Εργασία 2 στην

Ψηφιακή επεξεργασία εικόνας

Hough, Harris, Rot και αποκοπή εικόνων

Χρυσούλα Μόσχου
8/6/2020
moschouc@ece.auth.gr
ΑΕΜ 9045

Εισαγωγή

Η παρούσα εργασία αφορά στο πρώτο κομμάτι την υλοποίηση μιας σουίτας ρουτινών και στο δεύτερο την αξιοποίηση της σουίτας αυτής με σκοπό την υλοποίηση ενός LazyScanner. Οι ρουτίνες της σουίτας θα περιέχουν τις υλοποιήσεις για τον μετασχηματισμό Hough, τον Harris corner detector και την περιστροφή εικόνας. Παρακάτω παρουσιάζεται η λογική με την οποία προσεγγίστηκε σε κάθε ενότητα και το πως έγινε η υλοποίηση της στο πρόγραμμα matlab. Η εργασία πέρα από αυτή την αναφορά συνοδεύεται και από τα εξής αρχεία:

- myHoughTransform.m (Ενότητα 1.1)
- deliverable_1.m (Ενότητα 1.1)
- myDetectHarrisFeatures.m (Ενότητα 1.2)
- deliverable_2.m (Ενότητα 1.2)
- myImgRotation.m (Ενότητα 1.3)
- deliverable_3.m (Ενότητα 1.3)

1.1 Hough Transform

Σε αυτή την ενότητα ζητείται η υλοποίηση του Hough Transform μέσω της κατασκευής της συνάρτησης myHoughTransform η οποία δέχεται στην είσοδο μια δυαδική εικόνα img_binary, την διακριτότητα Drho στην διάσταση ρ σε pixels, την διακριτότητα Dtheta στην διάσταση θ σε rads και το πλήθος n των ισχυρότερων ευθειών που θέλουμε να επιστρέψουμε. Με βάση αυτές τις εισόδους η συνάρτηση επιστρέφει τον πίνακα μετασχηματισμού Hough H, τον πίνακα L που περιέχει τις n μεγαλύτερες τιμές και το πλήθος res των σημείων της δυαδικής εικόνας που δεν ανήκουν στις n πιο ισχυρές ευθείες που έχουν εντοπιστεί.

Αλγόριθμος για τον Hough transform

Για την υλοποίηση του μετασχηματισμού Hough η λογική που ακολουθείται είναι η κλασική.

Αρχικά ορίζουμε τις διαστάσεις του πίνακα H. Οι διαστάσεις του Hough εξαρτώνται από το εύρος των τιμών των μεταβλητών ρ και θ και την διακριτότητα τους. Για το ρ θέλουμε τα όρια να κυμαίνονται από

$-D \leq \rho \leq D$ (drho) όπου D η μέγιστη ακτίνα της εικόνας δηλαδή η διαγώνιος της και για το θ από $-90^o \leq \theta < 90^o$(dtheta αφού γίνει μετατροπή σε rads). Η οριζόντια διάσταση του H είναι ίση με το πλήθος των ρ (drho) ενώ η κάθετη ίση με το πλήθος των θ (dtheta). Είναι εμφανές ότι μεταβάλλοντας τις διακριτότητες Drho και Dtheta μεταβάλλονται και οι διαστάσεις του πίνακα H. Γενικά όσο μεγαλύτερη ακρίβεια θέλουμε τόσο πρέπει να αυξήσουμε τις διαστάσεις στον πίνακα συσσώρευσης H. Συνήθως ο πίνακας συσσώρευσης επιλέγεται να έχει ίδιες διαστάσεις με την εικόνα ωστόσο ανάλογα την εφαρμογή μπορεί να έχει και μικρότερες. Στα πλαίσια της εργασίας σε αυτό το στάδιο οι διακριτότητες επιλέχθηκαν ίσες με 1.

Αφού καθορίσουμε τις διαστάσεις του H θα πρέπει να εντοπίσουμε τα σημεία της δυαδικής εικόνας που δεν έχουν μηδενική τιμή και να τα συγκεντρώσουμε. Για κάθε ένα από αυτά τα σημεία δημιουργούμε ένα κελί (cell_r{i}) . Για κάθε κελί i υπολογίζουμε όλες τις τιμές ρ (cell_r{i}) που προκύπτουν από τη σχέση $\rho = x_i \cos\theta + y_i \sin\theta$ για όλες τις τιμές θ. Οι τιμές που προκύπτουν αντιστοιχίζονται στην κοντινότερη στάθμη του drho και αυξάνεται κατά 1 το αντίστοιχο σημείο του πίνακα H.

Εύρεση n ισχυρότερων ευθειών, κατασκευή πίνακα L και υπολογισμός του res

Αρχικά θεωρούμε ότι το res είναι ίσο με όλα τα pixel της δυαδικής εικόνας εισόδου.

Επίσης να σημειωθεί ότι επιθυμούμε ο πίνακας L να έχει n γραμμές και 2 στήλες. Στην πρώτη στήλη θα αποθηκεύεται η τιμή ρ μιας από τις n ισχυρότερες ευθείες και στην δεύτερη στήλη η αντίστοιχη τιμή θ.

Οι n ισχυρότερες ευθείες της εικόνας αντιστοιχούν στα n μεγαλύτερα τοπικά μέγιστα του πίνακα H. Αφού βρεθούν αυτές οι μέγιστες τιμές σκανάρουμε τον πίνακα H. Μόλις εντοπιστεί τοπικό μέγιστο σε μια θέση (i,j) εισάγουμε τις τιμές ρ και θ που αντιστοιχούν σε αυτή τη θέση στον πίνακα L και μειώνουμε το res κατά το πλήθος των pixel από τα οποία διέρχεται η ευθεία (ρ,θ) δηλαδή κατά H(i,j).

Εφαρμογή του μετασχηματισμού στην εικόνα img2 και προβολή των αποτελεσμάτων

Στο αρχείο deliverable_1.m χρησιμοποιείται η συνάρτηση myHoughTranform που κατασκευάστηκε για τον εντοπισμό ευθειών πάνω στην εικόνα img2.jpg.

Η συγκεκριμένη εικόνα (όπως και οι υπόλοιπες που δοθήκαν για τα υπόλοιπα βήματα) έχει αρκετά μεγάλες διαστάσεις κάτι που προκαλεί μεγάλες καθυστερήσεις στην εκτέλεση του κώδικα. Για το σκοπό αυτό πριν βρούμε τον μετασχηματισμό Hough της εικόνας κάνουμε downsampling μειώνοντας τις διαστάσεις της με τη χρήση της συνάρτησης imresize. Το κατά πόσο θα μειωθούν οι διαστάσεις της εικόνας καθορίζεται από τη μεταβλητή scale.

Η εικόνας μας έχει αρκετή πληροφορία στο εσωτερικό των φωτογραφιών της κάτι που μπορεί να δυσκολέψει την εύρεση ακμών. Για την βελτίωση των αποτελεσμάτων πέρα από το downsampling θα πρέπει να εφαρμοστούν και μερικά φίλτρα στην επεξεργασία της εικόνας πριν την εύρεση του Hough. Εφαρμόζεται ένα φίλτρο gauss για την εξομάλυνση της εικόνας και ένας μετασχηματισμός κατωφλοίωσης. Η κατωφλοίωση βοηθάει αρκετά έτσι ώστε η πληροφορία στο εσωτερικό των φωτογραφιών να ομογενοποιηθεί και να μην εντοπιστούν ακμές εκεί μέσα. Φυσικά αν θέλουμε να έχουμε εντοπισμό ακμών ακόμη και στο εσωτερικό των φωτογραφιών (π.χ. στον ορίζοντα ουρανού θάλασσας) μπορούμε να παραλείψουμε αυτό το βήμα.

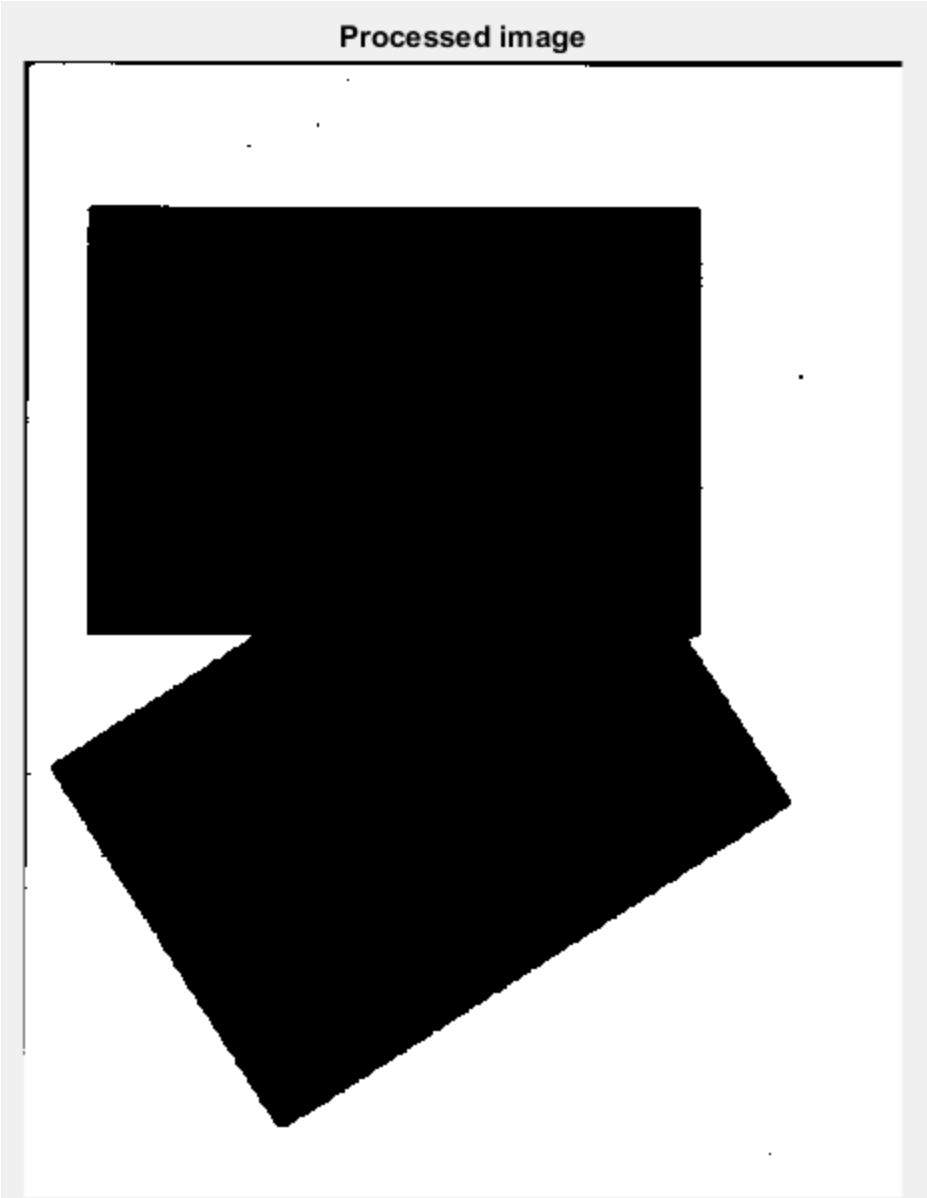
Αυτές οι δύο διαδικασίες βοηθούν πάρα πολύ στην δημιουργία της δυαδικής εικόνας που θα εισάγουμε στην myHoughTranform. Η δυαδική εικόνα προέρχεται από έναν edge detector. Σε αυτό το στάδιο χρησιμοποιήθηκε ο έτοιμος edge detector της matlab edge(method → ‘canny’). Προτιμήθηκε η χρήση αυτής της μεθόδου για να μην επηρεαστεί όσο το δυνατόν λιγότερο το αποτέλεσμα από τα pixel θορύβου. Επιπλέον με τη μέθοδο ‘canny’ τόσο οι ισχυρές όσο και οι αδύναμες ακμές. Για τις αδύναμες ακμές εξετάζεται το κατά πόσο ανήκουν σε πραγματική ακμή ή προέρχονται απλά από θόρυβο και στην πρώτη περίπτωση ενώνονται με τις πιο ισχυρές ακμές.

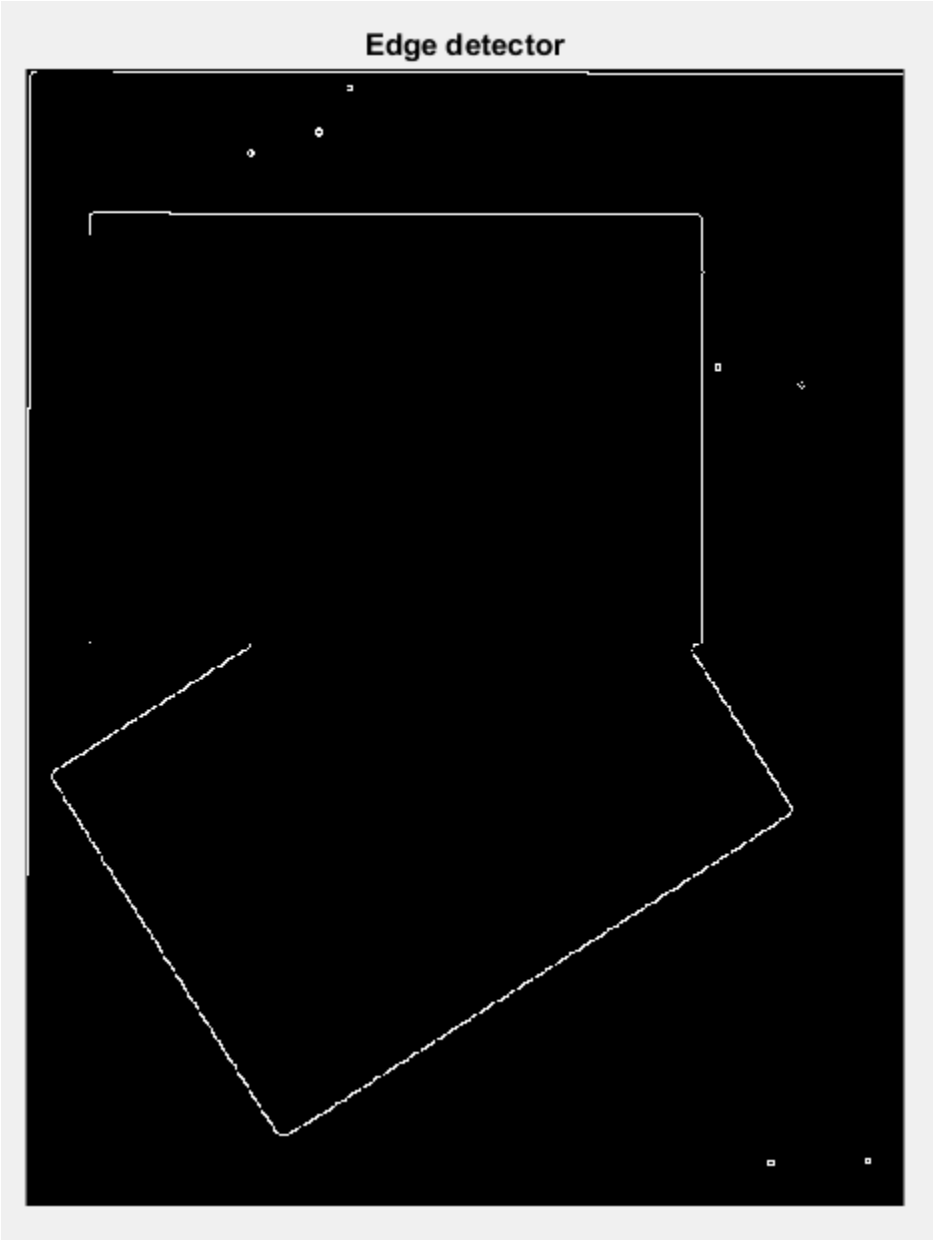
Έπειτα από αυτή την προ επεξεργασία καλείται η myHoughTransform. Στη συνέχεια προβάλλεται ο πίνακας Hough με κόκκινα τετράγωνα στα n τοπικά μέγιστα και η αρχική εικόνα του script με κόκκινες γραμμές όπου έχουν εντοπιστεί ακμές. Να σημειωθεί ότι επειδή οι τιμές ρ των n ευθειών είναι υπολογισμένες στην μικρή εικόνα θα πρέπει να πολλαπλασιαστούν με το scale για να προβληθούν σωστά στην αρχική οθόνη. Οι τιμές των θ δεν επηρεάζονται από

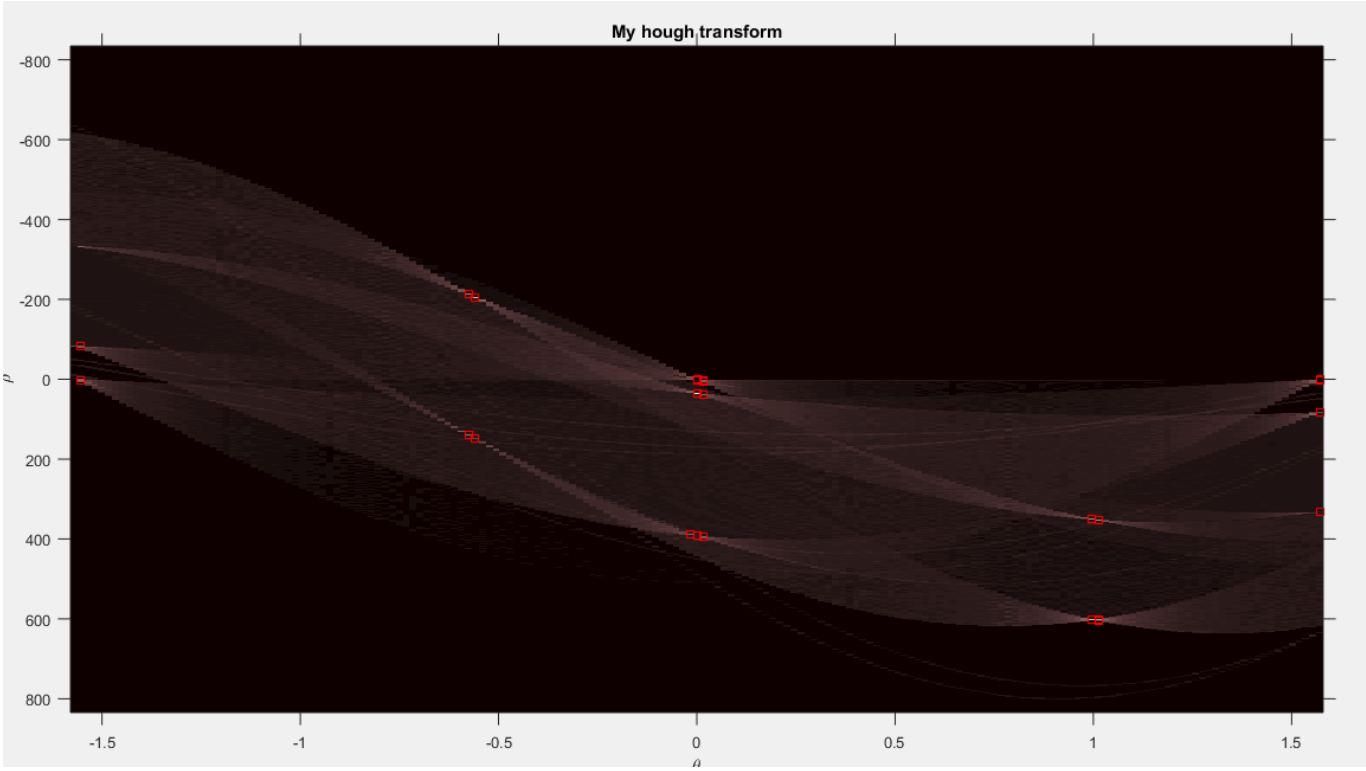
την κλιμάκωση. Τέλος εκτυπώνεται στην οθόνη το πλήθος των pixel της αρχικής εικόνας που προσεγγιστικά δεν ανήκουν στις ακμές. Η προσέγγιση και όχι ο ακριβής υπολογισμός οφείλεται στο downsampling της εικόνας. Το res που επιστρέφει η myHoughTransform αφορά την scaled binary image. Επομένως αναλογικά με την εικόνα σε σμίκρυνση υπολογίζονται τα pixel για την αρχική οθόνη (original_res).

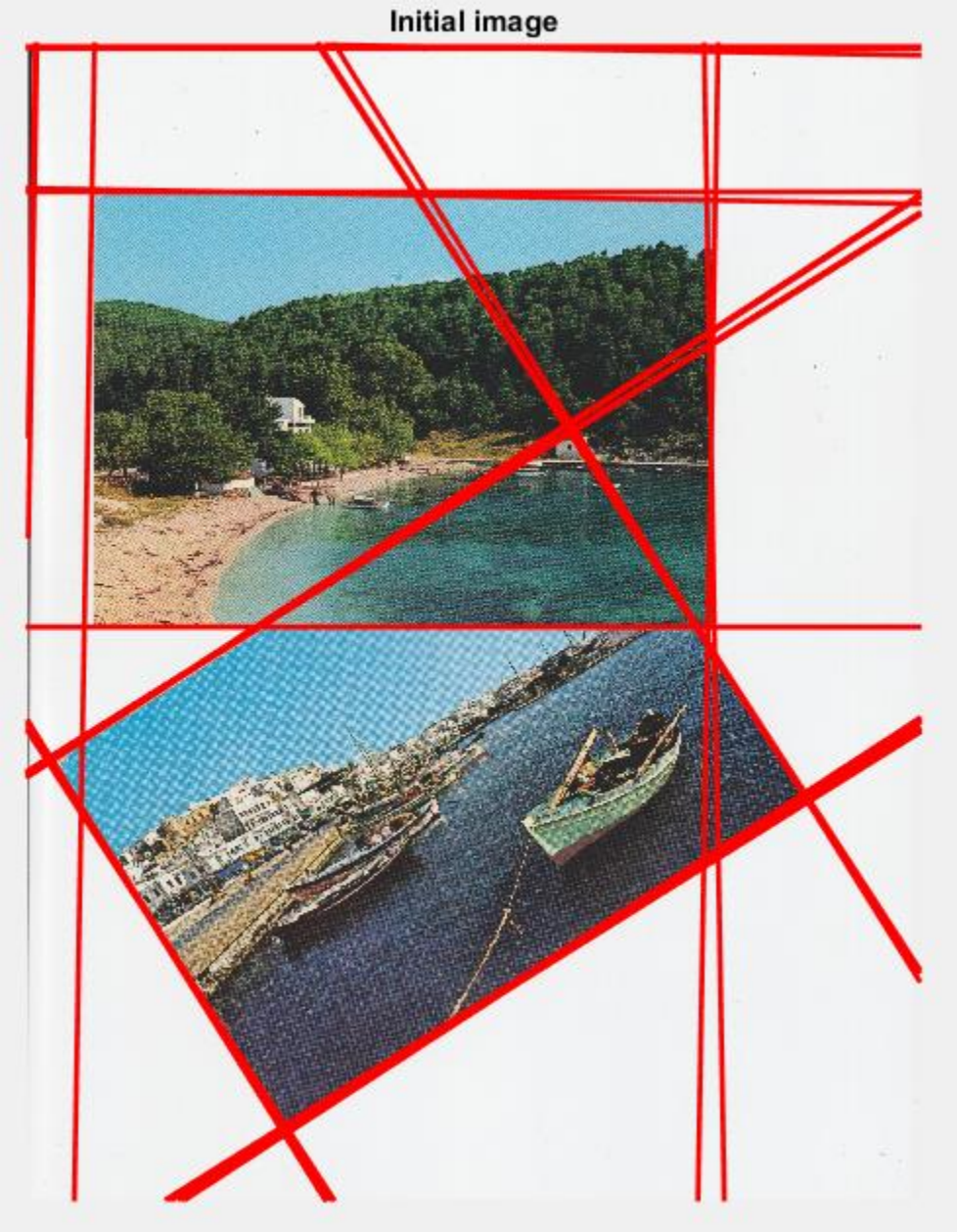
Τα αποτελέσματα του script φαίνονται παρακάτω για :

- Scale = 10
- Sigma στο φίλτρο gauss = 0.5
- Κατωφλοίωση στο 0.9
- Μέθοδο edge detector ‘canny’ και
- n = 50
- Dtheta = 1*pi/180 rad
- Drho = 1 pixel


Processed image


Edge detector


My hough transform
-800
-600
-400
-200
0
200
400
600
800
ρ
-1.5
-1
-0.5
0
0.5
1
1.5
θ


Initial image

1.2 Harris corner detector

Σε αυτή την ενότητα υλοποιείται η συνάρτηση myDetectHarrisFeatures η οποία βασίζεται στη μαθηματική ανάλυση της εκφώνησης για τον εντοπισμό γωνιών πάνω σε μια εικόνα. Η ανάλυση της εκφώνησης θα μπορούσε να εφαρμοστεί και για τον εντοπισμό ακμών αλλά στο παρών παραδοτέο εξετάζονται μόνο οι ακμές.

Για την υλοποίηση χρειάζεται αρχικά να υπολογίσουμε τις μερικές παραγώγους της φωτεινότητας $I(x_1, x_2)$ μιας gray scale εικόνας . Ο υπολογισμός τους γίνεται με τη χρήση συνέλιξης με υψιπερατή μάσκα. Η μάσκα που χρησιμοποιήθηκε είναι σύμφωνη με τη μέθοδο Sobel. Η μάσκα για τον υπολογισμό του Gy (columns) δηλαδή της μερικής παραγώγου κατά τη διεύθυνση y είναι η $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$. Για τον υπολογισμό του Gx (rows) δηλαδή της μερικής παραγώγου κατά τη διεύθυνση x είναι η $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$. Τα G_x και G_y υπολογίζονται με τη συνέλιξη του I (image) με τις μάσκες ή εναλλακτικά στον κώδικα με την χρήση της εντολής imfilter().

Αφού υπολογιστούν οι μερικές παράγωγοι G_x και G_y πρέπει να κατασκευαστεί ο πίνακας $G_x * G_y$, G_x^2 και G_y^2.

Σύμφωνα με τη θεωρία για να καταλήξουμε στον πίνακα M ο οποίος θα μας δώσει τα σημεία που είναι πάνω ή πολύ κοντά σε γωνία σχηματίζουμε τον συμμετρικό πίνακα $\begin{bmatrix} G_x^2 & G_x * G_y \\ G_x * G_y & G_y^2 \end{bmatrix}$ και του εφαρμόζουμε ένα φίλτρο Gauss.

Οι ιδιοτιμές του πίνακα M δίνουν την πληροφορία που μας ενδιαφέρει. Από τη σχέση $R = \det(M) - k * trace(M) \rightarrow R = \lambda_1 * \lambda_2 - k\ (\lambda_1 + \lambda_2)^2$ όπου k>0 , λ οι ιδιοτιμές μπορούμε πλέον να εξετάσουμε ποια σημεία βρίσκονται σε γωνία και ποια όχι. Όσα σημεία μας ενδιαφέρουν θα είναι θετικά (ή μεγαλύτερα από το threshold που θα ορίσουμε) ενώ τα υπόλοιπα θα είναι είτε μηδέν (ομαλά σημεία) είτε αρνητικά (θα βρίσκονται σε ακμή). Συνεπώς σκανάροντας τον πίνακα R μόλις εντοπίσουμε τις τιμές που μας ενδιαφέρουν τις αποθηκεύουμε στον πίνακα corners που θα περιέχει τις ισχυρότερες γωνίες τις εικόνας. Για υπολογιστική ευκολία ορίζουμε το μήκος του πίνακα corners ίσο με όλα τα pixel της εικόνας.

Η εφαρμογή του παραπάνω κώδικα γίνεται στο deliverable_2.m για την εικόνα im2.jpg. Όπως και για την προηγούμενη ενότητα έτσι και εδώ φροντίζουμε να εφαρμόσουμε κάποια επεξεργασία στην εικόνα πριν την εισάγουμε στην myDetectHarrisFeatures. Η επεξεργασία σε γενικά πλαίσια είναι ίδια με την προηγούμενη ενότητα όσον αφορά την φύση των φίλτρων. Πέρα από την προ επεξεργασία βελτίωση στο αποτέλεσμα μπορούμε να επιτύχουμε

μεταβάλλοντας και τις τιμές του κατωφλιού (threshold) της R που θεωρούμε ότι αντιστοιχούν σε γωνία.

Αφού υπολογιστούν οι ισχυρότερες γωνίες τις εικόνας (corners) με την myDetectHarrisFeatures προβάλλεται η αρχική εικόνα με κόκκινα τετράγωνα στις εντοπισμένες γωνίες. Να σημειωθεί ότι επειδή οι συντεταγμένες που περιέχει το corners αφορά την εικόνα σε κλιμάκωση θα πρέπει να πολλαπλασιαστούν με το scale για να προβληθούν σωστά στις διαστάσεις της αρχικής εικόνας.

Τα αποτελέσματα του script φαίνονται παρακάτω για :

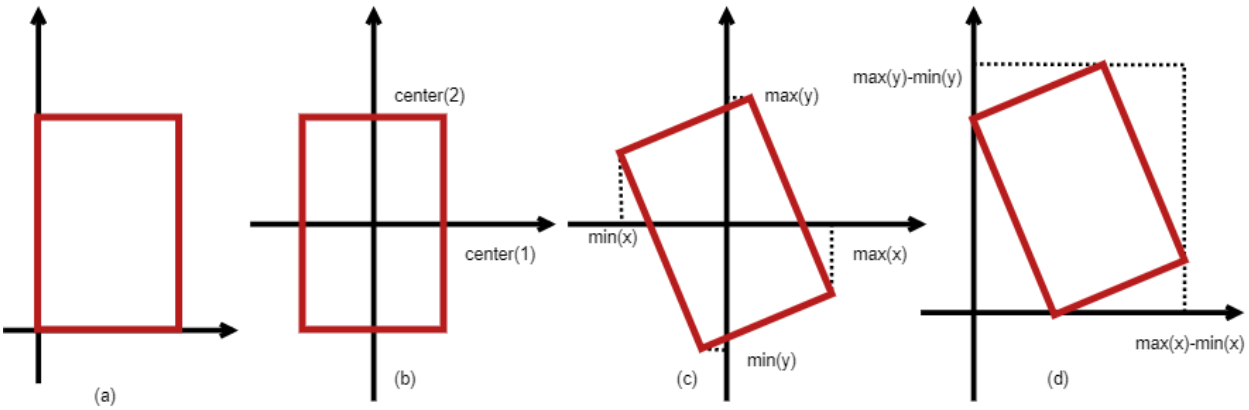
- Scale = 10
- Sigma = 2 στο φίλτρο gauss (στο στάδιο του preprocess)
- Κατωφλοίωση στο 0.9 (στο στάδιο του preprocess)
- Κατώφλι για το R (threshold) = 0.002
- Παράμετρος k = 0.04

1.3 Rotation

Σε αυτή την ενότητα ζητείται η υλοποίηση μιας συνάρτησης η οποία θα περιστρέφει μια εικόνα img αντίθετα από τη φορά του ρολογιού κατά γωνία angles σε rads και την επιστρέφει (rotImg). Πρέπει να εξασφαλιστεί ότι η εικόνα αφού περιστραφεί θα χωράει ολόκληρη στην εικόνα εξόδου. Θεωρούμε ότι τα pixel του background είναι μαύρα.

Για να μπορέσουμε να υλοποιήσουμε τη συνάρτηση περιστροφής θα πρέπει να βρούμε την αντίστροφη διαδικασία που αντιστοιχεί τα pixel της στραμμένης εικόνας στα pixel της αρχικής εικόνας εισόδου.

Πέρα από την αντίστροφη διαδικασία θα πρέπει να βρεθούν και οι διαστάσεις της καινούριας εικόνας.



Σχήμα. (α) η αρχική εικόνα ,(b) η αρχική εικόνα μετατοπισμένη ώστε το κέντρο της να βρίσκεται στο (0,0), (c) στροφή εικόνας κατά ϑ και (d) η τελική εικόνα στις νέες διαστάσεις εισόδου

Η διαδικασία με την οποία τα pixel της αρχικής εικόνας (A) μετασχηματίζονται στα pixel της τελικής στραμμένης εικόνας (B) φαίνεται στο παραπάνω σχήμα. Αρχικά μεταφέρουμε το κέντρο της εικόνας (center) στην αρχή των αξόνων. Έπειτα την περιστρέφουμε και τέλος την μεταφέρουμε ανάλογα με τα max και min των x και y που προκύπτουν (το πως προκύπτουν αυτές οι τιμές φαίνεται στη συνέχεια). Ο συνολικός μετασχηματισμός δίνεται από τους παρακάτω επιμέρους μετασχηματισμούς:

$$\begin{bmatrix} x_B \\ y_B \\ 1 \end{bmatrix} = T_{trans(2)} * T_{rot} * T_{trans(1)} * \begin{bmatrix} x_A \\ y_A \\ 1 \end{bmatrix}$$ όπου :

$$T_{trans(1)} = \begin{bmatrix} 1 & 0 & -center(1) \\ 0 & 1 & -center(2) \\ 0 & 0 & 1 \end{bmatrix},\ T_{trans(2)} = \begin{bmatrix} 1 & 0 & -min(x) \\ 0 & 1 & -min(y) \\ 0 & 0 & 1 \end{bmatrix},\ T_{rot} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Η αντίστροφη διαδικασία μπορεί να προκύψει αντιστρέφοντας τους παραπάνω πίνακες:

$$\begin{bmatrix} x_A \\ y_A \\ 1 \end{bmatrix} = {T_{trans(1)}}^{-1} * {T_{rot}}^{-1} * {T_{trans(2)}}^{-1} * \begin{bmatrix} x_B \\ y_B \\ 1 \end{bmatrix}$$ όπου:

$$ {T_{trans(1)}}^{-1} = \begin{bmatrix} 1 & 0 & center(1) \\ 0 & 1 & center(2) \\ 0 & 0 & 1 \end{bmatrix}, \quad {T_{trans(2)}}^{-1} = \begin{bmatrix} 1 & 0 & \min(x) \\ 0 & 1 & \min(y) \\ 0 & 0 & 1 \end{bmatrix} \text{ και}$$

$$ {T_{rot}}^{-1} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Τελικά:

$$\begin{bmatrix} x_A \\ y_A \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta(x_B + \min(x)) - sin\theta(y_B + \min(y)) + center(1) \\ sin\theta(x_B + \min(x)) + cos\theta(y_B + \min(y)) + center(2) \\ 1 \end{bmatrix}$$

Τα xB και yb θα τρέχουν στις νέες διαστάσεις της εικόνας οι οποίες φαίνονται στο Σχήμα (d). Για τον υπολογισμό των νέων διαστάσεων από τα min(x),min(y),max(x) και max(x) δεν χρειάζεται να μετασχηματίσουμε ολόκληρη την εικόνα αλλά μόνο τις τέσσερις γωνίες της (corners). Οι γωνίες πρώτα μεταφέρονται σύμφωνα με τον μετασχηματισμό $T_{trans(1)}$, έπειτα περιστρέφονται με τον πίνακα T_{rot} και υπολογίζονται τα παραπάνω ελάχιστα και μέγιστα.

Η νέα εικόνα θα βρεθεί χρησιμοποιώντας την παραπάνω αντίστροφη διαδικασία και εφαρμόζοντας bilinear interpolation. Για τον υπολογισμό των pixel με bilinear interpolation χρησιμοποιείται η έτοιμη συνάρτηση της matlab interp2(). Σε περίπτωση που η εικόνα είναι RGB η interp2() εφαρμόζεται για κάθε χρώμα χωριστά. Επομένως ικανοποιείται η απαίτηση ο κώδικας να λειτουργεί και για RGB και για gray scale εικόνες. Τέλος όσα pixel στη νέα εικόνα είναι κενά θέτονται μαύρα.

Στο deliverable_3.m εφαρμόζεται η παραπάνω συνάρτηση στην εικόνα im2.jpg για διάφορες επιλογές γωνίας. Σε αυτή την ενότητα δεν χρειάζεται να εφαρμόσουμε κάποιο φίλτρο στην εικόνα εισόδου. Ωστόσο το μέγεθος της προσαρμόζεται και εδώ για λόγους υπολογιστικής ταχύτητας. Τα αποτελέσματα φαίνονται παρακάτω.

1. Στροφή κατά 54◦ × π/180◦ rads

2. Στροφή κατά 213◦ × π/180◦ rads