Πανεπιστημιο Δυτικης Αττικης
Τμημα Μηχανικων Πληροφορικης και Υπολογιστων

Δεύτερη Εργαστηριακή Εργασία

Φοιτητής: ***Ιάκωβος Μαστρογιαννόπουλος***

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

Περίληψη

Αυτή είναι η δεύτερη εργαστηριακή εργασία για το μάθημα «Επεξεργασία Εικόνας» του φοιτητή Ιάκωβος Μαστρογιαννόπουλος, 713242017102. Η εργασία είναι χωρισμένη σε δύο μέρη. Στο πρώτο μέρος πρέπει να μελετηθούν οι χώροι χρώματος και να γίνει κατάτμηση μίας εικόνας. Στο δεύτερο μέρος πρέπει να γίνει ανίχνευση των ακμών μιας εικόνας. Η συγκεκριμένη εργασία υλοποιήθηκε στην γλώσσα προγραμματισμού Python, έκδοση 3.10.1 σε περιβάλλον Linux.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

1. Χώροι χρώματος και κατάτμηση

Σε αυτό το κεφάλαιο θα πραγματοποιηθεί ένα παράδειγμα ιστογράμματος και κατάτμησης για να μελετηθεί η διαφορά του RGB, του HSV και του συνδυασμού τους.

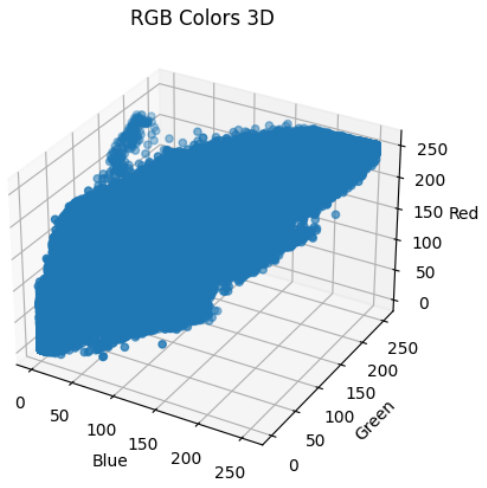
1.1. Επιλογή εικόνας. Η εικόνα η οποία επιλέχτηκε για να εκτελεστούν οι αλγόριθμοι πάνω της είναι η εικόνα του Σχήματος 1

Σχήμα 1: Οι διάφορες μορφές (regenerations) του Doctor από την σειρά «Doctor Who» του BBC, 1963 - 1989 και 2005 - σήμερα

1.1.1. 3D plot της original RGB εικόνας. Για να φορτωθεί η εικόνα στο script και να την θεωρήσει ως RGB χρειάζονται μερικές εντολές. Τα ακρώνυμα RGB σημαίνει Red (κόκκινο), Green (πράσινο) και Blue (Μπλε). Η OpenCV όμως τα διαβάζει ως BGR και θα πρέπει να γίνει η κατάλληλη μετατροπή σε RGB. Αυτό γίνετε με τον εξής τρόπο:

```
image = cv2.imread('the_doctor.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

Για να πραγματοποιηθεί 3D plot, θα πρέπει να γίνει διαχωρισμός των καναλιών. Υπάρχει έτοιμη συνάρτηση στο OpenCV που το κάνει αυτόματα. Τα αποτελέσματα της συνάρτησης παρουσιάζονται στο Σχήμα 2.



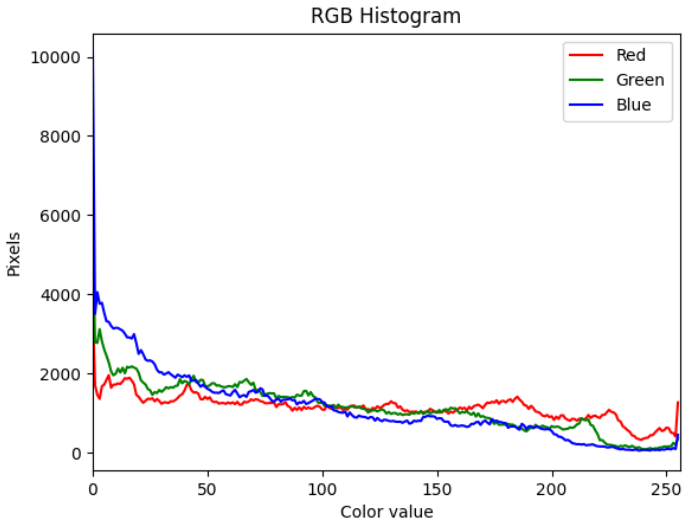
Σχήμα 2: RGB 3D Plot

1.1.2. Ιστόγραμμα της RGB εικόνας. Το ιστόγραμμα είναι ένα πολύ δυνατό εργαλείο στην επεξεργασία εικόνας. Απεικονίζει την διαφορά των τιμών μίας εικόνας ανά κανάλι. Μπορεί να υπολογισθεί πολύ εύκολα χρησιμοποιώντας την βιβλιοθήκη NumPy.

```
for channel_id, channel in zip(channel_ids, channels):
  histogram, bin_edges = np.histogram(image[:, :, channel_id], bins=bins, range=(0, bins))
```

Τροποποιώντας λίγο τον κωδικά, μπορεί να εισαχθεί σε ένα plot. Το αποτέλεσμα του plot παρουσιάζεται στο Σχήμα 3.

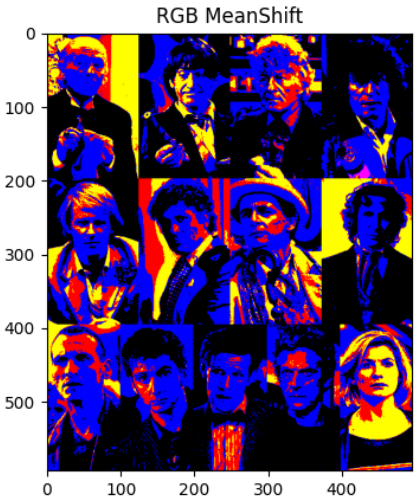
```
def plot_histogram(image, labels, bins,
                   title, filename, channels=('red', 'green', 'blue'),
                   channel_ids=(0, 1, 2)) -> None:
  plt.xlim([0, 256])

  for label, channel_id, channel in zip(labels, channel_ids, channels):
    histogram, bin_edges = np.histogram(
      image[:, :, channel_id],
      bins=bins,
      range=(0, bins)
    )
    plt.plot(bin_edges[0:-1], histogram, label=label, color=channel)

  plt.xlabel("Color value")
  plt.ylabel("Pixels")
  plt.title(title)
  plt.legend(loc="best")
  plt.savefig(f"{filename}.png")
  plt.show()

plot_histogram(image, ('Red', 'Green', 'Blue'), 256, "RGB Histogram", "rgb_histogram")
```



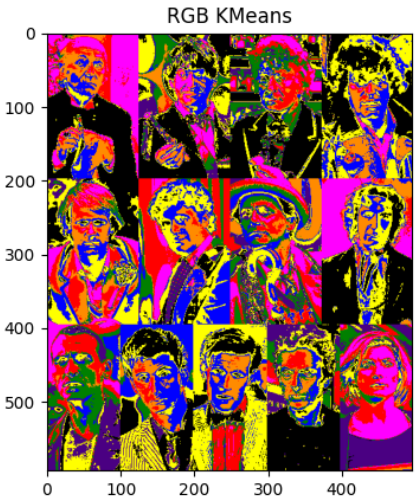
Σχήμα 3: RGB Histogram

1.1.3. Κατάτμηση της RGB εικόνας. Κατάτμηση είναι όταν μία εικόνα χωρίζεται σε μικρότερα τμήματα τα οποία είναι όμοια μαζί τους. Η κατάτμηση μπορεί να γίνει με πολλές μεθοδολογίες. Οι δύο που επιλέχτηκαν για το παράδειγμα είναι ο MeanShift και ο KMeans. Και οι δύο αλγόριθμοι είναι παρόμοιοι και κάνουν την ίδια δουλεία, εύρεση των clusters.

```
def segment_image(image, color_code, channel_amount=3) -> None:
  shape = image.shape
  flat_image = np.reshape(image, [-1, channel_amount])

  # MeanShift segmentation
  bandwidth = estimate_bandwidth(flat_image, quantile=0.1, n_samples=100)
  meanshift = MeanShift(bandwidth=bandwidth, bin_seeding=True)
  meanshift.fit(flat_image)
  segmentation(flat_image, shape, color_code, channel_amount, meanshift, "MeanShift")

  # KMeans segmentation
  k_means = MiniBatchKMeans()
  k_means.fit(flat_image)
  segmentation(flat_image, shape, color_code, channel_amount, k_means, "KMeans")

segment_image(image, "RGB")
```



Σχήμα 4: RGB MeanShift



Σχήμα 5: RGB KMeans

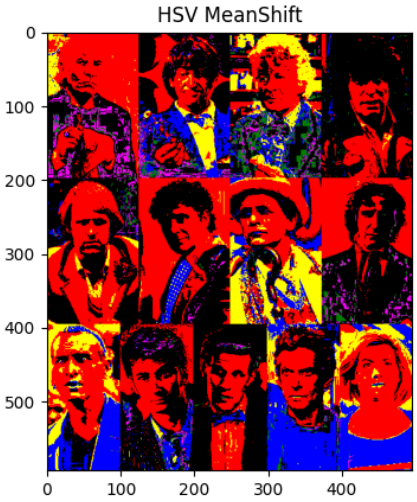
Στα Σχήματα 4 και 5 απεικονίζονται τα αποτελέσματα των αλγορίθμων.

1.2. Μετατροπή σε HSV και παρουσίαση αποτελεσμάτων. Το HSV είναι ένας εναλλακτικός τρόπος αναπαράστασης της εικόνας. Τα ακρώνυμα είναι Hue (απόχρωση), Saturation (κορεσμός) και Value (τιμή). Η μετατροπή από RGB σε HSV είναι η εξής:

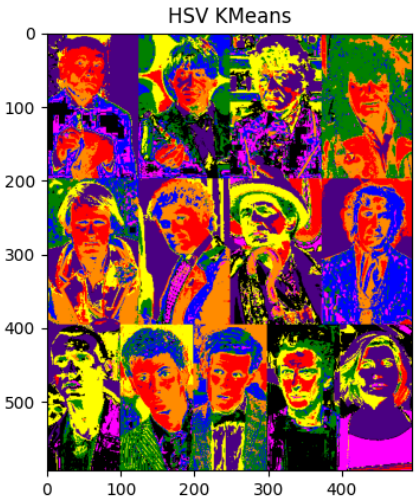
```
hsv_image = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)
save_image("hsv", hsv_image)
```

Η εικόνα σε HSV παρουσιάζεται στο Σχήμα 6. Να παρατηρηθεί πόσο πολύ έχει τροποποιηθεί η εικόνα. Στα Σχήματα 7 και 8 παρουσιάζονται τα αποτελέσματα των αλγορίθμων MeanShift και KMeans στην HSV εικόνα. Παρατηρείτε ότι τα αποτελέσματα έχουν επίσης τροποποιηθεί.

Σχήμα 6: Η HSV εικόνα



Σχήμα 7: HSV MeanShift



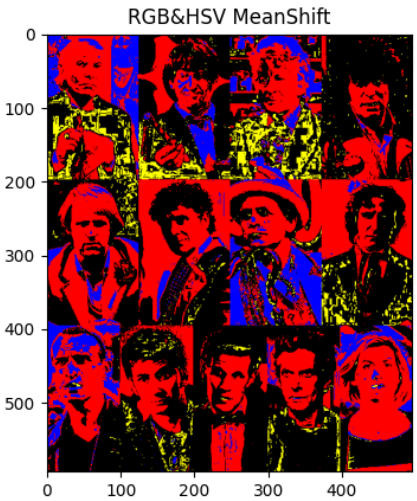
Σχήμα 8: HSV KMeans

1.3. Σχολιασμός αποτελεσμάτων RGB και HSV. Τα αποτελέσματα μεταξύ των δύο αλγορίθμων είτε ο KMeans είτε ο MeanShift είναι πολύ διαφορετικά μεταξύ τους με RGB και με το HSV. Ο MeanShift παράγει πολύ καλά με το RGB μοντέλο, ένω ο KMeans με το HSV μοντέλο.

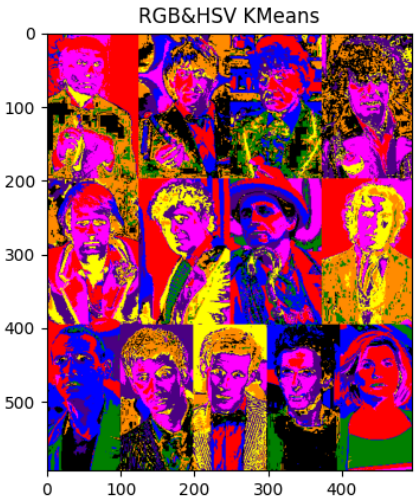
1.3.1. Συνδιασμός RGB και HSV. Μπορεί να εξελιχθεί λίγο παραπάνω το ερώτημα συνδυάζοντας τα δύο μοντέλα και περνώντας τα από τους δύο αλγορίθμους. Ο αλγόριθμος που συνδυάζει τα δύο μοντέλα είναι ο εξής:

```
combined_image = np.concatenate((image, hsv_image), axis=2)
```

Στα Σχήματα 9 και 10 φαίνονται τα αποτελέσματα των αλγορίθμων. Τα αποτελέσματα φαίνονται πολύ πιο καλύτερα από όταν ήταν ξεχωριστά τα μοντέλα.



Σχήμα 9: RGB και HSV MeanShift



Σχήμα 10: RGB και HSV KMeans

1.4. Silhouette Coefficient. Ο Silhouette Coefficient ή silhouette score είναι ένας αριθμός που καθορίζει το πόσο καλό είναι μία clustering τεχνική. Το εύρος τιμών του είναι από το -1 εώς το 1, όπου:

- 1: τα clusters μεταξύ τους είναι αρκετά μακρυά και εύκολα αναγνωρίσιμα
- 0: τα clusters είναι τελείως διαφορετικά μεταξύ τους
- -1: τα clusters έχουν ρυθμιστεί λάθος

Μαθηματικά:

$$SilhouetteScore = \frac{(b-a)}{max(a,b)} \tag{1}$$

όπου:

- το a είναι η μέση απόσταση των inter-cluster από κάθε σημείο
- το b είναι η μέση απόσταση όλων των clusters

Όνομα εικόνας	**Αριθμός clusters**	**Αποτέλεσμα Silhouette**
RGB MeanShift	5	0.43829281680682963
RGB KMeans	8	0.39261542176396413
HSV MeanShift	6	0.27099304439301336
HSV KMeans	8	0.3545841420133397
RGB και HSV MeanShift	4	0.262483435384828
RGB και HSV KMeans	8	0.33426919705784475

Πίνακας 1: Τα αποτελέσματα που γύρισε ο αλγόριθμος φαίνονται στον παρακάτω πίνακα. Οι παρατηρήσεις που έγιναν στο Κεφάλαιο 1.3 φαίνονται να επιβεβαιώνονται με τα αποτελέσματα.

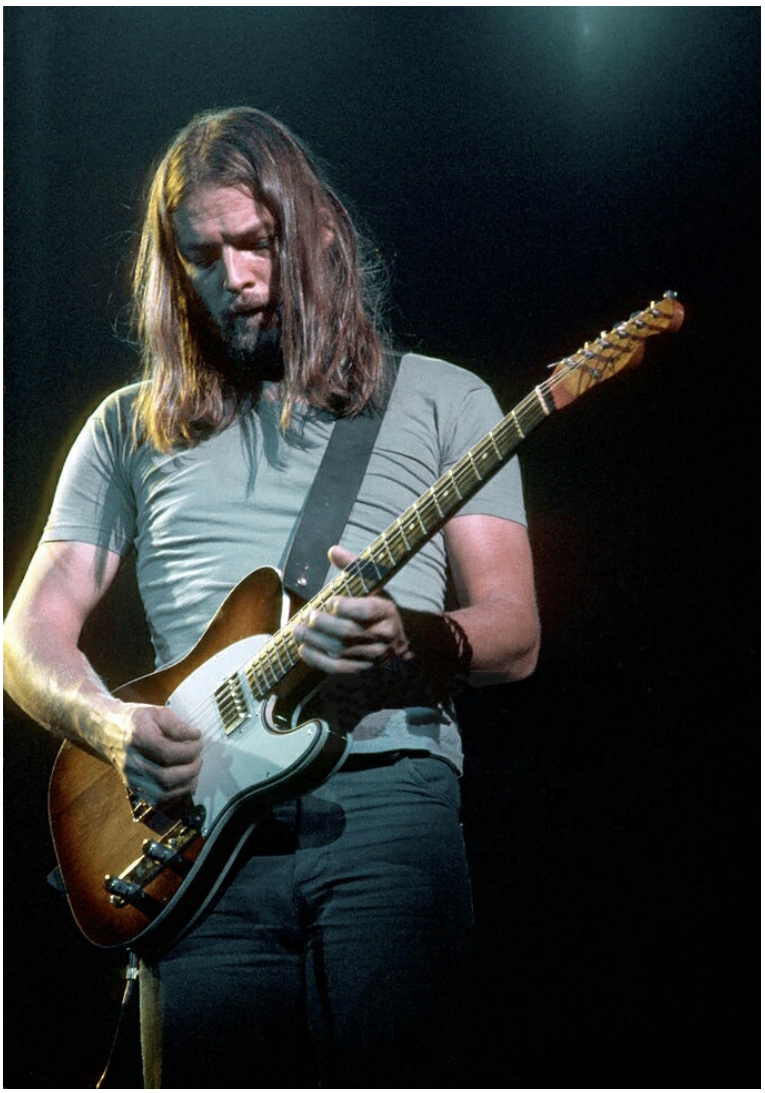
2. Μορφολογία και ανίχνευση ακμών

Στο αυτό το κομμάτι της εργασίας, το ζητούμενο είναι να γίνει ανίχνευση των ακμών με την χρήση της τεχνικής Canny σε μία grayscale εικόνα. Στην συνέχεια, πρέπει να προστεθεί και να παρατηρηθούν οι αλλαγές.

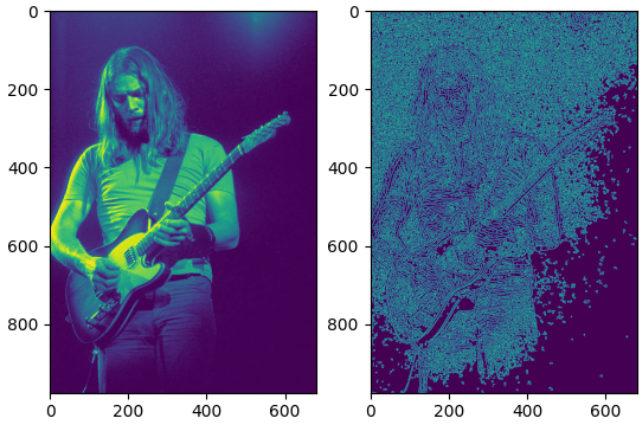
2.1. Τεχνική Canny. Η τεχνική ανίχνευση ακμών Canny είναι ένας διάσημος αλγόριθμος ανίχνευσης ακμών. Για να ολοκληρωθεί και να βρει τις ακμές περνάει από μερικά στάδια. Στην Python, παράγεται έτοιμο. Ο κώδικας είναι ο εξής:

```
def apply_edges(image, t_lower=100, t_upper=200, aperture_size=5, L2Gradient=True) -> any:
  return cv2.Canny(
    image, t_lower, t_upper,
    apertureSize=aperture_size, L2gradient=L2Gradient
  )
```

Η φωτογραφία που επιλέχτηκε για να γίνει η ανίχνευση των ακμών βρίσκετε στο Σχήμα 11, ενώ τα αποτελέσματα του αλγορίθμου στο Σχήμα 12.



Σχήμα 11: Ο David Gilmour, γνωστός για την συμμετοχή του στο συγκρότημα Pink Floyd και ως τρομερός μουσικός

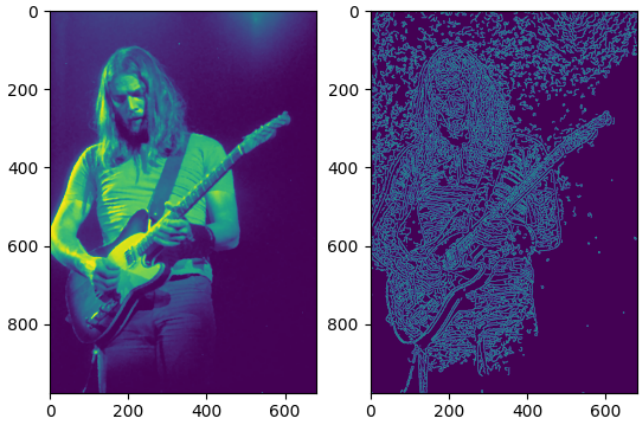


Σχήμα 12: Η σύγκριση του Original με την τεχνική canning. Να παρατηρηθεί ότι ο ίδιος φαίνεται ελάχιστα και πιάνει πολύ background noise ο αλγόριθμος

2.2. Γκαουσιανός θόρυβος. Εφαρμογή αλγορίθμου για προσθήκη Γκαουσιανού θορύβου στην εικόνα.

```
def add_noise_and_remove_it(image) -> any:
  gaussian_noise = np.random.normal(0, 1, image.size)
  gaussian_noise = gaussian_noise.reshape(image.shape[0], image.shape[1]).astype('uint8')
  noise = cv2.add(image, gaussian_noise)
  return cv2.medianBlur(noise, 5)
```

Στο Σχήμα 13, μπορεί να βγει το συμπέρασμα ότι προσθέτοντας θόρυβο στην εικόνα, μπορεί να βοηθήσει την τεχνική Canny για να εντοπίσει πιο σωστά τις ακμές της εικόνας.

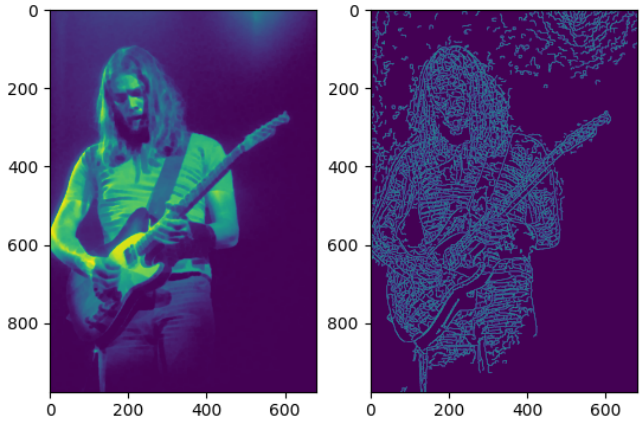


Σχήμα 13: Η σύγκριση της επαναφοράς της εικόνας με την τεχνική canning. Να παρατηρηθεί ότι ο ίδιος φαίνεται πολύ καλύτερα και μπορεί να παρατηρηθεί η μορφή του γνωστού κιθαρίστα

2.3. Συνδυασμός μορφολογίας με σύγκριση ακμών. Μορφολογία στην επεξεργασία εικόνας είναι το πως αναπαριστάτε η κάθε φιγούρα μέσα στην εικόνα. Υπάρχουν δύο μορφολογικοί τελεστές που χρησιμοποιήθηκαν σε αυτή την εργασία είναι ο erosion και ο dilation.

2.3.1. Μορφολογικός Τελεστής Erosion.

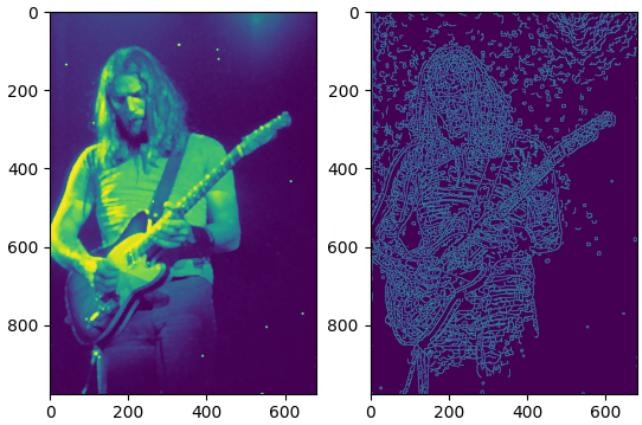
$$\mathrm{A} \ominus \mathrm{B} = \{z | (\mathrm{B})_z \subseteq \mathrm{A}\} \tag{2}$$



Σχήμα 14: Η σύγκριση της επαναφοράς της εικόνας αφού έχει περαστεί από erosion με την τεχνική canning. Η μορφή του κιθαρίστα είναι ακόμα πιο καθαρή από πριν, βεβαία περνάει και λίγο από το background noise μέσα στις ακμές του

2.3.2. Μορφολογικός Τελεστής Dilation.

$$A \oplus B = \{z|(\hat{B})_z \cap A\} \tag{3}$$



Σχήμα 15: Η σύγκριση της επαναφοράς της εικόνας αφού έχει περαστεί από dilation με την τεχνική canning. Η μορφή του κιθαρίστα είναι αρκετά ίδια με το erosion, με την διαφορά ότι έχει παραπάνω θόρυβο

2.4. Σύγκριμα αποτελεσμάτων με την χρήση αλγορίθμων. Για το πόσο καλά αποτελέσματα παράγει ο αλγόριθμος, θα ακολουθούν διάφορες τεχνικές. Για να γίνει οποιαδήποτε σύγκριση όμως χρειάζεται να γίνει μετατροπή των εικόνων σε ιστογράμματα για να είναι πιο επεξεργάσιμοι από τους καθημερινούς μοντέρνους υπολογιστές. Ο αλγόριθμος δοκιμάστηκε σε σταθερό υπολογιστή με Ryzen 5 2600 και 8 GB ram και είχε ελάχιστη καθυστέρηση.

2.4.1. Ευκλειδική Διαφορά. Η πρώτη διαφορά που θα μελετηθεί είναι η Ευκλειδική. Ο αλγόριθμος είναι ο εξής:

```
euclidean = 0
i = 0
while i < len(histogram_original) and i < len(histogram_edges):
  euclidean += (histogram_original[i] - histogram_edges[i]) ** 2
  i += 1
```

Όνομα εικόνας	**Αποτέλεσμα Ευκλειδικής Διαφοράς**
Original εικόνα	1.3595214842916974
Restored εικόνα	1.4140797849329154
Restored εικόνα με Erosion	1.4136366487406054
Restored εικόνα με Delation	1.4135856679087746

Πίνακας 2: Αποτελέσματα Ευκλειδικής διαφοράς

2.4.2. X^2 Διαφορά. Η δεύτερη διαφορά είναι η X^2 διαφορά. Δίνετε έτοιμο από το cv2.

Όνομα εικόνας	**Αποτέλεσμα X^2 Διαφοράς**
Original εικόνα	55.54426650712792
Restored εικόνα	9917.798371611698
Restored εικόνα με Erosion	1213.5299873964364
Restored εικόνα με Delation	7.813439429146478

Πίνακας 3: Αποτελέσματα X^2 διαφοράς

2.4.3. Bhattacharyya Διαφορά. Η τρίτη διαφορά είναι η Bhattacharyya διαφορά. Δίνετε έτοιμο από το cv2.

Όνομα εικόνας	**Αποτέλεσμα Bhattacharyya Διαφοράς**
Original εικόνα	0.9437122663789763
Restored εικόνα	0.9962561700845681
Restored εικόνα με Erosion	0.993916915962169
Restored εικόνα με Delation	0.9944792139506231

Πίνακας 4: Αποτελέσματα Bhattacharyya διαφοράς

2.4.4. Correlation Διαφορά. Η τέταρτη διαφορά είναι η correlation διαφορά. Δίνετε έτοιμο από το cv2.

Όνομα εικόνας	**Αποτέλεσμα Correlation Διαφοράς**
Original εικόνα	0.05002108348803092
Restored εικόνα	-0.028392138047218348
Restored εικόνα με Erosion	-0.022036753262860056
Restored εικόνα με Delation	-0.030544818302308076

Πίνακας 5: Αποτελέσματα Correlation διαφοράς

2.4.5. Intersection Διαφορά. Η τελευταία διαφορά είναι η Intersection διαφορά. Δίνετε έτοιμο από το cv2.

Όνομα εικόνας	**Αποτέλεσμα Intersection Διαφοράς**
Original εικόνα	0.08144057751633227
Restored εικόνα	0.0007671799830859527
Restored εικόνα με Erosion	0.0008191215456463397
Restored εικόνα με Delation	0.008301599882543087

Πίνακας 6: Αποτελέσματα Intersection διαφοράς