αντιστοίχιση αυτών με κάποιο κριτήριο ομοιότητας. Για το σκοπό αυτό χρησιμοποιείται ο αλγόριθμος NCC(*παράγραφος 4.3.1*). Το αρχείο που υλοποιεί τον αλγόριθμο είναι το *ncc.m*, που περιέχει τις συναρτήσεις:

```
1 function [I] = getWindow(img,x,y,winlen)
2 function [similarityMatrix] = ...
      ncc(imgLeftRegion,imgRightRegion,winSize,Xl,Yl,Xr,Yr)
```

Η συνάρτηση *getWindow* [**18**] δημιουργεί ένα παράθυρο γύρω απο μια δοσμένη τιμή της εικόνας (εδώ είναι η γωνία), προσέχοντας να μην ξεπερνάει τα όρια μεγέθους αυτής. Αρχικά ελέγχει ότι τα σημεία που δόθηκαν είναι εντός των ορίων της εικόνας. Απο τον τύπο $(2winlen + 1) \times (2winlen + 1)$ υπολογίζεται το μήκος της μιας πλευράς του παραθύρου και το $winlen$ που προκύπτει χρησιμοποιείται για να οριστούν η αρχική και τελική τιμή του παραθύρου που θα δημιουργηθεί. Αφαιρώντας την συντεταγμένη της γωνίας X ή Y αντίστοιχα απο το $winlen$ θα προκύψει η η αρχική τιμή του πίνακα-παράθυρο ενώ προσθέτοντας την τιμή του $winlen$, προκύπτει η τελική τιμή. Το παράθυρο που δημιουργείται θα έχει όρια [αρχική τιμή X : τελική τιμή X] και [αρχική τιμή Y : τελική τιμή Y]. Τα ορίσματα που δέχεται ως είσοδο είναι,

- *img*: η εικόνα για επεξεργασία
- *x*: η x συντεταγμένη (ή γωνία) του κέντρου του παραθύρου που θα δημιουργηθεί
- *y*: η y συντεταγμένη (ή γωνία) του κέντρου του παραθύρου που θα δημιουργηθεί
- *winlen*: το μέγεθος της μιας πλευράς του παραθύρου (το μέγεθος ολόκληρου του παραθύρου δίνεται απο την σχέση $(2winlen + 1) \times (2winlen + 1)$)

Δίνει στην έξοδό της:

- I : το παράθυρο που δημιουργήθηκε γύρω απο τη γωνία (πάντα εντός των διαστάσεων της εικόνας)

Έπειτα, η συνάρτηση *ncc* υπολογίζει τις τιμές της κανονικοποιημένης συσχέτισης σύμφωνα με τον τύπο ικανοποιώντας τις συνθήκες και περιορισμούς της *παραγράφου 4.3.1*. Δέχεται ως ορίσματα τα εξής:

- *imgLeftRegion*: η αριστερή εικόνα (ορίζεται ως αναφοράς)
- *imgRightRegion*: η δεξιά εικόνα (όπου τα αντίστοιχα σημεία πρέπει να οριστούν)
- *winSize*: το μέγεθος του παραθύρου
- *Xl*: η x συντεταγμένες των γωνιών της αριστερής εικόνας (πίνακας $N \times 1$ όπου N το πλήθος των γωνιών που βρέθηκαν)
- *Yl*: η y συντεταγμένες των γωνιών της αριστερής εικόνας (πίνακας $N \times 1$ όπου N το πλήθος των γωνιών που βρέθηκαν)
- *Xr*: η x συντεταγμένες των γωνιών της δεξιάς εικόνας (πίνακας $N \times 1$ όπου N το πλήθος των γωνιών που βρέθηκαν)
- *Yr*: η y συντεταγμένες των γωνιών της δεξιάς εικόνας (πίνακας $N \times 1$ όπου N το πλήθος των γωνιών που βρέθηκαν)

Και επιστρέφει,

- *similarityMatrix*: ο πίνακας ομοιότητας μεγέθους $N1 \times N2$ (οι τιμές της κανονικοποιημένης συσχέτισης) όπου $N1, N2$ είναι το πλήθος των γωνιών της αριστερής και δεξιάς εικόνας αντίστοιχα.

Συνοπτικά αυτό που συμβαίνει όταν εκτελείται η συνάρτηση ncc είναι: Για κάθε γωνία της αριστερής εικόνας δημιουργείται ένα παράθυρο γύρω απο αυτή μεγέθους $winSize$ και συγκρίνεται με κάθε γωνία της δεξιάς εικόνας (δημιουργείται αντίστοιχα ένα παράθυρο μεγέθους $winSize$ γύρω απο την επιλεγμένη γωνία) σύμφωνα με τον τύπο της κανονικοποιημένης συσχέτισης. Τελικά δημιουργείται ένας πίνακας που περιέχει τη σχέση κάθε γωνίας της αριστερής εικόνας με κάθε γωνία της δεξιάς.

6.4 Δημιουργία συνόλου απο αντίστοιχα ζευγάρια γωνιών

Ως εδώ έχει προκύψει με βάση κάποιους πρώτους περιορισμούς ο πίνακας ομοιότητας $similarityMatrix$. Επίσης σημαντικές παράμετροι είναι οι πίνακες Xl, Yl (συντεταγμένες γωνιών αριστερής εικόνας) και Xr, Yr (συντεταγμένες γωνιών δεξιάς εικόνας) που έχουν μέγεθος $N \times 1$ (N το πλήθος των γωνιών που βρέθηκαν). Στην συνέχεια θα πρέπει να δημιουργηθεί ένα πρώτο σετ (σύνολο) απο ζευγάρια γωνιών της αριστερής και δεξιάς εικόνας για αντιστοίχιση. Αυτό γίνεται μέσω της διαδικασίας που περιγράφηκε στην *παράγραφο 4.3.2*. Το αρχείο *initialSet.m* υλοποιεί αυτή τη διαδικασία και περιέχει την συνάρτηση:

```
1 function [L,ncc] = initialSet( ncc,leftcoor,rightcoor )
```

Η συνάρτηση παίρνει ως ορίσματα [**19**],

- *ncc*: ο πίνακας ομοιότητας
- *leftcoor*: οι συντεταγμένες των γωνιών της αριστερής εικόνας (της μορφής [X';Y'])
- *rightcoor*: οι συντεταγμένες των γωνιών της δεξιάς εικόνας (της μορφής [X';Y'])

Και επιστρέφει

- L: το σύνολο απο αντίστοιχα ζευγάρια γωνίες (μεγέθους $N \times 2$ όπου column 1 τα σημεία απο αριστερά και column 2 τα σημεία απο δεξιά)
- *ncc*: ο updated πίνακας ομοιότητας

6.5 Εφαρμογή πολλαπλών περιορισμών στα αντίστοιχα ζευγάρια γωνιών

Αφού έχει υπολογιστεί το αρχικό σύνολο αντίστοιχων ζευγαριών γωνιών, με την διαδικασία που περιγράφεται στην *παράγραφο 4.3.3* το σύνολο αυτό περιορίζεται ακόμη περισσότερο ώστε τελικά να εφαρμοστεί ο αλγόριθμος RANSAC. Το αρχείο που υλοποιεί την διαδικασία είναι το *multipleConstraints.m*. Περιέχει τις συναρτήσεις

```
1 function [D] = ...
      multipleConstraints(imgLeftRegion,imgRightRegion, ...
      winSize,Lunion,Xleft, Yleft,Xright,Yright,simMatrix)
2 function [norm] = ncc2(imgLeft,imgRight,winSize,Xl,Yl,Xr,Yr)
```

Η *ncc2* είναι αυτή που περιγράφηκε παραπάνω (*ncc.m*) με τη διαφορά ότι αλλάζουν οι περιορισμοί.

Η συνάρτηση *multipleConstraints* δέχεται τα εξής ορίσματα:

- *imgLeftRegion*: η αριστερή εικόνα
- *imgRightRegion*: η δεξιά εικόνα
- *winSize*: το μέγεθος του παραθύρου
- *Lunion*: το αρχικό σύνολο απο αντίστοιχα ζευγάρια γωνίες (μεγέθους $N \times 2$ όπου column 1 τα σημεία απο αριστερά και column 2 τα σημεία απο δεξιά)
- *Xleft*: η x συντεταγμένες των γωνιών της αριστερής εικόνας (πίνακας $N \times 1$ όπου N το πλήθος των γωνιών που βρέθηκαν)
- *Yleft*: η y συντεταγμένες των γωνιών της αριστερής εικόνας (πίνακας $N \times 1$ όπου N το πλήθος των γωνιών που βρέθηκαν)
- *Xright*: η x συντεταγμένες των γωνιών της δεξιάς εικόνας (πίνακας $N \times 1$ όπου N το πλήθος των γωνιών που βρέθηκαν)
- *Yright*: η y συντεταγμένες των γωνιών της δεξιάς εικόνας (πίνακας $N \times 1$ όπου N το πλήθος των γωνιών που βρέθηκαν)
- *simMatrix*: ο πίνακας ομοιότητας

Και επιστρέφει:

- *D*: ένας πίνακας μεγέθους $N \times N$ (όπου N το πλήθος του Lunion) που παράγεται απο τους πολλαπλούς περιορισμούς και χρησιμοποιείται έπειτα για τον υπολογισμό του τελικού συνόλου αντίστοιχων ζευγαριών-γωνιών

Αυτό που υλοποιείται εδώ είναι οι πολλαπλοί περιορισμοί της σχέσης *4.11*. Για την κλιση χρησιμοποιείται ο γενικός τύπος της κλίσης $\frac{y2-y1}{x2-x1}$ και ως απόσταση ο τύπος $\sqrt{(y2-y1)^2+(x2-x1)^2}$.

6.6 Δημιουργία τελικού περιορισμένου συνόλου απο αντίστοιχα ζευγάρια γωνιών

Στο σημείο αυτό απο τον πίνακα D που προέκυψε προηγουμένως, υπολογίζεται το τελικό σετ απο αντίστοιχα ζευγάρια γωνίες σύμφωνα με την *παράγραφο 4.3.4*. Το αρχείο και η συνάρτηση που υλοποιεί την διαδικασία είναι το *argMax.m*:

```
1 function [T] = argMax(D, Lunion)
```

Η συνάρτηση παίρνει ως είσοδο:

- D: ο πίνακας που προέκυψε απο τους πολλαπλούς περιορισμούς της προηγούμενης παραγράφου
- *Lunion*: το αρχικό σύνολο απο ζευγάρια γωνίες

Και επιστρέφει,

- T: το τελικό σετ απο αντίστοιχα ζευγάρια-γωνίες

6.7 Δημιουργία Πανοράματος

Αφού έχει υπολογιστεί το τελικό σύνολο απο αντίστοιχα ζευγάρια γωνίες, χρησιμοποιείται ο αλγόριθμος RANSAC απο τόν οποίο προκύπτει ένα σύνολο απο inliers, που θα χρησιμοποιηθούν τελικά για τη δημιουργία πανοράματος. Αυτό που κάνει ο RANSAC είναι να επιλέξει απο το σύνολο αντίστοιχων ζευγαριών γωνιών, όσο το δυνατόν πιο πολλά έγκυρα σημεία για την τελική αντιστοίχιση. Η διαδικασία του RANSAC περιγράφεται στην *παράγραφο 2.6*.

Το αρχείο που υλοποιεί την διαδικασία είναι το *mathcesWithRansac.m*:

```
1 function [H, inliers] = ...
      mathcesWithRansac(T, truecoor, Xleft, Yleft, ...
      Xright, Yright, imgLeft, imgRight, imgLeftRegion, t)
2 function [H, inliers] = ransacfithomography_vgg(x2, x1, t);
```

Οι συναρτήσεις που περιέχονται στην *ransacfithomography_vgg* έχουν συμπεριληφθεί αυτούσιες απο την υλοποίηση του Peter Covesi [**20**]. Έπειτα η διαδικασία ένωσης γίνεται με την βοήθεια της συνάρτησης *vgg_warp_H* του David Liebovitz [**21**].

Κεφάλαιο 7

Πειραματικά αποτελέσματα

7.1 Το περιβάλλον και οι παράμετροι

Οι παράμετροι έχουν οριστεί ως εξής: PC: CPU Intel Core i5 4670 @ 3.40 GHz, 8 GB memory, Matlab R2014a, image resolution 1280×720 . Other Parameters:

(i) Η διαφορά των y συντεταγμένων μεταξύ των εικόνων δεν είναι μεγαλύτερη απο $H/3$: $\lambda_h = H/3$.

(ii) Η οριζόντια επικάλυψη δεν είναι μεγαλύτερη απο $W/3$.

(iii) Η τιμή του κατωφλίου ομοιότητας λ_h στην *εξίσωση* *4.6* έχει οριστεί ίση με 0.75 και το μέγεθος του παραθύρου ομοιότητας σην *εξίσωση* *4.4* έχει οριστεί σε 7×7 (άρα το w είναι ίσο με 3) .

7.2 Εκτίμηση της αντιστοίχισης γωνιών

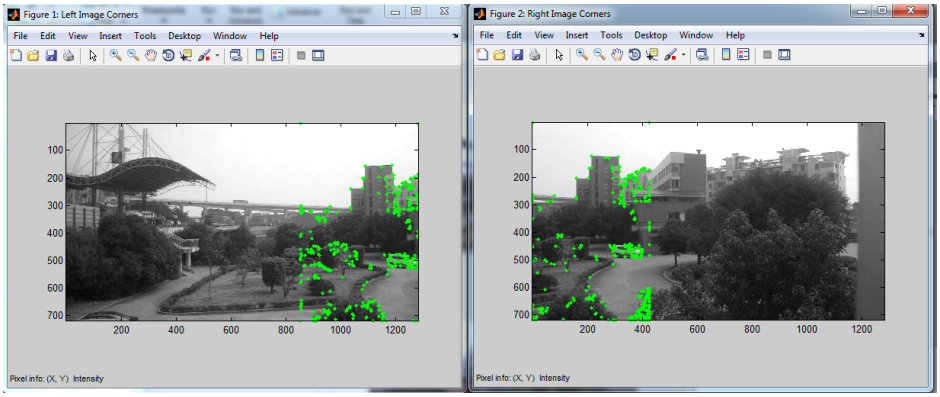
Τα πειραματικά αποτελέσματα εμφανίζονται στο *διάγραμμα* *7.1*. Για δύο αρχικές εικόνες με μερική επικάλυψη επιλέγονται οι γωνίες σύμφωνα με τον αλγόριθμο Harris & Stephens διάγραμμα 7.1 (α') . Επιλέγεται το δεξί ένα τρίτο της αριστερής εικόνας και το αριστερό ένα τρίτο της δεξιάς ως οι δύο περιοχές που θα εκτελεστεί το ταίριασμα. Για την αριστερή εικόνα το σύνολο των γωνιών που βρέθηκαν είναι ίσο με 390 και για την δεξιά 343. Στο διάγραμμα 7.1 (β') εμφανίζεται η αντιστοίχιση των γωνιών. Ο αριθμός των αντιστοιχισμένων γωνιών έχει περιοριστεί σε 133 μετά την εκτέλεση του NCC και των περιορισμών που περιγράφηκαν στην παράγραφο *4.3.3*. Έπειτα το

σύνολο των αντιστοιχισμένων γωνιών περιορίζεται ακόμα περισσότερο μετά την εκτέλεση του αλγορίθμου RANSAC όπως φαίνεται στο διάγραμμα 7.1 (γ') σε 70. Τελικά ο προτεινόμενος αλγόριθμος είναι αρκετά πιο γρήγορος λόγω των περιορισμών, αφού εάν δεν υπήρχαν, ο αλγόριθμος RANSAC θα έπρεπε να εκτελεστεί για όλες τις αντιστοιχίες γωνιών που προέκυψαν απο την εξίσωση NCC. Αυτό που συμβαίνει είναι ότι για την εκτέλεση του χρειάζεται να λάβει υπόψη ένα μικρό μέρος συνδυασμων μεταξύ των αντίστοιχων γωνιών, αφού ήδη απο την *εξίσωση 4.3* ο NCC αγνοεί όλες τις γωνίες που δεν πληρούν τους περιορισμούς θέσης και έπειτα αποφεύγει τον υπολογισμό εαν τα ζευγάρια δεν ικανοποιούν τους δύο πρώτους περιορισμούς της *εξίσωσης 4.11*.

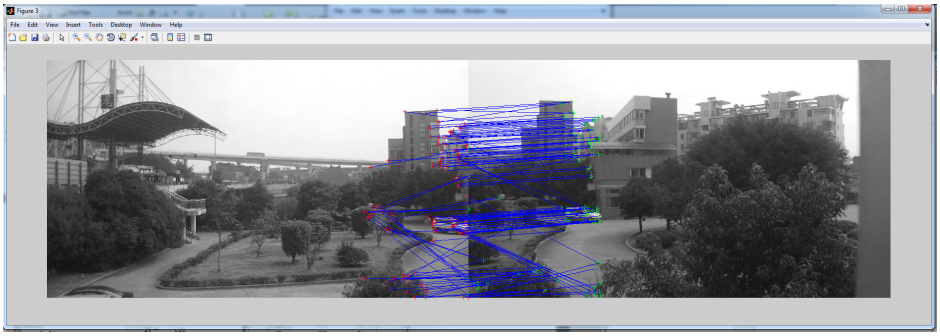
Ένα ακόμα παράδειγμα φαίνεται στο *διάγραμμα 7.2*, όπου το σύνολο των γωνιών για την αριστερή εικόνα είναι 676 και για την δεξιά 548. Μετά τον υπολογισμό της NCC και την αντιστοίχηση γωνιών, το σύνολο περιορίζεται σε 103 γωνιές και τελικά με την εκτέλεση του RANSAC σε 93.

7.3 Εκτίμηση της δημιουργίας πανοράματος

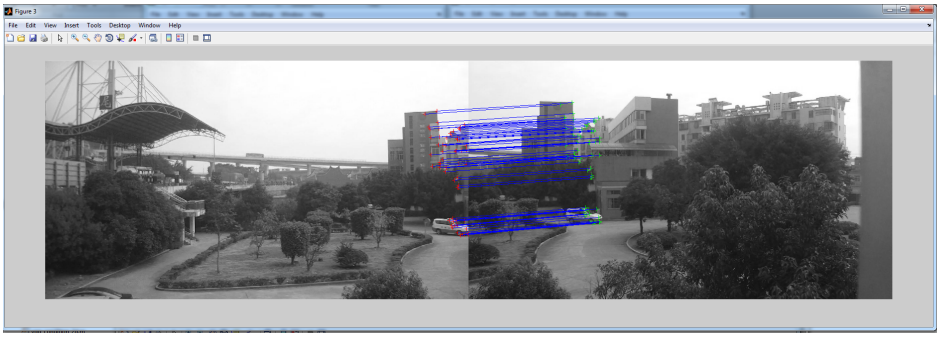
Αφού έχει εκτελεστεί ο αλγόριθμος Harris για τις δύο εικόνες και έχει παραχθεί το τελικό σύνολο των αντίστοιχων ζευγαριών για το ταίριασμα, επιλέγεται η αριστερή εικόνα ως εικόνα αναφοράς, και υπολογίζεται ο γεωμετρικός μετασχηματισμός με τη βοήθεια του αλγορίθμου RANSAC. Έπειτα "χαρτογραφούνται" οι συντεταγμένες της δεξιάς εικόνας (όχι της αναφοράς) στο σύστημα συντεταγμένων της αριστερής εικόνας (αναφοράς). Τα τελικά αποτελέσματα εμφανίζονται στα διαγράμματα 7.1, 7.2 (δ').



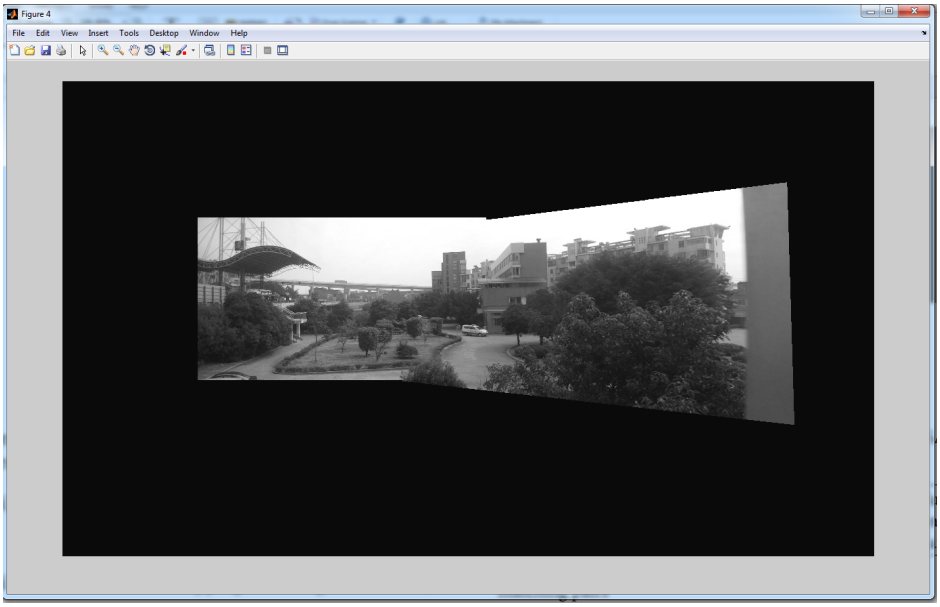
(α΄)



(β΄)



(γ΄)



(δ΄)

Διάγραμμα 7.1: Παράδειγμα 1

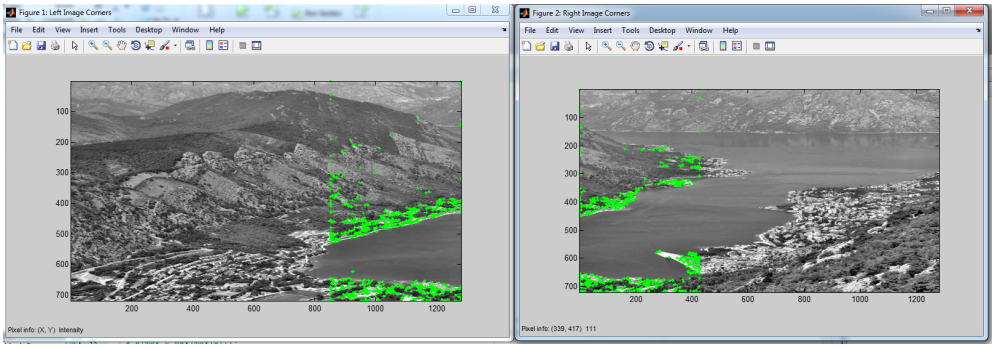


Figure 1: Left Image Corners
Figure 2: Right Image Corners

(α΄)

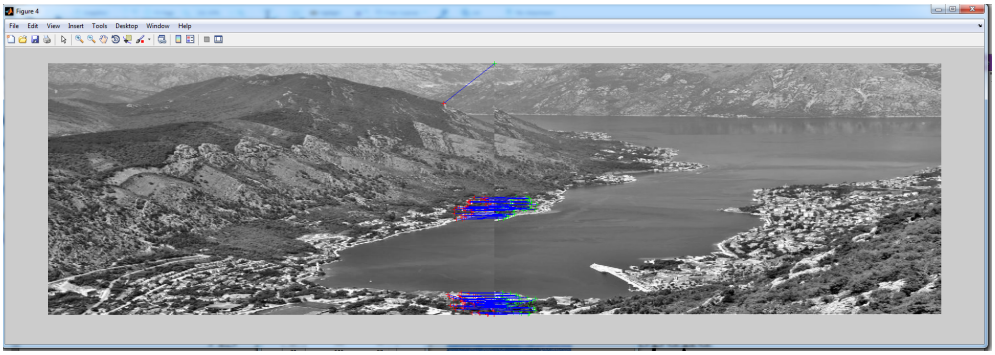


Figure 4

(β΄)

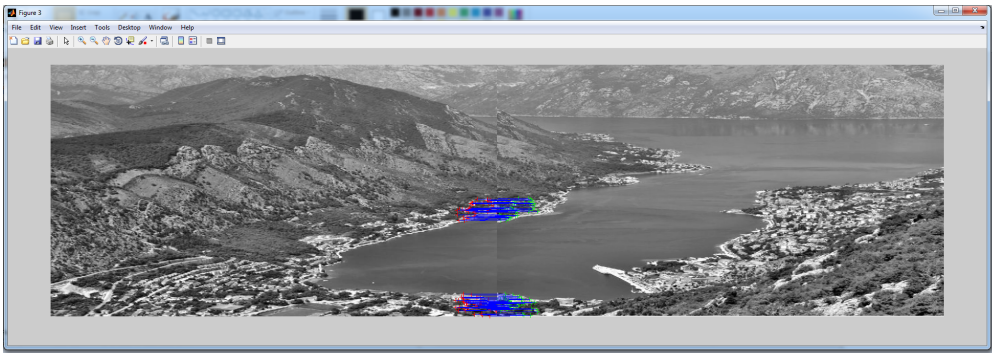


Figure 3

(γ΄)

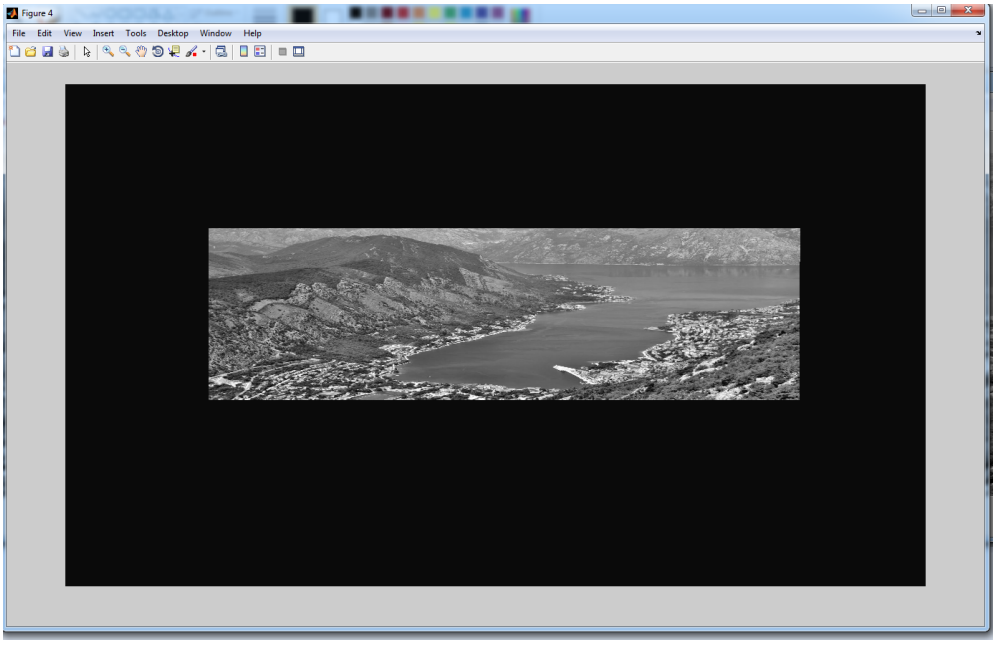


Figure 4

(δ΄)

Κεφάλαιο 8

Συμπεράσματα

Ο προτεινόμενος αλγόριθμος πραγματεύεται οτι ελαχιστοποιεί τα μειονεκτήματα των υπάρχοντων αλγορίθμων αφού μέσω τον πολλαπλών περιορισμών μειώνεται ο χρόνος εκτέλεσης. Στην παρούσα υλοποίηση πρέπει να σημειωθούν τα εξής:

Παρατηρήθηκε ότι ο συγκεκριμένος αλγόριθμος λειτουργεί πολυ αποδοτικά όταν πρόκειται για εικόνες ανάλυσης 1280×720. Στις περιπτώσεις όμως με εικόνες μεγαλύτερης ευκρίνειας, ο χρόνος εκτέλεσης αλλάζει δραματικά. Επισης οι περιορισμοί που αναφέρονται στην *εξίσωση 4.3* μπορεί να καταστήσουν το πρόγραμμα μη αποδοτικό, αφού όταν πρόκειται για απλούς χρήστες που τραβάνε φωτογραφίες σε πραγματικό χρονο, είναι πολύ εύκολο να μην τηρηθούν οι περιορισμοί. Αυτό θα έχει σαν αποτέλεσμα να θεωρηθεί το πρόγραμμα μη αξιόπιστο. Επιπλέον, ο RANSAC είναι ένας αλγόριθμος που λειτουργεί με βάση την πιθανότητα. Για αυτόν ακριβώς το λόγο η πιθανότητα να μην φέρει το επιθυμητό αποτέλεσμα είναι πάντα παρούσα, άσχετα με τους περιορισμούς που υποστηρίζει ο αλγόριθμος οτι ελαχιστοποιεί τα περιθώρια λάθους. Γενικά πρόκειται για έναν αποδοτικό και έξυπνα προσαρμοσμένο αλγόριθμο που όμως λειτουργεί υπο πολύ συγκεκριμένες συνθήκες, σχεδόν αμετάβλητες για ένα τέλειο αποτέλεσμα.

Εκτός απο τα παραπάνω, πρέπει να αναφερθεί ότι η συγκεκριμένη υλοποίηση δεν πληρεί στο μέγιστο το αποτέλεσμα που παρουσιάζεται στον προτεινόμενο αλγόριθμο [**1**], αφού κατα τη δημιουργία πανοράματος παρουσιάζεται κάποιο είδος παραμόρφωσης (”τραβήγματος”) της τελικής εικόνας προς τα δεξιά. Αυτό μπορεί να οφείλεται σε κάποιο είδος προβολικού μετασχηματισμού που εφαρμόζεται στον κώδικα του αλγορίθμου Ransac ο οποίος επιλέχθηκε αυτούσιος και ίσως δεν ταιριάζει με τα υπόλοιπα

δεδομένα 100%, θα μπορούσε επίσης να οφείλεται σε κάποια παρανόηση της προτεινόμενης μεθοδολογίας αφού οι πληροφορίες που δίνονται είναι αρκετά επιγραμματικές και η ακριβής μέθοδος που ακολουθήθηκε απο τους δημιουργούς δεν είναι γνωστή. Με παραπάνω πληροφορίες και πιο έμπειρους προγραμματιστές, το αποτέλεσμα του αλγορίθμου που προτείνεται θα είναι πιο αξιόπιστο.

Συνοψίζοντας, όσον αφορά το γραφικό και πρακτικό κομμάτι της εφαρμογής σε επίπεδο χρηστών, θα ήταν ενδιαφέρον να προστεθεί μια λειτουργία αυτόματης εύρεσης του κατάλληλου κατωφλίου για το βέλτιστο ταίριασμα, όπως επίσης και μια λειτουργία εισαγωγής πολλαπλών διαδοχικών εικόνων. Τέλος, όπως έχει ήδη αναφερθεί, ο καλύτερος προγραμματισμός του κώδικα αποτελεί βασικό πλεονέκτημα για την ταχύτητα και την αξιοπιστία μιας μελλοντικής επέκτασης της εφαρμογής αυτής.

Παράρτημα 1

Κώδικας

```
function Main(img1,img2,thresh)
% clc
% clearvars
% close all
% format
addpath(genpath('/SALIARI_AIKATERINI_1975_STITCH/matlabCode'))
tic;
disp('Prepare Enviroment:');

[imgLeft, imgRight, imgLeftRegion, imgRightRegion,l,r] = ...
    readImage(img1,img2);

% Other variables
winSize = 7;
sigma = 1;
radius = 1;
t = .001;   % Distance threshold for deciding outliers
displ = 0;

toc
fprintf('\n');
%----------------------
% 1) Corners detected
%----------------------
tic
```

```
25 disp('Corners Detected:');
26
27 [¬, Xleft, Yleft] = harris(imgLeftRegion, sigma, thresh, ...
       radius, displ); % 2 10000000 5 1000000 z 100000000 3 1000000
28 [¬, Xright, Yright] = harris(imgRightRegion, sigma, thresh, ...
       radius, displ); %8 1000000 912 20000000 city 10000000
29
30 truecoor = size(imgLeft,2)-size(imgLeftRegion,2);
31 set(figure,'position',get(0,'screensize'));
32 subplot(1,2,1),impixelinfo(imagesc(imgLeft)), axis image, ...
       colormap(gray), hold on; title('Left Image Corners'); ...
       plot(Yleft+truecoor,Xleft,'g.');
33 subplot(1,2,2),impixelinfo(imagesc(imgRight)), axis image, ...
       colormap(gray), hold on; title('Right Image Corners'); ...
       plot(Yright,Xright,'g.');
34 drawnow
35
36 toc
37 fprintf('\n');
38 %-------------------------------------
39 % 2) Create corner similarity matrix
40 %-------------------------------------
41 tic
42 disp('Calculate NCC and satisfy the 3 constraints:');
43
44 [similarityMatrix] = ncc(imgLeftRegion, imgRightRegion, ...
       winSize, Xleft, Yleft, Xright, Yright);
45
46 toc
47 fprintf('\n');
48 %-----------------------------------------------------
49 % 3) Generate an initial set of matching-corner pairs
50 %-----------------------------------------------------
51 tic
52 disp('Generate an initial set of matching-corner pairs:');
53
54 [¬,¬, Lunion, similarityMatrix] = initialSet( ...
       similarityMatrix,[Xleft';Yleft'],[Xright';Yright'] );
```

```
55
56 toc
57 fprintf('\n');
58 %-------------------------------------------------------
59 % 4) Apply Multiple Constraints on Matching-Corner Pairs
60 %-------------------------------------------------------
61 tic
62 disp('Apply Multiple Constraints on Matching-Corner Pairs:');
63
64 [D] = multipleConstraints(imgLeftRegion, imgRightRegion, ...
       winSize, Lunion, Xleft, Yleft, Xright, Yright, ...
       similarityMatrix );
65
66 toc
67 fprintf('\n');
68 %-------------------------------------------------------------
69 % 5) Generate the Final, Reduced Set of Matching-Corner Pairs
70 %-------------------------------------------------------------
71 tic
72 disp('Generate the Final, Reduced Set of Matching-Corner ...
       Pairs:');
73
74 [T] = argMax(D, Lunion);
75
76 toc
77 fprintf('\n');
78 %--------------------
79 % 6) Image Stitching
80 %--------------------
81 tic
82 disp('Image Stitching:');
83
84 [H,inliers] = mathcesWithRansac( T, truecoor, Xleft, Yleft, ...
       Xright, Yright, imgLeft, imgRight, imgLeftRegion, t );
85
86 bbox = [-600 3000 -600 1500]; % image space for mosaic
87
```

```
88  Im2w = vgg_warp_H(double(imgLeft), eye(3), 'linear', bbox); % ...
        warp image 1 to mosaic image
89  Im1w = vgg_warp_H(double(imgRight), H, 'linear', bbox);
90  figure; imshow(uint8(max(Im1w,Im2w)));
91
92  Im2w = vgg_warp_H(double(l), eye(3), 'linear', bbox); % warp ...
        image 1 to mosaic image
93  Im1w = vgg_warp_H(double(r), H, 'linear', bbox);
94  figure; imshow(uint8(max(Im1w,Im2w)));
95
96  toc
97  fprintf('\n');
98
99  end
100 %--------------------------------------------------------------------------
```

```
1  % READIMAGE - loads 2 images, convert to grayscale,change type,
2  %              saves the last one-third region from img1 and
3  %              the first one third region from img2
4  %
5  % Usage:  [img1, img2, img1Region, img2Region] = ...
       readImage(img1,img2)
6  %
7  % Arguments: img1        - left image.
8  %            img2        - right image.
9  % Returns:
10 %            img1        - left image (grayscale).
11 %            img2        - right image (grayscale).
12 %            img1Region  - last one-third region of left image.
13 %            img2Region  - first one-third region of right image.
14 %            img1rgb     - left image (rgb).
15 %            img2rgb     - right image (rgb).
16
17 % Reference:
18 % Minchen Zhu, Weizhi Wang, Binghan Liu, and Jingshan Huang. "A ...
       Fast Image Stitching Algorithm via Multiple-Constraint
19 % Corner Matching",
```

```
20 %
21 % Author:
22 % Saliari Aikaterini
23 % Informatics Engineering, T.E.I. of Central Macedonia
24 %
25 % November 2014
26
27 function [ img1, img2, img1Region, img2Region,img1rgb,img2rgb ] ...
       = readImage( img1, img2 )
28
29 % Read one image of size 1280x720
30 im = imread('city.jpg');
31 im = rgb2gray(im);
32
33 img1rgb=img1;
34 img2rgb=img2;
35 img1 = rgb2gray(img1);
36 img2 = rgb2gray(img2);
37
38 % Make the resolution same for two images
39 [height,width] = size(im);
40 img1 = imresize(img1, [height width]);
41 img2 = imresize(img2, [height width]);
42 img1rgb = imresize(img1rgb, [height width]);
43 img2rgb = imresize(img2rgb, [height width]);
44
45 % One-third region of the left image
46 [¬,width] = size(img1);
47 id = fix(width/3);
48 % ima = imgLeft(:, 1:id);
49 % imb = imgLeft(:, id+1:2*id);
50 img1Region = img1(:, 2*id+1:width);
51
52 % One-third region of the right image
53 [¬,width] = size(img2);
54 id = fix(width/3);
55 img2Region = img2(:, 1:id+2);
56
```

```
57 % Change image type to double
58 img1Region = double(img1Region);
59 img2Region = double(img2Region);
60
61 end
```

```
1  % HARRIS - Harris corner detector
2  %
3  % Usage:  [cim, r, c] = harris(im, sigma, thresh, radius, displ)
4  %
5  % Arguments:
6  %            im      - image to be processed.
7  %            sigma   - standard deviation of smoothing Gaussian.
8  %                      Typical values to use might be 1-3.
9  %            thresh  - threshold (optional). Try a value ¬1000.
10 %            radius  - radius of region considered in non-maximal
11 %                      suppression (optional). Typical values to
12 %                      use might be 1-3.
13 %            disp    - optional flag (0 or 1) indicating whether
14 %                      you want to display corners overlayed on
15 %                      the original image. This can be useful
16 %                      for parameter tuning.
17 %
18 % Returns:
19 %            cim     - binary image marking corners.
20 %            r       - row coordinates of corner points.
21 %            c       - column coordinates of corner points.
22 %
23 % If thresh and radius are omitted from the argument list 'cim'
24 % is returned as a raw corner strength image and r and c
25 % are returned empty.
26
27 % Reference:
28 % C.G. Harris and M.J. Stephens. "A combined corner and edge ...
       detector",
29 % Proceedings Fourth Alvey Vision Conference, Manchester.
30 % pp 147-151, 1988.
```

```
31 %
32 % Author:
33 % Peter Kovesi
34 % Department of Computer Science & Software Engineering
35 % The University of Western Australia
36 % pk@cs.uwa.edu.au  www.cs.uwa.edu.au/~pk
37 %
38 % March 2002
39
40 function [ cim, r, c ] = harris( im, sigma, thresh, radius, ...
       displ )
41
42     narginchk(2,5);
43
44     % Derivative masks
45     dx = [-1 0 1; -1 0 1; -1 0 1];
46     dy = dx';
47
48     % 1) Compute x and y derivatives of image
49     Ix = conv2(double(im), double(dx), 'same');
50     Iy = conv2(double(im), double(dy), 'same');
51
52     % 2)Generate Gaussian filter of size 6*sigma (+/- 3sigma)
53     % and of minimum size 1x1.
54     g = fspecial('gaussian', max(1,fix(6*sigma)), sigma);
55
56     % 3)Compute the sums of the products of derivatives
57     % at each pixel
58     Ix2 = conv2(double(Ix.^2), double(g), 'same');
59     Iy2 = conv2(double(Iy.^2), double(g), 'same');
60     Ixy = conv2(double(Ix.*Iy), double(g), 'same');
61
62     % 4) Compute the response of the detector at each pixel
63     % R = det(M) - k * (trace(M) ^ 2);
64     k = 0.04;                          % Constant between 0.04-0.06
65     cim = (Ix2.*Iy2 - Ixy.^2) - k*(Ix2 + Iy2).^2;
66     R=cim;
67     % perform nonmaximal suppression and threshold
```

```
68      if nargin > 2
69
70      % Extract local maxima by performing a grey scale
71      % morphological dilation and then finding points in the
72      % corner strength image that match the dilated image
73      % and are also greater than the threshold.
74      sze = 2*radius + 1;                  % Size of mask.
75      mx = ordfilt2(cim, sze^2, ones(sze)); % Grey-scale dilate.
76
77      % 5) Threshold on value of R
78      cim = (cim==mx)&(cim>thresh);   % Find maxima.
79
80      [r,c] = find(cim);                   % Find row,col coords.
81
82      if nargin==5 && displ                % overlay corners on image
83          figure, imagesc(im), axis image, colormap(gray), hold on
84          plot(c,r,'ys'), title('corners detected')
85      end
86
87      else                % leave cim as a corner strength
88      r = []; c = [];   % image and make r and c empty.
89      end
90  end
```

```
1  % NCC - computes normalized cross correlation between images
2  %       that satisfies some conditions.[range:from -1.0 to 1.0]
3  %
4  % Usage:  simMatrix = ncc(imgLeft,imgRight,winSize,Xl,Yl,Xr,Yr)
5  %
6  % Arguments:
7  %            imgLeft    - left image (reference).
8  %            imgRight   - right image (where the correspondences
9  %                         need to be located).
10 %            winsize    - similarity window size:
11 %                         (2winlen+1)x(2winlen+1).
12 %            Xl         - x coords of left image.
13 %            Yl         - y coords of left image.
```

```
14 %             Xr         - x coords of right image.
15 %             Yr         - y coords of right image.
16 %
17 % Returns:
18 %             simMatrix - normalised cross correlation values.
19
20 % Reference:
21 % Minchen Zhu, Weizhi Wang, Binghan Liu, and Jingshan Huang. "A ...
       Fast Image Stitching Algorithm via Multiple-Constraint
22 % Corner Matching",
23 %
24 % Author:
25 % Saliari Aikaterini
26 % Informatics Engineering, T.E.I. of Central Macedonia
27 %
28 % November 2014
29
30 function [ simMatrix ] = ncc( imgLeft, imgRight, winSize, Xl, ...
       Yl, Xr, Yr )
31
32 [height, ~] = size(imgLeft);
33 lh = height/3;
34 ln = 0.75;
35 w = (winSize-1)/2;          %windowsize 7x7
36 rowLeft = size(Xl,1);
37 colRight = size(Xr,1);
38 norm = zeros(rowLeft,colRight);
39 simMatrix =  zeros(rowLeft,colRight);
40
41 % Create the similarity matrix
42 Ydiff = bsxfun(@minus, Xl, Xr');
43 Xgreat = bsxfun(@ge, Yl, Yr');
44 for i = 1:rowLeft
45     [I1] = getWindow(imgLeft, Yl(i), Xl(i), w);
46     if I1>0
47     meanLeft = mean2(imgLeft(I1));
48
49     for j = 1:colRight
```

```
50          Dl = 0.0;
51          Dr = 0.0;
52          numerator = 0.0;
53          [I2] = getWindow(imgRight, Yr(j), Xr(j), w);
54         if I2>0
55          meanRight = mean2(imgRight(I2));
56
57          % If the difference between y coordinates of
58          % two corners is no greater than height/3 and the x
59          % coordinate of the left corner is greater than
60          % or equal to that of the right one....
61          if abs(Ydiff(i,j)) < lh  && Xgreat(i,j)==1
62             % ..compute ncc(i,j)...
63              for u = -w:1:w
64                    for v = -w:1:w
65                        if (Xl(i)+u ≥1 && Xl(i)+u ≤ ...
                              size(imgLeft,1)) ...
66                                && (Yl(i)+v ≥1 && Yl(i)+v ≤ ...
                                      size(imgLeft,2)) ...
67                                && (Xr(j)+u ≥1 && Xr(j)+u ≤ ...
                                      size(imgRight,1)) ...
68                                && (Yr(j)+v ≥1 && Yr(j)+v ≤ ...
                                      size(imgRight,2))
69                            Dl = Dl+( ...
                                  imgLeft(Xl(i)+u,Yl(i)+v)-meanLeft ...
                                  )*( ...
                                  imgLeft(Xl(i)+u,Yl(i)+v)-meanLeft );
70                            Dr = Dr+( ...
                                  imgRight(Xr(j)+u,Yr(j)+v)-meanRight ...
                                  )*( ...
                                  imgRight(Xr(j)+u,Yr(j)+v)-meanRight );
71                            numerator = numerator+( ...
                                  imgLeft(Xl(i)+u,Yl(i)+v)-meanLeft ...
                                  )*( ...
                                  imgRight(Xr(j)+u,Yr(j)+v)-meanRight );
72                        end
73                    end
74              end
```

```
75              denominator = sqrt(double(Dl*Dr));
76              norm(i,j) = numerator/denominator;
77          % ..else the ncc(i,j) is equal to 0
78          else
79              norm(i,j)=0;
80          end
81        end
82      end
83      end
84  end
85  % If there is a high intensity correlation between two corners
86  % (more than a threshold) leave ncc(i,j) as it is ...
87  norm = abs(norm);
88  cc = norm > ln;
89  for i=1:size(norm,1)
90      for j=1:size(norm,2)
91          if cc(i,j)==1
92              simMatrix(i,j)=norm(i,j);
93          % ..else ncc(i,j) is equal to 0
94          else
95              simMatrix(i,j)=0;
96          end
97      end
98  end
99
100 end
```

```
1  % INITIALSET - computes an initial set of matching corner-pairs
2  %              from ncc's maximum row,column indexes operations
3  %
4  % Usage:     Lunion = initialSet(ncc,leftcoor,rightcoor)
5  %
6  % Arguments:
7  %            ncc       -  normalised cross correlation values.
8  %            leftcoor  - coordinates of corners in left image.
9  %            rightcoor - coordinates of corners in right image.
10 % Returns:
```

```
11 %             Lunion - set of matching corner-pairs
12 %                      (column 1: points from left,
13 %                      column 2: points from right).
14 %             ncc     - updated ncc matrix.
15
16 % Reference:
17 % Minchen Zhu, Weizhi Wang, Binghan Liu, and Jingshan Huang. "A ...
       Fast Image Stitching Algorithm via Multiple-Constraint
18 % Corner Matching",
19 %
20 % Author:
21 % Saliari Aikaterini
22 % Informatics Engineering, T.E.I. of Central Macedonia
23 %
24 % November 2014
25
26 function [ p2ind, p1ind, L, ncc ] = initialSet( ...
       ncc,leftcoor,rightcoor )
27
28 [nccrows,¬] = size(ncc);
29 % In each row find the column index of max ncc value
30 [rmax, jmax] = max(ncc,[],2);
31
32 % In each column find the row index of max ncc value
33 [cmax, imax] = max(ncc,[],1);
34
35 L1 = [];
36 L2 = [];
37 % Store the pair (row index,column index) of size cornerLeft
38 for c = 1:size(jmax)
39     if rmax(c) ≠ 0
40     L1 = [L1; c, jmax(c)]; % s(i,j) value is: ncc(r,jmax(r));
41     end
42 end
43
44 % Store the pair (row index,column index) of size cornerRight
45 for r = 1:size(imax,2)
46     if cmax(r) ≠ 0
```

```
47      L2 = [L2; imax(r), r]; % s(i,j) value is:  ncc(imax(c),c);
48      end
49  end
50
51  % Union of left and right pairs
52  L = union(L1 ,L2, 'rows');
53
54  % Adjust similarity to 1 if a row and a column index have each ...
        other as
55  % component in pair
56  p1ind = zeros(1,length(leftcoor));  % Arrays for storing ...
        matched indices
57  p2ind = zeros(1,length(rightcoor));
58  indcount = 0;
59  for n = 1:nccrows
60      if imax(jmax(n)) == n  % consistent both ways
61          indcount = indcount + 1;
62          p1ind(indcount) = n;
63          p2ind(indcount) = jmax(n);
64          ncc(n, jmax(n)) = 1;
65
66      end
67  end
68
69  % Trim arrays of indices of matched points
70  p1ind = p1ind(1:indcount);
71  p2ind = p2ind(1:indcount);
72
73  % Extract matched points from original arrays
74  m1 = leftcoor(:,p1ind);
75  m2 = rightcoor(:,p2ind);
76
77  end
78
79  % ss=[];
80  % if size(Lleft,1)>size(Lright,1)
81  %     same=Lleft(1:size(Lright),:);
82  %     same1=Lright;
```

```
% else
%     same=Lright(1:size(Lleft),:);
%     same1=Lleft;
% end
%
%
% for i=1:size(same,1)
%    for j=1:size(same,1)
%     if  same(i,:)==same1(j,:)
%       ss=[ss; same(i,:)];
%       ncc(same(i,1),same(i,2))=1;
%     end
%    end
% end
% Lunion = union(same,same1,'rows');
```

```
% MULTIPLECONSTRAINTS - produce a generated  matrix D based
%                       on constraints of slope length
%                       and ncc values.
%
% Usage:    [D] = multipleConstraints(imgLeftRegion, ...
    imgRightRegion, winSize, Lunion, Xleft, Yleft, Xright, ...
    Yright, simMatrix )
%
% Arguments:
%           imgLeftRegion  - last one-third region
%                            of left image.
%           imgRightRegion - first one-third region
%                            of right image.
%           winsize        - length of one side of the window.
%           Lunion         - set of matching corner-pairs
%                            (column 1: points from left,
%                            column 2: points from right)
%           Xleft          - x coords of left image.
%           Yleft          - y coords of left image.
%           Xright         - x coords of right image.
%           Yright         - y coords of right image.
```

```
20 %             simMatrix       - updated normalised cross
21 %                               correlation values.
22 %
23 % Returns:
24 %             D               - matrix of size KxK (K the total
25 %                               number of Lunion) generated by
26 %                               multiple constraints.
27 %
28
29 % Reference:
30 % Minchen Zhu, Weizhi Wang, Binghan Liu, and Jingshan Huang. "A ...
       Fast Image Stitching Algorithm via Multiple-Constraint
31 % Corner Matching",
32 %
33 % Author:
34 % Saliari Aikaterini
35 % Informatics Engineering, T.E.I. of Central Macedonia
36 %
37 % November 2014
38
39 function [ D ] = multipleConstraints( imgLeftRegion, ...
       imgRightRegion, winSize, Lunion, Xleft, Yleft, Xright, ...
       Yright, simMatrix )
40
41 % Intialize matrix
42 D = zeros(size(Lunion,1),size(Lunion,1));
43
44 % Compute length
45 leni =  sqrt( ( Lunion(:,2)-Lunion(:,1) ).^2 + ( ...
       Lunion(:,2)-Lunion(:,1) ).^2 );
46
47 % Compute slope
48 slope=( Lunion(:,2)-Lunion(:,1) )./(  Lunion(:,2)-Lunion(:,1) );
49
50 % Constraint 1
51 con1 =  bsxfun(@minus, slope, slope');
52
53 % Constraint 2
```

```
54  con2 = bsxfun(@minus, leni, leni');
55
56  for m = 1:size(Lunion,1)
57   for n=1:size(Lunion,1)
58       if lt(abs(con1(m,n)),0.5) && lt(abs(con2(m,n)),14)
59          lmnX=( Xleft(Lunion(m,1))+Xleft(Lunion(n,1)) )/2;
60          lmnY=( Yleft(Lunion(m,1))+Yleft(Lunion(n,1)) )/2;
61          rmnX=( Xright(Lunion(m,2))+Xright(Lunion(n,2)) )/2;
62          rmnY=( Yright(Lunion(m,2))+Yright(Lunion(n,2)) )/2;
63
64          % Constaraint 3
65          con3 = ncc2(imgLeftRegion, imgRightRegion, winSize, ...
                round(lmnX), round(lmnY),round(rmnX), round(rmnY));
66         if abs(con3)>0.75
67          D(m,n) = ( simMatrix(Lunion(m,1),Lunion(m,2)) + ...
                simMatrix(Lunion(n,1),Lunion(n,2)) + abs(con3) )/3 ;
68         end
69       else
70          D(m,n)=0;
71      end
72   end
73  end
74
75  end
```

```
1  % ARGMAX - search for a pair T that has the strongest correlation
2  %          with all other pairs of matrix D using the argmax
3  %          (In mathematics, arg max stands for the argument
4  %          of the maximum, that is to say, the set of points
5  %          of the given argument for which the given function
6  %          attains its maximum value)
7  %
8  % Usage:     T = argMax(D,Lunion)
9  %
10 % Arguments:
11 %             D       - matrix of size KxK generated from
12 %                       multiple constraints.
```

```
13 %             Lunion - set of matching corner-pairs.
14 % Returns:
15 %             T      - matrix with final set of matching pairs.
16
17 % Reference:
18 % Minchen Zhu, Weizhi Wang, Binghan Liu, and Jingshan Huang. "A ...
       Fast Image Stitching Algorithm via Multiple-Constraint
19 % Corner Matching",
20 %
21 % Author:
22 % Saliari Aikaterini
23 % Informatics Engineering, T.E.I. of Central Macedonia
24 %
25 % November 2014
26
27 function [ T ] = argMax( D, Lunion )
28
29 sumRowD=sum(D,2);
30 [maxSum, targ]=max(sumRowD);
31 T = [];
32
33 for i = 1:size(D,1)
34     if maxSum≠0 && D(targ,i)≠0
35         T = [T; Lunion(i,1),Lunion(i,2)];
36     end
37 end
38
39 end
```

```
1 % MATCHESWITHRANSAC - fits affine fundamental matrix using
2 %                      RANSAC and figure the matching inliers.
3 %
4 % Usage:    [H, inliers] = mathcesWithRansac(T, truecoor, Xleft, ...
       Yleft, Xright, Yright, imgLeft, imgRight, imgLeftRegion, t)
5 %
6 % Arguments:
7 %             T              - the final set of matching-corner
```

```
8  %                                pairs.
9  %            truecoor          - the difference of one-third of
10 %                                width from original
11 %                                dimension in order to figure
12 %                                the whole image
13 %            Xleft             - x coords of left image.
14 %            Yleft             - y coords of left image.
15 %            Xright            - x coords of right image.
16 %            Yright            - y coords of right image.
17 %            imgLeft           - left image.
18 %            imgRight          - right image.
19 %            imgLeftRegion     - last one-third region
20 %                                of left image.
21 %
22 % Returns:
23 %            H                 - The 3x3 fundamental matrix
24 %                                such that x2'Fx1 = 0.
25 %            inliers           - An array of indices of the
26 %                                elements of x1, x2 that were
27 %                                the inliers for the best model.
28
29 % Reference:
30 % Minchen Zhu, Weizhi Wang, Binghan Liu, and Jingshan Huang. "A ...
       Fast Image Stitching Algorithm via Multiple-Constraint
31 % Corner Matching",
32 %
33 % Author:
34 % Saliari Aikaterini
35 % Informatics Engineering, T.E.I. of Central Macedonia
36 %
37 % November 2014
38
39 function [ H,inliers ] = mathcesWithRansac( T, truecoor, Xleft, ...
       Yleft, Xright, Yright, imgLeft, imgRight, imgLeftRegion, t )
40
41 points1 = [Xleft Yleft+truecoor]';
42 p1ind = zeros(1,length(points1));
43 points2 = [Xright Yright]';
```

```
44  p2ind = zeros(1,length(points2));
45
46  for i = 1:size(T,1)
47      p1ind(i) = T(i,1)';
48      p2ind(i) = T(i,2)';
49  end
50  p1ind = p1ind(1:size(T,1));
51  p2ind = p2ind(1:size(T,1));
52  m1 = points1(:,p1ind);
53  m2 = points2(:,p2ind);
54
55  x1 = [m1(2,:); m1(1,:); ones(1,length(m1))];
56  x2 = [m2(2,:); m2(1,:); ones(1,length(m1))];
57
58  [H, inliers] = ransacfithomography_vgg(x2, x1, t);
59
60  height=size(imgLeftRegion,1);
61  match_img = zeros(height, size(imgLeft,2)+size(imgRight,2), ...
        size(imgRight,3));
62  match_img(1:size(imgLeft,1),1:size(imgLeft,2),:) = imgLeft;
63  match_img(1:size(imgRight,1),size(imgLeft,2)+1:end,:) = imgRight;
64
65  figure,imshow(uint8(match_img)), hold on
66  plot(m1(2,inliers),m1(1,inliers),'r+');
67  plot(m2(2,inliers)+size(imgLeft,2),m2(1,inliers),'g+');
68
69  for n = inliers
70      line([m2(2,n)+size(imgLeft,2) m1(2,n)], [m2(1,n) m1(1,n)])
71  end
72
73  end
```

Παράρτημα 2

Κώδικας για το GUIDE

```
function varargout = gui(varargin)


%set(0,'defaultuicontrolunits','normalized');

% GUI MATLAB code for gui.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the ...
    local
%      function named CALLBACK in GUI.M with the given input ...
    arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises ...
    the
%      existing singleton*.  Starting from the left, property ...
    value pairs are
%      applied to the GUI before gui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes ...
    property application
```

```
20 %      stop.  All inputs are passed to gui_OpeningFcn via ...
       varargin.
21 %
22 %      *See GUI Options on GUIDE's Tools menu.  Choose "GUI ...
       allows only one
23 %      instance to run (singleton)".
24 %
25 % See also: GUIDE, GUIDATA, GUIHANDLES
26
27 % Edit the above text to modify the response to help gui
28
29 % Last Modified by GUIDE v2.5 20-Feb-2015 15:26:07
30
31 % Begin initialization code - DO NOT EDIT
32 gui_Singleton = 1;
33 gui_State = struct('gui_Name',       mfilename, ...
34                    'gui_Singleton',  gui_Singleton, ...
35                    'gui_OpeningFcn', @gui_OpeningFcn, ...
36                    'gui_OutputFcn',  @gui_OutputFcn, ...
37                    'gui_LayoutFcn',  [] , ...
38                    'gui_Callback',   []);
39 if nargin && ischar(varargin{1})
40     gui_State.gui_Callback = str2func(varargin{1});
41 end
42
43 if nargout
44     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
45 else
46     gui_mainfcn(gui_State, varargin{:});
47 end
48 % End initialization code - DO NOT EDIT
49
50
51 % --- Executes just before gui is made visible.
52 function gui_OpeningFcn(hObject, eventdata, handles, varargin)
53 % This function has no output args, see OutputFcn.
54 % hObject    handle to figure
```

```
55 % eventdata  reserved - to be defined in a future version of ...
       MATLAB
56 % handles    structure with handles and user data (see GUIDATA)
57 % varargin   command line arguments to gui (see VARARGIN)
58
59 % Choose default command line output for gui
60 handles.output = hObject;
61 addpath(genpath('/SALIARI_AIKATERINI_1975_STITCH/matlabCode'))
62
63 % Disable some panels and button till another action is complete
64 set(findall(handles.uipStep1, '-property', 'enable'), 'enable', ...
       'off');
65 set(findall(handles.uipStep2, '-property', 'enable'), 'enable', ...
       'off');
66 set(findall(handles.uipStep3, '-property', 'enable'), 'enable', ...
       'off');
67 set(findall(handles.uipStep4, '-property', 'enable'), 'enable', ...
       'off');
68 set(findall(handles.uipStep5, '-property', 'enable'), 'enable', ...
       'off');
69 set(findall(handles.uipStep6, '-property', 'enable'), 'enable', ...
       'off');
70 handles.flagImg1 = false;
71 handles.flagImg2 = false;
72 handles.flagIn = false;
73 handles.flagwinsize = false;
74 set(handles.eWinSize,'Enable','off');
75 % Update handles structure
76 guidata(hObject, handles);
77
78 % UIWAIT makes gui wait for user response (see UIRESUME)
79  %uiwait(hObject);
80
81
82 % --- Outputs from this function are returned to the command ...
       line.
83 function varargout = gui_OutputFcn(hObject, ¬, handles)
```

```
84 % varargout  cell array for returning output args (see ...
       VARARGOUT);
85 % hObject    handle to figure
86 % eventdata  reserved - to be defined in a future version of ...
       MATLAB
87 % handles    structure with handles and user data (see GUIDATA)
88
89 % Get default command line output from handles structure
90 varargout{1} = get(handles.output);
91
92
93 % --- Executes on button press in bLeft.
94 function bLeft_Callback(hObject, ¬, handles)
95
96 % Select image from computer
97 [fileName, folder] = uigetfile('*.*', 'Specify an image file');
98 if ¬ischar(fileName)
99     errordlg('Error!','No file selected');
100    return;
101 else
102
103 % Create the full file name
104 fullImageFileName = fullfile(folder, fileName);
105 img1 = imread(fullImageFileName );
106
107 % Create a handle to img1 for further use
108  handles.img1 = img1;
109
110 % Show image in axes
111 axes(handles.axesImg1);
112 imshow(img1);
113
114 % Check if condition is true when bOk is pressed
115 handles.flagImg1=true;
116
117 % Update data
118 guidata(hObject, handles);
119
```

```
120 end
121
122
123 % --- Executes on button press in bRight.
124 function bRight_Callback(hObject, ¬, handles)
125
126 % Select image from computer
127 [fileName, folder] = uigetfile('*.*', 'Specify an image file');
128 if ¬ischar(fileName)
129     errordlg('Error!','No file selected');
130    return;
131 else
132
133 % Create the full file name
134 fullImageFileName = fullfile(folder, fileName);
135 img2 = imread(fullImageFileName );
136
137 % Create a handle to img2 for further use
138 handles.img2 = img2;
139
140 % Show image in axes
141 axes(handles.axesImg2);
142 imshow(img2);
143
144 % Check if condition is true to continue
145 handles.flagImg2=true;
146
147 % Update data
148 guidata(hObject, handles);
149
150 end
151
152
153 % --- Executes on button press in bReset.
154 function bReset_Callback(hObject, eventdata, handles)
155 % hObject    handle to bReset (see GCBO)
156 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
```

```
157 % handles    structure with handles and user data (see GUIDATA)
158 cla(handles.axesImg1,'reset');
159 cla(handles.axesImg2,'reset');
160 close(gcbf)
161 clc
162 evalin( 'base', 'clearvars *' )
163 close all
164 gui
165
166
167 function eCommand_Callback(hObject, eventdata, handles)
168 % hObject    handle to eCommand (see GCBO)
169 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
170 % handles    structure with handles and user data (see GUIDATA)
171
172 % Hints: get(hObject,'String') returns contents of eCommand as ...
      text
173 %        str2double(get(hObject,'String')) returns contents of ...
      eCommand as a double
174
175
176 % --- Executes during object creation, after setting all ...
      properties.
177 function eCommand_CreateFcn(hObject, eventdata, handles)
178 % hObject    handle to eCommand (see GCBO)
179 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
180 % handles    empty - handles not created until after all ...
      CreateFcns called
181
182 % Hint: edit controls usually have a white background on Windows.
183 %       See ISPC and COMPUTER.
184 if ispc && isequal(get(hObject,'BackgroundColor'), ...
      get(0,'defaultUicontrolBackgroundColor'))
185     set(hObject,'BackgroundColor','white');
186 end
187
```

```
188
189 % --- Executes on button press in rbAutoStitch.
190 function rbAutoStitch_Callback(hObject, eventdata, handles)
191 % hObject    handle to rbAutoStitch (see GCBO)
192 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
193 % handles    structure with handles and user data (see GUIDATA)
194
195 % Hint: get(hObject,'Value') returns toggle state of rbAutoStitch
196
197
198 % --- Executes on button press in rbStepStitch.
199 function rbStepStitch_Callback(hObject, eventdata, handles)
200 % hObject    handle to rbStepStitch (see GCBO)
201 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
202 % handles    structure with handles and user data (see GUIDATA)
203
204 % Hint: get(hObject,'Value') returns toggle state of rbStepStitch
205
206
207 % --- Executes on button press in bStitch.
208 function bStitch_Callback(hObject, eventdata, handles)
209 % hObject    handle to bStitch (see GCBO)
210 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
211 % handles    structure with handles and user data (see GUIDATA)
212 tic
213 msg1='Please Wait...The RANSAC algorithm is being calculated';
214 set(handles.eMessages,'string',msg1);
215 T = handles.T;
216 t = .001;
217 truecoor = handles.truecoor;
218 Xleft = handles.Xleft;
219 Yleft = handles.Yleft;
220 Xright = handles.Xright;
221 Yright = handles.Yright;
222 imgLeft = handles.imgLeft;
```

```
223 imgRight = handles.imgRight;
224 img1 = handles.img1;
225 img2 = handles.img2;
226 imgLeftRegion = handles.imgLeftRegion;
227
228 [H,inliers] = mathcesWithRansac( T, truecoor, Xleft, Yleft, ...
        Xright, Yright, imgLeft, imgRight, imgLeftRegion, t );
229 bbox = [-600 3000 -600 1500]; % image space for mosaic
230
231 Im2w = vgg_warp_H(double(imgLeft), eye(3), 'linear', bbox); % ...
        warp image 1 to mosaic image
232 Im1w = vgg_warp_H(double(imgRight), H, 'linear', bbox);
233 figure; imshow(uint8(max(Im1w,Im2w)));
234 %[file,path] = uiputfile('*.jpg','Save Image');
235
236 Im2w = vgg_warp_H(double(img1), eye(3), 'linear', bbox); % warp ...
        image 1 to mosaic image
237 Im1w = vgg_warp_H(double(img2), H, 'linear', bbox);
238 figure; imshow(uint8(max(Im1w,Im2w)));
239 assignin ('base','Inliers',inliers)
240 toci = toc;
241 set(handles.eMessages,'string',sprintf('Image Stitching is ...
        complete. \n Elapsed time is %g', toci));
242 % Update data
243 guidata(hObject, handles);
244
245
246 % --- Executes on button press in bFinalPairs.
247 function bFinalPairs_Callback(hObject, eventdata, handles)
248 % hObject    handle to bFinalPairs (see GCBO)
249 % eventdata  reserved - to be defined in a future version of ...
        MATLAB
250 % handles    structure with handles and user data (see GUIDATA)
251 tic
252 D = handles.D;
253 Lunion = handles.Lunion;
254 truecoor = handles.truecoor;
255 Xleft = handles.Xleft;
```

```
256  Yleft = handles.Yleft;
257  Xright = handles.Xright;
258  Yright = handles.Yright;
259  imgLeft = handles.imgLeft;
260  imgRight = handles.imgRight;
261  imgLeftRegion = handles.imgLeftRegion;
262
263  sumRowD=sum(D,2);
264  [maxSum,targ]=max(sumRowD);
265  T = [];
266  ii=[];
267  for i = 1:size(D,1)
268      if maxSum≠0 && D(targ,i)≠0
269          T = [T; Lunion(i,1),Lunion(i,2)];
270          ii=[ii;i];
271      end
272  end
273  toci = toc;
274  set(handles.eMessages,'string',sprintf('Step5 is complete! \n ...
         The final reduced set of matching-corner pairs is shown in ...
         the figure. \n Elapsed time is %g', toci));
275  assignin ('base','Final_Reduced_Set',T)
276  handles.T = T;
277  set(findall(handles.uipStep6, '-property', 'enable'), 'enable', ...
         'on');
278
279  setappdata(0,'final_set',T)
280  final_set
281
282  mathces( T, truecoor, Xleft, Yleft, Xright, Yright, imgLeft, ...
         imgRight, imgLeftRegion )
283  % Update data
284  guidata(hObject, handles);
285
286  % --- Executes on button press in bConstraints.
287  function bConstraints_Callback(hObject, eventdata, handles)
288  % hObject    handle to bConstraints (see GCBO)
```

```
289 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
290 % handles    structure with handles and user data (see GUIDATA)
291 tic
292 msg1='Please Wait...The new Similarity Matrix is being ...
      calculated';
293 set(handles.eMessages,'string',msg1);
294 imgLeftRegion = handles.imgLeftRegion;
295 imgRightRegion = handles.imgRightRegion;
296
297
298 similarity = handles.similarity;
299 Xleft = handles.Xleft;
300 Yleft = handles.Yleft;
301 Xright = handles.Xright;
302 Yright = handles.Yright;
303
304 winSize = handles.winsize;
305 Lunion = handles.Lunion;
306 [D] = multipleConstraints(imgLeftRegion, imgRightRegion, ...
      winSize, Lunion, Xleft, Yleft, Xright, Yright, similarity );
307 toci = toc;
308 set(handles.eMessages,'string',sprintf('Step4 is complete! \n ...
      The updated similarity matrix values are shown in the ...
      figure. \n Elapsed time is %g', toci));
309 assignin ('base','Updated_Similarity_Matrix',D)
310 handles.D = D;
311 set(findall(handles.uipStep5, '-property', 'enable'), 'enable', ...
      'on');
312
313 setappdata(0,'Multiple_Constraint_Matrix',D)
314 Multiple_Constraint_Matrix
315 % Update data
316 guidata(hObject, handles);
317
318
319 % --- Executes on button press in bInitialSet.
320 function bInitialSet_Callback(hObject, eventdata, handles)
```

```
321 % hObject    handle to bInitialSet (see GCBO)
322 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
323 % handles    structure with handles and user data (see GUIDATA)
324 tic
325 similarity = handles.similarity;
326 Xleft = handles.Xleft;
327 Yleft = handles.Yleft;
328 Xright = handles.Xright;
329 Yright = handles.Yright;
330
331
332 [L1,L2,Lunion,similarity] = initialSet( ...
      similarity,[Xleft';Yleft'],[Xright';Yright'] );
333
334 handles.Lunion = Lunion;
335 handles.similarity = similarity;
336
337 toci = toc;
338 set(handles.eMessages,'string',sprintf('Step3 is complete!\n ...
      The initial set of matching-corner pairs is appear in the ...
      figure. \n Elapsed time is %g', toci));
339 assignin ('base','InitialSet',Lunion)
340 assignin ('base','L1',L1)
341 assignin ('base','L2',L2)
342 set(findall(handles.uipStep4, '-property', 'enable'), 'enable', ...
      'on');
343
344 setappdata(0,'Initial',Lunion)
345 Initial
346
347 % Update data
348 guidata(hObject, handles);
349 %--------------------------------------------------------------------
350
351 % --- Executes on button press in bNcc.
352 function bNcc_Callback(hObject, eventdata, handles)
353 % hObject    handle to bNcc (see GCBO)
```

```
354 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
355 % handles    structure with handles and user data (see GUIDATA)
356 tic
357 msg1='Please Wait... Similarity Matrix is being calculated';
358 set(handles.eMessages,'string',msg1);
359 imgLeftRegion = handles.imgLeftRegion;
360 imgRightRegion = handles.imgRightRegion;
361
362 Xleft = handles.Xleft;
363 Xright = handles.Xright;
364 Yleft = handles.Yleft;
365 Yright = handles.Yright;
366 winSize = handles.winsize;
367 [similarity] = ncc(imgLeftRegion, imgRightRegion, winSize, ...
      Xleft, Yleft, Xright, Yright);
368 handles.similarity = similarity;
369 assignin ('base','ncc',similarity)
370 toci = toc;
371 set(handles.eMessages,'string',sprintf('Step2 is complete! \n ...
      Check the ncc matrix values in the figure. \n Elapsed time ...
      is %g', toci));
372 set(findall(handles.uipStep3, '-property', 'enable'), 'enable', ...
      'on');
373 %set(handles.uitable1,'data',similarity);
374 setappdata(0,'Similarity_Matrix',similarity)
375 Similarity_Matrix
376
377 % Update data
378 guidata(hObject, handles);
379
380
381 % --- Executes on button press in bCorner.
382 function bCorner_Callback(hObject, eventdata, handles)
383 % hObject    handle to bCorner (see GCBO)
384 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
385 % handles    structure with handles and user data (see GUIDATA)
```

```
386 tic
387 % On button click compute the corners
388 img1 = handles.img1;
389 img2 = handles.img2;
390 imgRight = rgb2gray(img2);
391 imgLeft = rgb2gray(img1);
392 %addpath(genpath('C:\Users\SpiMeLo\Documents\MATLAB\s'));
393 addpath(genpath('/SALIARI_AIKATERINI_1975_STITCH/matlabCode'))
394 im = imread('city.jpg');
395 im = rgb2gray(im);
396 sigma = 1;
397 radius = 1;
398 displ = 0;
399 % Make the resolution same for two images
400 [height,width] = size(im);
401 imgLeft = imresize(imgLeft, [height width]);
402 imgRight = imresize(imgRight, [height width]);
403
404 img1 = imresize(img1, [height width]);
405 img2 = imresize(img2, [height width]);
406 handles.img1 = img1;
407 handles.img2 = img2;
408 % One-third region of the left image
409 [¬,width] = size(imgLeft);
410 id = fix(width/3);
411 % ima = imgLeft(:, 1:id);
412 % imb = imgLeft(:, id+1:2*id);
413 img1Region = imgLeft(:, 2*id+1:width);
414
415 % One-third region of the right image
416 [¬,width] = size(imgRight);
417 id = fix(width/3);
418 img2Region = imgRight(:, 1:id+2);
419
420 % Change image type to double
421 imgLeftRegion = double(img1Region);
422 imgRightRegion = double(img2Region);
423
```

```
424 thresh = handles.thresh;
425 [¬, Xleft, Yleft] = harris(imgLeftRegion, sigma, thresh, ...
        radius, displ);
426 [¬, Xright, Yright] = harris(imgRightRegion, sigma, thresh, ...
        radius, displ);
427 set(findall(handles.uipStep2, '-property', 'enable'), 'enable', ...
        'on');
428 handles.imgLeftRegion = imgLeftRegion;
429 handles.imgRightRegion = imgRightRegion;
430 handles.Xleft = Xleft;
431 handles.Xright = Xright;
432 handles.Yleft = Yleft;
433 handles.Yright = Yright;
434 handles.imgLeft = imgLeft;
435 handles.imgRight = imgRight;
436
437 figure('name','Left Image Corners');
438 truecoor = size(imgLeft,2)-size(imgLeftRegion,2);
439 handles.truecoor = truecoor;
440 impixelinfo(imagesc(uint8(imgLeft))),axis image, ...
        colormap(gray), hold on, plot(Yleft+truecoor,Xleft,'g.');
441 figure('name','Right Image Corners');
442 impixelinfo(imagesc(uint8(imgRight))),axis image, ...
        colormap(gray), hold on, plot(Yright,Xright,'g.');
443 toci = toc;
444 set(handles.eMessages,'string',sprintf('Step1 is complete! \n ...
        Corners are shown in figures. \n Elapsed time is %g', toci));
445 assignin ('base','right',[Xright,Yright])
446 setappdata(0,'Corners',[Xleft,Yleft])
447 Corners
448
449 % Update data
450 guidata(hObject, handles);
451
452
453 % --- Executes on button press in bOk.
454 function bOk_Callback(hObject, eventdata, handles)
455 % hObject    handle to bOk (see GCBO)
```

```
456 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
457 % handles    structure with handles and user data (see GUIDATA)
458
459 if handles.flagImg1==true && handles.flagImg2==true
460     set(handles.bLeft,'Enable','off');
461     set(handles.bRight,'Enable','off');
462 else
463     errordlg('Invalid or  Missing Input.Put the arrow to each ...
          field for a hint','Error','modal')
464     uicontrol(hObject)
465     return
466 end
467 thresh = str2double(get(handles.eSimThreshold,'string'));
468 if isnan(thresh)
469     errordlg('Invalid or Missing Input.Put the arrow to each ...
          field for a hint','Error','modal')
470     uicontrol(hObject)
471     return
472 else
473     if thresh < 100000 %|| ln > 1
474         errordlg('Invalid or Missing Input.Put the arrow to ...
              each field for a hint','Error','modal')
475         uicontrol(hObject)
476         return
477     else
478         handles.flagln = true;
479     end
480 end
481 handles.thresh = thresh;
482 % winsize = str2double(get(handles.eWinSize,'string'));
483 winsize = 7;
484 handles.winsize = winsize;
485
486 % if isnan(winsize)
487 %     errordlg('Invalid or Missing Input.Put the arrow to each ...
      field for a hint','Error','modal')
488 %     uicontrol(hObject)
```

```
489 %     return
490 % else
491 %     if mod(winsize,2) == 0
492 %         errordlg('Invalid or Missing Input.Put the arrow to ...
        each field for a hint','Error','modal')
493 %         uicontrol(hObject)
494 %         return
495 %     else
496 %         handles.flagwinsize=true;
497 %     end
498 % end
499 if handles.flagln==true %&& handles.flagwinsize==true
500     set(handles.eSimThreshold,'Enable','off');
501     %set(handles.eWinSize,'Enable','off');
502 else
503     errordlg('Invalid or Missing Input.Put the arrow to each ...
          field for a hint','Error','modal')
504     uicontrol(hObject)
505     return
506 end
507 if (get(handles.rbAutoStitch,'Value') == 0) %step stitch is ...
      selected
508 % Panel with Step1 is now enabled
509 set(findall(handles.uipStep1, '-property', 'enable'), 'enable', ...
      'on');
510 else
511 % diary('out')
512 %  diary on;
513 img1 = handles.img1;
514 img2 = handles.img2;
515 Main(img1, img2,thresh);
516 %  diary off;
517 end
518 % Update data
519 guidata(hObject, handles);
520
521
522 function eSimThreshold_Callback(hObject, eventdata, handles)
```

```
523 % hObject    handle to eSimThreshold (see GCBO)
524 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
525 % handles    structure with handles and user data (see GUIDATA)
526
527 % Hints: get(hObject,'String') returns contents of ...
      eSimThreshold as text
528 %        str2double(get(hObject,'String')) returns contents of ...
      eSimThreshold as a double
529 % ln = str2double(get(handles.eSimThreshold,'string'));
530 % if isnan(ln)
531 %   errordlg('You must enter a numeric value between ...
      0-1','Invalid Input','modal')
532 %   uicontrol(hObject)
533 %   return
534 % else
535 %     if ln < 0 || ln > 1
536 %         errordlg('You must enter a numeric value between ...
      0-1','Invalid Input','modal')
537 %         uicontrol(hObject)
538 %         return
539 %     else
540 %         handles.flagln = true;
541 %         % Update data
542 %         guidata(hObject, handles);
543 %     end
544 % end
545
546 % --- Executes during object creation, after setting all ...
      properties.
547 function eSimThreshold_CreateFcn(hObject, eventdata, handles)
548 % hObject    handle to eSimThreshold (see GCBO)
549 % eventdata  reserved - to be defined in a future version of ...
      MATLAB
550 % handles    empty - handles not created until after all ...
      CreateFcns called
551
552 % Hint: edit controls usually have a white background on Windows.
```

```
553 %        See ISPC and COMPUTER.
554 if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
555     set(hObject,'BackgroundColor','white');
556 end
557
558
559
560 function eWinSize_Callback(hObject, eventdata, handles)
561 % hObject    handle to eWinSize (see GCBO)
562 % eventdata  reserved - to be defined in a future version of ...
        MATLAB
563 % handles    structure with handles and user data (see GUIDATA)
564
565 % Hints: get(hObject,'String') returns contents of eWinSize as ...
        text
566 %        str2double(get(hObject,'String')) returns contents of ...
        eWinSize as a double
567 % winsize = str2double(get(hObject,'string'));
568 % if isnan(winsize)
569 %   errordlg('You must enter a numeric value','Invalid ...
        Input','modal')
570 %   uicontrol(hObject)
571 %   return
572 % else
573 %     if mod(winsize,2) == 0
574 %         errordlg('You must enter an odd value','Invalid ...
        Input','modal')
575 %         uicontrol(hObject)
576 %         return
577 %     else
578 %         handles.flagwinsize=true;
579 %         % Update data
580 %         guidata(hObject, handles);
581 %     end
582 % end
583
584
```

```
585 % --- Executes during object creation, after setting all ...
        properties.
586 function eWinSize_CreateFcn(hObject, eventdata, handles)
587 % hObject    handle to eWinSize (see GCBO)
588 % eventdata  reserved - to be defined in a future version of ...
        MATLAB
589 % handles    empty - handles not created until after all ...
        CreateFcns called
590
591
592
593 % Hint: edit controls usually have a white background on Windows.
594 %       See ISPC and COMPUTER.
595 if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
596     set(hObject,'BackgroundColor','white');
597 end
598
599
600 % --- Executes when selected object is changed in buttonGroup.
601 function buttonGroup_SelectionChangeFcn(hObject, eventdata, ...
        handles)
602 % hObject    handle to the selected object in buttonGroup
603 % eventdata  structure with the following fields (see ...
        UIBUTTONGROUP)
604 %   EventName: string 'SelectionChanged' (read only)
605 %   OldValue: handle of the previously selected object or empty ...
        if none was selected
606 %   NewValue: handle of the currently selected object
607 % handles    structure with handles and user data (see GUIDATA)
608 get(eventdata.NewValue, 'Tag');
609
610
611 % --- Executes on slider movement.
612 function slider3_Callback(hObject, eventdata, handles)
613 % hObject    handle to slider3 (see GCBO)
614 % eventdata  reserved - to be defined in a future version of ...
        MATLAB
```

```
615 % handles    structure with handles and user data (see GUIDATA)
616
617 % Hints: get(hObject,'Value') returns position of slider
618 %        get(hObject,'Min') and get(hObject,'Max') to determine ...
        range of slider
619
620
621
622 % --- Executes during object creation, after setting all ...
        properties.
623 function slider3_CreateFcn(hObject, eventdata, handles)
624 % hObject    handle to slider3 (see GCBO)
625 % eventdata  reserved - to be defined in a future version of ...
        MATLAB
626 % handles    empty - handles not created until after all ...
        CreateFcns called
627
628 % Hint: slider controls usually have a light gray background.
629 if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
630     set(hObject,'BackgroundColor',[.9 .9 .9]);
631 end
632
633
634 % --- Executes on slider movement.
635 function slider4_Callback(hObject, eventdata, handles)
636 % hObject    handle to slider4 (see GCBO)
637 % eventdata  reserved - to be defined in a future version of ...
        MATLAB
638 % handles    structure with handles and user data (see GUIDATA)
639
640 % Hints: get(hObject,'Value') returns position of slider
641 %        get(hObject,'Min') and get(hObject,'Max') to determine ...
        range of slider
642
643
644
```

```
645 % --- Executes during object creation, after setting all ...
        properties.
646 function slider4_CreateFcn(hObject, eventdata, handles)
647 % hObject    handle to slider4 (see GCBO)
648 % eventdata  reserved - to be defined in a future version of ...
        MATLAB
649 % handles    empty - handles not created until after all ...
        CreateFcns called
650
651 % Hint: slider controls usually have a light gray background.
652 if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
653     set(hObject,'BackgroundColor',[.9 .9 .9]);
654 end
655
656
657
658 function eMessages_Callback(hObject, eventdata, handles)
659 % hObject    handle to eMessages (see GCBO)
660 % eventdata  reserved - to be defined in a future version of ...
        MATLAB
661 % handles    structure with handles and user data (see GUIDATA)
662
663 % Hints: get(hObject,'String') returns contents of eMessages as ...
        text
664 %        str2double(get(hObject,'String')) returns contents of ...
        eMessages as a double
665
666
667 % --- Executes during object creation, after setting all ...
        properties.
668 function eMessages_CreateFcn(hObject, eventdata, handles)
669 % hObject    handle to eMessages (see GCBO)
670 % eventdata  reserved - to be defined in a future version of ...
        MATLAB
671 % handles    empty - handles not created until after all ...
        CreateFcns called
672
```

```
673 % Hint: edit controls usually have a white background on Windows.
674 %       See ISPC and COMPUTER.
675 if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
676     set(hObject,'BackgroundColor','white');
677 end
678
679
680 % --- Executes on button press in bClose.
681 function bClose_Callback(hObject, eventdata, handles)
682 % hObject    handle to bClose (see GCBO)
683 % eventdata  reserved - to be defined in a future version of ...
        MATLAB
684 % handles    structure with handles and user data (see GUIDATA)
685 close(handles.figure1)
686 close(gcbf)
687 clc
688 evalin( 'base', 'clearvars *' )
689 close all
```

Βιβλιογραφία

[1] Minchen Zhu, Weizhi Wang, Binghan Liu and Jingshan Huang, A Fast Image Stitching Algorithm via Multiple-Constraint Corner Matching, 2013

[2] UNICAMP, Additional Material Harris Detector. Διαθέσιμο στη διεύθυνση: http://www.ic.unicamp.br/rocha/teaching/2013s1/mc851/aulas/additional-material-harris-detector.pdf [πρόσβαση Νοέμβριος 2014]

[3] Robert Collins, Harris Corner Detector. Διαθέσιμο στη διεύθυνση: http://www.cse.psu.edu / rcollins/CSE486/lecture06.pdf [πρόσβαση Ιούλιος 2015]

[4] Μιχαήλ Γκιννής, Διπλωματική Εργασία. Διαθέσιμο στη διεύθυνση: http://nemertes.lis.upatras.gr/jspui/bitst ream/10889/6013/1/ΔιπλωματικήΕργασία - GKINNIS.pdf [πρόσβαση Ιούλιος 2015]

[5] Wikipedia, Kronecker product. Διαθέσιμο στη διεύθυνση: http://en.wikipedia.org /wiki/Kronecker_product [πρόσβαση Ιούλιος 2015]

[6] Wikipedia, Cross-correlation. Διαθέσιμο στη διεύθυνση: http://en.wikipedia.org /wiki/Cross-correlation [πρόσβαση Ιούλιος 2015]

[7] Δημήτριος Κωνσταντινίδης, Διπλωματική Εργασία. Διαθέσιμο στη διεύθυνση: http://vivliothmmy.ee.auth.gr/32 5/1/diploma.pdf [πρόσβαση Ιούλιος 2015]

[8] Institute of Geodesy and Photogrammetry, Programming normalized cross-correlation. Διαθέσιμο στη διεύθυνση: http://www.igp.ethz.ch /photogrammetry/education/lehrveranstaltungen/photogrammetryandmachinevision/ courematerial/uebung2a.pdf [πρόσβαση Νοέμβριος 2014]

[9] Martin A. Fischer, Robert C. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applica tions to Image Analysis and Automated Cartography, 1981

[10] Wikipedia, Fundamental matrix. Διαθέσιμο στη διεύθυνση: http://en.wiki pedia.org/wiki/Fundamental _matrix _28computer_vision29 [πρόσβαση Ιούλιος 2015]

[11] D. Capel, An Effective Bail-out Test for RANSAC Consensus Scoring, 2005

[12] O. Chum, J. Matas, J. Kittler, Locally optimized RANSAC, 2003

[13] Mathworks, Features. Διαθέσιμο στη διεύθυνση: http://www.mathworks.com /products/matlab/features.html [πρόσβαση Ιούλιος 2015]

[14] Κατσάνος Ευάγγελος, Διδακτικές Σημειώσεις Matlab. Διαθέσιμο στη διεύθυνση: http://edusoft.civil.auth .gr/TE4800/Matlab Notes and Codes/Matlab Notes.pdf [πρόσβαση Ιούλιος 2015]

[15] Mathworks, Image Processing Toolbox. Διαθέσιμο στη διεύθυνση: http://www.mathworks.com/products/imag e/index-b.html [πρόσβαση Ιούλιος 2015]

[16] Mathworks, Creating Graphical User Interfaces. Διαθέσιμο στη διεύθυνση: http://www.mathworks.com/help /pdf_doc/matlab/buildgui.pdf [πρόσβαση Ιούλιος 2015]

[17] UWA, Peter Kovesi, Harris Corner Detector. Διαθέσιμο στη διεύθυνση: http://www.csse.uwa.edu.au/ pk/re search/matlabfns/Spatial/harris.m [πρόσβαση Ιούλιος 2015]

[18] Thomas Denewiler. Διαθέσιμο στη διεύθυνση: http://denewiler.us/trac/cse252b /browser/hw2?rev=16 [πρόσβαση Νοέμβριος 2014]

[19] UWA, Peter Kovesi, Correlation. Διαθέσιμο στη διεύθυνση: http://www.csse.uwa.edu.au / pk/research/matlabfns /Match/matchbycorrelation.m [πρόσβαση Ιούλιος 2015]

[20] UWA, Peter Kovesi, Ransac. Διαθέσιμο στη διεύθυνση: http://www.csse.uwa.edu .au/ pk/research/matlabfns/Robust/ransacfithomography.m [πρόσβαση Ιούλιος 2015]

[21] ENS, Warp Function. Διαθέσιμο στη διεύθυνση: http://www.di.ens.fr/willow /events/cvml2013/scholarship/application/vgg_warp_H.m [πρόσβαση Ιούλιος 2015]

[22] R.I. Hartley, In Defense of the Eight-Point Algorithm, 1997

Η εργασία αυτή στοιχειοθετήθηκε με το πρόγραμμα XeLaTeX. Οι γραμματοσειρές που χρησιμοποιήθηκαν είναι οι Times New Roman και `Courier New`.