ΤΕΧΝΙΚΗ ΑΝΑΦΟΡΑ

`Για την Πρώτη Εργασία στο μάθημα

«Όραση Υπολογιστών»

Καθηγητής: Ιωάννης Πρατικάκης

Στέλιος Μούσλεχ

AM:57382

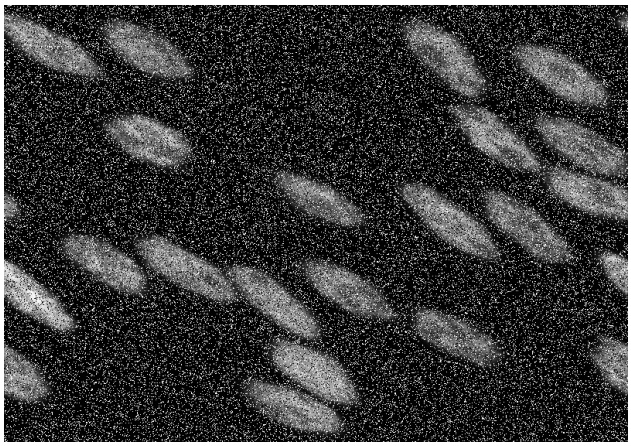
15/11/2019,Ξάνθη

ΕΙΣΑΓΩΓΗ

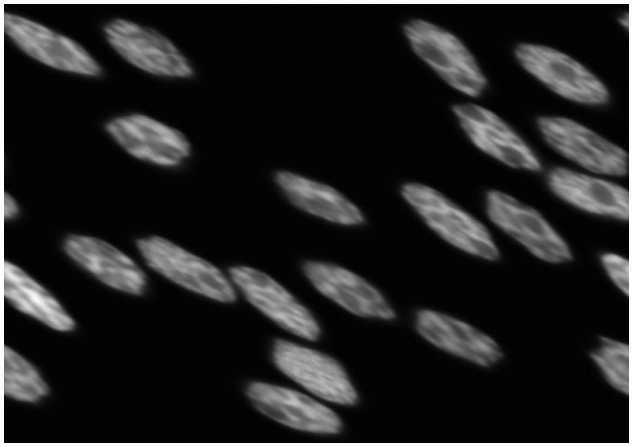
Στην παρακάτω τεχνική αναφορά θα περγραφεί η λύση της πρώτης εργασίας στο μάθημα «Όραση Υπολογιστών» αναλύοντας σε περιληπτικό επίπεδο το απαραίτητο θεωρητικό υπόβαθρο της επίλυσης καθώς θα γίνει και η ανάλυση μερικών ενδιάμεσων αποτελεσμάτων και προβλημάτων που αντιμετωπίστηκαν. Για κάθε εικόνα υπάρχει και ένα αρχείο με κώδικα για την κάθε λύση(mainN2.py και main2.py)

Τα ζητούμενα της Εργασίας ήταν για τις δύο παρακάτω εικόνες (N2.png, NF2.png) με κύτταρα να γράψω έναν αλγόριθμο για να γίνει

- Μέτρηση αριθμού αντικειμένων (κυττάρων) στο χώρο εικόνας
- Μέτρηση επιφάνειας (ως αριθμός εικονοστοιχείων) κάθε αντικειμένου
- Μέτρηση της μέσης τιμής διαβάθμισης του γκρι των εικονοστοιχείων που περιέχονται στα περιβάλλοντα κουτιά (bounding boxes) των αντικειμένων με τέτοιο τρόπο ώστε η ταχύτητα εκτέλεσης υπολογισμού να είναι ανεξάρτητη του μεγέθους του αντικειμένου.



Εικόνα 1α: N2.png



Εικόνα 1b: NF2.png

ΠΕΡΙΕΧΟΜΕΝΑ

1.Δημιουργία Φίλτρου για την αφαίρεση θορύβου

1.1 Ανάλυση του θορύβου και της επιλογής του φίλτρου

1.2 Υλοποίηση του Φίλτρου

2. Μέτρηση Κυττάρων στην εικόνα

2.1 Δημιουργία Binary Image – Thresholding

2.2 Μορφολογικοί Μετασχηματισμοί στην εικόνα

2.3 Μέτρηση αριθμών κυττάρων

3. Μέτρηση επιφάνειας για κάθε κύτταρο

4. Υπολογισμός της μέσης τιμής διαβάθμισης του γκρι για κάθε Bounding Box σε σταθερό χρόνο (constant time)

5. Μεθοδολογία και αποτελέσματα για την NF2.png (εικόνα χωρίς θόρυβο).

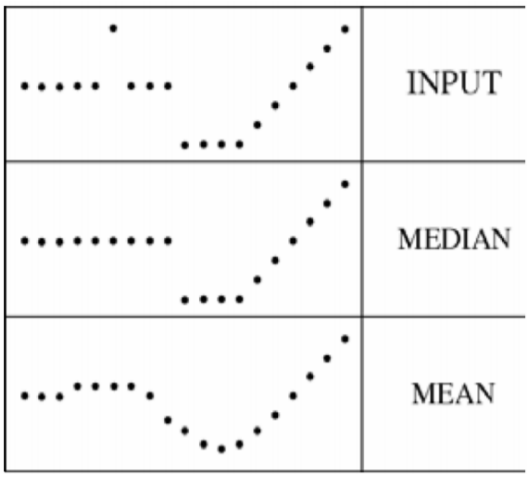
6.Πηγές Αναφορές.

1. Δημιουργία Φίλτρου για την αφαίρεση θορύβου

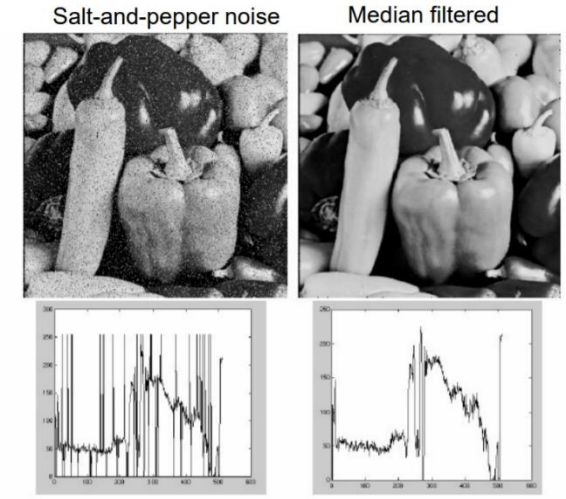
1.1 Ανάλυση του θορύβου και της επιλογής του φίλτρου

Στο πρώτο στάδιο της εργασίας, πρέπει να επιλέξουμε ένα φίλτρο για να αφαιρέσουμε τον θόρυβο της εικόνας N2.png (Εικόνα 1α). Είναι σημαντικό πρώτα να αναγνωρίσουμε το είδος και τα χαρακτηριστικά του θορύβου για να κάνουμε σωστή επιλογή φίλτρου. Στη συγκεκριμένη εικόνα έχουμε Salt-and-Pepper Noise. Ο θόρυβος αυτός, γνωστός και ως impulse noise , παρουσιάζεται σαν αραιά τοποθετημένα λευκά και μαύρα pixel (τιμές 255 και 0 αντίστοιχα σε ένα κανάλι διαβάθμισης του γκρι στην αναπαράσταση της εικόνας σε πίνακα τιμών) στην εικόνα μας. [1]

Όπως βλέπουμε έχουμε στην εικόνα μας πολλές ακραίες τιμές (outliers) συγκριτικά με τις τιμές των στοιχείων που έχουμε οπότε πρέπει να διαλέξουμε ένα φίλτρο που να είναι ανθεκτικό στα outliers αφαιρώντας το θόρυβο και διατηρώντας όσο καλύτερα γίνεται τις ακμές. Μια πολύ καλή επιλογή για αυτό το πρόβλημα είναι το Median filter(Εικόνα 2).

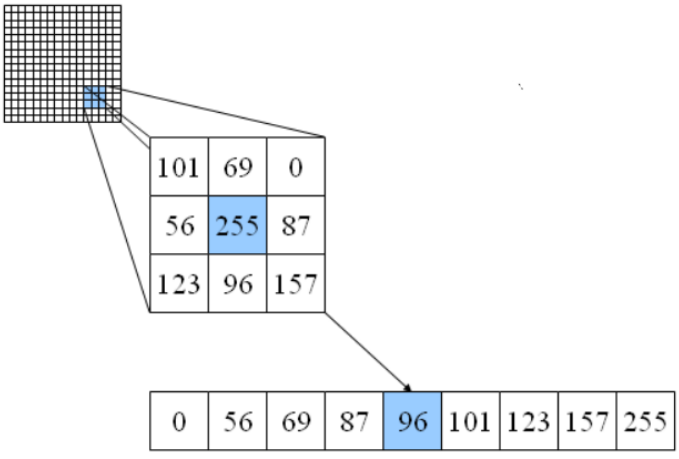


Εικόνα 2α - Σύγκριση εισόδου με outlier με το αποτέλεσμα φιλτραρίσματος με χρήση Median και Mean[2]



Εικόνα 2b - Βλέπουμε στο παράδειγμα αυτό πως αφαιρεί τον θόρυβο και δεν επηρεάζεται από το θόρυβο αυτό[2]

Στο median filter η τιμή του κάθε pixel είναι η μεσαία τιμή των pixel ενός παραθύρου γύρω από το pixel(Εικόνα 3)



Εικόνα 3 - Το pixel με την τιμή 255 θα πάρει τιμή 96[3]

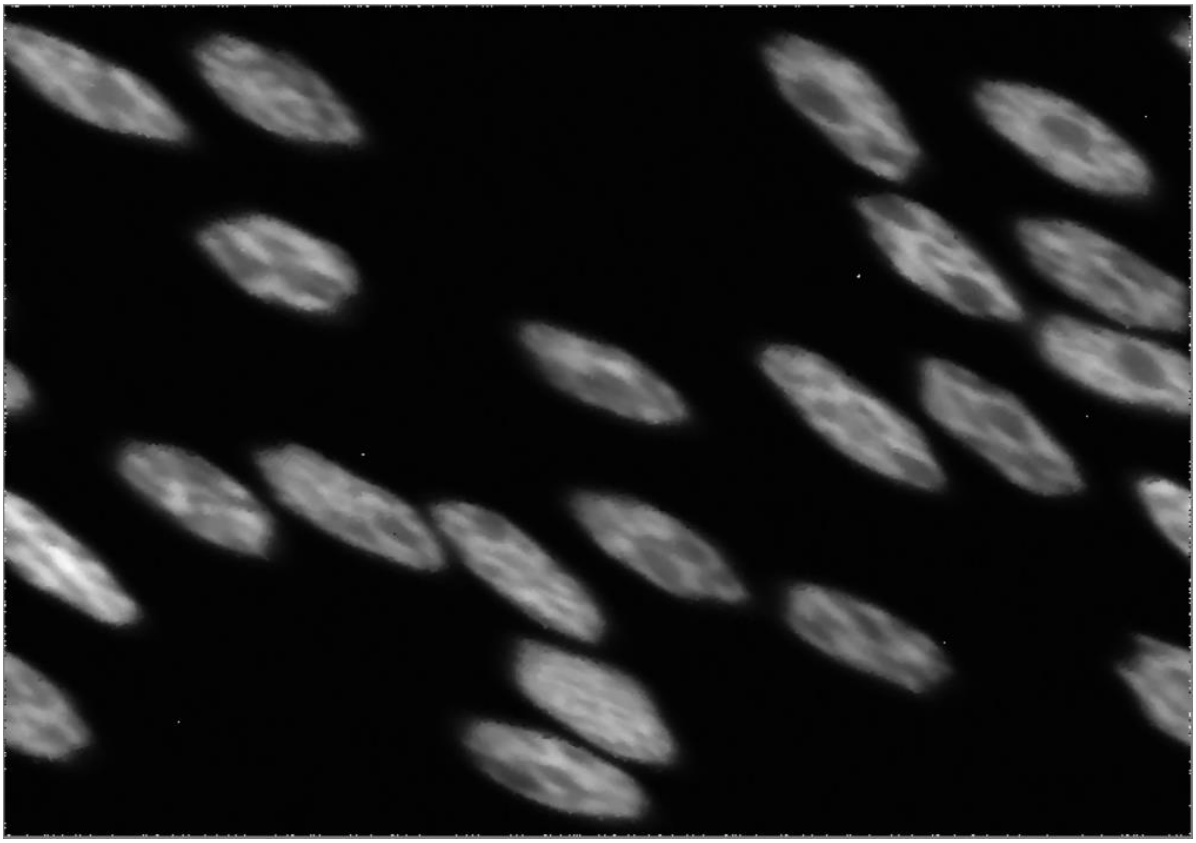
1.2 Υλοποίηση του φίλτρου

Περνώντας στην υλοποίηση με κώδικα του παραπάνω φίλτρου θα δημιουργήσουμε μια διπλή δομή επανάληψης ώστε να «περάσουμε» από κάθε pixel της εικόνας βάζοντας σε ένα πίνακα τις τιμές του pixel και των pixel γύρω από αυτό δημιουργώντας ουσιαστικά το παράθυρο για κάθε pixel. Στη συνέχεια τιμές του πίνακα αυτού τις κάνουμε sort και επιλέγουμε την μεσαία θέση του πίνακα για να πάρουμε το median.

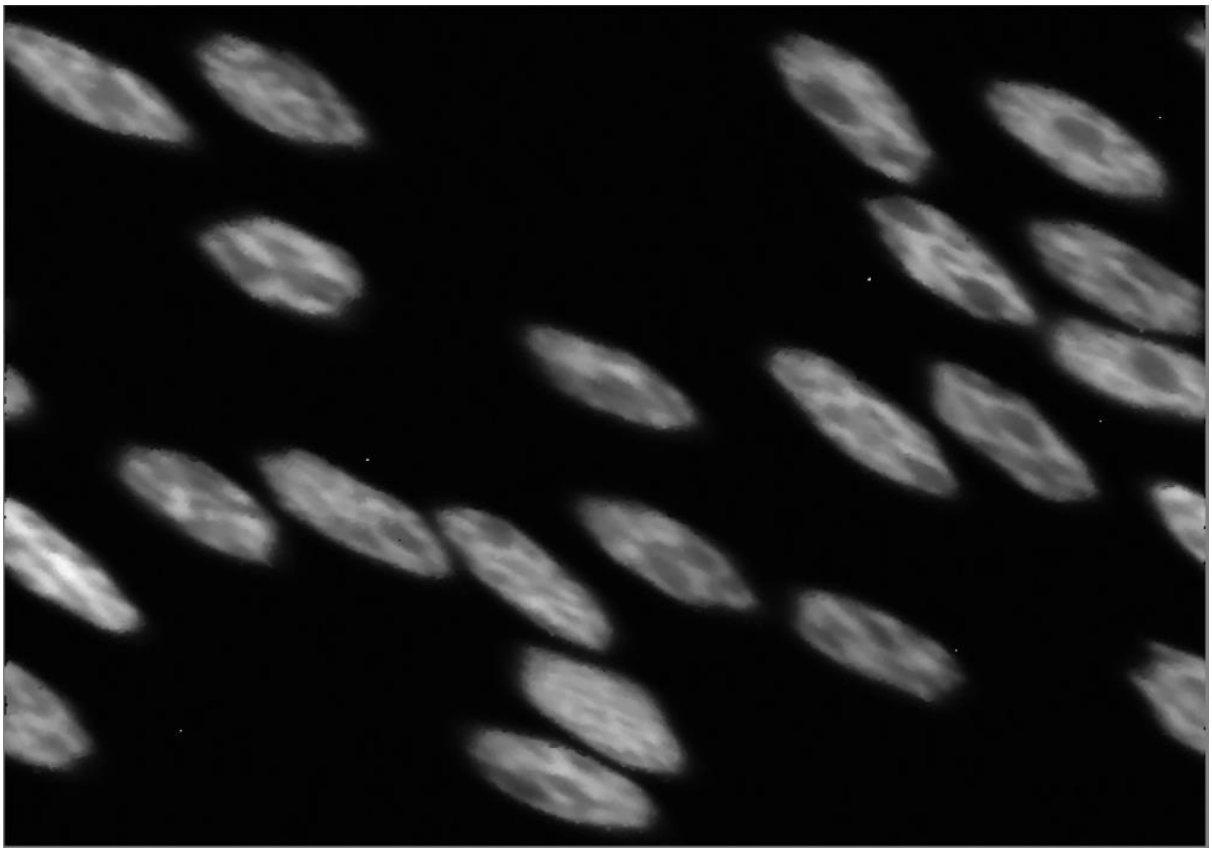
Στην υλοποίηση πρέπει να προσέξουμε να τοποθετούμε την τιμή που υπολογίσαμε για κάθε pixel σε έναν νέο πίνακα και να μην αλλάζουμε τιμές στον παλιό καθώς θα έχουμε λάθος αποτελέσματα.

Επίσης θα πρέπει να αποφασίσουμε το μέγεθος του παραθύρου (kernel) καθώς και πως θα διαχειριστούμε τα όρια της εικόνας (boundaries) καθώς εκεί το kernel βγαίνει εκτός εικόνας . Αναφορικά με το παράθυρο αποφάσισα να είναι 3x3(εφόσον μου αφαιρεί τον θόρυβο) ώστε να έχουμε το ελάχιστο blurring στην εικόνα για να διατηρήσω περισσότερη πληροφορία. Αναφορικά με τα όρια μπορούμε να[4]:

- Αποφύγουμε να υπολογίσουμε τα όρια αφαιρώντας ή μη τα pixel αυτά από την εικόνα(Εικόνα 4)
- Να κάνουμε padding στην εικόνα κατά 1 pixel σε κάθε πλευρά ώστε το 3χ3 παράθυρο να μπορεί να υπολογίσει για τα εσωτερικά pixel και στη συνέχεια να αφαιρέσουμε το pad αυτό έχοντας πλέον το μέγεθος της αρχικής εικόνας(Εικόνα 5)



Εικόνα 4- Το αποτέλεσμα αν δεν υπολογίσουμε τα border



Εικόνα 5- Το αποτέλεσμα αν κάνω padding κατά 1 pixel σε κάθε πλευρά με τιμή 0 (Μαύρο) και μετά κόψω τα pixel που πρόσθεσα

Επειδή θέλουμε να έχουμε πληροφορία στα όρια (πχ για να δούμε αν ένα κύτταρο ακουμπάει στα όρια) υλοποιήθηκε ο δεύτερος τρόπος (Εικόνα 5)

Συνολικά για τα παραπάνω:

Φορτώνουμε την εικόνα σε greyscale και φτιάνουμε το padding προσθέτοντας 1 pixel σε κάθε πλευρά με τιμή 0 (Μαύρο)

```
4.  source = cv2.imread("N2.png", cv2.IMREAD_GRAYSCALE)
5.  dst = cv2.copyMakeBorder(source, 1, 1, 1, 1, cv2.BORDER_CONSTANT, None, 0)
```

Δημιουργούμε την εικόνα που θα μπουν τα νέα pixel καθώς και το παράθυρο kernel

```
7.  final = dst[:]
8.  window = [0] * 9
```

Υλοποιούμε την for όπως περιγράψαμε από πάνω ξεκινώντας μια γραμμή και μία στήλη πιο μέσα στην αρχή και τελειώνοντας 1 γραμμή και 1 στήλη πριν την τελευταία καθώς προστέθηκε σε κάθε πλευρά 1 pixel και δεν χρειάζεται να υπολογίσουμε κάτι εκεί είναι εκτός εικόνας

```
9.  for y in range(1, dst.shape[0] - 1):
10.     for x in range(1, dst.shape[1] - 1):
11.         window[0] = dst[y - 1, x - 1]
12.         window[1] = dst[y, x - 1]
13.         window[2] = dst[y + 1, x - 1]
14.         window[3] = dst[y - 1, x]
15.         window[4] = dst[y, x]
16.         window[5] = dst[y + 1, x]
17.         window[6] = dst[y - 1, x + 1]
18.         window[7] = dst[y, x + 1]
19.         window[8] = dst[y + 1, x + 1]
20.         window.sort()
21.         final[y, x] = window[4]
```

Τέλος αφαιρούμε το 1 pixel padding σε κάθε πλευρά που προσθέσαμε για να έχουμε την τελική εικόνα που είδαμε στην εικόνα 5.

```
22. final = final[1:final.shape[0] - 1, 1:final.shape[1] - 1]
```

Εδώ να τονίσουμε ότι μερικά μικρά άσπρα που βλέπουμε στην εικόνα θα αφαιρεθούν στην συνέχεια στους μορφολογικούς μετασχηματισμούς και δεν αποτελούν πρόβλημα.

2. Μέτρηση Κυττάρων στην εικόνα

2.1 Δημιουργία Binary Image - Thresholding

Για να μπορέσουμε να μετρήσουμε τα κύτταρα καθώς και τα χαρακτηριστικά τους θα πρέπει να μετατρέψουμε την εικόνα σε δυαδική (Binary Image) και να κάνουμε κάποιους μετασχηματισμούς για να μπορέσουμε να βγάλουμε σωστά τα contours (στη συνέχεια θα αναφερθούμε σε αυτά.

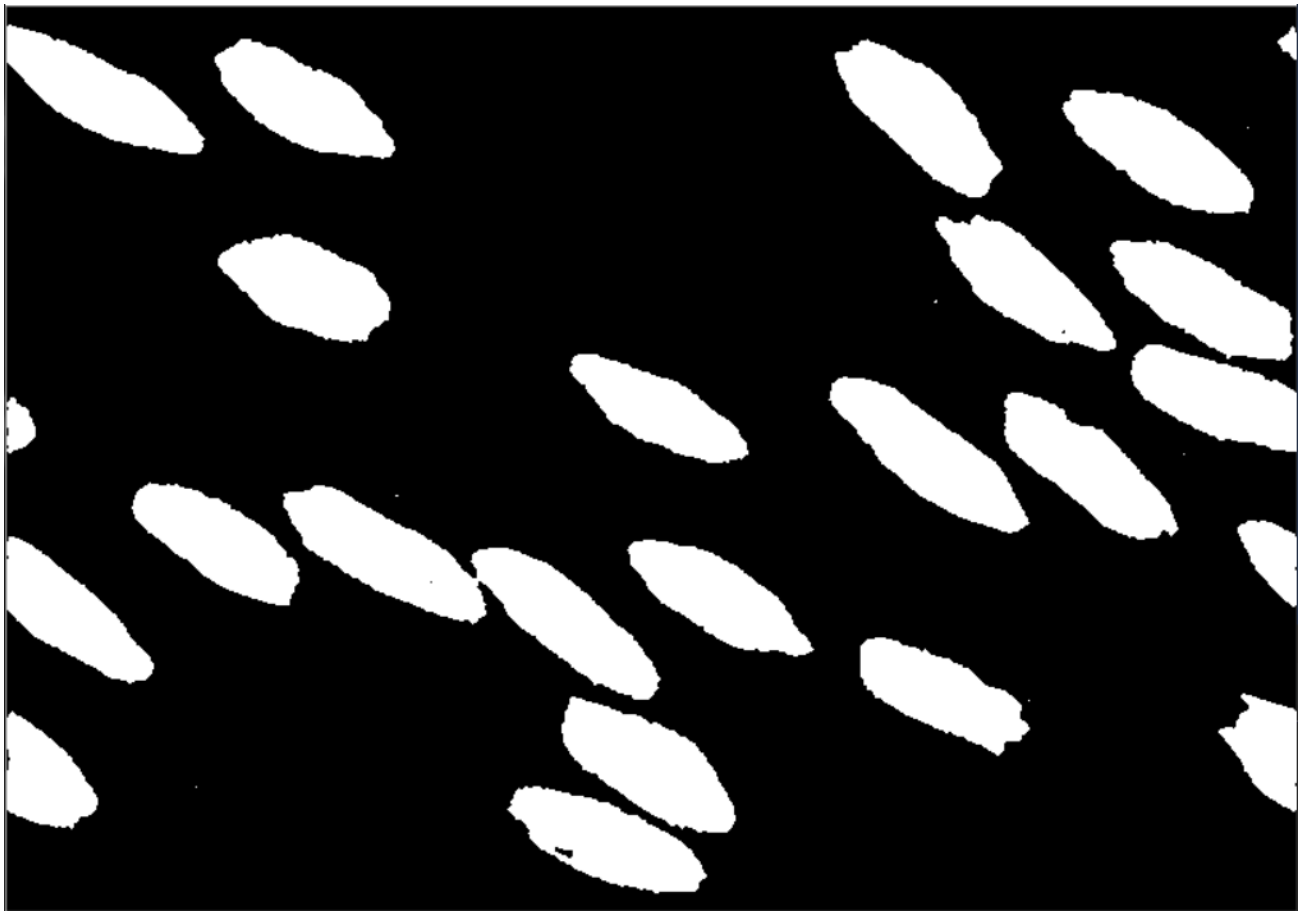
Για να μετατρέψουμε μια greyscale εικόνα σε δυαδική διαλέγουμε μια τιμή του γκρι (threshold) όπου αν κάποιο pixel έχει τιμή μεγαλύτερη από αυτή θα γίνει τελείως λευκό (255) αν έχει μικρότερη θα γίνει τελείως μαύρο αντίστοιχα (0)

Μπορούμε στην OpenCv να καλέσουμε μια συνάρτηση που να δώσουμε το threshold και να μετατραπεί η εικόνα, Στην περίπτωση της πρώτης εικόνας διάλεξα την τιμή

57 έπειτα από διαφορετικές δοκιμές με σκοπό να μην μικραίνουν τα κύτταρα πολύ και να μην έχουν μεγάλες «τρύπες» αλλά και ταυτόχρονα να μην είναι πολύ ενωμένα.Ο Κώδικας

```
1.  thresh, final_Binary= cv2.threshold(final, 57, 255, cv2.THRESH_BINARY)
```

Και εμφανίζουμε το αποτέλεσμα στην εικόνα 6.



Εικόνα 6- Η Binary Image

2.2 Μορφολογικοί Μετασχηματισμοί στην εικόνα

Παρατηρούμε ότι έχουμε 2 κύτταρα ενωμένα στην εικόνα (κάτω αριστερά) ,έχουμε ακόμα λίγο άσπρο θόρυβο και μερικές τρύπες μέσα στα κύτταρα. Για να μπορέσουμε να τα αντιμετωπίσουμε αυτά θα κάνουμε τους εξής μορφολογικούς μετασχηματισμούς στην εικόνα

- Opening
- Closing

Opening ουσιαστικά είναι η ακολουθία μετασχηματισμών erosion και στην συνέχεια dilation στην εικόνα μας.

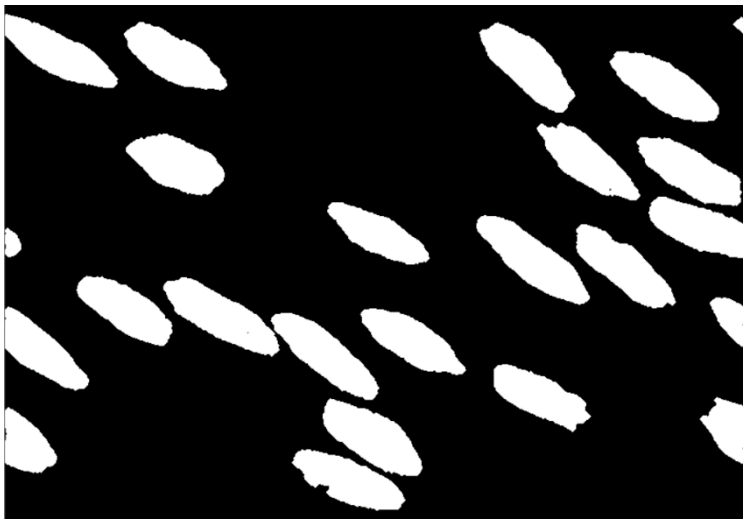
Closing είναι η αντίθετη διαδικασία, δηλαδή dilation και έπειτα erosion

Στο erosion, Ένα kernel που ονομάζεται structuring element (μπορεί να έχει διαφορετικά σχήματα) κάνει slide την εικόνα όπως στα φίλτρα και για κάθε pixel κάνει την τιμή του 255 μόνο αν όλα τα pixel στο structuring element έχουν τιμή αλλιώς την κάνει 0 .Πρακτικά το αποτέλεσμα μιας τέτοιας πράξης είναι να έχουμε «διάβρωση» των αντικειμένων. Και εξαφάνιση λευκών κουκίδων.

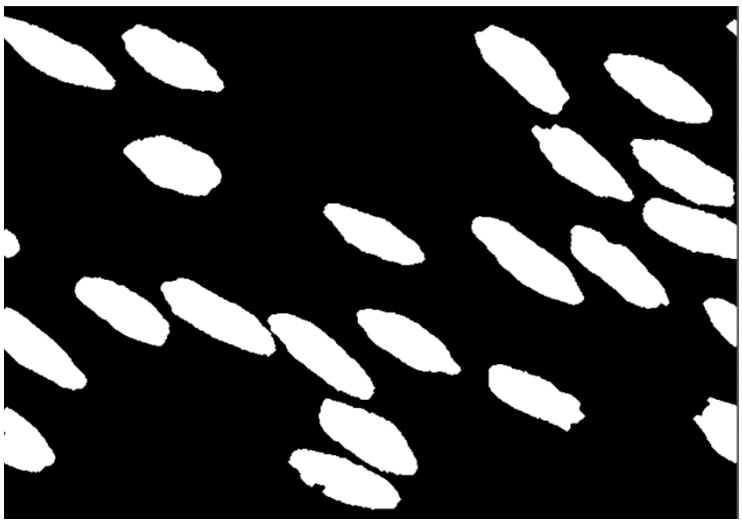
Στο dilation, η διαδικασία είναι ίδια μόνο που αυτή τη φορά το pixel θα γίνει λευκό αν έστω 1 από τα γύρω pixel στο structuring element είναι λευκό. [5]

Έτσι κάνοντας opening «σπάμε» τα δύο κύτταρα που είναι ενωμένα χωρίς να μικρύνουν ουσιαστικά τα κύτταρα μας και με το closing κλείνουμε τις τρύπες στα κύτταρα μας χωρίς να αλλάξουμε το μέγεθος τους. Στον κώδικα επέλεξα τον structuring element ως σταυρό 3χ3 καθώς μετα από πολλές δοκιμές μου έδινε τα καλύτερα αποτελέσματα, και έκανα 3 iteration το opening πριν το closing γιατί μόνο τότε έσπαγε τα κύτταρα:

```
32.kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
33.final_transformed = cv2.morphologyEx(final_Binary, cv2.MORPH_OPEN, kernel
,iterations=3)
34. final_transformed = cv2.morphologyEx(final_transformed, cv2.MORPH_CLOSE,
    kernel,iterations=1)
```



Εικόνα 7- Μετά το opening



Εικόνα 8 - Μετά το opening και closing

Εδώ να σημειώσω ότι στο κάτω κύτταρο στο κέντρο έχει φαγωθεί ένα κομμάτι του, ο λόγος που συνέβη αυτό είναι επειδή έκανα πρώτα opening και μετά closing , με αποτέλεσμα πρώτα η ακριανή τρύπα να ανοίξει, Ο λόγος που το άφησα έτσι είναι καθώς όταν προσπαθούσα να κάνω πρώτα closing και μετά opening το κύτταρα αυτό ήταν ολόκληρο σωστά αλλά η ένωση των δύο κυττάρων γινόταν πολύ μεγάλη με αποτέλεσμα για να αφαιρεθεί να έπρεπε να μειώσω αρκετά όλα τα κύτταρα, οπότε αποφάσισα να κρατήσω την σειρά αυτή για να έχω πιο σωστά μεγαλύτερο αριθμό κυττάρων

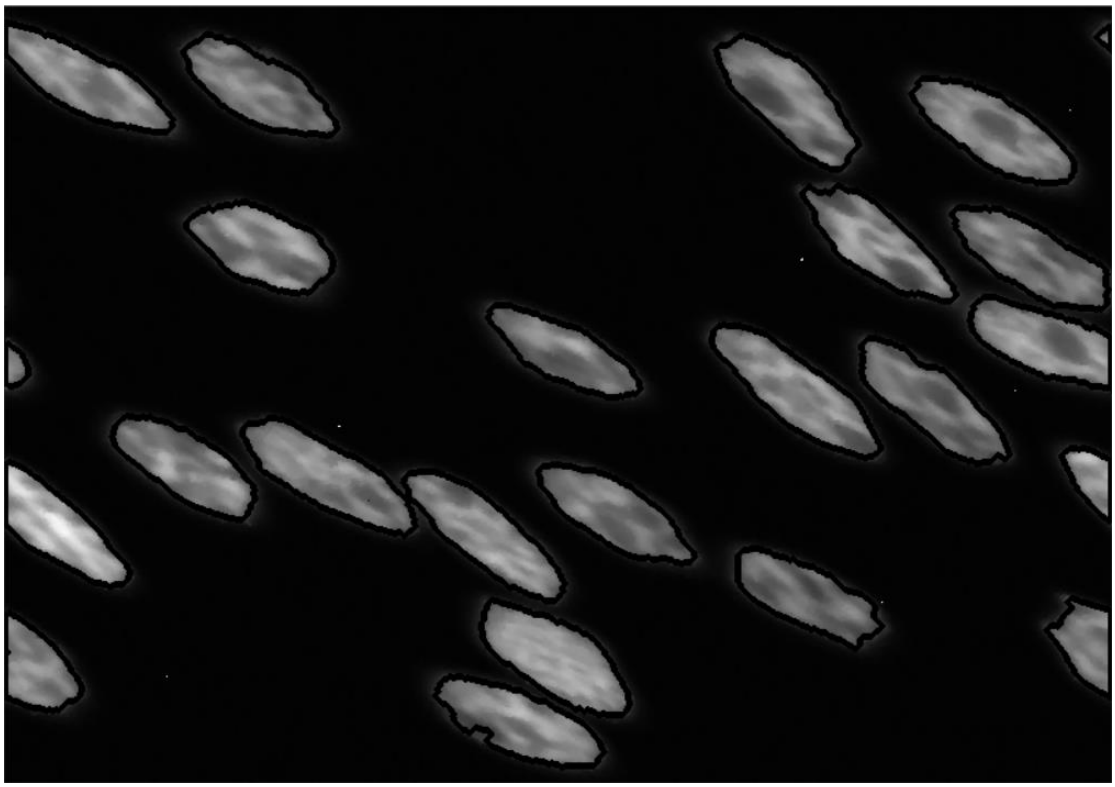
2.3 Μέτρηση αριθμών κυττάρων

Έχοντας στην εικόνα μας διαχωρίσει τα κύτταρα και γεμίσει τα εσωτερικά κενά μπορούμε να μετρήσουμε πλέον τα κύτταρα στην εικόνα μας (τα συνολικά και αυτά που δεν ακουμπάν σε κάποιο border της εικόνας). Αυτό μπορούμε να το κάνουμε μετρώντας τον αριθμό από συνεχόμενες περιοχές λευκών pixel. Η καμπύλη που ενώνει μια τέτοια περιοχή ονομάζεται Contour[6]

Έχοντας ετοιμάσει σωστά την εικόνα μας μπορούμε να με τη χρήση συνάρτηση της opencv να βρούμε όλα τα contour της εικόνας μας(οι παράμετροι αναφέρονται στην ιεραρχία των contour αν π.χ. είχαμε contour μέσα σε contour και στο τρόπο που αποθηκεύονται τα contour και στο πλαίσιο της εργασίας δεν χρειάζεται να τα χρησιμοποιήσουμε και η συνάρτηση επιστρέφει 3 πράγματα μια modified εικόνα τα contour και την ιεραρχία τους)

```
1.  _, contours, _ = cv2.findContours(final_transformed, cv2.RETR_LIST, cv2.CHAI
    N_APPROX_NONE)
```

Έτσι έχουμε έναν πίνακα με contours. Το μέγεθος αυτού του πίνακα μας δίνει τον αριθμό των κυττάρων που υπάρχουν στην εικόνα μαζί με αυτά που βρίσκονται σε border. Μπορούμε επίσης να τα σχεδιάσουμε πάνω στην εικόνα.(Εικόνα 9)



Εικόνα 9 Τα Contour στην De-noised εικόνα

Για να αφαιρέσουμε τα κύτταρα που είναι στα όρια έφτιαξα μια συνάρτηση που παίρνει ένα όρισμα contour, για αυτό παίρνω τις συντεταγμένες του bounding box του δηλαδή το σημείο πάνω αριστερά του και το ύψος και πλάτος του. Στη συνέχεια ελέγχω για αυτά τα σημεία αν είναι μεγαλύτερα ή στα όρια της εικόνας μου, αν είναι επιστρέφω true δηλαδή είναι στο όριο αλλιώς false δηλαδή δεν είναι στο όριο. Ελέγχω μετά με αυτή τη συνάρτηση το κάθε contour και αν είναι στο όριο το αφαιρώ από το πίνακα των contour.

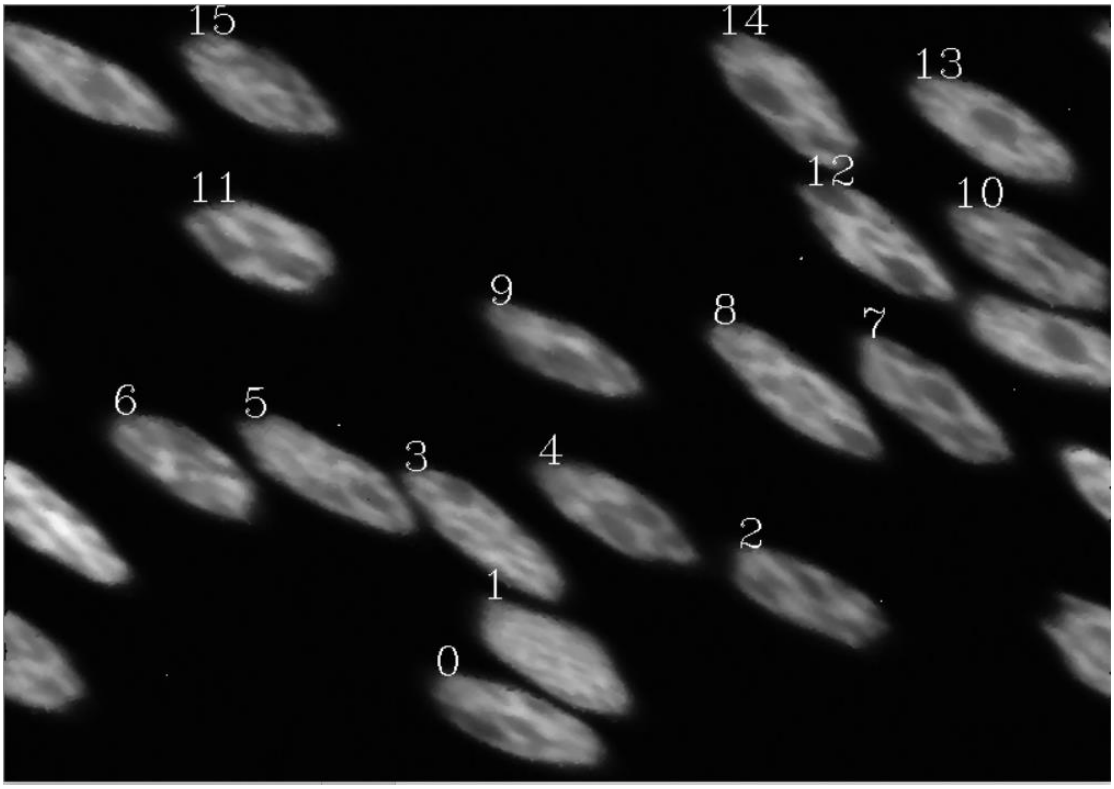
```
51.  def isContourAtBorder(contour,image):
52.      x, y, w, h = cv2.boundingRect(contour)
53.      xMin = 0
54.      yMin = 0
55.      xMax = image.shape[1]-1
56.      yMax = image.shape[0]-1
57.      if x <= xMin or y <= yMin or x+w >= xMax or y+h >= yMax:
58.          return True
59.      else:
60.          return False
61.
62.
63.  #find the bordering contours
64.  badconindex = np.array([])
65.  i=0
66.  for c in contours:
67.      if isContourAtBorder(c,final_transformed):
68.          badconindex = np.append(badconindex,i)
69.      i = i + 1
70.  contours = np.delete(contours, badconindex)
```

Άρα ο αριθμός των κυττάρων είναι για κάθε περίπτωση :

```
Number of Cells in the image including those touching the borders 24
Number of Cells in the image without those touching the borders 16
```

Εικόνα 10 Αριθμός κυττάρων τυπωμένος στην κονσόλα

Τέλος για να συνεχίσουμε στα επόμενα καλό είναι να τα αριθμήσουμε και να τα δούμε αριθμημένα πάνω στην εικόνα για να ξέρουμε για ποιο κύτταρο παίρνουμε τιμές



Εικόνα 11- Τα αριθμημένα Κύτταρα

3.Μέτρηση επιφάνειας για κάθε κύτταρο.

Για να μετρήσουμε την επιφάνεια του κάθε κυττάρου υλοποιήθηκαν 2 τρόποι:

Στον πρώτο, χρησιμοποιήθηκε η έτοιμη συνάρτηση contourArea(contour) της OpenCV και στον δεύτερο υλοποιήθηκε μια δικιά μου συνάρτηση για την μέτρηση της επιφάνειας. Ο λόγος που έγινε αυτό είναι γιατί η OpenCV για να υπολογίσει την επιφάνεια σε pixel χρησιμοποιεί την καμπύλη που είναι το contour και χρησιμοποιώντας το Green's Formula υπολογίζει το εμβαδόν σε pixel. Αυτό έχει ως αποτέλεσμα η επιφάνεια να μπορεί να είναι δεκαδικό νούμερο (στο αποτέλεσμα βλέπουμε και .5) και το αποτέλεσμα να διαφέρει από το να μετρήσουμε τα λευκά pixel εντός ενός contour σύμφωνα με το documentation της συνάρτησης αυτής [9]

Έτσι υλοποίησα και εγώ μια συνάρτηση για την μέτρηση της επιφάνειας που λειτουργεί με τον εξής τρόπο:

1.Για κάθε contour παίρνω τις συντεταγμένες του bounding box του

2.Ελέγχω για κάθε pixel εντός του bounding box του αν ανήκει στο εσωτερικό της καμπύλης ή και στο όριο του contour (ο λόγος που το κάνω αυτό είναι καθώς μπορεί να έχω overlapping των bounding box και ένα bounding box να πιάνει λευκό και από άλλο αντικείμενο

3.Αν ναι, ελέγχω αν το pixel είναι λευκό , αν είναι αυξάνω το μέγεθος κατά 1 αν όχι το αφήνω ίδιο

Η συνάρτηση :

```
78. def countAreaContour(contour):
79. x, y, w, h = cv2.boundingRect(contour)
80. size=0
81. for y1 in range(y,y+h):
82.     for x1 in range(x,x+w):
83.        if cv2.pointPolygonTest(contour, (x1,y1),False)>= 0:
84.             if final_transformed[y1][x1]==255:
85.                 size=size+1
86. return size
```

Εκτυπώνω στην κονσόλα για κάθε κύτταρο το μέγεθος του σε pixel (Εικόνα 12)

```
Area of cell number (using ccntourArea)  0  is 4389.5
Area of cell number   (using my method)  0  is 4528
Area of cell number (using ccntourArea)  1  is 4905.0
Area of cell number   (using my method)  1  is 5037
Area of cell number (using ccntourArea)  2  is 3934.5
Area of cell number   (using my method)  2  is 4058
Area of cell number (using ccntourArea)  3  is 4637.5
Area of cell number   (using my method)  3  is 4771
Area of cell number (using ccntourArea)  4  is 4116.5
Area of cell number   (using my method)  4  is 4243
Area of cell number (using ccntourArea)  5  is 4940.5
Area of cell number   (using my method)  5  is 5083
Area of cell number (using ccntourArea)  6  is 4157.0
Area of cell number   (using my method)  6  is 4276
Area of cell number (using ccntourArea)  7  is 4536.5
Area of cell number   (using my method)  7  is 4666
Area of cell number (using ccntourArea)  8  is 5029.0
Area of cell number   (using my method)  8  is 5170
Area of cell number (using ccntourArea)  9  is 3630.0
Area of cell number   (using my method)  9  is 3752
Area of cell number (using ccntourArea)  10  is 4570.5
Area of cell number   (using my method)  10  is 4704
Area of cell number (using ccntourArea)  11  is 4476.5
Area of cell number   (using my method)  11  is 4595
Area of cell number (using ccntourArea)  12  is 4285.5
Area of cell number   (using my method)  12  is 4414
Area of cell number (using ccntourArea)  13  is 4786.5
Area of cell number   (using my method)  13  is 4918
Area of cell number (using ccntourArea)  14  is 4880.5
Area of cell number   (using my method)  14  is 5006
Area of cell number (using ccntourArea)  15  is 4407.5
Area of cell number   (using my method)  15  is 4534
```

Εικόνα 12 - Το μέγεθος του κάθε κυττάρου με τις 2 μεθόδους

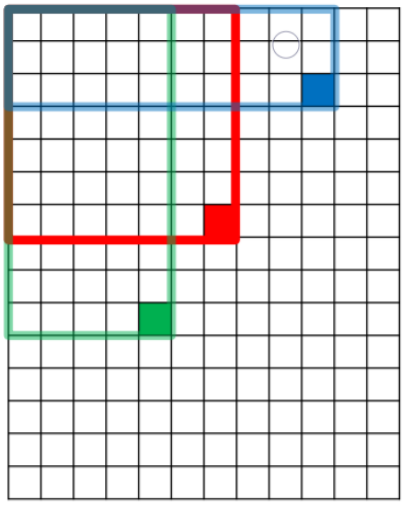
4. Υπολογισμός της μέσης τιμής διαβάθμισης του γκρι για κάθε Bounding Box σε σταθερό χρόνο (constant time)

Στο τελευταίο ερώτημα όπως είδαμε πρέπει να βρούμε την μέση τιμή του γκρί σε κάθε bounding box. Πρακτικά αυτό είναι το άθροισμα των τιμών του pixel σε ένα bounding box προς τον αριθμό των pixel στο bounding box.

Όμως θα πρέπει η υλοποίηση μας για κάθε box να είναι ανεξάρτητη του μεγέθους του (ουσιαστικά να υπολογίζεται σε πολυπλοκότητα O(1)). Για να το κάνουμε αυτό θα πρέπει να υπολογίσουμε από πριν την Integral Image (γνωστή και ως sum up table) της greyscale φιλτραρισμένης εικόνας μας.

Στην integral image το κάθε pixel είναι το άθροισμα των τιμών των pixel του παραλληλογράμμου πάνω και αριστερά του pixel (Μπορεί να υπολογιστεί σαν το άθροισμα των άνω και αριστερών γειτόνων του μαζί με τον εαυτό του(Εικόνα 13 [7])

$$I(x,y) = I(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1)$$

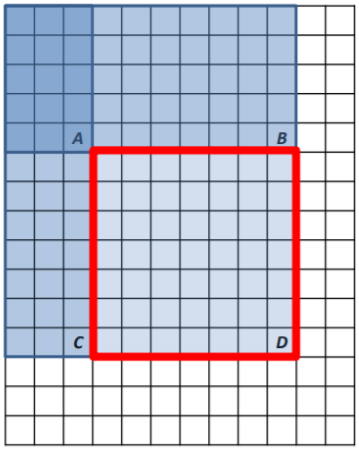


Εικόνα 13[7]

Έτσι αν θέλουμε το άθροισμα των pixel ενός παραλληλογράμμου στην εικόνα μας(bounding box εδώ) Μπορούμε με 1 πράξη σε σταθερό χρόνο να το υπολογίσουμε όπως φαίνεται στην εικόνα 14[7].

Solution is found using:

$$A + D - B - C$$



Εικόνα 14[7]

Στην υλοποίηση της OpenCv υπάρχει συνάρτηση που επιστρέφει την integral image . **Εδώ θέλει προσοχή καθώς η εικόνα που επιστρέφει η OpenCv έχει padding με 0 στην πάνω και αριστερή πλευρά.** Οπότε όταν παίρνουμε τις συντεταγμένες του bounding box για κάθε contour θα πρέπει να κάνουμε σωστή αντιστοίχιση των σημείων A B C και D στην integral image (ουσιαστικά +1 σε κάποιες συντεταγμένες όπως φάινεται παρακάτω).[8] Ο αντίστοιχος κώδικας:

```
integral_img=cv2.integral(final,cv2.CV_64F)
i=0
for c in contours:
    x, y, w, h = cv2.boundingRect(c)
    sumOfPixels=integral_img[y][x] + integral_img[y+h][x+w] - integral_img[y][x+w] - integral_img[y+h][x]
    meanGrayValue = sumOfPixels/(h*w)
    print("Mean Gray Value of bounding box of  Cell number ",i, "is ", meanGrayValue)
    i=i+1
cv2.waitKey(0)
```

Και τα αποτελέσματα στην παρακάτω εικόνα:

```
Mean Gray Value of bounding box of  Cell number  0 is  87.40823880597014
Mean Gray Value of bounding box of  Cell number  1 is  92.94268753372909
Mean Gray Value of bounding box of  Cell number  2 is  54.34982151963284
Mean Gray Value of bounding box of  Cell number  3 is  62.78080495356037
Mean Gray Value of bounding box of  Cell number  4 is  58.18020784128484
Mean Gray Value of bounding box of  Cell number  5 is  62.73698977283012
Mean Gray Value of bounding box of  Cell number  6 is  68.70674087816946
Mean Gray Value of bounding box of  Cell number  7 is  57.82658556043079
Mean Gray Value of bounding box of  Cell number  8 is  60.45901909150757
Mean Gray Value of bounding box of  Cell number  9 is  56.56814600840336
Mean Gray Value of bounding box of  Cell number  10 is  65.55188405797101
Mean Gray Value of bounding box of  Cell number  11 is  79.0430283224401
Mean Gray Value of bounding box of  Cell number  12 is  59.64456012493493
Mean Gray Value of bounding box of  Cell number  13 is  65.88593073593074
Mean Gray Value of bounding box of  Cell number  14 is  56.208273208273205
Mean Gray Value of bounding box of  Cell number  15 is  62.50442477876106
```

Εικόνα 15

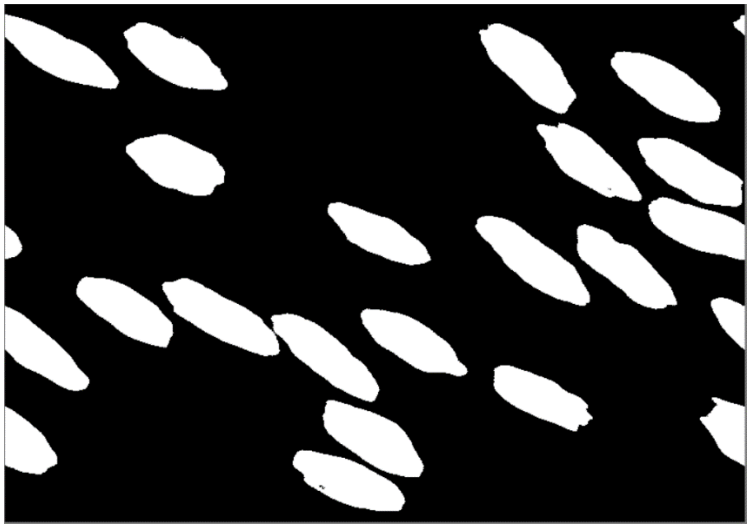
5.Μεθοδολογία και αποτελέσματα για την NF2.png (εικόνα χωρίς θόρυβο).

Έχοντας τελειώσει με την πρώτη εικόνα η μεθοδολογία στην δεύτερη εικόνα που μας δόθηκε είναι ακριβώς η ίδια με μερικές μικρές διαφορές(αρχείο mainNF2.py), Πιο συγκεκριμένα δεν έγινε χρήση φίλτρου καθώς δεν έχουμε θόρυβο, η τιμή του thresholding είναι διαφορετική(53) επειδή χωρίς το φίλτρο φαίνεται να είναι πιο καθαρή η εικόνα και μπορούμε να έχουμε και διαφορετικούς μετασχηματισμούς έχοντας και τα κύτταρα με καλύτερα αποτελέσματα όπως βλέπουμε στις παρακάτω εικόνες και στον κώδικα

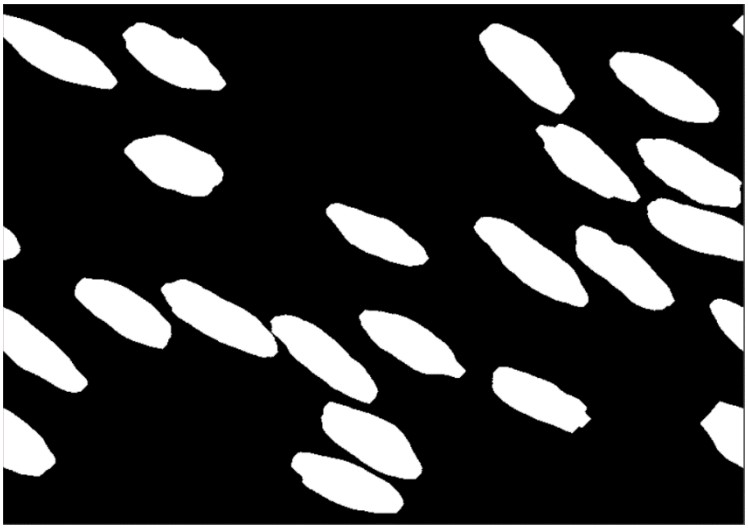
```
thresh, final_Binary= cv2.threshold(final, 53, 255, cv2.THRESH_BINARY)

cv2.imshow('Final_Picture1_Binary', final_Binary)  # Show the image
cv2.waitKey(0)

kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
final_transformed = cv2.morphologyEx(final_Binary, cv2.MORPH_CLOSE, kernel,iterations=1)
final_transformed = cv2.morphologyEx(final_transformed, cv2.MORPH_OPEN, kernel,iterations=5)
cv2.imshow('Final_Picture12', final_transformed)  # Show the image
cv2.waitKey(0)
```

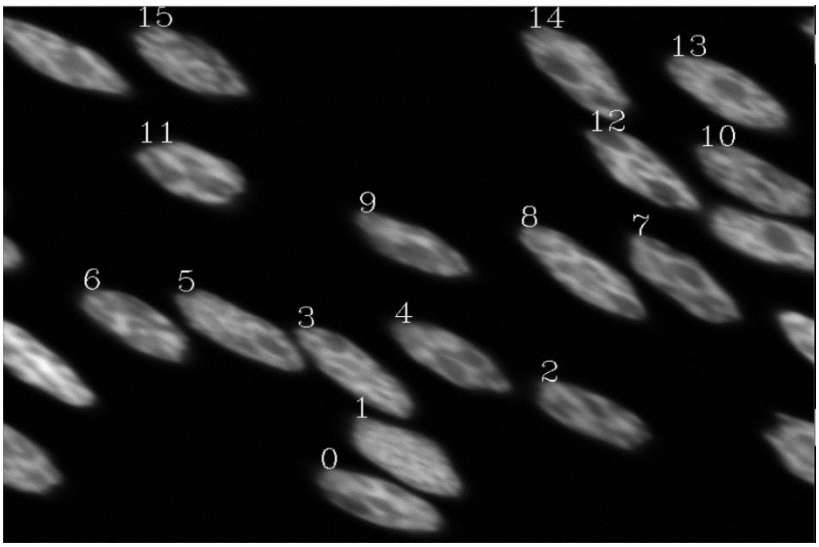


Εικόνα 16 NF2.png Binary with thresh value 53



Εικόνα 17 Μετά τους μετασχηματισμούς

Και τα αποτελέσματα για τα ίδια ερωτήματα:




```
Mean Gray Value of bounding box of  Cell number  0 is  89.28455284552845
Mean Gray Value of bounding box of  Cell number  1 is  93.02809952809953
Mean Gray Value of bounding box of  Cell number  2 is  54.950488683127574
Mean Gray Value of bounding box of  Cell number  3 is  62.267691770050675
Mean Gray Value of bounding box of  Cell number  4 is  57.74277726001864
Mean Gray Value of bounding box of  Cell number  5 is  64.04099264705883
Mean Gray Value of bounding box of  Cell number  6 is  68.46202225447509
Mean Gray Value of bounding box of  Cell number  7 is  57.7484070091597
Mean Gray Value of bounding box of  Cell number  8 is  59.91208163265306
Mean Gray Value of bounding box of  Cell number  9 is  55.95134575569358
Mean Gray Value of bounding box of  Cell number  10 is  65.69797101449275
Mean Gray Value of bounding box of  Cell number  11 is  80.1692924267551
Mean Gray Value of bounding box of  Cell number  12 is  58.90629156010230
Mean Gray Value of bounding box of  Cell number  13 is  65.26185897435897
Mean Gray Value of bounding box of  Cell number  14 is  56.64073426573427
Mean Gray Value of bounding box of  Cell number  15 is  63.2113130829945
```

```
Area of cell number (using ccntourArea)  0  is 4557.0
Area of cell number   (using my method)  0  is 4691
Area of cell number (using ccntourArea)  1  is 4952.0
Area of cell number   (using my method)  1  is 5083
Area of cell number (using ccntourArea)  2  is 3989.0
Area of cell number   (using my method)  2  is 4110
Area of cell number (using ccntourArea)  3  is 4698.5
Area of cell number   (using my method)  3  is 4831
Area of cell number (using ccntourArea)  4  is 4209.5
Area of cell number   (using my method)  4  is 4336
Area of cell number (using ccntourArea)  5  is 4970.0
Area of cell number   (using my method)  5  is 5111
Area of cell number (using ccntourArea)  6  is 4272.5
Area of cell number   (using my method)  6  is 4392
Area of cell number (using ccntourArea)  7  is 4599.0
Area of cell number   (using my method)  7  is 4726
Area of cell number (using ccntourArea)  8  is 5112.5
Area of cell number   (using my method)  8  is 5253
Area of cell number (using ccntourArea)  9  is 3702.5
Area of cell number   (using my method)  9  is 3823
Area of cell number (using ccntourArea)  10  is 4651.5
Area of cell number   (using my method)  10  is 4784
Area of cell number (using ccntourArea)  11  is 4540.5
Area of cell number   (using my method)  11  is 4659
Area of cell number (using ccntourArea)  12  is 4387.5
Area of cell number   (using my method)  12  is 4516
Area of cell number (using ccntourArea)  13  is 4868.5
Area of cell number   (using my method)  13  is 4999
Area of cell number (using ccntourArea)  14  is 4947.0
Area of cell number   (using my method)  14  is 5070
Area of cell number (using ccntourArea)  15  is 4495.0
Area of cell number   (using my method)  15  is 4619
```

Εικόνα 18 Τα αποτελέσματα για την NF2.png

6.ΠΗΓΕΣ-ΑΝΑΦΟΡΕΣ

[1] https://en.wikipedia.org/wiki/Salt-and-pepper_noise

[2] Διαφάνειες του μαθήματος στο eclass, 3ο μάθημα Διαφάνειες: 84,85

[3] https://www.southampton.ac.uk/~msn/book/new_demo/median/

[4] https://en.wikipedia.org/wiki/Median_filter Boundary Issues

[5] Διαφάνειες του μαθήματος στο eclass, 4ο μάθημα Διαφάνειες: 67,97

[6] http://aishack.in/tutorials/introduction-contours/

[7] Διαφάνειες του μαθήματος στο eclass, 4ο μάθημα Διαφάνειες: 46,48

[8] http://aishack.in/tutorials/integral-images-opencv/

[9]https://docs.opencv.org/trunk/d3/dc0/group__imgproc__shape.html#ga2c759ed9f497d4a618048a2f56dc97f1