Ψηφιακή Επεξεργασία Εικόνας
-Εργασία 3: Αποκατάσταση Παραμορφωμένης Εικόνας με Φίλτρα Wiener-

Α. Ντελόπουλος

Άνοιξη 2024

1 Εισαγωγή - Φίλτρα Wiener

Έστω δισδιάστατο τυχαίο σήμα $x(n_1, n_2)$ που παραμορφώνεται από γραμμικό χωρικά αναλλοίωτο σύστημα $h(n_1, n_2)$ και στη συνέχεια μολύνεται από θόρυβο $v(n_1, n_2)$, δηλαδή αντί για το αρχικό $x(n_1, n_2)$ έχουμε στα χέρια μας το $y(n_1, n_2)$:

$$\begin{aligned} y(n_1, n_2) &= [h * x](n_1, n_2) + v(n_1, n_2) \\ &= \sum_{k_1} \sum_{k_2} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2) + v(n_1, n_2) \end{aligned} \tag{1}$$

Αν επιχειρήσουμε να αναπαράξουμε το (μη διαθέσιμο) σήμα $x(n_1, n_2)$ εφαρμόζοντας ένα ΓΡΑΧΩΡΑΝ φίλτρο κρουστικής απόκρισης $g(n_1, n_2)$ στο σήμα $y(n_1, n_2)$ λαμβάνοντας ως αποτέλεσμα το

$$\hat{x}(n_1, n_2) = [g * y](n_1, n_2) \tag{2}$$

τότε σύμφωνα με τη θεωρία το βέλτιστο, με την έννοια ότι ελαχιστοποιεί την ισχύ του σφάλματος

$$J = \{[x(n_1, n_2) - \hat{x}(n_1, n_2)]^2\} \tag{3}$$

είναι το $g(n_1, n_2)$ με συνάρτηση μεταφοράς

$$G(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2) S_x(\omega_1, \omega_2)}{\|H(\omega_1, \omega_2)\|^2 S_x(\omega_1, \omega_2) + S_v(\omega_1, \omega_2)} \tag{4}$$

όπου $H(\omega_1, \omega_2)$ είναι η συνάρτηση μεταφοράς του συστήματος παραμόρφωσης $h(n_1, n_2)$ και $S_x(\omega_1, \omega_2)$, $S_v(\omega_1, \omega_2)$ το φάσμα ισχύος του αρχικού σήματος και του θορύβου αντίστοιχα.

Η χρήση της έκφρασης (4) για τον υπολογισμό του βέλτιστου φίλτρου προϋποθέτει γνώση τόσο της συνάρτησης μεταφοράς της παραμόρφωσης ($H(\omega_1, \omega_2)$) όσο και των δύο εμπλεκομένων φασμάτων. Συχνά αφενός εκτιμάμε την $H(\omega_1, \omega_2)$ από τα δεδομένα (βλ. παρακάτω) και αντί για την (4) χρησιμοποιούμε την παρακάτω απλούστευση:

$$\begin{aligned} G(\omega_1, \omega_2) &= \frac{H^*(\omega_1, \omega_2)}{\|H(\omega_1, \omega_2)\|^2 + \frac{S_v(\omega_1, \omega_2)}{S_x(\omega_1, \omega_2)}} \\ &\approx \frac{H^*(\omega_1, \omega_2)}{\|H(\omega_1, \omega_2)\|^2 + 1/K} \end{aligned} \tag{5}$$

και στη συνέχεια επιλέγουμε μέσω δοκιμών την παράμετρο K.

Ένας απλός τρόπος για την εκτίμηση της $h(n_1, n_2)$ ή κατευθείαν της $H(\omega_1, \omega_2)$ είναι η χρήση μιας παραμορφωμένης εικόνας της οποίας γνωρίζουμε την μη παραμορφωμένη εκδοχή. Τυπικό παράδειγμα είναι η εικόνα ενός άστρου για την οποία η μη παραμορφωμένη εκδοχή είναι μια κρουστική συνάρτηση στη θέση του αστεριού.

2 Παραδοτέα

Για τις ανάγκες της εργασίας θα πρέπει να κατασκευάσετε

1. Ένα σύνολο συναρτήσεων σε Python που θα υλοποιούν τη λειτουργία των φίλτρων Wiener
2. Ένα πρόγραμμα επίδειξης με το όνομα `demo.py`
3. Ένα report που θα εξηγείτε τη λειτουργία των συναρτήσεων, θα επιδεικνύετε και θα σχολιάζετε τα αποτελέσματα και θα περιγράφετε πιθανές παραδοχές που κάνατε κατά την υλοποίηση

Διευκρινίζεται ότι μπορείτε να χρησιμοποιήσετε κάθε supported έκδοση της Python (3.8 - 3.12), καθώς και να εισαγάγετε δικές σας επιμέρους βοηθητικές μεθόδους στην υλοποίηση, αρκεί όλα αυτά να καταγράφονται στο report σας.

2.1 `wiener_filtering.py` - Υλοποίηση φίλτρου Wiener

Να κατασκευάσετε συνάρτηση

```
x_hat: np.ndarray = my_wiener_filter(
    y: np.ndarray,
    h: np.ndarray,
    K: float
    )
```

Ορίσματα:

- `y`: ένας δισδιάστατος `numpy array`, που αναπαριστά μια grayscale εικόνα διαστάσεων $M \times N$ *στο πεδίο του χώρου,* που έχει προέλθει σύμφωνα με το μηχανισμό παραμόρφωσης (1)
- `h`: ένας δισδιάστατος `numpy array` (διαστάσεων $L \times P$) με την κρουστική απόκριση του ΓΡΑΧΩΡΑΝ μηχανισμού παραμόρφωσης της (1)
- `K`: αναπαριστά την αντίστοιχη παράμετρο της σχέσης (5)

Έξοδοι:

- `x_hat`: η grayscale εικόνα εξόδου, ως δισδιάστατος `numpy array` διαστάσεων ίδιων με της εισόδου, $M \times N$, που αντιστοιχεί στο ανακατασκευασμένο σήμα $\hat{x}$ της (2).

Υπόδειξη: η υλοποίησή σας θα πραγματοποιηθεί στο πεδίο της χωρικής συχνότητας. Επομένως, δεδομένου του ότι θα χρησιμοποιήσετε Διακριτό Μετασχηματισμό Fourier, θα πρέπει να πραγματοποιήσετε το κατάλληλο zero-padding (όπως απαιτεί η πράξη της συνέλιξης σύμφωνα με τη θεωρία).

2.2 `demo.py`

Κατασκευάστε πρόγραμμα επίδειξης `demo.py` που ξεκινά με τις παρακάτω εντολές, κατά τις οποίες φορτώνει μια δοθείσα grayscale εικόνα, εκτελεί κατάλληλη προεπεξεργασία καταλήγοντας σε ένα `numpy array` τύπου `float`, κανονικοποιημένο στο διάστημα `[0, 1]`, και τελικά κατασκευάζει μια παραμορφωμένη και θορυβώδη εκδοχή αυτής.

```
from scipy.ndimage import convolve
import hw3_helper_utils

# 'x' is the input grayscale image, of type float and normalized to [0, 1]
x = ...(load and preprocess image)...
```

```
# create white noise with level 0.02
v = 0.02 * np.random.randn(*x.shape)
# create motion blur filter
h = hw3_helper_utils.create_motion_blur_filter(length=20, angle=30)
# obtain the filtered image
y0 = convolve(x, h, mode="wrap")
# generate the noisy image
y = y0 + v
```

Το αρχείο με τις βοηθητικές υλοποιήσεις `hw3_helper_utils.py` σας δίνεται μαζί με την εκφώνηση.

Στη συνέχεια του `demo.py`, χρησιμοποιήστε με τον κατάλληλο τρόπο τη συνάρτηση `my_wiener_filter` που κατασκευάσατε, για να ανακατασκευάσετε μια προσέγγιση `x_hat` του `x`. Επιλέξτε το `K` με διαδοχικές δοκιμές. Για λόγους σύγκρισης, υπολογίστε και την `x_inv` ως το αποτέλεσμα του φιλτραρίσματος της `y` με το αντίστροφο φίλτρο $H(\omega_1, \omega_2)^{-1}$, όπως και την `x_inv0` ως το αποτέλεσμα του φιλτραρίσματος της -καθαρής από θόρυβο- `y0`.

Πιο συγκεκριμένα, επιδείξτε τη λειτουργία του αλγορίθμου:

- Για καθεμία από τις εικόνες εισόδου `checkerboard.tif, cameraman.tif` που σας δίνονται
- Για επίπεδα λευκού θορύβου $0.02, 0.2$
- Για μετασχηματισμούς motion blur `(length=10, angle=0), (length=20, angle=30)`

Υπόδειξη: καθώς μια παραμορφωμένη/ενθόρυβη εικόνα ενδέχεται να παίρνει τιμές κι εκτός του εύρους $[0, 1]$, *στα πλαίσια της απεικόνισης και μόνο*, μπορείτε να πλοτάρετε δίχως κανονικοποίηση την αντίστοιχη clipped εικόνα στο επιθυμητό range.

Τέλος, και με βάση την παραπάνω παρατήρηση, μπορείτε να στηριχτείτε στις παρακάτω εντολές για να δείξετε τα αποτελέσματα (ή με όποιον άλλο τρόπο επιθυμείτε):

```
fig, axs = plt.subplots(nrows=2, ncols=3)

axs[0][0].imshow(x, cmap='gray');
axs[0][0].set_title("Original image x")
axs[0][1].imshow(y0, cmap='gray');
axs[0][1].set_title("Clean image y0")
axs[0][2].imshow(y, cmap='gray');
axs[0][2].set_title("Blurred and noisy image y")
axs[1][0].imshow(x_inv0, cmap='gray');
axs[1][0].set_title("Inverse filtering noiseless output x_inv0")
axs[1][1].imshow(x_inv, cmap='gray');
axs[1][1].set_title("Inverse filtering noisy output x_inv")
axs[1][2].imshow(x_hat, cmap='gray');
axs[1][2].set_title("Wiener filtering output x_hat")
```

3 Για την υποβολή της εργασίας

Παραδώστε μία αναφορά με τις περιγραφές και τα συμπεράσματα που σας ζητούνται στην εκφώνηση. Η αναφορά θα πρέπει να επιδεικνύει την ορθή λειτουργία του κώδικά σύμφωνα με τις οδηγίες που σας δίνονται, και να σχολιάζει τα αποτελέσματα.

Στη σύντομη αναφορά σας να συμπεριλάβετε τα σχετικά figures, να περιγράψετε τη μεθοδολογία υλοποίησης της `my_wiener_filter(.)`, αλλά και τη μεθοδολογία υπολογισμού του K και την καμπύλη μεταβολής της J (όπως ορίζεται στη Σχέση 3) ως προς το K. Ειδικότερα, αν K_0 ειναι η τιμή που τελικά επιλέξατε να χρησιμοποιήσετε στο φίλτρο Wiener, παραστήστε τη μεταβολή του J για τις τιμές του K στην περιοχή του K_0.

Επειδή είναι πιθανή μία ολίσθηση της εικόνας εξόδου σε σχέση με την x, στην (3) μπορείτε να αντικαταστήσετε τη x με την x_{inv0}, με την παραδοχή ότι η x_{inv0} είναι πολύ καλό αντίγραφο της x, με την ίδια όμως ολίσθηση όπως και η x_{inv}.

Ο κώδικας θα πρέπει να είναι σχολιασμένος ώστε να είναι κατανοητό τι ακριβώς λειτουργία επιτελεί (σε θεωρητικό επίπεδο, όχι σε επίπεδο κλίσης συναρτήσεων). Επίσης, ο κώδικας θα πρέπει να εκτελείται και να υπολογίζει τα σωστά αποτελέσματα για *οποιαδήποτε* είσοδο πληροί τις υποθέσεις της εκφώνησης, και όχι μόνο για την εικόνα που σας δίνεται.

Απαραίτητες προϋποθέσεις για την βαθμολόγηση της εργασίας σας είναι ο κώδικας να εκτελείται χωρίς σφάλμα (μόνο demos που εκτελούνται επιτυχώς θα βαθμολογηθούν), καθώς και να τηρούνται τα ακόλουθα:

- Υποβάλετε ένα και μόνο αρχείο, τύπου zip.
- Το όνομα του αρχείου πρέπει να είναι `AEM.zip`, όπου `AEM` είναι τα τέσσερα ψηφία του Α.Ε.Μ. του φοιτητή της ομάδας.
- Το προς υποβολή αρχείο πρέπει να περιέχει τα αρχεία κώδικα Python και το αρχείο `report.pdf` το οποίο θα είναι η αναφορά της εργασίας.
- Η αναφορά πρέπει να είναι ένα αρχείο τύπου PDF, και να έχει όνομα `report.pdf`.
- Όλα τα αρχεία κώδικα πρέπει να είναι αρχεία κειμένου τύπου UTF-8, και να έχουν κατάληξη `py`.
- Το αρχείο τύπου zip που θα υποβάλετε δεν πρέπει να περιέχει κανέναν φάκελο.
- Μην υποβάλετε τις εικόνες που σας δίνονται για πειραματισμό.
- Μην υποβάλετε αρχεία που δεν χρειάζονται για την λειτουργία του κώδικά σας, ή φακέλους/αρχεία που δημιουργεί το λειτουργικό σας, πχ “Thumbs.db”, “.DS_Store”, “.directory”.
- Για την ονομασία των αρχείων που περιέχονται στο προς υποβολή αρχείο, χρησιμοποιείτε μόνο αγγλικούς χαρακτήρες, και όχι ελληνικούς ή άλλα σύμβολα, πχ “#”, “$”, “%” κλπ.