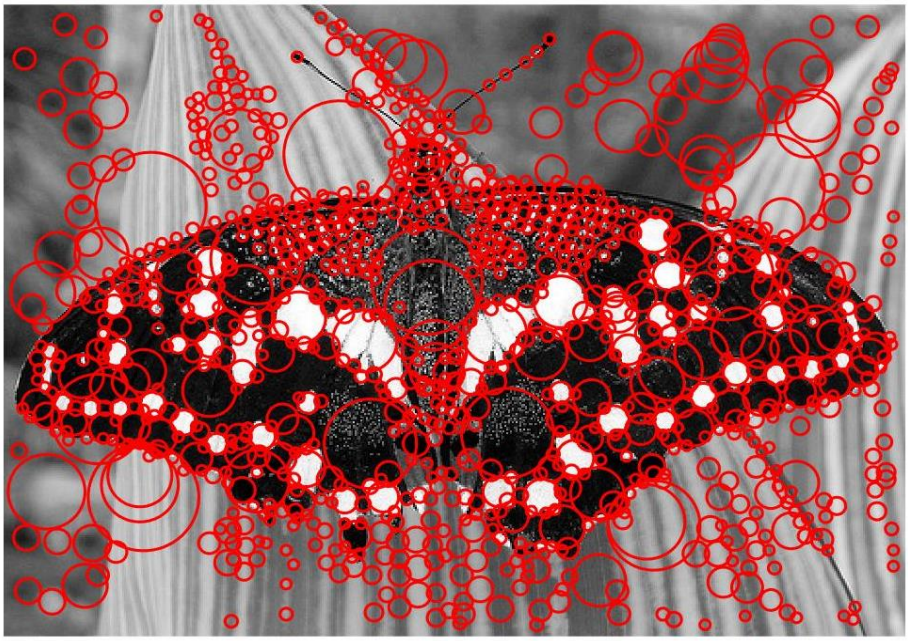
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ψηφιακή Επεξεργασία Εικόνας

Εργασία 3η – Εαρινό Εξάμηνο 2020/2021

Scale-Invariant Feature Transform (SIFT)



Καβελίδης Φραντζής Δημήτριος – ΑΕΜ 9351

Επικοινωνία: kavelids@ece.auth.gr

23/5/2021

Θέμα:

Η δεύτερη εργασία του μαθήματος αφορά την υλοποίηση του **SIFT detector.** Συγκεκριμένα υλοποιείται:

1. Υπολογισμός των space scales και των Difference-of-Gaussians (DoGs) των εικόνων εισόδου.
2. Εντοπισμός και φιλτράρισμα των salient keypoints.

Στόχος του αλγορίθμου είναι η εξαγωγή εύρωστων χαρακτηριστικών (features) από τις εικόνες βρίσκοντας και περιγράφοντας σημεία εύρωστα-ανθεκτικά (robust) στον θόρυβο (keypoints). Για αυτό το λόγο, ο αλγόριθμος λειτουργεί με αλλεπάλληλα blurs στην αρχική εικόνα μέσω ενός φίλτρου και στη συνέχεια τον υπολογισμό των DoGs και την εύρεση σε αυτά των σημείων που επηρεάστηκαν λιγότερο από το φιλτράρισμα. Προτάθηκε από τον David G. Lowe το 1999 [1] και στη συνέχεια το 2004 [2] με μία πιο εμπεριστατωμένη ανάλυση στην οποία παρουσίασε και τα heuristics του. Στη βιβλιογραφία, πολλές φορές ο αλγόριθμος αυτός χαρακτηρίζεται και ως μετασχηματισμός επειδή μετασχηματίζει τα δεδομένα της εικόνας σε αναλλοίωτες ως προς την κλιμάκωση συντεταγμένες σε σχέση με τα τοπικά χαρακτηριστικά της εικόνας. Επιπλέον, είναι ένας αλγόριθμος που χρησιμοποιεί μεγάλο πλήθος παραμέτρων, το οποίο τον καθιστά σε μεγάλο βαθμό σε σχέση με τους άλλους αλγόριθμους εξαγωγής χαρακτηριστικών ως **ευρετικό** αλγόριθμό [3], δηλαδή η τρέχουσα γνώση μας είναι ανεπαρκής για να ορίσει έναν συστηματικό τρόπο με τον οποίο θα πρέπει να οριστούν αυτές οι παράμετροι. Συνεπώς, οι επιλογές των παραμέτρων βρίσκονται πειραματικά, καθώς τα αποτελέσματα εξαρτώνται από πολλούς παράγοντες.

Ανάλυση:

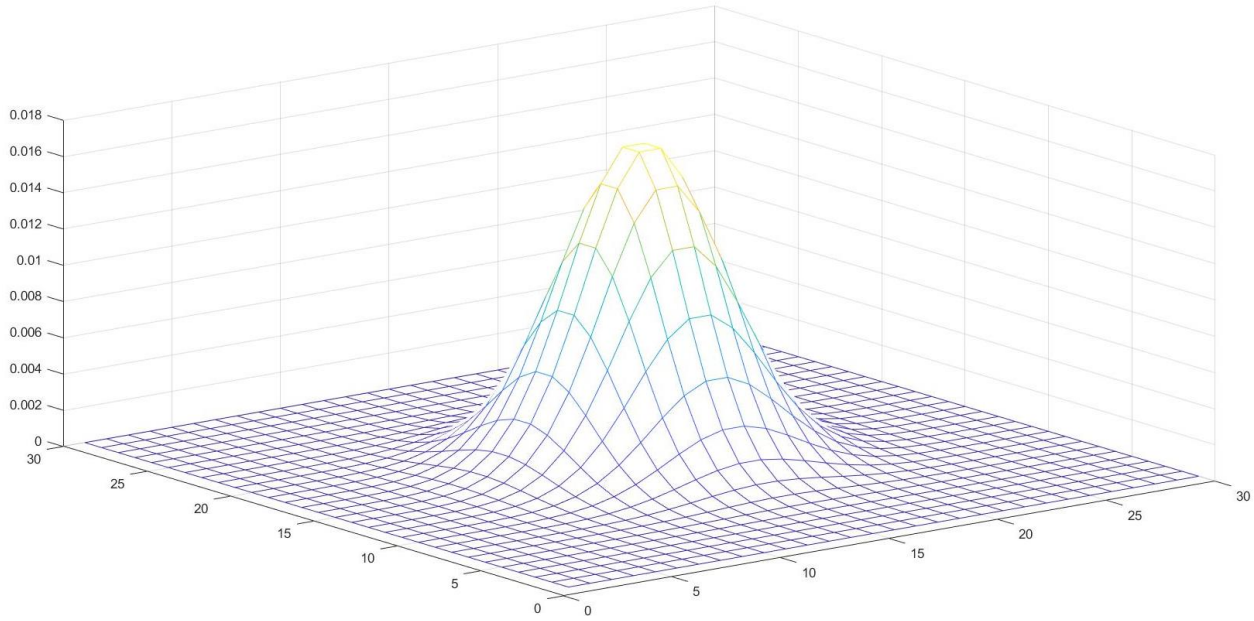
1. Gaussian Space Scales & Difference-of-Gaussians (DoGs)

Για τις ανάγκες του αλγορίθμου, φτιάχνουμε πρώτα την συνάρτηση **my2DGaussianFilter** η οποία επιστρέφει ένα Γκαουσιανό φίλτρο δύο διαστάσεων μεγέθους KxK το οποίο είναι κανονικοποιημένο και κεντραρισμένο γύρω από το (0,0), χρησιμοποιώντας για την υλοποίηση τον γνωστό τύπο:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{(-(x^2+y^2)/2\sigma^2)}$$

όπου σ είναι η τυπική απόκλιση που ορίζουμε.

Ενδεικτικά, το Gaussian φίλτρο μας με μέγεθος K = 30 και σ = 3:



Εικόνα 1 - Gaussian Filter (K = 30, σ = 3)

Στη συνέχεια, προχωράμε σε κατασκευή του Gaussian Scale Space και τον υπολογισμό των DoGs. Ξεκινώντας από την αρχική εικόνα $I(x, y)$, φιλτράρουμε κάνοντας συνέλιξη της εικόνας με ένα Gaussian φίλτρο $G(x, y, \sigma)$ και προκύπτει το πρώτο space scale της εικόνας $\rightarrow L_1(x, y, \sigma)$, ξεκινώντας έτσι την υλοποίηση του αλγορίθμου στο πρώτο octave.

Στη συνέχεια χρησιμοποιούμε για εικόνα εισόδου την L_1 και συνεχίζουμε την ίδια διαδικασία με $\sigma' = k\sigma$ όπου $k = \sqrt{2}$ για αριθμό επαναλήψεων που ορίζεται από την μεταβλητή levels.

Σε κάθε επόμενο octave, χρησιμοποιούμε για εικόνα εισόδου το space scale με τη διπλάσια τυπική απόκλιση (σ) από αυτό του αρχικού του προηγούμενου octave.

Για την εύρεση των DoGs σε κάθε octave, απλά αφαιρούμε το επόμενο space scale από το προηγούμενό του [1]:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

Η υλοποίηση των παραπάνω γίνεται με τη συνάρτηση **myDoGs** η οποία χρησιμοποιεί cell arrays για να αποθηκεύσει και να επιστρέψει τα space scales και τα DoGs της εικόνας εισόδου. Η συνάρτηση myDoGs χρησιμοποιεί ως φίλτρο την έξοδο της συνάρτησης **my2DGaussianFilter.**

Στο σημείο αυτό είναι σημαντικό να σημειωθεί ότι εφόσον (όπως αναφέρθηκε) οι επιλογές μας για τις τιμές των παραμέτρων είναι πειραματικές, κατά την διεξαγωγή των πειραμάτων δοκιμάστηκαν διαφορετικές τιμές στις παραμέτρους, διαφορετικές τεχνικές προ-επεξεργασίας της εικόνας (όπως sharpening, αύξηση/adjust στο contrast, resize, ακόμα και HPF για edge-detecting), αλλά και διαφορετικές υλοποιήσεις των ζητούμενων συναρτήσεων. Συγκεκριμένα, για τις παραπάνω 2 συναρτήσεις, **myDoGs, my2DGaussianFilter** έγιναν 2 διαφορετικές υλοποιήσεις με ονόματα : **myDoGs, my2DGaussianFilter *και* myDoGs2, my2DGaussianFilter2.** Η αξιολόγηση των αποτελεσμάτων

έγινε καθαρά με βάση το οπτικό αποτέλεσμα που παράχθηκε, περιμένοντας ουσιαστικά από τον αλγόριθμο να βρει σημεία τα οποία με κάποιο τρόπο κάνουν το αντικείμενο να «ξεχωρίζει», δηλαδή το συγκεκριμένο pixel να «ξεχωρίζει» από τα γύρω του. Η χρήση επομένως έγινε με βάση την ποιότητα των αποτελεσμάτων, και επιλέξαμε ως βασικές τις myDoGs2, my2DGaussianFilter2.

Demo 1:

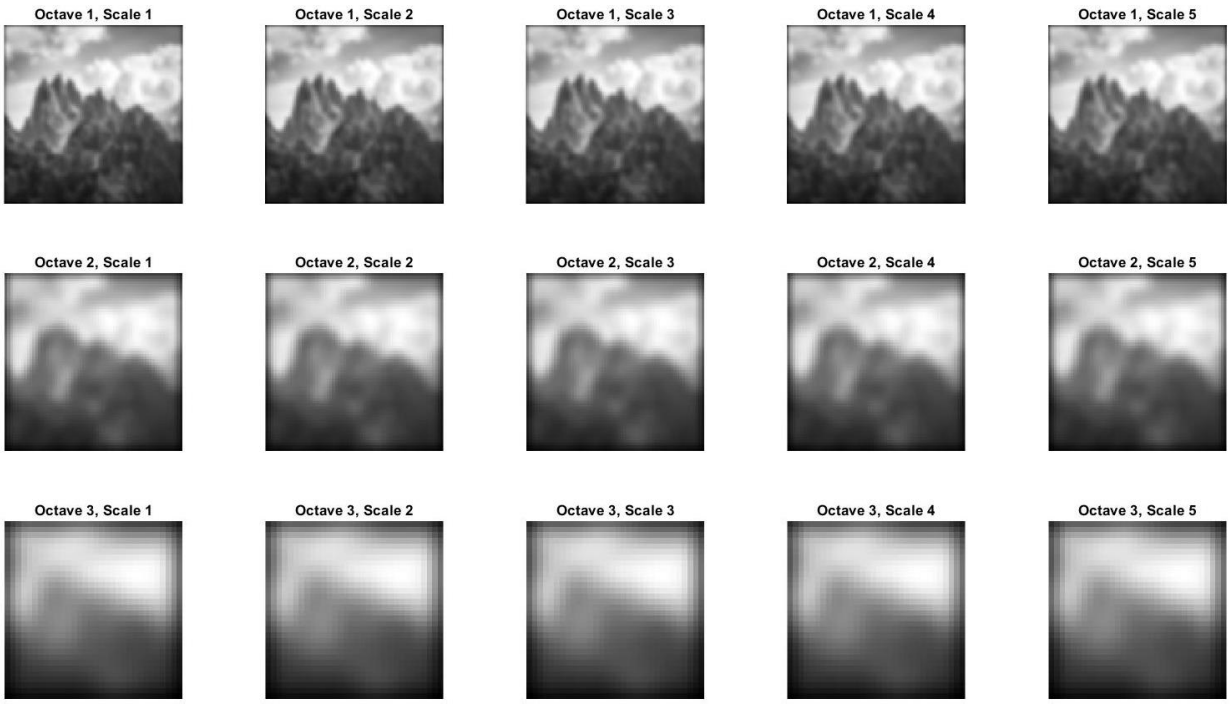
Στο συγκεκριμένο αρχείο (**demo1.m**) παρουσιάζονται τα αποτελέσματα της συνάρτησης myDoGs χρησιμοποιώντας τις εικόνες του δοσμένου αρχείου (dip_hw_3.mat). Παρουσιάζονται δηλαδή τα space scales αλλά και τα DoGs με σκοπό να έχουμε μία πιο ξεκάθαρη οπτική απέναντι στο αντικείμενο μελέτης μας αλλά και μία intuitive ματιά στο που περιμένουμε να βρει ο αλγόριθμος τα keypoints ενδιαφέροντος.

Τα πειράματα που έγιναν

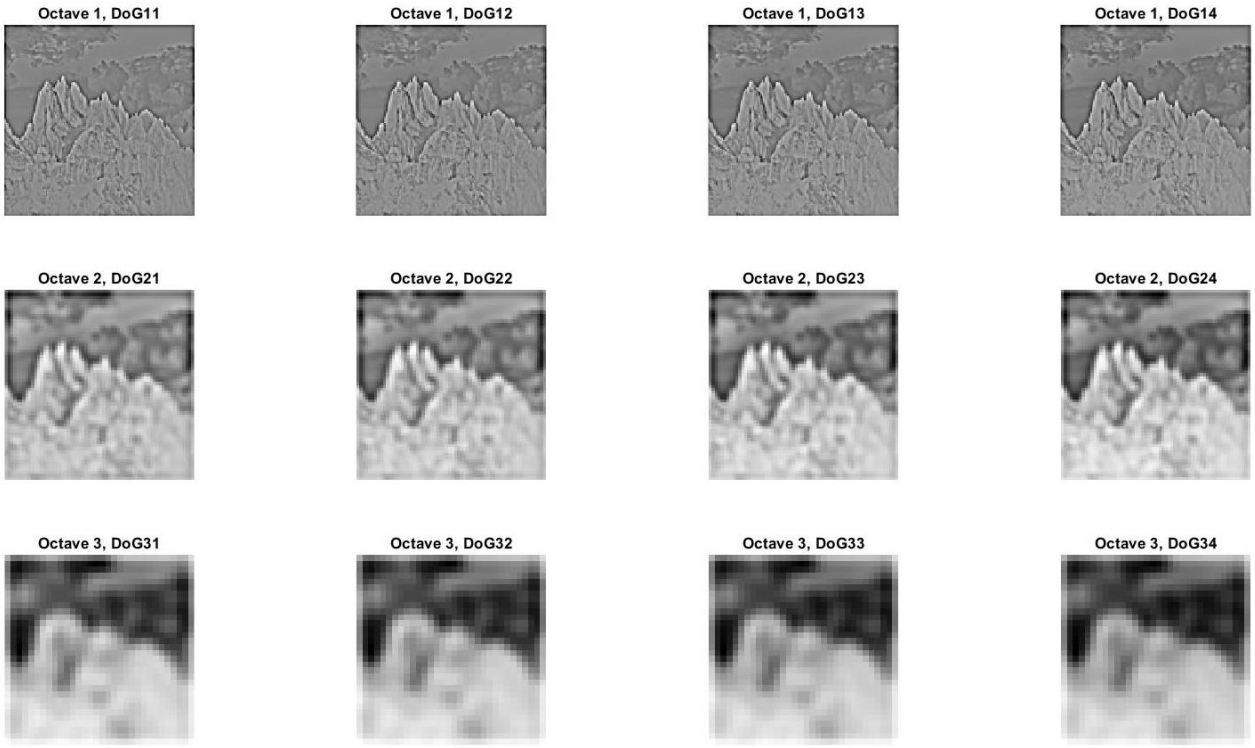
i) Πείραμα 1: $\sigma = \sqrt{2},\ K = 7,\ levels = 5,\ octaves = 3$ σε εικόνες 128x128
ii) Πείραμα 2: $\sigma = \sqrt{2},\ K = 7,\ levels = 3,\ octaves = 7$ σε εικόνες 128x128
iii) Πείραμα 3: $\sigma = \sqrt{2},\ K = 7,\ levels = 5,\ octaves = 3$ σε εικόνα 512x512

Αποτελέσματα:

i) Πείραμα 1:
Εικόνα : Βουνά – 128x128 pixels.

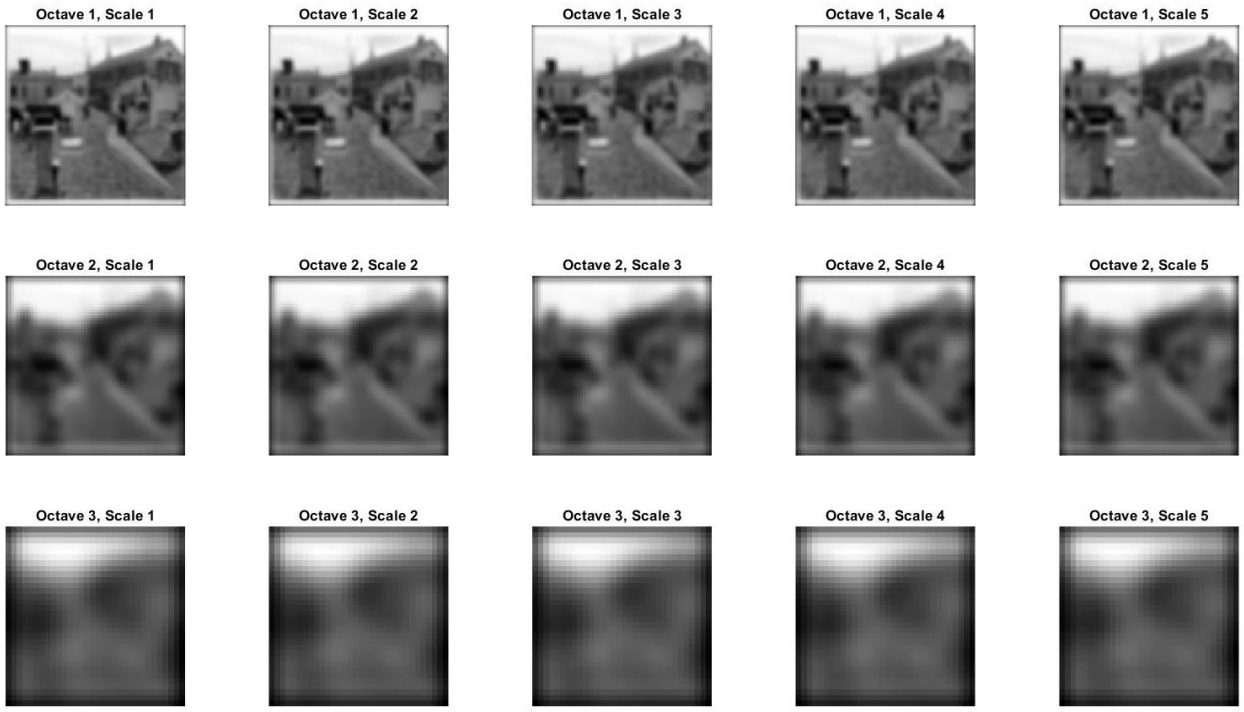


Εικόνα 2 - Space Scales / Βουνά / Πείραμα 1

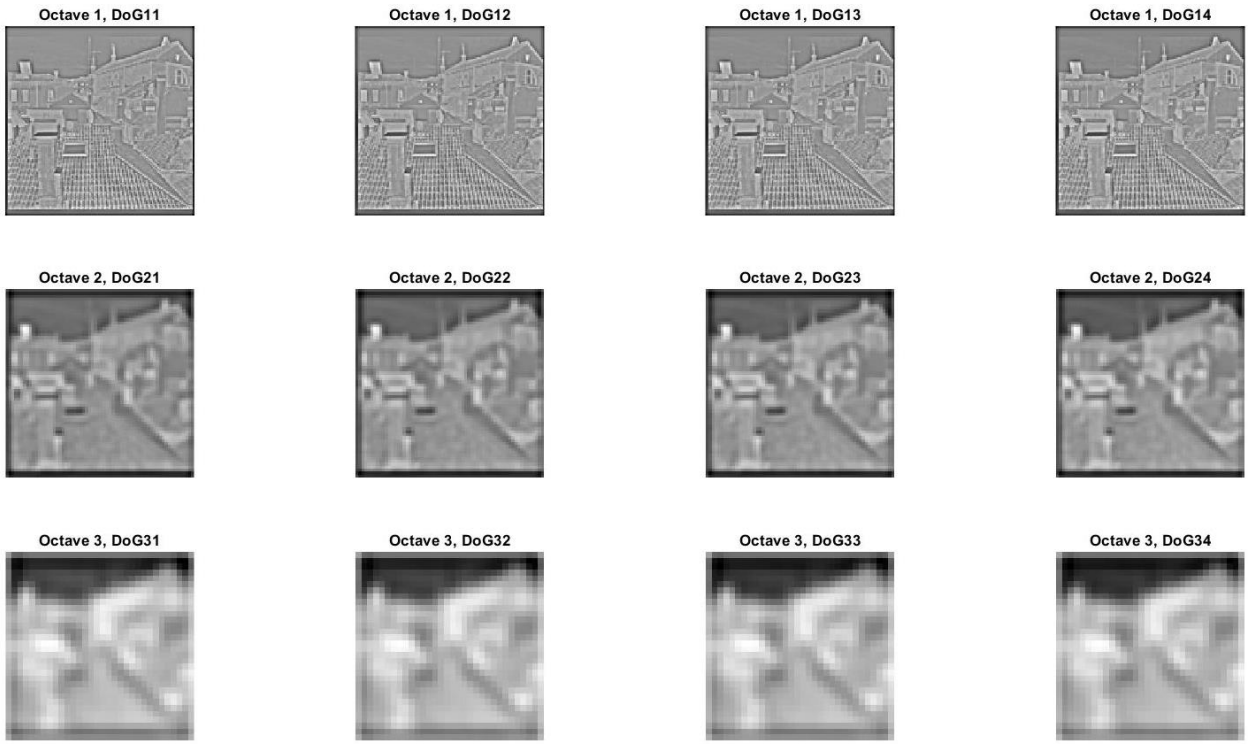


Εικόνα 3 - Difference-of-Gaussians (DoGs) / Βουνά / Πείραμα 1

Εικόνα : Στέγη – 128x128 pixels.



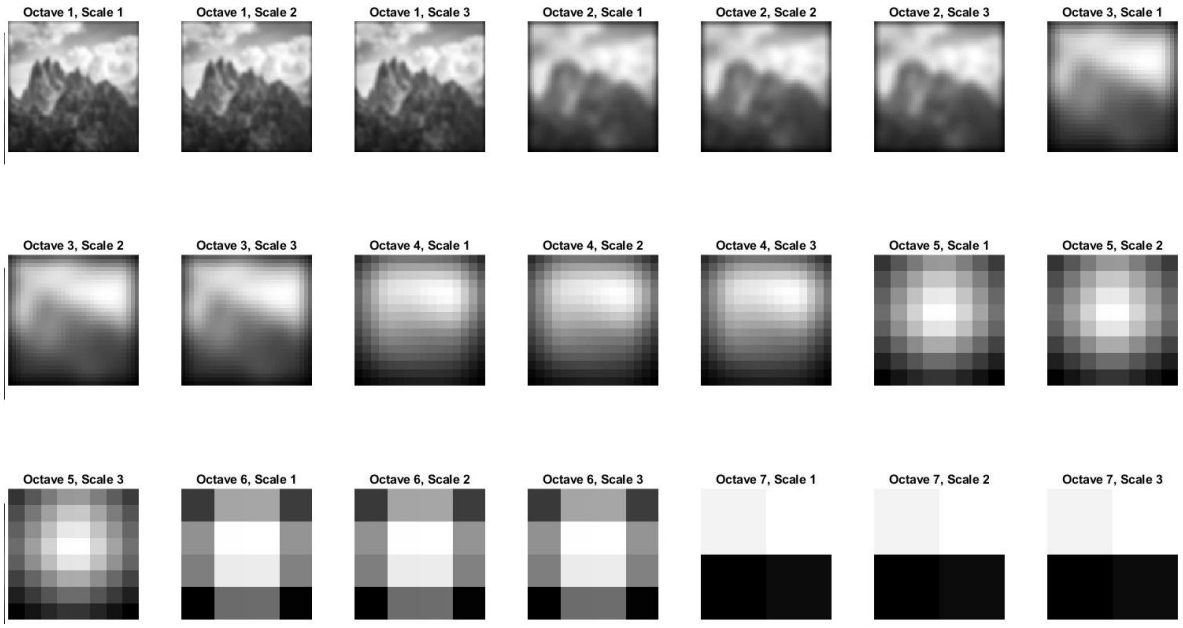
Εικόνα 4 - Space Scales / Στέγη / Πείραμα 1



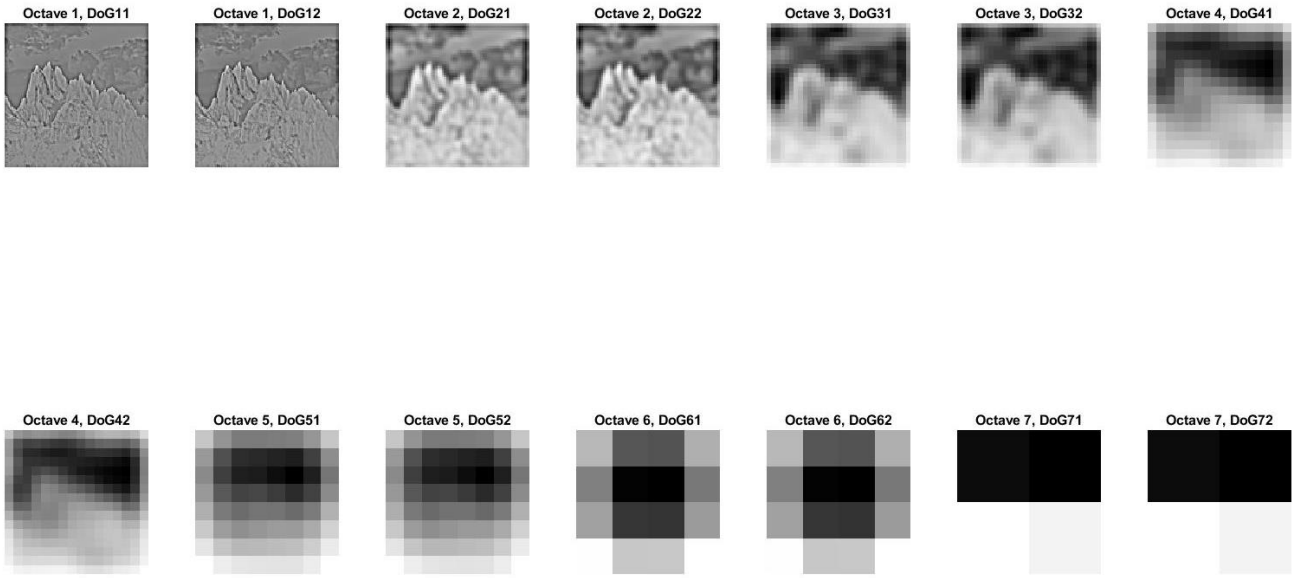
Εικόνα 5 - Difference-of-Gaussians (DoGs) / Στέγη / Πείραμα 1

Όπως βλέπουμε, λόγω του μικρού μεγέθους των εικόνων, η πληροφορία χάνεται πολύ γρήγορα από το downsampling και το blur των εικόνων. Αυτό φαίνεται ακόμα περισσότερο στη συνέχεια για 7 octaves, δηλαδή για 6 φορές downsampling, όπου η πληροφορία χάνεται εντελώς.

ii) <u>Πείραμα 2:</u>

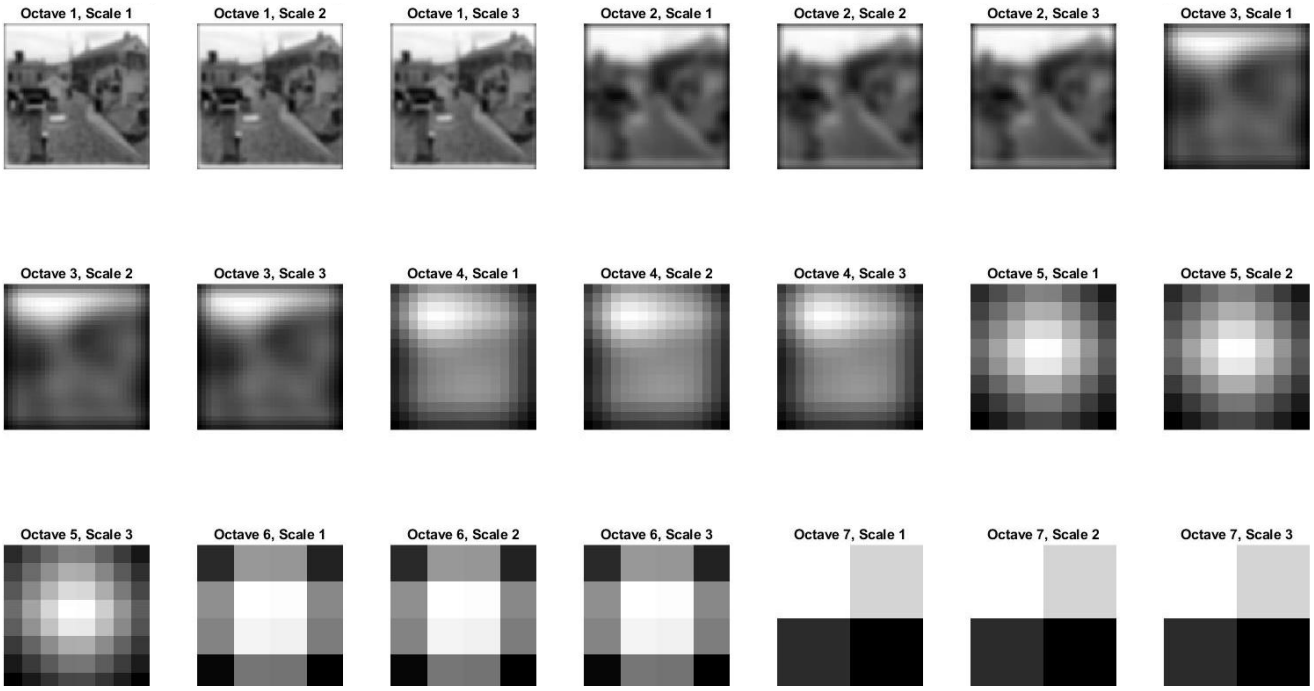


Εικόνα 6 - Space Scales / Βουνά / Πείραμα 2

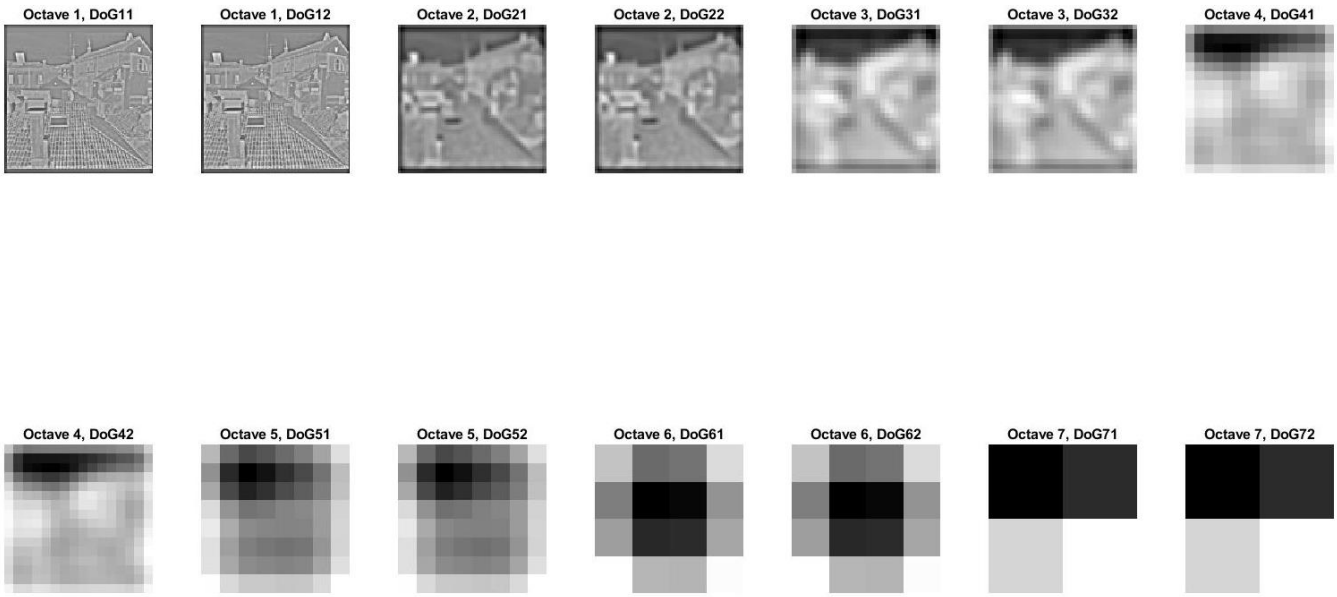


Εικόνα 7 - Difference-of-Gaussians (DoGs) / Βουνά / Πείραμα 2

και αντίστοιχα:



Εικόνα 8 - Space Scales / Στέγη / Πείραμα 2

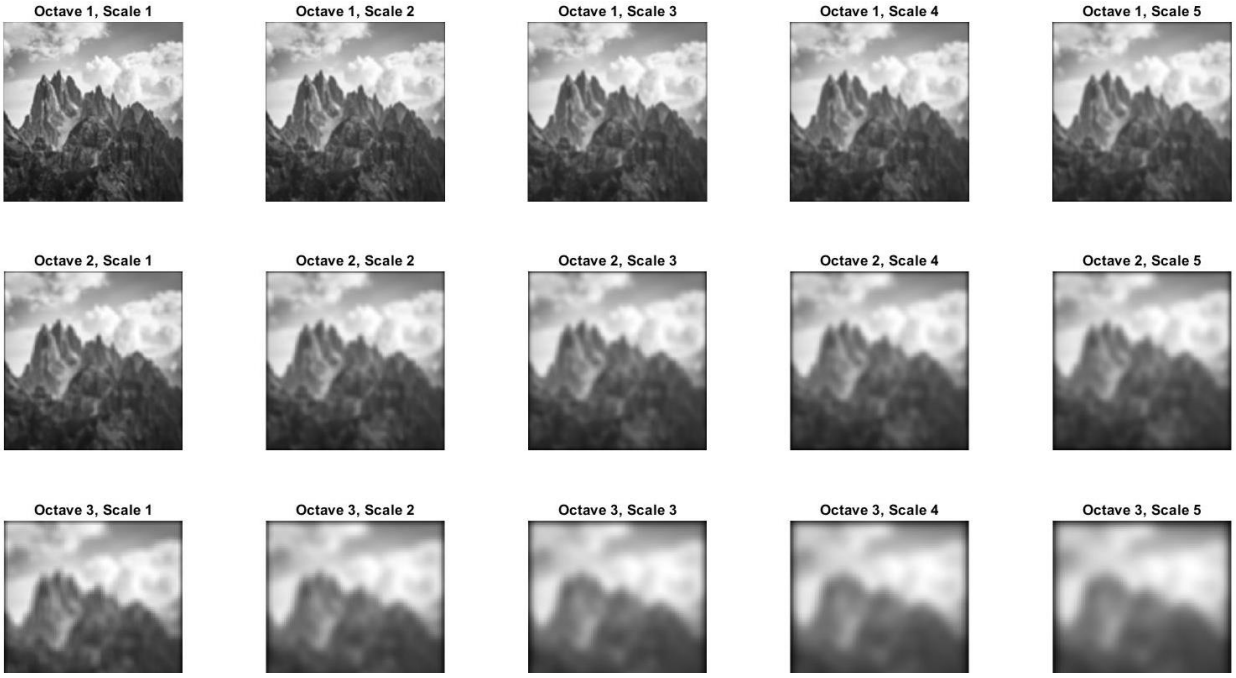


Εικόνα 9 - Difference of Gaussians (DoGs) / Στέγη / Πείραμα 2

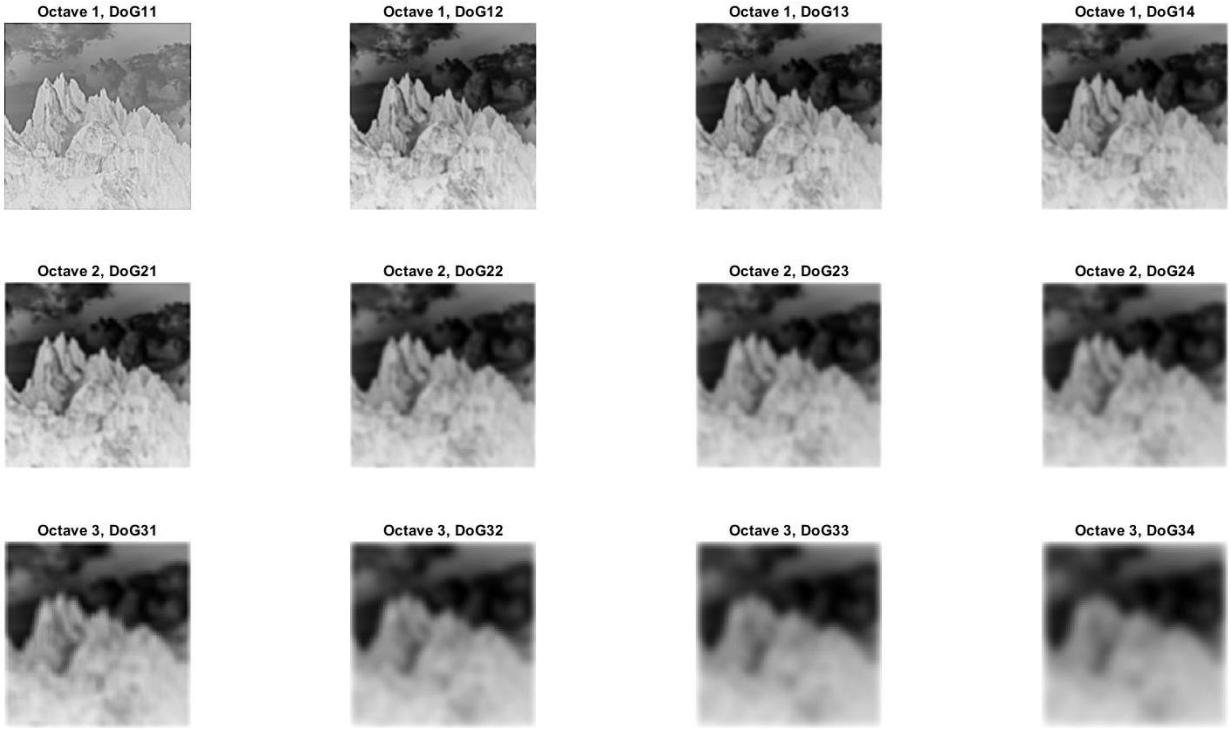
Είναι ξεκάθαρο ότι για αυτές τις τιμές στις παραμέτρους, το μέγεθος της εικόνας πρέπει να αλλάξει για να έχει νόημα ο αλγόριθμος.

Έτσι προχωράμε στο Πείραμα 3 όπου έχει γίνει resize (x4) της εικόνας με τα βουνά (τελικό μέγεθος 512x512) και στη συνέχεια η υλοποίηση του αλγορίθμου.

iii) <u>Πείραμα 3</u>:



Εικόνα 10 - Space Scales / Βουνά / Πείραμα 3 / Resized

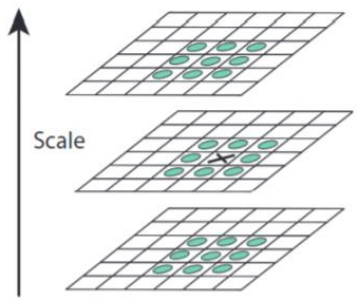


Εικόνα 11 - Difference-of-Gaussians (DoGs) / Βουνά / Πείραμα 3 / Resized

Όπως βλέπουμε, μεγαλώνοντας το μέγεθος της εικόνας αρχίζει να φαίνεται πως τα σημεία ενδιαφέροντος «αντιστέκονται» περισσότερο στο φιλτράρισμα που τους κάνουμε, εφόσον τώρα η εικόνα δεν αλλοιώνεται τόσο σημαντικά από το downsampling και η πληροφορία δεν έχει χαθεί. Σε κάθε περίπτωση, δεδομένου ότι συνήθως οι εικόνες είναι το πολύ μέχρι τα 1920 x 1200 pixel (φυσικά με εξαιρέσεις), ο αριθμός 7 για τις οκτάβες, δηλαδή 6 φορές downsampling θα κάνει ακόμα και μία εικόνα 2048x2048 να γίνει 32x32, και να χάσει όλη της την πληροφορία, επομένως δεν είναι χρήσιμο.

2. Εντοπισμός και φιλτράρισμα των salient keypoints

Για την εύρεση των keypoints, συγκρίνουμε κάθε pixel των DoGs (από κατάλληλο DoGs) με τα 26 γειτονικά του (γείτονες: 9 πάνω 9 κάτω και 8 στο ίδιο επίπεδο). Για να θεωρηθεί ένα DoGs κατάλληλο, πρέπει να βρίσκεται ανάμεσα σε 2 άλλα DoGs και το pixel το οποίο συγκρίνουμε να μη βρίσκεται στην περίμετρο.



Από την σύγκρισή του με τους γείτονές του, βλέπουμε αν αυτό το pixel είναι local maxima ή local minima, και αν είναι το θεωρούμε keypoint. Η συνάρτηση η οποία υλοποιεί τις παραπάνω συγκρίσεις και επιστρέφει τα salient keypoints και το id του (octave,scale) είναι η **myKeypoints**. Στην υλοποίηση της συγκεκριμένης συνάρτηση χρησιμοποιήθηκε επί πλέον από την εκφώνηση και ένα cell array για να είναι ευκολότερη εμφάνιση των keypoints στο εκάστοτε DoGs. Αξίζει να σημειωθεί παρόλα αυτά πως αυτό έκανε πιο αργό τον κώδικα.

Στη συνέχεια θέλουμε να φιλτράρουμε τα keypoints που βρήκαμε και να κρατήσουμε μόνο αυτά που θεωρούμε σημαντικά. Για να το πετύχουμε αυτό απλώς χρησιμοποιούμε ένα κατώφλι απόρριψης:

$$t = \frac{2^{1/n_{spo}} - 1}{2^{1/3} - 1} 0.015$$

Κάθε keypoint συγκρίνεται με το κατώφλι επί έναν πολλαπλασιαστικό παράγοντα p που δείχνει το πόσο αυστηροί είμαστε με τα εξαγόμενα keypoints, έτσι ώστε να κρατήσουμε μόνο αυτά που έχουν υψηλότερη αντίθεση και συνεπώς κατά πάσα πιθανότητα και μεγαλύτερου ενδιαφέροντος.

Για την απόρριψη των keypoints που περιγράφηκε παραπάνω χρησιμοποιείται η συνάρτηση **discardLowContrasted**.

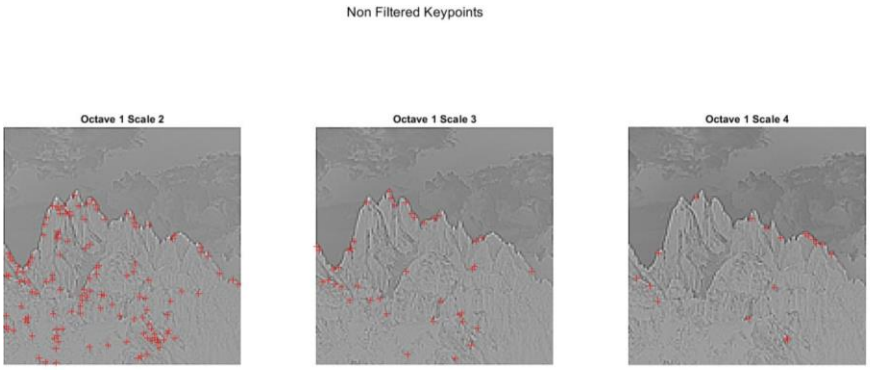
<u>Demo 2</u>:

Στο συγκεκριμένο αρχείο (**demo2.m**) παρουσιάζονται τα αποτελέσματα της συνάρτησης myKeypoints και discardLowContrasted χρησιμοποιώντας τις εικόνες του δοσμένου αρχείου (dip_hw_3.mat) αλλά και άλλες εικόνες. Παρουσιάζονται τα αποτελέσματα των DoGs πάνω στα οποία βρέθηκαν keypoints, καθώς και τα φιλτραρισμένα keypoints. Γίνανε τα εξής πειράματα:

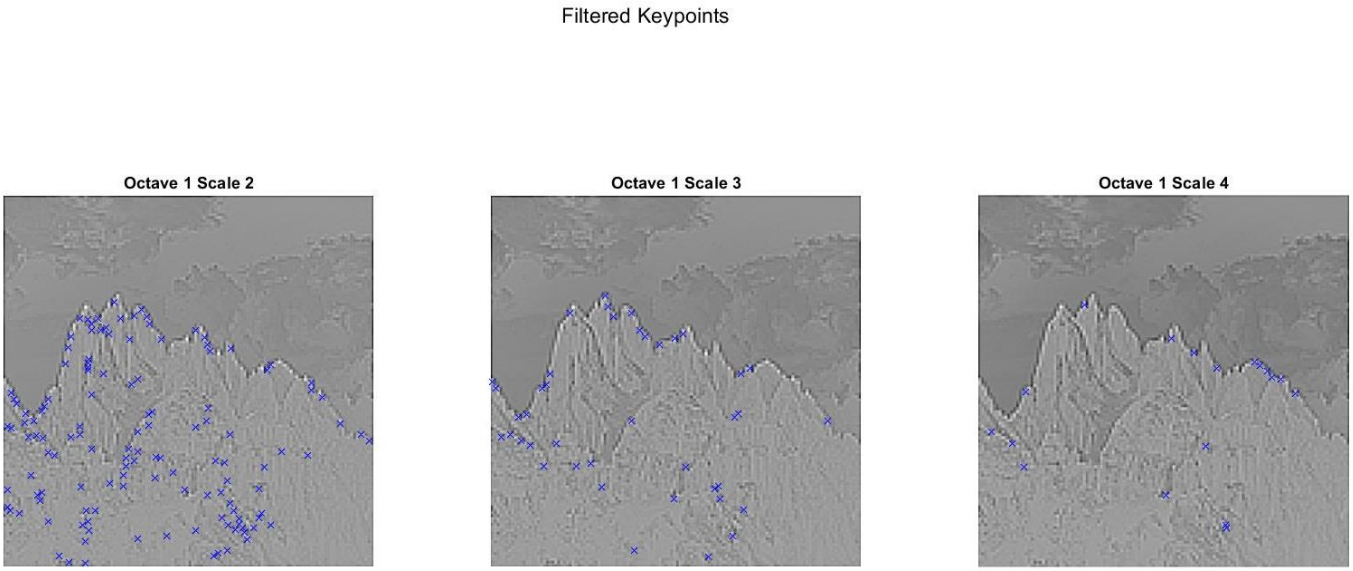
Στις παραμέτρους:

$$\sigma = \sqrt{2},\ K = 7,\ levels = 5,\ octaves = 3,\ p = 0.8 \qquad \text{(Lowe's Heuristics)}$$

1) Η εικόνα με τα βουνά (128x128) που δεν έβγαλε κανένα keypoint.
2) Η εικόνα με τα βουνά με resize (512x512) και sharpening ('Radius',2,'Amount',3):



Εικόνα 12 - Non-Filtered Keypoints / Βουνά / Lowe's Heuristics /Resize + Sharpening.

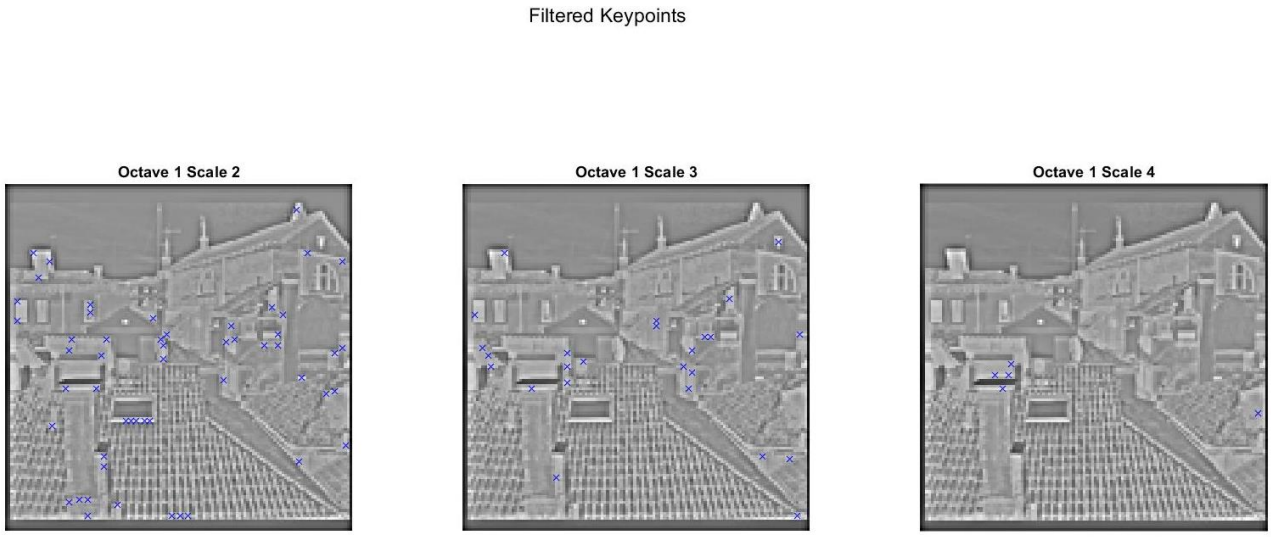


Εικόνα 13 - Filtered Keypoints / Βουνά / Lowe's Heuristics /Resize + Sharpening.

Το συγκεκριμένο αποτέλεσμα φαίνεται να είναι σχετικά ικανοποιητικό αφού βρίσκει τις κορυφές του βουνού καθώς και άλλες ακμές, παρόλα αυτά υπάρχουν και μερικά σημεία μικρότερου ενδιαφέροντος που έχουμε βρει.

3) Η εικόνα με τη στέγη (128x128) :

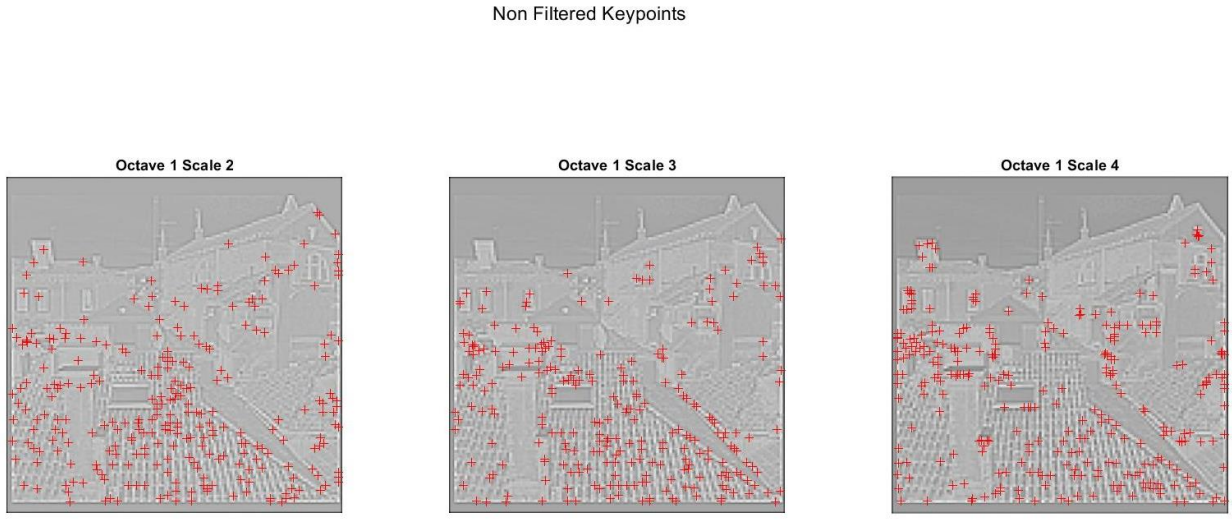
Εικόνα 14 - Non-Filtered Keypoints / Στεγη / Lowe's Heuristics



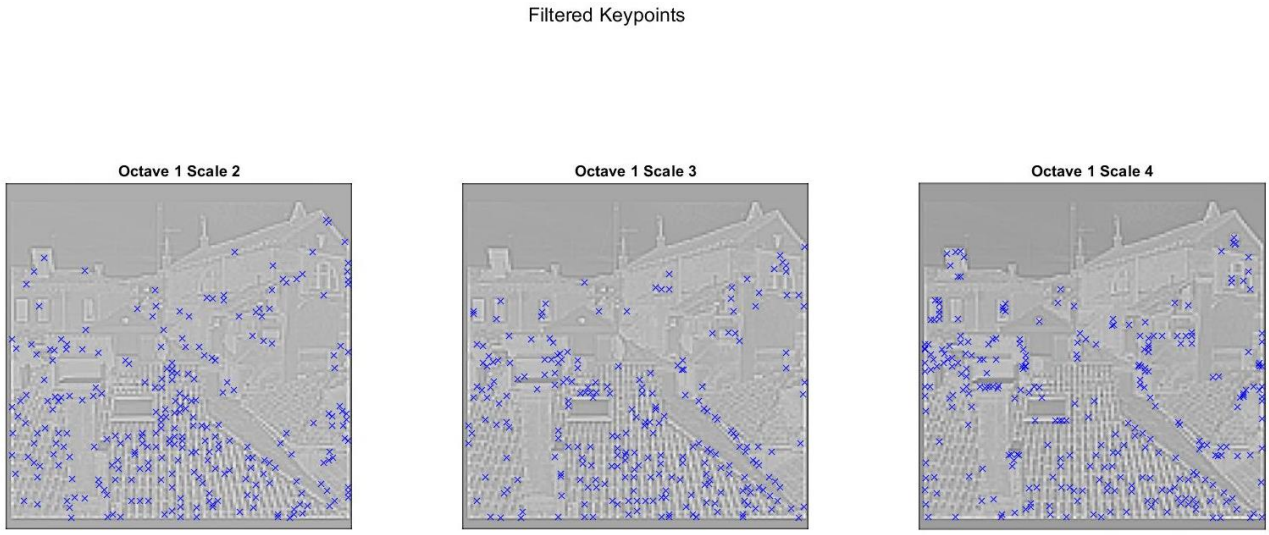
Εικόνα 15 - Filtered Keypoints / Βουνά / Lowe's Heuristics

Όπως βλέπουμε εδώ, λόγω του μικρού μεγέθους της εικόνας έχουμε βρει μόνο μερικά σημεία ενδιαφέροντος, ενώ έχουμε μηδαμινή απόρριψη. Παρόλα αυτά τα σημεία που βρήκαμε φαίνεται να είναι όντως σε ακμές.

4) Η εικόνα με τη στέγη με resize (512x512) και sharpening ('Radius',3,'Amount',1) και adjust:



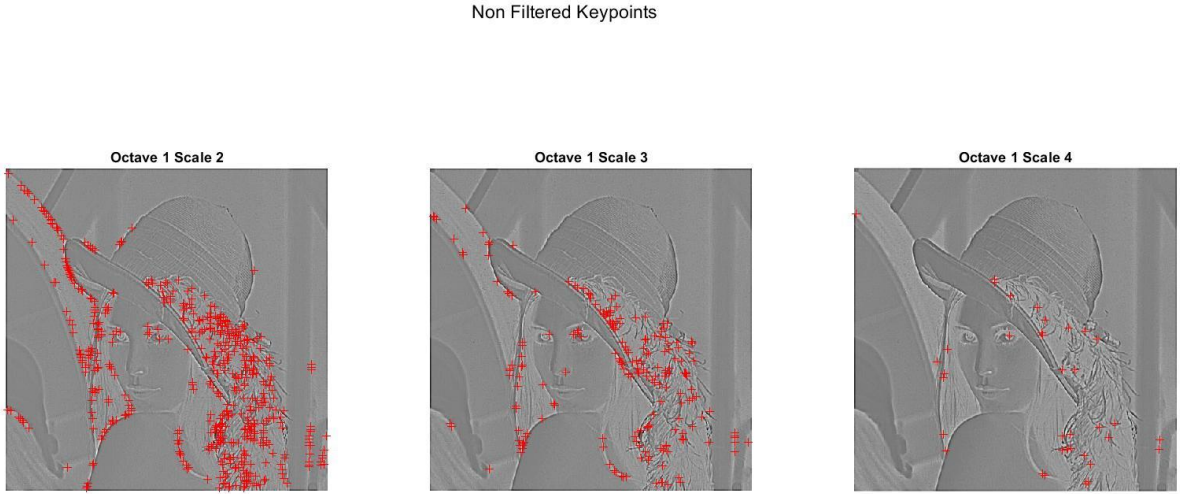
Εικόνα 16 - Non-Filtered Keypoints / Στεγη / Lowe's Heuristics / Resize-Sharpening-Adjust



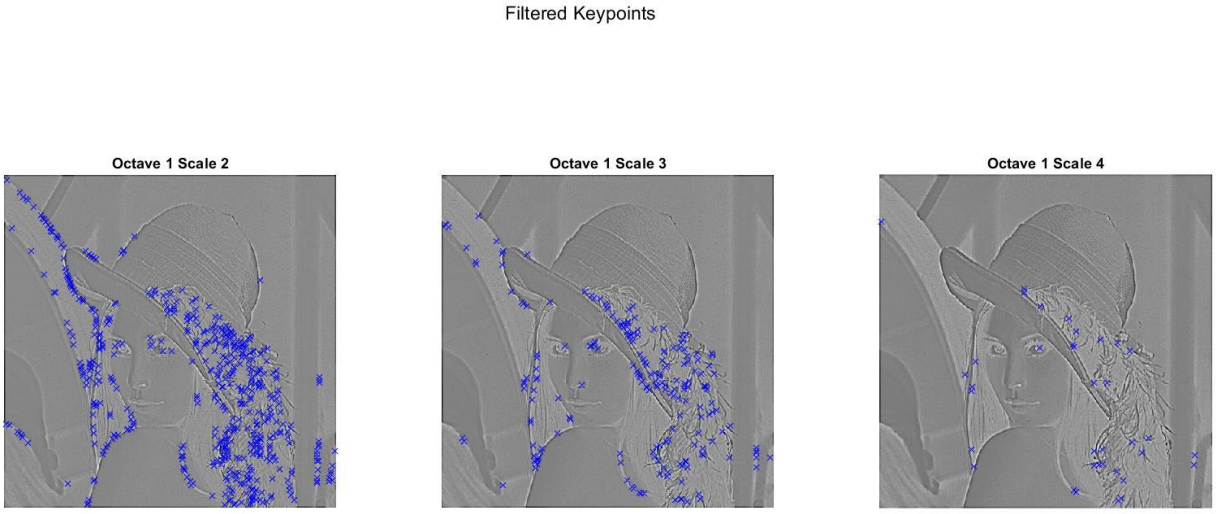
Εικόνα 17 - Filtered Keypoints / Στέγη / Lowe's Heuristics / Resize-Sharpening-Adjust

Είναι φανερό ότι εδώ έχει γίνει πολύ καλύτερα το detection. Τόσο η αύξηση του μεγέθους της εικόνας όσο και το να αυξήσουμε το contrast φαίνεται κάνει αυτά τα σημεία πιο εύκολα εντοπίσιμα. Για παράδειγμα έχουμε πιάσει τις 2 διαγώνιες ευθείες που βλέπουμε να ξεκινάνε σχεδόν από την κάτω δεξιά πλευρά της εικόνας και αποτελούνε ακμή της στέγης, καθώς και πολλά σημεία επάνω στις ακμές στα κεραμίδια.

5) Η κλασική (στην κοινότητα του DIP) εικόνα 'lenna.jpg' , 512x512:



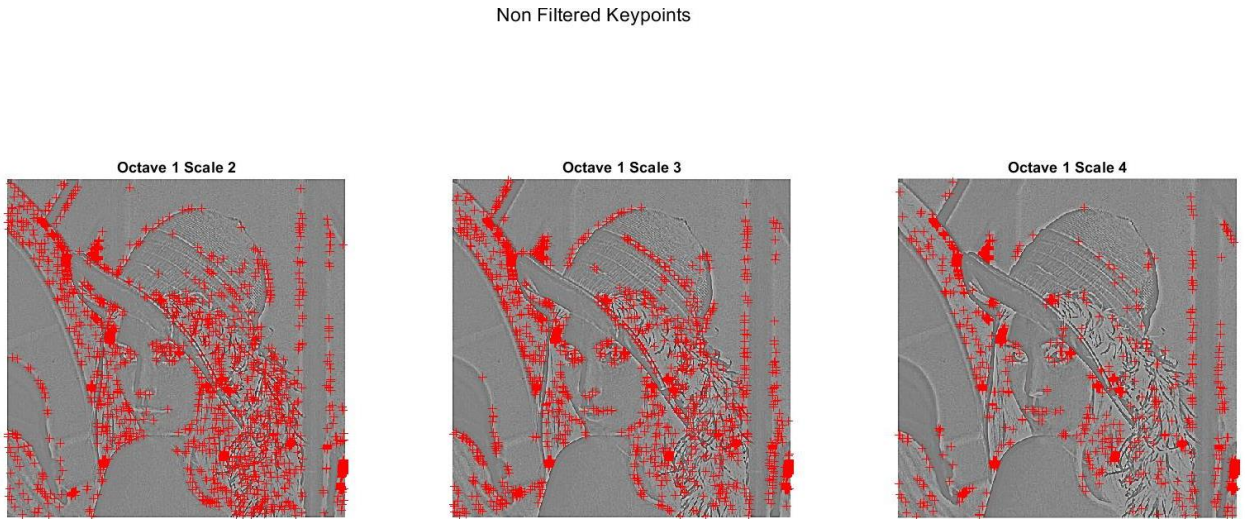
Εικόνα 18- Non-Filtered Keypoints / Lenna / Lowe's Heuristics



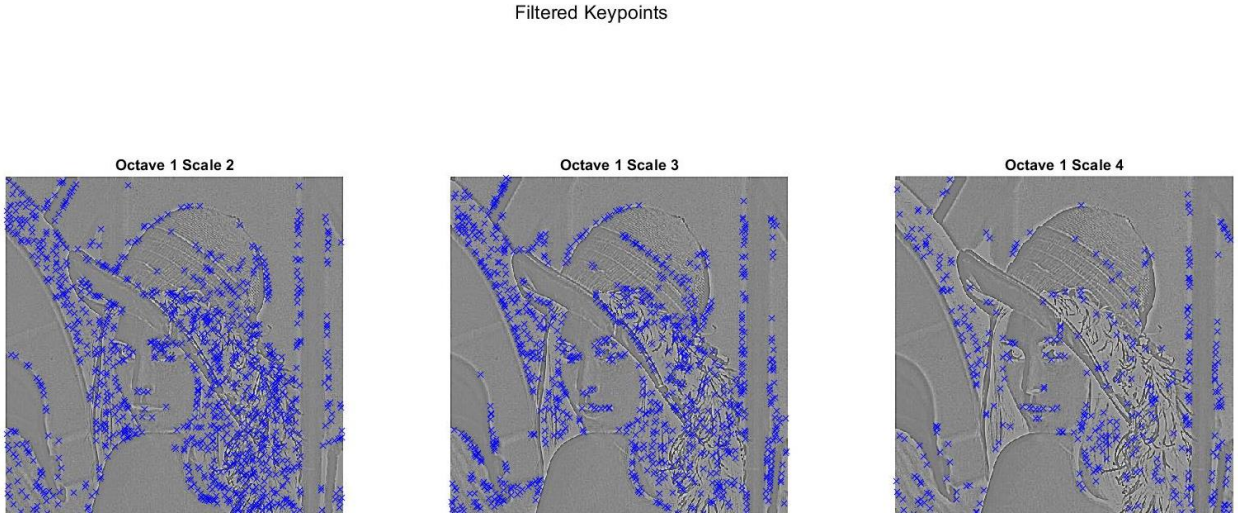
Εικόνα 19 - Filtered Keypoints / Lenna / Lowe's Heuristics

Βλέπουμε πάλι πως δεν έχουμε ιδιαίτερη απόρριψη. Έχουμε βρει ακμές στο background καθώς και ακμές στο πρόσωπο, στο καπέλο και στα μαλλιά της γυναίκας. Επίσης, λόγω των έντονων ακμών στα φτερά του καπέλου, βλέπουμε ότι κι εκεί υπάρχουν πολλά συσσωρευμένα σημεία ενδιαφέροντος.

6) Η εικόνα 'lenna.jpg' , 512x512, με sharpening ('Radius',6,'Amount',3) & Adjust Contrast:



Εικόνα 20 - Non-Filtered Keypoints / Lenna / Lowe's Heuristics / Sharpening & Adjust contrast

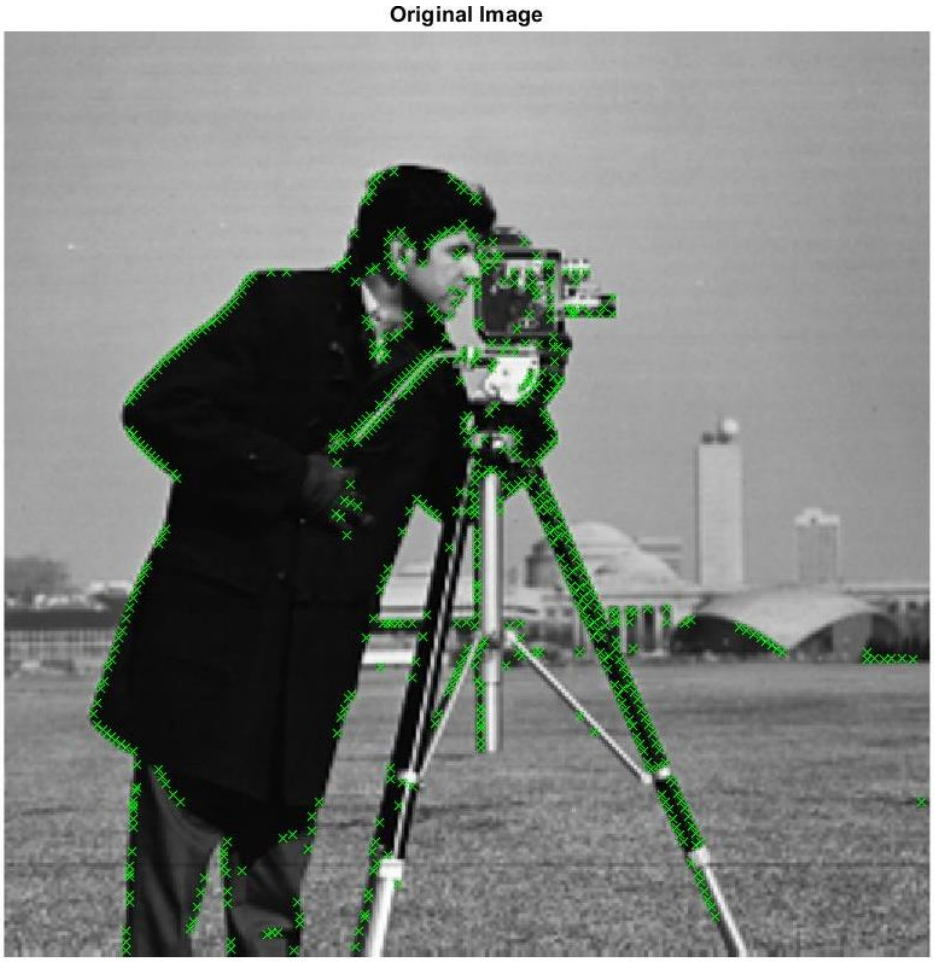


Εικόνα 21 - Filtered Keypoints / Lenna / Lowe's Heuristics / Sharpening & Adjust contrast

Είναι φανερό ότι εδώ έχουμε πετύχει πολύ καλύτερα το στόχο μας. Για να γίνει πιο ξεκάθαρο, θα εμφανίσουμε τα keypoints και πάνω στην αρχική εικόνα, αφού δεδομένου ότι είναι όλα στο ίδιο octave, δεν χρειάζεται να κάνουμε κάποιο rescale για την εμφάνιση:

Εικόνα 22 - Filtered Keypoints στην αρχική εικόνα / Lenna / Lowe's Heuristics / Sharpening & Adjust contrast

Αντίστοιχα, παίρνουμε για την εικόνα ‘cameraman.tif’, resized 512x512 (χρησιμοποιούμε πράσινο αυτή τη φορά για καλύτερο visualization)



Εικόνα 23 - Filtered Keypoints στην ‘cameraman.tif’ resized 512x512 / Lowe's Heuristics

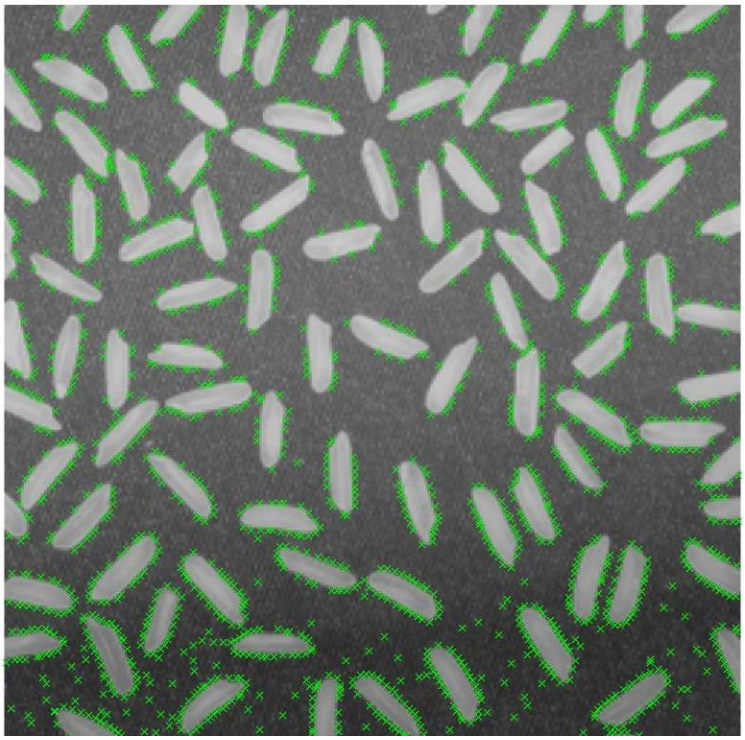
Επομένως, βλέπουμε ότι τα heuristics του Lowe δουλεύουν και στις δικές μας εικόνες σχετικά ικανοποιητικά όταν οι εικόνες έχουν ένα επαρκές μέγεθος, ενώ άλλες φορές ενδεχομένως να χρειάζονται και κάποια προ-επεξεργασία για να πάρουμε επιθυμητά αποτελέσματα.

Παρόλα αυτά, θα κάνουμε κάποια ακόμα πειράματα με διαφορετικές επιλογές από αυτές του Lowe, για να έχουμε μία πιο πλήρη τοποθέτηση. Επειδή οι παράμετροι είναι πάρα πολλές και η επιλογή τους γίνεται καθαρά πειραματικά με αξιολόγηση του αποτελέσματος, δεν θα σταθούμε πολύ στο να δείξουμε όλες τις δοκιμές. Αντ’ αυτού θα παρουσιαστούν κάποια αποτελέσματα που κρίνουμε ότι είναι καλά καθώς και οι εκάστοτε παράμετροι που χρησιμοποιήθηκαν.

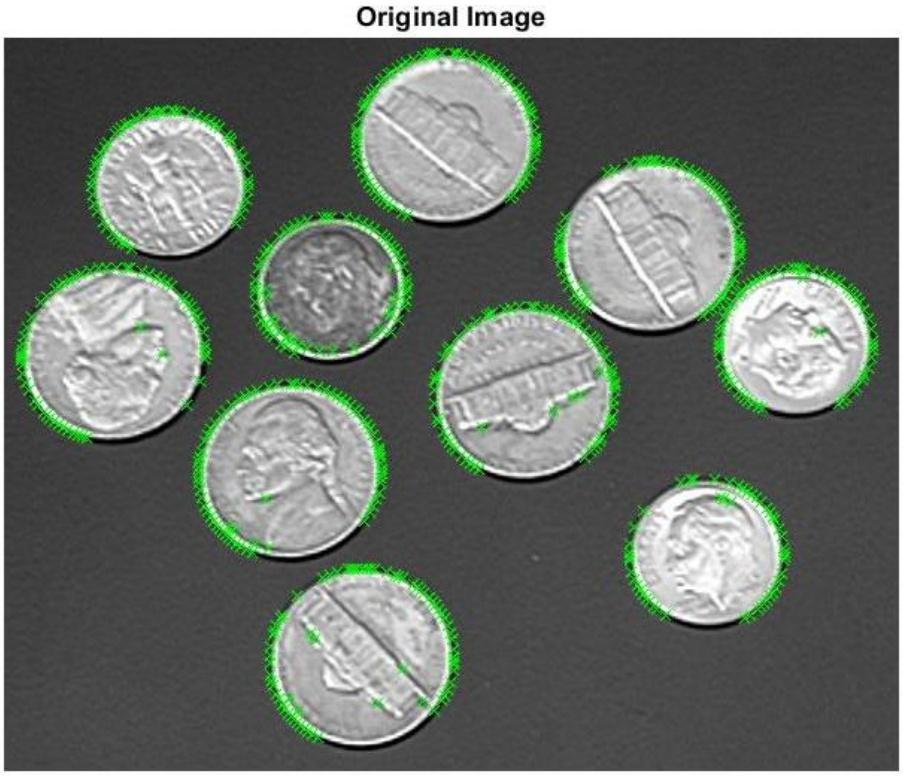
Εικόνα 24 - Filtered Keypoints στην αρχική εικόνα / Βουνά resized 512x512 / Lowe's Heuristics αλλά levels = 7 και p = 8 / Sharpening ('Radius',3,'Amount',3) & Adjust contrast

Εδώ παρόλο που δεν έχουν βρεθεί καλά όλες οι ακμές ανάμεσα στα βουνά, έχουν βρεθεί οι κορυφές, το οποίο σημαίνει ότι για παράδειγμα θα μπορούσαμε να κάνουμε image stitching διαφορετικών εικόνων που έχουν αλληλοεπικάλυψη στα εικονιζόμενα βουνά / κορυφές, και έτσι να φτιάξουμε μία πανοραμική φωτογραφία. Αν θέλαμε μόνο τις κορυφές, θα επιλέγαμε ακόμα μεγαλύτερο p πχ p = 15:

Εικόνα 25 - Filtered Keypoints στην αρχική εικόνα / Βουνά resized 512x512 / Lowe's Heuristics αλλά levels = 7 και p = 15 / Sharpening ('Radius',3,'Amount',3) & Adjust contrast



Εικόνα 26 - Filtered Keypoints στην αρχική εικόνα / Ρύζι ('rice.png') 512x512/ Lowe's Heuristics αλλά levels = 7 και p = 8 / Sharpening ('Radius',2,'Amount',2) & Adjust contrast



Εικόνα 27 - Filtered Keypoints στην αρχική εικόνα / Νομίσματα ('coins.png') 512x512 / Lowe's Heuristics / Resized & Sharpening ('Radius',2,'Amount',2)

Κάτι που παρατηρήθηκε έντονα λοιπόν είναι πως το sharpening της φωτογραφίας πριν την εφαρμογή του αλγόριθμου σε ορισμένες φωτογραφίες ήταν η ουσιώδης αλλαγή που οδήγησε σε εύρεση σημαντικών keypoints αλλά και μεγάλου πλήθους αυτών.

<u>Σύνοψη – Συμπεράσματα – Future work</u>:

Στην εργασία αυτή έγιναν διάφοροι πειραματισμοί πάνω στον αλγόριθμο Scale Invarient Feature Transform – SIFT Detector. Ο αλγόριθμος φάνηκε να δουλεύει σχετικά καλά με τις τιμές που πρότεινε ο Lowe [1],[2] , παρόλα αυτά είναι σημαντικό οι φωτογραφίες να είναι ήδη σε ένα μέγεθος που να μπορεί να υποστηρίξει τον αλγόριθμο έστω και για μερικά βήματα, προτού χαθεί η πληροφορία από το φιλτράρισμα. Μία πρακτική που δοκιμάστηκε στην συγκεκριμένη εργασία είναι η προ-επεξεργασία της εικόνας, ιδιαίτερα κάνοντας Sharpening & Adjust Contrast, ώστε να κάνουμε μία εικόνα με μεγαλύτερες αντιθέσεις και συνεπώς να «βοηθήσουμε» τον αλγόριθμο να βγάλει σωστά αποτελέσματα. Τέλος, μία παράμετρος που φαίνεται να δημιουργεί πάρα πολλά keypoints, πολλές φορές και όχι χρήσιμα, είναι η αύξηση στο μέγεθος του φίλτρου (K). Η επιλογή των παραμέτρων βέβαια, μπορεί να αλλάξει ανάλογα με το πόσα ξέρουμε για το πρόβλημα που έχουμε να αντιμετωπίσουμε, δηλαδή την φύση της εικόνας, το πλήθος των ακμών που εμφανίζονται καθώς και ο τελικός μας στόχος, δηλαδή τόσο το τι θέλουμε να κάνουμε με τα χαρακτηριστικά που εξάγουμε, όσο και το ποια στοιχεία της εικόνας θέλουμε να περιγράψουμε. Έτσι, δεδομένου ότι αυτή η εργασία αφορούσε την υλοποίηση του SIFT Detector, το επόμενο βήμα στο συγκεκριμένο ερευνητικό πεδίο είναι η συμπλήρωση του αλγόριθμου με τον SIFT Descriptor, καθώς και οι εφαρμογές αυτών, για παράδειγμα στο Computer Vision, στη ρομποτική (visual odometry).

Αναφορές:

[1] **D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999, pp. 1150-1157 vol.2, doi: 10.1109/ICCV.1999.790410.**

[2] **Lowe, David G.. "Distinctive Image Features from Scale-Invariant Keypoints." *Int. J. Comput. Vision* 60 , no. 2 (2004): 91--110.**

[3] ***Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., USA.***