Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Ηλεκτρονικής και Υπολογιστών

Εργασία στο Ψηφιακή Επεξεργασία Εικόνας

Hough, Harris, Rot και αποκοπή εικόνων

Εργασία 2

Όνομα: Στέφανος Γανωτάκης
ΑΕΜ: 7664
Email: sganotak@auth.gr

ΙΟΥΝΙΟΣ 2020

Περιεχόμενα

1 Εισαγωγή

Στην εργασία αυτή αναπτύχθηκαν προγράμματα σε matlab που υλοποιούν γνωστούς αλγορίθμους της Ψηφιακής Επεξεργασίας Εικόνας. Αρχικά έγινε μια υλοποίηση του μετασχηματισμού Hough για τον εντοπισμό ευθειών σε μια δυαδική εικόνα. Ο δεύτερος αλγόριθμος που υλοποιήθηκε είναι ο Harris Corner Detector και τέλος αναπτύχθηκε μια ρουτίνα περιστροφής εικόνων. Οι τρεις παραπάνω βασικοί αλγόριθμοι σε συνδυασμό με κάποια επιπρόσθετα βοηθητικά προγράμματα που δημιουργήθηκαν για το σκοπό της εργασίας, χρησιμοποιήθηκαν για την υλοποίηση ενός τελικού προγράμματος για την αποκοπή και ψηφιοποίηση φωτογραφιών ενός scanner.

1.1 Αντικείμενο και Δομή Εργασίας

Η εργασία έχει δομηθεί σε δύο ενότητες. Η πρώτη ενότητα περιλαμβάνει τρία συνολικά παραδοτέα που αφορούν τους βασικούς αλγόριθμούς που ζητήθηκε να αναπτυχθούν. Η δεύτερη ενότητα αφορά την υλοποίηση του scanner

- Στο πρώτο παραδοτέο της πρώτης ενότητας υλοποιείται ο μετασχηματισμός Hough. Το συνοδευτικό demo script που έχει αναπτυχθεί κάνει επίδειξη των ευθειών που εντοπίζει ο αλγόριθμος σε μια από τις εικόνες που δόθηκαν στην εκφώνηση
- Το δεύτερο παραδοτέο της πρώτης ενότητας υλοποιεί τον Harris Corner Detector. Στο συνοδευτικό demo script παρουσιάζονται οι γωνίες της εικόνας που εντοπίζονται από τον αλγόριθμο
- Το τρίτο παραδοτέο της πρώτης ενότητας, αφορά την ανάπτυξη ενός αλγορίθμου περιστροφής μιας εικόνας. Στο demo script γίνεται επίδειξη δυο ανεξάρτητων περιστροφών της εικόνας.
- Στην δεύτερη ενότητα της εργασίας παρουσιάζεται η υλοποίηση του αλγορίθμου myLazyScanner. Πιο συγκεκριμένα, γίνεται επεξήγηση της λογικής του αλγόριθμού βήμα προς βήμα, αναλύονται όλες οι επιπρόσθετες βοηθητικές συναρτήσεις που δημιουργήθηκαν και τέλος γίνεται παρουσίαση των αποτελεσμάτων του αλγορίθμου στις 5 εικόνες που δόθηκαν για πειραματισμό

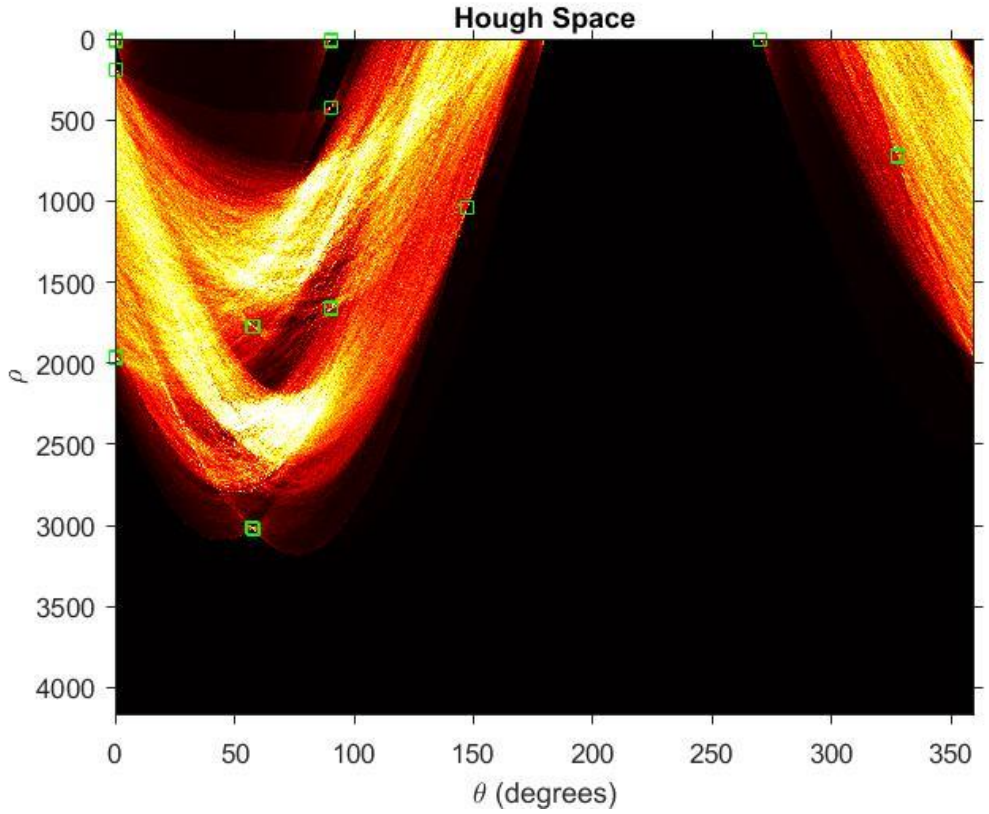
2 Ενότητα Α

2.1 1ο Παραδοτέο: Μετασχηματισμός Hough

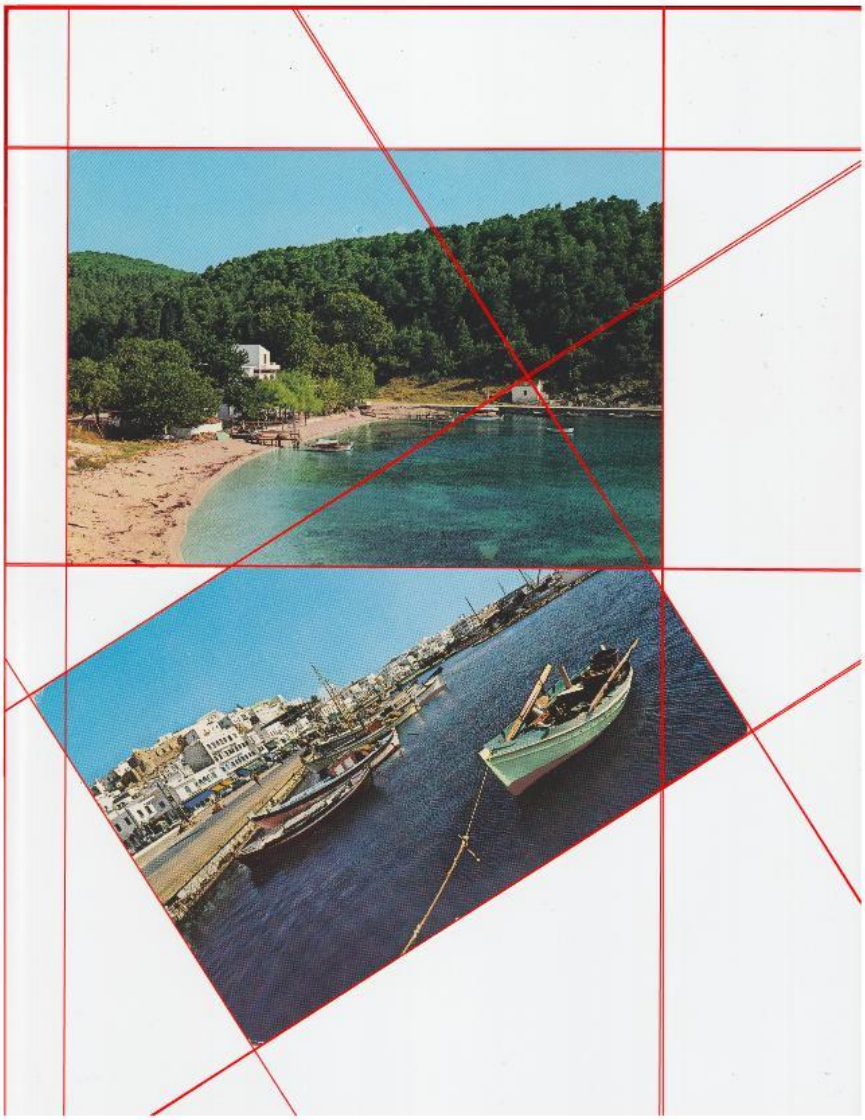
Η ρουτίνα [H, L, res] =myHoughTransform(img_binary , Drho , Dtheta , n) υλοποιήθηκε όπως ακριβώς ζητήθηκε από την εκφώνηση. Για την συγκεκριμένη υλοποιήση του αλγορίθμου αποφασίστηκε να χρησιμοποιηθούν τα διαστήματα *[0,ρmax]* για το *ρ* και *[0,2π]* για το *ϑ.* Τα βήματα διακριτοποίησης είναι τα Drho,Dtheta που δίνονται ως όρισμα στην συνάρτηση. Για την ταχύτερη εκτέλεση του αλγορίθμου έχει γίνει κατωφλίωση των τιμών της εικόνας έτσι ώστε, να εξετάζονται μόνο τα λευκά pixels που επιστρέφει ο edge detector. Αφού γίνει αρχικοποίηση του αθροιστή H, για κάθε τιμή στο διάστημα των *ϑ,* πραγματοποιείται ο υπολογισμός *ρ=xcosϑ+ysinϑ*. Τα ζευγάρια *[ρ,ϑ]* , δίνονται ως όρισμα στον αθροιστή Hough, όπου αυξάνεται η τιμή του ανάλογου bin . Τέλος εντοπίζονται τα n τοπικά μέγιστα του πίνακα H με φθίνουσα σειρά, επιστρέφοντας τα ζεύγη *[ρ,ϑ]* που αντιπροσωπεύουν τις n κυρίαρχες ευθείες που εντόπισε ο αλγόριθμος και αποθηκεύονται στον nx2 πίνακα L. Για τον εντοπισμό των residuals αναπτύχθηκε η βοηθητική συνάρτηση [res,linepixels] = myHoughRes(I,rhoScale,thetaScale,L). Στην συνάρτηση αυτή υπολογίζονται οι συντεταγμένες αρχής και τέλους των ευθειών που περιγράφονται από τα ζεύγη *[ρ,ϑ]* στο καρτεσιανό επίπεδο. Έπειτα με τη χρήση του αλγόριθμου του Bresenham για ευθείες που υπάρχει στο script bresenham.m, υπολογίζονται οι συντεταγμένες όλων των pixel που ανήκουν στις ευθείες που εντοπίστηκαν από τον μετασχηματισμό Hough. Αφού γίνει έλεγχος για επικαλυπτόμενα pixel, ο αριθμός των pixel που ανήκουν στις ευθείες αφαιρείται από τον συνολικό αριθμό των pixel της εικόνας έτσι ώστε να προκύψει ο δείκτης res. Να σημειωθεί ότι για τον αλγόριθμο Bresenham χρησιμοποιήθηκε έτοιμο script από την ιστοσελίδα File Exchange της Mathworks. Link εδώ

Στο demo script deliverable_1.m πραγματοποιείται μια επίδειξη των προγραμμάτων που αναλύθηκαν παραπάνω σε μια από τις δοκιμαστικές εικόνες που έχουν δωθεί στην εκφώνηση. Για την προεπεξεργασία της εικόνας επιλέχθηκε να γίνει υποδειγματοληψία σε κλίμακα 0.5 καθώς σε διαφορετική περίπτωση ο χρόνος εκτέλεσης του προγράμματος θα ήταν απαγορευτικός. Ο edge detector που επιλέχθηκε είναι ο canny, με τυπική απόκλιση 1 και τιμές κατωφλίωσης [0.05 0.1]. Επίσης εφαρμόστηκε ένα γκαουσιανό φίλτρο εξομάλυνσης με τυπική απόκλιση 5. Οι τιμές για Dro,Dtheta που επιλέχθηκαν είναι 1 και 0.25*π/180, ενώ ο αριθμός των ευθείων είναι 20.

Το γράφημα του μετασχηματισμού της εικόνας στο πεδίο του Hough παρουσιάζεται παρακάτω. Τα πράσινα τετράγωνα αντιπροσωπεύουν τα τοπικά μέγιστα που εντόπισε ο αλγόριθμος



Τέλος οι συνολικές ευθείες που εντοπίστηκαν φαίνονται στην παρακάτω εικόνα. Βλέπουμε ότι ο αλγόριθμος δούλεψε ικανοποιητικά και οι ευθείες που αντιστοιχούν στις ακμές των φωτογραφιών αναγνωρίζονται επιτυχώς. Εφόσον βλέπουμε ότι υπάρχει επικάλυψη σε κάποιες ευθείες, θα μπορούσαμε να μειώσουμε τον αριθμό n, ωστόσο επειδή τα αποτελέσματα του edge detector που χρησιμοποιήθηκε δεν ήταν πάντα αξιόπιστα, επιλέχθηκε ελαφρώς μεγαλύτερος αριθμός ευθειών ώστε να διασφαλιστεί η σωστή λειτουργία του προγράμματος.



2.2 2ο Παραδοτέο: Harris Corner Detector

Η συνάρτηση corners= myDetectHarrisFeatures(I) υλοποιεί τον Harris Corner detector όπως ζητήθηκε απ' την εκφώνηση. Τα βήματα του αλγορίθμου επεξηγούνται παρακάτω σε φυσική γλώσσα:

1. Υπολογισμός παραγώγων I_x , I_y της εικόνας στις κατευθύνσεις x,y

2. Υπολογισμός γινομένων των παραγώγων για κάθε pixel της εικόνας, $I_{x_2} = I_x * I_x$, $I_{y2} = I_y * I_y$, $I_{xy} = I_x * I_y$

3. Υπολογισμός των αθροισμάτων των γινομένων για κάθε pixel με τη χρήση γκαουσιανού φίλτρου εξομάλυνσης. $S_{x_2}(x,y) = G_\sigma * I_{x_2}$, $S_{y_2}(x,y) = G_\sigma * I_{y2}$, $S_{x_y}(x,y) = G_\sigma * I_{xy}$

4. Ορισμός του πίνακα H(x,y)=[$S_{x_2}(x,y)$ S_{xy} (x,y); S_{xy} (x,y) S_{y2} (x,y)]

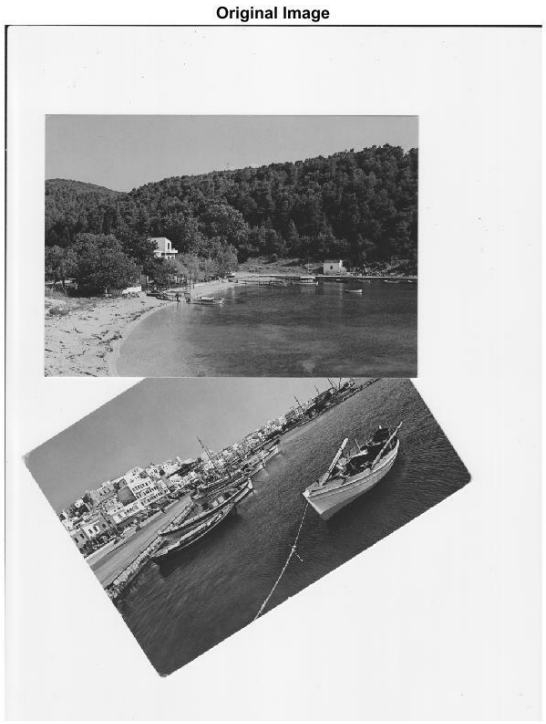
5. Υπολογισμός απόκρισης του detector για κάθε Pixel μέσω της σχέσης R=det(H)-k(Trace(H)), όπου k εμπειρικά ορισμένη σταθερά

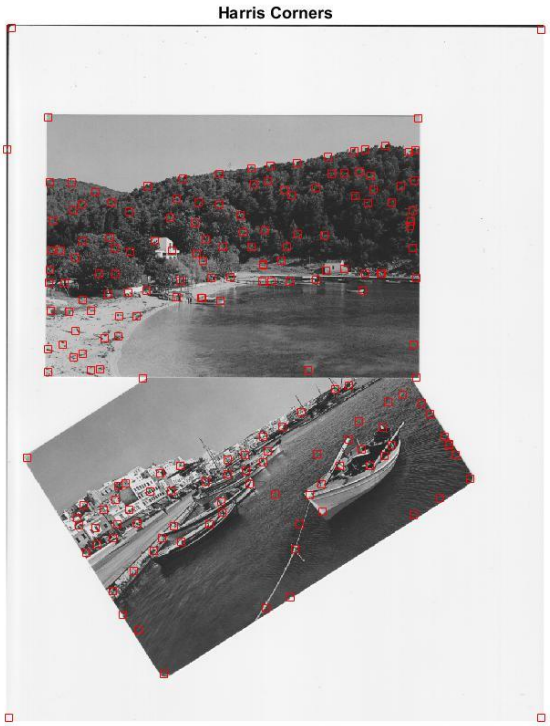
6. Υπολογισμός τοπικών μεγίστων του πίνακα R με ένα grayscale dilation φίλτρο και κατωφλίωση τους, τα αποτελέσματα της κατωφλίωσης είναι οι γωνίες harris

7. Εντοπισμός μοναδικών γωνιών harris.

Στην παρούσα υλοποίηση του αλγορίθμου υπάρχουν τρεις μεταβλητές παράμετροι. Η τυπική απόκλιση του gaussian φίλτρου, η ακτίνα του dilation φίλτρου και το κατώφλι για τον εντοπισμό γωνιών. Προφανώς για διαφορετικές εικόνες ή και για διαφορετική τιμή υποδειγματοληψίας της εικόνας πρέπει να γίνει ανάλογη προσαρμογή των παραμέτρων.

Παρακάτω απεικονίζονται οι γωνίες που εντόπισε ο αλγόριθμος Harris στην εικόνα im2 της εκφώνησης. Όπως και πριν για λόγους ταχύτητας, έγινε υποδειγματοληψία της εικόνας σε κλίμακα 0.5.





Όπως βλέπουμε η λειτουργία του αλγορίθμου είναι ικανοποιητική. Παρόλο που επιλέχθηκε σχετικά μικρή τιμή κατωφλιού και δεν επιστράφηκαν πολλές γωνίες, παρατηρούμε ότι οι γωνίες των φωτογραφιών που μας ενδιαφέρουν αναγνωρίστηκαν επιτυχώς.

2.3 3ο Παραδοτέο: Περιστροφή Εικόνας

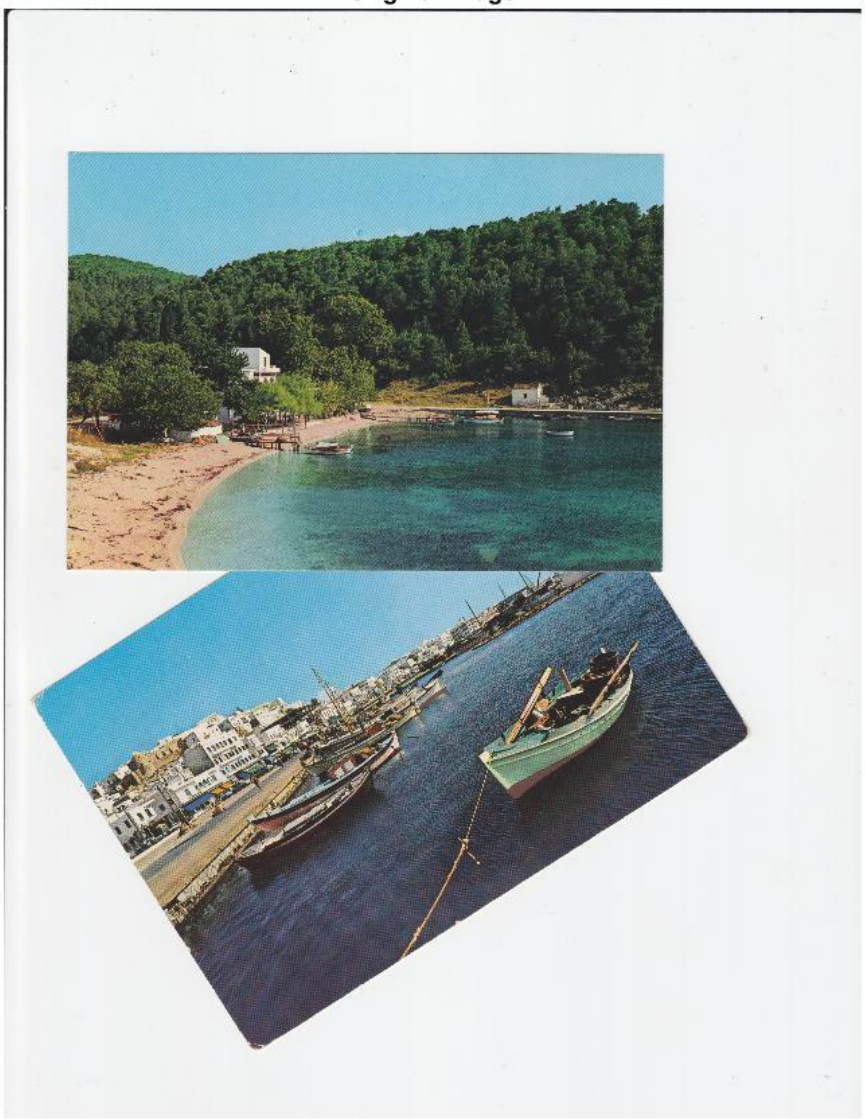
Για το 3ο παραδοτέο του 1ου μέρους αναπτύχθηκε η συνάρτηση rotImg = myImgRotation(img , angle). Αρχικά γίνεται δυναμικός υπολογισμός των διαστάσεων της νέας εικόνας. Σε περίπτωση που δεν έχουμε περιστροφή κατά ακέραιο πολλαπλάσιο του π/2, η εικόνα επεκτείνεται με μαύρα pixel. Αφού υπολογιστούν τα κεντρικά pixel της αρχικής και της τελικής εικόνας, πραγματοποιείται η περιστροφή μέσω inverse mapping με πίνακα περιστροφής

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

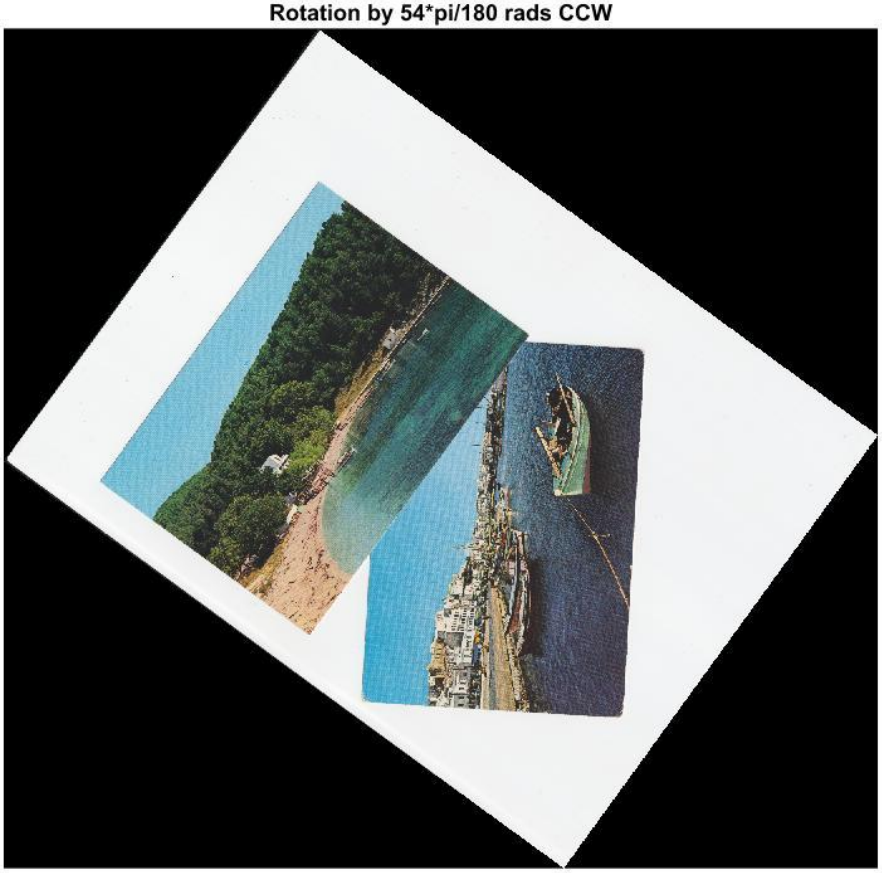
Για την σωστή αντιστοίχιση των pixels χρησιμοποιήθηκε διγραμμικός μετασχηματισμός σε 4 κατευθύνσεις με στρογγυλοποίηση προς τα κάτω όπως ζητήθηκε από την εκφώνηση. Για τον διγραμμικό μετασχηματισμό αναπτύχθηκε η βοηθητική συνάρτηση interp=myBlInterp(img,yx,zpad).

Παρακάτω απεικονίζονται οι δυο ανεξάρτητες περιστροφές που ζητήθηκαν από την εκφώνηση.

Original Image



Rotation by 54*pi/180 rads CCW



Rotation by 213*pi/180 rads CCW

3 Ενότητα Β

3.1 Εισαγωγή

Για το 2ο μέρος της εργασίας αναπτύχθηκε το script myLazyScanner.m, με το οποίο επιτυγχάνεται ο αυτόματος εντοπισμός και διαχωρισμός των φωτογραφιών που υπάρχουν στην σκαναρισμένη εικόνα. Εκτός των τριών βασικών αλγορίθμων που υλοποιήθηκαν στο 1ο μέρος της εργασίας, αναπτύχθηκε πλήθος βοηθητικών συναρτήσεων και αλγορίθμων έτσι ώστε να εξασφαλιστεί η σωστή λειτουργία του προγράμματος.

3.2 Ο αλγόριθμος myLazyScanner

3.2.1 Συνοπτική Περιγραφή Αλγορίθμου

Τα βήματα του αλγορίθμου παρουσιάζονται παρακάτω σε φυσική γλώσσα

1. Φόρτωση, Υποδειγματοληψία και Προεπεξεργασία εικόνας
2. Εντοπισμός γραμμών με μετασχηματισμό Hough
3. Εντοπισμός τομών των γραμμών Hough
4. Εντοπισμός γωνιών με Harris Corner Detector
5. Διασταύρωση τομών Hough και γωνιών Harris
6. Σύζευξη των πολύ κοντινών κόμβων
7. **Για κάθε** κόμβο
    1. Εύρεση των κάθετων σε αυτόν κόμβους και της μεταξύς τους απόστασης
8. Εύρεση πιθανών παραλληλογράμμων και δημιουργία δυαδικών μασκών από αυτά
9. Απόρριψη πλεοναζουσών μασκών και εύρεση του αριθμού των μοναδικών εικόνων
10. **Για κάθε** μοναδική μάσκα που βρέθηκε
    1. Υπερδειγματοληψία μάσκας στην αρχική κλίμακα μεγέθους της εικόνας
    2. Υπολογισμός κλίσης εικόνας σε σχέση με τον οριζόντιο άξονα
    3. Αποκοπή εικόνας μέσω της μάσκας και διόρθωση κλίσης εικόνας
    4. Αποθήκευση εικόνας

3.2.2 Επεξήγηση Βημάτων

Βήμα 1:

Όπως και στην 1η ενότητα έγινε υποδειγματοληψία της αρχικής εικόνας σε κλίμακα 0.5. Όσον αφορά την επιλογή των παραμέτρων, χρησιμοποιήθηκαν σχεδόν οι ίδιες με πριν δηλαδή προτιμήθηκε ο canny edge detector με τυπική απόκλιση 1 και τιμές κατωφλίωσης [0.05 0.1], γκαουσιανό φίλτρο εξομάλυνσης με τυπική απόκλιση 5, οι τιμές για Dro,Dtheta που επιλέχθηκαν είναι 1 και 0.25*π/180, ενώ ο αριθμός των ευθείων είναι 15.

Βήματα 2,3 & 4:

Οι γραμμές που εντοπίζει ο μετασχηματισμός Hough βρέθηκαν με τη χρήση της myHoughTransform και οι γωνίες του Harris Detector με την χρήση της myDetectHarrisFeatures. Οι τομές των ευθειών Hough, βρέθηκαν με απλούς γεωμετρικούς υπολογισμούς.

Βήμα 5:

Καθώς προκύπτουν αρκετές τομές Hough σε σημεία εκτός των φωτογραφιών, τα οποία θα πρόσθεταν πολυπλοκότητα στην επίλυση του προβήματος επιλέχθηκε να γίνει μια διασταύρωση μεταξύ αυτών και των γωνιών. Επειδή τα pixel των γωνιών Harris δεν συμπίπτουν ακριβώς με αυτά των τον τομών, δημιουργήθηκε μια κυκλική μάσκα μικρής ακτίνας (50-100 pixel, ανάλογα και με τη κλίμακα υποδειγματοληψίας), γύρω από κάθε κόμβο Hough. Αν η μάσκα ενός κόμβου Hough εμπεριέχει γωνίες Harris, ο κόμβος αυτός διατηρείται ενώ σε διαφορετική περίπτωση απορρίπτεται.

Βήμα 6

Για την περαιτέρω εξάλειψη του θορύβου στον αριθμό των κόμβων, επιλέχθηκε να γίνει ομαδοποίηση και κανονικοποίηση αυτών που βρίσκονται πολύ κοντά. Όπως και στο Βήμα 5, επιλέχθηκε ένα κατώφλι απόστασης (πάλι 50-100 pixel). Έτσι όσοι κόμβοι έχουν πολύ μικρή απόσταση μεταξύ τους ενώνονται.

Βήμα 7

Για το μέρος αυτό υλοποιήθηκε η βοηθητική ρουτίνα vertices = findVertices(groupedcorners,minangle,maxangle), η οποία πραγματοποιεί σάρωση του πίνακα με τους κόμβους που και εντοπίζει ποιοι από αυτούς είναι υποψήφιες κορυφές τετραπλεύρου, μέσω ελέγχου της γωνίας. Ο cell πίνακας που επιστρέφεται περιλαμβάνει για κάθε κορυφή, τα ζεύγη των κόμβων που ικανοποιούν το κριτήριο της γωνιότητας και τις αποστάσεις τους απ' τον κόμβο. Στην παρούσα περίπτωση έγινε έλεγχος καθετότητας μεταξύ των κόμβων, ωστόσο λόγω των

στρογγυλοποιήσεων που έχουν προηγηθεί, δεν ήταν εφικτό οι κόμβοι να δημιουργούν τέλεια κάθετες πλευρές, οπότε επιλέχθηκε ένα όριο χαλάρωσης στις γωνίες, περίπου 10 μοιρών.

Βήμα 8

Στο βήμα αυτό παράγονται οι δυαδικές μάσκες οι οποίες και θα βοηθήσουν στην αποκοπή των εικόνων που θα ακολουθήσει. Οι δυο βοηθητικές ρουτίνες που αναπτύχθηκαν για αυτό το βήμα είναι οι:

- vertex=LocateVertex(vertices,index,point,edge)
- binmasks_final = findBinaryMasks(groupedcorners,vertices,imH,imW)

Με τη χρήση των οποίον εντοπίζονται όλα τα πιθανά τετράπλευρα που εμπεριέχει ο cell πίνακας vertices. Έπειτα μέσω της συνάρτησης poly2mask δημιουργούνται δυαδικές μάσκες με κορυφές τα τετράπλευρα που βρέθηκαν και αποθηκεύονται στον cell array binmasks_final.

Βήμα 9

Δεδομένου ότι οι μάσκες που δημιουργούνται είναι πολύ περισσότερες σε πλήθος από τον πραγματικό αριθμό των φωτογραφιών που υπάρχουν στην εικόνα. Η συνάρτηση imageMasks = findImageCount(binaryMasks,threshold) πραγματοποιεί έλεγχο για μάσκες οι οποίες προκύπτουν από συνένωση άλλων μασκών που βρίσκονται στην ίδια δομή και τις εξαλείφει. Στον τελικό πίνακα imageMasks αποθηκεύονται οι μάσκες οι οποίες αντιστοιχούν στα μοναδικά παραλληλόγραμμα της εικόνας

Βήμα 10

Στο τελευταίο βήμα του αλγορίθμου γίνεται η αποκοπή, περιστροφή και αποθήκευση των τελικών εικόνων. Αρχικά γίνεται υπερδειγματοληψία των δυαδικών μασκών στην αρχική κλίμακα της εικόνας με σκοπό να γίνει αποκοπή των φωτογραφιών στο αρχικό τους μέγεθος. Πρέπει να σημειωθεί ότι επειδή η διαδικασία της περιστροφής των εικόνων χωρίς υποδειγματοληψία ήταν ιδιαίτερα χρονοβόρα, λόγω του μεγάλου μεγέθους τους, οι εντολές που αφορούν το rescaling έχουν απενεργοποιηθεί από προεπιλογή εντός του κώδικα με μορφή σχολίων. Ο αναγνώστης μπορεί να τις ενεργοποιήσει με δική του επιλογή για να δει το αποτέλεσμα του αλγορίθμου στην αρχική εικόνα. Η αποκοπή των εικόνων έγινε με την βοήθεια των συναρτήσεων του Computer Vision Toolbox της MATLAB, regionprops και imcrop. Με τη χρήση των Extrema Points της regionprops, γίνεται υπολογισμός της γωνίας της κάτω πλευράς του παραλληλόγραμμου με τον άξονα των x. Η γωνία αυτή δίνεται ως όρισμα στην myImgRotation, όπου γίνεται και η διορθωτική περιστροφή της εικόνας. Τέλος οι εικόνες αποθηκεύονται στο ίδιο directory με τη μορφή που ζητήθηκε από την εκφώνηση.

- Παρατήρηση

Προκειμένου να επιτευχθεί μεγαλύτερη ακρίβεια κατά την αποκοπή στην περίπτωση που οι φωτογραφίες που ανιχνεύθηκαν έχουν μια αρκετά μεγάλη κλίση, έχει αναπτυχθεί μια εναλλακτική υλοποίηση του παραπάνω βήματος. Στην εν λόγω υλοποίηση, αφού εντοπιστεί η γωνία, η δυαδική μάσκα και η αρχική εικόνα περιστρέφονται ξεχωριστά και στη συνέχεια πραγματοποιείται η εφαρμογή της μάσκας στην εικόνα και η αποκοπή. Αυτή η μέθοδος απαιτεί πολύ μεγαλύτερο χρόνο σε επίπεδο υπολογισμών, ειδικά αν γίνει αφού έχουμε επιστρέψει στο αρχικό μέγεθος της εικόνας και για αυτό το λόγω έχει απενεργοποιηθεί από προεπιλογή στο πρόγραμμα. Αν ο αναγνώστης επιθυμεί να ελέγξει την ορθότητα της παραπάνω υλοποίησης, προτείνεται να αντικαταστήσει την myImgRotation με την imrotate της MATLAB για εξοικονόμηση χρόνου.

3.3 Αποτελέσματα

Τα αποτελέσματα που επιστρέφει ο αλγόριθμος για τις 5 εικόνες της εκφώνησης παρουσιάζονται παρακάτω. Επίσης για την εικόνα 2 μόνο, συμπεριλαμβάνεται και το αποτέλεσμα της αποκοπής με τη χρήση της μεθόδου 2

<u>Εικόνα 1</u>

- Αρχική εικόνα

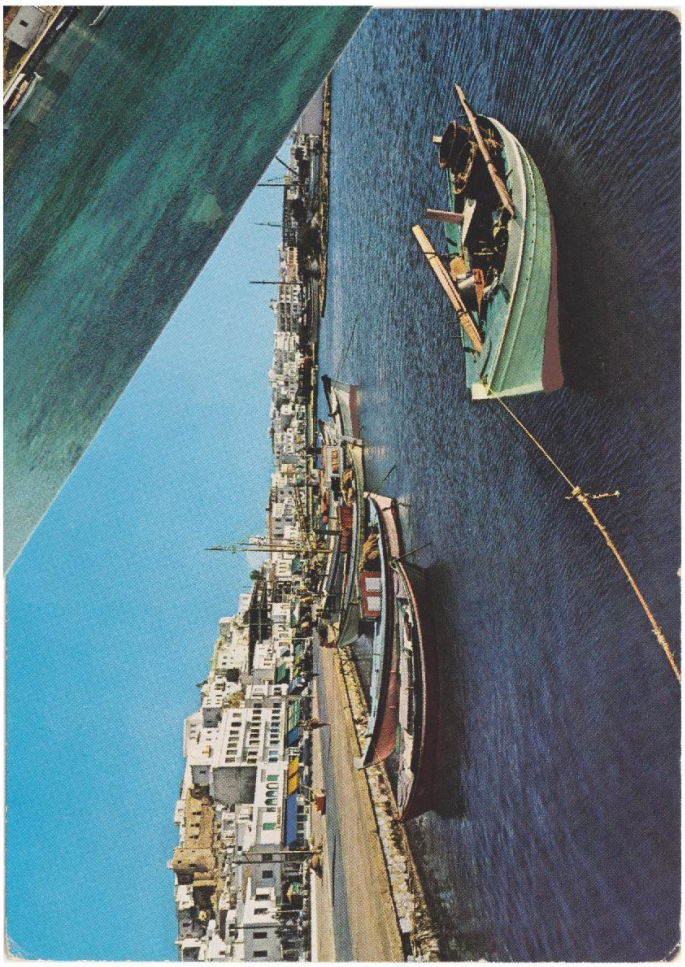
- Εικόνες που εντοπίστηκαν

Εικόνα 2

- Αρχική Εικόνα

- Εικόνες που εντοπίστηκαν

Μέθοδος 2



Εικόνα 3

- Αρχική εικόνα

- Εικόνες που εντοπίστηκαν

Εικόνα 4

- Αρχική Εικόνα

- Εικόνες που εντοπίστηκαν

Εικόνα 5

- Αρχική Εικόνα

- Εικόνες που εντοπίστηκαν

➢ Παρατηρήσεις

Από τα παραπάνω αποτελέσματα βλέπουμε ότι ο αλγόριθμος δούλεψε ικανοποιητικά σε γενικές γραμμές. Βεβαίως υπάρχουν αρκετά περιθώρια βελτίωσης. Βλέπουμε ότι η δεύτερη φωτογραφία στην 5η δοκιμαστική εικόνα δεν εντοπίστηκε καν, καθώς ο Harris Corner Detector απέτυχε να εντοπίσει όλες τις γωνίες τις φωτογραφίας και κατ'επέκταση να σχηματιστεί παραλληλόγραμο από τον αλγόριθμο myLazyScanner. Πιθανώς με διαφορετική επιλογή παραμέτρων στην myDetectHarrisFeatures ο αλγόριθμος θα ήταν πιο αποτελεσματικός. Επίσης πρέπει να σημειωθεί ότι στις εικόνες με περιεστρεμμένες φωτογραφίες χρησιμοποιήθηκαν πιο χαλαρά όρια γωνιών από αυτά των ορθά στοιχισμένων, τα οποία υπάρχουν ως σχόλια στον κώδικα. Δυστυχώς λόγω πίεσης χρόνου ήταν αδύνατο να εντοπιστεί ένα ενιαίο σετ παραμέτρων για να ικανοποιηθεί η one-script-fits-all λογική που ζητήθηκε από την εκφώνηση.