



ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρώτη εργαστηριακή εργασία

Φοιτητής: *Ιάκωβος Μαστρογιαννόπουλος*

Μάθημα: *Επεξεργασία Εικόνας* – Καθηγητής: *Ευτύχιος Πρωτοπαπαδάκης*

Abstract

Στην πρώτη εργαστηριακή εργασία, ζητείται να υλοποιηθούν δύο παραδείγματα επεξεργασίας εικόνας. Το πρώτο παράδειγμα, πρέπει να χρησιμοποιηθεί ο αλγόριθμος `solarize` για να δημιουργηθεί το φαινόμενο *Sabbatier*. Στο δεύτερο παράδειγμα, πρέπει να προστεθεί θόρυβος σε μία εικόνα, με την χρήση του `salt and pepper` και να τον αφαιρεί με χρήση φίλτρων. Υλοποιήθηκε σε γλώσσα προγραμματισμού Python, έκδοση 3.9.9 σε περιβάλλον Linux.

Contents

1	Μέρος πρώτο: εκμάθηση μιας νέας σημειακής λειτουργίας	3
1.1	Υλοποίηση της συνάρτησης solarize	3
1.2	Εφαρμογή της συνάρτησης Solarize	3
1.3	Αποτελέσματα εφαρμογής	3
2	Μέρος δεύτερο: διαχείριση θορύβου και επιλογή φίλτρων	5
2.1	Υποβάθμιση εικόνας με την χρήση θορύβου	5
2.1.1	Υλοποίηση του αλγορίθμου salt and pepper	5
2.1.2	Υλοποίηση του αλγορίθμου poisson	6
2.2	Επαναφορά της εικόνας μέσω φίλτρων	6
2.2.1	Γκαουσιανό φίλτρο	6
2.2.2	Φίλτρο του μέσου όρου	7
2.2.3	Φίλτρο της ενδιάμεσης τιμής	7
2.3	Εφαρμογή αλγορίθμων	7
2.3.1	Εφαρμογή θορύβων	7
2.3.2	Εφαρμογή φίλτρων	8
2.4	Σύγκριση με την χρήση αλγορίθμων	9
2.4.1	Υλοποίηση του αλγορίθμου SSIM	9
2.4.2	Υλοποίηση του αλγορίθμου Mean Square Error	9
2.4.3	Αποτελέσματα	10
2.5	Συμπεράσματα	10

Listings

1	Solarize Function	3
2	Salt And Pepper	5
3	Poisson	6
4	Gauss Sharp	6
5	Average Sharp	7
6	Median Blur	7
7	SSIM Algorithm	9
8	Mean Squared Error Algorithm	9

List of Figures

1	The Legend of Zelda - The Windwaker HD	3
2	Phil Collins - Face Value	7

List of Tables

1	Τα αποτελέσματα των συγκρίσεων μεταξύ του πρωτότυπου και της εξομάλυνσης του Poisson	4
2	Εφαρμογή των αλγορίθμων του θορύβου στις εικόνες	8
3	Εφαρμογή φίλτρων στην εικόνα με τον θόρυβο Salt and Paper	8
4	Εφαρμογή φίλτρων στην εικόνα με τον θόρυβο Poisson	9
5	Τα αποτελέσματα των συγκρίσεων μεταξύ του πρωτότυπου και της εξομάλυνσης του Salt and Pepper	10
6	Τα αποτελέσματα των συγκρίσεων μεταξύ της εικόνας με τον θόρυβο και της εξομάλυνσης του Salt and Pepper	10
7	Τα αποτελέσματα των συγκρίσεων μεταξύ του πρωτότυπου και της εξομάλυνσης του Poisson	10
8	Τα αποτελέσματα των συγκρίσεων μεταξύ της εικόνας με τον θόρυβο και της εξομάλυνσης του Poisson	10

1. Μέρος πρώτο: εκμάθηση μιας νέας σημειακής λειτουργίας

1.1. Υλοποίηση της συνάρτησης solarize. Το solarize είναι ένας αλγόριθμος στην επεξεργασία εικόνας η οποία δημιουργεί το φαινόμενο Sabattier. Το φαινόμενο Sabattier είναι το φαινόμενο στις φωτογραφίες, όπου η εικόνα είναι μαγνητοσκοπημένη σε ένα αρνητικό.

Θα πρέπει να δημιουργηθεί μία συνάρτηση, η οποία θα υλοποιεί το φαινόμενο Sabattier και στην συνέχεια θα την κάνει ασπρόμαυρη.

Αυτό μπορεί να υλοποιηθεί πολύ εύκολα με την χρήση του NumPy. Το NumPy κοιτάει τις τιμές της εικόνας και όπου είναι μεγαλύτερο από την τιμή κατωφλίου (threshold), τότε βάζει την αντίθετη τιμή, διαφορετικά δεν την αλλάζει.

```
def solarize(image, threshold):  
    return np.where((image < threshold), image, ~image)
```

Listing 1: Solarize Function

1.2. Εφαρμογή της συνάρτησης Solarize. Η συνάρτηση θα χρησιμοποιήσει την εικόνα του σχήματος 1. Οι τιμές του threshold πρόκειται να είναι 64, 128 και 192.



Figure 1: Screenshot από το βιντεοπαιχνίδι «The Legend of Zelda - The Windwaker HD» της Nintendo, 2013

1.3. Αποτελέσματα εφαρμύγης. Τα αποτελέσματα των διαφορετικών τιμών του threshold εμφανίζονται στον πίνακα 1. Παρατηρώντας τα αποτελέσματα, μπορεί να παρατηρηθεί ότι όσο πιο πολύ μεγαλώνει η τιμή του threshold, τόσο πιο πολύ η original εικόνα δεν είναι αναγνωρίσιμη.




Τιμή threshold	Αποτέλεσμα
64	 A grayscale screenshot from a video game. The scene shows a character on a platform with a large, stylized 'S' logo in the background. The image is somewhat blurry and has a low level of contrast.
128	 A grayscale screenshot from a video game, similar to the one above. The image is slightly sharper and has a bit more contrast than the threshold 64 version.
192	 A grayscale screenshot from a video game, similar to the ones above. The image is the sharpest and has the highest contrast of the three, with more detail visible in the background and foreground.

Table 1: Τα αποτελέσματα των συγκρίσεων μεταξύ του πρωτότυπου και της εξομάλυνσης του Poisson

2. Μέρος δεύτερο: διαχείριση θορύβου και επιλογή φίλτρων

Σε αυτό το μέρος της εργασίας, πρόκειται να γίνει υποβάθμιση της εικόνας προσθέτοντας θόρυβο στην αρχική εικόνα και στην συνέχεια να γίνει η επιλογή των καταλλήλων φίλτρων για να επανέλθει στην αρχική της κατάσταση. Θόρυβος στην επιστήμη των σημάτων ονομάζονται τα στοχαστικά σήματα τα οποία έχουν τυχαίες τιμές. Στην επεξεργασία εικόνας, το σήμα αντιστοιχείται με το pixel στην οθόνη και ο θόρυβος με τις φυσικές καταστάσεις που γίνετε τριγύρω από την φωτογραφία. Για παράδειγμα, έστω ότι ένας φωτογράφος τραβάει φωτογραφία ένα ποτάμι. Το ποτάμι δεν θα είναι πότε σταθερό. Θα πρέπει να μπορεί να βρίσκεται σε θέση να αφαιρέσει τον θόρυβο στην φωτογραφία του με την χρήση των καταλλήλων φίλτρων. Για την εργασία, θα πρέπει να βρεθεί ένας τρόπος ώστε να υποβαθμιστεί η original εικόνα, με σκοπό να επαναφερθεί στην αρχική της κατάσταση.

2.1. Υποβάθμιση εικόνας με την χρήση θορύβου. Για να γίνει η υποβάθμιση της εικόνας, θα πρέπει να προστεθεί ο θόρυβος.

Ποιοι είναι γνωστοί και εύκολοι αλγόριθμοι που μπορούν να υλοποιηθούν για να προστεθεί ο θόρυβος;

Οι δύο εύκολοι αλγόριθμοι που μπορούν να χρησιμοποιηθούν είναι ο **salt and pepper** και ο **poisson**

2.1.1. Υλοποίηση του αλγορίθμου salt and pepper. Ο αλγόριθμος salt and pepper είναι ένας αλγόριθμος ο οποίος δουλεύει πολύ καλά σε ασπρόμαυρες φωτογραφίες. Όπως φαίνεται και από το όνομα, το salt (δηλαδή το αλάτι, το άσπρο) βρίσκει όλα τα άσπρα σημεία του pepper (πιπέρι, δηλαδή μαύρο) και το pepper το αντίθετο.

Ο αλγόριθμος μπορεί να υλοποιηθεί με την χρήση του NumPy και του random. Αυτό που κάνει ο αλγόριθμος με λίγα λόγια είναι να φτιάχνει ένα αντίγραφο της εικόνας. Στην συνέχεια, παίρνει έναν τυχαίο αριθμό, τον οποίο τον ελέγχει με ένα όρισμα. Μαθηματικά:

$$new_image[i][j] = \begin{cases} 0 & \text{εάν } random_number < probability \\ 255 & \text{εάν } random_number > (1 - probability) \\ \text{αλλιώς } image[i][j] & \end{cases} \quad (1)$$

Όπου:

- *noise_image* είναι το αντίγραφο της εικόνας
- *probability* είναι το όρισμα που δέχεται
- *image* είναι η εικόνα η ίδια

Η υλοποίηση στην Python:

```
def salt_and_pepper(image, probability):
    noise_image = np.zeros(image.shape, np.uint8)

    for row in range(image.shape[0]):
        for col in range(image.shape[1]):
            rand = random()
            if rand < probability:
                noise_image[row][col] = 0
            elif rand > (1 - probability):
                noise_image[row][col] = 255
            else:
                noise_image[row][col] = image[row][col]

    return noise_image
```

Listing 2: Salt And Pepper

2.1.2. Υλοποίηση του αλγορίθμου poisson.

“Αν ένα φαινόμενο επαναλαμβάνεται τυχαία στον χρόνο (ή χώρο) με μέσο αριθμό επαναλήψεων λ σε ένα χρονικό (χωρικό) διάστημα, η συνάρτηση πιθανότητας της τυχαίας μεταβλητής X , η οποία μπορεί να είναι ο αριθμός επαναλήψεων στο διάστημα αυτό.

$$f_x(x) = e^{-\lambda} * \frac{\lambda^x}{x!}, x = 0, 1, \dots \quad (2)$$

Στην περίπτωση αυτή, λέμε ότι η X ακολουθεί **κατανομή Poisson** με παράμετρο λ και γράφουμε”

$$X \sim P_o(\lambda) \quad (3)$$

(N. Μυλωνάς, B. Παπαδόπουλος)

Στην επεξεργασία εικόνας, μπορεί να προστεθεί θόρυβος από την κατανομή του Poisson. Το NumPy το έχει ήδη εφαρμοσμένο.

```
def poisson(image):
    noise_image = np.random.poisson(image).astype(np.uint8)
    return image + noise_image
```

Listing 3: Poisson

2.2. Επαναφορά της εικόνας μέσω φίλτρων.

“Ο όρος φίλτρο έχει προέλθει από την επεξεργασία στο πεδίο της συχνότητας όπου αναφέρεται στην αποδοχή ή την απόρριψη συγκεκριμένων συχνοτήτων μιας εικόνας.”

(R. C. Gonzalez, R. E. Woods)

Για το συγκεκριμένο παράδειγμα, θα πρέπει να αφαιρεθούν με την χρήση τριών φίλτρων. Τα τρία φίλτρα τα οποία χρησιμοποιούνται είναι το **Γκαουσιανό φίλτρο**, το **φίλτρο του μέσου όρου** και το **φίλτρο της ενδιάμεσης τιμής**.

2.2.1. Γκαουσιανό φίλτρο.

“Αν το πλήθος των κλάσεων είναι πολύ μεγάλο (τείνει στο άπειρο) και το πλάτος κλάσεων είναι πολύ μικρό (τείνει στο μηδέν), τότε τα πολύγωνα συχνοτήτων παίρνουν τη μορφή μίας ομαλής καμπύλης, οι οποίες ονομάζονται **καμπύλες συχνοτήτων** ή **καμπύλες σχετικών συχνοτήτων**. Σε πολλές περιπτώσεις μεταβλητών η καμπύλη σχετικών συχνοτήτων έχει μία κωδωνοειδή μορφή. Η καμπύλη αυτή ονομάζεται **κανονική κατανομή**.”

(N. Μυλωνάς, B. Παπαδόπουλος)

Το Γκαουσιανό φίλτρο είναι ένα φίλτρο το οποίο χρησιμοποιεί την κανονική κατανομή για να επαναφέρει την εικόνα στην αρχική της κατάσταση. Η βιβλιοθήκη cv2 έχει μερικές συναρτήσεις που υλοποιούν το Γκαουσιανό φίλτρο.

```
def calc_gauss_sharp(image):
    kernel = np.ones((5, 5), np.float32) / 25
    gauss_image = cv2.GaussianBlur(image, (5, 5), 0)

    kernel = np.array(
        [
            [0, -1, 0],
            [-1, 5, -1],
            [0, -1, 0]
        ]
    )

    return cv2.filter2D(gauss_image, -1, kernel)
```

Listing 4: Gauss Sharp

2.2.2. Φίλτρο του μέσου όρου. Υπάρχουν πολλά φίλτρα μέσου όρου, αλλά αυτό το οποίο θα χρησιμοποιηθεί είναι το φίλτρο αριθμητικού μέσου.

“Το φίλτρο αριθμητικού μέσου αποτελεί και το πιο απλό από τα φίλτρα υπολογισμού του μέσου όρου τιμής.” (R. C. Gonzalez, R. E. Woods)

Εστω S_{xy} το σύνολο των συντεταγμένων μίας εικόνας $m * n$ διαστάσεων που έχει κέντρο το (x, y) Τότε:

$$\hat{f}(x, y) = \frac{1}{m * n} * \sum_{(r, c) \in S_{xy}} g(r, c) \quad (4)$$

```
def calc_average_sharp(image):
    kernel = np.array(
        [
            [0, -1, 0],
            [-1, 5, -1],
            [0, -1, 0]
        ]
    )
    return cv2.filter2D(image, -1, kernel)
```

Listing 5: Average Sharp

2.2.3. Φίλτρο της ενδιάμεσης τιμής.

“Το πιο γνωστό φίλτρο στατιστικής διάταξης είναι το φίλτρο ενδιάμεσης τιμής που αντικαθιστά την τιμή ενός εικονοστοιχείου με την τιμή διάμεσου των επιπέδων έντασης που ανήκουν στη γειτονιά αυτού του εικονοστοιχείου.” (R. C. Gonzalez, R. E. Woods)

$$\hat{f}(x, y) = \text{median}\{g(r, c)\} \quad (5)$$

```
def calc_median(image):
    median_blur = cv2.medianBlur(image, 5)
```

Listing 6: Median Blur

2.3. Εφαρμογή αλγορίθμων.

2.3.1. Εφαρμογή θορύβων. Οι αλγόριθμοι πρόκειται να εφαρμοστούν πάνω στο Σχήμα 2. Στον πίνακα 2 απεικονίζονται οι νέες εικόνες με τον θόρυβο. Παρατηρείται ότι και στις δύο περιπτώσεις, η αρχική εικόνα αλλάζει πάρα πολύ, κυρίως στον Poisson.

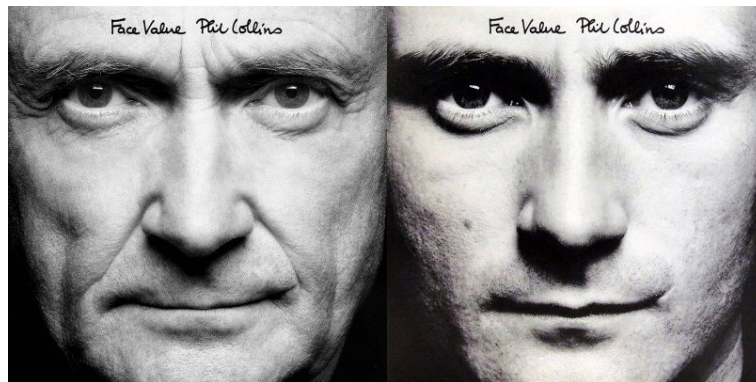


Figure 2: Δύο φωτογραφίες του διάσημου τραγουδιστή και drummer των Genesis, Phil Collins που τραβήχτηκαν σε δύο χρονικές περιόδους για το πρώτο του solo album, Face Value, 1981 και 2015

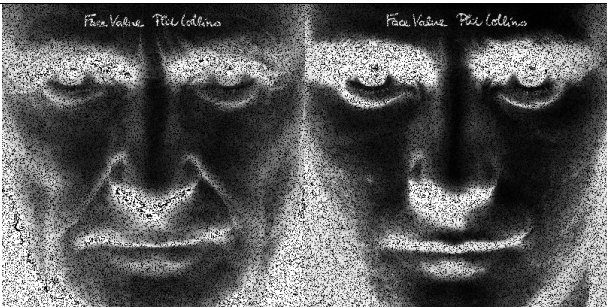
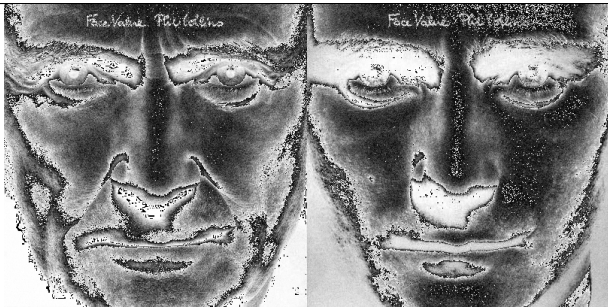
Salt and Paper	Poisson
	

Table 2: Εφαρμογή των αλγορίθμων του θορύβου στις εικόνες

2.3.2. Εφαρμογή φίλτρων. Στον πίνακα 3 φαίνονται τα αποτελέσματα από τα φίλτρα στην εικόνα με τον θόρυβο Salt and Pepper, ενώ στον πίνακα 4 τα αποτελέσματα από τον Poisson.



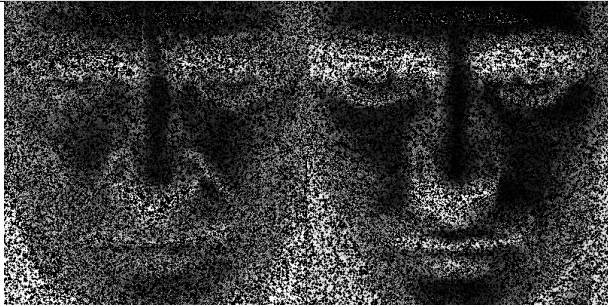
Φίλτρο	Εικόνα
Γκαουσιανό φίλτρο	
Φίλτρο μέσου όρου	
Φίλτρο διάμεσης τιμής	

Table 3: Εφαρμογή φίλτρων στην εικόνα με τον θόρυβο Salt and Paper




Φίλτρο	Εικόνα
Γκαουσιανό φίλτρο	
Φίλτρο μέσου όρου	
Φίλτρο διάμεσης τιμής	

Table 4: Εφαρμογή φίλτρων στην εικόνα με τον θόρυβο Poisson

2.4. Σύγκριση με την χρήση αλγορίθμων. Για το τελευταίο βήμα, θα χρειαστεί να γίνει σύγκριση των εικόνων μεταξύ (α) της πρωτότυπης εικόνας με την εξομάλυνση και (β) της εικόνας με τον θόρυβο με την εξομάλυνση. Οι δύο αλγόριθμοι που θα χρησιμοποιηθούν είναι ο αλγόριθμος **Structural Similarity Index (SSIM)** και ο **Mean Square Error**.

2.4.1. Υλοποίηση του αλγορίθμου SSIM. Ο αλγόριθμος **Structural Similarity Index (SSIM)** βρίσκει την διαφορά μεταξύ δύο παρόμοιων εικόνων. Στο skimage.metrics υπάρχει έτοιμη συνάρτηση ονοματι `ssim` που το υλοποιεί.

```
def calc_simil(name, original_image, image):
    simil_score, _ = ssim(original_image, image, full=True)
    print(f"{name}_SSIM_score is: {simil_score}")
```

Listing 7: SSIM Algorithm

2.4.2. Υλοποίηση του αλγορίθμου Mean Square Error. Ο αλγόριθμος **Mean Square Error** είναι ένας μετρητής που βρίσκει τον μέσο όρο των λαθών της δύναμης. Ο αλγόριθμος μπορεί να υλοποιηθεί πολύ απλά με το NumPy.

```
def calc_mean(name, original_image, image):
    mean_squared_error = np.square(np.subtract(original_image, image)).mean()
    print(f"{name}_MeanSquaredError is: {mean_squared_error}")
```

Listing 8: Mean Squared Error Algorithm

2.4.3. Αποτελέσματα. Τα αποτελέσματα μπορούν να φανούν στους πίνακες 5, 6, 7 και 8.

Φίλτρο	Αποτέλεσμα SSIM	Αποτέλεσμα Mean Square Error
Γκαουσιανό φίλτρο	0.22224864314683754	95.79599344863892
Φίλτρο μέσης τιμής	0.1008267520614758	94.49013601462973
Φίλτρο διάμεσης τιμής	0.7457517695290394	40.69571883331589

Table 5: Τα αποτελέσματα των συγκρίσεων μεταξύ του πρωτότυπου και της εξομάλυνσης του Salt and Pepper

Φίλτρο	Αποτέλεσμα SSIM	Αποτέλεσμα Mean Square Error
Γκαουσιανό φίλτρο	0.5256954158313445	94.97967477171832
Φίλτρο μέσης τιμής	0.8065523248733352	73.49161068659988
Φίλτρο διάμεσης τιμής	0.0813248901272799	53.02096866552962

Table 6: Τα αποτελέσματα των συγκρίσεων μεταξύ της εικόνας με τον θόρυβο και της εξομάλυνσης του Salt and Pepper

Φίλτρο	Αποτέλεσμα SSIM	Αποτέλεσμα Mean Square Error
Γκαουσιανό φίλτρο	0.43015755824644086	105.33047737502233
Φίλτρο μέσης τιμής	0.12037766178876502	97.6301775147929
Φίλτρο διάμεσης τιμής	0.5137574899674178	108.9787480989354

Table 7: Τα αποτελέσματα των συγκρίσεων μεταξύ του πρωτότυπου και της εξομάλυνσης του Poisson

Φίλτρο	Αποτέλεσμα SSIM	Αποτέλεσμα Mean Square Error
Γκαουσιανό φίλτρο	0.7198557213172727	75.2685503882174
Φίλτρο μέσης τιμής	0.5889702706987753	99.21519127634552
Φίλτρο διάμεσης τιμής	0.42011072522698933	78.37677099175538

Table 8: Τα αποτελέσματα των συγκρίσεων μεταξύ της εικόνας με τον θόρυβο και της εξομάλυνσης του Poisson

2.5. Συμπεράσματα. Ο αλγόριθμος Salt and Pepper φαίνεται να αλλάζει την εικόνα λιγότερο από την Poisson. Παρ'όλ'αυτά, στον Poisson φαίνονται τα χαρακτηριστικά του προσώπου καλύτερα από ότι στο Salt and Pepper. Στα φίλτρα, το φίλτρο διάμεσης τιμής είναι το καλύτερο φίλτρο, επαναφέρει την εικόνα αρκετά πιο κοντά από ότι στις υπόλοιπες και αυτό φαίνεται στα αποτελέσματα του SSIM και του Mean Square Error.

Βιβλιογραφία

- Ψηφιακή Επεξεργασία Εικόνας (R. C. Gonzalez, R. E. Woods, Απόδοση Σ. Κόλλιας, εκδόσεις Τζιόλα, 2021)
- Πιθανότητες και Στατιστική (Ν. Μυλωνάς, Β. Παπαδόπουλος, εκδόσεις Τζιόλα, 2018)
- Σήματα και Συστήματα (Π. Φωτόπουλος, Α. Βελώνη, εκδόσεις Σύγχρονη Εκδότικη, 2008)