Ανίχνευση ακμών και κύκλων

Βογιατζής Χαρίσιος ΑΕΜ: 9192

May 22, 2025

Abstract

Σε αυτήν την εργασία υλοποιούμε και εξετάζουμε τρεις αλγορίθμους για *ανίχνευση* ακμών και κύκλων σε grayscale εικόνες: *sobel*, *LoG* και *Hough*.

1 Εισαγωγή

Στόχος μας είναι:

- **Ανίχνευση ακμών**: Χρήση Τελεστή Sobel και Τελεστή Laplacian of Gaussian (LoG).
- **Ανίχνευση κύκλων**: Χρήση μεθόδου Hough.

2 Υλοποίηση

2.1 Συναρτήσεις

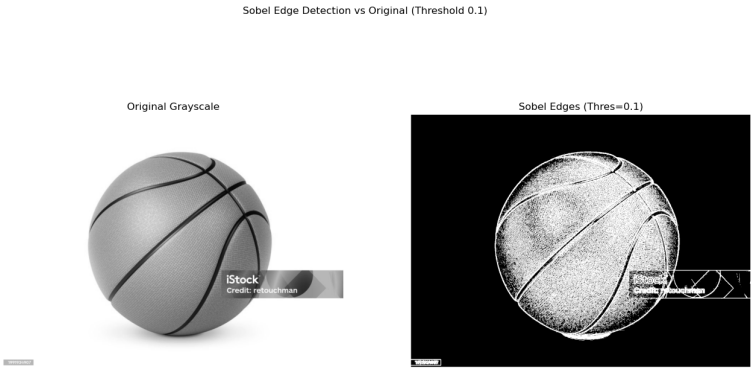
- `fir_conv.py`: Βοηθητική συνάρτηση για τον υπολογισμό της συνέλιξης μιας μάσκας (kernel) και της εικόνας.
- `sobel_edge.py`: Συνάρτηση που χρησιμοποιεί τον τελεστή Sobel για να υπολογίσει το μέτρο της κλίσης σε κάθε σημείο της εικόνας, εντοπίζοντας έτσι περιοχές με μεγάλες χωρικές συχνότητες που αντιστοιχούν σε ακμές. Χρησιμοποιεί δύο συνελικτικούς kernels για να προσεγγίσει τις παραγώγους στην οριζόντια και στην κατακόρυφη διεύθυνση. Το μέτρο της κλήσης, έπειτα, περνάει από ένα κατώφλι (threshold) για να δημιουργηθεί ένας δυαδικός πίνακας-εικόνα που περιέχει τις ακμές.
- `log_edge.py`: Συνάρτηση που χρησιμοποιεί τον τελεστή Laplacian of Gaussian (LoG). Αρχικά χρησιμοποιείται ένα Gaussian (βαθυπερατό) φίλτρο για να εξομαλύνει τον θόρυβο και στη συνέχεια τον τελεστή Laplace για να εντοπίσει περιοχές με μεγάλες αλλαγές έντασης. Οι ακμές βρίσκονται στα zero-crossings της εξόδου του LoG.
- `circ_hough.py`: Συνάρτηση που χρησιμοποιεί τη μέθοδο Hough για να ανιχνεύση (στην περίπτωση μας) κύκλους. Επιλέχθηκε να χρησιμοποιηθεί η μέθοδος sobel για την εξαγωγή της binary εικόνας και της χρήσης της ως είσοδο εδώ.

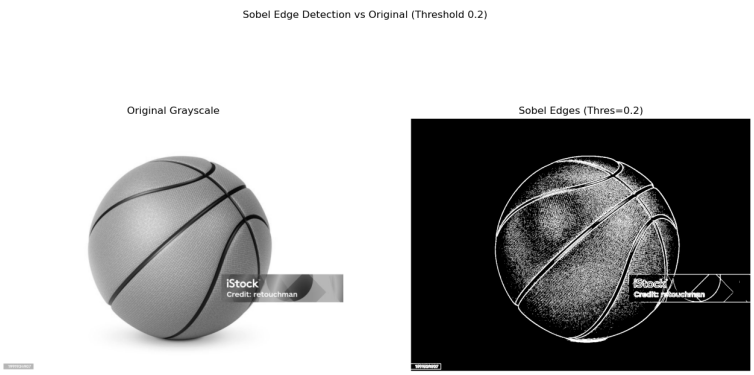
2.2 Demo Script (`demo.py`)

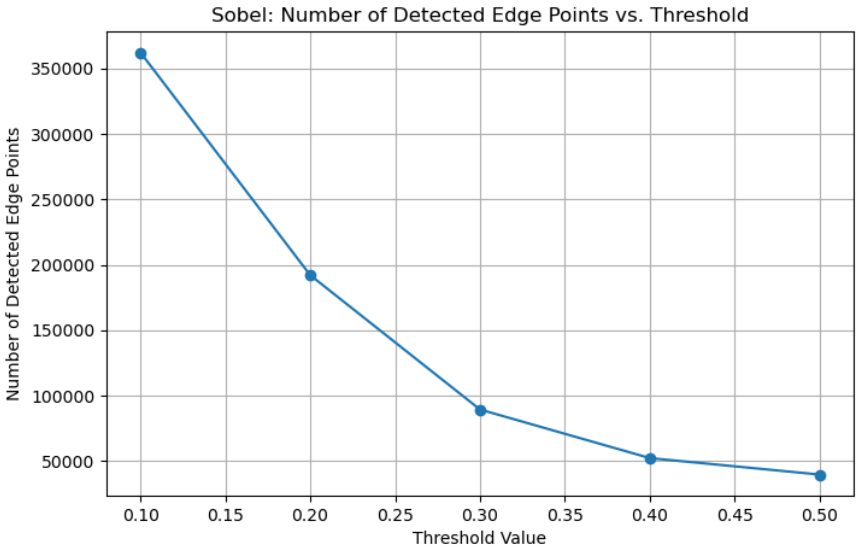
Έχει δημιουργηθεί ένα script επίδειξης demo.py το οποίο παράγει τις output εικόνες που παρουσιάζονται στα αποτελέσματα καθώς και συγκρίσεις με την grayscale εκδοχή της αρχικής εικόνας. Το script χρησιμοποιεί multithreading και βελτιστοποιήσεις καθώς οι αρχικοί χρόνοι εκτέλεσης ήταν υπερβολικά μεγάλοι. Δοκιμάστηκαν πολλοί συνδιασμοί διαφορετικών παραμέτρων (V_min, κλπ) ώστε να βελτιστοποιηθεί το runtime και να βελτιωθεί ο εντοπισμός κύκλων.

3 Αποτελέσματα

3.1 sobel_edge() για διάφορες threshold τιμές

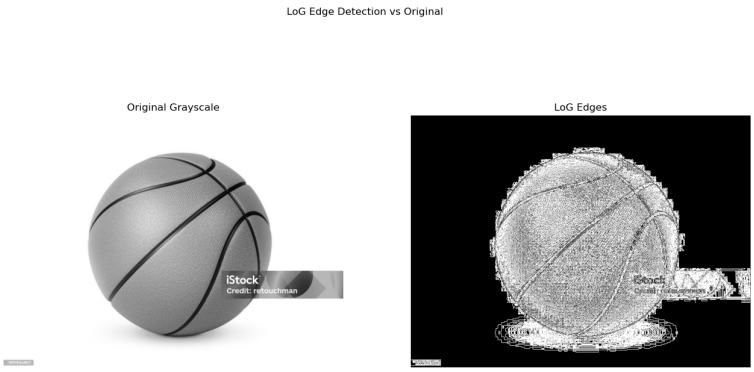






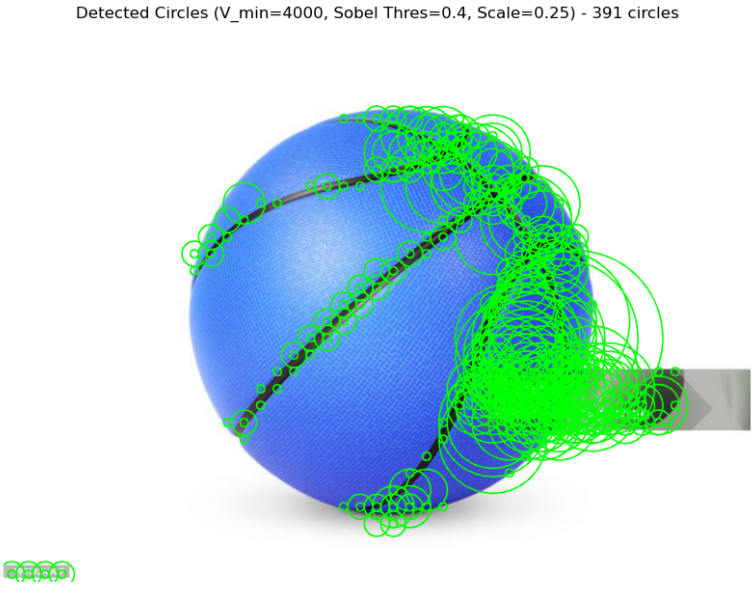
Όπως είναι αναμενόμενο καθώς αυξάνουμε το κατώφλι, λιγότερα σημεία εντοπίζονται ως σημεία ακμών (λευκά σημεία στα binary διαγράμματα). Αυτό επιβεβαιώνεται και από την καμπύλη του τελευταίου διαγράμματος. Παρατηρούμε ότι μία τιμή κατωφλίου στο [0.4-0.5] έχει το καλύτερο αποτέλεσμα.

3.2 log_edge()

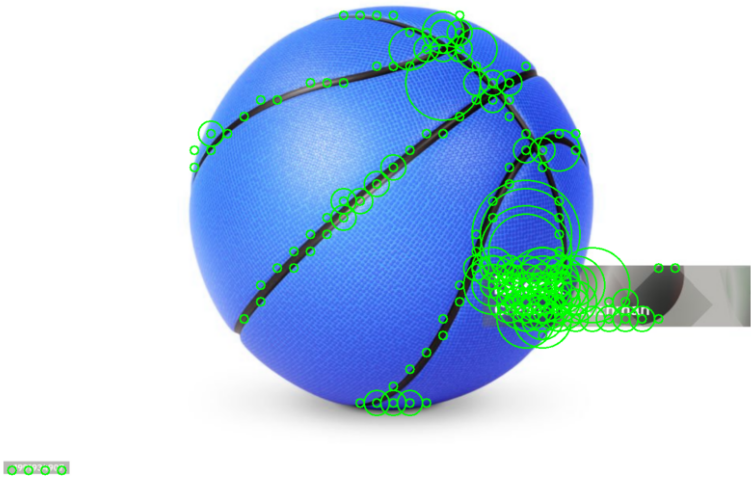


Όπως διαπιστώνουμε το αποτέλεσμα εδώ δεν είναι ικανοποιητικό και είναι υποδεέστερο. Πιθανώς με άλλη μάσκα (kernel) να είχαμε καλύτερο αποτέλεσμα, ή ακόμη και να υπάρχει κάποιο μικρό bug στην υλοποίηση.

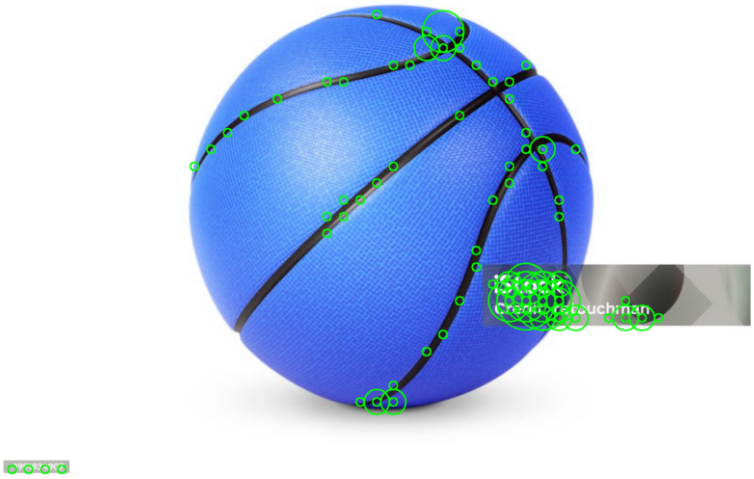
3.3 circ_hough()



Detected Circles (V_min=5000, Sobel Thres=0.4, Scale=0.25) - 189 circles



Detected Circles (V_min=6000, Sobel Thres=0.4, Scale=0.25) - 92 circles



Επιλέχθηκε να γίνει scale down της αρχικής εικόνας στο .25 για να μειωθεί ο χρόνος εκτέλεσης, γεγονός που επηρεάζει την ανίχνευση των κύκλων της εικόνας.

3.4 Σημείωση

Η αναφορά θα έπρεπε να είναι πληρέστερη αλλά οι δοκιμές των διαφορετικών παραμέτρων ήταν πολύ χρονοβόρες και περιόρισαν τον εναπομείναντα χρόνο μου.

4 Κώδικας

Μπορείτε να βρείτε τον κώδικα και στο Github.