# Ψηφιακή Επεξεργασία Εικόνας

# Εργασία 3

Όνομα: Πορλού Χάιδω

ΑΕΜ: 9372

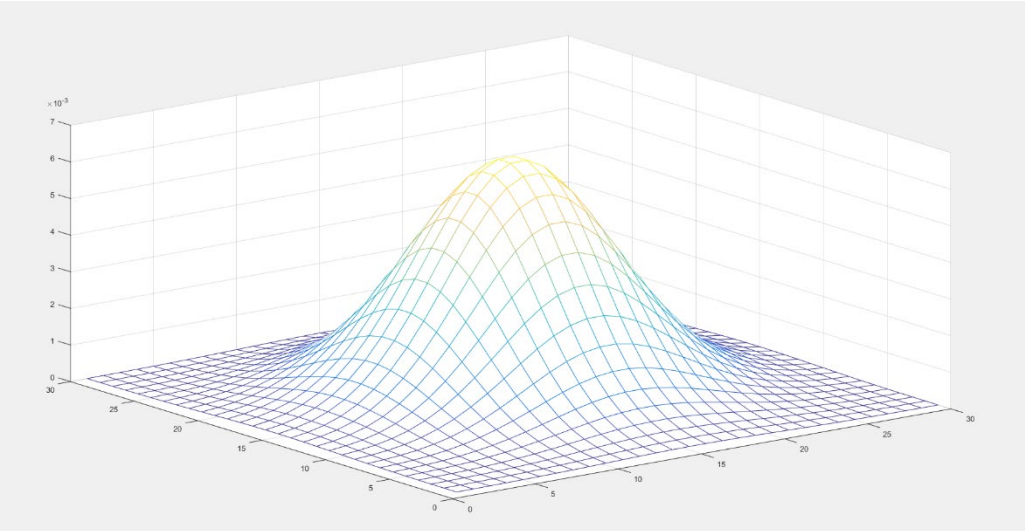
---

Στην τρίτη και τελευταία εργασία του μαθήματος «Ψηφιακή Επεξεργασία Εικόνας» καλούμαστε να υλοποιήσουμε έναν SIFT detector από την αρχή. Πιο συγκεκριμένα, μετά τη δημιουργία ενός gaussian φίλτρου, προχωράμε στον υπολογισμό των spacescales και των Difference-of-Gaussians. Στη συνέχεια, χρησιμοποιούμε τα DoGs για τον εντοπισμό των salient keypoints, και, τέλος, μέσω κατάλληλης διαδικασίας, καταλήγουμε στο φιλτράρισμα τους, ώστε να απομείνουν τα «σωστότερα» keypoints που μας βοηθούν στον εντοπισμό ακμών και περιγραμμάτων.

## 1. Gaussian Space Scale

Σαν πρώτο βήμα της εργασίας ορίζεται η υλοποίηση ενός 2D gaussian filter μέσω της συνάρτησης **my2DGaussianFilter()** μεγέθους K x K (το οποίο ορίζουμε εμείς), και ενός συγκεκριμένου sigma.

Το φίλτρο είναι κεντραρισμένο γύρω από το (0,0), αλλά και κανονικοποιημένο (παίρνει τιμές ανάμεσα στο 0 και το 1). Παρακάτω παρατίθεται ένα gaussian φίλτρο με K = 30 και σ = 5.



Επόμενο βήμα αποτελεί ο υπολογισμός των gaussian space scales. Στο σημείο αυτό σημειώνω πως παρέκκλινα λίγο από την εκφώνηση της εργασίας (μετά από συζήτηση με συμφοιτητές αλλά και προτάσεις του διδάσκοντος). Η διαδικασία λοιπόν έχει ως εξής:

Έστω I(x,y) μία κανονικοποιημένη (τιμές στο [0,1]), grayscale εικόνα εισόδου. Εφαρμόζοντας, μέσω 2D convolution, το Gaussian φίλτρο G(x,y) παίρνουμε το scale space της εικόνας L1(x,y):

$L1(x,y) = I(x,y) * G(x,y)$

Στην πράξη, αυτό που κάνει η εξίσωση αυτή είναι ενα low-pass-filtering ή αλλιώς blurring στην εικόνα I(x,y).

Η διαδικασία συνεχίζει χρησιμοποιώντας την **αρχική** εικόνα I(x,y) σαν εικόνα εισόδου στην παραπάνω εξίσωση με σ’ = kσ) όπου και παράγεται η εικόνα L1(x,y,σ’). Για τις ανάγκες της εργασίας θα χρησιμοποιήσουμε $k = \sqrt{2}$.

Η διαδικασία τερματίζει μετά από έναν αριθμό επαναλήψεων που ορίζεται από την μεταβλητή levels. Για παράδειγμα αν levels = 4 τότε έχουν παραχθεί τα space scales:

- $L1(x, y, 2\sqrt{2}\sigma)$
- $L1(x, y, 2\sigma)$
- $L1(x, y, \sqrt{2}\sigma)$
- $L1(x, y, \sigma)$

με σειρά δημιουργίας από κάτω προς το πάνω. Τα levels αυτά ανήκουν στο πρώτο **octave**.

Συνεχίζοντας, χρησιμοποιούμε το σ του $3^{ου}$ κατά σειρά δημιουργίας spacescale του προηγούμενου octave (εδώ σ’ = 2σ) για να ξεκινήσουμε τη δημιουργία του επόμενου octave. Σαν αρχική εικόνα χρησιμοποιούμε την υποδειγματοληπτούμενη (με scale factor 2) αρχική εικόνα, και επαναλαμβάνουμε την διαδικασία του gaussian blurring όπως παραπάνω, όσες φορές υποδεικνύει η μεταβλητή levels.

Η μεταβλητή octaves που δίνουμε σαν όρισμα στην συνάρτηση που θα δημιουργηθεί ορίζει φυσικά τον αριθμό των octaves που τελικά θα δημιουργηθούν.

Ώστε να προχωρήσουμε στη διαδικασία εντοπισμού των salient keypoints, πρέπει πρώτα να ορισθούν τα Difference-of-Gaussians (DoGs), τα οποία είναι η διαφορά 2 γειτονικών space scales σε ένα octave, δηλαδή:

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

Σημειώνεται πως, φυσικά, ο αριθμός των DoGs για κάθε octave θα είναι levels – 1.

Η συνάρτηση που καλούμαστε να υλοποιήσουμε είναι η εξής:

**function [spacescales , DoGs] = myDoGs(img, K, sigma, levels, octaves)**

όπου οι μεταβλητές επιστροφής spacescales και DoGs αποτελούν cell arrays που χρησιμοποιούνται σαν 4D πίνακες, img η αρχική grayscale εικόνα, K το μέγεθος του gaussian φίλτρου, sigma η τυπική απόκλιση της gaussian κατανομής σ, levels και octaves ο αριθμός των levels κάθε octave και των octaves αντίστοιχα.

---

**Demo 1**

Στο πρώτο demo της εργασίας, καλούμαστε να παρουσιάσουμε τη λειτουργία της συνάρτησης που υλοποιήσαμε, myDoGs(). Μας δίνονται ήδη 2 grayscale εικόνες τις οποίες θα χρησιμοποιήσουμε για την παρουσίαση των αποτελεσμάτων μας.

Οι παράμετροι που θα χρησιμοποιηθούν είναι:

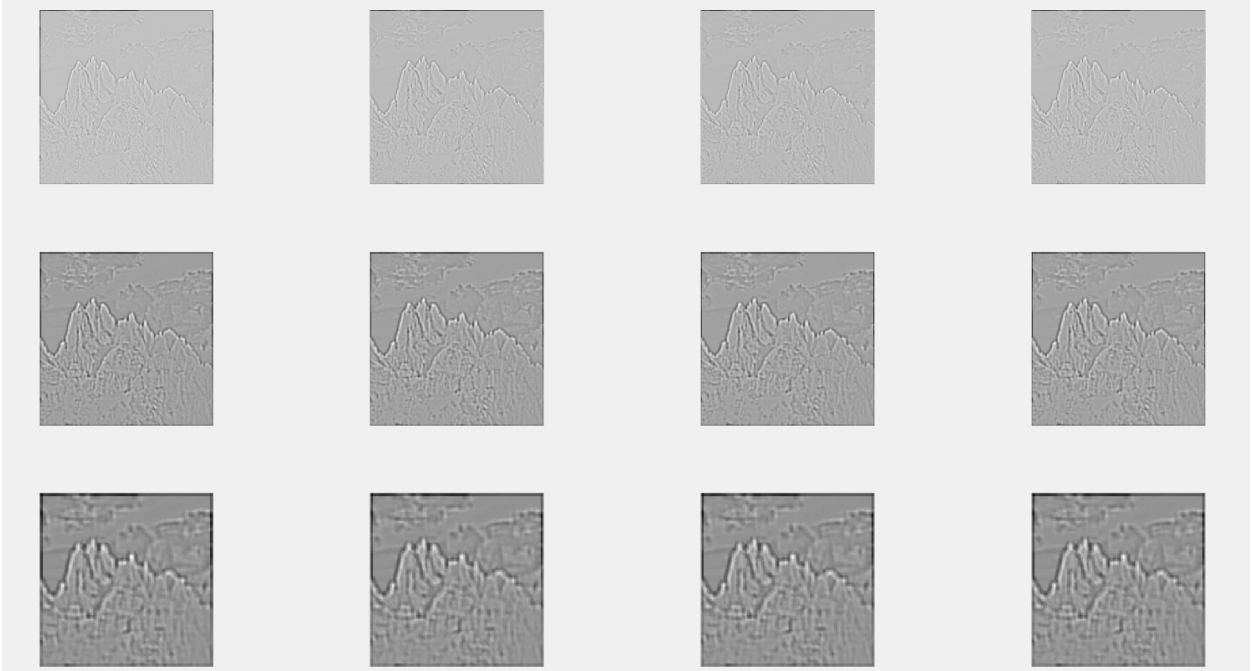
1. $\sigma = \sqrt{2}$, K = 7, levels = 5 και octaves = 3
2. $\sigma = \sqrt{2}$, K = 7, levels = 3 και octaves = 7

Παρακάτω παρουσιάζονται τα αποτελέσματα για τις 2 παραπάνω περιπτώσεις (spacescales & DoGs), για τις εικόνες mountains και roofs αντίστοιχα.

1. mountains (σ = √2, K = 7, levels = 5 και octaves = 3)

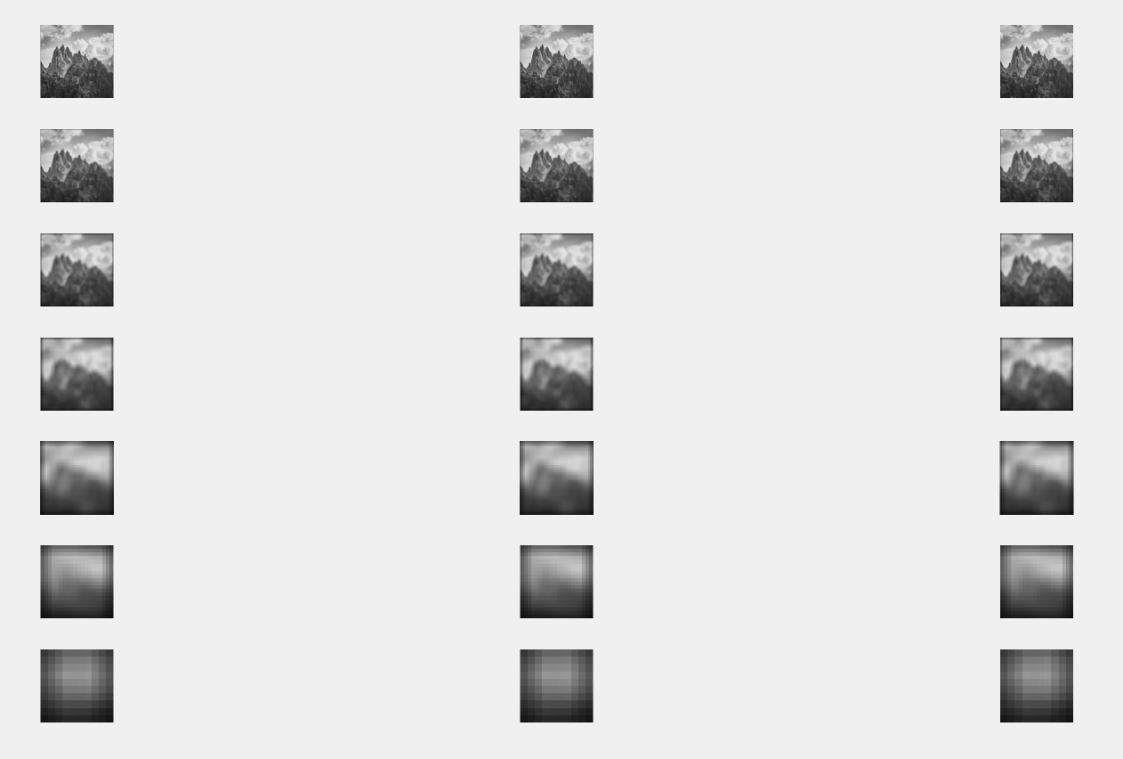
**SPACESCALES**

**DOGS**



2. mountains (σ = √2, K = 7, levels = 3 και octaves = 7)

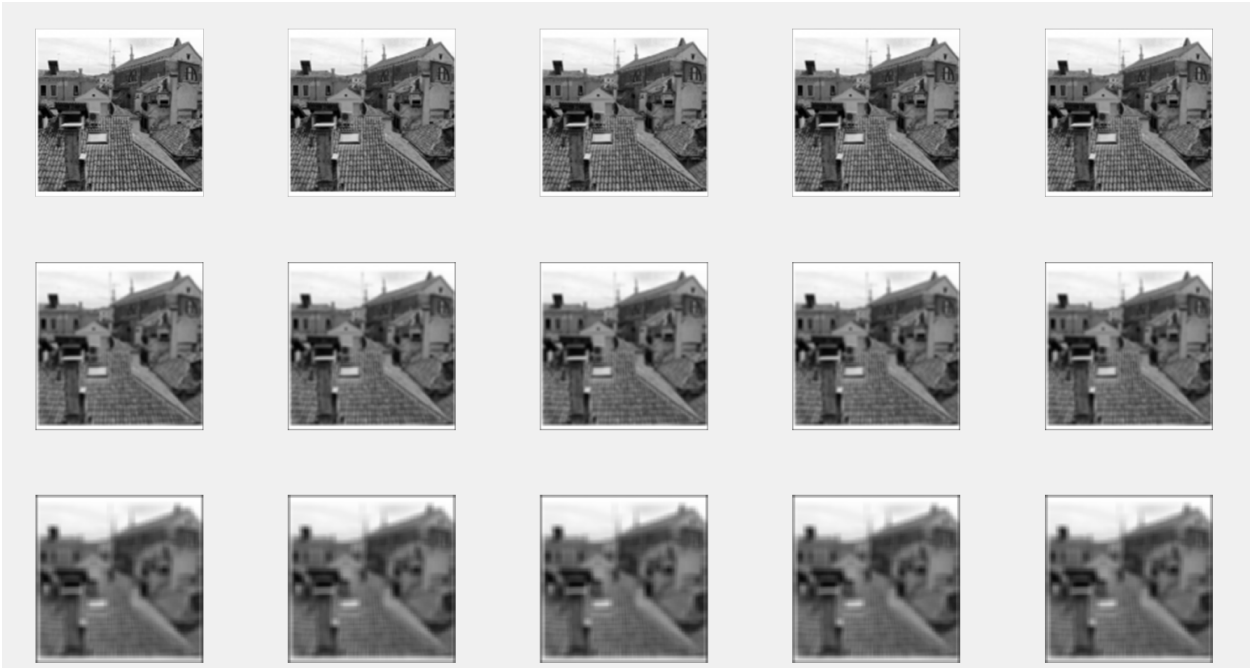
**SPACESCALES**



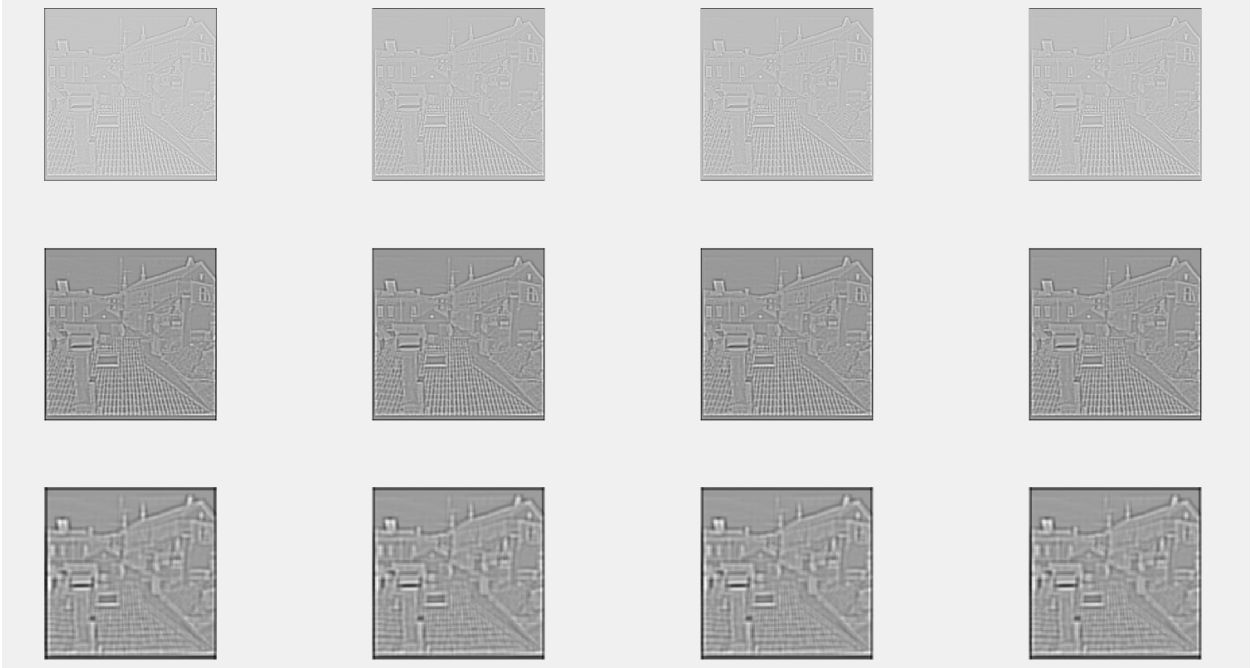
**DOGS**

3. roofs (σ = √2, K = 7, levels = 5 και octaves = 3)

**SPACESCALES**



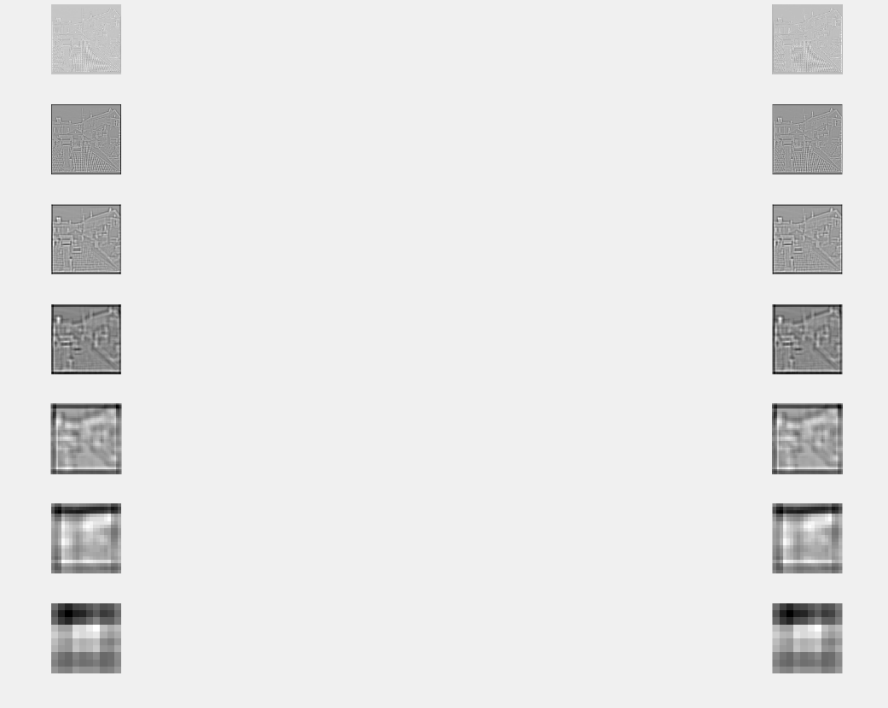
**DOGS**



4. roofs (σ = √2, K = 7, levels = 3 και octaves = 7)

**SPACESCALES**

**DOGS**



Αξίζει να γίνουν κάποια σχόλια για τις παραπάνω διαδικασίες. Αρχικά, παρατηρείται πως η διαφορά στο blurring ανάμεσα σε spacescales και DoGs στο ίδιο octave είναι σχετικά μικρή και δεν μπορεί να παρατηρηθεί εύκολα με το μάτι. Η διαφορά ανάμεσα στα octaves από την άλλη είναι πιο έντονη (όπως και είναι επιθυμητό και αναμενόμενο). Η διαφορά αυτή δεν οφείλεται μόνο στην αλλαγή του σ αλλά φυσικά και στο downsampling που γίνεται στο αντίστοιχο βήμα.

Επιπροσθέτως, το σημαντικότερο ίσως σχόλιο για τα παραπάνω είναι πως καλύτερα αποτελέσματα (σε αυτό αλλά και στο επόμενο βήμα) φαίνεται να έχει η επιλογή πολλών levels σε κάθε octave και λίγα octave, σε αντίθεση με την επιλογή πολλών octaves που το καθένα αποτελείται από λίγα levels. Το παραπάνω έχει τη βάση του στην προσπάθεια ανίχνευσης μικρών διαφορών ανάμεσα στα level του ίδιου octave όσο η εικόνα βρίσκεται σε ικανοποιητικό μέγεθος, ώστε να μπορέσουμε στη συνέχεια να εντοπίσουμε keypoints μέσω των DoGs. Στην δεύτερη περίπτωση των 3 level για παράδειγμα, έχουμε μόνο 2 DoGs, τα οποία θα είναι οριακά άχρηστα στο επόμενο βήμα, καθώς δεν περιέχουν αρκετή πληροφορία για επαρκή εντοπισμό ακμών.

Στην περίπτωση των πολλών octave επίσης πρέπει να σχολιαστεί πως, με διαδοχικές υποδειγματοληψίες χάνεται πάρα πολύ μεγάλο μέρος της πληροφορίας, και δεν μπορούν τα levels των μεγαλύτερων octave να χρησιμεύσουν περαιτέρω. Για παράδειγμα, στην δεύτερη περίπτωση, τα DoGs στα octaves 6 και 7 είναι τόσο pixelated που μόνο σε μερικώς ή πλήρως λανθασμένο εντοπισμό keypoints θα μπορούσαν να οδηγήσουν.

Συνεπώς, η καλύτερη επιλογή για τον αριθμό των levels και octaves ώστε να μπορέσει να προχωρήσει η διαδικασία και να δώσει σωστά αποτελέσματα στην πορεία είναι ένας σχετικα μεγάλος αριθμός από levels (5 με 8) και ένας σχετικά μικρός αριθμός από octaves (3 με 4).

Φυσικά, το τελευταίο σχόλιο (ο αριθμός των octaves) μπορεί να διαμορφωθεί κατάλληλα ανάλογα και με το μέγεθος της εικόνας που εισάγουμε στην συνάρτηση. Για παράδειγμα, αν οι εικόνες μας αντί για 128x128 ήταν 1280x1280 η επιλογή περισσότερων octave για λεπτομερέστερη μελέτη δεν θα ήταν και τόσο άστοχη.

## 2. Εντοπισμός των salient keypoints

Στο δεύτερο μέρος της εργασίας, καλούμαστε να προχωρήσουμε στον εντοπισμό salient keypoints, δηλαδή σημειών που αποτελούν ακρότατα στην εικόνα μας, και άρα προσδιορίζουν ακμές και περιγράμματα λόγω του έντονου contrast τους.

Η διαδικασία εντοπισμού γίνεται με τη χρήση των DoGs που δημιουργήσαμε στο προηγούμενο βήμα, και συμπεριλαμβάνει σύγκριση των εσωτερικών σημείων με τους γείτονες τους (μόνο των εσωτερικά DoGs), ώστε να εντοπιστούν τοπικά ακρότατα. Πιο συγκεκριμένα, κάθε pixel των DoGs (εκτός από τα pixels που ανήκουν στο περίγραμμα) συγκρίνεται με τους 3^2 - 1 γείτονες του, με τους 3^2 γείτονες του ανώτερου DoG αλλά και με τους 3^2 γείτονες το κατώτερου DoG (σύνολο 3^3 - 1 γείτονες). Το σημείο αποτελεί keypoint αν είναι μεγαλύτερο ή μικρότερο από όλους τους γείτονές του, αν είναι δηλαδή τοπικό ακρότατο σε αυτόν τον 3D χώρο. Στο παραδοτέο κώδικα έχω διαφοροποιήσει ελάχιστα τη διαδικασία, καθώς στην λίστα των γειτόνων συμπεριλαμβάνω και το αρχικό σημείο, και μετα προχωρώ σε σύγκριση ισότητας με το max της λίστας, σε αντίθεση με έλεγχο ακρότατου. Φυσικά, η σύγκριση έχει κάποια σημασια μόνο μεταξύ των DoGs του ίδιου octave.

Η συνάρτηση που καλούμαστε να υλοποιήσουμε είναι η:

**function keypoints = myKeypoints(DoGs)**

η οποία δέχεται σαν όρισμα το cell array των DoGs που δημιουργήσαμε στο προηγούμενο βήμα, και επιστρέφει μια λίστα σημείων που αποτελούν keypoints της εικόνας. Εδώ η λίστα αυτή έχει τη μορφή Nx4 πίνακα, με N τον αριθμό των keypoints που βρήκαμε. Τα στοιχεία του keypoint ορίζονται σε κάθε 4άδα ως (o, s, m, n), δηλαδή octave, level και συντεταγμένες m και n.

### Απόρριψη των low contrasted keypoints

Σαν τελευταίο κομμάτι της εργασίας, και πλήρως συνδεδεμένο με το παραπάνω βήμα, καλούμαστε να υλοποιήσουμε μια συνάρτηση που θα «φιλτράρει» τα μικρής αντίθεσης keypoints ώστε να έχουμε ένα ξεκάθαρο τελικό αποτέλεσμα, στο οποίο θα φαίνονται με ευκρίνεια περιγράμματα και σημεία «μεγάλης σημασίας», τα οποία μπορούν να χρησιμοποιηθούν μετά σε διαδικασίες π.χ. reconstruction, οι οποίες όμως είναι εκτός των πλαισίων της συγκεκριμένης εργασίας.

Η γενική ιδέα είναι πως γίνεται μια σύγκριση «φωτεινότητας» σε όλα τα pixel που έχουμε ήδη χαρακτηρίσει ως keypoints, και ανάλογα με την αυστηρότητα που διακατέχει τη διαδικασία, λίγα ή πολλά από αυτά απορρίπτονται ως low contrasted, ότι δηλαδή δεν δίνουν αρκετά σημαντική ή περαιτέρω χρήσιμη πληροφορία.

Η συνάρτηση που υλοποιήθηκε είναι η εξής:

**function keypointsHighC = discardLowContrasted(DoGs, keypoints, t, p)**

όπου DoGs το γνωστό πλέον cell array, keypoints ο πίνακας-λίστα των keypoints από την προηγούμενη συνάρτηση, t μία σταθερά που θα ορισθεί παρακάτω και p ο «βαθμός αυστηρότητας» της διαδικασίας (όσο μεγαλύτερο είναι το p, τόσα περισσότερα keypoints φιλτράρονται και απορρίπτονται). Φυσικά, η επιστρεφόμενη μεταβλητή keypointsHighC είναι ο πίνακας-λίστα των pixel που πέρασαν τον έλεγχο και αποτελούν keypoints με μεγάλη αντίθεση. Η μορφή του είναι ακριβώς η ίδια με την μορφή του πίνακα keypoints της προηγούμενης συνάρτησης.

Η σταθερά t ορίζεται ως εξής:

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

όπου nspo = levels – 2. Εδώ (επειδή δεν αναφερόταν κάτι παραπάνω) έκανα την αυθαίρετη επιλογή να θεωρήσω πως τα levels που αναφέρονται είναι ο αριθμός levels ανά octave των DoGs και όχι των spacescales (δηλαδή τυπικά levels – 3 όσον αφορά ένα spacescales octave), αν και τα αποτελέσματα ήταν παρόμοια και στις 2 περιπτώσεις.

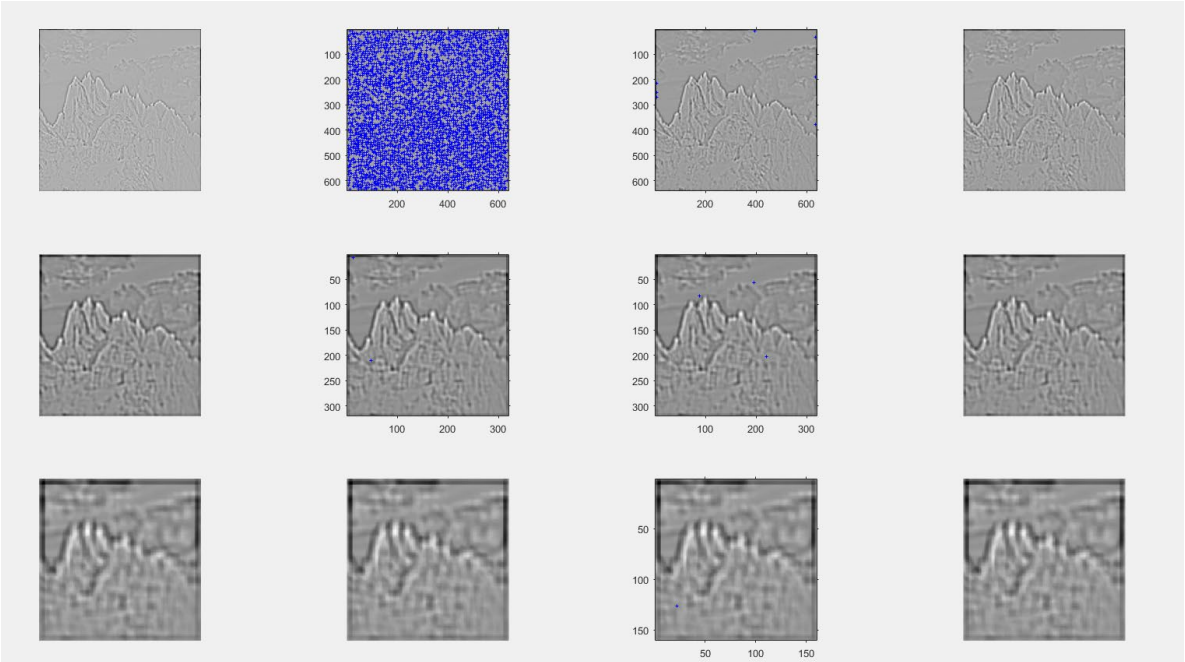
## Demo 2

Στο δεύτερο και τελευταίο demo της εργασίας καλούμαστε να εφαρμόσουμε τις παραπάνω συναρτήσεις και να παρουσιάσουμε τα keypoints που προκύπτουν με και χωρίς φιλτράρισμα. Οι προτεινόμενες αρχικές τιμές είναι σ = √2, K = 7, levels = 5 και octaves = 3 (καθώς και k = √2). Αρχικά το p δίνεται ίσο με 0.8. Τα αποτελέσματα για τα παραπάνω είναι αρκετά κακά (δε βρίσκονται kypeoints ή βρίσκονται πολύ λίγα) οπότε έχω προχωρήσει σε αλλαγές κάποιων παραμέτρων. Στη συνέχεια παρουσιάζονται αλλαγές διαφόρων παραμέτρων ώστε να

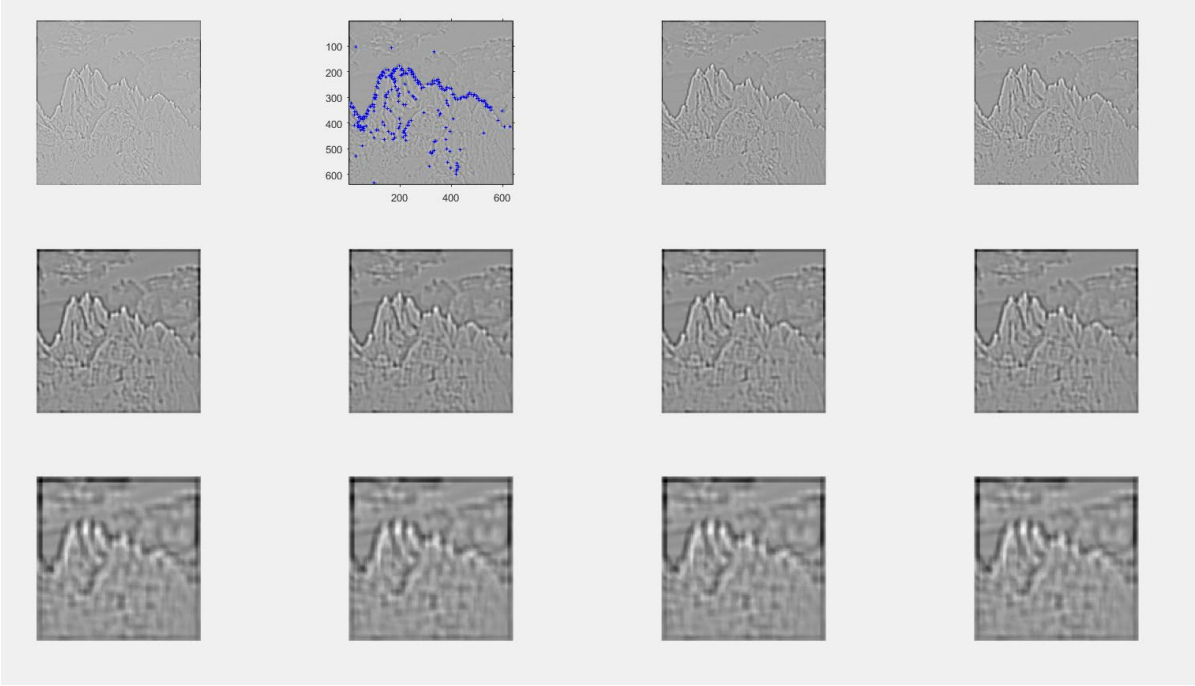
μελετηθεί το καλύτερο αποτέλεσμα. Σημειώνεται ότι γίνεται ένα resize στις εικόνες (x5) ώστε να εντοπίζονται καλύτερα τα keypoints.

1. mountains (σ = √2, K = 13, levels = 5, octaves = 3, k = √2, p = 0.8)

**KEYPOINTS**

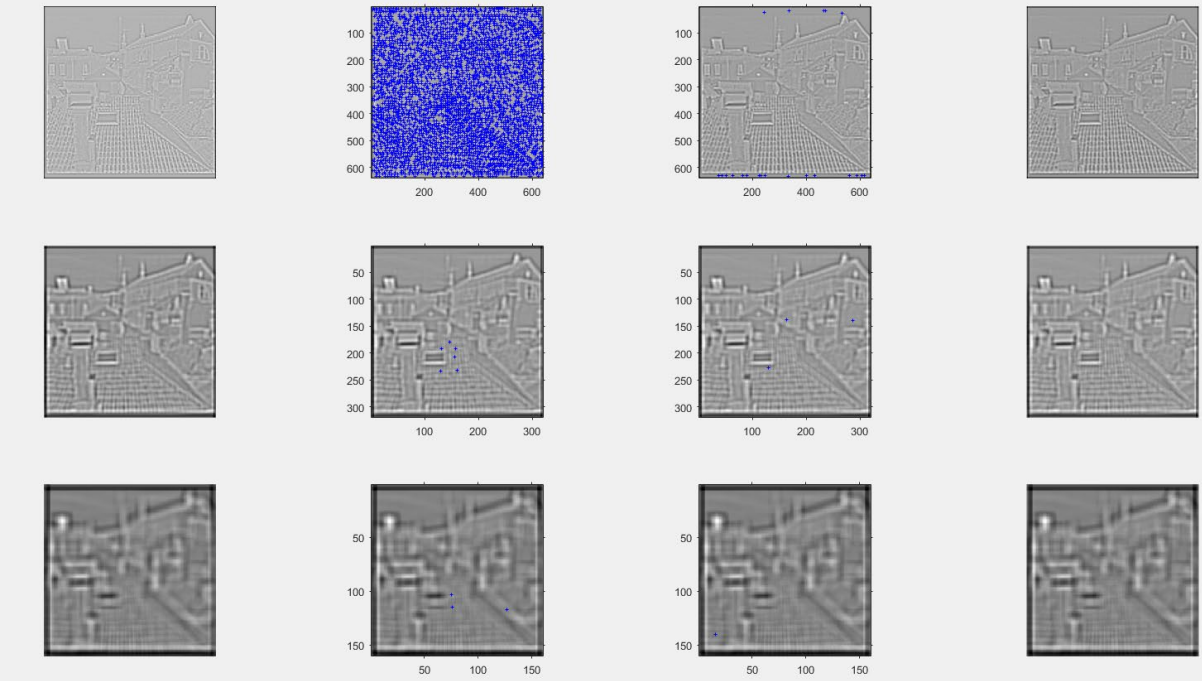


**HIGH CONTRASTED KEYPOINTS**

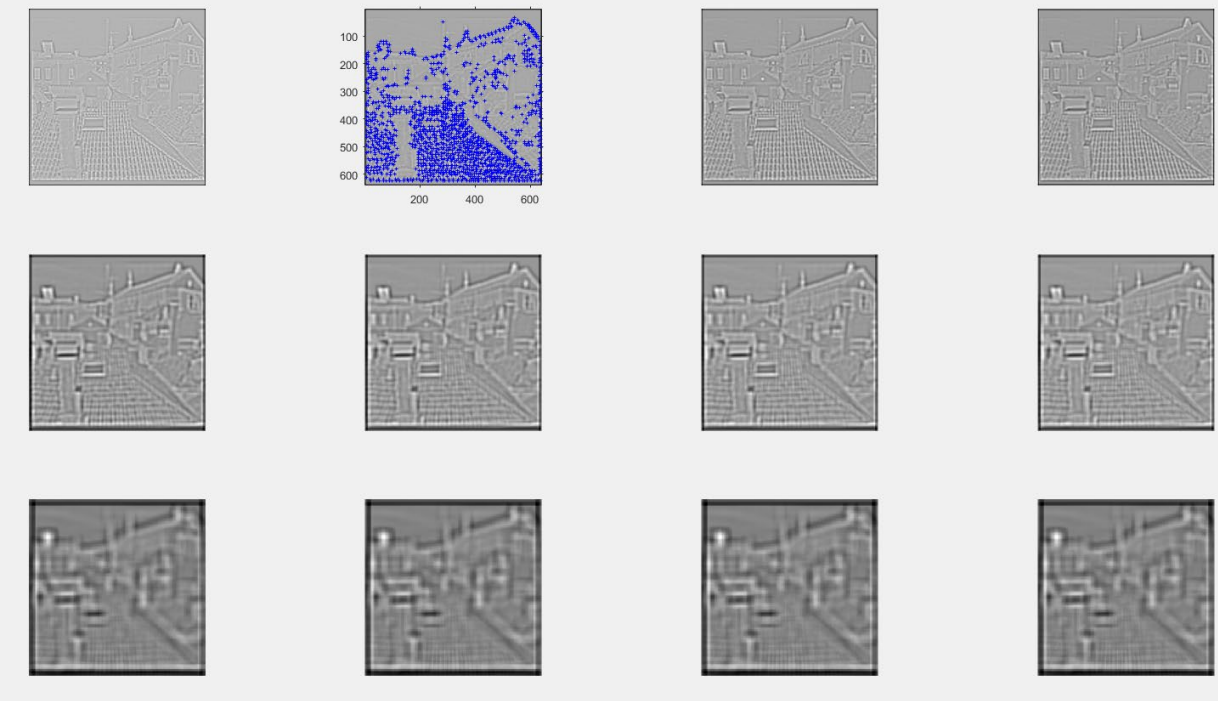


2. roofs (σ = √2, K = 13, levels = 5, octaves = 3, k = √2, p = 0.8)

**KEYPOINTS**

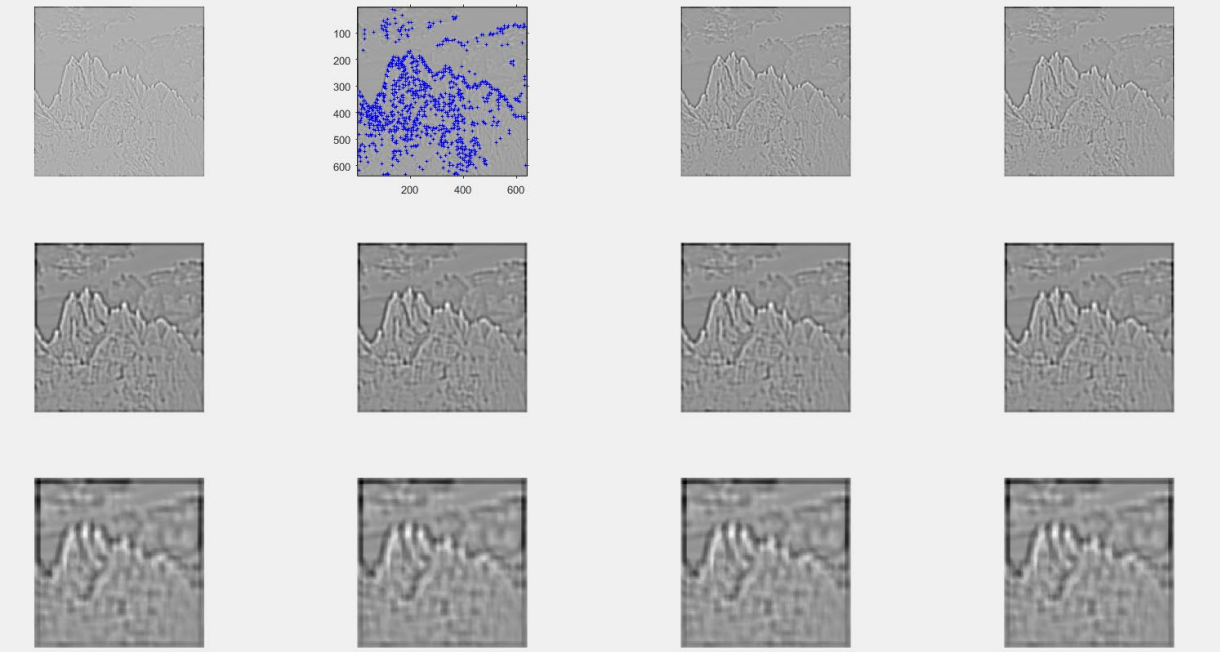


**HIGH CONTRASTED KEYPOINTS**



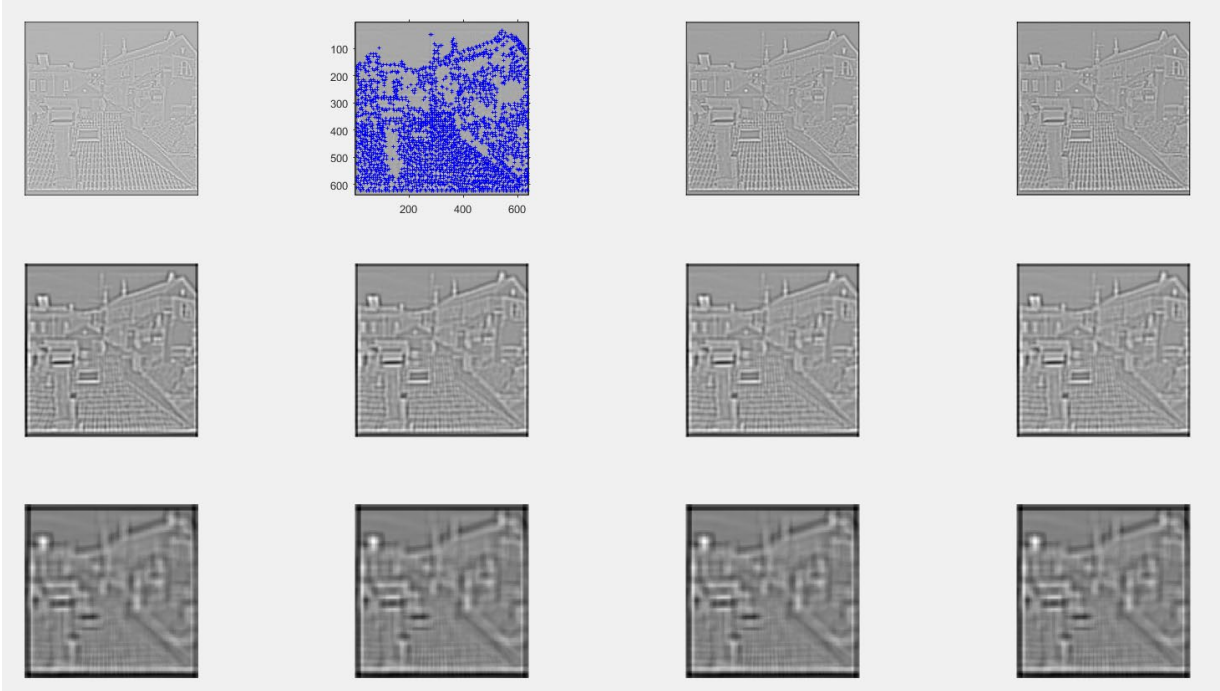
3. mountains (σ = √2, K = 13, levels = 5, octaves = 3, k = √2, **p = 0.4**)

**HIGH CONTRASTED KEYPOINTS**




4. roofs (σ = √2, K = 13, levels = 5, octaves = 3, k = √2, **p = 0.4**)

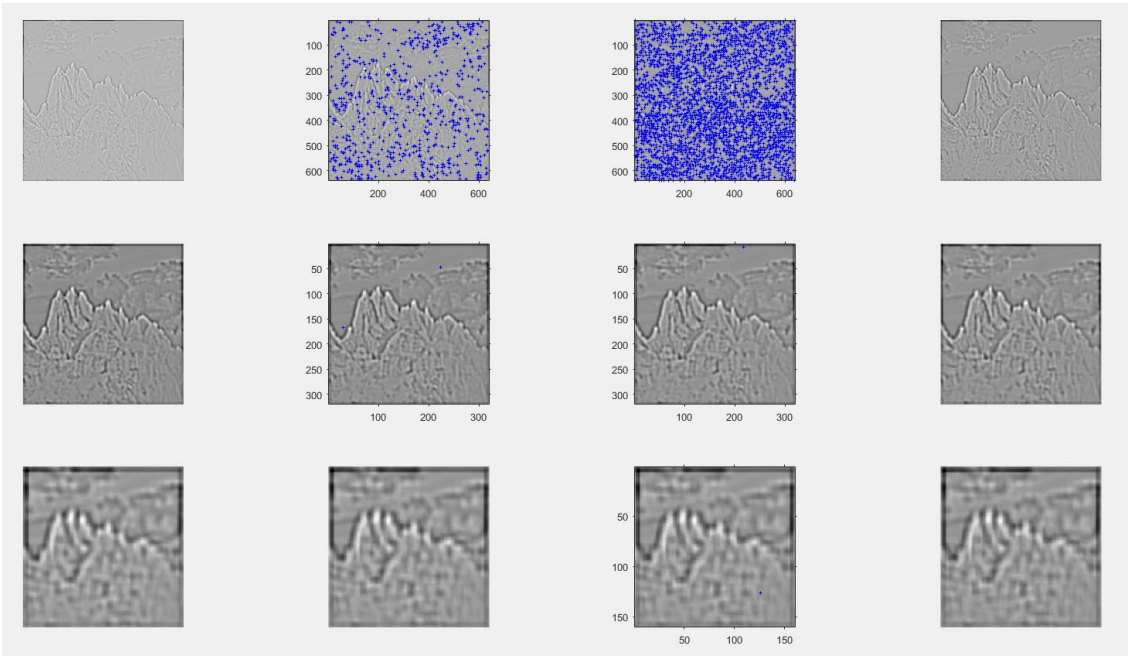
**HIGH CONTRASTED KEYPOINTS**



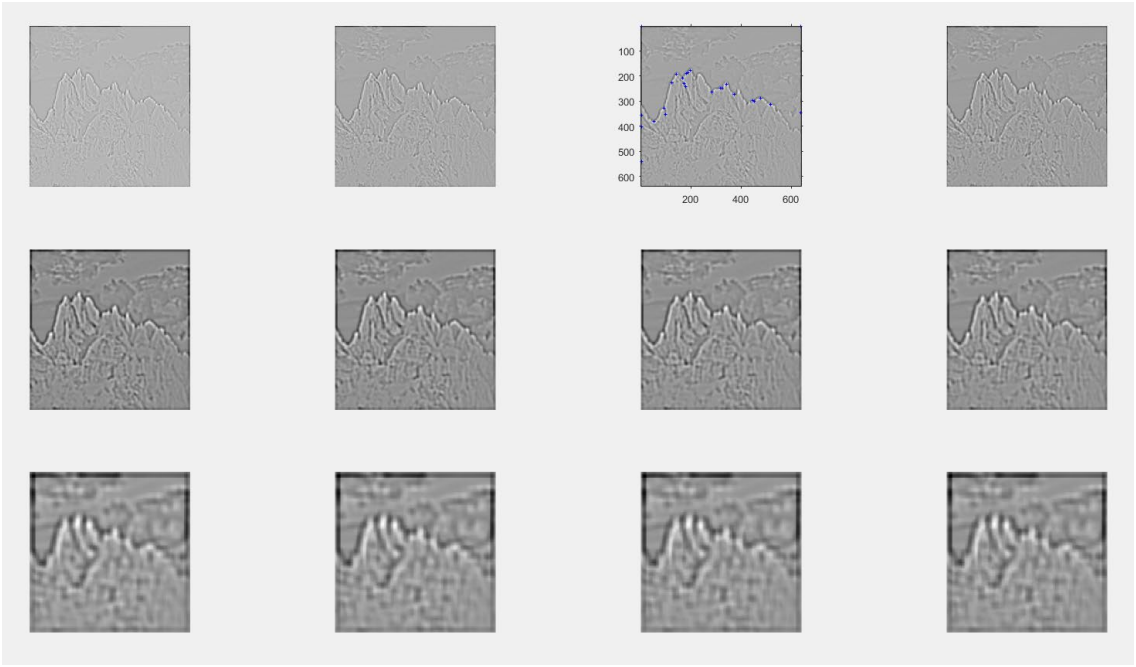
Παρατηρείται πως δεν εμφανίζονται πολλά keypoints σε όλα τα levels, και ειδικά στα levels των μεγαλύτερων octave. Μετά από παρατήρηση, αυτό φαίνεται να αλλάζει με την αλλαγή της σταθεράς k, οπότε προχωράμε σε μικρότερη τιμή της.

1. mountains (σ = √2, K = 13, levels = 5, octaves = 3, k = 1.2, p = 0.8)

**KEYPOINTS**

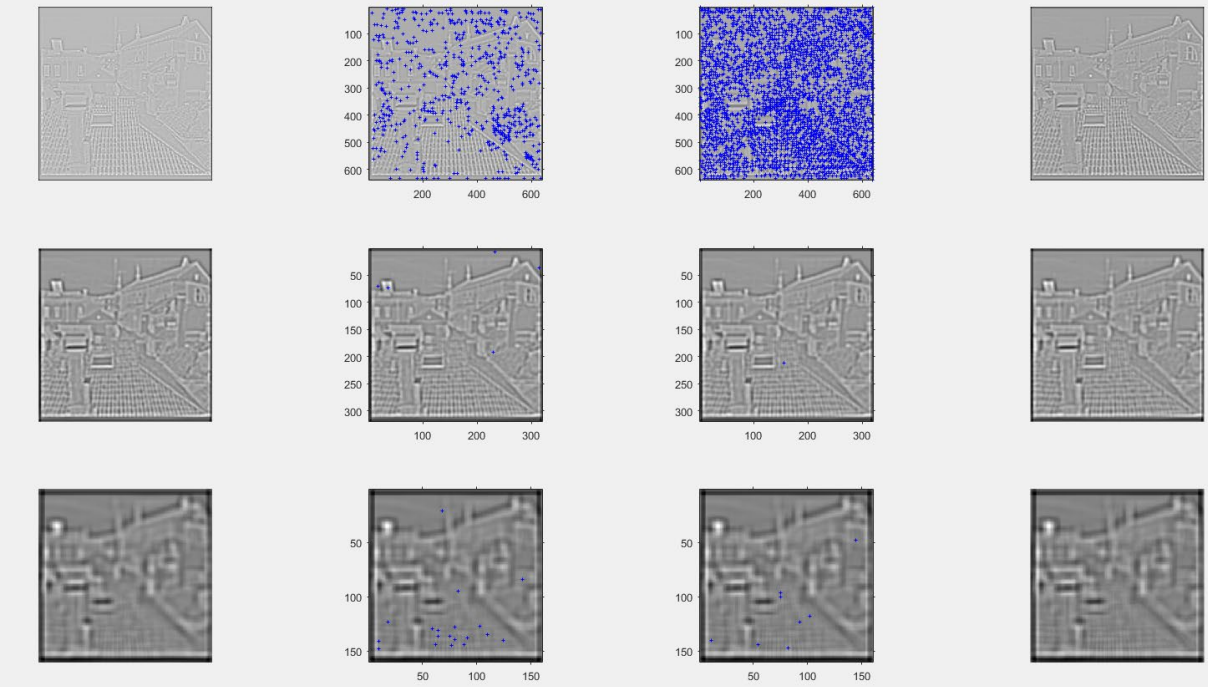


**HIGH CONTRASTED KEYPOINTS**

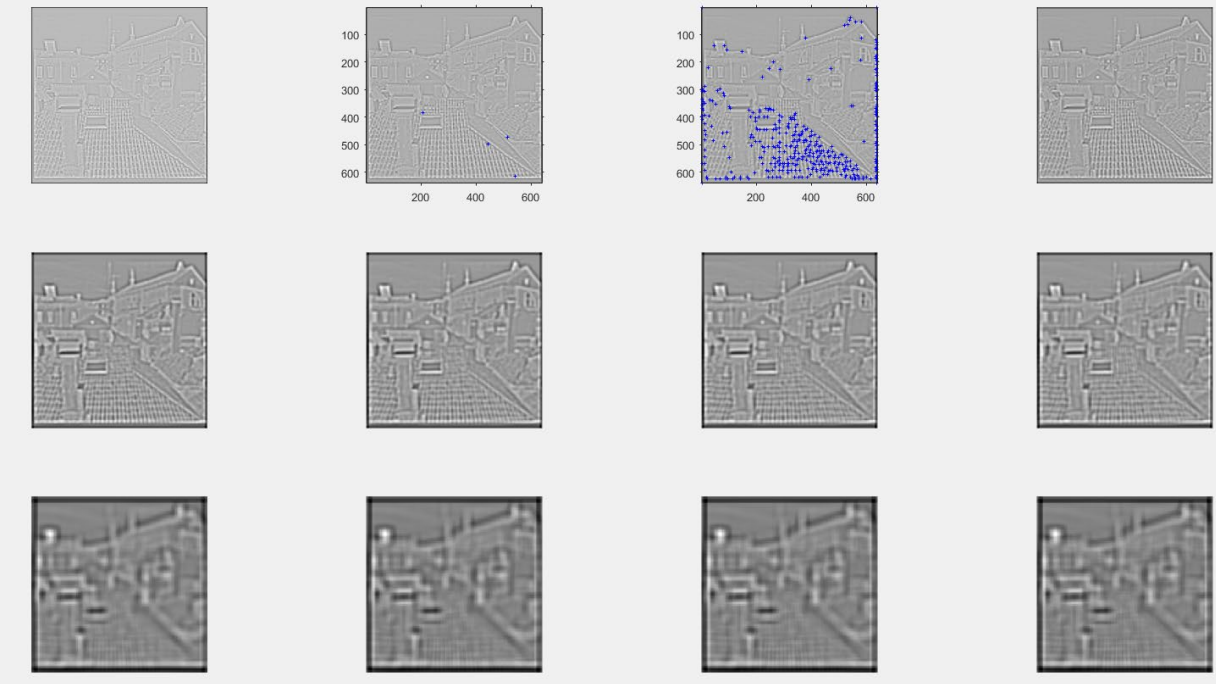


2. roofs (σ = √2, K = 13, levels = 5, octaves = 3, k = 1.2, p = 0.8)

**KEYPOINTS**

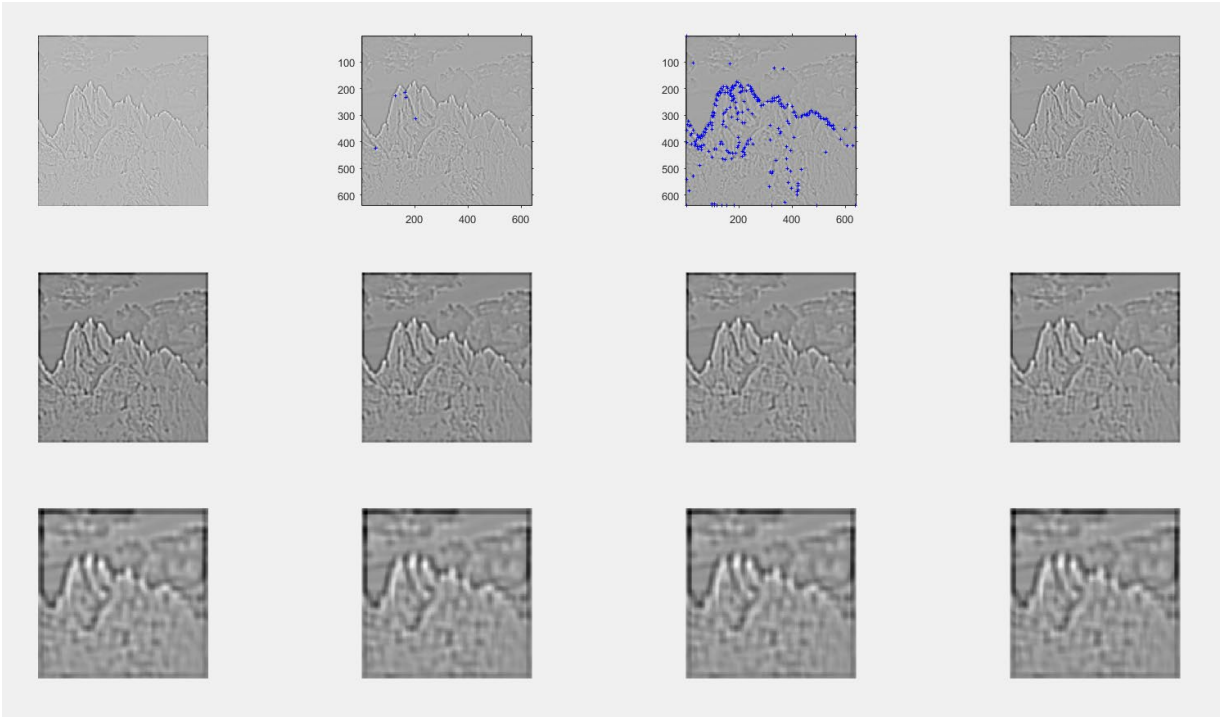


**HIGH CONTRASTED KEYPOINTS**



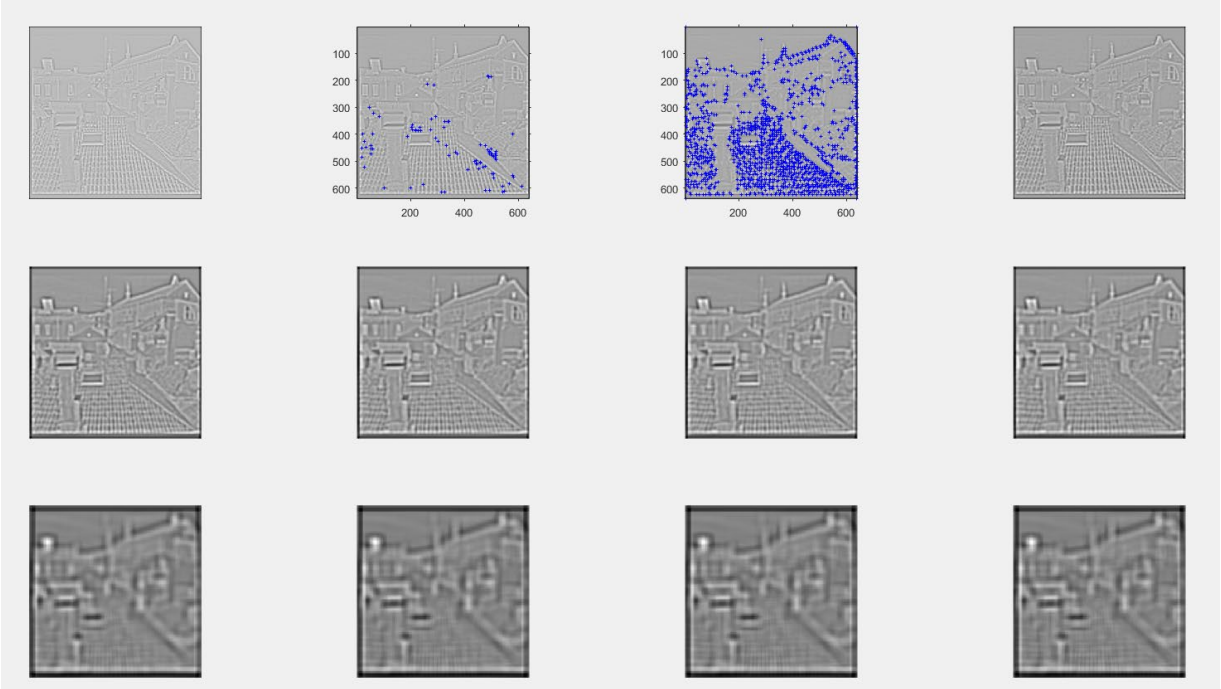
3. mountains (σ = √2, K = 13, levels = 5, octaves = 3, k = 1.2, **p = 0.4**)

**HIGH CONTRASTED KEYPOINTS**




4. roofs (σ = √2, K = 13, levels = 5, octaves = 3, k = 1.2, **p = 0.4**)

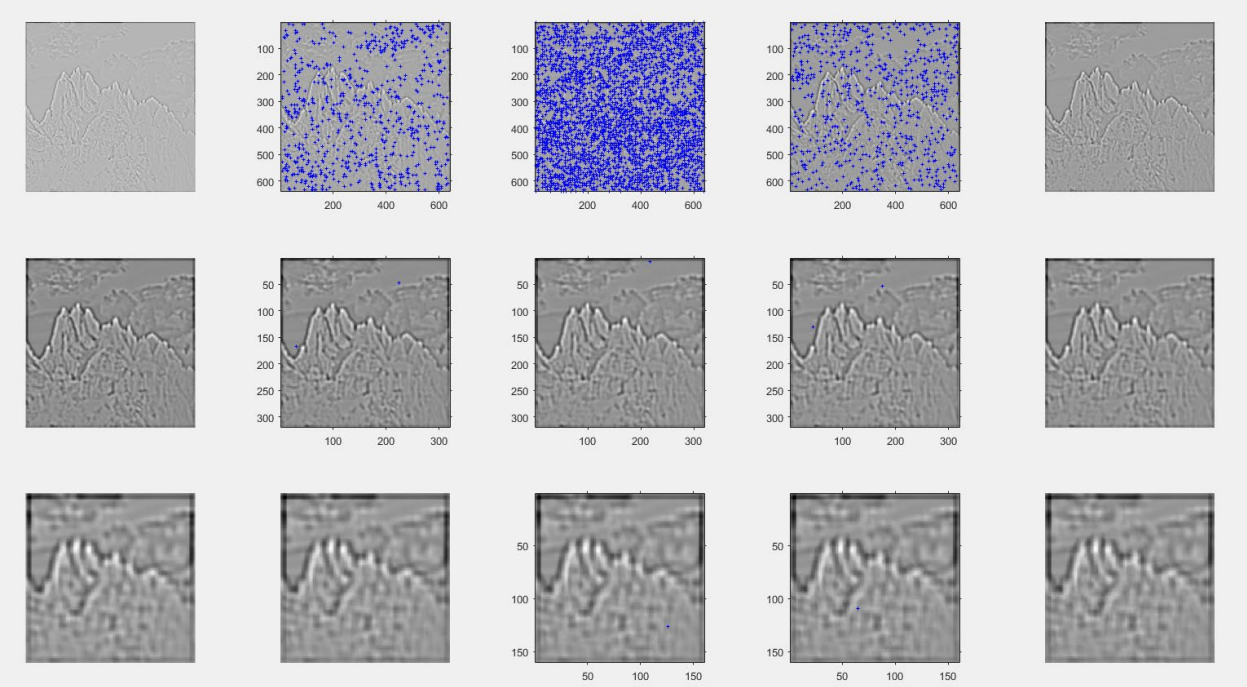
**HIGH CONTRASTED KEYPOINTS**



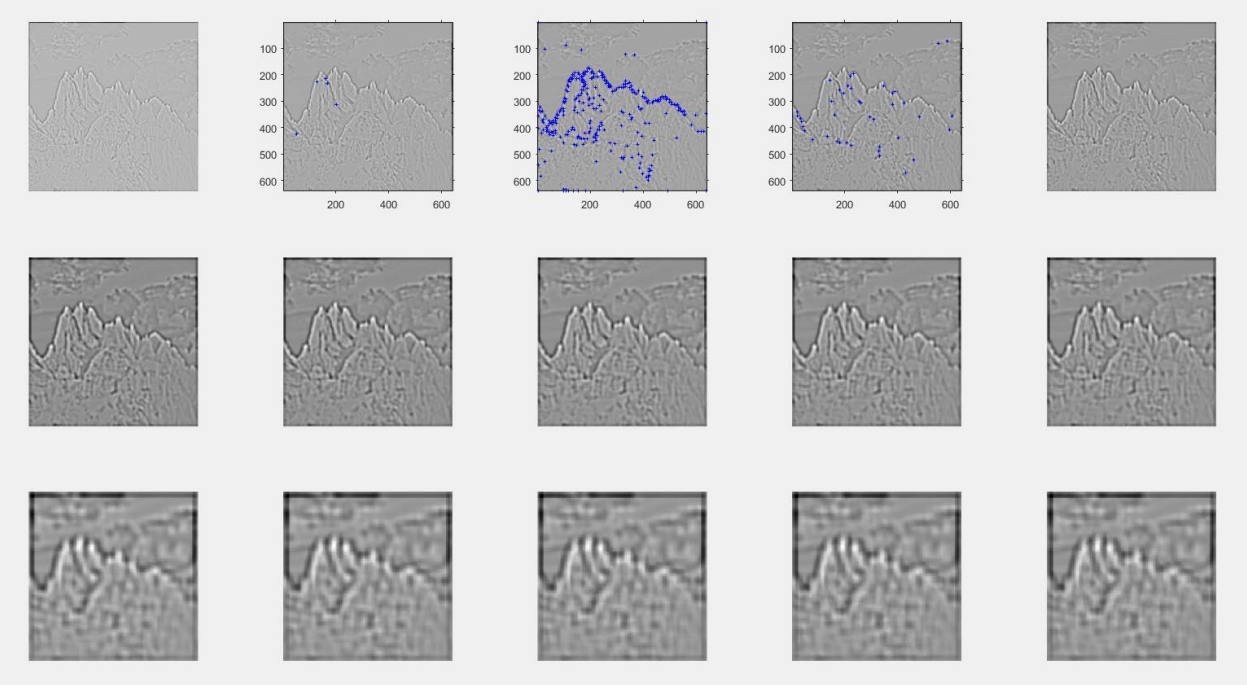
Και κάποιες δοκιμές παραπάνω με levels = 6:

1. mountains (σ = √2, K = 13, levels = 6, octaves = 3, k = 1.2, p = 0.6)

**KEYPOINTS**

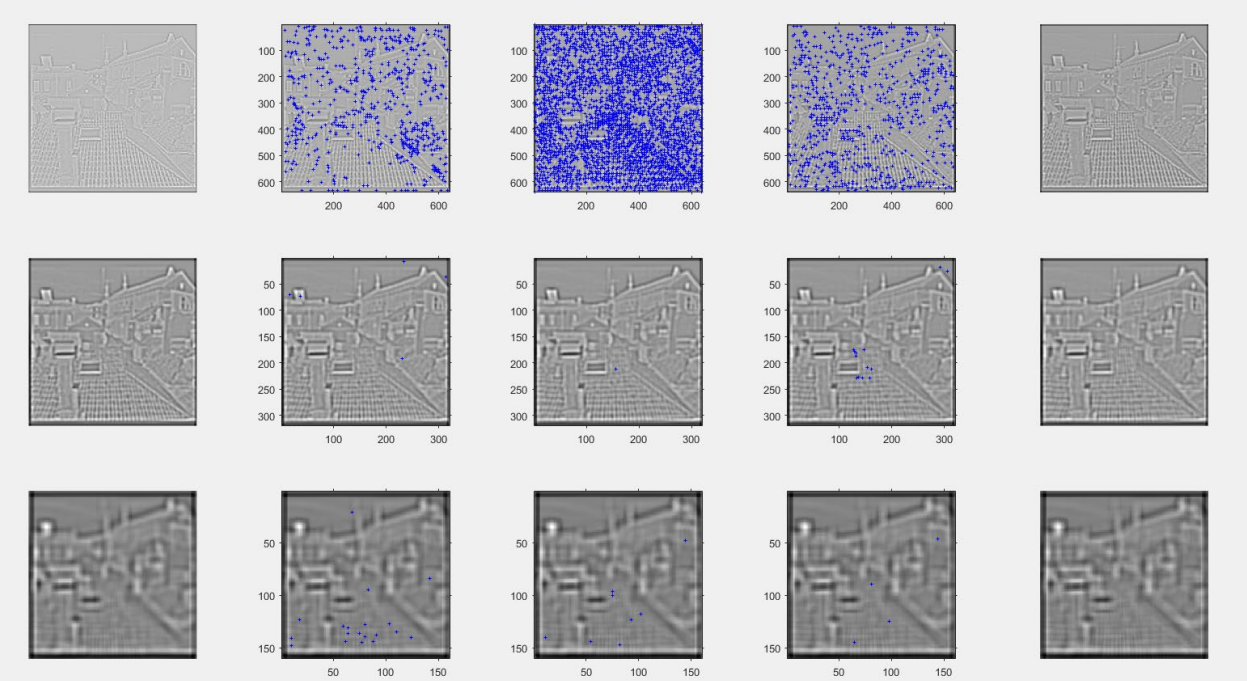


**HIGH CONTRASTED KEYPOINTS**

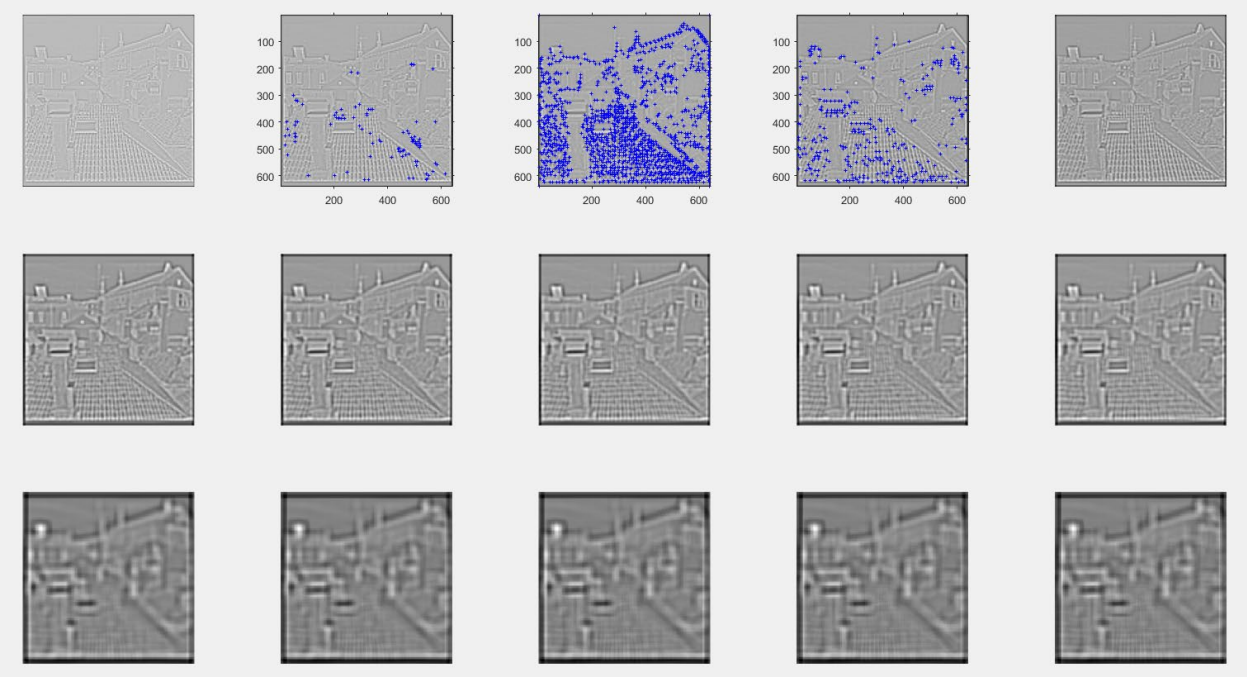


2. roofs (σ = √2, K = 13, levels = 6, octaves = 3, k = 1.2, p = 0.6)

**KEYPOINTS**



**HIGH CONTRASTED KEYPOINTS**

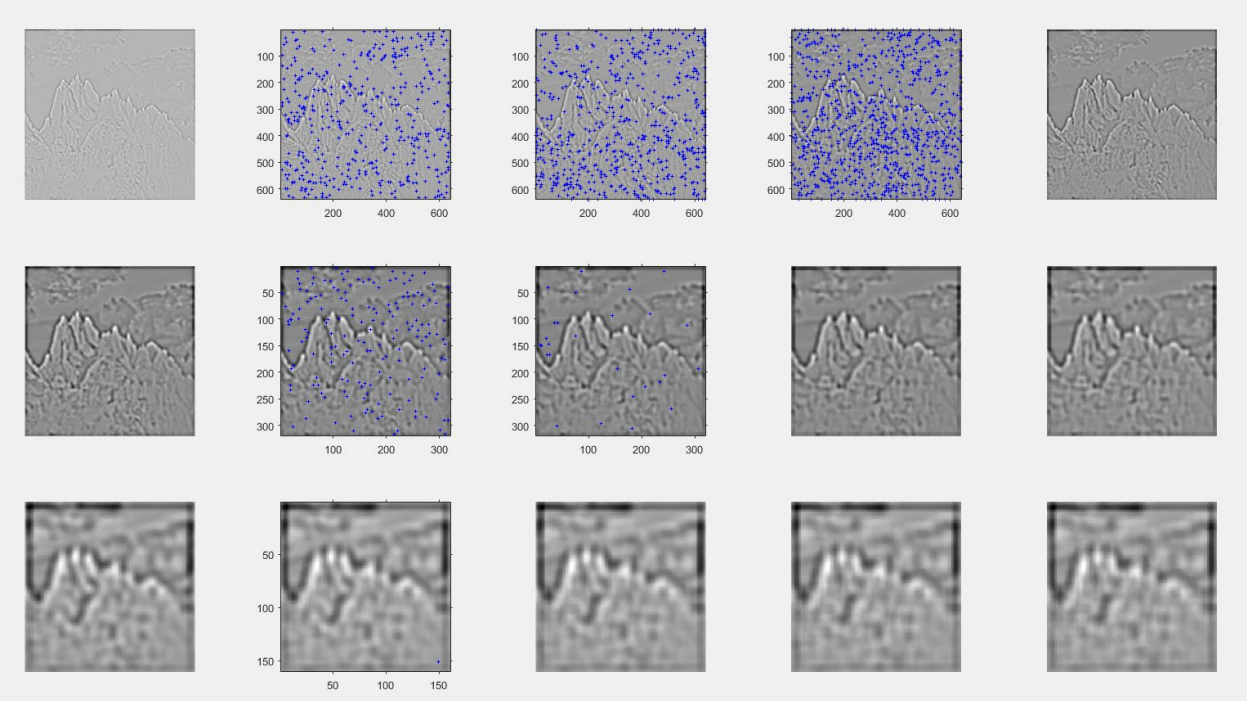


Αυτό που αξίζει να σημειωθεί είναι πως τα keypoints (αφιλτράριστα και μη) εντοπίζονται συνήθως στο πρώτο octave. Αυτό κατά πάσα πιθανότητα έχει να κάνει με την λεπτομέρεια της φωτογραφίας (καθώς δεν έχει υποστεί ακόμα downsampling). Ο αλγόριθμος για τον έλεγχο των high contrasted keypoints λειτουργεί ικανοποιητικά, απλά η κλασσική τιμή του p που δίνεται (0.8) πολλές φορές πρέπει να αλλαχθεί αν υπάρξουν αλλαγές σε άλλες παραμέτρους. Το k επίσης φαίνεται να επηρεάζει το αποτέλεσμα, και παρατηρείται ότι δε πρέπει να ξεφεύγει από το σύνολο [1, 1.5] καθώς αλλιώς παρατηρούνται πολύ χειρότερα αποτελέσματα. Σημειώνεται επίσης πως το μεγαλύτερο K δίνει συνήθως περισσότερα/καλύτερα keypoints, οπότε αξίζει ο πειραματισμός πάνω και σε αυτό.

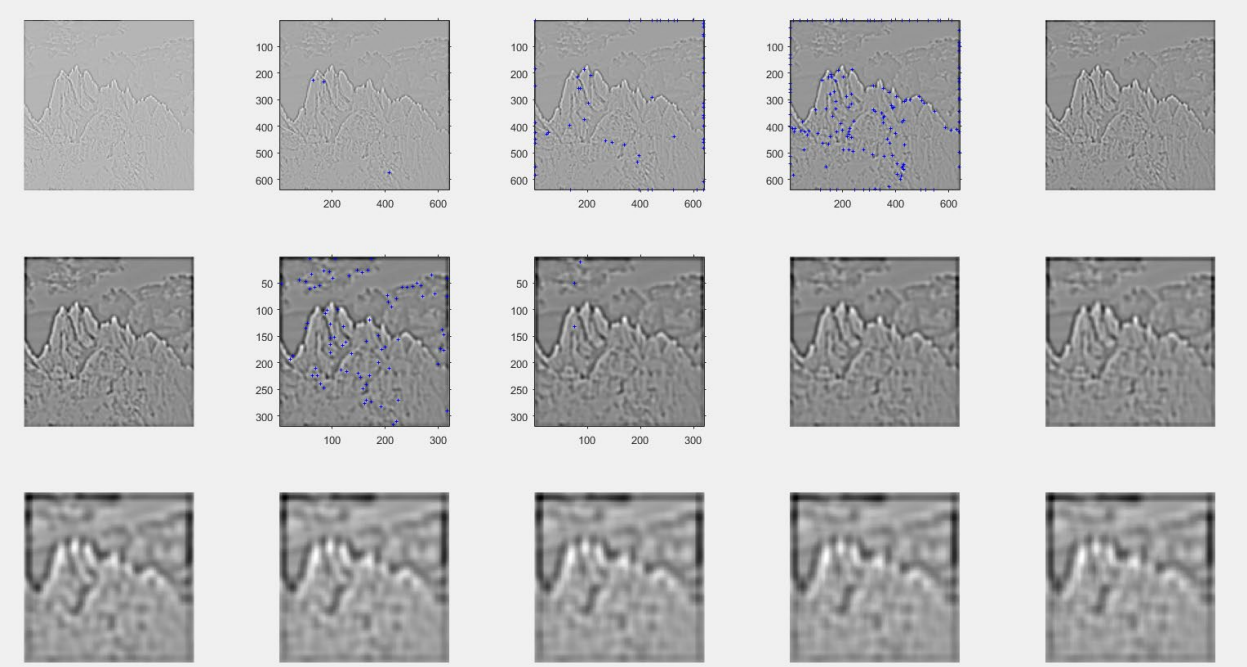
Παρουσιάζεται παρακάτω ένα τελευταίο σετ πειραμάτων με K = 19:

1. mountains (σ = √2, K = 19, levels = 6, octaves = 3, k = 1.2, p = 0.6)

**KEYPOINTS**

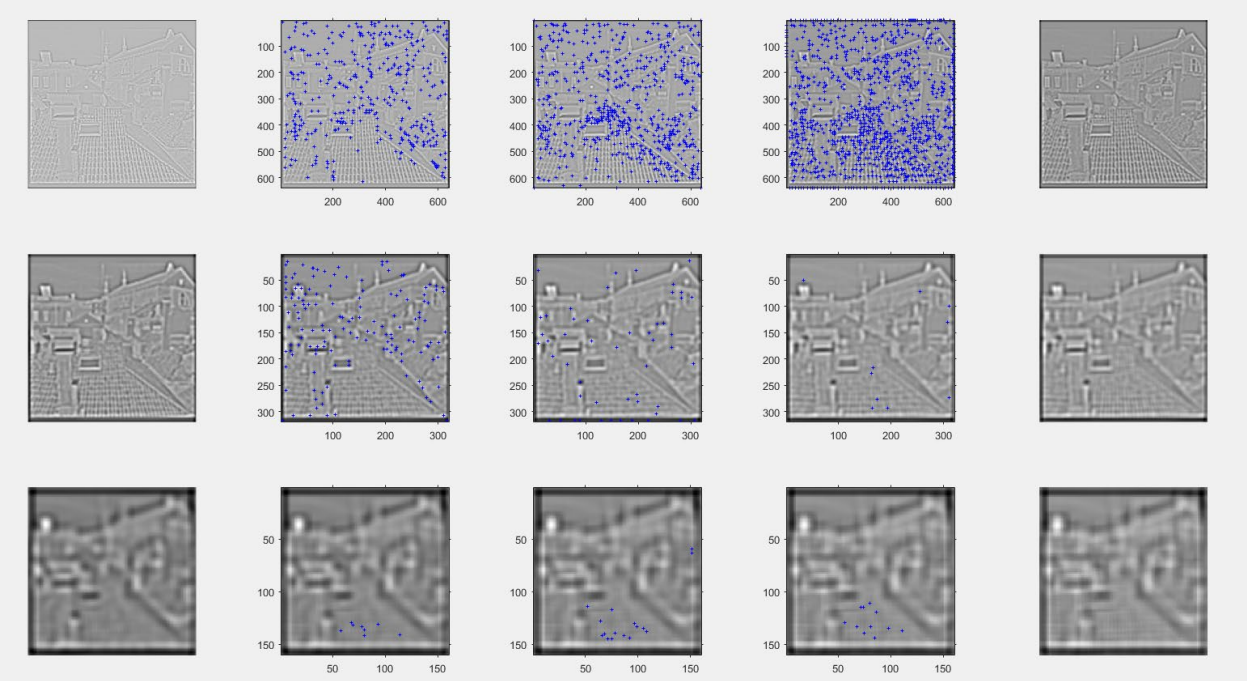


## HIGH CONTRASTED KEYPOINTS

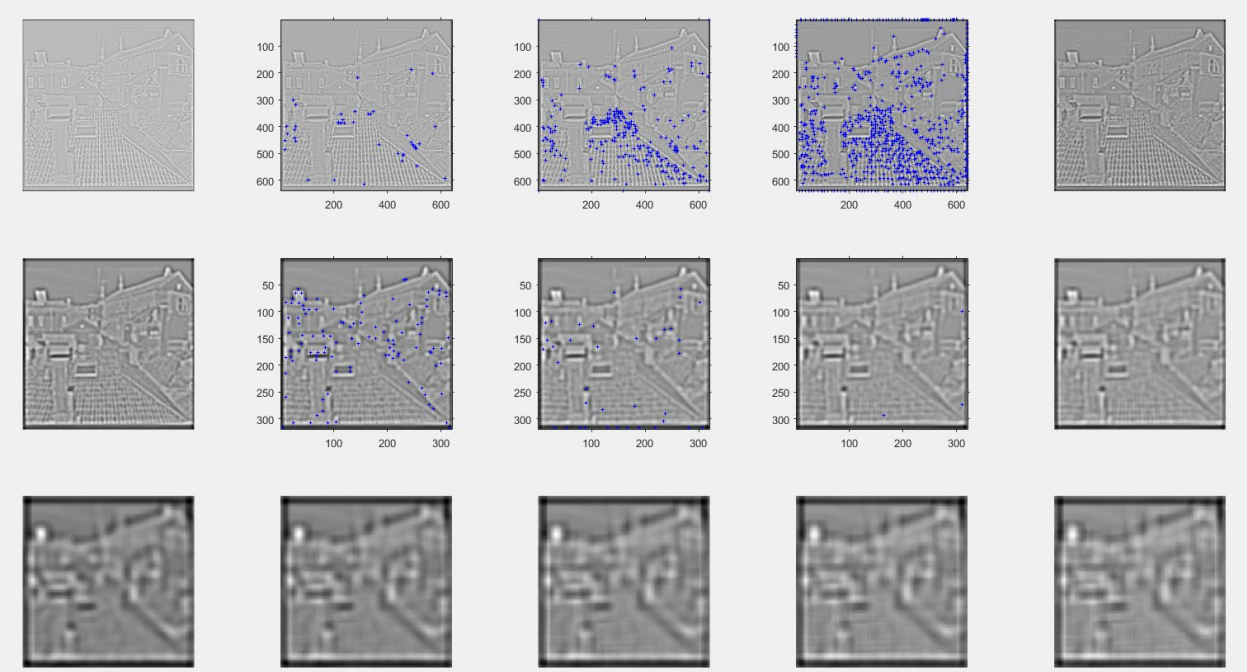


2. roofs (σ = √2, K = 19, levels = 6, octaves = 3, k = 1.2, p = 0.6)

## KEYPOINTS



**HIGH CONTRASTED KEYPOINTS**



Τελευταίο σχόλιο που θα μπορούσε να αναφερθεί είναι πως φωτογραφίες όπως αυτές των βουνών έχουν πολύ καλύτερο αποτέλεσμα, λόγω της έλλειψης μεγάλης λεπτομέρειας, καθώς και της πιο ξεκάθαρης δομής της (οι βουνοκορφές φαίνονται με το μάτι, οπότε αντίστοιχα εντοπίζονται σχετικά εύκολα και από το «μάτι» του κώδικα). Όπως και να’χει, ο περαιτέρω πειραματισμός αξίζει, καθώς μόνο έτσι μπορεί να βρεθεί η καλύτερη εκδοχή του αλγορίθμου καθώς και οι τιμές των παραμέτρων που οδηγούν σε ακριβέστερα αποτελέσματα.