

Τσόχλας Ιωάννης

AM 4993 | cs04993@uoi.gr

ΜΥΕ037

Ψηφιακή Επεξεργασία Εικόνας

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2023-2024



ΕΡΓΑΣΙΑ

Project Goal :

Στόχος της εργασίας είναι η εκμάθηση και εφαρμογή τεχνικών ψηφιακής επεξεργασίας εικόνας χρησιμοποιώντας τη γλώσσα προγραμματισμού Python και βιβλιοθήκες όπως Numpy, SciPy και Matplotlib

- Κατασκευή τμημάτων εικόνας (image patches) και κανονικοποίησή τους.
- Υλοποίηση συνέλιξης (convolution) και χρήση Γκαουσιανού φίλτρου.
- Ανίχνευση ακμών μέσω τελεστών όπως Sobel και LoG.

Github Reposiotry : <https://github.com/GiannisTsochlas/PsifiakiEpejergasiaEikonas>

Contents

Τοπικές περιοχές/Τμήματα εικόνας	2
ΤΦιλτράρισμα στο πεδίο του χώρου	3
Αξιοσημείωτες Πηγες Μελέτης	10

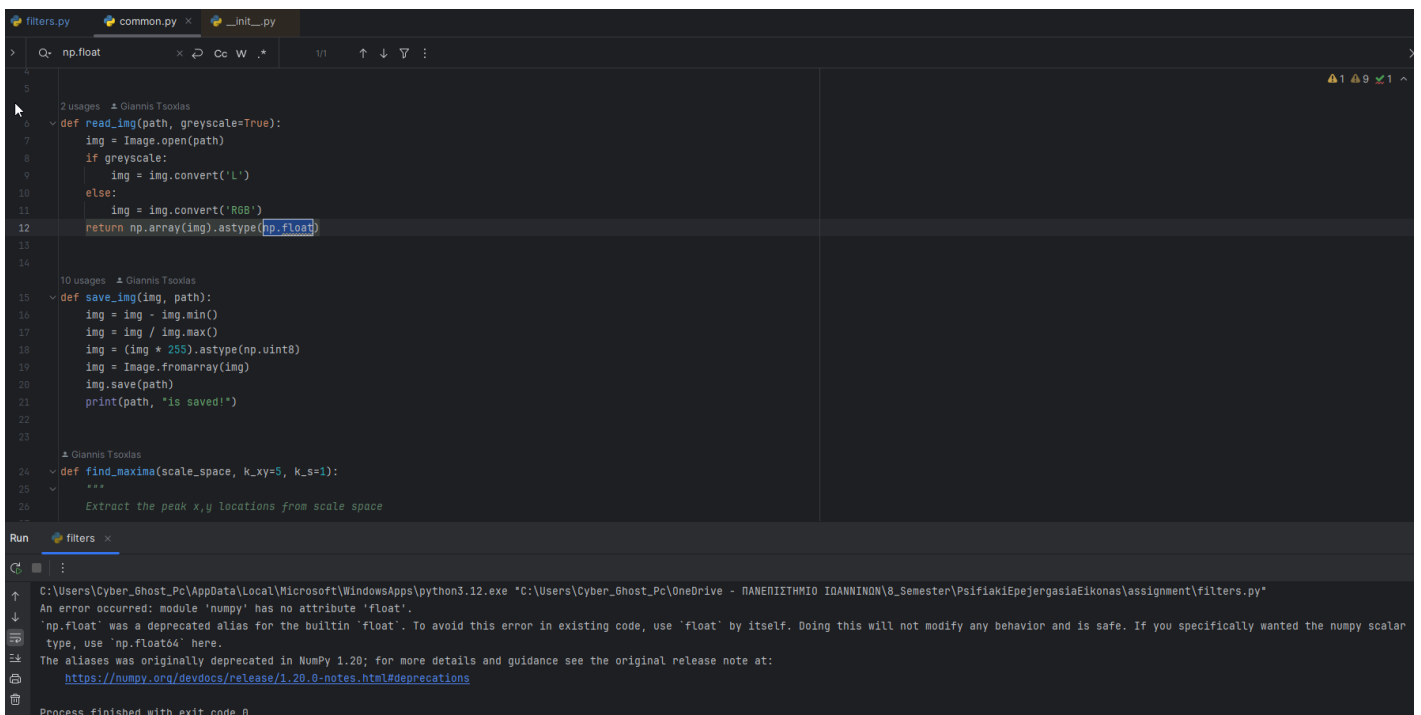
Μέρος 1 Τοπικές περιοχές/Τμήματα εικόνας (Patches)

Αξιοσημείωτο θεωρώ να επισημάνω τις αλλαγές πλαισίων που έκανα για να τρέξει ο κώδικας μιας και μετά την υλοποίηση της **image_patches**, υπήρχαν errors.

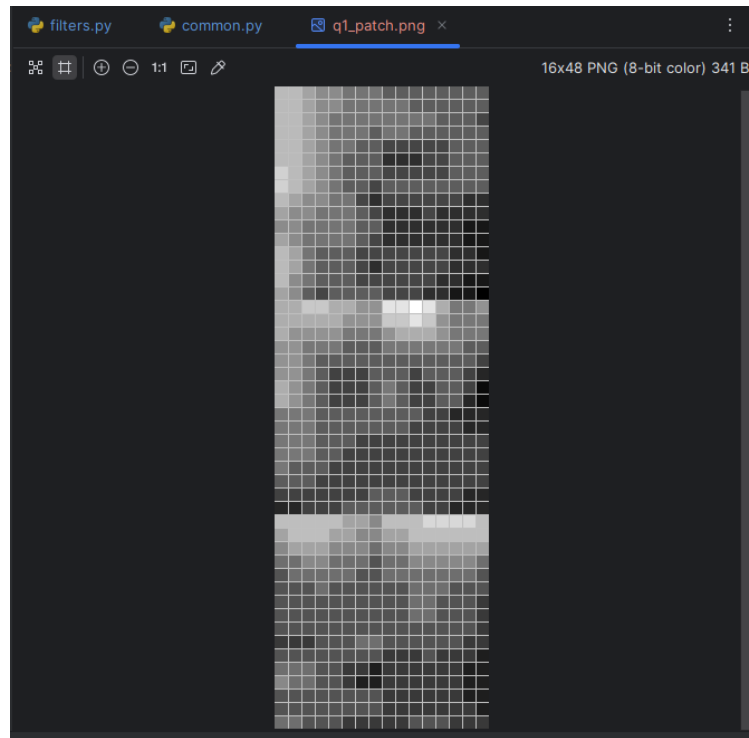
- Debug για τη διαχείριση των errors (τα παρακάτω αντικείμενα είναι κενά γιατί δεν τα έχουμε υλοποιήσει ακόμα για αυτό το λόγο βάζουμε μια συνθήκη τύπου try catch για να καταφέρουμε να έχουμε κάποιο αποτέλεσμα στην κονσόλα:

```
10 usages 1 Giannis Tsoulas
15 def save_img(img, path):
16     if img is not None:                για το error
17         img = img - img.min()
18         img = img / img.max()          img = img - img.min()
19         img = (img * 255).astype(np.uint8)      ^^^^^^^
20         img = Image.fromarray(img)             AttributeError: 'NoneType' object has no attribute 'min'
21         img.save(path)
22         print(path, "is saved!")             Process finished with exit code 1
23     else:
24         print("Image is None, cannot save.")
```

- Η κονσόλα μας επισήμανε να αλλάξουμε τη μεταβλητή `np.float` σε `np.float64`



```
filters.py  common.py  _init_.py
> Q: np.float  x C: W * 1/1 ↑ ↓ ▾ ⋮
5
6 2 usages 1 Giannis Tsoulas
7 def read_img(path, grayscale=True):
8     img = Image.open(path)
9     if grayscale:
10         img = img.convert('L')
11     else:
12         img = img.convert('RGB')
13     return np.array(img).astype(np.float)
14
15 10 usages 1 Giannis Tsoulas
16 def save_img(img, path):
17     img = img - img.min()
18     img = img / img.max()
19     img = (img * 255).astype(np.uint8)
20     img = Image.fromarray(img)
21     img.save(path)
22     print(path, "is saved!")
23
24 1 Giannis Tsoulas
25 def find_maxima(scale_space, k_xy=5, k_s=1):
26     """
27     Extract the peak x,y locations from scale space
28
29 Run filters
30 C:\Users\Cyber_Ghost_Pc\AppData\Local\Microsoft\WindowsApps\python3.12.exe "C:\Users\Cyber_Ghost_Pc\OneDrive - ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ\8_Semester\PsifiakiEpejergasiaEikonas\assignment\filters.py"
31 An error occurred: module 'numpy' has no attribute 'float'.
32 'np.float' was a deprecated alias for the builtin 'float'. To avoid this error in existing code, use 'float' by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use 'np.float64' here.
33 The aliases was originally deprecated in NumPy 1.20; for more details and guidance see the original release note at:
34 https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
35 Process finished with exit code 0
```



Τα 3 τμήματα τις εικόνας από το αρχείο grace_hopper.png

- (a) Η κανονικοποίηση των τμημάτων της εικόνας ώστε να έχουν μηδενική μέση τιμή έχει αρκετά πλεονεκτήματα . Βοηθά στη βελτίωση της ποιότητας της εικόνας και της απόδοσης των αλγορίθμων επεξεργασίας εικόνας, μειώνοντας παράλληλα την επίδραση του θορύβου.

The mean value and the standard deviation of the new image show that the effect of noise is reduced.

Υπόδειξη:

Το δεύτερο σενάριο, όπου η τιμή της σκοτεινής αντιστοιχεί σε -1 και η τιμή της φωτεινής σε 1, είναι πιο κατάλληλο για να μετρήσουμε την ομοιότητα χρησιμοποιώντας εσωτερικά γινόμενα. Αυτό οφείλεται στο γεγονός ότι η συμμετρική αυτή κλίμακα παρέχει μια πιο ισορροπημένη και λεπτομερή μέτρηση των διαφορών φωτεινότητας

- (b)
1. Όταν το αντικείμενο μετακινείται μέσα στην εικόνα, η μηδενική μέση τιμή διασφαλίζει ότι η συνολική φωτεινότητα της εικόνας δεν επηρεάζεται από τη θέση του αντικειμένου, γιατί οι θετικές και αρνητικές τιμές θα αλληλοακυρώνονται.
 2. Η αλλαγή κλίμακας μπορεί να επηρεάσει την ένταση των τιμών των εικονοστοιχείων.
 3. Η σταθερή μέση τιμή βοηθά στη διατήρηση της σύγκρισης ανεξάρτητα από την ένταση του φωτός.

Συμπέρασμα

Η χρήση τμημάτων με μηδενική μέση τιμή είναι ιδιαίτερα χρήσιμη για προβλήματα όπως η αντιστοίχιση και η αναγνώριση αντικειμένων.

Μέρος 2 Φιλτράρισμα στο πεδίο του χώρου

Ζήτημα 2: Συνέλιξη και Γκαουσιανό φίλτρο (Convolution and Gaussian Filter)

- a) Επιλέγουμε ένα συγκεκριμένο μέγεθος φίλτρου k . Για το παράδειγμα αυτό, ας θεωρήσουμε ότι το $k=3$

Ορίζουμε ένα 2D φίλτρο Gaussian G μεγέθους 3×3 .
$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Ορίζουμε δύο 1D φίλτρα Gaussian G_y και G_x :
$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

Εφαρμόζουμε τη συνέλιξη με το 1D φίλτρο

Gaussian κατά τον άξονα x :
$$I_x(x, y) = (I * G_x)(x, y) = \sum_{\tau=-\infty}^{\infty} I(x - \tau, y) G_x(\tau)$$

Εφαρμόζουμε τη συνέλιξη με το 1D φίλτρο

Gaussian κατά τον άξονα y :
$$(I_x * G_y)(x, y) = \sum_{\eta=-\infty}^{\infty} I_x(x, y - \eta) G_y(\eta)$$

Οι διακυμάνσεις των 1D φίλτρων είναι σ_x και σ_y τότε η διακύμανση του 2D φίλτρου είναι:

$$\sigma_{2\Delta}^2 = \sigma_x^2 + \sigma_y^2$$

Για συμμετρικά φίλτρα ισχύει $\sigma_x = \sigma_y = \sigma$, η διακύμανση του 2Δ φίλτρου είναι:

$$\sigma_{2\Delta}^2 = 2\sigma^2$$

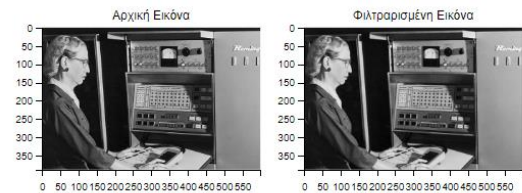
b) Code commitment done

c) Αποτέλεσμα εκτέλεσης του αρχείου “filters.py”

```
def main():
    # There is tolerance for the kernel.

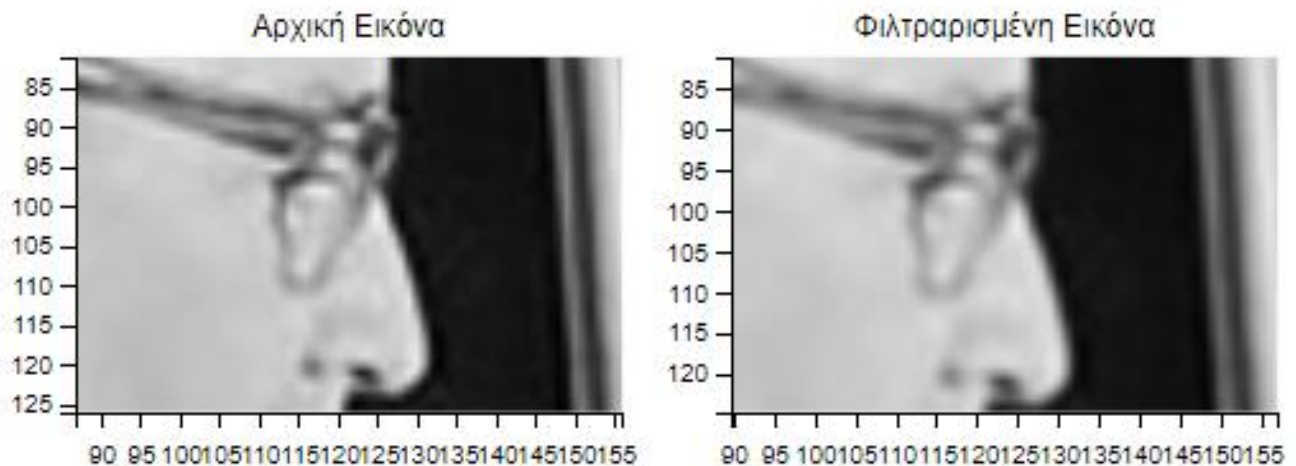
    kernel_size = 3
    sigma = 0.572
    kernel_gaussian = gaussian_kernel(kernel_size, sigma)
    filtered_gaussian = convolve(img, kernel_gaussian)
    save_img(filtered_gaussian, path='./gaussian_filter/q2_gaussian.png')

    # plot the original and filtered images
    plt.figure(figsize=(10, 5))
    plt.subplot(*args: 1, 2, 1)
    plt.title("Αρχική Εικόνα")
    plt.imshow(img, cmap='gray')
    plt.subplot(*args: 1, 2, 2)
    plt.title("Φιλτραρισμένη Εικόνα")
    plt.imshow(filtered_gaussian, cmap='gray')
    plt.show()
```



Το Gaussian φιλτράρισμα εφαρμόζει μία θόλωση στην εικόνα, μειώνοντας τον θόρυβο και τις λεπτομέρειες, οδηγώντας σε μια πιο ομαλή και μαλακή όψη της εικόνας.

Παρατηρώντας πιο προσεκτικά μπορούμε να το διαπιστώσουμε και στην υλοποίησή μας .



d) Είναι καλή ιδέα για ένα φίλτρο εξομάλυνσης να έχει άθροισμα συντελεστών ίσο με 1 για τους εξής λόγους:

- **Διατήρηση της Φωτεινότητας:** Όταν οι συντελεστές του φίλτρου αθροίζουν στο 1, η συνολική φωτεινότητα της εικόνας δεν μεταβάλλεται.
- **Σταθερότητα της Έντασης:** Ένα μη κανονικοποιημένο φίλτρο μπορεί να αυξήσει ή να μειώσει τις εντάσεις της εικόνας, προκαλώντας ανεπιθύμητες αλλαγές στη φωτεινότητα και τον αντίθεση.



Όπως φαίνεται και από τα plot που δημιουργήσαμε

e) Οι παράγωγοι της εικόνας υπολογίζονται ως εξής:–

$$I_x(x,y)=I(x+1,y)-I(x-1,y)\Rightarrow$$

$$I_x(x,y)=(-1 \cdot I(x-1,y))+(0 \cdot I(x,y))+(1 \cdot I(x+1,y))\Rightarrow$$

$$k_x=[-1,0,1]$$

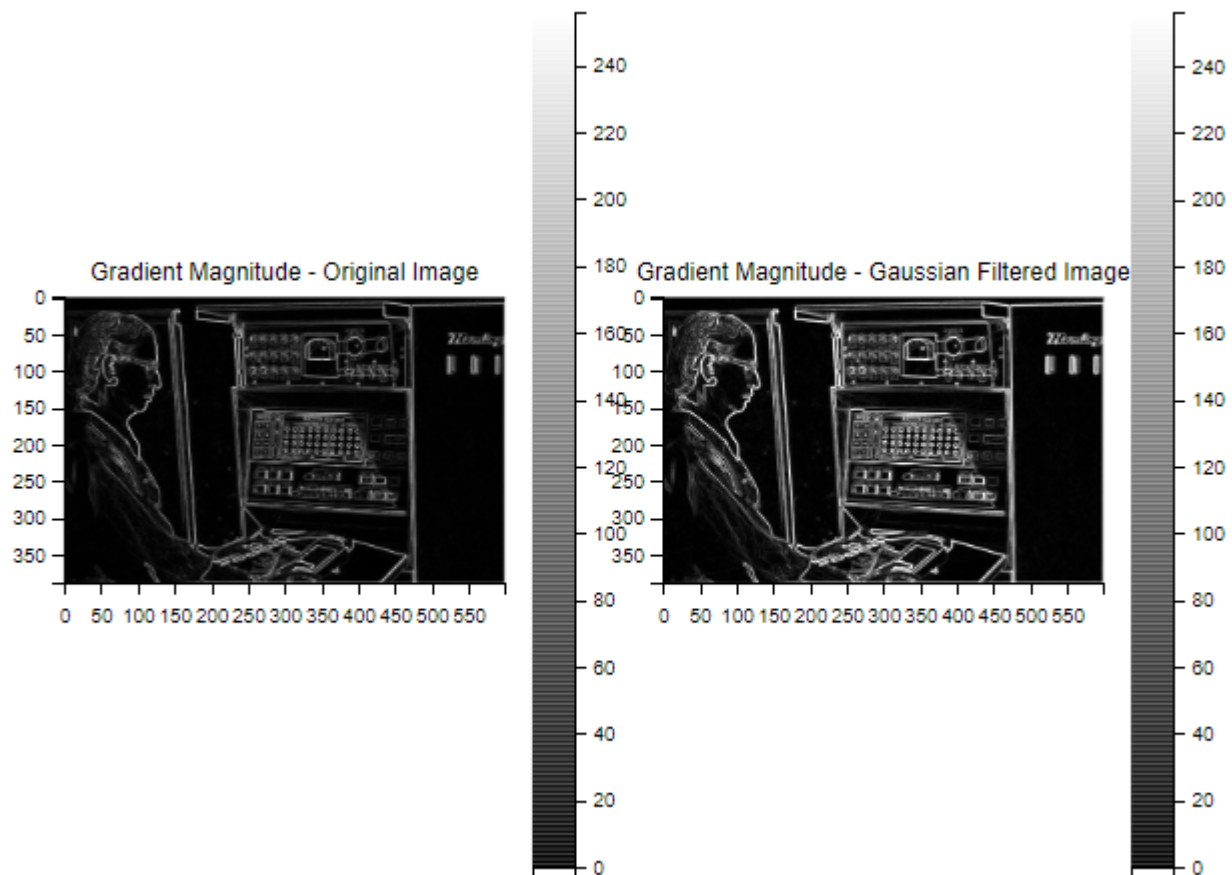
$$I_y(x,y)=I(x,y+1)-I(x,y-1)\Rightarrow$$

$$I_y(x,y)=(-1 \cdot I(x,y-1))+(0 \cdot I(x,y))+(1 \cdot I(x,y+1))\Rightarrow$$

$$K_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

f) Code commitment done

g) **Plots:**



Μετά την εφαρμογή της ρουτίνας `edge_detection()` στις δύο εικόνες, μπορούμε να παρατηρήσουμε τα εξής:

- **Αρχική Εικόνα:** Οι κλίσεις είναι πιο έντονες και οι άκρες είναι πιο ευδιάκριτες. Η εικόνα περιέχει περισσότερο θόρυβο, ο οποίος επηρεάζει τις κλίσεις.
- **Φίλτραρισμένη Εικόνα:** Οι κλίσεις είναι πιο ομαλές και οι άκρες είναι λιγότερο ευδιάκριτες σε σχέση με την αρχική εικόνα.

Ζήτημα 3: Τελεστής Sobel

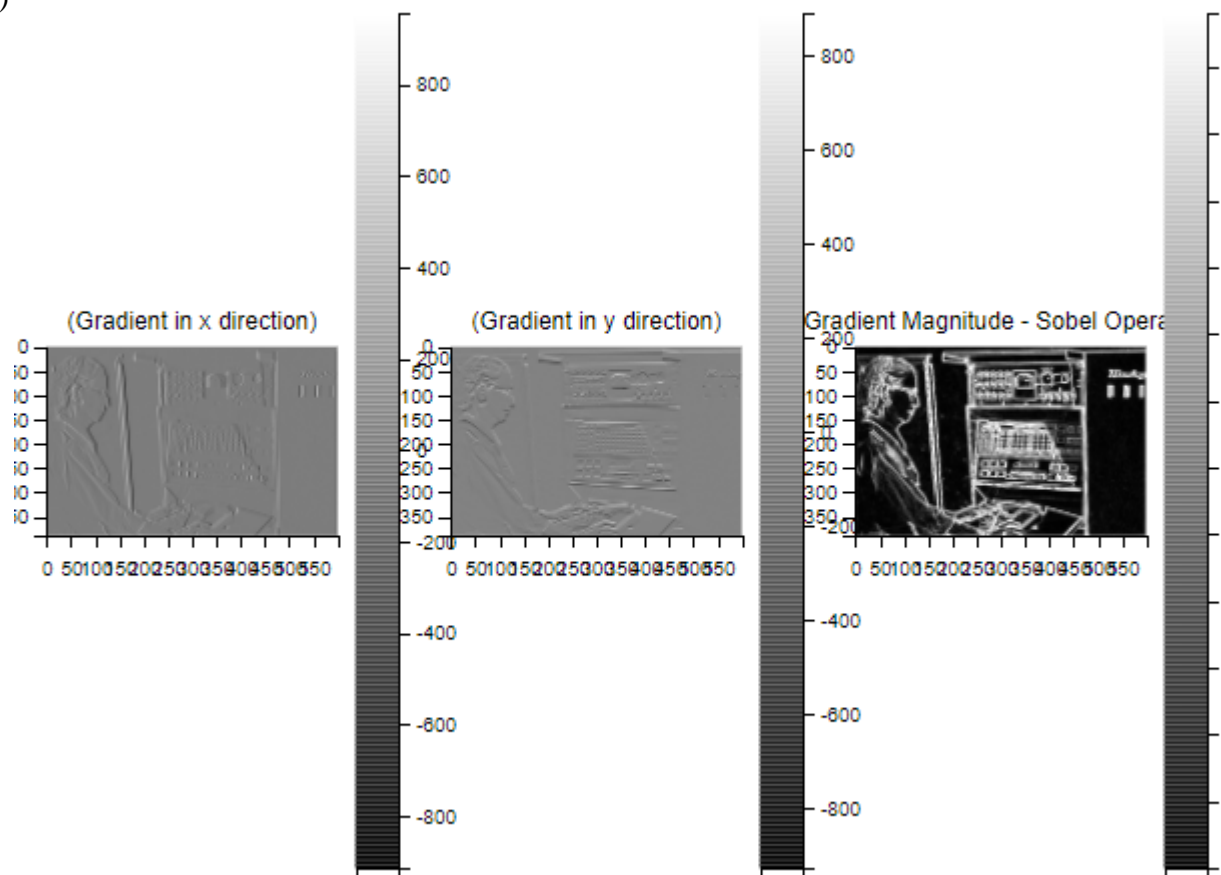
a) Η οριζόντια παράγωγος της φιλτραρισμένης εικόνας I' είναι: $\frac{\partial}{\partial x}(I * G_S) = I * \frac{\partial G_S}{\partial x}$

Η εφαρμογή του φίλτρου Sobel S_x σε μια εικόνα που έχει φιλτραριστεί με το Gaussian φίλτρο G_S προσεγγίζει την οριζόντια παράγωγο της φιλτραρισμένης εικόνας. Δηλαδή: $\frac{\partial}{\partial x}(I * G_S) \approx I * S_x$

Με αυτόν τον τρόπο, αποδεικνύουμε ότι οι τελεστές Sobel μπορούν να χρησιμοποιηθούν για να προσεγγίσουν τις παραγώγους της φιλτραρισμένης εικόνας.

b) Code commitment done

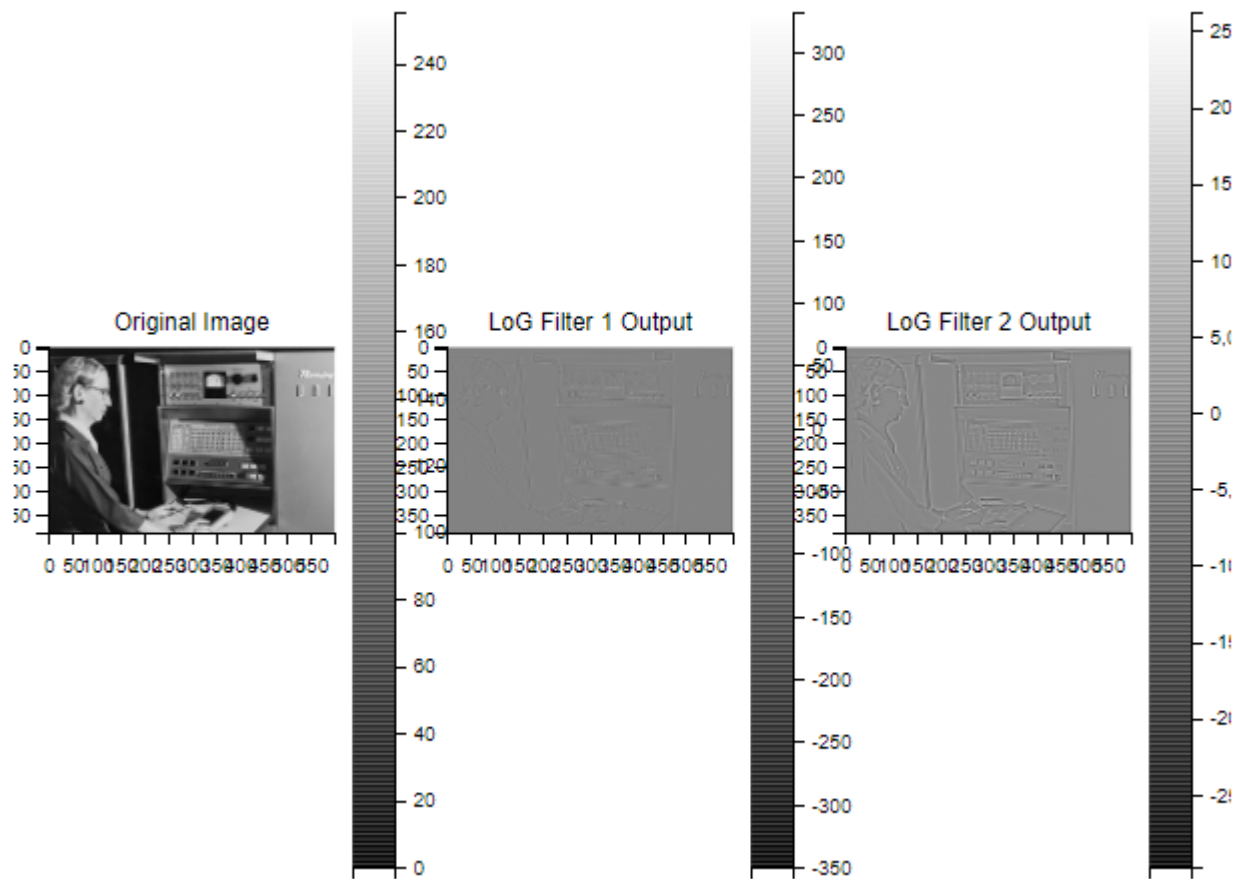
c)



- **$I \cdot S_x$** : Αναδεικνύονται οι οριζόντιες ακμές στην εικόνα, καθώς το φίλτρο Sobel S_x είναι σχεδιασμένο να εντοπίζει αλλαγές στην ένταση κατά την οριζόντια κατεύθυνση.
- **$I \cdot S_y$** : Αναδεικνύονται οι κατακόρυφες ακμές στην εικόνα, καθώς το φίλτρο Sobel S_y είναι σχεδιασμένο να εντοπίζει αλλαγές στην ένταση κατά την κατακόρυφη κατεύθυνση.
- **Gradient Magnitude**: Το μέγεθος της κλίσης αναδεικνύει όλες τις ακμές, συνδυάζοντας τις πληροφορίες από τις παραγώγους κατά x & y .

Ζήτημα 4: Λαπλασιανή της Γκαουσιανής της εικόνας (LoG)

Τα φίλτρα LoG (Laplacian of Gaussian) χρησιμοποιούνται για την ανίχνευση ακμών στην εικόνα.



LoG Filter 1: Είναι μικρότερο σε μέγεθος και μπορεί να ανιχνεύσει λεπτές ακμές και μικρές μεταβολές.

LoG Filter 2: Είναι μεγαλύτερο και είναι πιο κατάλληλο για την ανίχνευση μεγαλύτερων και πιο εκτεταμένων ακμών στην εικόνα

Το Laplacian τονίζει περιοχές της εικόνας όπου υπάρχει απότομη μεταβολή στην ένταση των εικονοστοιχείων. Οι ανιχνευμένες περιοχές είναι τα εικονοστοιχεία όπου το φίλτρο έχει υψηλή απόκριση κατ' απόλυτη τιμή

Αξιοσημείωτες Πηγες Μελέτης

- [Ecourse Ψηφιακή Επεξεργασία Εικόνας](#)
- [OpenGenus IQ: Laplacian Filter](#)
- [Automatic Addison: How the Laplacian of Gaussian Filter Works](#)
- [TheAILearner: Laplacian of Gaussian \(LoG\)](#)
- [University of Edinburgh: Laplacian of Gaussian \(LoG\)](#)
- [StackExchange: Gaussian Blurring](#)
- Scipy Documentation: Convolution and Gaussian Filters
- [AskPython: Gaussian Kernel](#)
- [Delft Stack: Gaussian Kernel](#)
- [Wikipedia: Sobel Operator](#)
- OpenCV Documentation: Sobel
- GeeksforGeeks: Sobel Operator
- TheAILearner: Sobel Edge Detection