# Ψηφιακή Επεξεργασία Εικόνας
# Εργασία 2: Hough, Harris, Rot και αποκοπή εικόνων

Πλευρίδη Βασιλική Βαρβάρα (AEM:10454)

Μάϊος 2024

## 1 Εισαγωγή

Στόχος της παρούσας εργασίας είναι ο εντοπισμός των ακμών, των πλευρών αλλά και η περιστροφή μίας εικόνας με την χρήση του μετασχηματισμού Hough, του Harris corner detector και της προσαρμοσμένα κατάλληλα συνάρτησης περιστροφής. Στην συνέχεια, όλες αυτές οι ρουτίνες μπορούν να χρησιμοποιηθούν συνδιαστικά για τον εντοπισμό και την αποκοπή εικόνων από μια άτακτη τοποθέτηση εγγράφων πάνω στην επιφάνεια ενός scanner.

## 2 Hough Transform

Ο αλγόριθμος Hough χρησιμοποιείται για την ανίχνευση γεωμετρικών μορφών, όπως γραμμές, σε μια εικόνα. Η παρακάτω υλοποίηση αποτελεί μια απλή μορφή του μετασχηματισμού αυτού για τον εντοπισμό γραμμών.

### 2.1 Συνάρτηση my_hough_transform

Ο αλγόριθμος λαμβάνει ως είσοδο μια δυαδική εικόνα, στην οποία τα λευκά πιξελς αντιπροσωπεύουν τα αντικείμενα. Επιπλέον, καθορίζονται οι παράμετροι $d\rho$ και $d\theta$, τα βήματα για τις παραμέτρους ρ και ϑ αντίστοιχα, καθώς και ο αριθμός $n$ των κορυφαίων γραμμών που επιθυμούμε να εντοπίσουμε. Στην συνέχεια, δημιουργείται ένας πίνακας $H$ με διαστάσεις που καθορίζονται από τα βήματα $d\rho$ και $d\theta$, προκειμένου να αντιστοιχίσει το χώρο των παραμέτρων (ρ,ϑ).Για κάθε pixel στη δυαδική εικόνα που αντιστοιχεί σε αντικείμενο, δηλαδή που είναι 1, υπολογίζονται οι τιμές των παραμέτρων ρ και ϑ για όλες τις δυνατές γραμμές που μπορεί να διέρχονται από αυτό το pixel. Οι τιμές αυτές ενημερώνουν τον πίνακα H. Αναφορικά με τον εντοπισμό των $n$ κυρίαρχων ευθειών, χρησιμοποιούνται οι έτοιμες συναρτήσεις $np.argpartition$ και $np.column\_stack$ για τον εντοπισμό των θέσεων μεγαλύτερων τιμών στον πίνακα H. Οι δείκτες αυτοί χρησιμοποιούνται εν τέλει για τον υπολογισμό των αντίστχοιων τιμών ρ,ϑ των κυρίαρχων ευθειών, οι οποίες αποθηκεύονται σε έναν επιστραφόμενο πίνακα $L$. Τέλος υπολογίζεται και επιστρέφεται το πλήθος res των σημείων της εικόνας εισόδου που δεν ανήκουν στις $n$ ευθείες που έχουν εντοπιστεί. Να σχολιαστεί ότι αυτή η καταμέτρηση δεν είναι ακριβής καθώς τα σημεία τομής καταμετρόνται πολλαπλά.

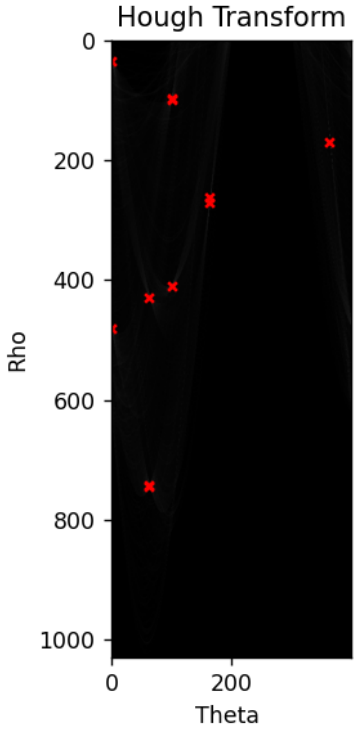
### 2.2 Αρχείο script deliverable_1.py

Στο αρχείο αυτό, ουσιαστικά ελέγχεται ο κώδικας υλοποίησης του μετασχηματισμού Hough, χρησιμοποιώντας ως όρισμα την εικόνα im2.jpg η οποία δίνεται μαζί με την εκφώνηση της εργασίας. Πριν το κάλεσμα της συνάρτησης απαιτείται η μετατροπή της εικόνας σε grayscale εικόνα και στη συνέχεια η κατωφλίωση της με τη βοήθεια ενός edge detector. Σε αυτή την υλοποίηση επιλέχθηκε ο canny και έπειτα από διάφορες δοκιμές επιλέχθηκε ως threshold το 235 και ο χρωματισμός των αντικειμένων που το περνάνε σε άσπρο χρώμα (τιμή 255). Μετά από αυτό, καλείται η υλοποιημένη συνάρτηση και εκτυπώνονται ο πίνακας $H$ με τονισμένες με '$x$' τις κορυφές του πίνακα L αλλά και η αρχική εικόνα εισόδου με αποτυπώμενες τις γραμμές του περιγράμματός των 2 εικόνων που παρουσιάζονται εσωτερικά σε αυτήν. Να σχολιαστεί ότι επιλέχθηκε ο εντοπισμός μόνο αυτών των ευθειών και αγνοήθηκαν οι λεπτομέρειες εσωτερικά των εικόνων για πιο γρήγορη και με λιγότερους θορύβους

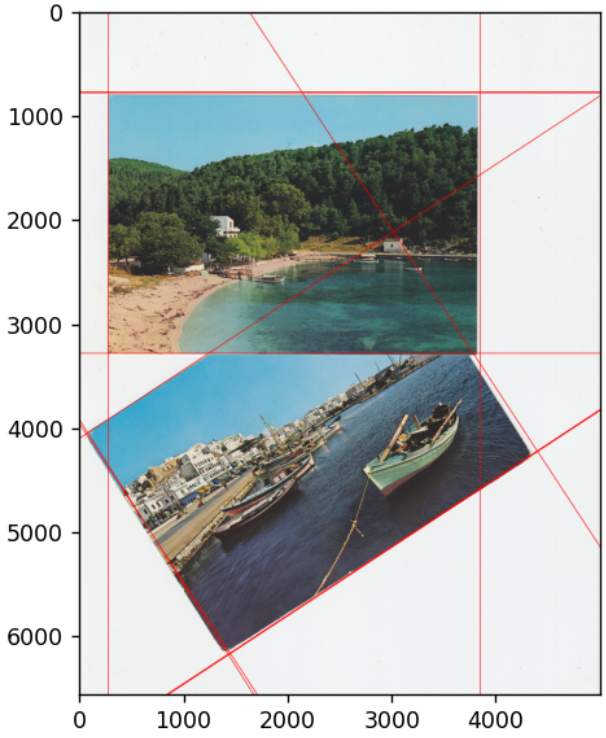
υλοποίηση. Επίσης, με αυτόν τον τρόπο θα χρησιμοποιηθεί και αργότερα στο γενικότερο πρόβλημα της παρούσας εργασίας. Τέλος, σημαντικό είναι να τονιστεί ότι η αρχική εικόνα περικόπηκε ελάχιστα από τις τέρμα αριστερά και πάνω πλευρές έτσι ώστε να αποφευχθεί η ερμηνεία θορύβου ως ευθεία.

### 2.3 Αποτελέσματα αλγορίθμου

Όπως φαίνεται και από τα παρακάτω γραφήματα, ο μετασχηματσιμός φαίνεται να λειτουργεί αρκετά καλά. Να σχολιαστεί ότι παρόλο που οι ζητούμενες ευθείες εντοπισμού ειναι 8, χρειάστηκε να ζητηθούν οι 11 κυρίαρχες ευθείες για να μπορέσουν να εντοπιστούν όλες.




Σχήμα 1: Hough Transform




Σχήμα 2: Αρχική εικόνα με κυρίαρχες ευθείες

## 3 Harris corner detector

Ο αλγόριθμος Harris Corner Detection χρησιμοποιείται για τον αυτόματο εντοπισμό σημείων κορυφής σε μια εικόνα, τα οποία συχνά αντιπροσωπεύουν γωνίες ή σημεία ενδιαφέροντος. Παρακάτω παρουσιάζεται η επεξήγηση του αλγορίθμου καθώς και παρουσιάση των αποτελεσμάτων μέσω του αρχείου deliverable_1.py.

### 3.1 Συνάρτηση my_corner_harris

Η συγκεκριμένη συνάρτηση κατά την υλοποίηση της, ακολουθώντας την θεωρητική ανάλυση που εμπεριέχεται και στην εκφώνηση της εργασίας, ακολουθεί τα παρακάτω βήματα:

1. **Υπολογισμός Παραγώγων:** Χρησιμοποιούνται φίλτρα Sobel για τον υπολογισμό των παραγώγων της εικόνας σε σχέση με τις x και y κατευθύνσεις.
2. **Ασαφή Φίλτρο:** Εφαρμόζεται ασαφή φίλτρο Gaussian στις παραγώγους για μείωση του θορύβου.
3. **Υπολογισμός Πίνακα M:** Υπολογίζονται τα στοιχεία του πίνακα M χρησιμοποιώντας τις παραγώγους. Από αυτόν τον πίνακα υπολογίζονται το determinant και το ίχνος του πίνακα M.

4. **Υπολογισμός Απόκρισης Harris:** Υπολογίζεται η απόκριση Harris χρησιμοποιώντας τον τύπο $R = \det(M) - k \cdot \text{trace}(M)^2$.

5. **Επιστροφή πίνακα R**

### 3.2 Συνάρτηση my_corner_peaks

Η υλοποίηση αυτής της συνάρτησης είναι πολύ απλή καθώς απλά μέσω του κατωφλίου R.max() * rel_threshold ορίζει ποια δείγματα της απόκρισης R θα θεωρούνται οι τελικές θέσεις γωνιών της αρχικής εικόνας, εφαρμόζοντας έναν απλό ανισοτικό έλεγχο στον πίνακα εισόδου R.

### 3.3 Αρχείο script deliverable_2.py

Στο αρχείο αυτό, ουσιαστικά ελέγχεται ο κώδικας υλοποίησης του Harris corner detector, χρησιμοποιώντας ως όρισμα την εικόνα im2.jpg η οποία δίνεται μαζί με την εκφώνηση της εργασίας. Πριν το κάλεσμα της συνάρτησης απαιτείται η μετατροπή της εικόνας σε grayscale εικόνα και στη συνέχεια η κατωφλίωση της όπως και πριν με το threshold=235. Οποιοδήποτε δηλαδή σημείο με χρώμα κάτω από αυτό το όριο, θεωρείται 0. Πριν το κάλεσμα της συνάρτησης my_corner_harris και στην συνέχεια της my_corner_peaks, η εικόνα φιλτραρίστηκε με ένα ασαφή φίλτρο Gaussian για να μειωθεί όσο περισσότερο γίνεται ο θόρυβος. Αφού καλεστούν οι συναρτήσεις, οι γωνίες που προκύπτουν από την εξοδο της συνάρτησης my_corner_peaks απεικονίζονται πάνω στην αρχική εικόνα. Να σχολιαστεί ότι οι διάστασεις του παραθύρου όλων των φίλτρων θεωρήθηκαν ίσες με $round(4 * sigma) + 1$ έτσι ώστε να οδηγούν σε μονούς αριθμούς, όπως και απαιτείται απο την συνάρτηση $cv2.GaussianBlur$ για να χρησιμοποιηθεί.

### 3.4 Αποτελέσματα αλγορίθμου

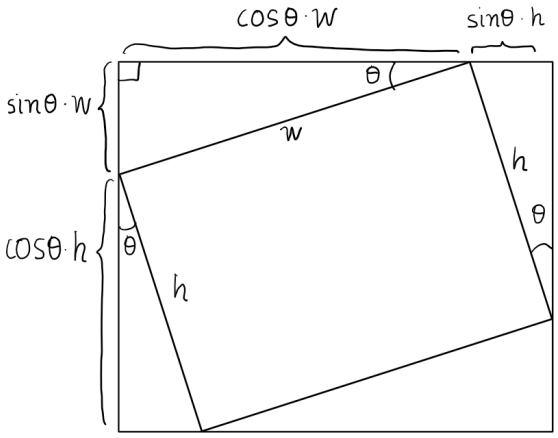
Παρακάτω φαίνεται η εικόνα με ζωγραφισμένες τοπιμέσμες γωνίες των επιμέρους εικόνων:

# 4 Rotation εικόνας

Η ενότητα αυτή αφορά την περιστροφή της εικόνας με βάση το κέντρο αυτής. Παρακάτω θα εξηγηθεί η λογική που χρησιμοποιήθηκε για την υλοποίηση της συνάρτησης υλοποίησης σε python και θα ακολουθήσει η παρουσίαση των αποτελεσμάτων.

## 4.1 Συνάρτηση my_img_rotation

Ο υπολογισμός των διαστάσσεων της νέας εικόνας που θα εμπεριέχει την περιστραμμένη εικόνα βασίζεται στην παρακάτω λογική: και άρα οι νέες διαστάσεις της επιστρεφόμενης εικόνας θα είναι οι:




$$\text{new\_w} = \text{int}(w \cdot |\cos(\theta)| + h \cdot |\sin(\theta)|) \tag{1}$$

$$\text{new\_h} = \text{int}(h \cdot |\cos(\theta)| + w \cdot |\sin(\theta)|) \tag{2}$$

Στην συνέχεια, υπολογίζεται ο πίνακας περιστροφής κατα γωνία theta περί άξονα μεκατεύθυνση που δίνεται από το μοναδιαίο διάνυσμα u. Ο υπολογισμός του γίνεται με την βοήθεια του τύπου του Rodrigues, στον οποίο ο πίνακας R υπολογίζεται από την παρακάτω μαθηματική εξίσωση:

$$R = \begin{bmatrix} (1-\cos a)u_x^2 + \cos a & (1-\cos a)u_x u_y - (\sin a)u_z & (1-\cos a)u_x u_z + (\sin a)u_y \\ (1-\cos a)u_y u_x + (\sin a)u_z & (1-\cos a)u_y^2 + \cos a & (1-\cos a)u_y u_z - (\sin a)u_x \\ (1-\cos a)u_z u_x - (\sin a)u_y & (1-\cos a)u_z u_y + (\sin a)u_x & (1-\cos a)u_z^2 + \cos a \end{bmatrix}$$

Χρησιμοποιώντας λοιπόν ως διάνυσμα κέντρου περιστρφής το υ=[0,0,1] και απλοποίώντας τον πίνακα R σε αυτόν των δύο διαστάσεων, προκύπτει:

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

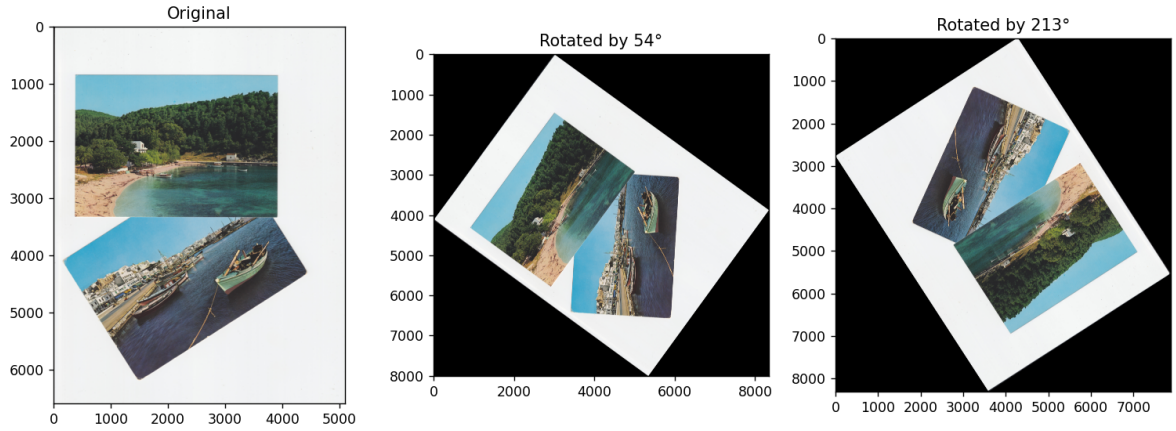
Να σημειωθεί ότι το κέντρο περιστροφής όπως προαναφέρθηκε είναι το (0,0) και έτσι απαιτείται η μεταφορά του κέντρου σε αυτό το σημείο (- new_center) οπού εκεί γίνεται η περιστροφή (*R)και στην συνέχεια η επαναφορά στο αρχικό (+center) για την αντιστοίχηση των pixels της περιστραμμένης με αυτά της αρχικής. Ο χρωματισμός του κάθε pixel,όπως αναφέρεται και από την εκφώνηση , γίνεται με την βοήθεια του μέσου όρου των γειτονικών του. Τέλος, να σχολιαστεί οτι τα pixels γύρω από την περιστραμμένη εικόνα ορίστηκαν μαύρα όπως ζητείται. Ο αλγόριθμος λειτουργεί και για πολύχρωμες αλλά και για grayscale εικόνες.

## 4.2 Αρχείο script deliverable_2.py

Στο αρχείο αυτό,δημιουργείται ένα subplot, στο οποίο παρουσιάζονται οι ζητούμενες περιστραμένες εικόνας συγκριτικά με την αρχική.

### 4.3 Αποτελέσματα αλγορίθμου

Το ζητούμενα γραφήματα παρουσιάζονται παρακάτω:




# 5 Πρόβλημα άτακτης τοποθέτησης εγγράφων πάνω στην επιφάνεια ενός scanner

Η επίλυση αυτού του προβλήματος είναι αρκετά δύσκολη και δυστυχώς η υλοποίηση του δεν ολοκληρώθηκε λόγω χρόνού. Παρόλα αυτά παρακάτω παρουσιάζεται μια σκιαγράφηση της λογικής της: Για να απομονωθούν οι εικόνες αρκεί να βρεθούν οι γωνίες της κάθε μιας. Γνωρίζοντας τα γωνιακά pixels , η αποκοπή τους είναι πολύ εύκολή. Παρόλα αυτά, ενώ υπάρχουν σε λίστα οι γωνίες χάρη στον Harris corner detector, δεν γνωρίζουμε ποιες αντιστοιχούν σε ποια εικόνα. Η σκέψη είναι λοιπόν, με την βοήθεια του Hough Transform , να εντοπιστούν ποιες ακμές περνάνε από ποιες γωνίες(point_on_line,find_lines_for_point).Ξεκινώντας λοιπόν με μία και βρίσκοντας τις άλλες δύο γωνίες εύκολα που ανήκουν σε μια κοινή ευθεία με αυτές και τέλος υπολογίζοντας την τελευταία με βάση τις συντεταγμένες αυτών σκεπτόμενοι ότι θέλουμε να δημιουργήσουμε ορθογώνιο πλαίσιο, πετυχένεται η οριοθέτηση της εικόνας. Να σχολιαστεί ότι δημιουργήθηκε η merge_close_points που αναλαμβάνει να συγχωνεύσει κοντινά σημεία από την λίστα των γωνιών σε ένα σύνολο σημείων, με βάση ένα καθορισμένο κατώφλι απόστασης (threshold),έτσι ώστε να έχουμε μόνο τις 8 γωνίες των εικόνων.

Σίγουρα η παραπάνω λογική δεν καλύπτει όλες τις περιπτώσεις και κατά την υλοποίηση θα παρουσιαζόντουσαν εμπόδια, παρόλα αυτά αποτελεί την βάση επίλυσης του προβλήματος. Μαζί με τα υπόλοιπα αρχεία, εμπεριέχεται και το ημιτελές my_lazy_scanner.py το οποίο υλοποιεί μέχρι ένα σημείο αυτά που προαναφέρθηκαν.