

ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ

Εργασία 3: Αποκατάσταση παραμορφωμένης εικόνας με φίλτρα Wiener

Μαυρογεωργιάδη Χριστίνα 10439

Ιούλιος 2024

1 Εισαγωγή

Έστω ένα δισδιάστατο σήμα $x(n_1, n_2)$ το οποίο παραμορφώνεται από μια συνάρτηση υποβάθμισης $h(n_1, n_2)$. Η συνάρτηση h είναι η χρουστική απόχριση ενός γραμμικού χωρικά αναλλοιώτου συστήματος. Στη συνέχεια, η έξοδος του συστήματος υποβαθμίζεται από προσθετικό θόρυβο $v(n_1, n_2)$.

$$y(n_1, n_2) = [h \star x](n_1, n_2) + v(n_1, n_2) \quad (1)$$

Θεωρούμε ότι τόσο το σήμα x , όσο και ο θόρυβος v είναι στάσιμες δισδιάστατες στοχαστικές διαδικασίες, οι οποίες είναι ασυχέτιστες μεταξύ τους. Στόχος μας είναι να υπολογίσουμε μια εκτίμηση του αρχικού άγνωστου σήματος $x(n_1, n_2)$ βασιζόμενοι στη μέτρηση του αλλοιωμένου σήματος $y(n_1, n_2)$. Θεωρούμε ότι η εκτίμηση $\hat{x}(n_1, n_2)$ είναι η έξοδος ενός γραμμικού χωρικά αναλλοιώτου φίλτρου $g(n-1, n_2)$.

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

Η εκτίμηση ενός βέλτιστου φίλτρου βασίζεται στην ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος:

$$J = \mathbb{E}\{(x(n_1, n_2) - \hat{x}(n_1, n_2))^2\} \quad (3)$$

Αποδεικνύεται ότι η συνάρτηση μεταφοράς του βέλτιστου φίλτρου δίνεται από:

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

όπου $H(\omega_1, \omega_2)$ είναι η συνάρτηση μεταφοράς του συστήματος παραμόρφωσης και $S_v(\omega_1, \omega_2), S_x(\omega_1, \omega_2)$ είναι το φάσμα ισχύος του θορύβου και του σήματος x αντίστοιχα. Η χρήση του παραπάνω φίλτρου προϋποθέτει τη γνώση τόσο της συνάρτησης μεταφοράς της παραμόρφωσης όσο και των δύο εμπλεκώμενων φασμάτων, τα οποία συνήθως δεν είναι γνωστά. Η συνάρτηση μεταφοράς του συστήματος παραμόρφωσης μπορεί να εκτιμηθεί χρησιμοποιώντας κάποια μέθοδο μοντελοποίησης συστήματος. Από την άλλη μεριά, ο λόγος $\frac{S_x(\omega_1, \omega_2)}{S_v(\omega_1, \omega_2)}$ συνήθως προσεγγίζεται από μια σταθερά K , η οποία επιλέγεται μέσω δοκιμών και προσαρμόζεται ανάλογα με το SNR. Συνεπώς, η συνάρτηση μεταφοράς του φίλτρου αποκατάστασης γίνεται:

$$G(\omega_1, \omega_2) \approx \frac{H^*(\omega_1, \omega_2)}{\|H(\omega_1, \omega_2)\|^2 + \frac{1}{K}} \quad (5)$$

Στόχος της εργασίας είναι η υλοποίηση ενός φίλτρου Wiener μέσω της γλώσσας προγραμματισμού Python και η επίδειξη των αποτελεσμάτων της επεξεργασίας πάνω σε παραμορφωμένης εικόνες εισόδου. Η υλοποίηση πραγματοποιείται με την έκδοση της Python 3.11.

2 Υλοποίηση φίλτρου Wiener

Η υλοποίηση του φίλτρου γίνεται στο αρχείο `wiener_filtering.py`. Στο αρχείο αυτό ορίζεται η παρακάτω συνάρτηση:

```
def my_wiener_filter(y: np.ndarray, h: np.ndarray, k: float) -> np.ndarray:
```

Σκοπός αυτής της συνάρτησης είναι να υπολογίσει την εκτίμηση μιας αρχικής άγνωστης εικόνας $x(n_1, n_2)$, εφαρμόζοντας το φίλτρο Wiener πάνω στην υποβαθμισμένη εικόνα $y(n_1, n_2)$, η οποία ορίζεται ως η έξοδος του συστήματος της εξίσωσης (1).

Έστω ότι η άγνωστη εικόνα έχει διαστάσεις $M \times N$ και το φίλτρο παραμόρφωσης $h(n_1, n_2)$ έχει διαστάσεις $L \times P$. Η υλοποίηση του φίλτρου πραγματοποιείται στο πεδίο της χωρικής συχνότητας. Επομένως, το φίλτρο παραμόρφωσης $h(n_1, n_2)$ πρέπει να προσαρμοστεί στο μέγεθος της εικόνας μέσω κατάλληλου zero-padding. Συγκεκριμένα, ο πίνακας h επεκτείνεται με μηδενικά ώστε να αποκτήσει διαστάσεις $M \times N$. Οι τιμές του φίλτρου $L \times P$ βρίσκονται στο άνω αριστερό άκρο του τελικού πίνακα.

```
h_padded = np.zeros((m, n))
h_padded[:l, :p] = h
```

Στη συνέχεια, υπολογίζεται ο Διακριτός Μετασχηματισμός Fourier της υποβαθμισμένης εικόνας, $Y(\omega_1, \omega_2)$, και του φίλτρου παραμόρφωσης, $H(\omega_1, \omega_2)$. Οι μετασχηματισμοί έχουν τις ίδιες διαστάσεις $M \times N$.

```
y_dft = np.fft.fft2(y)
h_dft = np.fft.fft2(h_padded)
```

Για τον υπολογισμό του Διακριτού Μετασχηματισμού Fourier χρησιμοποιείται η συνάρτηση `fft2()` του `numpy.fft module` της Python, η οποία υλοποιεί τον μετασχηματισμό μέσω του αλγορίθμου Fast Fourier Transform. Υπολογίζονται ο συζυγής του μετασχηματισμού $H(\omega_1, \omega_2)$, $H^*(\omega_1, \omega_2)$, καθώς και το τετράγωνο του μέτρου του, $\|H(\omega_1, \omega_2)\|^2$, όπως αυτά απαιτούνται για την υλοποίηση του φίλτρου Wiener στην εξίσωση (5). Όλοι οι υπολογισμοί πραγματοποιούνται στοιχείο-προς-στοιχείο.

```
h_dft_conj = np.conjugate(h_dft)
h_dft_magn2 = np.absolute(h_dft)**2
```

Η εξίσωση του φίλτρου Wiener (ορίζεται στην εξίσωση (5)) υλοποιείται από την παρακάτω εντολή:

```
wiener_filter = np.divide(h_dft_conj, (h_dft_magn2 + 1/k))
```

Ο μετασχηματισμός της υποβαθμισμένης εικόνας $Y(\omega_1, \omega_2)$ πολλαπλασιάζεται με το φίλτρο Wiener $G(\omega_1, \omega_2)$ στο πεδίο της χωρικής συχνότητας

$$\hat{X}(\omega_1, \omega_2) = Y(\omega_1, \omega_2) \cdot G(\omega_1, \omega_2)$$

και προκύπτει η εκτίμηση της άγνωστης εικόνας στο πεδίο της συχνότητας.

```
x_hat_dft = np.multiply(wiener_filter, y_dft)
```

Τέλος, για να εξάγουμε την εκτίμηση της άγνωστης εικόνας στο πεδίο του χώρου εφαρμόζουμε τον Αντίστροφο Διακριτό Μετασχηματισμό Fourier στο σήμα $\hat{X}(\omega_1, \omega_2)$:

```
x_hat = np.fft.ifft2(x_hat_dft)
```

μέσω της συνάρτησης `ifft2()` του `numpy.fft module`.

Με αυτόν τον τρόπο, προκύπτει η εκτίμηση στο πεδίο του χώρου $\hat{x}(n_1, n_2)$. Η εκτίμηση αυτή θα πρέπει θεωρητικά να περιλαμβάνει πραγματικές τιμές. Ωστόσο, λόγω υπολογιστικών ανακριβειών, οι τιμές του πίνακα `x_hat` μπορεί να περιλαμβάνουν ένα μικρό φανταστικό μέρος. Για το λόγο αυτό, για να λάβουμε την τελική εκτίμηση πρέπει να εξάγουμε το πραγματικό μέρος του πίνακα `x_hat`:

```
x_hat = np.real(x_hat)
```

Η συνάρτηση επιστρέφει την τελική εκτίμηση `x_hat`.

3 Βοηθητικές συναρτήσεις

Ορίζονται δύο βοηθητικές συναρτήσεις:

- μια συνάρτηση υπολογισμού της βέλτιστης τιμής της παραμέτρου K του φίλτρου Wiener
- μια συνάρτηση υλοποίησης του αντίστοφου φίλτρου $H^{-1}(\omega_1, \omega_2)$

3.1 Υλοποίηση αντίστροφου φίλτρου

Το αντίστροφο φίλτρο είναι η αντίστροφη τιμή του μετασχηματισμού Fourier του φίλτρου παραμόρφωσης, $H^{-1}(\omega_1, \omega_2)$. Η έξοδος του φίλτρου είναι:

$$\hat{X}(\omega_1, \omega_2) = H^{-1}(\omega_1, \omega_2) \cdot Y(\omega_1, \omega_2)$$

όπου $Y(\omega_1, \omega_2)$ είναι ο Διακριτός Μετασχηματισμός Fourier της υποβαθμισμένης εικόνας και δίνεται από:

$$Y(\omega_1, \omega_2) = H(\omega_1, \omega_2) \cdot X(\omega_1, \omega_2) + N(\omega_1, \omega_2)$$

και προκύπτει από την εφαρμογή του μετασχηματισμού Fourier στην εξίσωση (1).

Συνεπώς, η εκτίμηση της άγνωστης εικόνας είναι:

$$\hat{X}(\omega_1, \omega_2) = X(\omega_1, \omega_2) + \frac{N(\omega_1, \omega_2)}{H(\omega_1, \omega_2)}$$

Η απλή αυτή προσέγγιση για την ανάκτηση της αρχικής εικόνας δεν λαμβάνει υπόψιν τα πιθανοτικά χαρακτηριστικά του θορύβου και της εικόνας εισόδου που είναι πιθανόν να γνωρίζουμε, όπως το φάσμα ισχύος. Επιπλέον, είναι πιθανόν να οδηγήσει στο αντίθετο από το επιλυμητό αποτελέσμα, ενισχύοντας τον θόρυβο. Αυτό συμβαίνει σε περίπτωση που υπάρχουν πολύ μικρές τιμές στην συνάρτηση μεταφοράς $H(\omega_1, \omega_2)$. Τότε, η συνεισφορά του σήματος θορύβου $N(\omega_1, \omega_2)$ στην τελική εκτίμηση υπερισχύει σημαντικά τον αρχικού σήματος $X(\omega_1, \omega_2)$ που επιλυμούμε να ανακτήσουμε.

Η υλοποίηση του αντίστροφου φίλτρου γίνεται στο αρχείο `inverse_filtering.py`. Στο αρχείο αυτό ορίζεται η παρακάτω συνάρτηση:

```
def my_inverse_filter(y: np.ndarray, h: np.ndarray) -> np.ndarray:
```

Σκοπός αυτής της συνάρτησης είναι να υπολογίσει την εκτίμηση της αρχικής άγνωστης εικόνας $x(n_1, n_2)$, εφαρμόζοντας το αντίστροφο φίλτρο πάνω στην υποβαθμισμένη εικόνα $y(n_1, n_2)$, η οποία ορίζεται ως η έξοδος του συστήματος της εξίσωσης (1).

Η υλοποίηση της συνάρτησης ακολουθεί τα ίδια ακριβώς βήματα με την συνάρτηση `my_wiener_filter()` που περιγράφηκαν στην ενότητα 2, με μόνη διαφορά στον ορισμό του φίλτρου που εφαρμόζεται στην εικόνα εισόδου $y(n_1, n_2)$. Ειδικότερα, το αντίστροφο φίλτρο δίνεται από:

```
epsilon = 1e-10
h_dft_inv = 1/(h_dft + epsilon)
```

Κατά την αντιστροφή των τιμών του φίλτρου, κάποιες τιμές είναι μηδενικές ή πολύ κοντά στο μηδέν. Αυτό οδηγεί στην εμφάνιση run-time error λόγω της διαίρεσης με το μηδέν. Για να διαχειριστούμε αυτή την κατάσταση, προσθέτουμε στον μετασχηματισμό $H(\omega_1, \omega_2)$ μια πολύ μικρή τιμή `epsilon` η οποία δεν επηρεάζει πρακτικά την συνάρτηση μεταφοράς $H(\omega_1, \omega_2)$ αλλά ταυτόχρονα επιλύει το πρόβλημα της διαίρεσης με το μηδέν.

3.2 Υπολογισμός βέλτιστης τιμής παραμέτρου K

Η παράμετρος K του φίλτρου Wiener είναι ίση με το λόγο του φάσματος ισχύος της αρχικής εικόνας $x(n_1, n_2)$ προς το φάσμα ισχύος του θορύβου $v(n_1, n_2)$.

$$K = \frac{S_x(\omega_1, \omega_2)}{S_v(\omega_1, \omega_2)}$$

και συνεπώς εξαρτάται από την χωρική συχνότητα. Θεωρώντας ότι το φάσμα ισχύος της αρχικής εικόνας και του θορύβου είναι σταθερά και ανεξάρτητα της συχνότητας, μπορούμε να προσεγγίσουμε την παράμετρο K από το λόγο της ισχύος της εικόνας προς την ισχύ του θορύβου (Signal-to-Noise-Ratio):

$$K = \frac{P_x}{P_v}$$

αν γνωρίζουμε τα P_x, P_v .

Η προσέγγιση που ακολουθείται για τον υπολογισμό του K είναι να ορίσουμε ένα σύνολο τιμών της παραμέτρου γύρω από την τιμή του SNR και να εφαρμόσουμε το φίλτρο στην υποβαθμισμένη εικόνα για όλες τις τιμές του συνόλου. Από αυτές τις τιμές θα επιλεγεί η τιμή η οποία ελαχιστοποιεί το Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error - MSE), το οποίο δίνεται από την εξίσωση (3).

Είναι σημαντικό να παρατηρήσουμε ότι είναι πιθανή μια ολίσθηση της εικόνας εξόδου $\hat{x}(n_1, n_2)$ σε σχέση με την

αρχική εικόνα $x(n_1, n_2)$. Για να διαχειριστούμε το πρόβλημα που μπορεί να δημιουργηθεί στον υπολογισμό του σφάλματος, αντικαθιστούμε στη σχέση (3) το σήμα $\hat{x}(n_1, n_2)$ με το σήμα $x_inv0(n_1, n_2)$, το οποίο είναι η έξοδος του αντίστροφου φίλτρου με είσοδο την παραμορφωμένη εικόνα (χωρίς προσθετικό θόρυβο). Η εικόνα $x_inv0(n_1, n_2)$ υφίσταται την ίδια ολίσθηση με την εικόνα $\hat{x}(n_1, n_2)$, επομένως η ανεπιθύμητη επίδραση της ολίσθησης στην τιμή του σφάλματος αναιρείται.

Η υλοποίηση του υπολογισμού της παραμέτρου K γίνεται στο αρχείο `optimal_wiener_parameter.py`. Στο αρχείο αυτό ορίζεται η παραχώτω συνάρτηση:

```
def calculate_k(x: np.ndarray, y: np.ndarray, h: np.ndarray, noise_var: float) -> float:
```

Σκοπός της συνάρτησης είναι να υπολογίσει την βέλτιστη τιμή της παραμέτρου K για δεδομένη την ισχύ της εικόνας εισόδου και την ισχύ του θορύβου, όπως περιγράφεται από την παραπάνω μεθοδολογία. Η συνάρτηση υπολογίζει την τιμή του SNR:

```
k0 = power_x / noise_var
```

και στη συνέχεια, ορίζει ένα παράθυρο γύρω από αυτό. Το εύρος του παραθύρου καθώς και η αρχή και το τέλος του καθορίζονται από κάποιες παραμέτρους μέσα στη συνάρτηση (`num`, `a`, `b` αντίστοιχα), οι οποίες μπορούν να αλλαχθούν ανάλογα με τις απαιτήσεις του προβλήματος.

```
num = 1000
```

```
a = 9
```

```
b = 3
```

```
k_values = np.linspace(k0/a, b*k0, num)
```

Οι τιμές των παραμέτρων επιλέχθηκαν μετά από δοκιμές με βάση το απαιτούμενο διάστημα τιμών K που παρήγαγε ένα τοπικό ελάχιστο για τις δεδομένες εικόνες εισόδου.

Μια απλούστερη προσέγγιση αν δεν γνωρίζουμε το SNR θα ήταν απλά να παράγουμε διάφορα διαστήματα τιμών του K και μετά από δοκιμές να επιλέξουμε το κατάλληλο, ωστόσο θα απαιτούσε περισσότερους υπολογισμούς για την εύρεση του κατάλληλου παραθύρου.

Στη συνέχεια, υπολογίζουμε την τιμή του Μέσου Τετραγωνικού Σφάλματος για κάθε μια από τις τιμές της παραμέτρου K που ορίζονται στον πίνακα `k_values`:

```
for i in range(num):
    k = k_values[i]
    x_hat = wiener_filtering.my_wiener_filter(y, h, k)
    mse[i] = np.mean((x_inv0 - x_hat)**2)
```

Υπολογίζουμε τον δείκτη του πίνακα σφάλματος `mse` ο οποίος αντιστοιχεί στην ελάχιστη τιμή του και ανακτούμε την αντίστοιχη βέλτιστη τιμή του K από τον πίνακα `k_values`.

```
idx_opt = np.argmin(mse)
k_opt = k_values[idx_opt]
```

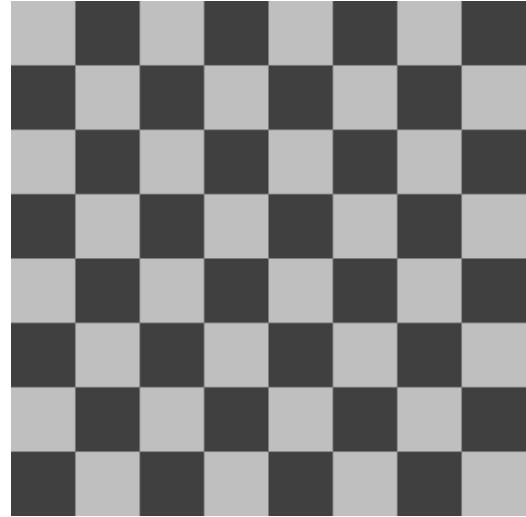
Τέλος, η συνάρτηση εμφανίζει το Μέσο Τετραγωνικό Σφάλμα σε συνάρτηση με την τιμή του K . Πάνω στην καμπύλη τονίζεται το σημείο ελαχιστοποίησής της.

4 Επίδειξη αποτελεσμάτων

Για την επίδειξη της λειτουργίας των δύο φίλτρων αποκατάστασης που συζητήθηκαν παραπάνω, δηλαδή το αντίστροφο φίλτρο και το φίλτρο Wiener, χρησιμοποιούνται οι δύο παραχώτω εικόνες:



(α') Input Image 1



(β') Input Image 2

Σχήμα 1: Original Images

Οι εικόνες αυτές διέρχονται από το μοντέλο της εξίσωσης (1), οπότε προκύπτει μια υποβαθμισμένη εικόνα, η οποία έχει υποστεί παραμόρφωση και έχει αλλοιωθεί από προσθετικό ύδρυβο.

Για τις ανάγκες της εργασίας, το φίλτρο παραμόρφωσης ορίζεται στο αρχείο `hw3_helper_utils.py` και επιστρέφεται από τη συνάρτηση `create_motion_blur_filter()`. Συγκεκριμένα, η συνάρτηση αυτή δημιουργεί ένα φίλτρο που προσομοιάζει την επίδραση της θόλωσης λόγω κίνησης σε μια εικόνα. Η συνάρτηση δέχεται ως είσοδο δύο παραμέτρους `length`, `angle`.

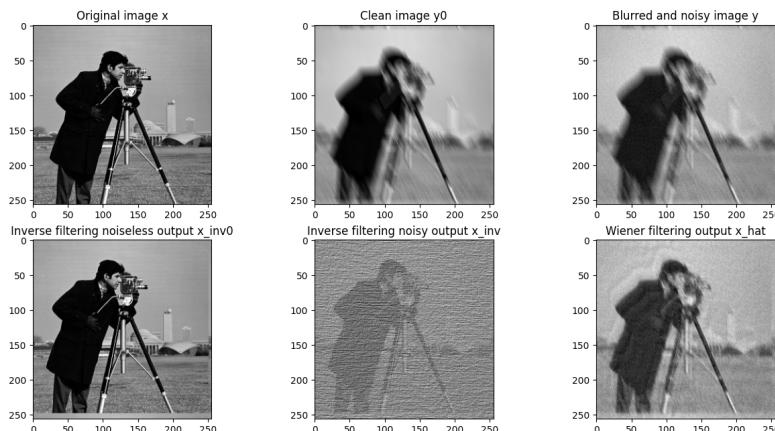
Αφού εφαρμοστεί το φίλτρο παραμόρφωσης, στην κάθε εικόνα εξόδου προστίθεται θόρυβος, ο οποίος ακολουθεί Gaussian κατανομή με μέση τιμή 0 και τυπική απόκλιση που προσδιορίζεται από την μεταβλητή `noise_level`. Τέλος, οι υποβαθμισμένες εικόνες διέρχονται από τα φίλτρα αποκατάστασης. Η παράμετρος K του φίλτρου Wiener ορίζεται ως η βέλτιστη τιμή που υπολογίζεται από τη συνάρτηση `calculate_k()` με βάση το επίπεδο του θορύβου και την ισχύ των εικόνων εισόδου.

Σημειώνεται ότι το σήμα θορύβου που παράγεται σε κάθε εκτέλεση του προγράμματος `demo.py` είναι το ίδιο έτσι ώστε τα αποτελέσματα να είναι αναπαράξιμα. Δηλαδή, η γεννήτρια τυχαίων αριθμών είναι ντετερμινιστική.

Το πρόγραμμα επίδειξης εκτελείται για τους παρακάτω συνδυασμούς παραμέτρων του φίλτρου παραμόρφωσης και του προσθετικού θορύβου:

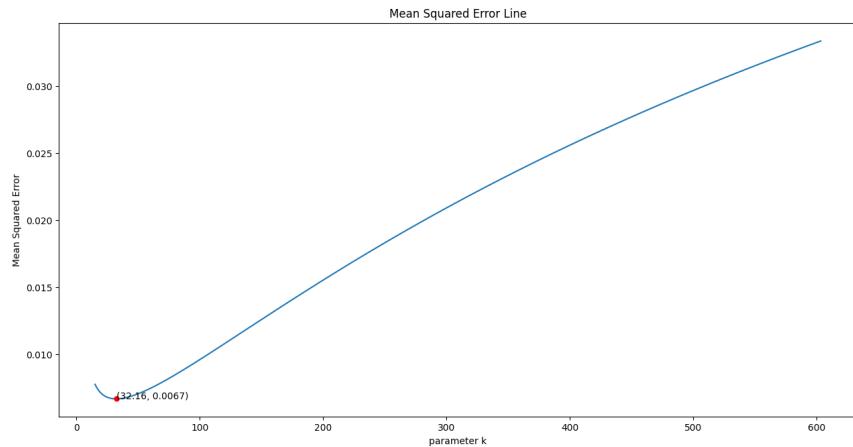
- (`length = 10, angle = 0`) και (`length = 20, angle = 30`)
- `noise_level = 0.02` και `noise_level = 0.2`

Τα αποτελέσματα της επεξεργασίας φαίνονται παρακάτω.

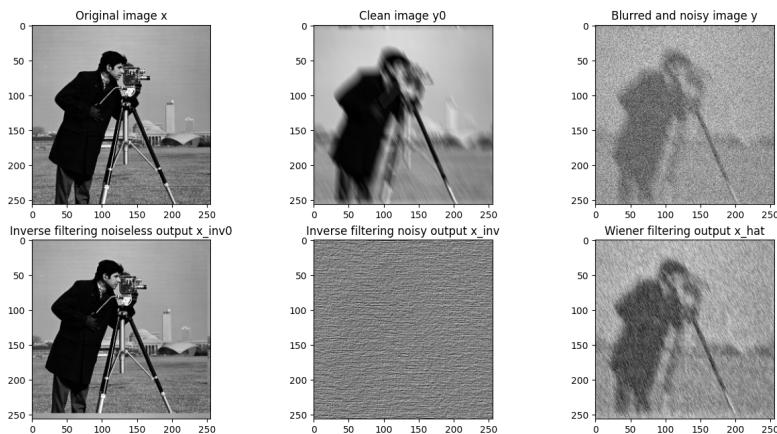


Σχήμα 2: Noise level = 0.02, length = 20, angle = 30

Signal-to-Noise-Ratio: 150.82965103889404
Optimal value of Wiener filter k parameter: 32.15887454583026

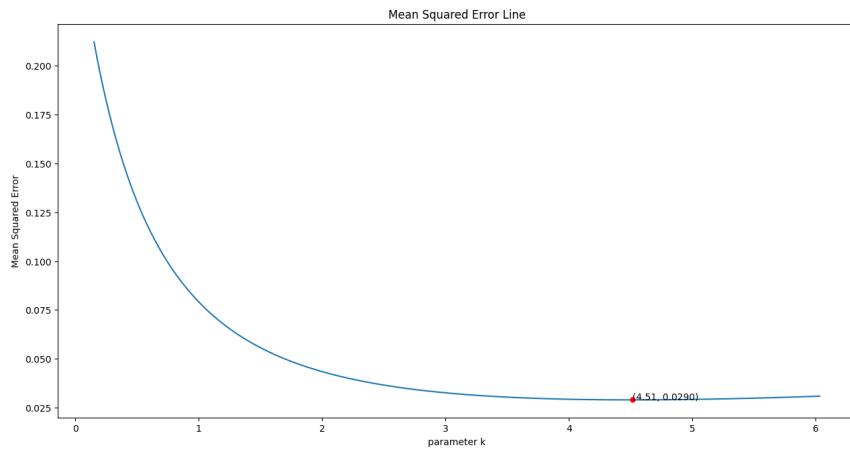


$\Sigma\chi\eta\mu\alpha$ 3: Noise level = 0.02, length = 20, angle = 30

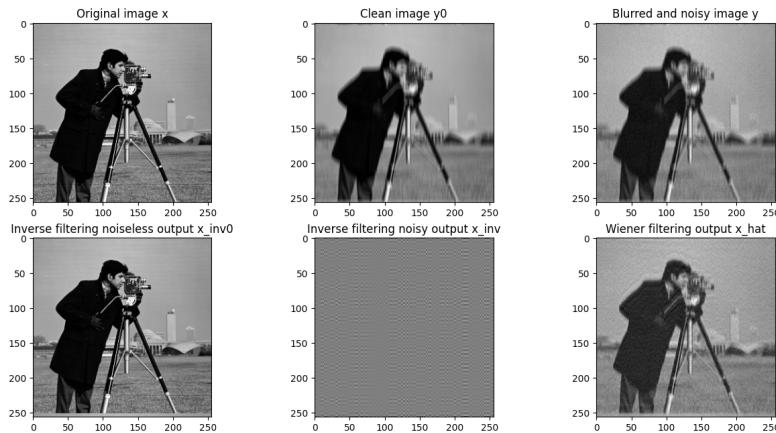


$\Sigma\chi\eta\mu\alpha$ 4: Noise level = 0.2, length = 20, angle = 30

Signal-to-Noise-Ratio: 1.5082965103889403
Optimal value of Wiener filter k parameter: 4.51401892568654

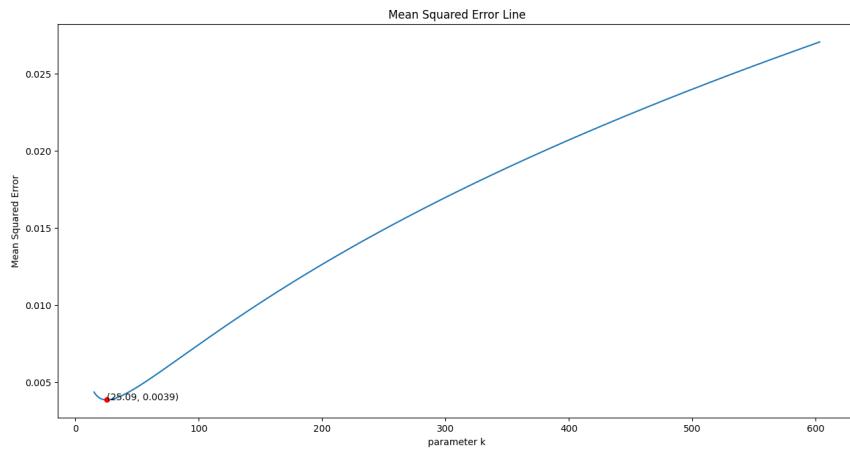


$\Sigma\chi\eta\mu\alpha$ 5: Noise level = 0.2, length = 20, angle = 30

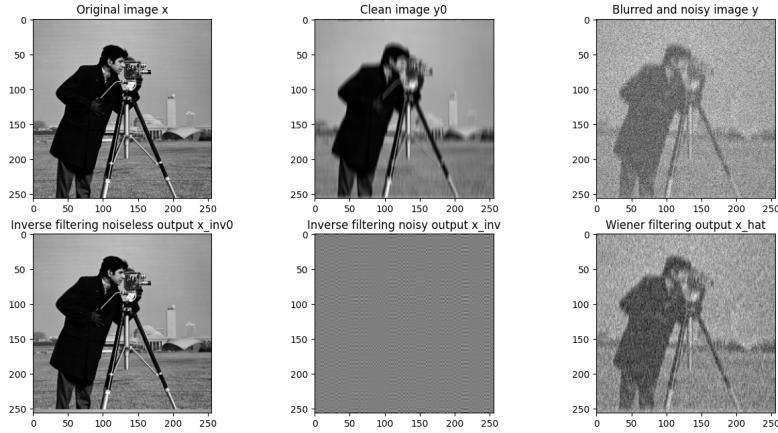


$\Sigma\chi\eta\mu\alpha$ 6: Noise level = 0.02, length = 10, angle = 0

Signal-to-Noise-Ratio: 150.82965103889404
 Optimal value of Wiener filter k parameter: 25.092980983647838



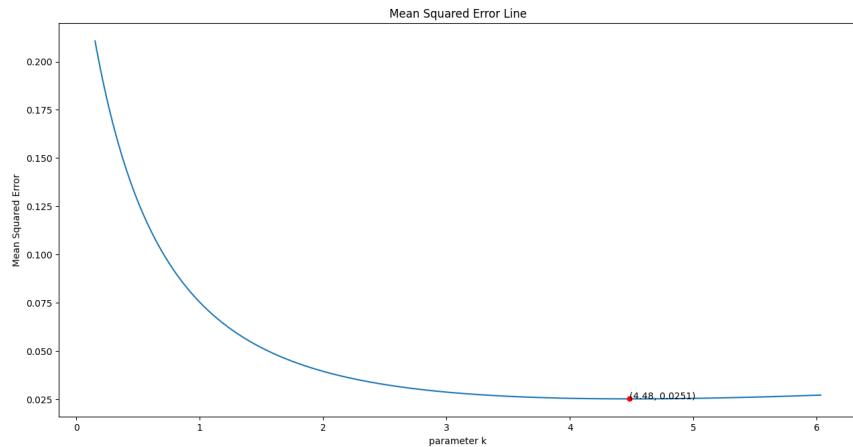
$\Sigma\chi\eta\mu\alpha$ 7: Noise level = 0.02, length = 10, angle = 0



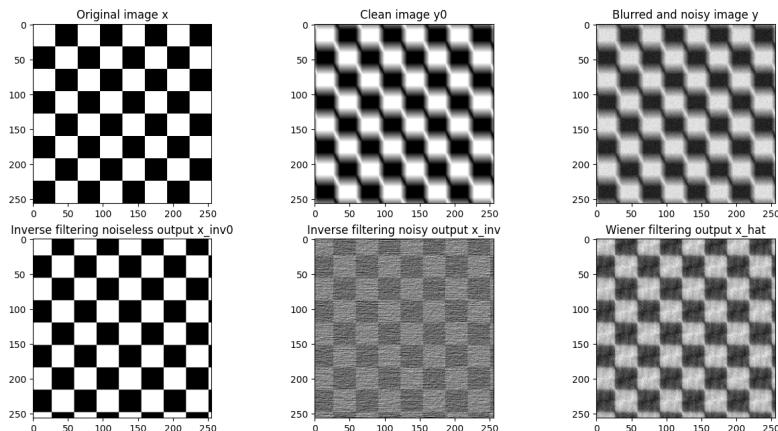
$\Sigma\chi\mu\alpha$ 8: Noise level = 0.2, length = 10, angle = 0

Signal-to-Noise-Ratio: 1.5082965103889403

Optimal value of Wiener filter k parameter: 4.48457770251078

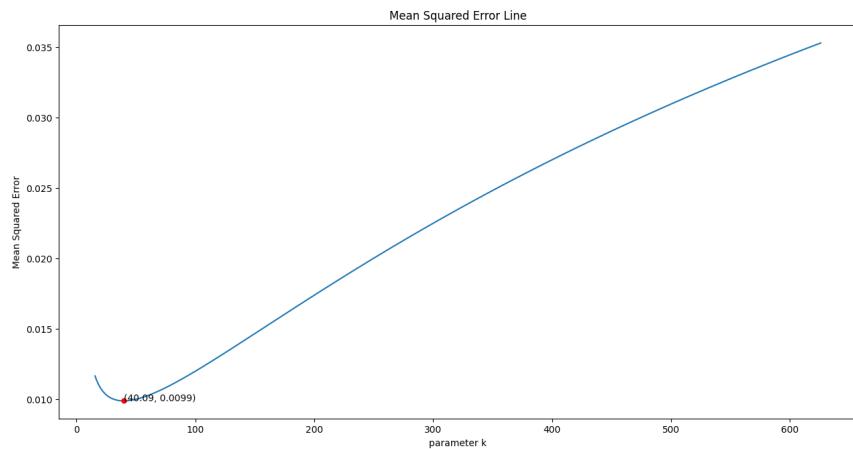


$\Sigma\chi\mu\alpha$ 9: Noise level = 0.2, length = 10, angle = 0

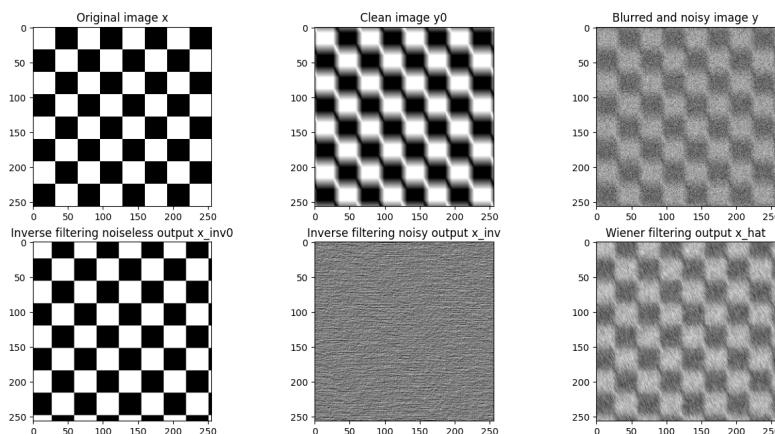


$\Sigma\chi\mu\alpha$ 10: Noise level = 0.02, length = 20, angle = 30

Signal-to-Noise-Ratio: 156.4888173627534
Optimal value of Wiener filter k parameter: 40.08557393706566

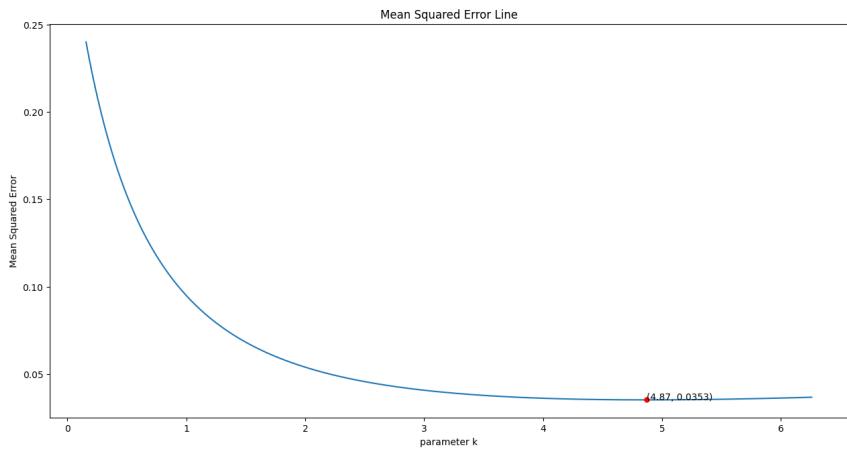


$\Sigma\chi\eta\mu\alpha$ 11: Noise level = 0.02, length = 20, angle = 30

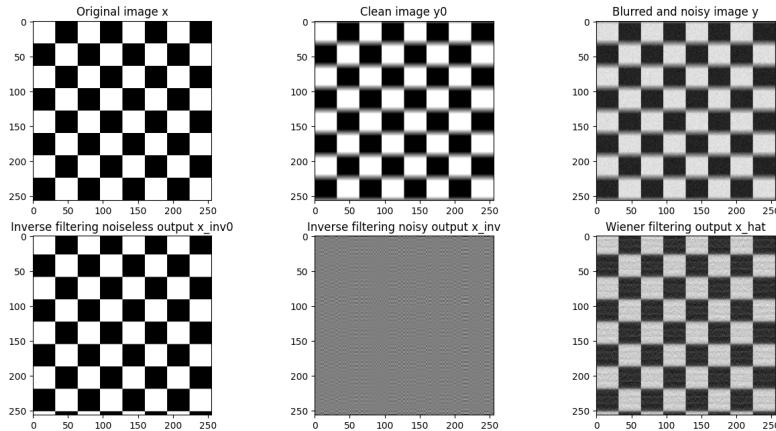


$\Sigma\chi\eta\mu\alpha$ 12: Noise level = 0.2, length = 20, angle = 30

Signal-to-Noise-Ratio: 1.5082965103889403
Optimal value of Wiener filter k parameter: 4.51401892568654

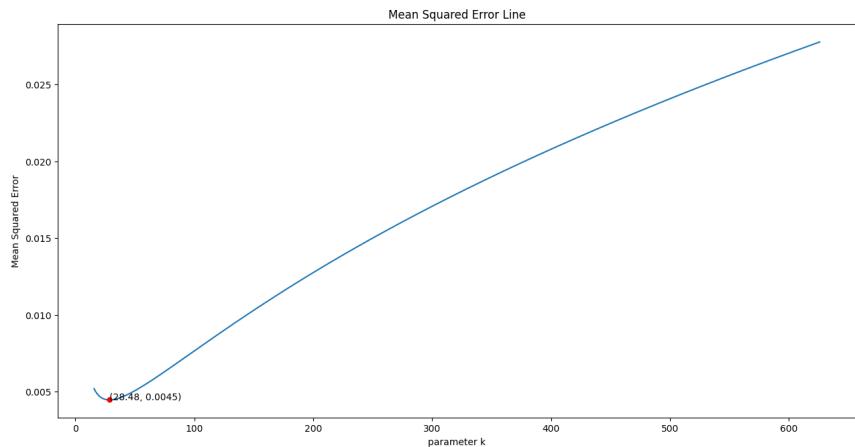


$\Sigma\chi\nu\alpha$ 13: Noise level = 0.2, length = 20, angle = 30

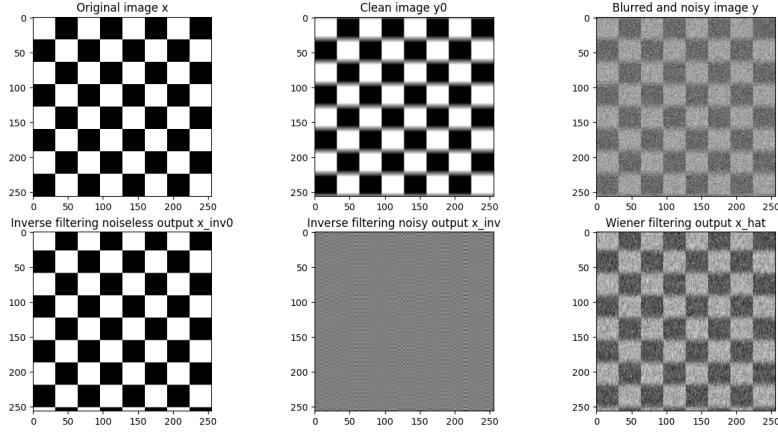


$\Sigma\chi\nu\alpha$ 14: Noise level = 0.02, length = 10, angle = 0

Signal-to-Noise-Ratio: 156.4888173627534
 Optimal value of Wiener filter k parameter: 28.478145141690256



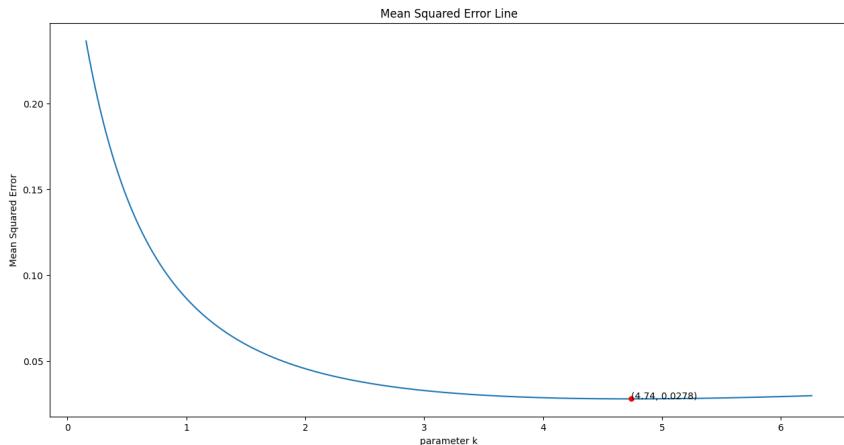
$\Sigma\chi\nu\alpha$ 15: Noise level = 0.02, length = 10, angle = 0



Σχήμα 16: Noise level = 0.2, length = 10, angle = 0

Signal-to-Noise-Ratio: 1.564888173627534

Optimal value of Wiener filter k parameter: 4.738368605010939



Σχήμα 17: Noise level = 0.2, length = 10, angle = 0

Παρατηρούμε ότι το αντίστροφο φίλτρο δεν επιφέρει ικανοποιητικά αποτελέσματα καθώς ενισχύει τον θόρυβο. Ειδικότερα, για το υψηλότερο επίπεδο θορύβου, χάνεται όλη η πληροφορία των αρχικών εικόνων και η έξοδος του φίλτρου αποτελείται μόνο από θόρυβο. Επιπλέον, αν και το φίλτρο Wiener επιφέρει καλύτερα αποτελέσματα από το αντίστροφο φίλτρο, δεν καταφέρνει να απαλείψει εντελώς την επίδραση του θορύβου. Επιτυγχάνει όμως να μειώσει αρκετά τη θόλωση. Ωστόσο, διαπιστώνουμε ότι δημιουργείται ένα ripple γύρω από τα περιγράμματα των αντικειμένων.

Τέλος, σχολιάζουμε ότι η τιμή του SNR στην παράμετρο K δεν επιφέρει την ελαχιστοποίηση του σφάλματος. Επιλέγονται τιμές του K που ελαχιστοποιούν το σφάλμα, οι οποίες είναι είτε μικρότερες είτε μεγαλύτερες από το SNR ανάλογα με το επίπεδο του θορύβου.