ΜΥΕ037 Ψηφιακή Επεξεργασία Εικόνας

Εαρινό Εξάμηνο 2024

Εργασία: 30% του συνολικού βαθμού

Διδάσκων: Άγγελος Γιώτης

ΠΑΡΑΔΟΣΗ: Πέμπτη, 26 Ιουνίου, 2024 23:59

Γενικές Οδηγίες

Σε αυτήν την Εργασία, θα κατεβάσετε από τη σελίδα ecourse του μαθήματος το αρχείο **assignment.zip** και θα αποσυμπιέσετε το φάκελο "**assignment**/" τοπικά στον Η/Υ σας. Θα χρειαστεί να εργαστείτε σε γλώσσα **Python** (έκδοση Python τουλάχιστον 3.7). Για διευκόλυνσή σας, προτείνεται να εγκαταστήσετε το εργαλείο/διαχειριστή πακέτων εικονικού περιβάλλοντος Anaconda για Python 3.7.x. που περιλαμβάνει τις περισσότερες από τις βιβλιοθήκες που χρειάζεστε για την εργασία. Επίσης, μπορείτε να εργαστείτε σε κάποιο χρήσιμο IDE για περιβάλλον Windows/Linux (π.χ. PyCharm**,** η οποία είναι διαθέσιμη με «free educational license», με τον ακαδημαϊκό λογαριασμό σας).

Θα χρησιμοποιήσουμε τα ακόλουθα πακέτα σε αυτή την εργασία:

- Numpy
- SciPy
- Matplotlib

Η εργασία περιλαμβάνει 2 μέρη:

1. Υποβολή του κώδικα Python που θα συμπληρώσετε για τις ρουτίνες image_patches()**,** convolve(), edge_detection(), sobel_operator() και main(), στο αρχείο assignment/**filters.py** καθώς και των φακέλων με τα αποτελέσματα εκτέλεσης κάθε ρουτίνας για κάθε αντίστοιχο ερώτημα/ζήτημα.
2. Απάντηση/**αναφορά** (**report**) στα θεωρητικά ερωτήματα κάθε ζητήματος που σχετίζεται με τις παραπάνω ρουτίνες, σε μορφή PDF. Συγκεντρωτικά όλες οι απαντήσεις σας σε κάθε ζήτημα θα βρίσκονται σε ένα PDF αρχείο: **assignment.pdf.** Μπορείτε να χρησιμοποιήσετε εργαλεία όπως CombinePDF για την συνένωση των παραγόμενων PDF (με τις επιλεγμένες εικόνες και τις απαντήσεις σας σε κάθε ζήτημα) σε ένα αρχείο.

Απαντήστε στα παρακάτω ζητήματα ακολουθώντας τις ακόλουθες οδηγίες:

- Οι ασκήσεις είναι **ατομικές** - δεν επιτρέπεται η μεταξύ σας συνεργασία για την υλοποίηση/παράδοσή τους.
- **Δεν** επιτρέπεται να χρησιμοποιήσετε κώδικα που τυχόν θα βρείτε στο **διαδίκτυο** (είτε αυτούσιο, είτε **παραγόμενο από AI**). Η απευθείας χρήση κώδικα τρίτων θα έχει σαν αποτέλεσμα τον αυτόματο μηδενισμό σας.
- Οι λύσεις σας συνίστανται σε επίπεδο **κώδικα Python** που θα γράψετε μέσα στο αρχείο **filters.py** (το αντίστοιχο ζήτημα επισημαίνεται με **κόκκινο** χρώμα) και σε **αναφορά/απάντηση** σε κάθε ερώτημα καθώς και των επιλογών σας, ως προς τις παραγόμενες εικόνες των ζητημάτων (επισημαίνεται με **μωβ** χρώμα) σε μορφή PDF για κάθε ζήτημα (στο τέλος συνένωση των PDF σε ενιαίο αρχείο).
- Ο κώδικάς σας πρέπει να σχολιαστεί εκτενώς! Καλά σχολιασμένος κώδικας θα συνεκτιμηθεί στην αξιολόγησή σας. Απουσία σχολίων συνιστά κριτήριο μείωσης βαθμού!
- **Συνολικά** θα πρέπει να **παραδώσετε** τα αρχεία "**filters.py"**, τους φακέλους "**image_patches/", "gaussian_filter/, "sobel_operator/", "log_filter",** και το αρχείο/αναφορά **assignment.pdf**, συνολικά σε ένα αρχείο "**project.zip**", καθώς και ένα συνοδευτικό αρχείο "**onoma.txt",** που θα περιέχει το Ον/μο σας και τον Α.Μ. σας. Οι απαντήσεις θα παραδοθούν με την εντολή:

  turnin assignment@mye037 onoma.txt project.zip
- Μπορείτε να χρησιμοποιήσετε βασικά πακέτα γραμμικής άλγεβρας (π.χ. NumPy, Matplotlib, SciPy), αλλά δεν επιτρέπεται να χρησιμοποιείτε τα πακέτα/βιβλιοθήκες που επιλύουν άμεσα τα προβλήματα, εκτός και αν αναφέρεται διαφορετικά η χρήση συγκεκριμένου πακέτου σε κάποιο ζήτημα. Αν δεν είστε βέβαιοι για κάποιο συγκεκριμένο πακέτο/βιβλιοθήκη ή συνάρτηση που θα χρησιμοποιήσετε, μη διστάσετε να ρωτήσετε τον διδάσκοντα.
- Συνιστάται ιδιαίτερα να αρχίσετε να εργάζεστε στις ασκήσεις σας το συντομότερο δυνατό!
- **Late Policy**: Εργασίες που υποβάλλονται καθυστερημένα θα λαμβάνουν μείωση βαθμού 10% για κάθε 24 ώρες καθυστέρησης. Οι εργασίες δεν θα γίνονται δεκτές 96 ώρες (4 ημέρες) μετά την προθεσμία παράδοσης. Για παράδειγμα, παράδοση της εργασίας 2 ημέρες μετά την προθεσμία βαθμολογείται με άριστα το 24 (από 30).

1. Τοπικές περιοχές/Τμήματα εικόνας (Patches)

Ζήτημα 1: Patches [5 μονάδες]

Ένα patch/περιοχή/τμήμα είναι ένα μικρό κομμάτι μιας εικόνας. Πολλές φορές εστιάζουμε σε τοπικές περιοχές μιας εικόνας αντί να λειτουργούμε σε ολόκληρη την εικόνα.

(a) **Συμπληρώστε τη συνάρτηση** **image_patches** στο αρχείο **filters.py**. Η συνάρτηση αυτή θα πρέπει να διαιρεί μια ασπρόμαυρη εικόνα σε ένα σύνολο από μη επικαλυπτόμενα τμήματα εικόνας 16 επί 16 pixel. Κανονικοποιήστε κάθε τμήμα ώστε να έχει μέση τιμή μηδέν και διακύμανση ίση με τη μονάδα**.**
- **Εμφανίστε (plot) και τοποθετήστε στην αναφορά σας** **τρία** τμήματα εικόνας 16x16 από το αρχείο "grace_hopper.png" φορτωμένο σε κλίμακα του γκρι. (2 μονάδες)

(b) **Συζητήστε στην αναφορά σας** γιατί πιστεύετε ότι είναι καλό τα τμήματα να έχουν μηδενική μέση τιμή. (1 μονάδα)

Υπόδειξη: Υποθέστε ότι θέλετε να μετρήσετε την ομοιότητα μεταξύ των τμημάτων υπολογίζοντας τα εσωτερικά γινόμενα μεταξύ διαφορετικών τμημάτων για να βρείτε μια αντιστοιχία. Σκεφτείτε πώς οι τιμές των τμημάτων και η προκύπτουσα ομοιότητα που λαμβάνουμε από τον υπολογισμό των εσωτερικών γινομένων θα επηρεάζονταν υπό διαφορετικές συνθήκες φωτισμού/έντασης φωτεινότητας. Πείτε ότι σε μία περίπτωση, μια τιμή του σκούρου αντιστοιχεί σε 0 ενώ το φωτεινό αντιστοιχεί σε 1. Σε ένα άλλο σενάριο, μια σκοτεινή τιμή αντιστοιχεί σε -1 ενώ το φωτεινό αντιστοιχεί σε 1. Ποιο από τα δύο θα ήταν πιο κατάλληλο για να μετρηθεί η ομοιότητα χρησιμοποιώντας εσωτερικά γινόμενα;

(c) Παλαιότερες εργασίες στην ψηφιακή επεξεργασία εικόνας χρησιμοποιούσαν τμήματα ως περιγραφές/αναπαραστάσεις του τοπικού περιεχομένου της εικόνας για εφαρμογές ευθυγράμμισης, συρραφής εικόνων, ταξινόμησης και εντοπισμού αντικειμένων.

Συζητήστε στην αναφορά σας σε 2-3 προτάσεις, γιατί τα τμήματα από την προηγούμενη ερώτηση (b) θα ήταν καλά ή κακά για προβλήματα όπως η αντιστοίχιση (matching) ή η αναγνώριση ενός αντικειμένου. Εξετάστε πώς θα έμοιαζαν αυτά τα τμήματα αν αλλάζαμε τη θέση, την κλίμακα, την ένταση φωτεινότητας του αντικειμένου κ.λπ. (2 μονάδες).

2. Φιλτράρισμα στο πεδίο του χώρου

Υπόβαθρο: Υπάρχει διαφορά μεταξύ της συνέλιξης (convolution) και της συσχέτισης (cross correlation): στη συσχέτιση, υπολογίζετε το εσωτερικό γινόμενο (δηλαδή, το ζυγισμένο άθροισμα των pixel της εικόνας $\boldsymbol{np.sum(F * I[y1:y2, x1:x2])}$ μεταξύ του πυρήνα/φίλτρου και κάθε

παραθύρου/τμήματος στην εικόνα. Στη συνέλιξη, υπολογίζετε το εσωτερικό γινόμενο μεταξύ του ανεστραμμένου πυρήνα/φίλτρου και κάθε παραθύρου/τμήματος στην εικόνα. Προκειμένου να αποσαφηνίσουμε αυτή τη διάκριση, θα δούμε τι συμβαίνει αρχικά στη μια διάσταση (1Δ). Ας υποθέσουμε ότι η είσοδος/σήμα f έχει N στοιχεία (δηλαδή, δεικτοδοτείται από i για $0 \leq i < N$) και το φίλτρο/πυρήνας g έχει M στοιχεία (δηλαδή, έχει δείκτες j για $0 \leq j < M$). Σε όλες τις παρακάτω περιπτώσεις, μπορείτε να υποθέσετε συμπλήρωση μηδενικών (zero padding) της εισόδου/σήματος f. Η **1-Δ** (cross-correlation) **συσχέτιση** συνήθως δίνεται από τον τύπο:

$$h[i] = \sum_{j=0}^{M-1} g[j]f[i+j] \quad (1)$$

ή για κάθε στοιχείο i, το άθροισμα όλων των γινομένων μεταξύ του φίλτρου στη θέση j και της εισόδου στη θέση $i + j$ για όλες τις έγκυρες τιμές του j. Αν θέλετε να σκεφτείτε αυτή τη διαδικασία με όρους γινομένων πινάκων, μπορείτε να το θεωρήσετε ως $\boldsymbol{h}_i = \boldsymbol{g}^T \boldsymbol{f}_{i:i+M-1}$
Από τις δύο επιλογές, αυτή είναι συνήθως πιο διαισθητική.

Όταν κάνουμε 1Δ συνέλιξη, από την άλλη πλευρά, αναδιατάσσουμε το φίλτρο από το τελευταίο προς το πρώτο στοιχείο και στη συνέχεια κάνουμε το φιλτράρισμα. Στην επεξεργασία σήματος, αυτό συνήθως αναλύεται με κατάλληλες τεχνικές δεικτοδότησης («index manipulation»). Γενικά, η 1Δ συνέλιξη παίρνει τη μορφή:

$$(f * g)[i] = \sum_{j=0}^{M-1} g[M-j-1]f[i+j] \quad (2)$$

η οποία είναι διαφορετική επειδή ξεκινάμε από το 0 αντί του 1. Μπορείτε να επαληθεύσετε ότι καθώς ο δείκτης *j* πηγαίνει από 0 σε (*M - 1*), ο νέος δείκτης (*M - j - 1*) πηγαίνει από (*M - 1*) σε 0.

Στη γενική περίπτωση, αν σας ζητηθεί να εφαρμόσετε συνέλιξη με δοθέν φίλτρο, απλώς μπορείτε να κάνετε τα εξής: (1) στην αρχή της συνάρτησης και μόνο 1 φορά, υπολογίστε το $g = g[::-1]$ αν είναι 1Δ ή αλλιώς το $G = G[::-1, ::-1]$, (2) κάντε το φιλτράρισμα με αυτό το ανεστραμμένο φίλτρο. Ο λόγος για τον οποίο γίνεται όλος αυτός ο κόπος είναι ότι η συνέλιξη είναι αντιμεταθετική ($f * g = g * f$) και «προσεταιριστική» ($f * (g * h) = (f * g) * h$). Η συσχέτιση από την άλλη δεν ικανοποιεί αυτές τις ιδιότητες.
Όλα αυτά θα σας χρειαστούν:

- Στην υλοποίηση της συνέλιξης στο ζήτημα 2(b), στη συνάρτηση *convolve()* του *filters.py*.
- Όταν ασχολούμαστε με μη-συμμετρικά φίλτρα (όπως οι κατευθυντικές παράγωγοι [-1,0,1]). Ένα συμμετρικό φίλτρο όπως το Γκαουσιανό δεν επηρεάζεται από τη διάκριση αυτή επειδή αν το αναστρέψετε οριζόντια/κατακόρυφα, παραμένει το ίδιο. Ωστόσο, για τα ασύμμετρα φίλτρα, μπορείτε να πάρετε διαφορετικά αποτελέσματα. Στην περίπτωση των κατευθυντικών παραγώγων, αυτό αναστρέφει το πρόσημο. Αυτό μπορεί να οδηγήσει σε

εξόδους με αντίθετες τιμές ή να σας δίνει απαντήσεις που είναι σχεδόν σωστές αλλά πρέπει να πολλαπλασιαστούν με -1. Συνοψίζοντας: αν έχετε κάτι που είναι σωστό εκτός από ένα «-1», και χρησιμοποιείτε μια κατευθυντική παράγωγο, τότε το έχετε κάνει με συσχέτιση.

Ζήτημα 2: Συνέλιξη και Γκαουσιανό φίλτρο (Convolution and Gaussian Filter) [16 μονάδες]

Όπως ήδη γνωρίζετε, ένα Γκαουσιανό φίλτρο έχει τιμές που ακολουθούν Gaussian κατανομή πιθανότητας. Συγκεκριμένα, οι τιμές του φίλτρου είναι:

$$1\Delta \text{ πυρήνας: } G(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{x^2}{2\sigma^2}} \qquad 2\Delta \text{ πυρήνας: } G(x,y) = \frac{1}{2\pi\sigma^2}\, e^{-\frac{x^2+y^2}{2\sigma^2}}$$

όπου το 0 είναι το κέντρο του φίλτρου (τόσο στη 1Δ όσο και στη 2Δ, "zero centered") και η σ είναι μια ελεύθερη παράμετρος που ελέγχει το βαθμό της θόλωσης/εξομάλυνσης της εικόνας. Μια χρήσιμη ιδιότητα λοιπόν που κάνει τα πράγματα πιο γρήγορα, έγκειται στο ότι η εφαρμογή ενός 2Δ Gaussian φίλτρου σε μια εικόνα έχει το ίδιο αποτέλεσμα με την διαδοχική εφαρμογή δύο 1Δ Gaussian φίλτρων, το ένα κάθετο και το άλλο οριζόντιο (η σειρά εφαρμογής δεν παίζει ρόλο).

(a) **Να δείξτε στην αναφορά σας ότι** μια συνέλιξη με ένα φίλτρο Gaussian 2Δ ισοδυναμεί με τη διαδοχική εφαρμογή ενός κάθετου και οριζόντιου 1Δ φίλτρου Gauss. **Προσδιορίστε στην αναφορά σας** τη σχέση μεταξύ των διακυμάνσεών (σ^2) των 1Δ και 2Δ φίλτρων. (3 μονάδες)

Συμβουλή: Επιλέξτε ένα συγκεκριμένο μέγεθος φίλτρου k. Ορίστε ένα 2Δ Gaussian φίλτρο $\boldsymbol{G} \in \mathbb{R}^{k\times k}$ και δύο 1Δ Gaussian φίλτρα $\boldsymbol{G}_y \in \mathbb{R}^{k\times 1}$ και $\boldsymbol{G}_x \in \mathbb{R}^{1\times k}$. Ένα χρήσιμο γεγονός που μπορείτε να χρησιμοποιήσετε είναι ότι για οποιοδήποτε k, οποιοδήποτε κατακόρυφο φίλτρο $\boldsymbol{X} \in \mathbb{R}^{k\times 1}$ και οποιοδήποτε οριζόντιο φίλτρο $\boldsymbol{Y} \in \mathbb{R}^{1\times \kappa}$, η συνέλιξη $\boldsymbol{X} * \boldsymbol{Y}$ είναι ίση με το γινόμενο $\boldsymbol{XY}$. Μπορεί να βρείτε ιδιαίτερα χρήσιμο το γεγονός ότι $\boldsymbol{Y} * \boldsymbol{Y}^T = \boldsymbol{Y}\boldsymbol{Y}^T$ και ότι η συνέλιξη είναι συμμετρική (commutative). Εξετάστε τα επιμέρους στοιχεία.

(b) **Ολοκληρώστε τη συνάρτηση** **convolve()** στο αρχείο **filters.py**. Βεβαιωθείτε ότι υλοποιείτε συνέλιξη και όχι φιλτράρισμα συσχέτισης (δηλαδή, αντιστρέψτε τον πυρήνα μόλις τον λάβετε). Για λόγους συνέπειας στα όρια της εικόνας, πρέπει να χρησιμοποιήσετε συμπλήρωση μηδενικών (**zero-padding**) κατά την υλοποίηση της συνέλιξης. (4 μονάδες)

Υπόδειξη: Μπορείτε να χρησιμοποιήσετε τη συνάρτηση **scipy.ndimage.convolve()** **για να ελέγξετε την ορθότητα** της υλοποίησή σας, **όχι** για την απευθείας υλοποίηση του ερωτήματος. Ανατρέξτε στο documentation (αν χρειαστείτε περισσότερες λεπτομέρειες). Για τη συμπλήρωση μηδενικών, χρησιμοποιήστε την παράμετρο **mode='constant'**.

(c) **Εμφανίστε το αποτέλεσμα εκτέλεσης του αρχείου** "filters.py" στο αντίστοιχο ερώτημα που ακολουθεί ("TODO Zitima 2", στη συνάρτηση main) και τοποθετήστε το στην αναφορά σας και στη συνέχεια περιγράψτε τι κάνει το Gaussian φιλτράρισμα στην εικόνα σε μία πρόταση. «**main**»: **Διαβάστε/φορτώστε** την εικόνα "grace_hopper.png" ως είσοδο και **εφαρμόστε ένα φίλτρο** Gaussian που είναι 3x3 με τυπική απόκλιση σ=0.572». (2 μονάδες)

(d) **Εξηγήστε στην αναφορά σας** (ζήτημα 2.d) γιατί είναι μια καλή ιδέα για ένα φίλτρο εξομάλυνσης να έχει άθροισμα συντελεστών ίσο με 1. (2 μονάδες)

Συμβουλή: Ένας τρόπος για να σας βοηθήσει στην ερμηνεία του λόγου αυτού, είναι να παρατηρήστε ότι εάν αθροίσετε όλες τις τιμές του φίλτρου Gaussian στο 2(c), θα πρέπει να πάρετε ένα άθροισμα κοντά στο 1. Μάλιστα, μπορείτε να το κάνετε να αθροιστεί ακριβώς στο 1 διαιρώντας όλες τις τιμές του φίλτρου με το άθροισμά τους. Όταν αυτό το φίλτρο εφαρμόζεται στην εικόνα 'grace_hopper.png', ποιες είναι οι εντάσεις εξόδου της φιλτραρισμένης εικόνας (ελάχιστη, μέγιστη, εύρος); Τώρα σκεφτείτε ένα φίλτρο Gaussian με μέγεθος 3x3 και τυπική απόκλιση σ=2 (αλλά μην το περιορίσετε να αθροίζουν οι συντελεστές στο 1 - απλά χρησιμοποιήστε απευθείας τις τιμές). Υπολογίστε το άθροισμα όλων των τιμών του φίλτρου σε αυτήν την περίπτωση. Τι συμβαίνει με τις εντάσεις της εικόνας εξόδου σε αυτήν την περίπτωση; Εάν προσπαθείτε να αποτυπώσετε (plot) τις εικόνες που προκύπτουν χρησιμοποιώντας το `matplotlib.pyplot` για να συγκρίνετε τη διαφορά, ορίστε το `vmin = 0` και το `vmax = 255` για να παρατηρήσετε έντονα τη διαφορά.

(e) Πολλές φορές αντιμετωπίζουμε την εικόνα ως μια συνάρτηση $I(x, y)$ ή $I: \mathbb{R}^2 \to \mathbb{R}$, κατά την προσπάθειά μας να εντοπίσουμε ακμές (περιοχές απότομης μεταβολής έντασης). Σε τέτοιες περιπτώσεις δίνουμε μεγάλη προσοχή στις παραγώγους. Έστω λοιπόν οι "παράγωγοι":

$$I_x(x, y) = I(x + 1, y) - I(x - 1, y) \approx 2\frac{\partial I}{\partial x}(x, y)$$

$$I_y(x, y) = I(x, y + 1) - I(x, y - 1) \approx 2\frac{\partial I}{\partial x}(x, y)$$

Όπου I_x είναι η (δύο φορές) παράγωγος και επομένως λαμβάνεται με ένα παράγοντα 2, το οποίο δεν επηρεάζει ιδιαίτερα το αποτέλεσμα.

Να υπολογίσετε και να γράψετε στην αναφορά σας (ζήτημα 2.e) τους πυρήνες συνέλιξης για τις παραγώγους: **(i)** $k_x \in \mathbb{R}^{1\times3}$: $I_x = I * k_x$ και **(ii)** $k_y \in \mathbb{R}^{3\times1}$: $I_y = I * k_y$ (2 μονάδες)

(f) Ακολουθήστε τις αναλυτικές οδηγίες στο αρχείο **filters.py** και **ολοκληρώστε τη ρουτίνα edge_detection()**, η οποία έχει ως έξοδο το μέγεθος της κλίσης/παραγώγου της εικόνας

(gradient magnitude). (2 μονάδες)

(g) **Χρησιμοποιήστε την αρχική εικόνα και την εικόνα που έχει φιλτραριστεί** από το Γκαουσιανό φίλτρο ως είσοδο, αντίστοιχα, και χρησιμοποιήστε τη ρουτίνα "edge_detection()" για να λάβετε τα μεγέθη/μέτρα των κλίσεων τους (ως αποτέλεσμα της "main()" στο αρχείο filters.py). **Σχεδιάστε (plot)** και τις **δύο εξόδους** και **τοποθετήστε τις στην αναφορά σας** (για το ζήτημα 2.g). Συζητήστε στην αναφορά σας τη **διαφορά** μεταξύ των δύο εικόνων σε 2-3 προτάσεις. (1 μονάδα)

Ζήτημα 3: Τελεστής Sobel [6 μονάδες]

(a) Παρακάτω σας δίνονται τα φίλτρα Sobel S_x, S_y τα οποία σχετίζονται με ένα Gaussian πυρήνα G_S:

$$S_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \quad S_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}, \quad G_S = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Παρουσιάστε στην αναφορά σας το ακόλουθο αποτέλεσμα: Εάν η εικόνα εισόδου είναι I και χρησιμοποιούμε το G_S ως το Gaussian φίλτρο μας, ο υπολογισμός της οριζόντιας παραγώγου (δηλαδή, $\frac{\partial}{\partial x} I(x, y)$) της φιλτραρισμένης εικόνας μπορεί να προσεγγιστεί από την εφαρμογή του φίλτρου Sobel (δηλαδή, τον υπολογισμό του $I * S_x$). (2 μονάδες)

Συμβουλή: Πρέπει να αξιοποιήσετε τον οριζόντιο πυρήνα k_x που υπολογίσατε στο ζήτημα 2.e.i. Η οριζόντια παράγωγος της εικόνας που έχει διέλθει από το Gaussian φίλτρο είναι: $(I * G_S) * k_x$. Μπορείτε να εκμεταλλευτείτε τις ιδιότητες της συνέλιξης έτσι ώστε να δείξετε ότι τα δύο φίλτρα είναι τα ίδια. Εάν το κάνετε σωστά, μπορείτε να αγνοήσετε εντελώς την εικόνα.

(b) **Ολοκληρώστε την υλοποίηση της συνάρτησης** **sobel_operator()** στο αρχείο **filters.py** με τους πυρήνες που σας δίνονται στο (α). (3 μονάδες)

(c) **Εμφανίστε (plot) τις ακόλουθες ποσότητες** στην αναφορά σας (αποτέλεσμα εκτέλεσης "main()"): $I * S_x$, $I * S_y$, καθώς και το μέτρο της κλίσης της φιλτραρισμένης (με Sobel) εικόνας "grace hopper.png". (1 μονάδα)

Ζήτημα 4: Λαπλασιανή της Γκαουσιανής της εικόνας (LoG) [3 μονάδες]

(a) Στο αρχείο **filters.py**, σας δίνονται δύο φίλτρα LoG (Laplacian of Gaussian). Δεν απαιτείται να δείξετε ότι είναι LoG, καλό είναι όμως να γνωρίζετε πώς φαίνεται ένα φίλτρο LoG. **Συμπεριλάβετε στην αναφορά σας τα ακόλουθα**: **i) τις εξόδους** αυτών των δύο φίλτρων

LoG και **ii) τους λόγους για τους οποίους διαφέρουν**. **Εξηγήστε στην αναφορά σας** εάν αυτά τα φίλτρα μπορούν να ανιχνεύσουν ακμές. Μπορούν να ανιχνεύσουν κάτι άλλο;

Συμβουλή: Με την έννοια ανιχνευμένες περιοχές εννοούμε τα εικονοστοιχεία όπου το φίλτρο έχει υψηλή απόκριση κατ' απόλυτη τιμή.

Οδηγίες υποβολής

Στο αρχείο **project.zip** που θα υποβάλετε μη ξεχάσετε να συμπεριλάβετε τα ακόλουθα:

- **filters.py**
- **image_patches/**: κατάλογος της εικόνας που θα περιέχει τις επιλεγμένες σας 3 περιοχές της αρχικά δοθείσας εικόνας.
- **gaussian_filter/**: κατάλογος με τη φιλτραρισμένη εικόνα και τις αποκρίσεις ανίχνευσης ακμών με το Γκαουσιανό φίλτρο.
- **sobel_operator/**: κατάλογος με τις φιλτραρισμένες εξόδους του ανιχνευτή ακμών Sobel.
- **log_filter/**: κατάλογος με τις αποκρίσεις/εξόδους των τελεστών/φίλτρων LoG.
- **assignment.pdf:** αρχείο (συνένωση PDF) με τις απαντήσεις σας (αναφορά) στα θεωρητικά ερωτήματα και τις προκύπτουσες εικόνες (plots) κάθε ζητήματος.

TURNIN:

turnin assignment@mye037 onoma.txt project.zip