

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ακαδημαϊκό Έτος: 2019-2020



ΟΡΑΣΗ ΥΠΟΛΟΓΙΣΤΩΝ

1η Εργαστηριακή Άσκηση

Θέμα: Εντοπισμός Σημείων Ενδιαφέροντος και Εξαγωγή
Χαρακτηριστικών σε Εικόνες

Τζε Χριστίνα-Ουρανία | 03116079
Ψαρουδάκης Ανδρέας | 03116001

27 Απριλίου 2020

Περιεχόμενα

1.1	Δημιουργία Εικόνων Εισόδου	2
1.1.1		2
1.1.2		3
1.2	Υλοποίηση Αλγορίθμων Ανίχνευσης Ακμών	3
1.2.1		4
1.2.2		4
1.2.3		5
1.2.4		5
1.3	Αξιολόγηση των Αποτελεσμάτων Ανίχνευσης Ακμών	5
1.3.1		5
1.3.2		6
1.3.3		7
1.4	Εφαρμογή των Αλγορίθμων Ανίχνευσης Ακμών σε Πραγματικές Εικόνες	9
1.4.1		9
1.4.2		9
2.1	Ανίχνευση Γωνιών	11
2.1.1		12
2.1.2		13
2.1.3		14
2.2	Πολυκλιμακωτή Ανίχνευση Γωνιών	15
2.2.1		15
2.2.2		15
2.3	Ανίχνευση Blobs	17
2.3.1		17
2.3.2		17
2.4	Πολυκλιμακωτή Ανίχνευση Blobs	18
2.4.1		18
2.5	Επιτάχυνση με την χρήση Box Filters και Ολοκληρωτικών Εικόνων	19
2.5.1		20
2.5.2		20
2.5.3		20
2.5.4		22
3.1	Τάιριασμα Εικόνων υπό Περιστροφή και Αλλαγή Κλίμακας	23
3.2	Κατηγοριοποίηση Εικόνων	25
3.2.1		25
3.2.2		26
3.2.3		26
3.2.4		26

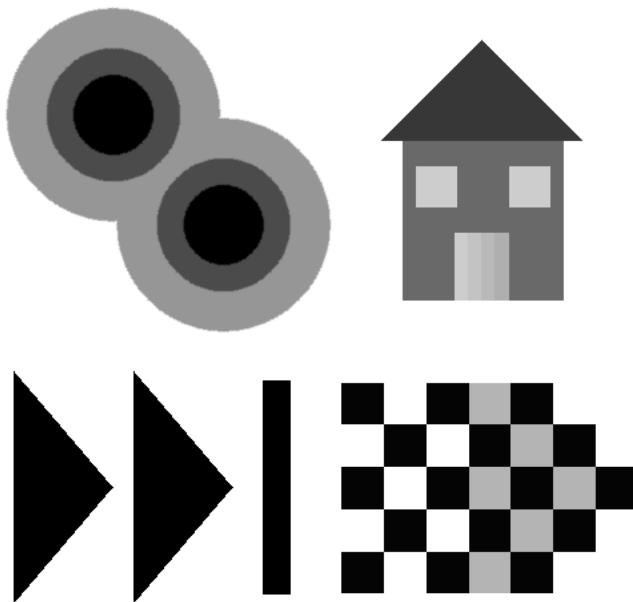
Εισαγωγή

Στόχος της παρούσας εργαστηριακής άσκησης υπόρξει η εξοικείωσή μας με μεθόδους εντοπισμού σημείων ενδιαφέροντος σε εικόνες και εξαγωγής των χαρακτηριστικών τους. Συγκεκριμένα, υλοποιήσαμε γνωστούς αλγορίθμους εύρεσης ακμών, γωνιών και blobs τόσο σε μία κλίμακα όσο και πολυκλιμακωτά. Στη συνέχεια, πειραματιστήκαμε με τις τιμές παραμέτρων των αλγορίθμων αυτών προκειμένου να καταλήξουμε στα καλύτερα δυνατά αποτελέσματα. Στην παρούσα αναφορά γίνεται μια σύντομη ανάλυση των θεωρητικών εννοιών που χρησιμοποιήθηκαν καθώς και παρουσίαση των πειραματικών αποτελεσμάτων.

Μέρος 1: Ανίχνευση Ακμών σε Γκρίζες Εικόνες

1.1 Δημιουργία Εικόνων Εισόδου

1.1.1 Αρχικά διαβάζουμε και τυπώνουμε την εικόνα edgetest_20.png σε grayscale με χρήση των συναρτήσεων **imread** και **imshow** των βιβλιοθηκών cv2 και matplotlib αντίστοιχα. Πρωτίστως, έχουμε κανονικοποίσει την εικόνα μας στο διάστημα [0,1] διαιρώντας όλες τις τιμές των pixels με 255.



1.1.2 Προσθέτουμε στην εικόνα μας λευκό gaussian όρυβο με μηδενική μέση τιμή και τυπική απόκλιση σ_n , τέτοια ώστε ο PSNR να έχει συγκεκριμένες τιμές. Έτσι, κατασκευάζουμε όρυβοις εικόνες $I(x, y) = I_o(x, y) + n(x, y)$ οι οποίες θα χρησιμοποιηθούν στη συνέχεια σαν είσοδοι στον αλγόριθμο ανίχνευσης ακμών. Χρησιμοποιούμε 2 τιμές για τον PSNR του όρυβου και συγκεκριμένα:

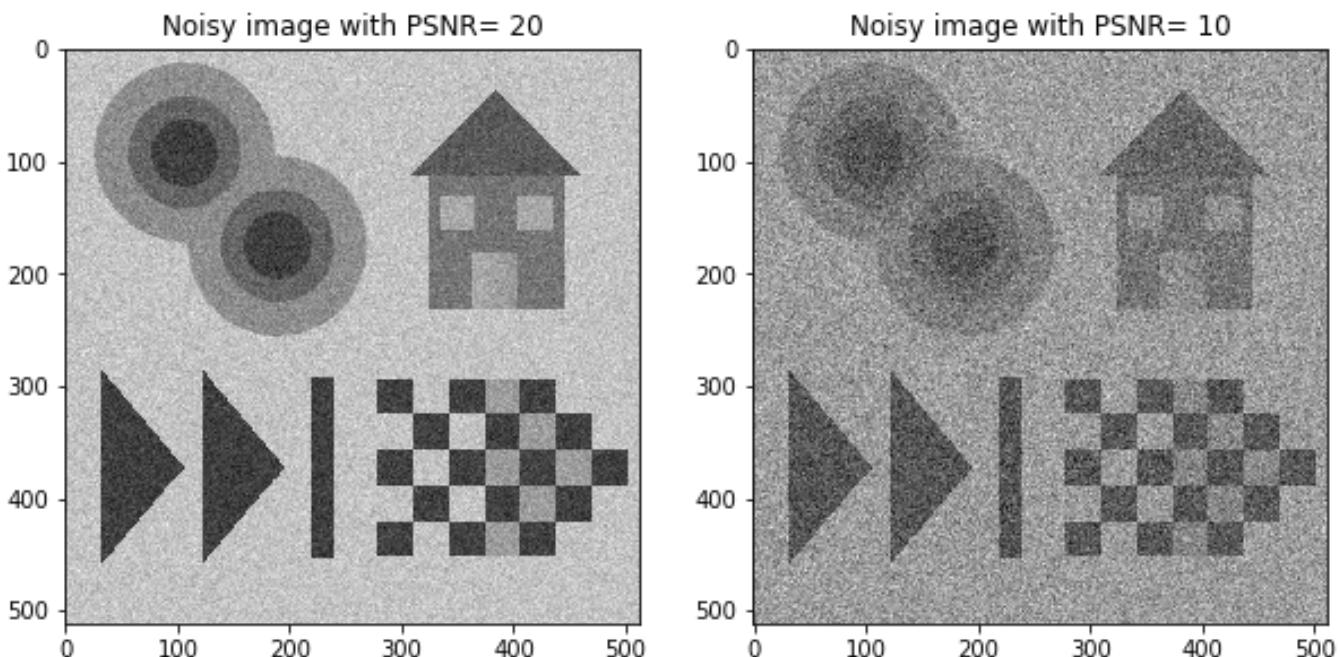
1. $\text{PSNR} = 20 \text{ dB}$
2. $\text{PSNR} = 10 \text{ dB}$

Η τιμή της τυπικής απόκλισης, σ_n , προκύπτει από το PSNR σύμφωνα με την ακόλουθη σχέση:

$$\text{PSNR} = 20 \log_{10}\left(\frac{I_{\max} - I_{\min}}{\sigma_n}\right) \Leftrightarrow \sigma_n = (I_{\max} - I_{\min}) \cdot 10^{-\frac{\text{PSNR}}{20}}$$

όπου I_{\max} και I_{\min} η μέγιστη και η ελάχιστη τιμή pixel της εικόνας.

Η δημιουργία του όρυβου γίνεται με χρήση της συνάρτησης **random.normal** της βιβλιοθήκης numpy



1.2 Υλοποίηση Αλγορίθμων Ανίχνευσης Ακμών

Στο ερώτημα αυτό κατασκευάζουμε την συνάρτηση **EdgeDetect** η οποία υλοποιεί τον αλγορίθμο ανίχνευσης ακμών που περιγράφεται στα ακόλουθα βήματα. Η **EdgeDetect** παίρνει για ορίσματα μια grayscale εικόνα, την τυπική απόκλιση σ της Gaussian, την παράμετρο θ_{edge} και μία boolean παράμετρο η οποία καθορίζει τον τύπο προσέγγισης της Laplacian (γραμμική ή μη-γραμμική). Η συνάρτηση επιστρέφει έναν πίνακα ίδιων διαστάσεων με την αρχική εικόνα με τιμή 1 στα pixels οποία ανιχνεύθηκε ακμή και 0 σε όλα τα υπόλοιπα.

1.2.1 Δημιουργούμε τις χρουστικές αποκρίσεις δύο διαχριτών γραμμικών φίλτρων που προσεγγίζουν τα συνεχή φίλτρα με τις εξής χρουστικές αποκρίσεις:

1. Δισδιάστατη Gaussian $G_\sigma(x, y)$
2. Laplacian-of-Gaussian (LoG) $h(x, y) = \nabla^2 G_\sigma(x, y)$

Και για τις δύο περιπτώσεις χρησιμοποιούμε για την Gaussian την ίδια τυπική απόκλιση (χωρική κλίμακα) σ ενώ οι πυρήνες έχουν έκταση $n \times n$ με $n = \text{ceil}(3 \cdot \sigma) \cdot 2 + 1$.

Για την κατασκευή των 2 παραπάνω φίλτρων δημιουργούμε τη συνάρτηση **CreateFilters**. Για την υλοποίηση του πρώτου φίλτρου εκμεταλλεύμαστε το γεγονός ότι η δισδιάστατη $n \times n$ Gaussian είναι διαχωρίσιμη και κατά συνέπεια μπορεί να προκύψει άμεσα ως γινόμενο δύο μονοδιάστατων Gaussian μεγέθους n , τις οποίες και ορίζουμε με χρήση της συνάρτησης **getGaussianKernel** της βιβλιοθήκης cv2. Για την LoG εφαρμόζουμε άμεση υλοποίηση του πυρήνα μέσω της συνάρτησης **meshgrid** της βιβλιοθήκης numpy. Στην συνέχεια αξιοποιούμε από το [1] τον αναλυτικό μαθηματικό τύπο ορισμού της:

$$LoG = \frac{\exp\left(\frac{-r^2}{2\sigma^2}\right)}{2\pi\sigma^4} \cdot \left(\frac{r^2}{\sigma^2} - 2\right), \quad r = \sqrt{x^2 + y^2}$$

Εναλλακτικά θα μπορούσε να γίνει υλοποίηση της LoG μέσω εφαρμογής Λαπλασιανού τελεστή στον Γκαουσιανό πυρήνα (χρήση συνάρτησης Laplacian της βιβλιοθήκης cv2), ωστόσο επιλέχθηκε η προηγούμενη μέθοδος ώστε να αποφευχθεί ενδεχόμενο σφάλμα προσέγγισης.

1.2.2 Στο βήμα αυτό πραγματοποιούμε συνέλιξη της δισδιάστατης Gaussian, G_σ , του προηγούμενου ερωτήματος με την αρχική θορυβώδη εικόνα, I , και παίρνουμε σαν έξοδο μία εξομαλυμένη εκδοχή της πρώτης, $I_\sigma = G_\sigma * I$. Στη συνέχεια, προσεγγίζουμε την Laplacian L της εξομαλυμένης εικόνας. Για τον σκοπό αυτό χρησιμοποιούμε τις ακόλουθες δύο διαφορετικές εναλλακτικές:

1. Γραμμική (L1): συνέλιξη της I με την LoG h:

$$L_1 = \nabla^2(G_\sigma * I) = (\nabla^2 G_\sigma) * I = h * I$$

2. Μη-γραμμική (L2): μη-γραμμική εκτίμηση της Laplacian της I_σ με μορφολογικά φίλτρα:

$$L_2 = I_\sigma \oplus B + I_\sigma \ominus B - 2I_\sigma$$

όπου B το ακόλουθο δομικό στοιχείο: $B = \begin{array}{|c|c|c|} \hline & \bullet & \\ \hline \bullet & \bullet & \bullet \\ \hline & \bullet & \\ \hline \end{array}$

Για την δισδιάστατη συνέλιξη της θορυβώδους εικόνας με τον Γκαουσιανό πυρήνα χρησιμοποιούμε την συνάρτηση **filter2D** της βιβλιοθήκης cv2 ενώ για τα μορφολογικά φιλτραρίσματα της εικόνας με το δομικό στοιχείο B τις συναρτήσεις **dilate** για το dilation και **erode** για το erosion.

1.2.3 Τα σημεία μηδενισμού (zero crossings) της Laplacian που προσδιορίσαμε στο προηγούμενο ερώτημα αποτελούν πιθανές ακμές της αρχικής εικόνας. Για την εύρεσή τους εφαρμόζουμε έναν αλγόριθμο ο οποίος αποτελείται από τα ακόλουθα βήματα:

1. Δημιουργούμε την δυαδική εικόνα προσήμου X η οποία έχει τιμή 1 στα pixels εκείνα όπου η Laplacian είναι μεγαλύτερη ή ίση του μηδενός και 0 αλλού.
2. Βρίσκουμε το περίγραμμα Y της X εφαρμόζοντας την ακόλουθη σχέση:

$$Y = (X \oplus B) - (X \ominus B) \approx \partial X$$

Τα zerocrossings και επομένως οι πιθανές ακμές της αρχικής εικόνας αντιστοιχούν στα σημεία εκείνα όπου η δυαδική εικόνα Y έχει τιμή 1.

1.2.4 Από τα σημεία που βρήκαμε στο ερώτημα 1.2.3 απορρίπτουμε τα zerocrossings σε σχετικά ομαλές περιοχές. Τελικά, επιλέγονται ως ακμές τα zerocrossings Y στα οποία η εξομαλυμένη εικόνα I_σ έχει σχετικά μεγάλη κλίση, δηλαδή τα pixels (i, j) για τα οποία ισχύει ότι:

$$Y[i, j] = 1 \text{ και } \|\nabla I_\sigma[i, j]\| > \theta_{edge} \cdot \max_{x, y} \|\nabla I_\sigma\|$$

Με άλλα λόγια, κρατάμε τα pixels εκείνα που αντιστοιχούν σε μη-ομαλές περιοχές της εικόνας, δηλαδή σε περιοχές όπου εμφανίζονται έντονες αλλαγές. Η επιλογή των σημείων αυτών καθορίζεται από την παράμετρο θ_{edge} οπως φαίνεται και από την παραπάνω σχέση. Στον κώδικα μας, για τον υπολογισμό του ανάδελτα της εξομαλυμένης εικόνας γίνεται χρήση της συνάρτησης **gradient** της βιβλιοθήκης numpy.

1.3 Αξιολόγηση των Αποτελεσμάτων Ανίχνευσης Ακμών

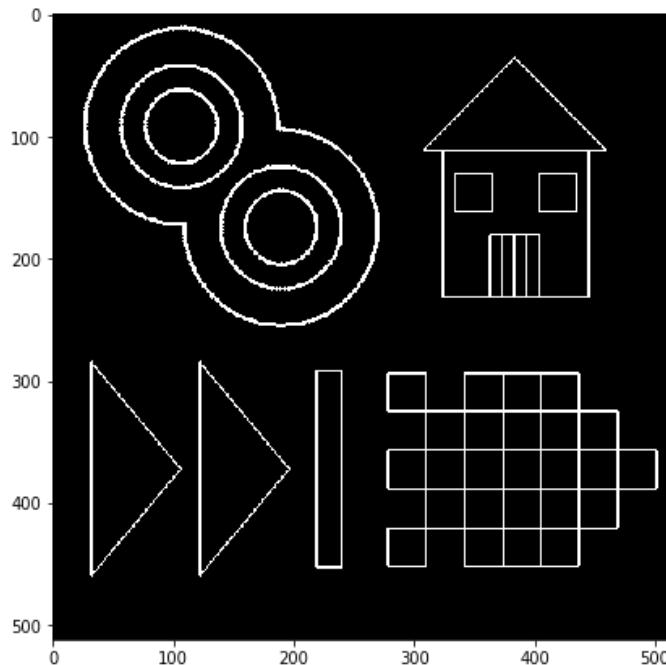
1.3.1 Για την ποσοτική αξιολόγηση των αποτελεσμάτων ανίχνευσης ακμών είναι αναγκαίος ο υπολογισμός των αληθινών ακμών στην αρχική καθαρή εικόνα I_0 . Ο υπολογισμός αυτός περιλαμβάνει αρχικά την εφαρμογή στην I_0 ενός απλού τελεστή ακμών (edge operator):

$$M = (I_0 \oplus B) - (I_0 \ominus B) \approx \partial I_0$$

Στη συνέχεια σχηματίζεται η δυαδική εικόνα T η οποία προκύπτει από κατωφλιοποίηση της εξόδου M ως εξής:

$$T = M > \theta_{realedge}$$

Η συνθήκη αυτή είναι αναγκαία καθώς μόνο αρκετά έντονες αλλαγές, μεγαλύτερες από ένα κατώφλι $\theta_{realedge}$, αντιστοιχούν σε πραγματικές ακμές. Για τον εντοπισμό των αληθινών ακμών ορίζουμε την συνάρτηση **RealEdgeDetect** η οποία δέχεται για ορίσματα την αρχική καθαρή εικόνα I_0 και την παράμετρο $\theta_{realedge}$ ενώ επιστρέφει την δυαδική εικόνα T . Παρακάτω φαίνεται το αποτέλεσμα της RealEdgeDetect για $\theta_{realedge} = 0.0392$. Η τιμή αυτή ήταν η μέγιστη τιμή της παραμέτρου $\theta_{realedge}$ για την οποία πετύχαμε να είναι εμφανείς όλες οι ακμές της αρχικής εικόνας συμπεριλαμβανομένων και των ακμών στην πόρτα του σπιτιού.



1.3.2 Στο βήμα αυτό πραγματοποιούμε ποσοτική αξιολόγηση των αποτελεσμάτων ανίχνευσης ακμών. Χρησιμοποιώντας τις δυαδικές εικόνες των αληθινών ακμών T και των ακμών D που ανίχνευσε ο αλγόριθμός μας στην θορυβώδη εικόνα εισόδου, υπολογίζουμε το ακόλουθο κριτήριο ποιότητας του αποτελέσματος ανίχνευσης:

$$C = \frac{Pr(D|T) + Pr(T|D)}{2}$$

όπου:

- $Pr(D|T)$ είναι το ποσοστό των ανιχνευθεισών ακμών που είναι αληθινές, δηλαδή το ποσοστό των ακμών που ορθά εντόπισε ο αλγόριθμος μας προς το σύνολο όλων των ακμών που εντόπισε. (Precision).
- $Pr(T|D)$ είναι το ποσοστό των αληθινών ακμών που ανιχνεύθησαν, δηλαδή το ποσοστό των ακμών που ορθά εντόπισε ο αλγόριθμος μας προς το σύνολο των πραγματικών ακμών (Recall).

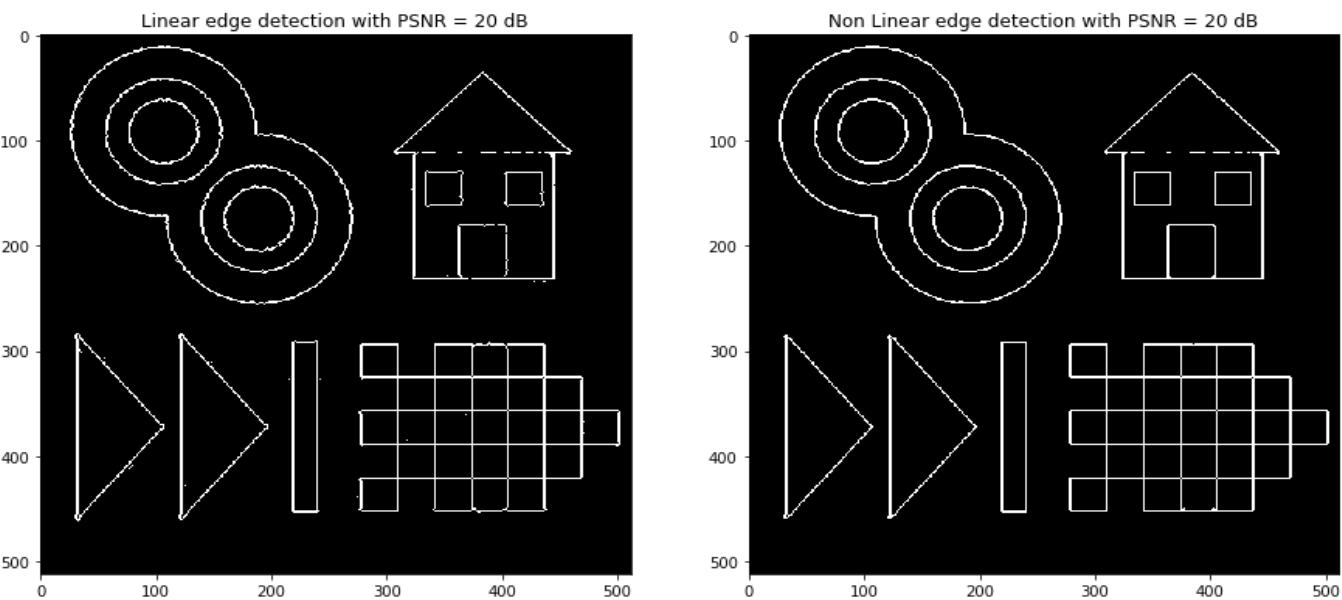
Αν τα D και T ειδωθούν σαν διακριτά σύνολα (με στοιχεία τα pixels στα οποία η δυαδική εικόνα έχει τιμή 1), τότε ισχύει:

$$Pr(T|D) = \frac{card(D \cap T)}{card(T)} \quad \text{και} \quad Pr(D|T) = \frac{card(D \cap T)}{card(D)}$$

όπου το $card(.)$ δίνει τον αριθμό των στοιχείων ενός συνόλου. Για τον υπολογισμό του $card$ κάθε δυαδικής εικόνας κάνουμε χρήση της συνάρτησης **sum**. Έτσι λοιπόν, με εφαρμογή των παραπάνω τύπων εξάγουμε τα ποσοστά ακρίβειας των μεθόδων ανίχνευσης που κατασκευάσαμε (γραμμική και μη γραμμική μέθοδος).

1.3.3 Τώρα για κάθε διαφορετική είσοδο (διαφορετικό PSNR) πειραματίζόμαστε με τις τιμές των παραμέτρων σ και θ_{edge} ώστε να πετύχουμε όσο το δυνατόν καλύτερα αποτελέσματα. Αρχικά χρησιμοποιούμε τις ενδεικτικές τιμές της εκφώνησης ($\sigma = 1.5$, $\theta_{edge} = 0.2$ για είσοδο με $PSNR = 20dB$ και $\sigma = 3$, $\theta_{edge} = 0.2$ για είσοδο με $PSNR = 10dB$). Έπειτα, δοκιμάζουμε μικρές αποκλίσεις των τιμών αυτών μέχρις ότου μεγιστοποιηθεί το κριτήριο ποιότητας του αποτελέσματος ανίχνευσης.

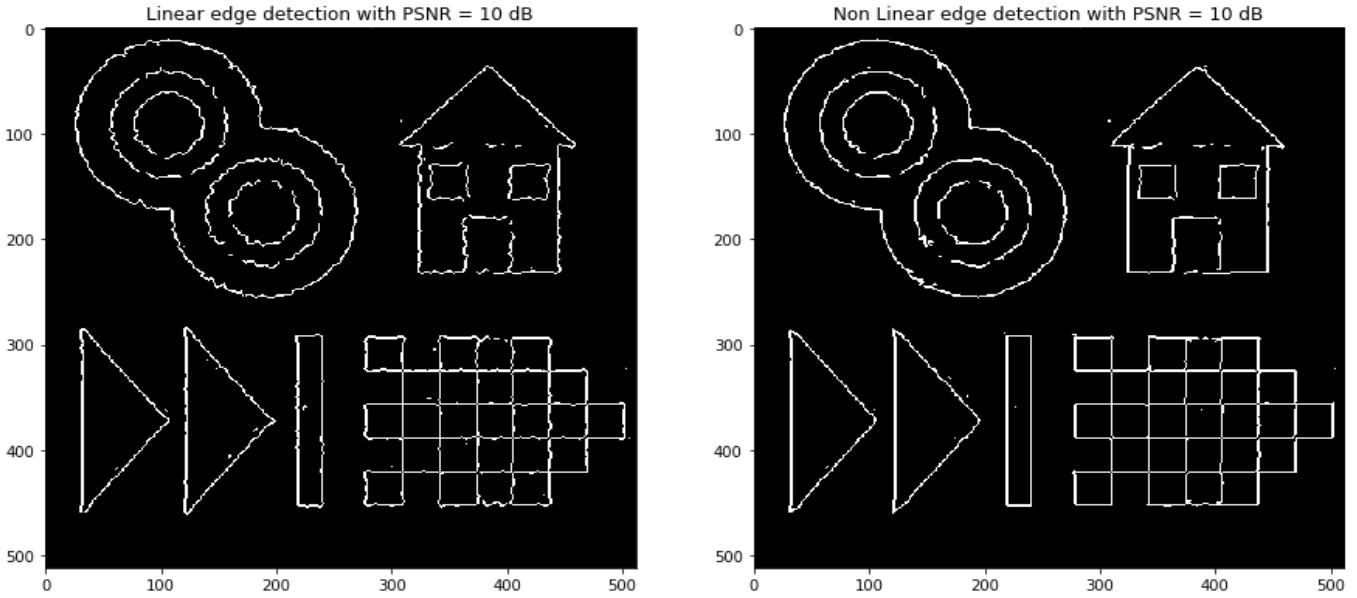
Για είσοδο με **PSNR = 20dB** καταλήγουμε ότι τα καλύτερα αποτελέσματα προκύπτουν για $\sigma = 1.75$ και $\theta_{edge} = 0.18$:



Για την **γραμμική προσέγγιση** το ποσοστό που λαμβάνουμε είναι **89.02%** ενώ για την **μη γραμμική προσέγγιση** είναι ίσο με **90.266%**. Παρατηρούμε επομένως, ότι η μη-γραμμική προσέγγιση της Laplacian οδηγεί σε ελαφρώς καλύτερα αποτελέσματα παρόλο που και οι δύο προσεγγίσεις είναι αρκετά ικανοποιητικές. Η βελτίωση που προκύπτει με χρήση της μη γραμμικής μεθόδου γίνεται αισθητή αν εστιάσουμε την προσοχή μας στα παράθυρα και στην πόρτα του σπιτιού, όπου στην περίπτωση αυτή οι ακμές φαίνονται ιδανικές. Επίσης, στην

μη-γραμμική προσέγγιση έχουν εντοπισθεί λανθασμένα ως ακμές ορισμένα pixels που βρίσκονται εντός του πλέγματος στο κάτω δεξιά τμήμα της εικόνας. Να σημειωθεί ότι για καμία τιμή των παραμέτρων σ και θ_{edge} δεν κατέστη δυνατός ο εντοπισμός των κάθετων ακμών εντός της πόρτας του σπιτιού.

Για είσοδο με **PSNR = 10dB** καταλήγουμε ότι τα καλύτερα αποτελέσματα προκύπτουν για $\sigma = 3$ και $\theta_{edge} = 0.2$:



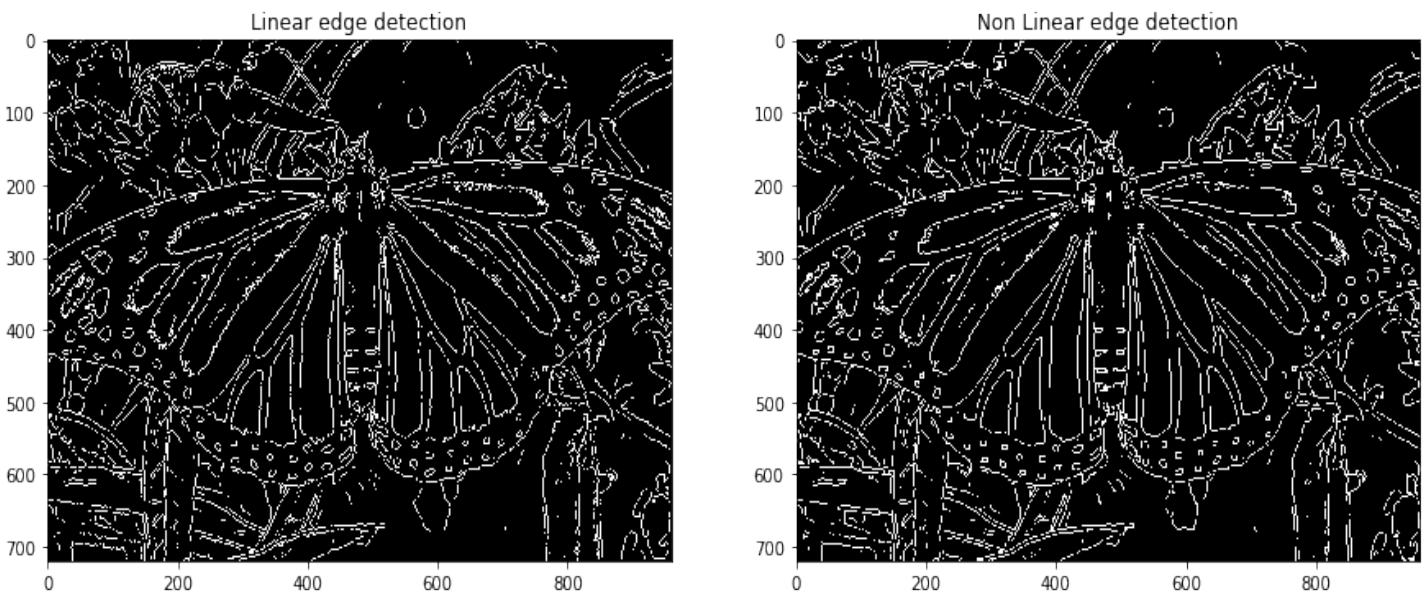
Για την γραμμική προσέγγιση το ποσοστό που λαμβάνουμε είναι **74.64%** ενώ για την μη γραμμική προσέγγιση είναι ίσο με **81.663%**. Παρατηρούμε ότι σε αυτή την περίπτωση η μη-γραμμική προσέγγιση της Laplacian οδηγεί σε αισθητά καλύτερα αποτελέσματα. Η βελτίωση που προκύπτει με χρήση της μη γραμμικής μεθόδου γίνεται αισθητή αν πάλι εστιάσουμε την προσοχή μας στα παράθυρα και στην πόρτα του σπιτιού, όπου στην περίπτωση αυτή οι ακμές φαίνεται να προσεγγίζουν αρκετά περισσότερο τις ιδανικές από ότι στη γραμμική μέθοδο. Επίσης, στην μη-γραμμική προσέγγιση έχουν εντοπισθεί λανθασμένα πολλά pixels που βρίσκονται εντός των τριγώνων και εντός του πλέγματος στο κάτω δεξιά τμήμα της εικόνας. Να σημειωθεί ότι και εδώ, όπως ήταν αναμενόμενο, για καμία τιμή των παραμέτρων σ και θ_{edge} δεν κατέστη δυνατός ο εντοπισμός των κάθετων ακμών εντός της πόρτας του σπιτιού.

Συμπερασματικά, η μη-γραμμική προσέγγιση φαίνεται να οδηγεί σε αισθητά καλύτερα αποτελέσματα στην περίπτωση έντονου ύφορύβου ενώ για μικρή ισχύς ύφορύβου και οι δύο μέθοδοι παρουσιάζουν παρεμφερή ικανοποιητικά αποτελέσματα.

1.4 Εφαρμογή των Αλγορίθμων Ανίχνευσης Ακμών σε Πραγματικές εικόνες

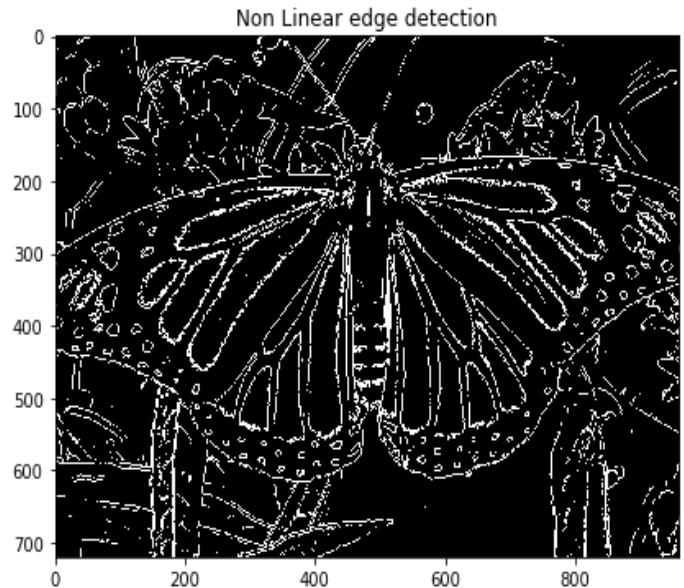
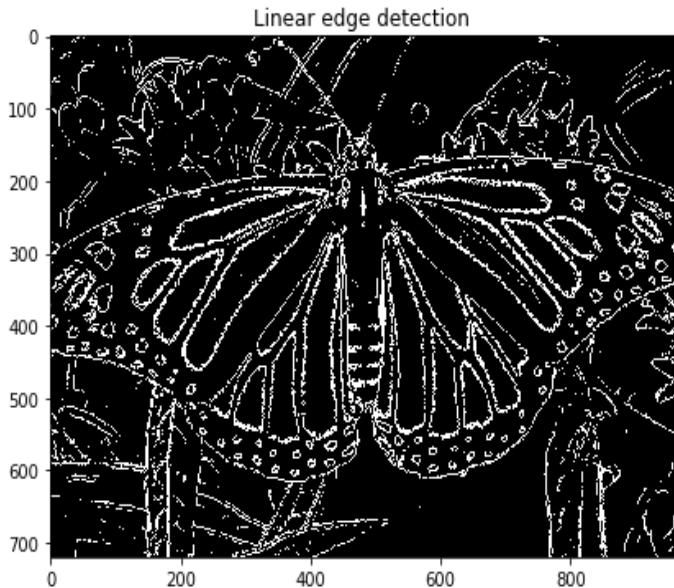
1.4.1 Στη συνέχεια εφαρμόζουμε τον αλγόριθμο ανίχνευσης ακμών στην πραγματική εικόνα butterfly.jpg χωρίς την προσθήκη θορύβου και εξετάζουμε την αποτελεσματικότητά του. Για τον σκοπό αυτό διαβάζουμε αρχικά την εικόνα μας σε grayscale με χρήση της συνάρτησης **imread** της cv2 και έπειτα την κανονικοποιούμε στο διάστημα $[0,1]$. Για την εφαρμογή του αλγορίθμου καλούμε την συνάρτηση **EdgeDetect** δύο φορές, μία για κάθε προσέγγιση της Laplacian (γραμμική και μη-γραμμική) με ορίσματα την εικόνα και τις παραμέτρους σ και θ_{edge} με τιμές όπως φαίνονται στο ακόλουθο ερώτημα.

1.4.2 Πειραματίζόμαστε με τις τιμές των παραμέτρων σ και θ_{edge} μέχρι να πετύχουμε όσο το δυνατόν καλύτερα ποιοτικά αποτελέσματα. Ο πειραματισμός αυτός τελικά δίνει τις τιμές $\sigma = 1.95$ και $\theta_{edge} = 0.1$ για τις οποίες η ανίχνευση ακμών φαίνεται παρακάτω:

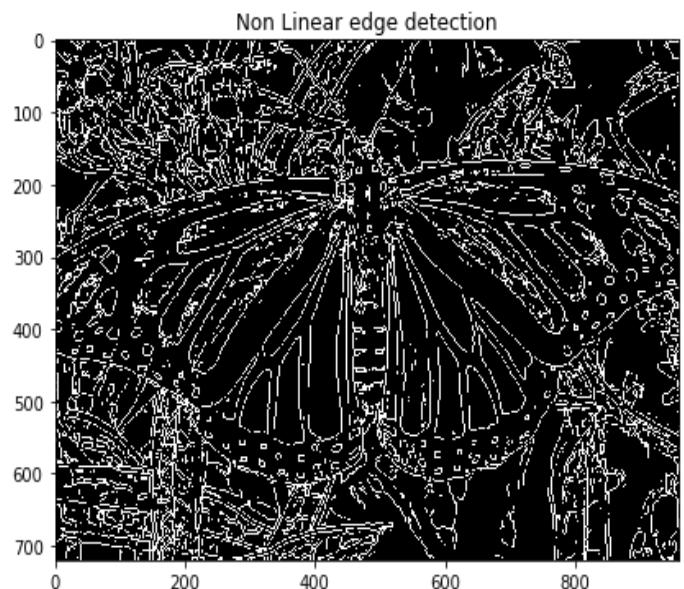
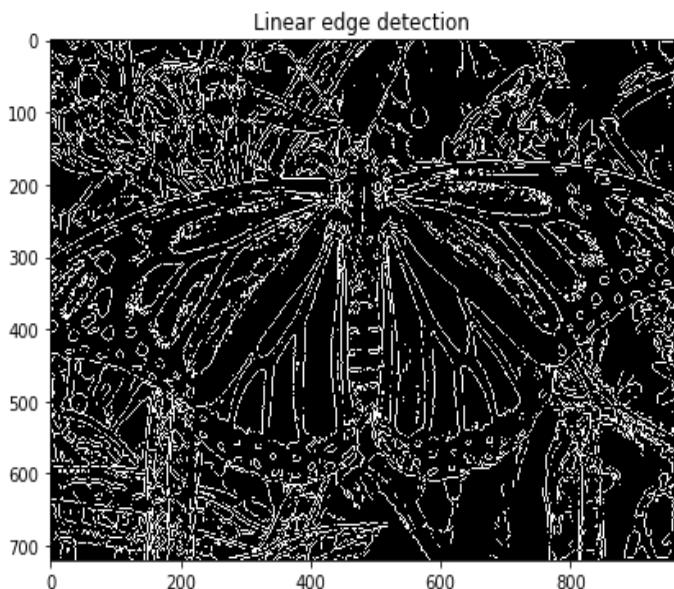


Σε αντίθεση με πριν παρατηρούμε ότι στην περίπτωση των πραγματικών εικόνων οι δύο τρόποι προσέγγισης της Laplacian (γραμμικός και μη-γραμμικός) καταλήγουν σε πολύ κοντινά αποτελέσματα. Παρατηρούμε επίσης ότι για τις συγκεκριμένες τιμές των παραμέτρων ανιχνεύονται ικανοποιητικά τόσο οι ακμές της πεταλούδας που βρίσκεται στο foreground όσο και των λουλουδιών στο background. Σε περίπτωση που δεν μας ενδιαφέρει η ανίχνευση των ακμών των λουλουδιών τότε μπορούμε να επικεντρωθούμε στην πεταλούδα μεταβάλλοντας απλά τις τιμές των παραμέτρων.

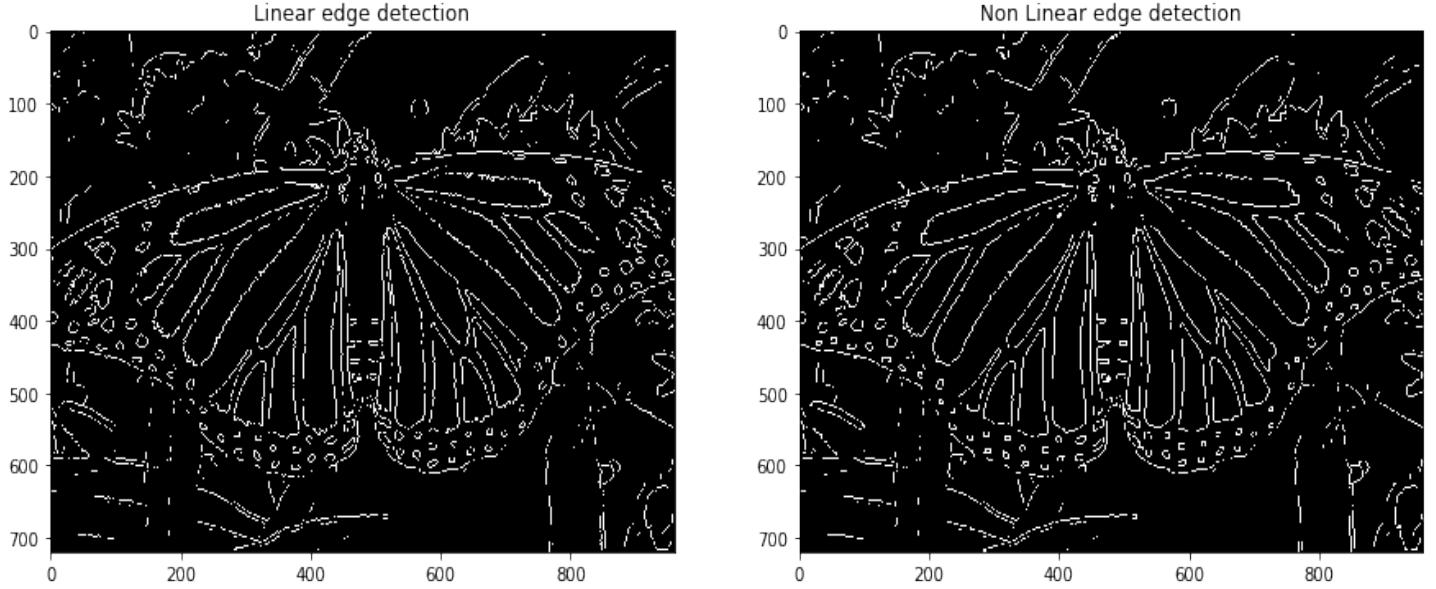
Πιο συγκεκριμένα, για $\sigma = 0.77$ και $\theta_{edge} = 0.1$ λαμβάνουμε τα ακόλουθα αποτελέσματα στα οποία η ανίχνευση ακμών έχει εστιάσει περισσότερο στην πεταλούδα.



Διατηρώντας σταθερή την τιμή της παραμέτρου σ και μεταβάλλοντας την θ_{edge} παρατηρούμε ότι για $\theta_{edge} < 0.1$ τα αποτελέσματα στα οποία καταλήγουμε δεν είναι ευδιάκριτα. Παραδείγματος χάριν, για $\sigma = 1.95$ και $\theta_{edge} = 0.05$ καταλήγουμε στα εξής:



Αντιθέτως, για τιμές μεγαλύτερες από 0.1 ανιχνεύονται όλο και λιγότερες ακμές. Για $\sigma = 1.95$ και $\theta_{edge} = 0.2$ εξακολουθούμε να διαχρίνουμε τις ακμές της πεταλούδας ενώ οι περισσότερες του background έχουν εξαφανιστεί.



Μέρος 2: Ανίχνευση Σημείων Ενδιαφέροντος

Στο μέρος αυτό ασχολούμαστε με την ανίχνευση σημείων ενδιαφέροντος σε εικόνες, όπως γωνίες και blobs τόσο σε μία κλίμακα όσο και πολυχλιμακωτά. Για τον σκοπό αυτό υλοποιούμε ένα σύνολο από ανιχνευτές ως συναρτήσεις με ορίσματα την εικόνα σε grayscale και τις απαιτούμενες παραμέτρους ($\sigma, \rho, s, N, k, \theta_{corn}$). Ως έξοδο επιστρέφουν τις συντεταγμένες (x, y) (όπου το x καθορίζει την στήλη και το y την γραμμή του αντίστοιχου pixel) των σημείων ενδιαφέροντος καθώς επίσης και την κλίμακα στην οποία ανιχνεύθηκαν. Για την οπτικοποίηση των αποτελεσμάτων ανίχνευσης χρησιμοποιούμε την έτοιμη συνάρτηση **interest_points_visualization** στην οποία τα ανιχνευθέντα σημεία αναπαρτιστώνται ως κύκλοι με κέντρο τα σημεία αυτά και ακτίνα ανάλογης της κλίμακας στην οποία ανιχνεύθηκαν.

2.1 Ανίχνευση Γωνιών

Μια διαισθητικά απλή κατηγορία σημείων ενδιαφέροντος είναι η επιλογή σημείων που αντιστοιχούν σε γωνίες της εικόνας. Για την ανίχνευση των γωνιών υλοποιούμε την κλασσική μέθοδο των Harris-Stephens [6], η οποία επεξηγείται συνοπτικά παρακάτω. Η μέθοδος αυτή βασίζεται στο γεγονός ότι οι γωνίες αποτελούν σημεία της εικόνας με μεγάλη αλλαγή φωτεινότητας προς όλες τις κατευθύνσεις. Η ιδέα αυτή διατυπώνεται μαθηματικά από την ακόλουθη σχέση η οποία εκφράζει την αλλαγή της φωτεινότητας για μία μετατόπιση (u, v) :

$$E(u, v) = \sum_{x,y} \underbrace{w(x, y)}_{\text{window function}} \underbrace{[I(x+u, y+v) - I(x, y)]^2}_{\text{shifted intensity}} \underbrace{I(x, y)}_{\text{intensity}}$$

Για ανίχνευση γωνιών η παραπάνω συνάρτηση, και επομένως ο δεύτερος όρος, πρέπει να μεγιστοποιηθεί. Αποδεικνύεται ότι με εφαρμογή αναπτύγματος κατά Taylor η παραπάνω σχέση

γράφεται στην μορφή:

$$E(u, v) \approx [u \ v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

όπου

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} = w(x, y) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Σύμφωνα με την Harris-Stephens μέθοδο αν λ_1 και λ_2 οι ιδιοτιμές, $\det(M) = \lambda_1 \lambda_2$ η ορίζουσα και $\text{trace}(M) = \lambda_1 + \lambda_2$ το ίχνος του πίνακα M τότε οι γωνίες ανιχνεύονται στα σημεία εκείνα για τα οποία η τιμή R είναι μεγάλη, δηλαδή οι ιδιοτιμές λ_1 και λ_2 είναι μεγάλες, όπου R το κριτήριο γωνιότητας:

$$R = \det(M) - k(\text{trace}(M))^2$$

Όλα τα παραπάνω υλοποιούνται στην συνάρτηση **AngleDetect** η οποία δέχεται για ορίσματα την εικόνα σε grayscale και τις επιπλέον απαιτούμενες παραμέτρους, όπως επεξηγείται λεπτομερώς παρακάτω. Η συνάρτηση αυτή επιστρέφει έναν $N \times 3$ πίνακα στον οποίο οι δύο πρώτες στήλες είναι οι συντεταγμένες των ανιχνευθέντων σημείων και η τρίτη η χλίμακα στην οποία ανιχνεύθηκαν. Τέλος επιστρέφει έναν πίνακα τύπου Boolean με τιμές 1 στα σημεία ενδιαφέροντος και 0 αλλού, ο οποίος χρησιμοποιείται σε επόμενο βήμα κατά την πολυχλιμακωτή ανίχνευση γωνιών.

2.1.1 Υπολογίζουμε τα στοιχεία J_1, J_2, J_3 του δομικού τανυστή \mathbf{J} σε κάθε pixel (x, y) της εικόνας, σύμφωνα με τις σχέσεις:

$$\begin{aligned} J_1(x, y) &= G_\rho * \left(\frac{\partial I_\sigma}{\partial x} \cdot \frac{\partial I_\sigma}{\partial x} \right) \\ J_2(x, y) &= G_\rho * \left(\frac{\partial I_\sigma}{\partial x} \cdot \frac{\partial I_\sigma}{\partial y} \right) \\ J_3(x, y) &= G_\rho * \left(\frac{\partial I_\sigma}{\partial y} \cdot \frac{\partial I_\sigma}{\partial y} \right) \end{aligned}$$

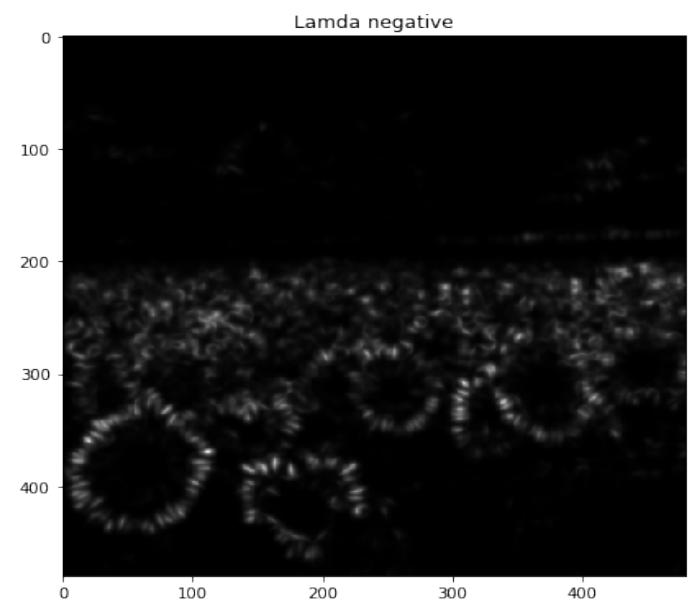
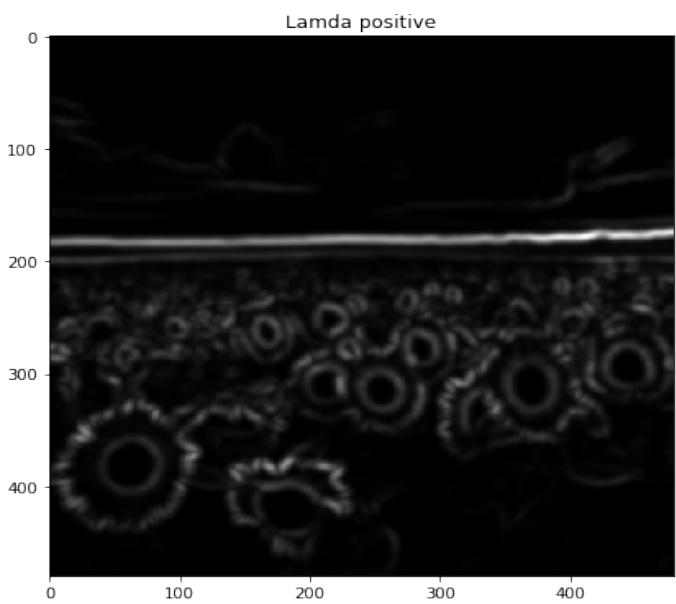
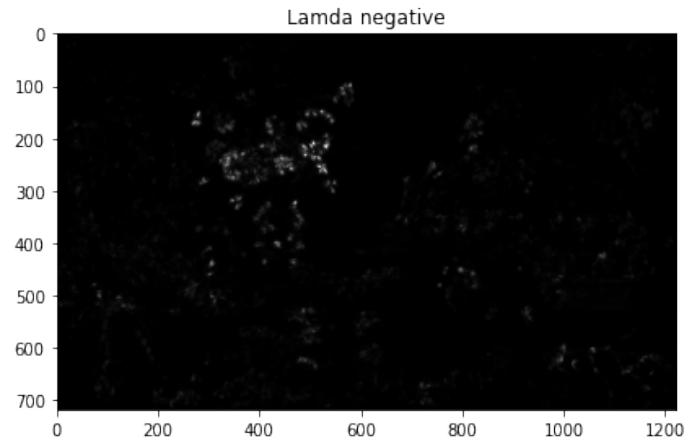
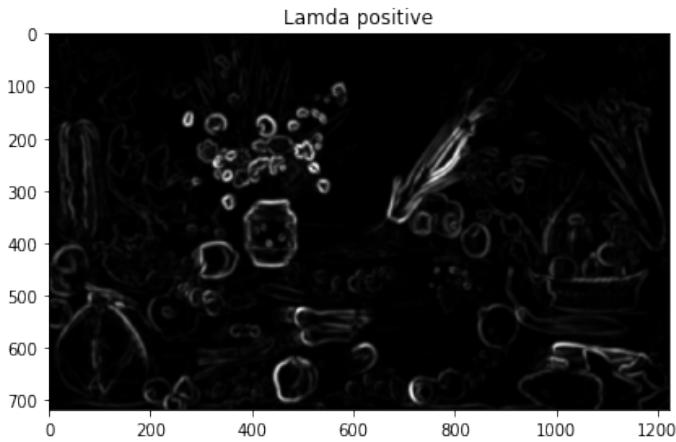
όπου: $I_\sigma = G_\sigma * I$ και G_ρ, G_σ δισδιάστατοι Gaussian πυρήνες ομαλοποίησης με τυπικές αποκλίσεις σ (χλίμακα διαφόρισης) και ρ (χλίμακα ολοκλήρωσης) αντίστοιχα.

Για την υλοποίηση των δισδιάστατων Gaussian G_ρ, G_σ καλούμε τη συνάρτηση **CreateFilters** που ορίσαμε στο μέρος 1.2.1, με ορίσματα ρ και σ αντίστοιχα. Για τον υπολογισμό των μερικών παραγώγων αξιοποιούμε την συνάρτηση **gradient** της βιβλιοθήκης numpy ενώ για την υλοποίηση των συνελίξεων χρησιμοποιούμε τη συνάρτηση **filter2D** της βιβλιοθήκης cv2. Οι πολλαπλασιαμοί μεταξύ των μερικών παραγώγων γίνονται με χρήση του τελεστή «*» της python ώστε ο πολλαπλασιασμός να πραγματοποιηθεί σημείο προς σημείο.

2.1.2 Υπολογίζουμε τις ιδιοτιμές λ_-, λ_+ του τανυστή J σε κάθε pixel, σύμφωνα με τις σχέσεις:

$$\lambda_{\pm}(x, y) = \frac{1}{2} \left(J_1 + J_3 \pm \sqrt{(J_1 - J_3)^2 + 4J_2^2} \right)$$

Χρησιμοποιούντας τις ενδεικτικές τιμές παραμέτρων της εκφώνησης ($\sigma = 2$, $\rho = 2.5$, $k = 0.05$, $\theta_{corn} = 0.005$, $s = 1.5$, $N = 4$) σχεδιάζουμε σαν γκρίζες εικόνες τα $\lambda_+(x, y)$ και $\lambda_-(x, y)$ τόσο για την εικόνα Caravaggio2.jpg όσο και για την sunflowers.png :



Παρατηρούμε ότι η ιδιοτιμή λ_+ δείχνει έντονες τις ακμές των αρχικών εικόνων, ενώ η ιδιοτιμή λ_- εστιάζει σε πιο συγκεκριμένα σημεία, όπως οι γωνίες. Επίσης, και οι δύο ιδιοτιμές λαμβάνουν χαμηλές τιμές στις ομαλές περιοχές των αρχικών εικόνων (π.χ ουρανός στην εικόνα sunflowers.png).

2.1.3 Με βάση τις υπολογισμένες ιδιοτιμές λ_- και λ_+ εφαρμόζουμε το ακόλουθο κριτήριο γωνιότητας (cornerness criterion):

$$R(x, y) = \lambda_- \lambda_+ - k \cdot (\lambda_- + \lambda_+)^2$$

όπου k είναι μια μικρή θετική σταθερά. Στη συνέχεια επιλέγουμε σαν γωνίες τα pixels (x, y) τα οποία:

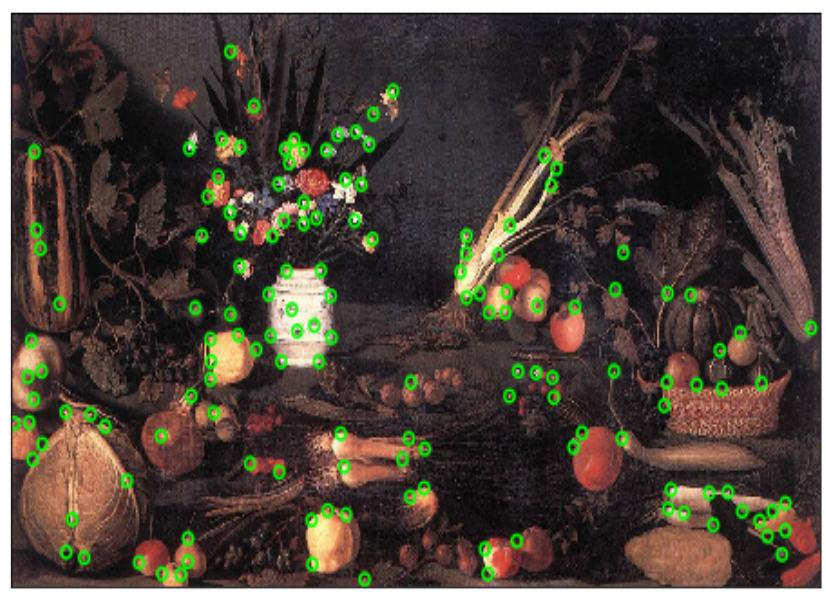
1. Είναι μέγιστα του R εντός τετραγωνικών παραθύρων που τα περιβάλλουν το μέγευθος των οποίων εξαρτάται από την κλίμακα σ .
2. Αντιστοιχούν σε τιμή του R μεγαλύτερη από ένα ποσοστό του ολικού μεγίστου του R , δηλαδή $R(x, y) > \theta_{corn} \cdot R_{max}$, όπου θ_{corn} ένα κατάλληλα επιλεγμένο κατώφλι.

Η υλοποίηση της συνθήκης (1) σε python γίνεται με χρήση των εντολών που δίνονται στην εκφώνηση. Κατά την εφαρμογή της συνθήκης (2) κάνουμε χρήση της συνάρτησης **max** για την εύρεση του μέγιστου στοιχείου του πίνακα R . Συνδυάζουμε τις δύο συνθήκες μέσω του λογικού τελεστή «&». Το αποτέλεσμα είναι ένας boolean πίνακας με τιμές True στις θέσεις εκείνες όπου ανιχνεύθηκε γωνία. Μέσω της συνάρτησης **argwhere** της βιλβιοθήκης numpy βρίσκουμε τις συντεταγμένες των σημείων όπου εντοπίστηκε γωνία, δηλαδή των σημείων του πίνακα με τιμή True. Έπειτα, χρησιμοποιούμε τη συνάρτηση **flip** της βιλβιοθήκης numpy για να αντιστρέψουμε τις συντεταγμένες των σημείων αυτών μιας και για την οπτικοποίηση των αποτελεσμάτων ανίχνευσης χρησιμοποιούμε την έτοιμη συνάρτηση **interest_points_visualization**, στην οποία πρέπει να δώσουμε ως είσοδο τις συντεταγμένες στη μορφή $(x, y) = (\text{στήλη}, \text{γραμμή})$.

Καλούμε λοιπόν τώρα τη συνάρτηση **AngleDetect** με τις ενδεικτικές τιμές της εκφώνησης ($\sigma = 2$, $\rho = 2.5$, $k = 0.05$, $\theta_{corn} = 0.005$). Στη συνέχεια, δοκιμάζουμε μικρές αποκλίσεις των τιμών αυτών μέχρις ότου επιτευχθεί η καλύτερη δυνατή ανίχνευση γωνιών. Για την παρουσίαση των βέλτιστων αποτελεσμάτων χρησιμοποιούμε την δοθείσα συνάρτηση **interest_points_visualization** με ορίσματα την εκάστοτε έγχρωμη εικόνα και τον αντίστοιχο $N \times 3$ πίνακα που μας επιστρέφει η **AngleDetect**:



(α') Sunflowers Angle Detection



(β') Caravaggio Angle Detection

Κατά τον πειραματισμό μας με τις τιμές των διαφόρων παραμέτρων προέκυψαν τα ακόλουθα συμπεράσματα: Μεγάλες τιμές της κλίμακας σ καθιστούν αδύνατο τον εντοπισμό μικρών

γωνιών. Ακόμα, η παράμετρος k επηρεάζει σημαντικά τον αριθμό των ανιχνευθέντων γωνιών καθώς υπεισέρχεται στον τύπο υπολογισμού του κριτηρίου γωνιότητας R . Καλύτερα αποτελέσματα λαμβάνουμε για μικρές τιμές του k (π.χ $k = 0.01$). Τέλος, καθοριστικό ρόλο στη ανίχνευση παίζει και το κατώφλι θ_{corn} . Συγκεκριμένα, όσο μειώνουμε την τιμή του τόσες περισσότερες γωνίες ανιχνεύουμε. Ωστόσο, για πολύ μικρές τιμές της παραμέτρου ανιχνεύονται λανθασμένα και σημεία που δεν είναι γωνίες αλλά εμφανίζουν παρόμοιες ιδιότητες, όπως μεγάλη μεταβολή στην φωτεινότητα.

2.2 Πολυκλιμακωτή Ανίχνευση Γωνιών

Στο μέρος αυτό θα επεκτείνουμε την παραπάνω μέθοδο για ανίχνευση γωνιών σε μία μόνο κλίμακα ώστε να ανιχνεύει γωνίες σε πολλαπλές κλίμακες και να επιστρέψει εκτός από τα σημεία που αντιστοιχούν σε γωνίες και την κλίμακα στην οποία ανιχνεύτηκαν. Η μέθοδος που θα υλοποιήσουμε (Harris-Laplacian) αποτελείται από δύο στάδια. Αρχικά, επιλέγουμε ορισμένα σημεία ενδιαφέροντος σε κάθε κλίμακα (αυτό γίνεται όπως πριν με χρήση της συνάρτησης **AngleDetect**). Στη συνέχεια γίνεται η τελική επιλογή των σημείων που παρουσιάζουν μέγιστο σε κάποια μετρική αμετάβλητη ως προς την κλίμακα, όπως επεξηγείται και παρακάτω. Όμοια με πριν κατασκευάζουμε την συνάρτηση **MultiScalarAngleDetect** η οποία δέχεται για ορίσματα την εικόνα σε grayscale, τις αρχικές κλίμακες s_0, r_0 διαφόρισης και ολοκλήρωσης, τον παράγοντα κλιμάκωσης s , ένα κατάλληλα επιλεγμένο κατώφλι θ_{corn} και τέλος τον αριθμό των κλιμάκων N .

2.2.1 Για κάθε κλίμακα (από ένα προκαθορισμένο σύνολο) εφαρμόζουμε τον απλό αλγόριθμο Harris για να βρούμε τις γωνίες για την κλίμακα αυτή. Οι προκαθορισμένες κλίμακες ολοκλήρωσης και διαφόρισης s_0, \dots, s_{N-1} και r_0, \dots, r_{N-1} είναι της μορφής:

$$\begin{aligned}\sigma_i &= s^i \sigma_0 \\ \rho_i &= s^i \rho_0\end{aligned}$$

για $i = 0, \dots, N - 1$ όπου s ο παράγοντας κλιμάκωσης και N ο αριθμός των κλιμάκων.

2.2.2 Στη συνέχεια επιλέγουμε την χαρακτηριστική κλίμακα για κάθε σημείο που ανιχνεύτηκε στο προηγούμενο βήμα. Αρχικά υπολογίζουμε την κανονικοποιημένη Λαπλασιανή της Γκαουσιανής για τις διαφορετικές κλίμακες διαφόρισης του προηγούμενου βήματος σύμφωνα με την σχέση:

$$|LoG(\mathbf{x}, \sigma_i)| = \sigma_i^2 |L_{xx}(\mathbf{x}, \sigma_i) + L_{yy}(\mathbf{x}, \sigma_i)|$$

όπου

$$L_{xx}(x, y, \sigma_i) = \frac{\partial^2 I_{\sigma_i}(x, y)}{\partial x^2} \quad \text{και} \quad L_{yy}(x, y, \sigma_i) = \frac{\partial^2 I_{\sigma_i}(x, y)}{\partial y^2}$$

Παρατηρούμε ότι η κανονικοποιημένη LoG είναι ισοδύναμη με την απόλυτη τιμή της συνέλιξης της εικόνας εισόδου I με την LoG, πολλαπλασιασμένη με το αντίστοιχο σ_i αφού:

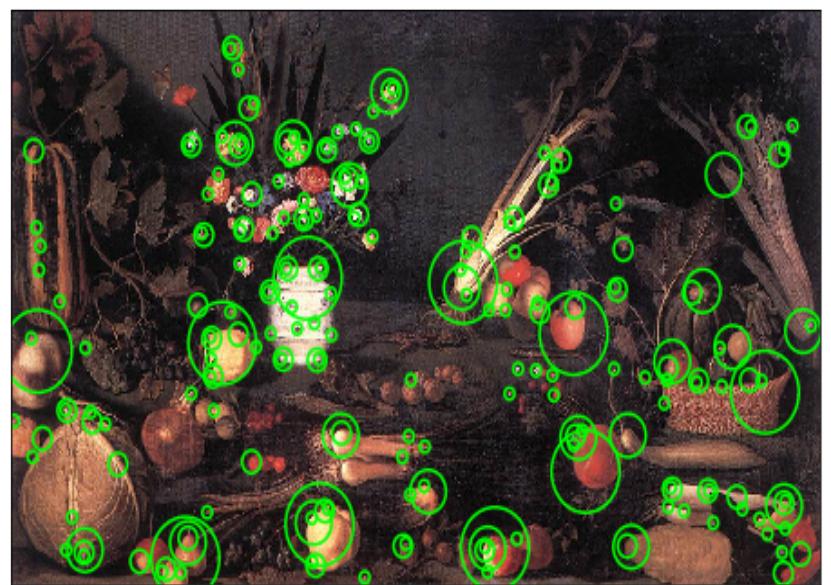
$$\begin{aligned}|\nabla^2 G_{\sigma_i} * I| &= \left| \left(\frac{\partial^2 G_{\sigma_i}}{\partial x^2} + \frac{\partial^2 G_{\sigma_i}}{\partial y^2} \right) * I \right| = \left| \frac{\partial^2 G_{\sigma_i}}{\partial x^2} * I + \frac{\partial^2 G_{\sigma_i}}{\partial y^2} * I \right| \Leftrightarrow \\ |\nabla^2 G_{\sigma_i} * I| &= \left| \frac{\partial^2 (G_{\sigma_i} * I)}{\partial x^2} + \frac{\partial^2 (G_{\sigma_i} * I)}{\partial y^2} \right| = \left| \frac{\partial^2 I_{\sigma_i}}{\partial x^2} + \frac{\partial^2 I_{\sigma_i}}{\partial y^2} \right| = |L_{xx}(\mathbf{x}, \sigma_i) + L_{yy}(\mathbf{x}, \sigma_i)|\end{aligned}$$

Έχοντας πλέον υπολογίσει την κανονικοποιημένη LoG απορρίπτουμε τα σημεία εκείνα για τα οποία η κλίμακα που ανιχνεύθηκαν δεν μεγιστοποιεί την LoG μετρική σε μια γειτονιά δύο διαδοχικών κλιμάκων. Για την LoG εφαρμόζουμε όπως και στο ερώτημα 1.2.1 άμεση υλοποίηση του πυρήνα της μέσω της συνάρτησης **meshgrid** της βιβλιοθήκης numpy και του αναλυτικού μαθηματικού τύπου ορισμού της. Στη συνέχεια, για τον υπολογισμό της κανονικοποιημένης LoG πραγματοποιούμε μέσω της **filter2D** της cv2 τη συνέλιξη της αρχικής εικόνας I με την LoG, πάροντας την απόλυτη τιμή του αποτελέσματος και πολλαπλασιάζουμε με το αντίστοιχο σ_i^2 .

Καλούμε λοιπόν τώρα τη συνάρτηση **MultiScalarAngleDetect** με τις ενδεικτικές τιμές της εκφώνησης ($\sigma = 2$, $\rho = 2.5$, $k = 0.05$, $\theta_{corn} = 0.005$, $s = 1.5$, $N = 4$). Στη συνέχεια, δοκιμάζουμε μικρές αποκλίσεις των τιμών αυτών μέχρις ότου επιτευχθεί η καλύτερη δυνατή ανίχνευση γωνιών. Για την παρουσίαση των βέλτιστων αποτελεσμάτων χρησιμοποιούμε την δοθείσα συνάρτηση **interest_points_visualization** με ορίσματα την εκάστοτε έγχρωμη εικόνα και τον αντίστοιχο $N \times 3$ πίνακα που μας επιστρέφει η **MultiScalarAngleDetect**:



(α') Sunflowers MultiScalarAngle Detection



(β') Caravaggio MultiScalarAngle Detection

Με την πολυκλιμακωτή ανίχνευση παρατηρούμε πως εντοπίζουμε ένα μεγαλύτερο αριθμό γωνιών σε σχέση με πριν, γεγονός που οφείλεται στην εξέταση πολλαπλών κλιμάκων. Έτσι, επιτυγχάνουμε ένα πιο ολοκληρωμένο αποτέλεσμα. Τα σημεία που ανιχνεύει ο αλγόριθμος δηλώνονται με κύκλους μεταβλητής ακτίνας, ανάλογης της εκάστοτε κλίμακας ανίχνευσης.

2.3 Ανίχνευση Blobs

Μία άλλη σημαντική κατηγορία σημείων ενδιαφέροντος είναι τα blobs τα οποία ορίζονται ως περιοχές με κάποια ομοιογένεια που διαφέρουν σημαντικά από την γειτονιά τους. Για την εύρεση τέτοιων περιοχών, σε αντιστοιχία με το κριτήριο γωνιότητας της μεθόδου Harris, γίνεται χρήση των μερικών παραγώγων δεύτερης τάξης της εικόνας και συγκεκριμένα η ορίζουσα του πίνακα Hessian [1]:

$$H(x, y) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix}$$

όπου

$$L_{xy} = \frac{\partial^2 I_\sigma(x, y)}{\partial x \partial y}$$

Οι παράγωγοι δεύτερης τάξης στην ουσία μετρούν την τοπική καμπυλότητα στο εκάστοτε σημείο (θεωρώντας την εικόνα ως επιφάνεια). Σε αντιστοιχία με την λογική του Harris Detector, τα ιδιοδιανύσματα δίνουν τις κατευθύνσεις της μέγιστης και της ελάχιστης καμπυλότητας και οι ιδιοτιμές το μέγεθος της καμπυλότητας. Ομοίως με το κριτήριο γωνιότητας ορίζουμε το κριτήριο σημαντικότητας:

$$R(x, y) = \det(H(x, y)) = L_{xx} \cdot L_{yy} - L_{xy}^2$$

και αναζητούμε τοπικά μέγιστα πάνω από ένα προκαθορισμένο κατώφλι. Στο μέρος αυτό ορίζουμε την συνάρτηση **BlobsDetect** η οποία δέχεται για ορίσματα την εικόνα σε grayscale και τις παραμέτρους σ και θ_{corn} ενώ επιστρέφει έναν πίνακα $N \times 3$ με τις δύο πρώτες στήλες να αποτελούν τις συντεταγμένες των ανιχνευθέντων σημείων και την τρίτη την κλίμακα στην οποία ανιχνεύθηκαν. Επιπλέον επιστρέφει έναν Boolean πίνακα με τιμές 1 στα σημεία ενδιαφέροντος και 0 αλλού ο οποίος χρησιμοποιείται μετέπειτα κατά την πολυκλιμακωτή ανίχνευση blobs.

2.3.1 Υπολογίζουμε τις μερικές παραγώγους δεύτερης τάξης της εικόνας L_{xx} , L_{xy} , L_{yy} για μια δεδομένη κλίμακα σ και για κάθε pixel εφαρμόζουμε το κριτήριο:

$$R(x, y) = \det(H(x, y)) \Leftrightarrow R(x, y) = L_{xx} \cdot L_{yy} - L_{xy}^2$$

Για τον υπολογισμό των μερικών παραγώγων κάνουμε χρήση της συνάρτησης **gradient** της βιβλιοθήκης numPy.

2.3.2 Επιλέγουμε ως σημεία ενδιαφέροντος, τα σημεία που αποτελούν τοπικά μέγιστα και έχουν τιμή μεγαλύτερη από ένα κατάλληλα ορισμένο κατώφλι, ακριβώς όπως και στην ανίχνευση γωνιών με την μέθοδο Harris. Η διαδικασία επιλογής σημείων με βάση τις δύο αυτές συνθήκες αναλύθηκε στην ενότητα 2.1.3.

Καλούμε λοιπόν τώρα τη συνάρτηση **BlobsDetect** με τις ενδεικτικές τιμές της εκφώνησης ($\sigma = 2$, $\theta_{corn} = 0.005$). Στη συνέχεια, δοκιμάζουμε μικρές αποκλίσεις των τιμών αυτών μέχρις ότου επιτευχθεί η καλύτερη δυνατή ανίχνευση blobs. Για την παρουσίαση των βέλτιστων αποτελεσμάτων χρησιμοποιούμε την δυθείσα συνάρτηση **interest_points_visualization** με ορίσματα την εκάστοτε έγχρωμη εικόνα και τον αντίστοιχο $N \times 3$ πίνακα που μας επιστρέφει η **BlobsDetect**:



(α') Sunflowers Blobs Detection



(β') Caravaggio Blobs Detection

Όπως και στην περίπτωση της ανίχνευσης γωνιών μίας κλίμακας, οι παραμέτροι σ και θ_{corn} επηρεάζουν τον εντοπισμό των blobs με ανάλογο τρόπο. Ωστόσο, εδώ πετυχαίνουμε καλύτερα αποτελέσματα για μεγαλύτερες τιμές της παραμέτρου θ_{corn} (π.χ $\theta_{corn} = 0.14$).

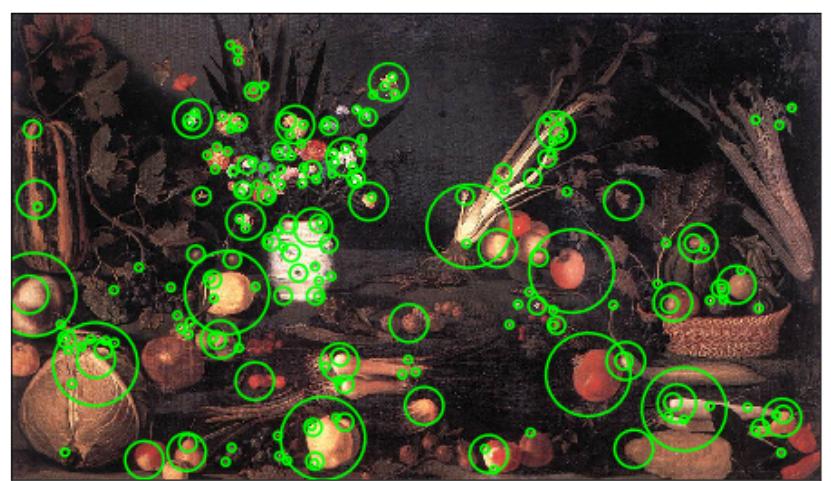
2.4 Πολυκλιμακωτή Ανίχνευση Blobs

Στο μέρος αυτό επεκτείνουμε την παραπάνω μέθοδο για ανίχνευση blobs σε μία μόνο κλίμακα ώστε να ανιχνεύει σημεία ενδιαφέροντος σε πολλαπλές κλίμακες και να επιστρέψει εκτός από τις συντεταγμένες τους και την κλίμακα στην οποία ανιχνεύθηκαν. Η μέθοδος που ακολουθούμε είναι ίδια με αυτήν της πολυκλιμακωτής ανίχνευσης γωνιών που εξηγήθηκε λεπτομερώς προηγουμένως. Η συνάρτηση **MultiScalarBlobsDetect** δέχεται για ορίσματα την εικόνα σε grayscale, τις παραμέτρους σ , s , N , θ_{corn} και υλοποιεί τα παραπάνω.

2.4.1 Για την παρουσίαση των βέλτιστων αποτελεσμάτων της πολυκλιμακωτής ανίχνευσης blobs καλούμε την συνάρτηση **interest_points_visualization** με ορίσματα την εκάστοτε έγχρωμη εικόνα και τον αντίστοιχο $N \times 3$ πίνακα που μας επιστρέφει η **MultiScalarBlobsDetect**:



(α') Sunflowers MultiBlobs Detection



(β') Caravaggio MultiBlobs Detection

Με την πολυκλιμακωτή ανίχνευση παρατηρούμε πως εντοπίζουμε ένα μεγαλύτερο αριθμό από blobs σε σχέση με πριν. Για παράδειγμα, ανιχνεύονται τα κέντρα των ηλιοτροπίων, γεγονός μη εφικτό με την ανίχνευση μονής κλίμακας. Έτσι, έχουμε ένα πιο πλήρες αποτέλεσμα. Και σε αυτή την περίπτωση, τα σημεία ανίχνευσης του αλγορίθμου αναπαριστώνται με κύκλους μεταβλητής ακτίνας, ανάλογης της εκάστοτε κλίμακας ανίχνευσης.

2.5 Επιτάχυνση με την χρήση Box Filters και Ολοκληρωτικών Εικόνων

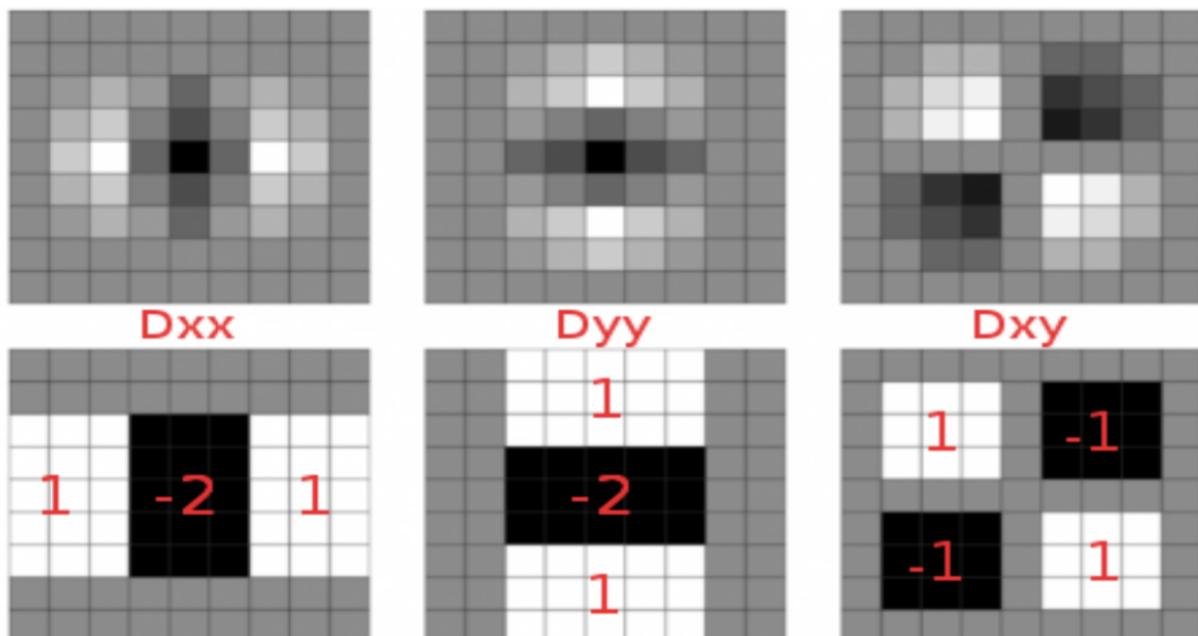
Στα προηγούμενα ερωτήματα εφαρμόσαμε την Hessian μέθοδο για την ανίχνευση των blobs. Ωστόσο, η διαδικασία αυτή είναι υπολογιστικά ακριβή καθώς ο υπολογισμός της Hessian για κάθε κλίμακα αντιστοιχεί στην συνέλιξη της εικόνας με αυξανόμενου μεγέθους φίλτρα. Για την επιτάχυνση της μεθόδου αυτής προτάθηκε στο [2] η προσέγγιση των φίλτρων 2ης παραγάγου με Box Filters, δηλαδή με φίλτρα που βασίζονται σε αθροίσματα ορθογώνιων περιοχών. Η επιλογή αυτή γίνεται γιατί η υλοποίηση τέτοιων αθροισμάτων πραγματοποιείται πολύ αποτελεσματικά με χρήση των Ολοκληρωτικών Εικόνων. Θεωρώντας μία εικόνα $I(x, y)$ μεγέθους $N \times M$ η ολοκληρωτική της εικόνα ορίζεται ως:

$$S(x, y) = \sum_{i \leq x} \sum_{j \leq y} I(i, j)$$

δηλαδή κάθε στοιχείο της ισούται με το άθροισμα όλων των στοιχείων της αρχικής εικόνας I που βρίσκονται πάνω και αριστερά από το στοιχείο αυτό. Με την χρήση της ολοκληρωτικής εικόνας οποιοδήποτε άθροισμα εντός ενός δοθέντος παραθύρου που οριοθετείται παραδείγματος χάριν από τα σημεία A, B, C και D υπολογίζεται άμεσα (με πολυπλοκότητα $O(1)$) σύμφωνα με την ακόλουθη σχέση:

$$\sum_{ABCD} I(i, j) = S_A + S_C - S_B - S_D$$

Μια οπτικοποίηση της προσέγγισης των φίλτρων 2ης παραγάγου με Box Filters παρουσιάζεται στις ακόλουθες εικόνες:



Επιλέγουμε να εφαρμόσουμε μια εναλλακτική αποδοτικότερη υλοποίηση από την παραπάνω, με 2 επικαλυπτόμενα υποπαράθυρα για τις δύο πρώτες περιπτώσεις (έναντι 3 όπως φαίνεται στις δύο πρώτες εικόνες) [5]. Το πρώτο υποπαράθυρο Γ_1 καλύπτει και τα 3 υποπαράθυρα της παραπάνω υλοποίησης και έχει βάρος 1. Το δέυτερο υποπαράθυρο Γ_2 που χρησιμοποιήσουμε ταυτίζεται με το ενδιάμεσο (μαύρο) παράθυρο του σχήματος, με την διαφορά ότι έχει βάρος ίσο με -3, αντί για -2 (εφόσον έχουμε ήδη λάβει μια φορά την περιοχή αυτή μέσω του υποπαραθύρου Γ_1). Τα φίλτρα αυτά εφαρμόζονται πάνω στην ολοκληρωτική εικόνα μετά από κατάλληλο shifting.

2.5.1 Για την διαχείριση των άκρων της εικόνας εφαρμόζουμε αρχικά padding ίσο με το μισό του μεγέθους του φίλτρου. Το padding της εικόνας γίνεται με mirroring, επαναλαμβάνοντας τα ακριανά pixels, δηλαδή τα pixels των ακμών της. Αυτό επιτυγχάνεται μέσω της συνάρτησης **pad** της βιβλιοθήκης numpy, θέτοντας το τρίτο όρισμά της ως “edge”. Έπειτα, υπολογίζουμε την ολοκληρωτική εικόνα της padded εκδοχής της αρχικής εικόνας, μέσω της συνάρτησης **cumsum**.

2.5.2 Ορίζουμε κατάλληλες παραμέτρους για τις διαστάσεις των φίλτρων καθώς και για το shifting και καλούμε την συνάρτηση **ShiftedImage**. Η συνάρτηση αυτή δέχεται ως ορίσματα την ολοκληρωτική εικόνα, τις παραμέτρους για τις διαστάσεις των φίλτρων καθώς και τις παραμέτρους για το image shifting ενώ επιστρέφει τα S_A, S_B, S_C και S_D τα οποία χρησιμοποιούνται για τον υπολογισμό των L_{xx}, L_{xy}, L_{yy} .

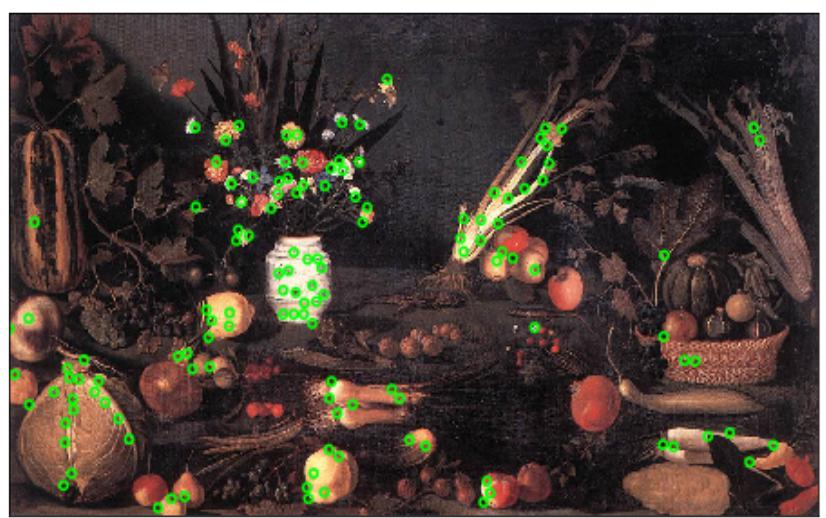
2.5.3 Βρίσκουμε τα τοπικά μέγιστα του χριτηρίου $R(x, y)$ με τον ίδιο τρόπο που εξηγήσαμε στα προηγούμενα ερωτήματα.

$$R(x, y) = L_{xx}(x, y)L_{yy}(x, y) - (0.9L_{xy})^2$$

Καλούμε λοιπόν τώρα τη συνάρτηση **BoxFilter** με τις ενδεικτικές τιμές της εκφώνησης ($\sigma = 2$, $\theta_{corn} = 0.005$). Στη συνέχεια, δοκιμάζουμε μικρές αποκλίσεις των τιμών αυτών μέχρις ότου επιτευχθεί η καλύτερη δυνατή ανίχνευση blobs. Για την παρουσίαση των βέλτιστων αποτελεσμάτων χρησιμοποιούμε την δοθείσα συνάρτηση **interest_points_visualization** με ορίσματα την εκάστοτε έγχρωμη εικόνα και τον αντίστοιχο $N \times 3$ πίνακα που μας επιστρέφει η **BoxFilter**:

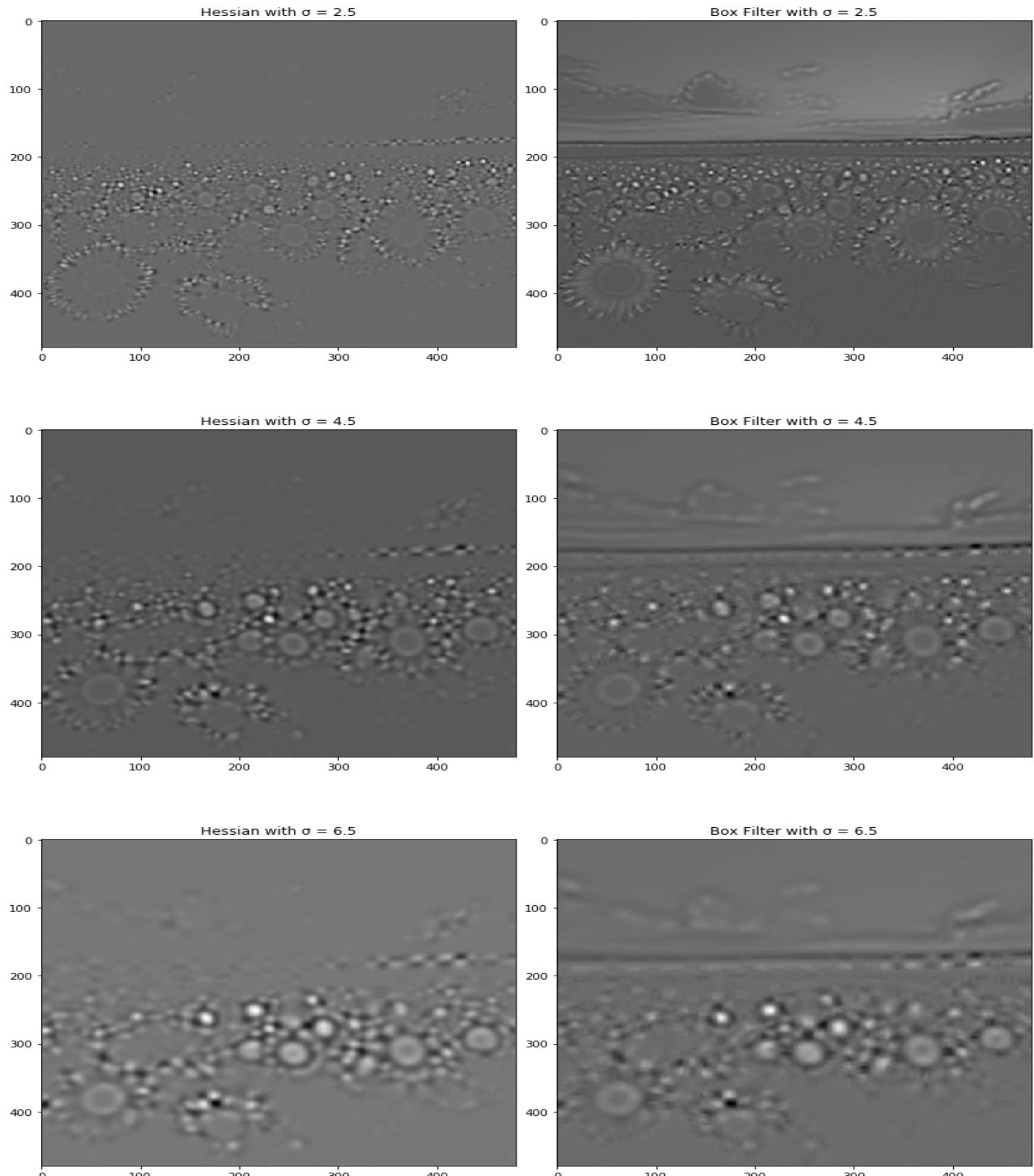


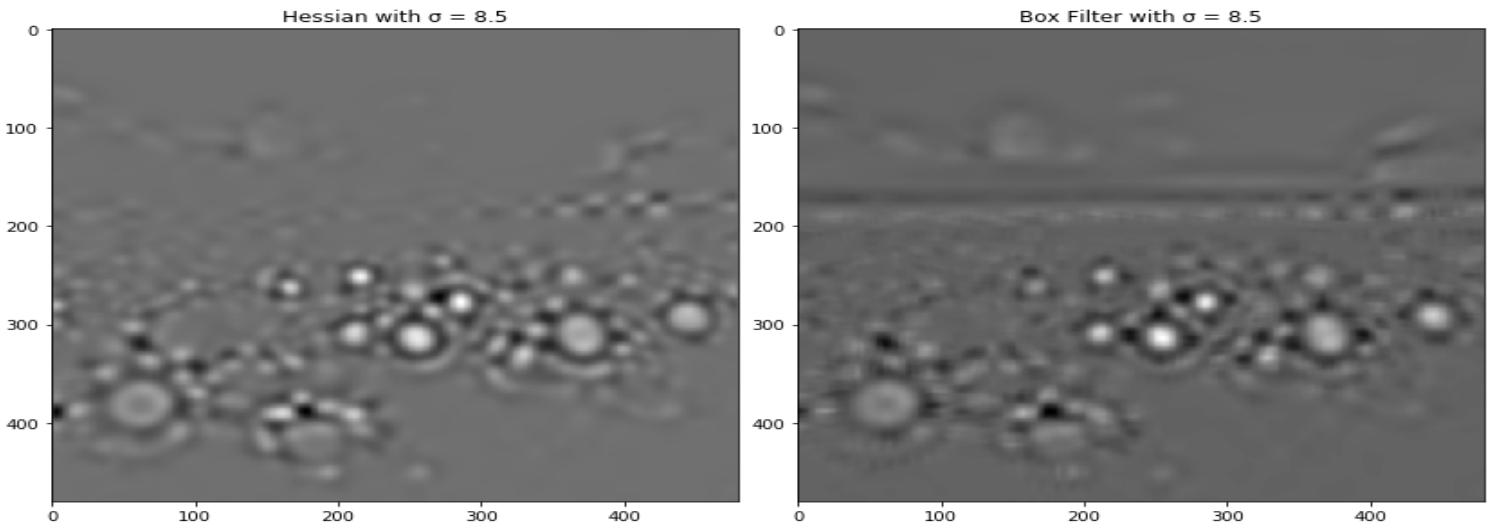
(α') Sunflowers BoxFilter Detection



(β') Caravaggio BoxFilter Detection

Οπτικοποιούμε (ως εικόνα) το χριτήριο R της απλής Hessian μεθόδου και το παραπάνω προσεγγιστικό χριτήριο R για ορισμένες κλίμακες σ . Συγκεκριμένα επιλέγουμε τις κλίμακες $\sigma = 2.5, 4.5, 6.5, 8.5$. Τα αποτελέσματα που προκύπτουν και από τις δύο μεθόδους είναι τα ακόλουθα:





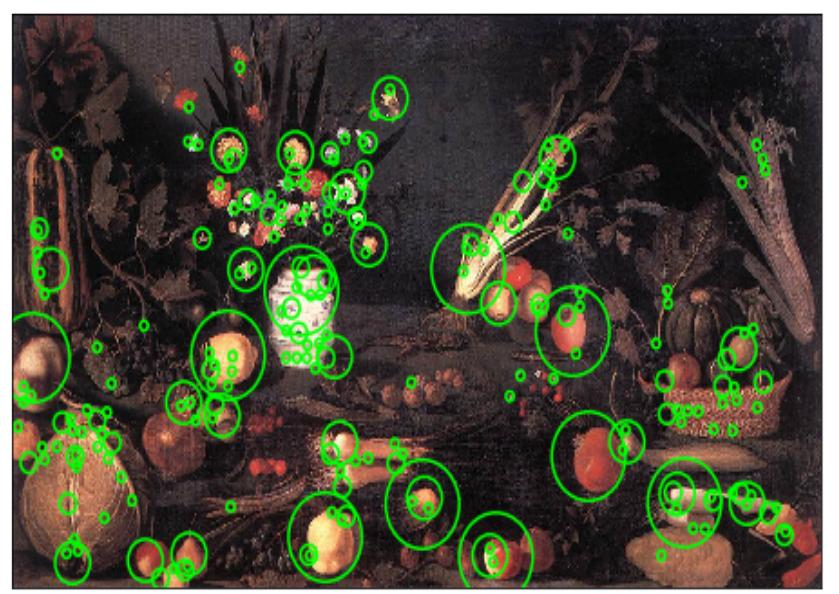
Παρατηρούμε πως η μέθοδος με τα Box Filters δίνει αρκετά καλύτερα αποτελέσματα για μικρή κλίμακα σ . Όσο η κλίμακα μεγαλώνει τόσο η ανάλυση των εικόνων χειροτερεύει μιας και γίνεται εστίαση σε όλο και πιο γενικά χαρακτηριστικά, με αποτέλεσμα οι διαφορές μεταξύ των δύο μεθόδων να γίνονται αρκετά δυσδιάχριτες.

2.5.4 Σαν τελευταίο βήμα του μέρους αυτού επεκτείνουμε πάλι τον αλγόριθμο ώστε να ανιχνεύει σημεία ενδιαφέροντος σε πολλαπλές κλίμακες. Η διαδικασία που ακολουθείται είναι ίδια με αυτήν της πολυκλιμακωτής ανίχνευσης γωνιών και blobs που εξηγήθηκε λεπτομερώς στην υποενότητα 2.2.2. Για τον λόγο αυτό δεν αναλύεται περαιτέρω.

Για την απεικόνιση των καλύτερων αποτελεσμάτων πολυκλιμακωτής ανίχνευσης blobs με την μέθοδο των Box filters καλούμε την συνάρτηση **interest_points_visualization** με ορίσματα την εκάστοτε έγχρωμη εικόνα και τον αντίστοιχο $N \times 3$ πίνακα που μας επιστρέφει η **MultiScalarBoxFilter**:



(α') Sunflowers MultiScalarBoxFilter Detection



(β') Caravaggio MultiScalarBoxFilter Detection

Μέρος 3: Εφαρμογές σε Ταίριασμα και Κατηγοριοποίηση Εικόνων με Χρήση Τοπικών Περιγραφητών στα Σημεία Ενδιαφέροντος

Τα σημεία ενδιαφέροντος που ανιχνεύσαμε στο Μέρος 2, δίνουν μια εκτίμηση περιοχών οι οποίες περιέχουν σημαντικά χαρακτηριστικά της εικόνας. Από τις περιοχές αυτές εξάγουμε τώρα τοπικούς περιγραφητές (local descriptors), που κωδικοποιούν μια γειτονιά (με ακτίνα που εξαρτάται από την κλίμακα) γύρω από τα σημεία ενδιαφέροντος. Οι τοπικοί περιγραφητές που χρησιμοποιούμε είναι οι ακόλουθοι:

- SURF (Speed Up Robust Features) [1]: Οι τοπικοί περιγραφητές SURF για κάθε σημείο ενδιαφέροντος προκύπτουν από τα εξής τρία βήματα:
 1. Εύρεση των κατευθυντικών παραγώγων με την χρήση των Haar Wavelets
 2. Εύρεση της κατεύθυνσης για μία περιοχή γύρω από το σημείο για εξαγωγή περιστροφικά ανεξάρτητων περιγραφητών.
 3. Υπολογισμός ενός 64-διάστατου διανύσματος χαρακτηριστικών για ένα τετραγωνικό παράθυρο με μέγεθος που εξαρτάται από την κλίμακα και κατεύθυνση αυτή που υπολογίστηκε προηγουμένως.
- HOG (Histogram of Oriented Gradients) [1]: Τα HOGs παρέχουν μία πυκνή επικαλυπτόμενη περιγραφή των περιοχών μίας εικόνας και υπολογίζονται σε ένα πυκνό πλέγμα ομοιόμορφα κατανεμημένων κελιών. Με τον τρόπο αυτό αναδεικνύουν την πληροφορία για το τοπικό σχήμα κωδικοποιώντας τις κατεύθυνσεις της κλίσης της εικόνας σε ιστογράμματα, ουσιαστικά κωδικοποιώντας ‘χοντρικά’ πληροφορία για τις ακμές των εικόνων. Τα βασικά βήματα της μεθόδου αυτής είναι τα ακόλουθα:
 1. Υπολογισμός Παραγώγου
 2. Χωρική Διαμέριση της Εικόνας
 3. Δημιουργία Ιστογραμμάτων Κατεύθυνσης
 4. Κανονικοποίηση Ιστογραμμάτων

Ο τελικός περιγραφητής αντιστοιχεί στο καθολικό ιστόγραμμα που προκύπτει από την συνένωση (concatenation) όλων των τοπικών ιστογραμμάτων.

3.1 Ταίριασμα Εικόνων υπό Περιστροφή και Αλλαγή Κλίμακας

Στο μέρος αυτό εξετάζουμε την ικανότητα εύρεσης της περιστροφής και της κλίμακας με την χρήση των ανιχνευτών σημείων ενδιαφέροντος που υλοποιήσαμε και τους παραπάνω τοπικούς περιγραφητές. Συγκεκριμένα, για την διεξαγωγή του πειράματος, χρησιμοποιούμε 3 εικόνες που έχουν υποστεί 20 παραμορφώσεις (5 περιστροφές, 4 κλίμακώσεις). Θεωρούμε μια εικόνα ως αναφορά και προσπαθούμε να αντιστοιχίσουμε του τοπικούς περιγραφητές της με αυτούς των υπόλοιπων παραμορφωμένων εικόνων. Η διαδικασία αυτή ονομάζεται ταίριασμα (matching) και υλοποιείται με εφαρμογή του αλγορίθμου RANSAC, ο οποίος αναλύεται παρακάτω.

Ο RANSAC [1] είναι μια επαναληπτική μέθοδος που εκτιμά τις παραμέτρους ενός μαθηματικού μοντέλου (π.χ. ευθεία ή μετασχηματισμό) από ένα σύνολο δεδομένων που περιέχει outliers. Το

μαθηματικό μοντέλο στην περίπτωσή μας, είναι ο μετασχηματισμός ομοιότητας και το σύνολο των inliers είναι οι ‘καλές’ αντιστοιχίσεις. Τα βήματα της μεθόδου συνοψίζονται ως εξής:

1. Επιλέγουμε τυχαία σε κάθε επανάληψη ένα μικρό σύνολο (π.χ 5) αντιστοιχίσεων και εφαρμόζουμε τον αλγόριθμο DLT (Direct Linear Transformation) [3] για να κάνουμε μια εκτίμηση των παραμέτρων του μετασχηματισμού.
2. Εφαρμόζουμε το μετασχηματισμό αυτό σε όλο το σύνολο αντιστοιχίσεων και κρατάμε ένα υποσύνολο αυτών (inliers) που δεν ξεπερνάει κάποιο κατώφλι απόστασης (παράμετρος ανοχής).
3. Διαλέγουμε τον μετασχηματισμό (εκτίμηση της περιστροφής και της κλίμακας) με τα περισσότερα inliers, που είναι και οι ζητούμενες ποιοτικές αντιστοιχίσεις.

Στην παρούσα εργαστηριακή άσκηση δεν ασχολούμαστε με τις υλοποιήσεις των περιγραφητών καθώς μας δίνονται έτοιμοι στις συναρτήσεις **featuresSURF(I,points)**, **featuresHOG(I,points)** για SURF και HOG αντίστοιχα. Για την εκτίμηση της περιστροφής και της κλίμακας των εικόνων γίνεται χρήση της έτοιμης συνάρτησης **matching_evaluation** στην οποία δίνουμε κάθε φορά ως είσοδο έναν συνδυασμό ανιχνευτών (AngleDetect, MultiScalarAngleDetect, BlobsDetect, MultiScalarBlobsDetect, MultiScalarBoxFilter) και περιγραφητών (SURF, HOG). Η συνάρτηση αυτή επιστρέφει το μέσο σφάλμα εκτίμησης κλίμακας και το μέσο σφάλμα εκτίμησης γωνίας περιστροφής για κάθε ένα από τα 3 ζεύγη εικόνων (αρχική και παραμορφωμένη) που χρησιμοποιούμε. Για την εξαγωγή των αντίστοιχων αποτελεσμάτων τροποποιούμε κατάλληλα το αρχείο **test_matching_evaluation.py** έτσι ώστε να τυπώνει για όλες τις εικόνες τα Avg. Scale και Theta Errors για κάθε δυνατό συνδυασμό (συνολικά 10).

	Detectors									
	Angle		MultiAngle		Blobs		MultiBlobs		MultiBoxFilter	
Avg. Error	Surf	Hog	Surf	Hog	Surf	Hog	Surf	Hog	Surf	Hog
Scale Error 1	0.002	0.132	0.001	0.324	0.005	0.099	0.001	0.177	0.001	0.189
Theta Error 1	0.156	16.429	0.112	13.595	1.389	21.464	0.065	19.960	0.151	13.826
Scale Error 2	0.002	0.206	0.004	0.099	0.060	0.158	0.004	0.249	0.004	0.071
Theta Error 2	0.185	17.362	0.257	12.555	2.101	26.448	0.134	18.149	0.190	14.953
Scale Error 3	0.003	0.156	0.001	0.085	0.017	0.281	0.002	0.219	0.002	0.224
Theta Error 3	0.324	18.316	0.157	16.364	4.392	23.843	0.114	17.404	0.135	18.918

Σχολιασμός Αποτελεσμάτων

Παρατηρούμε ότι ο περιγραφητής Hog δίνει σε όλες τις περιπτώσεις χειρότερα μέσα σφάλματα από ότι ο Surf. Συγκεκριμένα, στην περίπτωση του Avg. Scale Error και οι δύο περιγραφητές δίνουν μικρά σφάλματα (δεν υπερβαίνουν το 0.281 σε καμία περίπτωση), με τον Hog ωστόσο να επιστρέφει κατά βάση τιμές που είναι τουλάχιστον 10 φορές μεγαλύτερες. Από την άλλη, η διαφορά στο Avg. Theta Error είναι αρκετά μεγάλη μεταξύ των δύο περιγραφητών, με τον Hog να δίνει σφάλματα δύο τάξεων μεγέθους μεγαλύτερα στην πλειοψηφία των περιπτώσεων. Οι αποκλίσεις των τιμών αυτών οφείλονται στην διαφορετική υλοποίηση των 2 περιγραφητών (π.χ ο Surf είναι scale/rotation invariant). Μπορούμε να παρατηρήσουμε ακόμα, πως οι πολυχλιμακωτοί ανιχνευτές αποδίδουν γενικά καλύτερα σε σχέση με τους αντίστοιχους ανιχνευτές μίας κλίμακας, δίνοντας μικρότερα Avg. Scale και Theta Errors. Αυτό είναι αναμενόμενο καθώς εξετάζουν πολλαπλές κλίμακες ενώ παράλληλα επιτυγχάνουν μια βελτιωμένη αντίληψη του χώρου.

3.2 Κατηγοριοποίηση Εικόνων

Στο μέρος αυτό αξιολογούμε την επίδοση και την καταλληλότητα των διαφόρων ανιχνευτών και περιγραφητών σε ένα τυπικό πρόβλημα κατηγοριοποίησης εικόνων. Συγκεκριμένα, χρησιμοποιούμε ένα σύνολο εικόνων από τη βάση Pascal VOC2005. Κάθε εικόνα ανήκει σε μια από τις τρεις ακόλουθες κλάσεις: αυτοκίνητο, άνθρωπος και ποδήλατο. Στόχος μας είναι η κατηγοριοποίηση της κάθε μίας στη σωστή κλάση αξιοποιώντας σαν χαρακτηριστικά αναγνώρισης τους περιγραφητές. Ένας τέτοιος στόχος απαιτεί μία ενιαία περιγραφή ίδιας διάστασης για κάθε εικόνα. Για τον λόγο αυτό χρησιμοποιείται η προσέγγιση των Bag of Visual Words (BoVW) [1]. Ουσιαστικά, αντί να χρησιμοποιούμε απευθείας τους τοπικούς περιγραφητές, υπολογίζουμε ιστογράμματα εμφάνισης συγκεκριμένων οπτικών προτύπων στο σύνολο της εικόνας. Τα οπτικά πρότυπα αυτά, ή οπτικές λέξεις, αποτελούν το λεξικό και εξάγονται από ένα σύνολο των τοπικών περιγραφών. Τα βήματα για την κατασκευή της (BoVW) αναπαράστασης συνοψίζονται παρακάτω:

1. Δημιουργία του οπτικού λεξικού. Για τον προσδιορισμό των οπτικών λέξεων εφαρμόζουμε στους τοπικούς περιγραφητές, που αντιστοιχούν στο σύνολο των εικόνων εκπαίδευσης (train set), έναν αλγόριθμο συσταδοποίησης (π.χ k-means). Το πλήθος των συστάδων που προκύπτουν ορίζεται έτσι ώστε να είναι μεγάλο (π.χ. 500), ενώ τα κέντρα τους αποτελούν τις οπτικές λέξεις του λεξικού.
2. Υπολογίζουμε την ευκλείδεια απόσταση του κάθε τοπικού περιγραφητή από τις οπτικές λέξεις. Στη συνέχεια, αντιστοιχούμε στους τοπικούς περιγραφητές τις οπτικές λέξεις με την μικρότερη ευκλείδεια απόσταση. Το ιστόγραμμα κάθε εικόνας προκύπτει από τη συχνότητα εμφάνισης των οπτικών προτύπων του λεξικού που αντιστοιχίστηκαν στους τοπικούς της περιγραφητές.
3. Κανονικοποιούμε το ιστόγραμμα διαιρώντας με την l_2 νόρμα

3.2.1 Αρχικά εξάγουμε τα χαρακτηριστικά όλων των εικόνων της βάσης με την έτοιμη συνάρτηση **FeatureExtraction** η οποία δέχεται σαν παραμέτρους έναν συγκεκριμένο συνδυασμό detector και descriptor.

3.2.2 Στην συνέχεια πρέπει να γίνει ο διαχωρισμός των εικόνων στα σύνολα train και test καθώς και η δημιουργία ετικετών (label) που θα δείχνουν την κλάση στην οποία θα ανήκει κάθε εικόνα. Για τον σκοπό αυτό χρησιμοποιούμε την συνάρτηση `createTrainTest` που μας δίνεται, η οποία επιστρέφει μεταξύ άλλων δύο λίστες, `data_train,data_test`, που περιέχουν σε μορφή πίνακα τους περιγραφητές των εικόνων του συνόλου εκπαίδευσης και επαλήθευσης αντίστοιχα.

3.2.3 Κατασκευή αναπαράστασης Bag Of Visual Words (**Bonus Ερώτημα**)

Στο μέρος αυτό υλοποιούμε την δική μας αναπαράσταση Bag Of Visual Words η οποία βρίσκεται στο αρχείο **MyBagOfWords.py**. Αρχικά, πραγματοποιούμε την συνένωση όλων των περιγραφητών (ιδίου τύπου) του συνόλου εκπαίδευσης σε έναν πίνακα χαρακτηριστικών ανεξάρτητα από την κλάση στην οποία ανήκουν. Αυτό γίνεται με την συνάρτηση `vstack` της βιβλιοθήκης `numpy`. Στη συνέχεια εφαρμόζουμε τον αλγόριθμο `kmeans` σε ένα τυχαίο υποσύνολο του πίνακα των περιγραφητών (της τάξης του 50 % του συνολικού μεγέθους) με αριθμό κέντρων (οπτικών λέξεων) ίσο με 500. Η τυχαία επιλογή ενός δείγματος του πίνακα γίνεται με χρήση της `np.random.choice` ενώ η υλοποίηση του αλγορίθμου συσταδοποίησης βρίσκεται στην συνάρτηση με όνομα `kmeans` η οποία βασίζεται στην έτοιμη **KMeans** της βιβλιοθήκης `sklearn`. Αφού εξάγουμε τις οπτικές λέξεις κατασκευάζουμε για κάθε εικόνα του συνόλου εκπαίδευσης αλλά και επαλήθευσης τα ιστογράμματα. Για την δημιουργία των ιστογραμμάτων των εικόνων του train set διατρέχουμε μία προς μία όλες τις εικόνες που ανήκουν σε αυτό και υπολογίζουμε για κάθε τοπικό περιγραφητή της την ευκλείδια απόστασή του από όλα τα κέντρα (clusters) που βρήκαμε προηγουμένως. Αυτό γίνεται με χρήση της συνάρτησης `euclidean` του πακέτου `distances` της βιβλιοθήκης `scipy`. Αφού υπολογίσουμε όλες τις αποστάσεις κρατάμε για κάθε περιγραφητή το κέντρο από το οποίο απείχε λιγότερο. Για την κατασκευή του ιστογράμματος, το οποίο προκύπτει από την συχνότητα εμφάνισης των οπτικών λέξεων του λεξικού στην αντίστοιχη εικόνα, χρησιμοποιούμε την έτοιμη συνάρτηση `histogram` της βιβλιοθήκης `numpy`. Τελευταίο βήμα αποτελεί η κανονικοποίηση κάθε ιστογράμματος με βάση την L2 νόρμα, για την οποία αξιοποιούμε την συνάρτηση `linalg` της `numpy`. Ακολουθούμε την ίδια ακριβώς διαδικασία και για το test set. Τελικά, η συνάρτηση `MyBagOfWords` επιστρέφει δύο πίνακες, `train_hist` και `test_hist`, οι οποίοι περιέχουν τα κανονικοποιημένα ιστογράμματα όλων των εικόνων που ανήκουν στα αντίστοιχα σύνολα train και test.

3.2.4 Το τελικό στάδιο αποτελείται από την κατηγοριοποίηση των εικόνων με βάση την BoVW αναπαράσταση. Για την κατηγοριοποίηση χρησιμοποιείται ενάς SVM Support Vector Machine ταξινομητής κατάλληλα προσαρμοσμένος για πολλαπλές κλάσεις. Η όλη διαδικασία υλοποιείται με τη συνάρτηση `svm`, η οποία επιστρέφει το αποτέλεσμα της αναγνώρισης καθώς και το συνολικό ποσοστό επιτυχίας. Χρησιμοποιώντας τις προτεινόμενες τιμές παραμέτρων ($\sigma = 2$, $\rho = 2.5$, $k = 0.05$, $\theta_{corn} = 0.005$, $s = 1.5$, $N = 4$) εξάγουμε τα ποσοστά ακριβείας για κάθε συνδυασμό πολυχλιμακωτού ανιχνευτή και περιγραφητή, τόσο για την έτοιμη υλοποίηση `BagOfWords` όσο και για την δική μας. Τα αποτελέσματα παρουσιάζονται στον ακόλουθο πίνακα:

	Detectors					
	MultiAngle		MultiBlobs		MultiBoxFilter	
	Surf	Hog	Surf	Hog	Surf	Hog
BagOfWords Accuracy	60.690 %	66.483 %	57.655 %	65.241 %	56.690 %	64.138 %
MyBagOfWords Accuracy	59.448 %	65.517 %	56.276 %	66.207 %	55.172 %	63.586 %

Σχολιασμός Αποτελεσμάτων

Παρατηρούμε πως για όλους του πολυκλιμακωτούς ανίχνευτές, ο περιγραφητής Hog παράγει καλύτερα ποσοστά ακριβείας σε σχέση με τον SURF. Συγκεκριμένα, σε κάθε περίπτωση, ο Hog αποδίδει καλύτερα κατά ένα ποσοστό 5 – 10%, με την μεγαλύτερη απόκλιση να εμφανίζεται στην περίπτωση του MultiBlobs (7,586%). Μπορούμε λοιπόν, να συμπεράνουμε πως η αποδοτικότερη μέθοδος ανίχνευσης για το πρόβλημα του Image Classification είναι η ανίχνευση γωνιών/blobs με χρήση περιγραφητή Hog. Μάλιστα, από ότι φαίνεται στον πίνακα, λαμβάνουμε το μεγαλύτερο ποσοστό ακριβείας (66.483%) για την MultiAngle, αν και παρατηρούμε πως και στις μενόδους ανίχνευσης blobs τα ποσοστά είναι εξίσου υψηλά. Έτσι, δεν μπορούμε να αποφανθούμε με βεβαιότητα ποια είναι η βέλτιστη μέθοδος. Αυτό οφείλεται στο γεγονός ότι πραγματοποιείται μια τυχαία επιλογή features στην BoVW (συναρτήσεις `random.choice` της βιβλιοθήκης `numpy`, `kmeans`), με αποτέλεσμα τα ποσοστά που εξάγουμε να μην είναι καθορισμένα αλλά να διαφέρουν για κάθε νέα εκτέλεση του κώδικα.

Αναφορικά με την δική μας υλοποίηση (MyBagOfWords), τα ποσοστά ακριβείας είναι παρεμφερή με αυτά της έτοιμης (BagOfWords) που δόθηκε στις διευκρινίσεις της άσκησης. Οι διαφορές είναι της τάξης του 1% και είναι πιθανό να οφείλονται απλά και μόνο στην τυχαία εξαγωγή του συνόλου εκπαίδευσης και του συνόλου επαλήθευσης, όπως εξηγήθηκε προηγουμένως. Φυσικά, μπορεί και να αποτελούν συνέπεια διαφορετικής υλοποίησης της μενόδου BoW. Τέλος, αξίζει να αναφερθεί πως τόσο τα ποσοστά που εξάγαμε μέσω της δικής μας υλοποίησης όσο και αυτά που εξάγαμε μέσω της έτοιμης, δεν υπερβαίνουν σε καμία περίπτωση το 67%. Αυτό συμβαίνει διότι δεν λάβαμε υπόψην μας κάποιου είδους χωρική πληροφορία αλλά επικεντρωθήκαμε μόνο στη συχνότητα εμφάνισης συγκεκριμένων features (οπτικών λέξεων). Το γεγονός αυτό αποτελεί το βασικότερο μειονέκτημα μιας τέτοιας αναπαράστασης και καθιστά την μέθοδο BoW αναποτελεσματική για μεγάλο εύρος σύνθετων προβλημάτων κατηγοριοποίησης. Ένας ενδεικνύμενος τρόπος αντιμετώπισης του προβλήματος αυτού είναι η χρήση χωρικών πειραμίδων (spatial pyramids [4]). Σύμφωνα με την προσέγγιση αυτή, η εικόνα υποδιαιρείται σε επιμέρους τμήματα σε κάθε επίπεδο της πυραμίδας και ένας περιγραφητής BoVW εξάγεται σε κάθε τμήμα της εικόνας.

Αναφορές

- [1] Μαραγκός Π. *Ανάλυση Εικόνων και Όραση Υπολογιστών*. Ε.Μ.Π., 2014
- [2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.
- [3] R. Hartley and A. Zisserman, *Multiple-View Geometry in Computer Vision* Cambridge University Press, 2003.
- [4] S. Lazebnik and J. Ponce, “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories”, in *Proc. CVPR*, 2006.
- [5] [https://www.researchgate.net/publication/282055016_An_Analysis_of_the_SURF_Method?fbclid=IwAR1Wou3llgcBTeFxQMsOItW3E0dYfTGJe0YmP1UWuzzky7tdonY4fK3RKWU](https://www.researchgate.net/publication/282055016_An_Analysis_of_the_SURF_Method)
- [6] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html
- [7] <https://docs.python.org/3.7/?fbclid=IwAR0cf6f4hf6iaTp-0j212UNwc5wdnqr2d0yboEVL1Wq3R6LMEFCrELcMPMg>