Αναφορά 2ης εργασίας του μαθήματος «Ψηφιακή Επεξεργασία Εικόνας»

Χρυσολόγου Γεώργιος (ΑΕΜ: 10782)

Στα πλαίσια της εργασίας αυτής, κλήθηκα να υλοποιήσω ένα σύνολο συναρτήσεων σε κώδικα γλώσσας Python. Συγκεκριμένα, κατασκεύασα την συνάρτηση ‘’fir_conv’’ μέσω της οποίας πραγματοποιείται συνέλιξη FIR μίας εικόνας με μία δοσμένη μάσκα, τις συναρτήσεις ‘’ sobel_edge’’ και ‘’ log_edge’’ μέσω των οποίων πραγματοποιείται ανίχνευση ακμών με χρήση των τελεστών Sobel και Laplacian of Gaussian αντίστοιχα καθώς και την συνάρτηση ‘’circ_hough’’ μέσω της οποίας πραγματοποιείται ανίχνευση των κύκλων μίας binary εικόνας (τιμή 1 για τα pixel-ακμές και 0 για τα υπόλοιπα).

Περιγραφή λειτουργίας συναρτήσεων

Ως εικόνα εισόδου ορίζεται από την εκφώνηση της εργασίας η εικόνα ‘’basketball_large.png’’, η οποία πρόκειται για μία RGB εικόνα διαστάσεων 1070x1442 (Υψος x Πλάτος). Σε αυτήν, διακρίνεται μία μπάλα μπάσκετ, με μαύρες καμπύλες και μικρά κυκλικά σημάδια που της προσδίδουν υφή, καθώς και δύο ορθογώνιες ετικέτες, μία στα δεξιά της μπάλας και μία στην κάτω αριστερή περιοχή της εικόνας. Η εικόνα εισόδου μετατρέπεται σε ασπρόμαυρη και οι τιμές της φωτεινότητας όλων των pixel της διαιρούνται δια 255. Η τελική κανονικοποιημένη BW εικόνα αποτελεί την εικόνα (δισδιάστατο πίνακα) που εισάγεται ως όρισμα στις συναρτήσεις που υλοποιούνται στα πλαίσια της εργασίας.

fir_conv(**in_img_array:** np.ndarray**, h:** np.ndarray**, in_origin:** np.ndarray**, mask_origin:** np.ndarray)

-> **out_img_array:** np.ndarray**, out_origin:** np.ndarray

Η συνάρτηση fir_conv υλοποιεί τη δισδιάστατη συνέλιξη μιας εικόνας εισόδου με μία μάσκα (kernel) και επιστρέφει την εικόνα εξόδου μαζί με τη νέα θέση της αρχής των αξόνων. Ως είσοδους δέχεται τον δισδιάστατο πίνακα in_img_array που αναπαριστά την κανονικοποιημένη BW εικόνα, τη μάσκα h, καθώς και δύο μονοδιάστατους πίνακες: τον in_origin, που δηλώνει τη θέση της αρχής των αξόνων της εικόνας εισόδου, και τον mask_origin, που δηλώνει τη θέση της αρχής των αξόνων της μάσκας. Κατά την κλήση της συνάρτησης, εφαρμόζεται τεχνική zero-padding στην εικόνα εισόδου, σύμφωνα με το mask_origin, ώστε η μάσκα να μπορεί να εφαρμοστεί σε κάθε σημείο της εικόνας, ακόμα και στα άκρα, χωρίς να υπερβεί τα όριά της. Η εικόνα επεκτείνεται συμμετρικά με γραμμές και στήλες που περιέχουν μηδενικά, με αποτέλεσμα τη δημιουργία ενός επεκταμένου πίνακα in_img_array_padded. Στη συνέχεια, αρχικοποιείται με μηδενικά στοιχεία ένας πίνακας εξόδου out_img_array, ίδιων διαστάσεων με την αρχική εικόνα. Για κάθε θέση (n1, n2) της εικόνας εισόδου, υπολογίζεται η αντίστοιχη θέση (m1, m2) στον επεκταμένο πίνακα, έτσι ώστε η μάσκα να "τοποθετηθεί" με το mask_origin της ευθυγραμμισμένο πάνω από τη θέση αυτή. Η αντίστοιχη περιοχή του in_img_array_padded που καλύπτεται από τη μάσκα πολλαπλασιάζεται στοιχειομετρικά με τα στοιχεία της μάσκας και το άθροισμα των γινομένων καταχωρείται στο αντίστοιχο στοιχείο του πίνακα εξόδου. Η διαδικασία αυτή επαναλαμβάνεται για όλα τα στοιχεία του πίνακα εισόδου. Τέλος, η θέση της αρχής των αξόνων της εικόνας εξόδου υπολογίζεται ως out_origin = in_origin + mask_origin και η συνάρτηση επιστρέφει τον πίνακα-εικόνα εξόδου out_img_array και τον μονοδιάστατο πίνακα out_origin.

sobel_edge(**in_img_array**: np.ndarray, **thres**: float)

-> **out_img_array**: np.ndarray

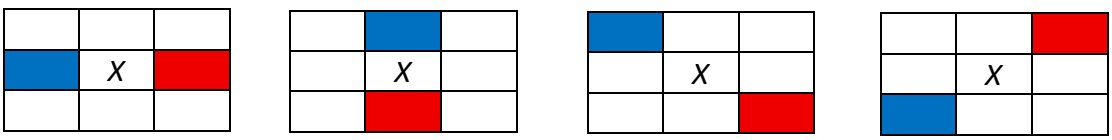
Η συνάρτηση sobel_edge πραγματοποιεί ανίχνευση των ακμών της εικόνας εισόδου μέσω της χρήσης του τελεστή Sobel. Ως εισόδους δέχεται τον δισδιάστατο πίνακα in_img_array που αναπαριστά την κανονικοποιημένη BW εικόνα και μία float τιμή thres η οποία ορίζει το όριο της τιμής των pixels της εικόνας που προκύπτει από την εφαρμογή του τελεστή Sobel στην εικόνα εισόδου, πάνω από το οποίο το pixel θα θεωρείται pixel ακμής. Κατά την κλήση της συνάρτησης, δημιουργούνται οι δύο μάσκες G_{x1} και G_{x2}, σύμφωνα με την εκφώνηση. Κατόπιν, ορίζεται η θέση (0,0) ως αρχή των αξόνων της εικόνας εισόδου, η θέση (1,1) ως αρχή των αξόνων των μασκών (αποτελεί το κέντρο των μασκών) και πραγματοποιείται συνέλιξη της εικόνας εισόδου με καθεμία από τις δύο μάσκες. Τα στοιχεία των δύο πινάκων που προκύπτουν υψώνονται στο τετράγωνο, αθροίζονται στοιχείο προς στοιχείο και εφαρμόζεται τετραγωνική ρίζα στα στοιχεία του πίνακα που προκύπτει από την προηγούμενη πρόσθεση, με αποτέλεσμα την διαμόρφωση του πίνακα g. Τέλος, αρχικοποιείται με μηδενικά στοιχεία ένας πίνακας εξόδου out_img_array, ίδιων διαστάσεων με την αρχική εικόνα (και τον πίνακα g) και διατρέχονται όλα τα στοιχεία του πίνακα g. Για ένα συγκεκριμένο στοιχείο, αν η τιμή του είναι μεγαλύτερη από την float τιμή thres, το στοιχείο του πίνακα εξόδου out_img_array της ίδιας θέσης αποκτά την τιμή 1, δηλαδή θεωρείται pixel ακμής, αλλιώς παραμένει με την τιμή 0. Η διαδικασία αυτή επαναλαμβάνεται για όλα τα στοιχεία του πίνακα g και τελικά επιστρέφεται ο πίνακα εξόδου out_img_array.

log_edge(**in_img_array**: np.ndarray, **thres**: float, **mode**: int)

-> **out_img_array**: np.ndarray

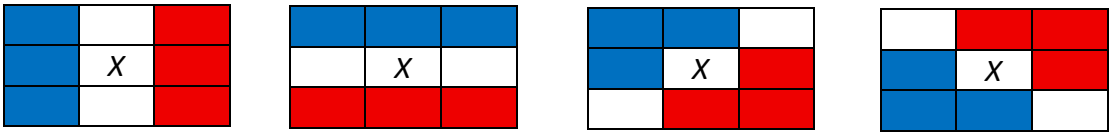
Η συνάρτηση log_edge πραγματοποιεί ανίχνευση των ακμών της εικόνας εισόδου μέσω της χρήσης του τελεστή Laplacian of Gaussian. Ως εισόδο δέχεται τον δισδιάστατο πίνακα in_img_array που αναπαριστά την κανονικοποιημένη BW εικόνα. **Πέρα από τις υποδείξεις της εκφώνησης, έχουν προστεθεί ως είσοδοι στην συνάρτηση μία float τιμή thres η οποία ορίζει το όριο της τιμής των pixels της εικόνας που προκύπτει από την εφαρμογή του τελεστή LoG στην εικόνα εισόδου, πάνω από το οποίο το pixel θα θεωρείται pixel ακμής και μία int τιμή mode η οποία ορίζει τον τρόπο με τον οποίο θα πραγματοποιηθεί ο έλεγχος zero-crossing**. Κατά την κλήση της συνάρτησης, δημιουργείται η 5x5 μάσκα που ορίζεται στην εκφώνηση. Κατόπιν, ορίζεται η θέση (0,0) ως αρχή των αξόνων της εικόνας εισόδου, η θέση (2,2) ως αρχή των αξόνων της μάσκας (αποτελεί το κέντρο της μάσκας) και πραγματοποιείται συνέλιξη της εικόνας εισόδου με την μάσκα, με αποτέλεσμα την διαμόρφωση του πίνακα-εικόνας conv_img. Στην συνέχεια, αρχικοποιείται με μηδενικά στοιχεία ένας πίνακας εξόδου out_img_array, ίδιων διαστάσεων με την αρχική εικόνα (και τον πίνακα conv_img). Ακολουθεί ο έλεγχος για την εύρεση των σημειών zero-crossings. Παρά την αρχική υλοποίηση των πρώτων δύο τρόπων (mode= 1 και mode=2), η αδυναμία τους να επιτύχουν ικανοποιητικά αποτελέσματα κατά τον συνδυασμό τους με την μέθοδο circ_hough για την εύρεση κύκλων στην εικόνα εισόδου οδήγησε στον σχεδιασμό και του τρίτου τρόπου ελέγχου για σημεία zero-crossings (mode=3). Αναλυτικά:

mode=1: Μέσω μίας διπλής for η οποία διατρέχει όλα τα στοιχεία του πίνακα conv_img και πολλαπλών εντολών if, ελέγχεται αν ένα grid 3x3, με το pixel που εξετάζεται να βρίσκεται στο κέντρο, εμφανίζει κάποιο από τα παρακάτω μοτίβα:

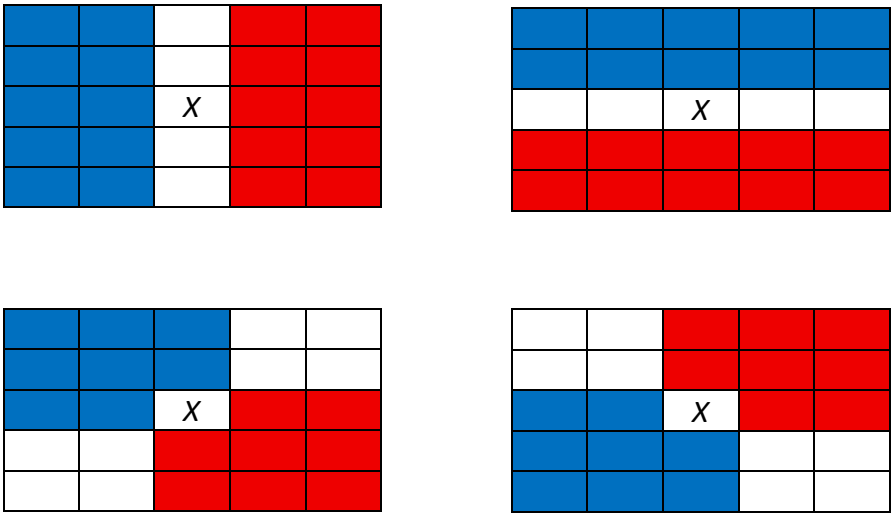


Για την παραπάνω περίπτωση (mode=1), ελέγχεται αν εμφανίζεται οποιοδήποτε από τα παραπάνω μοτίβα όπου το μπλε και το κόκκινο pixel έχουν διαφορετικά πρόσημα, και η απόλυτη τιμή της διαφοράς των τιμών των δύο αυτών pixels είναι μεγαλύτερη από την float τιμή thres. Αν ο έλεγχος επαληθευτεί για κάποιο από τα μοτίβα, το στοιχείο του πίνακα εξόδου out_img_array, το οποίο βρίσκεται στην ίδια θέση με το στοιχείο του πίνακα conv_img που ελέγχεται, αποκτά την τιμή 1, δηλαδή θεωρείται pixel ακμής.

mode=2: Μέσω μίας διπλής for η οποία διατρέχει όλα τα στοιχεία του πίνακα conv_img και πολλαπλών εντολών if, ελέγχεται αν ένα grid 3x3, με το pixel που εξετάζεται να βρίσκεται στο κέντρο, εμφανίζει κάποιο από τα παρακάτω μοτίβα:



mode=3: Μέσω μίας διπλής for η οποία διατρέχει όλα τα στοιχεία του πίνακα conv_img και πολλαπλών εντολών if, ελέγχεται αν ένα grid 5x5, με το pixel που εξετάζεται να βρίσκεται στο κέντρο, εμφανίζει κάποιο από τα παρακάτω μοτίβα:



Για τις δύο παραπάνω περιπτώσεις (mode=2 και mode=3) ελέγχεται αν εμφανίζεται οποιοδήποτε από τα παραπάνω μοτίβα όπου τα μπλε pixels έχουν ίδιο πρόσημο, τα κόκκινα pixels έχουν ίδιο πρόσημο αλλά τα μπλε και τα κόκκινα έχουν διαφορετικά μεταξύ τους πρόσημα. Επιπλέον, εξετάζεται αν η απόλυτη τιμή της διαφοράς του μέσου όρου των τιμών των μπλε pixels μείον του μέσου όρου των τιμών των κόκκινων pixels είναι μεγαλύτερη από την float τιμή thres. Αν ο έλεγχος επαληθευτεί για κάποιο από τα μοτίβα, το στοιχείο του πίνακα εξόδου out_img_array, το οποίο βρίσκεται στην ίδια θέση με το στοιχείο του πίνακα conv_img που ελέγχεται, αποκτά την τιμή 1, δηλαδή θεωρείται pixel ακμής.

Με καθέναν από τους παραπάνω τρεις τρόπους, εξετάζεται για ένα συγκεκριμένο pixel αν οι φωτεινότητες των γειτονικών pixels εμφανίζουν μεταξύ τους διαφορές, με συνέπεια την αυξημένη πιθανότητα το pixel αυτό να αποτελεί pixel ακμής. Η παραπάνω διαδικασία, για όποια από τις τρεις τιμές mode επιλεχθεί, επαναλαμβάνεται για όλα τα pixels του πίνακα-εικόνας conv_img και τελικά επιστρέφεται η εικόνα εξόδου out_img_array.

circ_hough(**in_img_array**: np.ndarray, **R_max**: float, **dim**: np.ndarray, **V_min**: int)

-> **centers**: np.ndarray, **radii**: np.ndarray

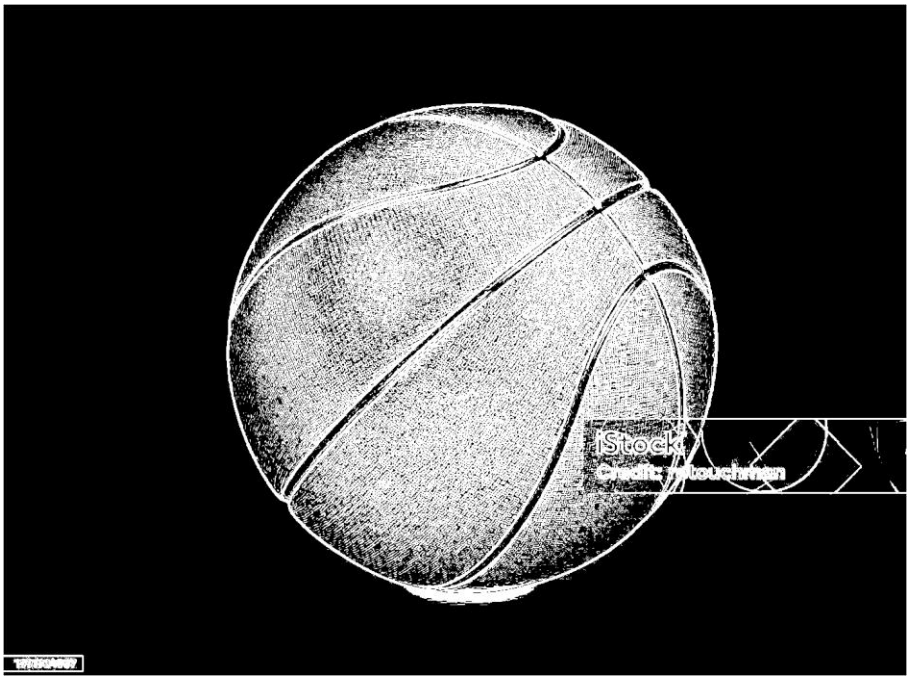
Η συνάρτηση circ_hough πραγματοποιεί ανίχνευση των κύκλων μίας binary εικόνας με τιμές 1 για τα pixels ακμών και 0 για τα υπόλοιπα. Ως εισόδους δέχεται τον δισδιάστατο πίνακα in_img_array, ο οποίος αναπαριστά μία binary εικόνα όπως περιγράφεται παραπάνω (πίνακας εξόδου μίας εκ των μεθόδων sobel_edge ή log_edge), μία float τιμή R_max η οποία αποτελεί την μέγιστη ενδεχόμενη ακτίνα κύκλου που αναζητείται από την μέθοδο, έναν μονοδιάστατο πίνακα dim που περιέχει το πλήθος των διαμερίσεων της οριζόντιας και της κατακόρυφης διάστασης της εικόνας εισόδου καθώς και του διαστήματος [R_min, R_max] **(πέρα από τις υποδείξεις τις εκφώνησεις, ως R_min ορίζεται η τιμή 50 για την αποφυγή ανίχνευσης μικρών κύκλων που προκύπτουν λόγω της υφής που παρουσιάζεται στην εικόνα εισόδου)** και μία int τιμή V_min η οποία ορίζει το όριο ελαχίστων ψήφων πάνω από τις οποίες ένας υποψήφιος κύκλος τελικά ανιχνεύεται από την μέθοδο. Κατά την κλήση της μεθόδου, αρχικοποιείται με μηδενικά στοιχεία ένας τρισδιάστατος πίνακας ψηφοφορίας Votes_array με διαστάσεις που ορίζονται από τις τιμές των στοιχείων του πίνακα dim. Ένα στοιχείο Votes_array[i][j][k] = L υποδεικνύει ότι ο κύκλος με κέντρο (x,y)=(i, j) και ακτίνα R=k έχει συλλέξει L ψήφους, δηλαδή διέρχεται από L pixels ακμών. Επιπλέον, υπολογίζονται διακριτές θέσεις pixels και τιμές ακτινών με βάση τις διαμερίσεις που ορίζει ο πίνακας dim στην οριζόντια και την κατακόρυφη διάσταση της binary εικόνας εισόδου καθώς και στο διάστημα [R_min, R_max]. Κατόπιν, μέσω μίας διπλής for, διατρέχεται κάθε στοιχείο του πίνακα-εικόνας εισόδου in_img_array. Για ένα συγκεκριμένο στοιχείο-pixel (i, j), ελέγχεται αν αποτελεί pixel ακμής, και αν αυτό επαληθευτεί, υπολογίζονται για κάθε τιμή ακτίνας R τα στοιχεία-pixel που απέχουν από το στοιχείο-pixel αυτό απόσταση R, προς κατευθύνσεις που ορίζουν 72 διαφορετικές γωνίες από 0 έως 2π. Κάθε τέτοιο στοιχείο-pixel που προκύπτει αποτελεί υποψήφιο κέντρο κύκλου, ελέγχεται αν βρίσκεται εντός των διαστάσεων του πίνακα-εικόνας και αντιστοιχίζεται στην πλησιέστερη διακριτή θέση pixel από αυτές που προέκυψαν λόγω της διαμέρισης των διαστάσεων της εικόνας. Τελικά, προστίθεται μία ψήφος σε κάθε στοιχείο Votes_array στο οποίο αντιστοιχήθηκε κάποιο υποψήφιο κέντρο εντός των διαστάσεων της εικόνας, με την αντίστοιχη ακτίνα. Τέλος, ελέγχονται οι τιμές των στοιχείων του διαμορφωμένου, πλέον, πίνακα ψηφοφορίας Votes_array, και για αυτά που έχουν τιμή μεγαλύτερη ίση με V_min, οι θέσεις των πρώτων δύο διαστάσεων (οριζόντια και κατακόρυφη θέση του pixel που αποτελεί κέντρο κύκλου) αποθηκεύονται στον πίνακα centers ενώ η θέση της τρίτης διάστασης (τιμή ακτίνας) αποθηκεύεται στον πίνακα radii. Οι δύο πίνακες centers και radii επιστρέφονται.

Παρουσίαση αποτελεσμάτων

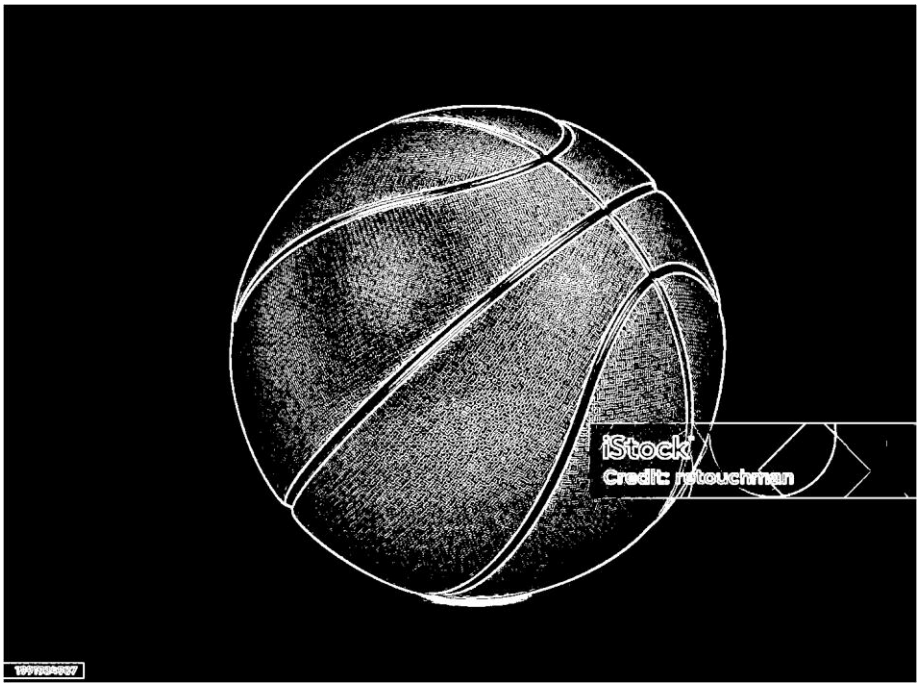
Παρακάτω παρουσιάζονται και σχολιάζονται τα απαιτούμενα αποτελέσματα για τις μεθόδους sobel_edge, log_edge και circ_hough.

sobel_edge:

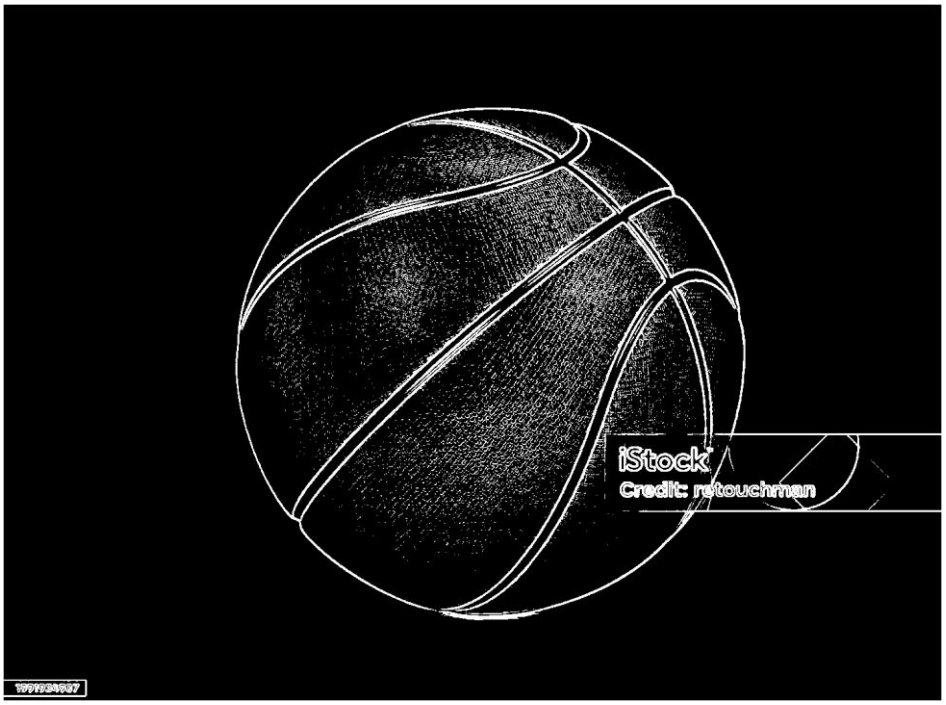
thres = 0.1



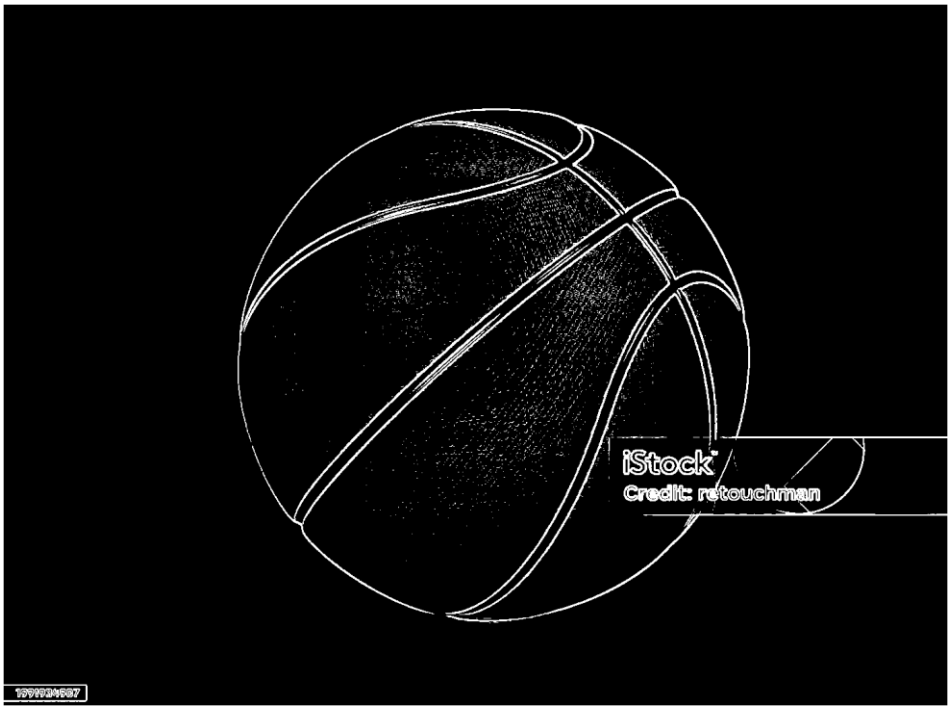
thres = 0.2



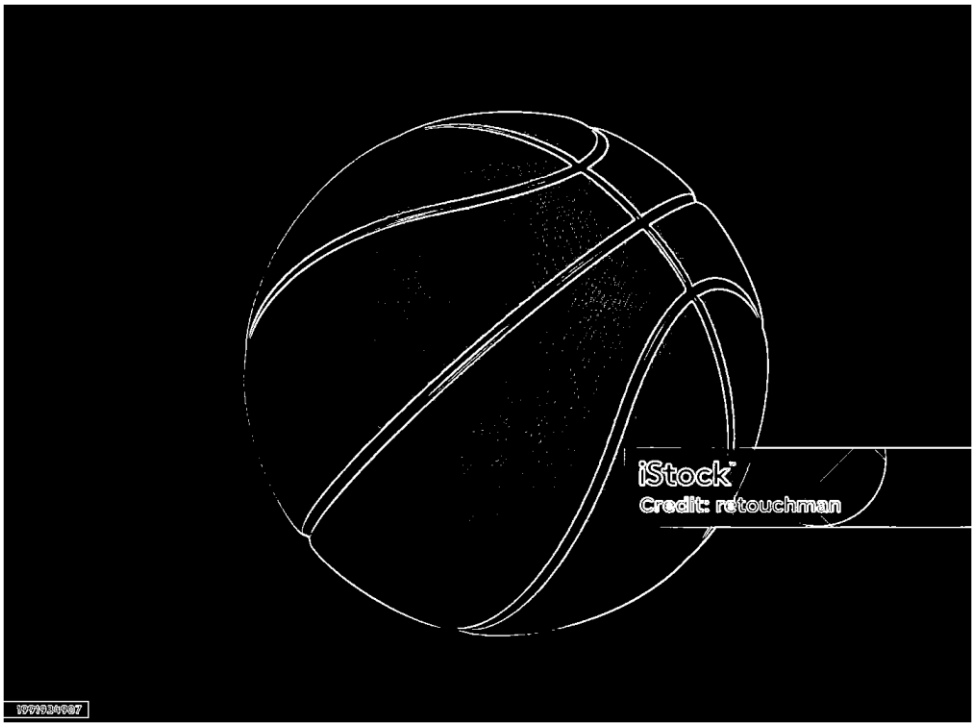
thres = 0.3



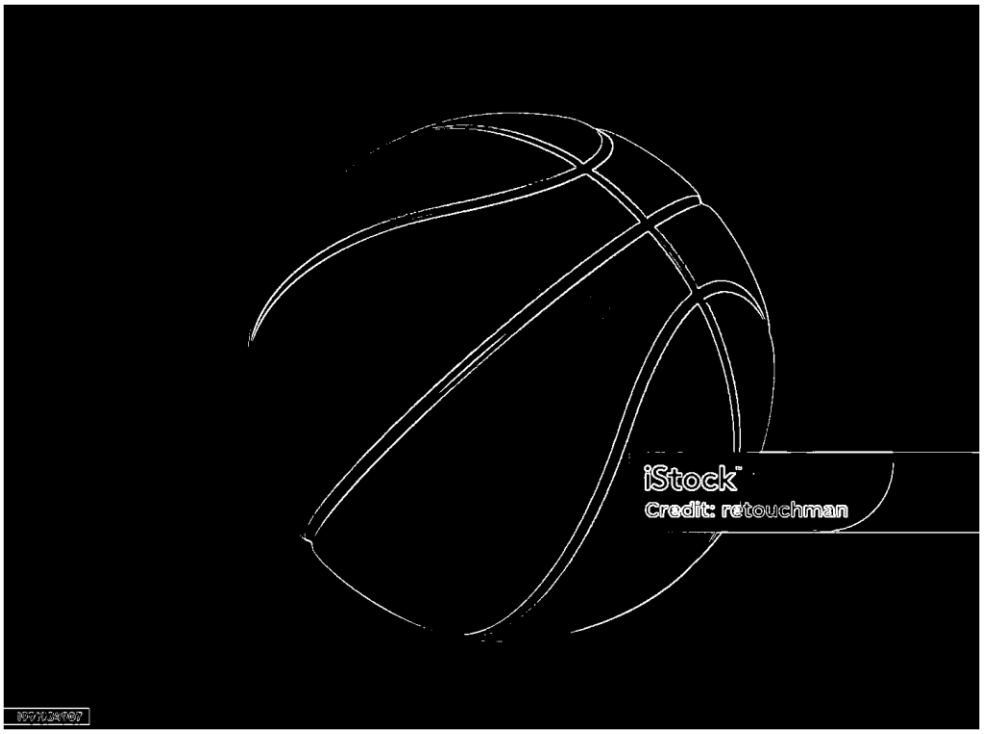
thres = 0.4



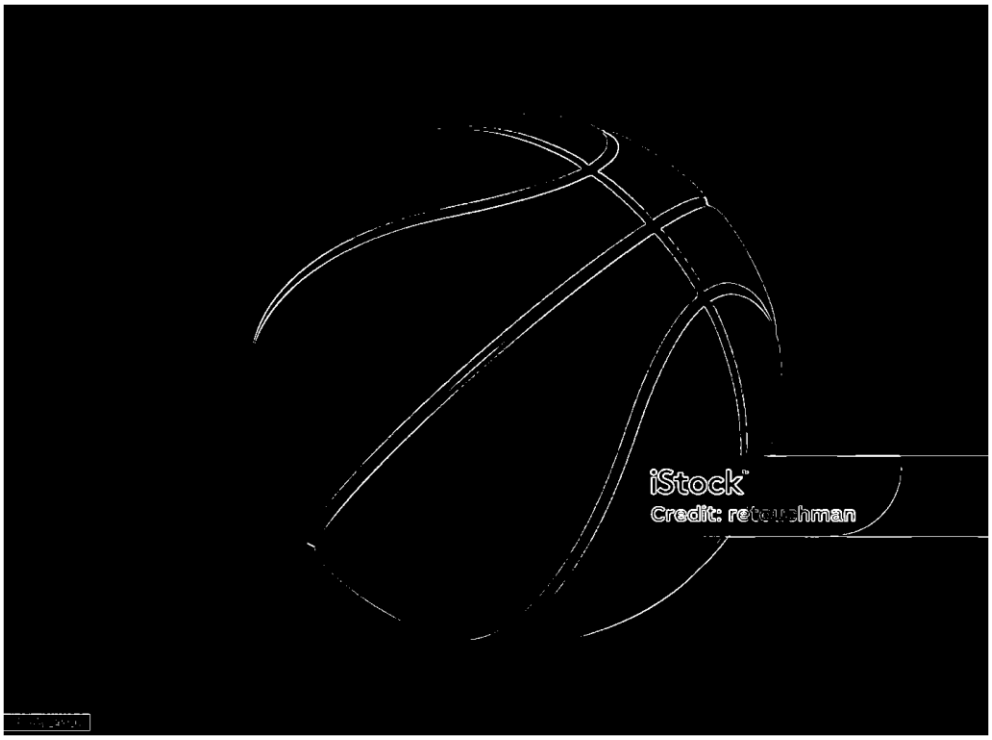
thres = 0.5



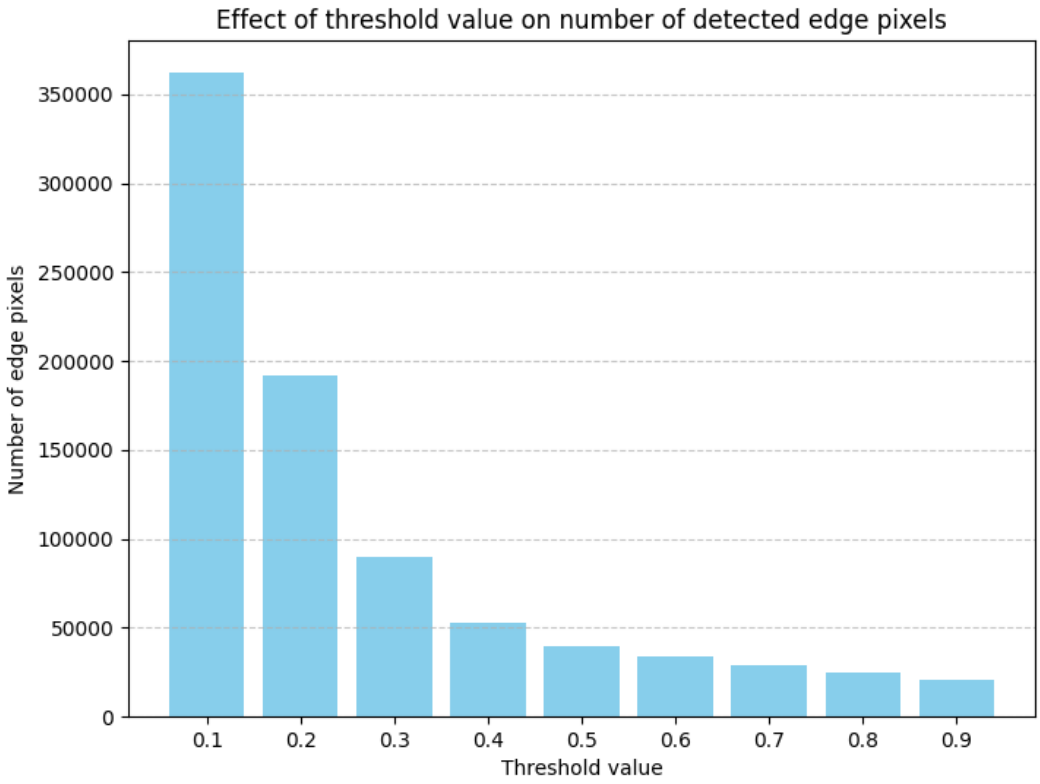
thres = 0.7



thres = 0.9



Ραβδόγραμμα πλήθους ανιχνευμένων pixel ακμών προς τιμή threshold



Είναι εμφανές από το το ραβδόγραμμα ότι καθώς αυξάνεται η τιμή του threshold, το πλήθος των pixels που θεωρούνται pixels ακμών μειώνεται. Η μείωση αυτή είναι αρχικά μεγάλη και σταδιακά γίνεται ολοένα πιο μικρή.

Για thres=0.1, παρατηρείται ότι εντοπίζονται επιτυχώς οι ακμές του περιγράμματος της μπάλας και των μαύρων καμπυλών της. Ανιχνεύονται, επιπλέον, οι ακμές των δύο ετικετών που διακρίνονται στην φωτογραφία. Ωστόσο, η ευαισθησία της μεθόδου για τόσο μικρή τιμή threshold έχει ως συνέπεια την παρερμηνεία της ύφης της μπάλας, των ‘’σπυριών’’ της, ως ακμές, λόγω της μικρής αλλαγής φωτεινότητας στις περιοχές αυτές. Ακόμη, ως ακμή θεωρείται και η σκιά της μπάλας στο χαμηλότερο τμήμα της εικόνας.

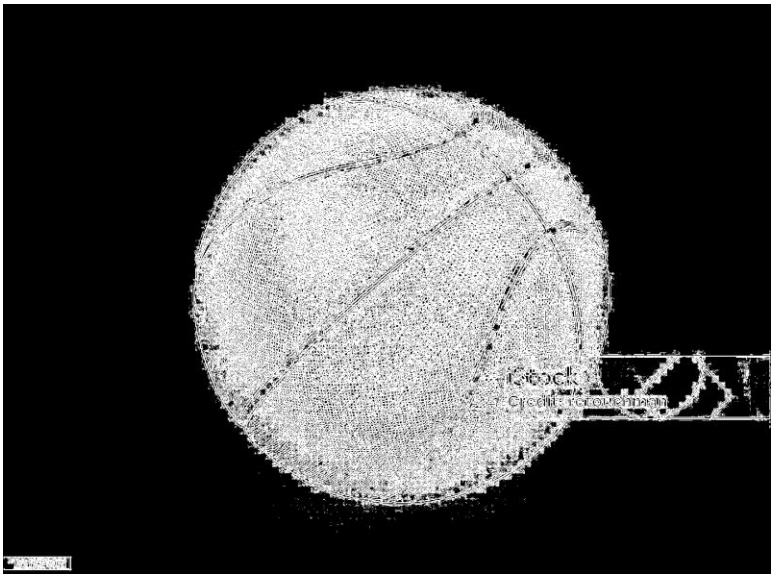
Με την αύξηση της τιμής του threshold, γίνεται αντιληπτή η αποτελεσματικότητα της μεθόδου. Για thres = 0.3 και 0.4, το περίγραμμα, οι μαύρες καμπύλες της μπάλας και οι χαρακτήρες (γράμματα, ψηφία) των ετικετών ανιχνεύονται επιτυχώς ενώ ο ‘’θόρυβος’’ που προκαλείται από την υφή της μπάλας είναι σημαντικά μειωμένος.

Εντούτοις, για τιμές thres = 0.5 και πάνω, παρά την σχεδόν εξ’ ολοκλήρου εξάλειψη του θορύβου της υφής, η μέθοδος αδυνατεί να ανιχνεύσει τμήματα του περιγράμματος και, σταδιακά, και των καμπυλών της μπάλας, ενώ η διάκριση των χαρακτήρων των ετικετών καθίσταται προβληματική. Για τιμή thres = 0.9, δεν είναι έυκολο να αντιληφθεί κανείς ότι η αρχική εικόνα απεικονίζει μία μπάλα.

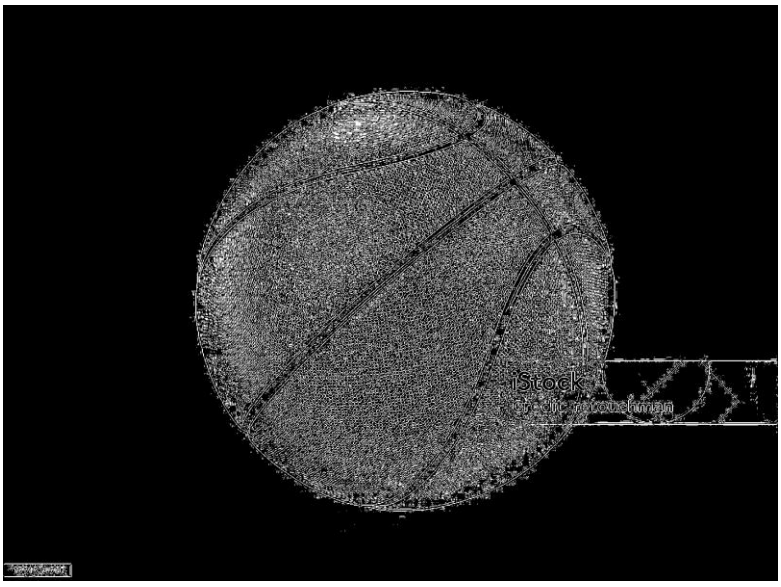
Λαμβάνοντας υπ’ όψιν τις παραπάνω παρατηρήσεις, συμπεραίνεται ότι η τιμή thres=0.4 αποδεικνύεται η καταλληλότερη μεταξύ όσων δοκιμάστηκαν, καθώς με αυτήν η μέθοδος συνδυάζει ανίχνευση όλων των επιθυμητών ακμών, ενώ αποφεύγει την παρερμηνεία των περιοχών που η μπάλα παρουσιάζει υφή ως ακμές.

log_edge:

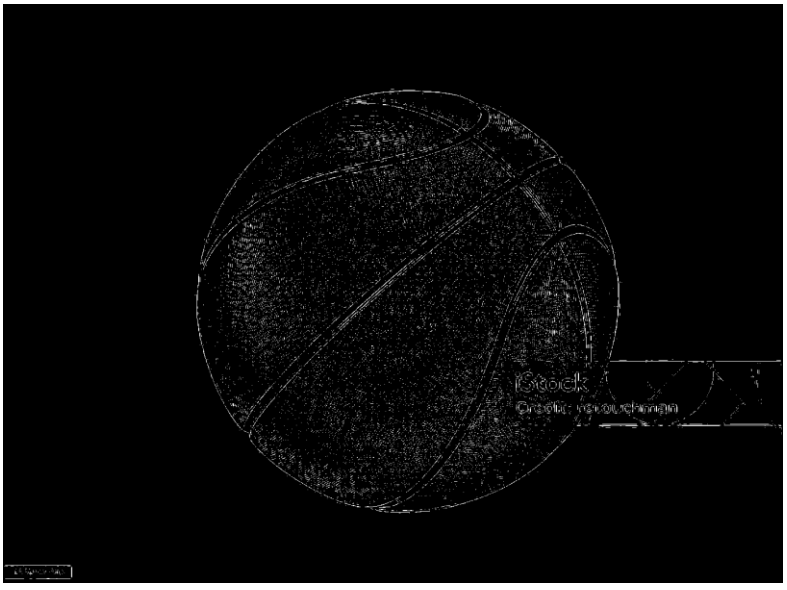
thres = 0.1, mode=1



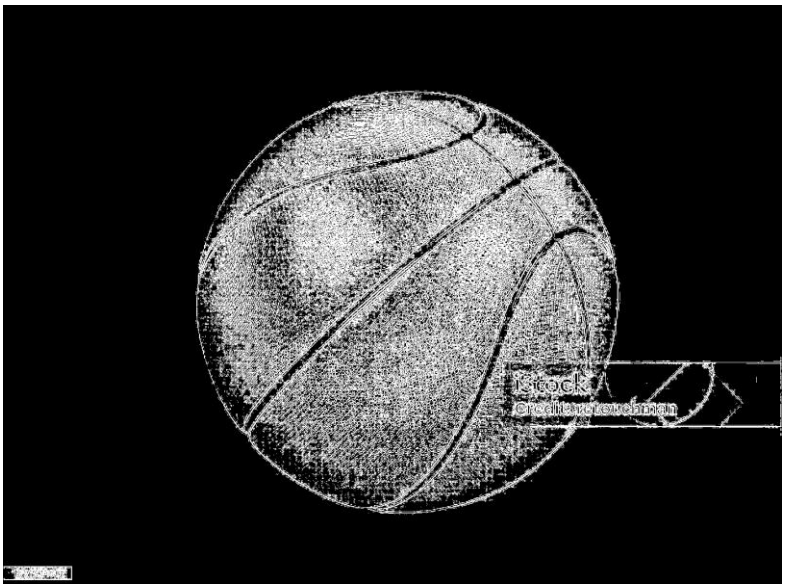
thres = 0.1, mode = 2



thres = 0.1, mode = 3

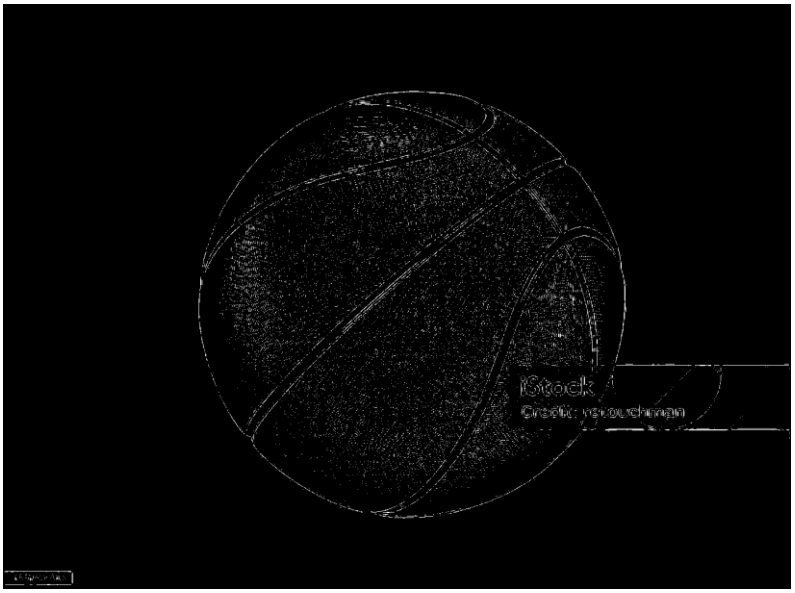


thres = 0.3, mode=1

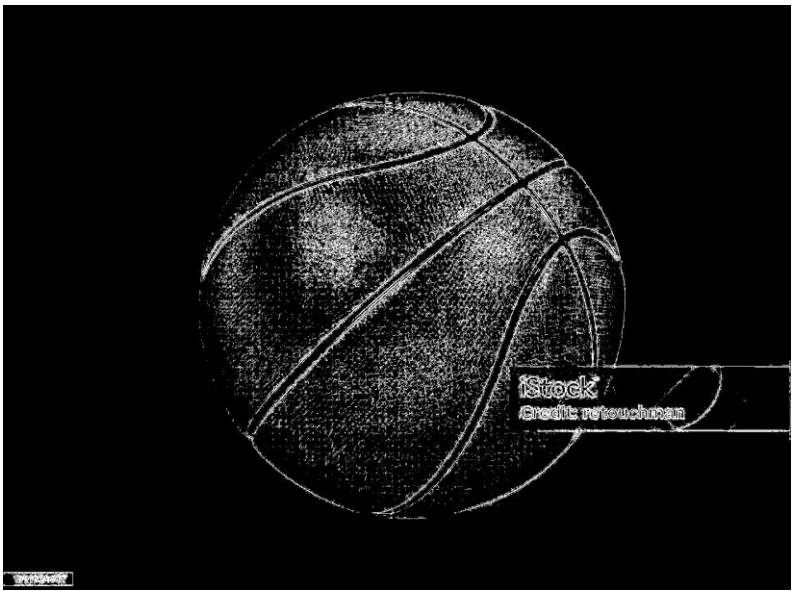


thres = 0.3, mode = 2

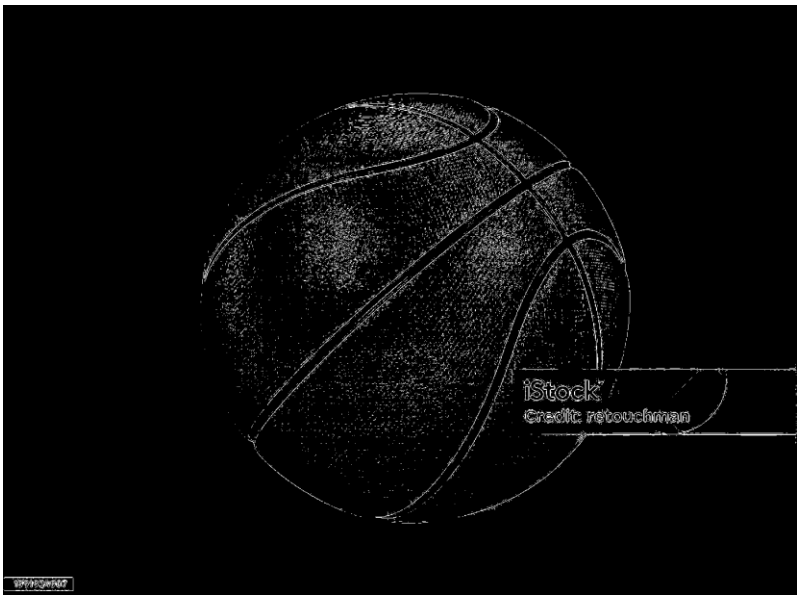
thres = 0.3, mode = 3



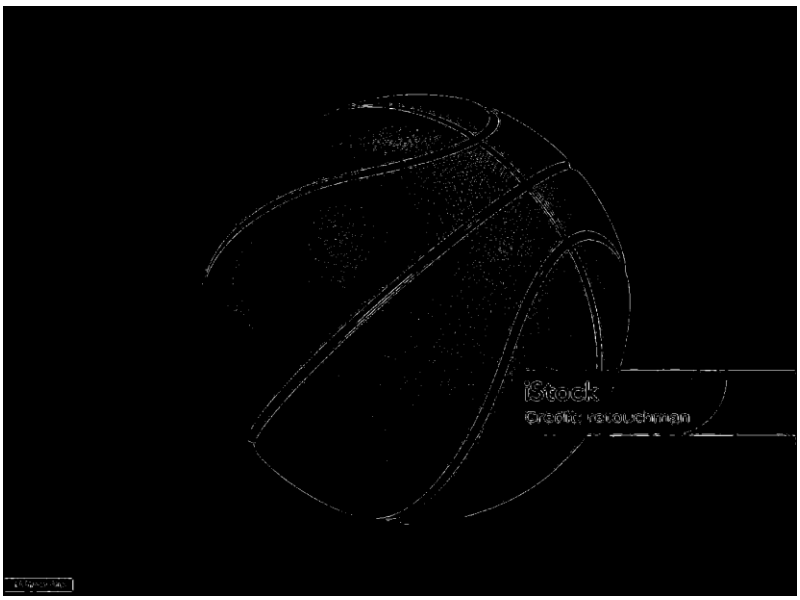
thres = 0.6, mode=1



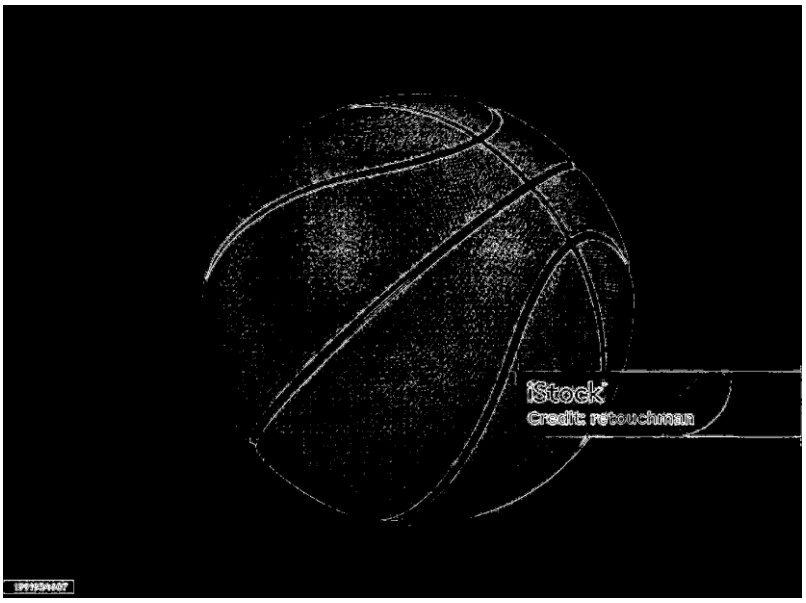
thres = 0.6, mode = 2



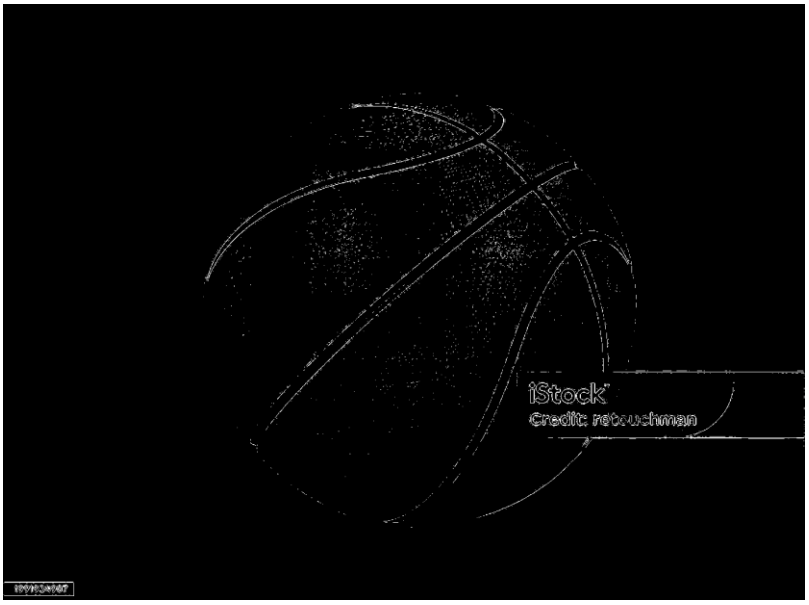
thres = 0.6, mode = 3



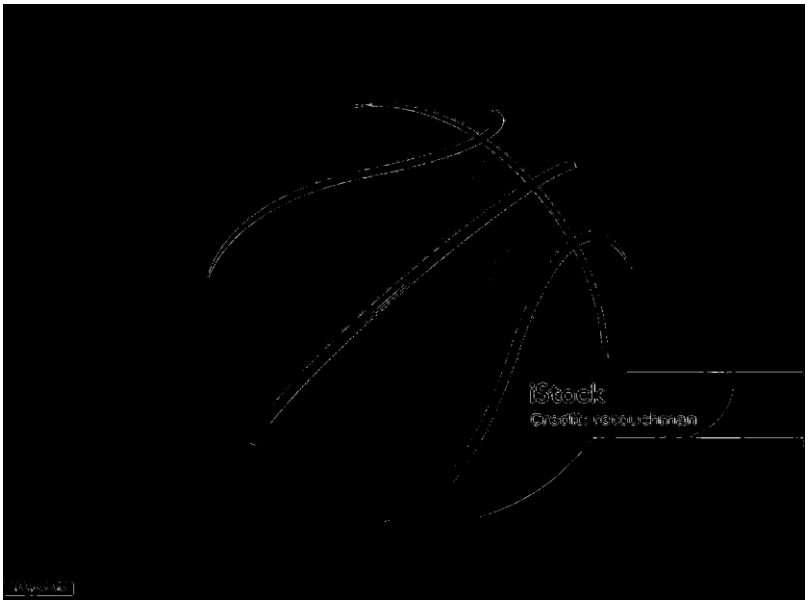
thres = 0.9, mode=1



thres = 0.9, mode = 2

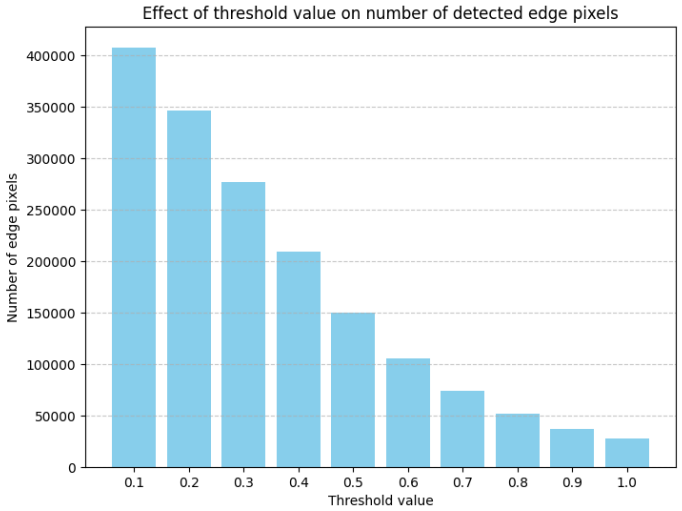


thres = 0.9, mode = 3

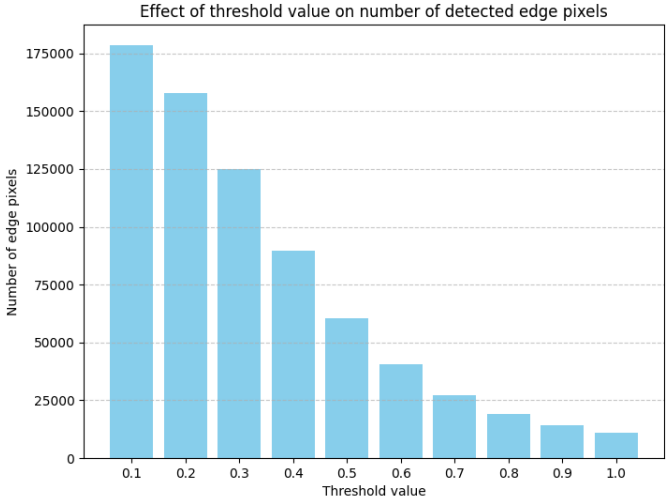


Ραβδογράμματα πλήθους ανιχνευμένων pixel ακμών προς τιμή threshold για καθένα από τα τρία modes

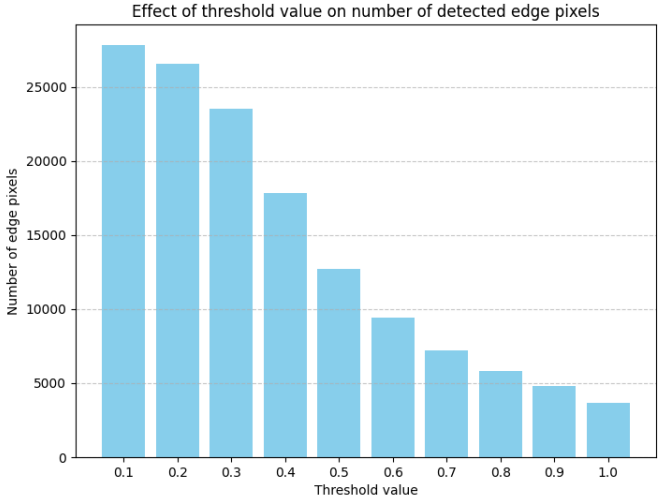
mode = 1



mode = 2



mode = 3



Όπως και στην περίπτωση της μεθόδου sobel_edge, από την παρατήρηση των ραβδογραμμάτων γίνεται αντιληπτό ότι η αύξηση του τιμής threshold οδηγεί σε μείωση του πλήθους των pixels που θεωρούνται pixels ακμών και για τα 3 modes ελέγχου για σημεία zero-crossing. Σε αντίθεση, όμως, με την μέθοδο sobel_edge, στην μέθοδο log_edge παρατηρείται ομαλότερη σχέση μεταξύ των δύο τιμών (εδώ η αύξηση του threshold, όταν βρίσκεται σε χαμηλές τιμές, δεν οδηγεί σε τόσο ραγδαία μείωση των ανιχνευμένων pixel ακμών σε σχέση με την sobel_edge).

Συγκρίνοντας τα αποτελέσματα και των τριών modes της μεθόδου με αυτά της sobel_edge, παρατηρούνται σημαντικά περισσότερα ανιχνευμένα pixel ακμών πάνω στα ‘’σπυριά’’ της μπάλας, ειδικά όσον αφορά τα modes 1 και 2. Ως συμπέρασμα, η ανίχνευση ακμών με χρήση του τελεστή LoG αποδεικνύεται ιδιαίτερα πιο ευαίσθητη σε μικρές αλλαγές της φωτεινότητας στην εικόνα σε σχέση με χρήση του τελεστή Sobel. Σημαντικό είναι, επιπλέον, να αναφερθεί ότι καθώς η τιμή του threshold αυξάνεται και ο ‘’θόρυβος’’ που προκαλεί η πλούσια υφή της μπάλας μειώνεται, μεγάλη είναι και η μείωση του πλήθους των ανιχνευμένων pixel ακμών που αποτελούν τις επιθυμητές ακμές που αναζητούνται, με αποτέλεσμα οι τελευταίες να μην είναι τόσο ευδιάκριτες όσο με χρήση της μεθόδου sobel_edge. Σημειώνεται, επίσης, ότι σε καμία από τις δοκιμές με την μέθοδο log_edge δεν επιτεύχθηκε ανίχνευση ακμών που να οδηγήσει σε ευδιάκριτους χαρακτήρες των ετικετών της εικόνας.

Μεταξύ των τριών modes, είναι εμφανές τόσο από τα μοτίβα που αναζητόνται κατά την υλοποίηση τους σε κώδικα όσο και από τα αποτελέσματα που αποφέρουν, ότι το mode 3 αποτελεί την αυστηρότερη προσέγγιση ελέγχου για σημεία zero-crossings, ενώ ακολουθεί το mode 2. Το συμπέρασμα αυτό επαληθεύεται και από τα ραβδογράμματα που αντιστοιχούν στα 3 modes, στα οποία γίνεται αντιληπτό ότι όσο πιο σύνθετα είναι τα μοτίβα που αναζητά η μέθοδος για να αποφασίσει αν ένα pixel αποτελεί σημείο zero-crossing, τόσο μικρότερο είναι το πλήθος των pixel ακμών που τελικά ανιχνεύονται. Συγκεκριμένα, το mode 1 οδηγεί στην ανίχνευση πολλών περισσότερων pixel ακμών σε σύγκριση με τα άλλα δύο, ενώ το mode 2, με την σειρά του, οδηγεί σε binary εικόνα με πόλλα περισσότερα pixel ακμές σε σχέση με το mode 3.

Για τιμή threshold = 0.1, παρατηρείται ότι τα modes 1 και 2 οδηγούν σε απογοητευτικά αποτελέσματα, καθώς υπερβολικός αριθμός pixels θεωρούνται pixels ακμών, σε αντίθεση με το mode 3 από το οποίο προκύπτει μία ικανοποιητική binary εικόνα ακμών, αφού διακρίνεται εύκολα το περίγραμμα της μπάλας και οι καμπύλες της.

Με αύξηση του threshold, οι binary εικόνες που προκύπτουν και από τα 3 modes παρουσιάζουν λιγότερο ‘’θόρυβο’’. Ωστόσο, όταν η τιμή του threshold αγγίζει την τιμή 0.6, παρατηρείται απώλεια τμημάτων στις επιθυμητές ακμές των binary εικόνων. Περαιτέρω αύξηση του threshold οδηγεί σε ακόμη χειρότερα αποτελέσματα, με το περίγραμμα της μπάλας να μην είναι εύκολα διακριτό.

circ_hough:

Μοναδικό κύκλο που εντοπίζεται στην εικόνα αποτελεί το περίγραμμα της μπάλας. Αυτός είναι ο κύκλος που καλείται να ανιχνεύσει η μέθοδος circ_hough. Έπειτα από ποικιλία διαφορετικών δοκιμών, κατέστη αντιληπτό ότι η απόδοση της μεθόδου αυτής εξαρτάται σε σημαντικό βαθμό από τις τιμές των εισόδων dim, R_max και V_min.

Πιο συγκεριμένα, ανάλογα με τις διαμερίσεις των διαστάσεων της εικόνας και του εύρους τιμών της ακτίνας R, οι οποίες ορίζονται από τον πίνακα dim, επηρεάζεται τόσο η ακρίβεια όσο και η ταχύτητα της μεθόδου. Για λίγες διαμερίσεις, ενώ η μέθοδος δεν απαιτεί σημαντικό χρόνο για την ολοκλήρωση της, είναι πιθανό να μην καταλήξει στην ανίχνευση του κύκλου με ικανοποιητική ακρίβεια. Όσον αφορά την μεταβλητή R_max, με επιλογή μικρής τιμής ενδέχεται η μέθοδος να αναζητήσει αποκλειστικά κύκλους μικρότερης ακτίνας από την πραγματική ακτίνα του περιγράμματος της μπάλας. Αντίθετα, ο ορισμός μίας μεγάλης τιμής στην R_max, σε συνδυασμό με ατυχή επιλογή των διαμερίσεων του εύρους τιμών της ακτίνας, οδηγεί σε κίνδυνο η τιμή της ακτίνας του επιθυμητού κύκλου να μην εξεταστεί ποτέ, παρά να πραγματοποιηθεί έλεγχος μόνο για ακτίνες που δεν έχουν παρόμοια τιμή με την ζητούμενη. Τέλος, σημαντική είναι και η επιλογή κατάλληλου V_min, καθώς μικρή τιμή συχνά οδηγεί σε ανίχνευση ανεπιθύμητων κύκλων ενώ μεγάλη τιμή είναι δυνατόν να έχει ως συνέπεια την αδυναμία τελικής ανίχνευσης οποιουδήποτε κύκλου.

Σε συνδυασμό με την sobel_edge

Ως binary εικόνα εισόδου στην circ_hough ορίζεται η έξοδος της συνάρτησης sobel_edge με thres = 0.4 . Η επιλογή αυτή πραγματοποιήθηκε λόγω της αποτελεσματικής ανίχνευσης της ακμής που αποτελεί περίγραμμα της μπάλας και της ταυτόχρονης αποφυγής ανίχνευσης ‘’θορύβου’’ στις περιοχές της μπάλας που παρουσιάζουν έντονη υφή, γνωρίσματα που παρουσιάζει η μέθοδος για αυτήν την τιμή threshold.

Έπειτα από πολλές δοκιμές, οι τιμές των dim και R_max ορίστηκαν:

dim = [200, 200, 120]

R_max = 400 (R=[50, 400])

Ακολουθεί η ασπρόμαυρη εικόνα εισόδου με χαραγμένους τους κύκλους που ανιχνεύει η μέθοδος circ_hough για πέντε διαφορετικές τιμές V_min:

1) V_min = 600

2) V_min = 750

3) V_min = 900

4) V_min = 1050

5) V_min = 1200

Από την παρατήρηση των πέντε παραπάνω εικόνων, αναδεικνύεται η σημαντικότητα της κατάλληλης επιλογής V_min. Είναι φανερό ότι για μικρές τιμές V_min, πολλοί υποψήφιοι κύκλοι θεωρούνται έγκυροι

και χαράσσονται, αν και δεν αποτελούν πραγματικούς κύκλους της εικόνας. Καθώς το V_min αυξάνεται, ολοένα και λιγότεροι κύκλοι χαράσσονται, τελικά. Θεωρητικά, η μέθοδος λειτουργεί σωστά αν ο κύκλος με τον μέγιστο αριθμό ψήφων είναι ο κύκλος που συμπίπτει με το περίγραμμα της μπάλας. Καταλληλότερη επιλογή V_min αποτελεί μία τιμή ελαφρώς μικρότερη από το πλήθος ψήφων που λαμβάνει ο κύκλος αυτός. Περαιτέρω αύξηση της τιμής V_min έχει ως συνέπεια η μέθοδος να μην ανιχνεύει κάποιον κύκλο.

Σε συνδυασμό με την log_edge:

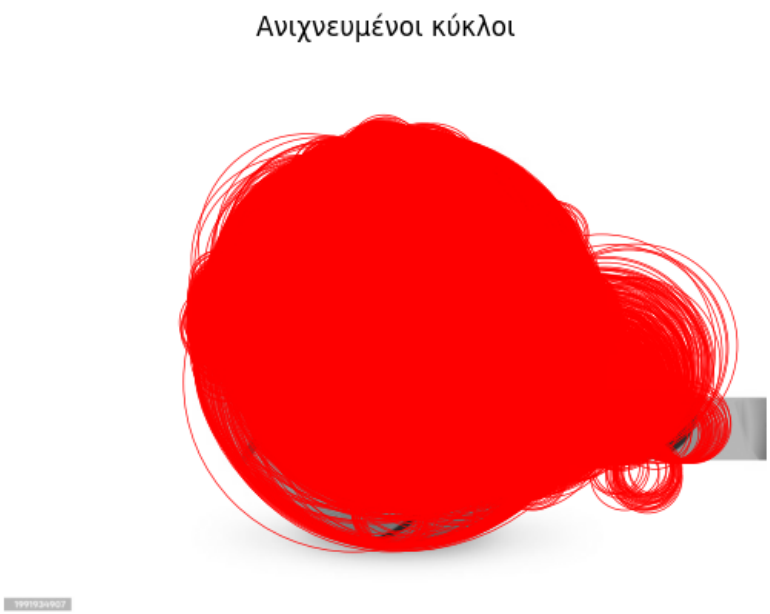
Όπως και παραπάνω, ορίστηκαν:

dim = [200, 200, 120]

R_max = 400 (R=[50, 400])

Ακολουθεί η ασπρόμαυρη εικόνα εισόδου με χαραγμένους τους κύκλους που ανιχνεύει η μέθοδος circ_hough για πέντε διαφορετικές τιμές V_min:

1) V_min = 180



2) V_min = 250

3) V_min = 300

4) V_min = 350

Ανιχνευμένοι κύκλοι

5) V_min = 390

Ανιχνευμένοι κύκλοι

Παρόμοια συμπεράσματα προκύπτουν και με την εφαρμογή της μεθόδου circ_hough στην binary εικόνα που προκύπτει από την μέθοδο log_edge. Για μικρές τιμές V_min, ανιχνεύεται μεγάλος αριθμός κύκλων, ενώ με αύξηση της τιμής V_min, το πλήθος των κύκλων μειώνεται σημαντικά. Για επιλογή τιμής V_min λίγο μικρότερης από τον αριθμό ψήφων του κύκλου που συμπίπτει με το περίγραμμα της μπάλας, ο κύκλος αυτός είναι ο μοναδικός που ανιχνεύεται.

Συγκρίνοντας την απόδοση της μεθόδου circ_hough σε συνδυασμό με την sobel_edge(thres = 0.4) και την αντίστοιχη σε συνδυασμό με την log_edge (thres = 0.3, mode = 3), παρατηρείται ότι η δεύτερη είναι σημαντικά ταχύτερη. Αυτό συμβαίνει καθώς η binary εικόνα που προκύπτει από την μέθοδο log_edge οδηγεί σε αρκετά μικρότερο πλήθος pixel ακμών. Ως αποτέλεσμα, η μέθοδος circ_hough ελέγχει λιγότερους υποψήφιους κύκλους και ολοκληρώνεται γρηγορότερα.