Όραση Υπολογιστών

1η Εργασία

Περιεχόμενα

1.Εισαγωγή

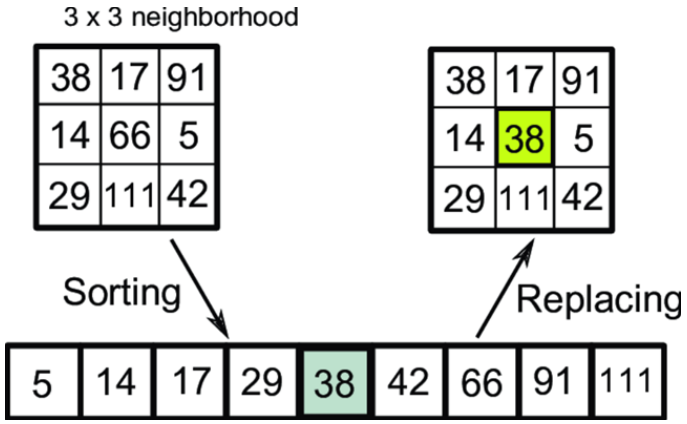
Στόχος της παρακάτω εργασίας είναι η ανάλυση εικόνων μικροσκοπίου με τεχνικές όρασης υπολογιστών. Πιο συγκεκριμένα, σκοπός είναι η ανίχνευση όλων των επιμέρους κυττάρων της εικόνας και η τοποθέτησή τους σε περιβάλλοντα κουτιά, αριθμημένα με ένα μοναδικό αύξοντα αριθμό. Επίσης, γίνεται ο υπολογισμός της επιφάνειας του κάθε κυττάρου και της επιφάνειας του περιβάλλοντος κουτιού του, ως αριθμός εικονοστοιχείων και της μέσης τιμής διαβάθμισης του γκρι των εικονοστοιχείων που περιέχονται στο κάθε περιβάλλον κουτί. Ο τελευταίος υπολογισμός ζητείται να υλοποιηθεί με τέτοιο τρόπο ώστε η ταχύτητα εκτέλεσης υπολογισμού να είναι ανεξάρτητη του μεγέθους της υποπεριοχής.

2.Αποθορυβοποίηση εικόνας

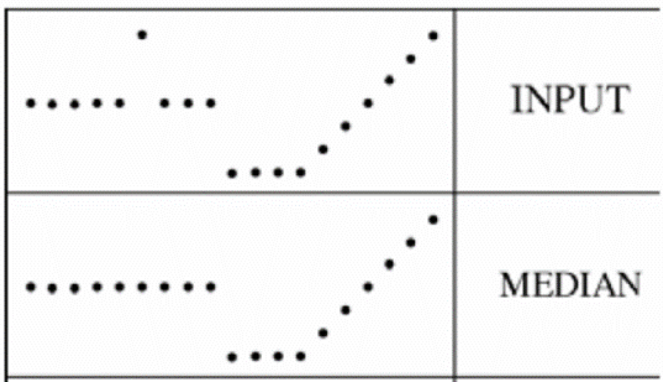
Η ανίχνευση των κυττάρων θα γίνει σε δύο εικόνες. Μία πρωτότυπη εικόνα και μία στην οποία έχει προστεθεί θόρυβος τύπου “αλατιού και πιπεριού”. Για την εικόνα στην οποία έχει προστεθεί θόρυβος είναι αναγκαία η απομάκρυνσή του, με τη χρήση κατάλληλου γραμμικού ή μη γραμμικού φίλτρου απόρριψης θορύβου. Για το σκοπό αυτό χρησιμοποιήθηκε το φίλτρο median, το οποίο είναι μη γραμμικό φίλτρο αποθορυβοποίησης. Το φίλτρο median είναι πολύ αποτελεσματικό στην απομάκρυνση θορύβου τύπου “αλατιού και πιπεριού”, ο οποίος υπάρχει στην εικόνα, για αυτό και χρησιμοποιήθηκε αυτή η μέθοδος.

Το φίλτρο median εφαρμόζεται σε μια εικόνα, μέσω ενός παραθύρου (kernel), το οποίο διατρέχει ολόκληρη την εικόνα, επιλέγει την ενδιάμεση ένταση στο παράθυρο και την τοποθετεί στην τιμή του κεντρικού εικονοστοιχείου του παραθύρου. Το παράθυρο μετακυλύει σε όλη την εικόνα, ώστε η τιμή όλων των εικονοστοιχείων να πάρει κάποια ενδιάμεση τιμή του εκάστοτε παραθύρου.

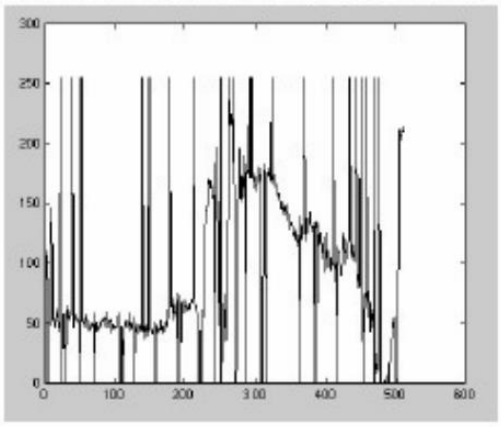
Στην παρακάτω εικόνα φαίνεται ο τρόπος υπολογισμού της ενδιάμεσης τιμής και η τοποθέτησή της στο κεντρικό εικονοστοιχείο του παραθύρου.



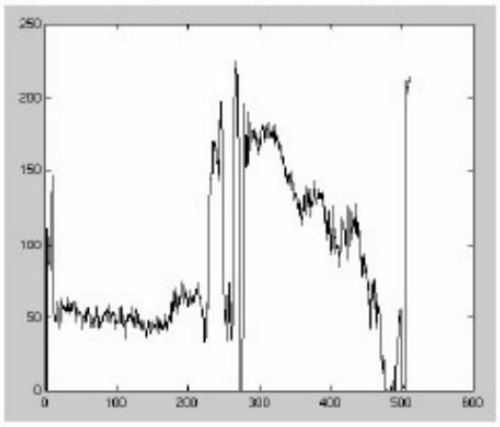
Με την τεχνική αυτή επιτυγχάνεται η απομάκρυνση των ακραίων τιμών (outliers) που μπορεί να περιέχει η εικόνα. Τέτοιες ακραίες τιμές υπάρχουν λόγω της παρουσίας θορύβου τύπου “αλατιού και πιπεριού”. Οπότε, μετά την εφαρμογή του φίλτρου απομακρύνονται οι ακραίες τιμές, άρα και ο θόρυβος αυτού του τύπου.



Στην εικόνα αριστερά υπάρχει θόρυβος που απεικονίζεται με τη μορφή κατακόρυφων γραμμών. Στην εικόνα δεξιά ο θόρυβος έχει απομακρυνθεί, για αυτό δεν υπάρχουν κατακόρυφες γραμμές όπως πριν.



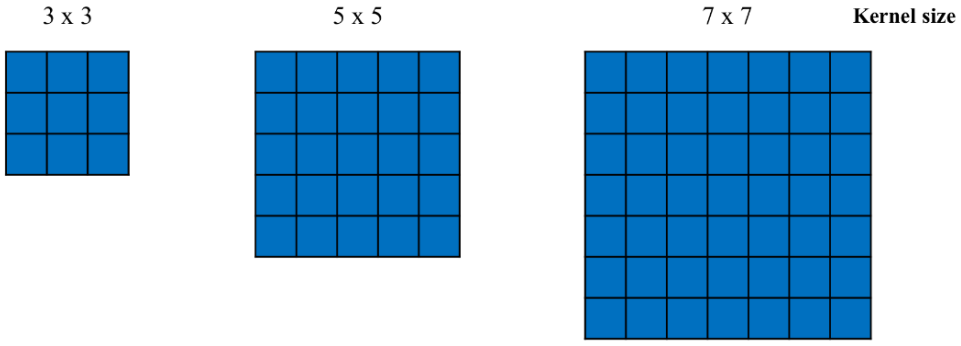
Input



Median

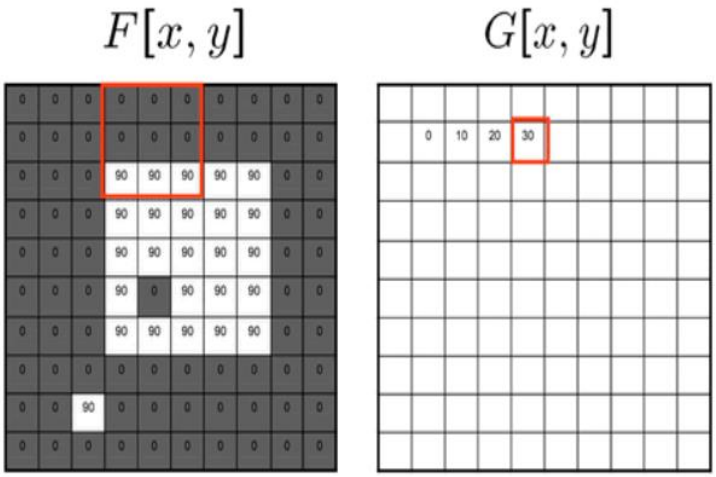
Παρόλο που το φίλτρο median διατηρεί τις ακμές στις εικόνες, είναι επίσης γνωστό ότι αφαιρεί τις λεπτές λεπτομέρειες της εικόνας, όπως γραμμές. Για παράδειγμα, τα φίλτρα με μέγεθος παραθύρου 3×3 αφαιρούν γραμμές πλάτους ενός εικονοστοιχείου και τα φίλτρα με μέγεθος παραθύρου 5×5 αφαιρούν γραμμές πλάτους 2 εικονοστοιχείων. Οπότε, είναι αναγκαίο να ρυθμιστεί το πλάτος του παραθύρου, ανάλογα με το μέγεθος του θορύβου που θέλουμε να αφαιρέσουμε και ανάλογα με τα μέγεθος των μικρότερων λεπτομερειών της εικόνας που θέλουμε να διατηρηθούν. Αν μετά την εφαρμογή του φίλτρου ο θόρυβος παραμένει, τότε πρέπει να αυξηθεί το μέγεθος του παραθύρου. Αντίθετα, αν χαλάει η ανάλυση της τελικής εικόνας (θόλωμα), τότε πρέπει να μειωθεί το μέγεθος του παραθύρου.

Το παράθυρου, συνήθως επιλέγεται να είναι τετράγωνο και το πλάτος του να έχει περιττό αριθμό εικονοστοιχείων, ώστε να υπάρχει κάποιο κεντρικό εικονοστοιχείο, στο οποίο θα τοποθετείται η ενδιάμεση τιμή. Τα συνηθέστερα παράθυρα που χρησιμοποιούνται είναι 3×3, 5×5, 7×7, 9×9 κτλ.



Ένα θέμα που προκύπτει μετά την εφαρμογή ενός 3x3 φίλτρου είναι το γεγονός ότι η εικόνα που θα προκύψει θα είναι μικρότερη κατά δύο εικονοστοιχεία, κατά πλάτος και κατά μήκος. Αυτό συμβαίνει επειδή το παράθυρο (kernel) καθώς μετακυλύει στην εικόνα, οι τιμές θα τοποθετούνται στο κεντρικό εικονοστοιχείο,

επομένως στο περιθώριο δεν θα τοποθετηθεί κάποια τιμή. Στην παρακάτω εικόνα το παράθυρο ξεκίνησε να μετακυλύει από την πάνω αριστερή γωνία της εικόνας, οπότε το κεντρικό εικονοστοιχείο στο οποίο τοποθετήθηκε η τιμή, ήταν στη δεύτερη γραμμή και στήλη. Αυτό έχει ως αποτέλεσμα η πρώτη και η τελευταία γραμμή και στήλη να παραμένουν κενές.



Γενικά, η νέα εικόνα μετά την εφαρμογή του φίλτρου, θα είναι μικρότερη κατά kernel size - 1 εικονοστοιχεία, κατά πλάτος και κατά μήκος.

Σε περίπτωση που είναι απαραίτητο η εικόνα να διατηρήσει τις αρχικές της διαστάσεις, μπορούμε να γεμίσουμε (padding) το περιθώριο της αρχικής εικόνας κατά (kernel size – 1) δια 2, ώστε μετά την εφαρμογή του φίλτρου να παραμείνουν οι αρχικές διαστάσεις της εικόνας. Για παράδειγμα, σε 3x3 φίλτρο πρέπει να γεμίσει το περιθώριο κατά ένα εικονοστοιχείο.

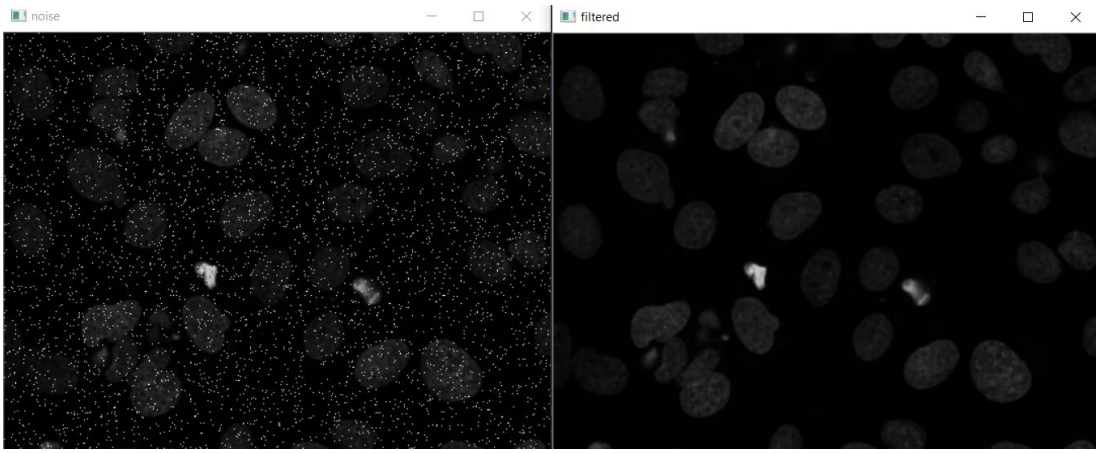
Παρακάτω φαίνεται η υλοποίηση της συνάρτησης median που παίρνει σαν παραμέτρους το μέγεθος του παραθύρου (kernel) και την εικόνα στην οποία θα εφαρμοστεί το φίλτρο. Το γέμισμα του περιθωρίου (padding), γίνεται με τη συνάρτηση **cv2.copyMakeBorder** της opencv που παίρνει σαν ορίσματα την εικόνα στην οποία θα γίνει το γέμισμα, τον αριθμό των εικονοστοιχείων που θα γεμίσουν πάνω, κάτω, αριστερά και δεξιά από την εικόνα και τη μέθοδο με την οποία θα γεμίσουν. Σε αυτήν την περίπτωση χρησιμοποιήθηκε η μέθοδος **cv2.BORDER_REFLECT**, στην οποία το περιθώριο θα είναι αντανάκλαση των στοιχείων περιγράμματος της αρχικής εικόνας. Η μέθοδος αυτή φαίνεται να λειτουργεί πολύ αποτελεσματικά σε σχέση με το γέμισμα με κάποιο σταθερό αριθμό ή με το 0 που θα μπορούσε να αλλοιώσει σε μεγάλο βαθμό την επιλογή της ενδιάμεσης τιμής.

```
def median_fiter(image, kernel_size):
    image_median = image.copy()
    k = kernel_size // 2
    image_median_boarder = cv2.copyMakeBorder(image_median, k, k, k,
k, cv2.BORDER_REFLECT)
    for i in range(k, image_median_boarder.shape[0]-k):
        for j in range(k, image_median_boarder.shape[1]-k):
            kernel = image_median_boarder[i-k:i+kernel_size-k, j-
k:j+kernel_size-k]
            median_value = np.median(kernel)
            image_median[i-k, j-k] = median_value
    return image_median
```

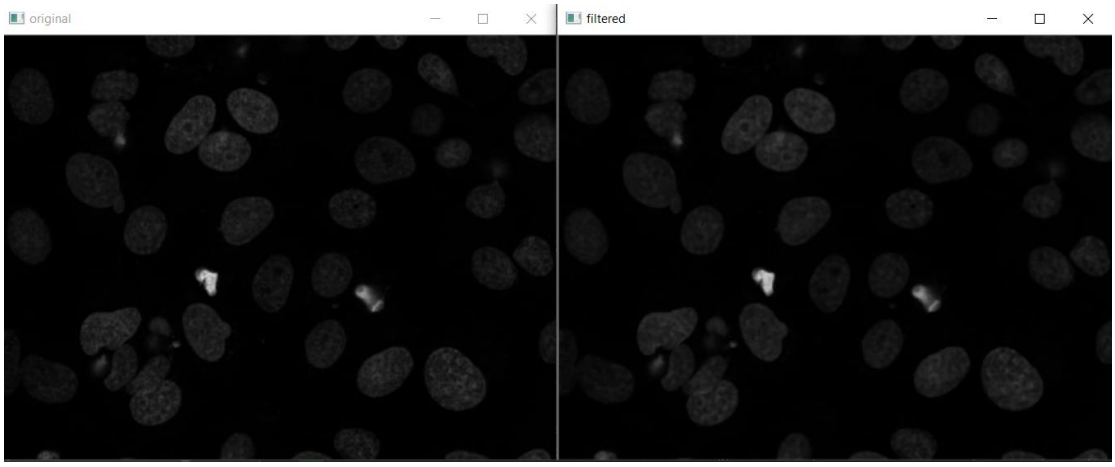
Με την παρακάτω εντολή εφαρμόζεται η συνάρτηση median στην εικόνα με το θόρυβο με μέγεθος παραθύρου 3x3.

```
filtered_img = median_fiter(noise_img, 3)
```

Αριστερά φαίνεται η εικόνα που περιέχει το θόρυβο τύπου “αλατιού και πιπεριού” και δεξιά η εικόνα στην οποία εφαρμόστηκε το φίλτρο median.



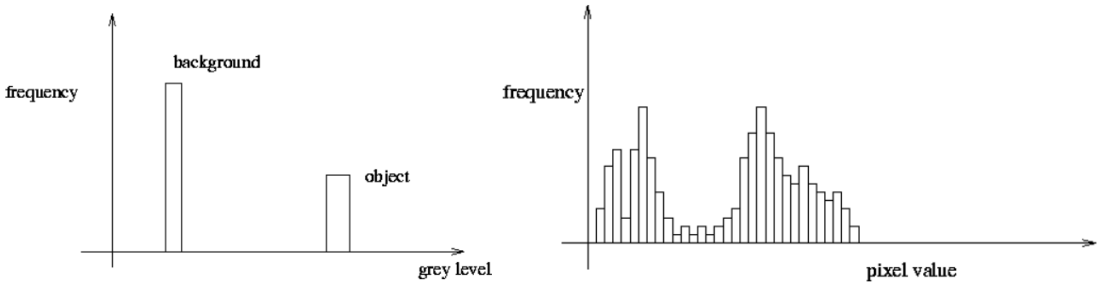
Παρατηρούμε ότι ο θόρυβος έχει απομακρυνθεί και πλέον η εικόνα είναι ίδια με την αρχική. Αριστερά φαίνεται η αρχική εικόνα και δεξιά η εικόνα στην οποία εφαρμόστηκε το φίλτρο median, οπότε πλέον δεν παρατηρείται καμία διαφορά στις δύο εικόνες.



3. Μετατροπή της εικόνας διαβάθμισης του γκρι σε δυαδική εικόνα

Οι δυαδικές εικόνες είναι εικόνες που έχουν κβαντιστεί σε δύο τιμές, που συνήθως δηλώνονται με τιμές εικονοστοιχείων 0 και 255, που αντιπροσωπεύουν το μαύρο και το άσπρο ή 0 και 1 αντίστοιχα. Ο κύριος λόγος που οι δυαδικές εικόνες είναι ιδιαίτερα χρήσιμες στον τομέα της επεξεργασίας εικόνας είναι επειδή επιτρέπουν τον εύκολο διαχωρισμό ενός αντικειμένου από το φόντο.

Αριστερά φαίνεται το ιδανικό ιστόγραμμα ενός αντικειμένου σε πιο σκούρο φόντο και δεξιά το ιστόγραμμα μιας εικόνας που δείχνει τη συχνότητα εμφάνισης κάθε τιμής της κλίμακας του γκρι.



Μια δυαδική εικόνα μπορεί να παραχθεί από μια εικόνα σε κλίμακα του γκρι με τη χρήση της τεχνικής του κατωφλίου (thresholding). Με βάση την εικόνα επιλέγεται ένα κατώφλι (threshold), με το οποίο γίνεται σύγκριση όλων των τιμών των εικονοστοιχείων της εικόνας. Εάν η τιμή του εικονοστοιχείου της αρχικής εικόνας είναι μικρότερη από το κατώφλι, στη δυαδική εικόνα που θα παραχθεί, η τιμή του συγκεκριμένου εικονοστοιχείου θα είναι ίση με 0 δηλαδή θα έχει μαύρο χρώμα, διαφορετικά θα είναι ίση με μια μέγιστη τιμή που συνήθως επιλέγεται να είναι το άσπρο χρώμα (τιμή 255). Η συνάρτηση **cv2.threshold** χρησιμοποιείται για την εφαρμογή του κατωφλίου. Το πρώτο όρισμα είναι η εικόνα που θα μετατραπεί σε δυαδική και θα πρέπει να είναι μια εικόνα σε κλίμακα του γκρι. Το δεύτερο όρισμα είναι η τιμή κατωφλίου (threshold) που χρησιμοποιείται για την ταξινόμηση των τιμών των εικονοστοιχείων. Το τρίτο όρισμα είναι η μέγιστη τιμή που εκχωρείται σε τιμές εικονοστοιχείων που είναι μεγαλύτερες από το κατώφλι.

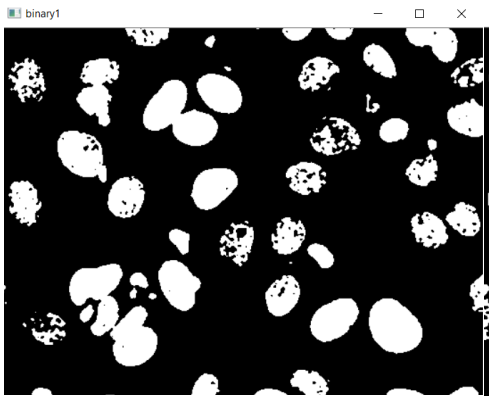
```
thr, binary_img = cv2.threshold(filtered_img, 6, 255,
cv2.THRESH_BINARY)
```

Το κατώφλι που επιλέχθηκε για τη συγκεκριμένη εικόνα της εργασίας μετά από δοκιμές, είναι η τιμή 6. Οπότε, όποιο εικονοστοιχείο έχει τιμή μεγαλύτερη του 6, τότε η τιμή του στη δυαδική εικόνα θα γίνεται 255. Ο διαχωρισμός των κυττάρων από το φόντο είναι πολύ καλός. Αυτό συμβαίνει επειδή το φόντο είναι μαύρο και τα κύτταρα γκρι, οπότε στη δυαδική εικόνα το φόντο θα παραμείνει μαύρο και τα εικονοστοιχεία των κυττάρων θα πάρουν την τιμή 255, θα γίνουν δηλαδή άσπρα.

Όταν το φόντο και τα αντικείμενα περιέχουν παρόμοιες τιμές της κλίμακας του γκρι, τότε το κατώφλι είναι δύσκολο να προσδιοριστεί και η τεχνική thresholding μπορεί να μη λειτουργεί τόσο αποτελεσματικά.

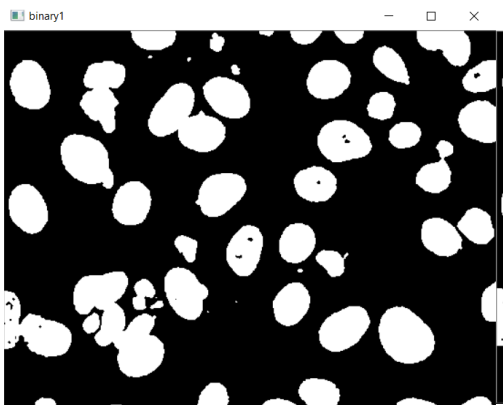
Στις παρακάτω εικόνες φαίνονται κάποιες δοκιμές που έγιναν για να επιλεχθεί το κατάλληλο κατώφλι. Είναι φανερό ότι όταν το κατώφλι είναι 15 και 10, τα κύτταρα έχουν μαύρες τρύπες και φαίνονται αλλοιωμένα, όταν είναι 3 και 1 εμφανίζεται υπερβολικός θόρυβος στην εικόνα (άσπρα στίγματα που στην πραγματικότητα δεν είναι κύτταρα) και κάποια κύτταρα που είναι διαφορετικά ενώνονται μεταξύ τους. Όταν η τιμή του κατωφλίου είναι 6 και 7 φαίνεται να έχουμε τα καλύτερα αποτελέσματα. Δεν θέλουμε ούτε διαφορετικά κύτταρα να ενωθούν μεταξύ τους, αλλά ούτε να υπάρχουν μαύρες τρύπες μέσα στα κύτταρα, γιατί τότε θα μετρηθεί λάθος επιφάνεια. Μας ενδιαφέρει τόσο ο εντοπισμός των κυττάρων, όσο και ο υπολογισμός της επιφάνειάς τους, οπότε θέλουμε να διατηρηθούν όσο γίνεται οι αρχικές τους διαστάσεις. Επιλέξαμε σαν τιμή κατωφλίου το 6, επειδή παρατηρούμε ότι τα κύτταρα έχουν τις λιγότερες δυνατές μαύρες τρύπες και δεν υπάρχουν κύτταρα τα οποία ενώνονται.

Threshold = 15

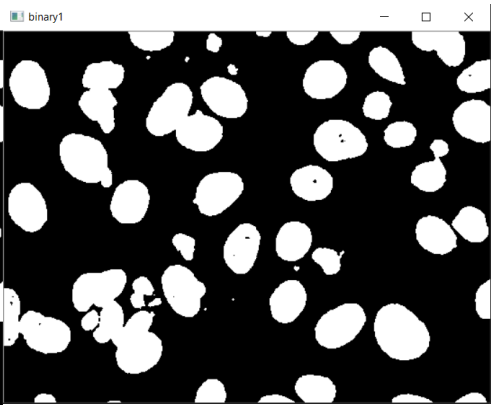


Threshold = 10

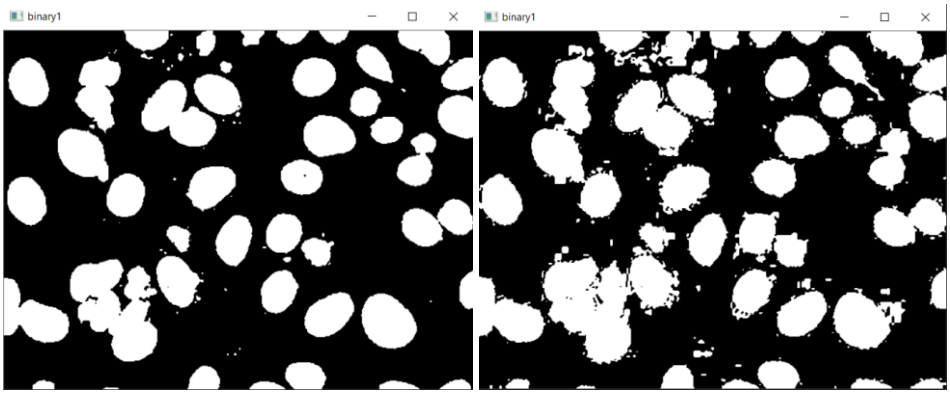
Threshold = 7



Threshold = 6



Threshold = 3 Threshold = 1



4. Μορφολογικοί μετασχηματισμοί

Στη δυαδική εικόνα που δημιουργήθηκε έχουν παραμείνει ορισμένες μαύρες τρύπες μέσα στα κύτταρα και κάποια άσπρα στίγματα στο φόντο. Τέτοιου είδους θόρυβος είναι δυνατό να απομακρυνθεί με διάφορους μορφολογικούς μετασχηματισμούς.

Οι παρακάτω πράξεις που θα αναφερθούν εφαρμόζονται σε δυαδικές εικόνες με τη βοήθεια ενός δομικού στοιχείου (structuring element). Ένα δομικό στοιχείο είναι μια μικρή εικόνα που χρησιμοποιείται ως κινούμενο παράθυρο και μπορεί να έχει οποιοδήποτε σχήμα και μέγεθος.

Διαστολή (Dilation)

Η πράξη dilation οδηγεί στη διαστολή των αντικειμένων, καθώς θέτει την τιμή 1 στα εικονοστοιχεία που συνορεύουν με τα αντικείμενα της εικόνας.

Πιο συγκεκριμένα το δομικό στοιχείο διατρέχει όλα τα εικονοστοιχεία της εικόνας και κατά τη διαδικασία αυτή αν καλύπτει κάποιο εικονοστοιχείο από κάποιο αντικείμενο, θέτει σε όλα τα εικονοστοιχεία τα οποία καλύπτει τη δεδομένη στιγμή την τιμή 255. Οποιοδήποτε άλλο εικονοστοιχείο της εικόνας παίρνει την τιμή 0.

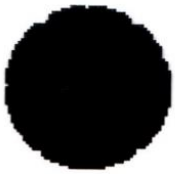
Η διαστολή της εικόνας f από το δομικό στοιχείο s δίνεται από τον τύπο:

$$f \oplus s$$

Με τη συγκεκριμένη μορφολογική πράξη τα αντικείμενα μεγεθύνονται.


Αρχική εικόνα Μετά από dilation



Αρχική εικόνα Μετά από dilation Αρχική εικόνα Μετά από dilation

Με την τεχνική της διαστολής μπορούν να διορθωθούν οι παραπάνω ατέλειες στα σχήματα. Τα άσπρα κενά στο αντικείμενο καλύπτονται.

<u>Διάβρωση (Erosion)</u>

Η πράξη erosion οδηγεί στη διάβρωση των αντικειμένων, καθώς θέτει την τιμή 0 στα εικονοστοιχεία που βρίσκονται στο περίγραμμα των αντικειμένων της εικόνας.

Πιο συγκεκριμένα το δομικό στοιχείο διατρέχει όλα τα εικονοστοιχεία της εικόνας και κατά τη διαδικασία αυτή αν ολόκληρο το δομικό στοιχείο χωράει μέσα σε κάποιο αντικείμενο, θέτει σε όλα τα εικονοστοιχεία τα οποία καλύπτει τη δεδομένη στιγμή την τιμή 255. Οποιοδήποτε άλλο εικονοστοιχείο της εικόνας παίρνει την τιμή 0.

Η διάβρωση της εικόνας f από το δομικό στοιχείο s δίνεται από τον τύπο:

$$f \ominus s$$

Με τη συγκεκριμένη μορφολογική πράξη τα αντικείμενα σμικρύνονται.


Αρχική εικόνα Μετά από erosion

Αρχική εικόνα Μετά από erosion Αρχική εικόνα Μετά από erosion

Με την τεχνική της διάβρωσης μπορούν να διορθωθούν οι παραπάνω ατέλειες στα σχήματα. Τα σχήματα που ακουμπάνε μεταξύ τους είναι δυνατόν να διαχωριστούν και οι μαύρες ατέλειες μπορεί να εξαφανιστούν.

Άνοιγμα (Opening)

Η πράξη opening είναι στην πραγματικότητα μια πράξη διάβρωσης που ακολουθείται από μια πράξη διαστολής. Χρησιμοποιείται για την εκκαθάριση της εικόνας από δομές μικρότερες του δομικού στοιχείο, χωρίς να αλλοιώνεται η δομή των αντικειμένων που είναι μεγαλύτερα του δομικού στοιχείου. Ο θόρυβος τύπου “αλατιού” μπορεί να εξαφανιστεί με αυτήν την τεχνική.

Αρχική εικόνα Μετά από opening

Επειδή η διαστολή και η διάβρωση γίνονται κατά τον ίδιο βαθμό, τα αντικείμενα με τα το opening διατηρούν το αρχικό τους μέγεθος.

Κλείσιμο (Closing)

Η πράξη closing είναι στην πραγματικότητα μια πράξη διαστολής που ακολουθείται από μια πράξη διάβρωσης. Χρησιμοποιείται για να γεμίσει κενά της εικόνας που είναι μικρότερα του δομικού στοιχείου, χωρίς να επηρεάσει τα αντικείμενα που είναι μεγαλύτερα του. Ο θόρυβος τύπου “πιπεριού” μπορεί να εξαφανιστεί με αυτήν την τεχνική.

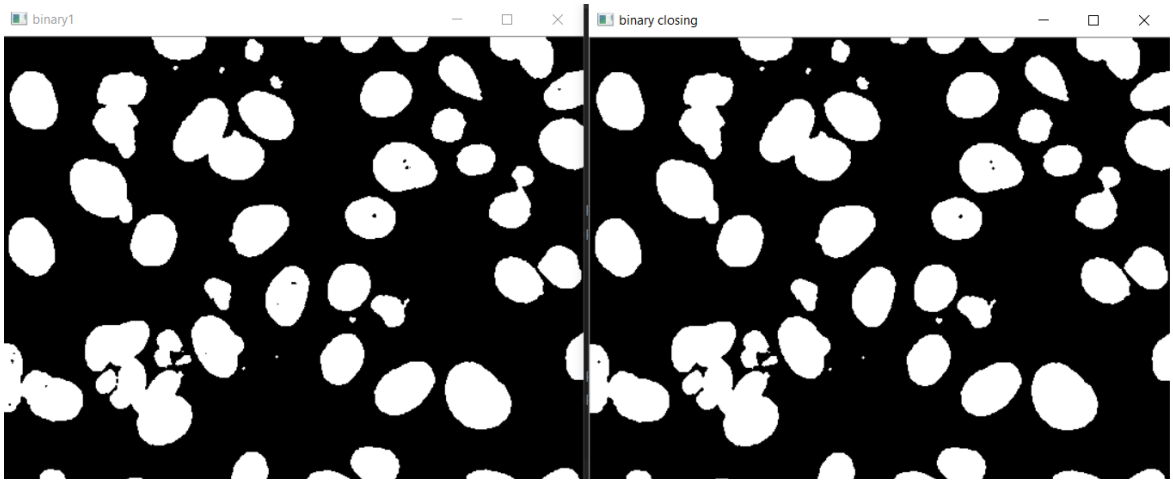
Αρχική εικόνα Μετά από closing

Επειδή η διαστολή και η διάβρωση γίνονται κατά τον ίδιο βαθμό, τα αντικείμενα με τα το opening διατηρούν το αρχικό τους μέγεθος, όπως ακριβώς και στο opening.

Στη συγκεκριμένη εργασία, επειδή οι ατέλειες που έχουμε έχουν κυκλικό σχήμα κυρίως και τα κύτταρα είναι επίσης κυκλικά, επιλέχθηκε το δομικό στοιχείο να έχει σχήμα έλλειψης.

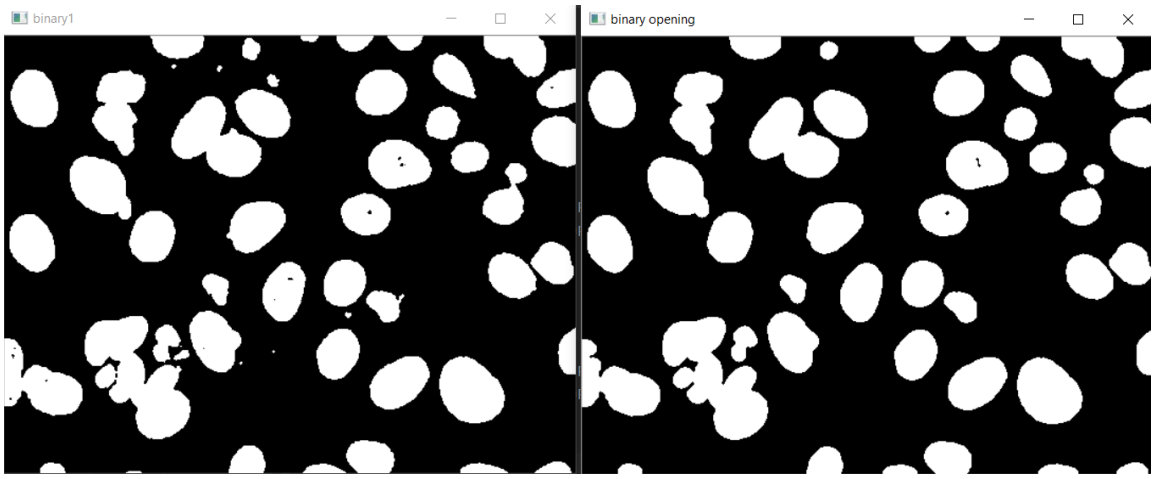
```
# Closing
strel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
binary_img = cv2.morphologyEx(binary_img, cv2.MORPH_CLOSE, strel)

# Opening
strel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (13,13))
binary_img = cv2.morphologyEx(binary_img, cv2.MORPH_OPEN, strel)
```

Για να εξαφανιστούν οι μαύρες τρύπες μέσα στα κύτταρα χρησιμοποιήθηκε η τεχνική closing. Όσο μεγαλύτερο το μέγεθος του δομικού στοιχείου, τόσες περισσότερες τρύπες εξαφανίζονται, όμως υπάρχει ο κίνδυνος κάποια κύτταρα να ενωθούν μεταξύ τους. Επειδή διαπιστώσαμε ότι είναι δύσκολος ο διαχωρισμός των διαφορετικών κυττάρων με τη χρήση της τεχνικής opening, επιλέξαμε να χρησιμοποιήσουμε μικρό μέγεθος για το δομικό στοιχείο στο closing, ώστε να αποφύγουμε τη συνένωση διαφορετικών κυττάρων, για αυτό το λόγο κάποιες τρύπες παρέμειναν. Αυτές οι ατέλειες μπορεί να μην είναι τόσο σημαντικές όσον αφορά τον εντοπισμό των κυττάρων (τα κύτταρα θα μπορούσαν να εντοπιστούν σωστά ακόμα και με την ύπαρξη τρυπών), όμως επηρεάζουν πάρα πολύ τον υπολογισμό της επιφάνειας των κυττάρων που ζητείται στην εργασία (η επιφάνεια αν υπάρχουν μαύρες τρύπες θα υπολογιστεί μικρότερη από την πραγματική).

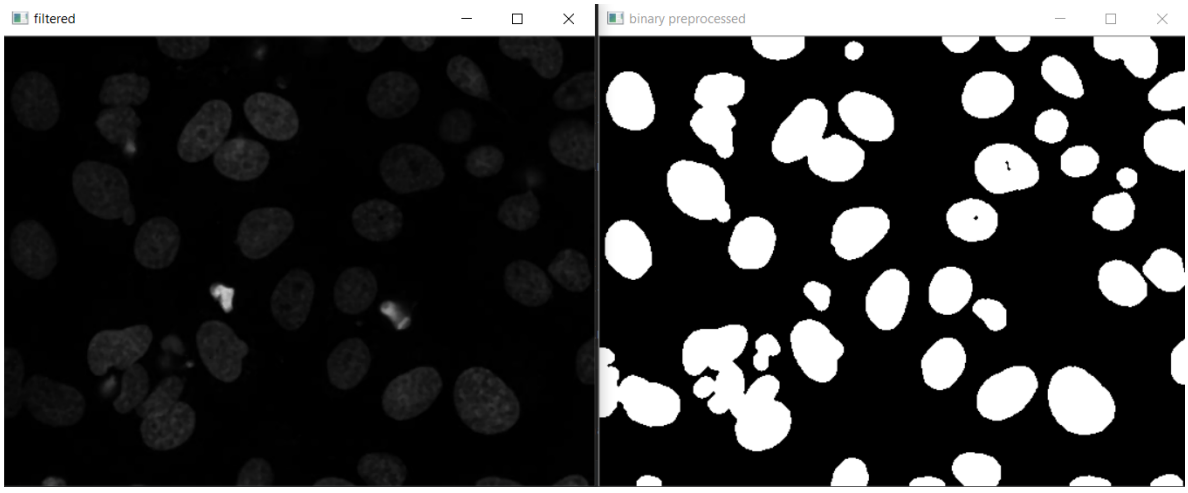


Αρχική εικόνα Μετά από closing

Επειδή στο φόντο δημιουργήθηκαν λευκά στίγματα που δεν είναι κύτταρα στην πραγματικότητα, χρησιμοποιήθηκε η τεχνική του opening για την εξαφάνιση αυτών των ατελειών που θα μπορούσαν λανθασμένα να θεωρηθούν σαν κύτταρα. Το μέγεθος του δομικού στοιχείου πρέπει να είναι μεγαλύτερο από τη μεγαλύτερη ατέλεια που θέλουμε να αφαιρέσουμε και μικρότερο από το μικρότερο αντικείμενο που θέλουμε να διατηρηθεί στην εικόνα. Αν είναι πολύ μεγάλο το μέγεθός του, υπάρχει ο κίνδυνος να χαθεί κάποιο κύτταρο και αν είναι πολύ μικρό μπορεί να παραμείνουν τα πιο μεγάλα λευκά στίγματα.



Αρχική εικόνα Μετά από opening



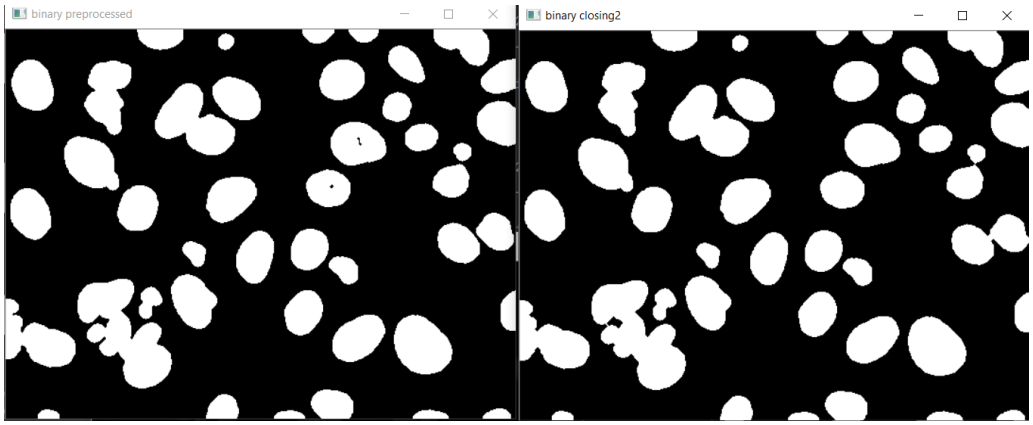
Αρχική εικόνα Μετά από μορφολογικές πράξεις

Μετά την εφαρμογή των μορφολογικών τεχνικών παρατηρούμε ότι όλα τα λευκά στίγματα από το φόντο που δεν είναι κύτταρα έχουν εξαφανιστεί. Επίσης, οι περισσότερες μαύρες τρύπες εξαφανίστηκαν, μόνο δύο έχουν παραμείνει.

Επειδή οι τρύπες που παρέμειναν είναι μεγάλες και θα έχουμε μεγάλο σφάλμα στον υπολογισμό της επιφάνειας των δύο κυττάρων επιλέξαμε να προχωρήσουμε σε περεταίρω μορφολογικές πράξεις.

```
# Closing
strel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
binary_img = cv2.morphologyEx(binary_img, cv2.MORPH_CLOSE, strel,
iterations=2)

# Opening
strel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (13 ,13))
binary_img = cv2.morphologyEx(binary_img, cv2.MORPH_OPEN, strel)
```

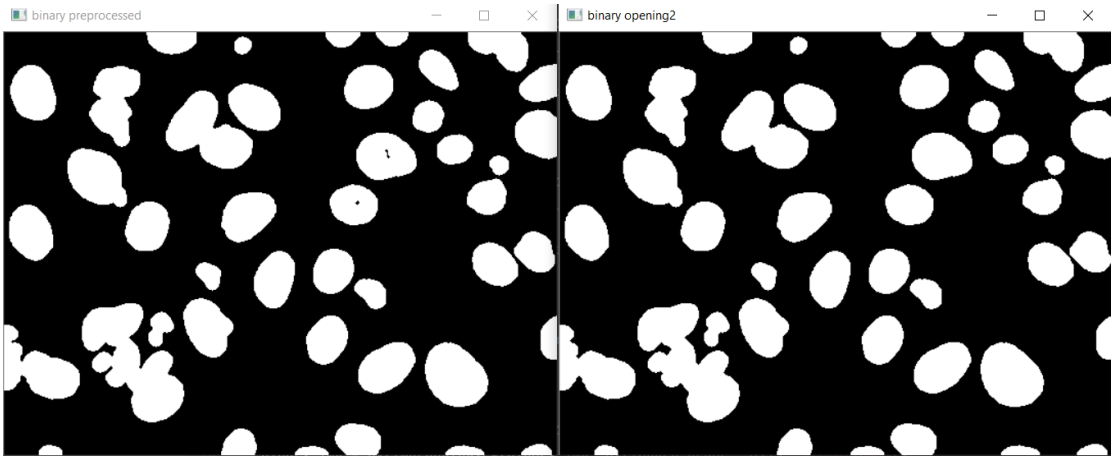
Εφαρμόστηκε πάλι η πράξη closing για να εξαφανιστούν οι τρύπες, απλά τώρα με τη χρήση του ορίσματος iterations, είναι δυνατόν να εφαρμοστεί η ίδια πράξη με το ίδιο δομικό στοιχείο όσες φορές θέλουμε, εδώ εφαρμόστηκε δύο φορές.



Προηγούμενη επεξεργασία Μετά από εκ νέου closing

Παρατηρούμε ότι οι τρύπες εξαφανίστηκαν, όμως μερικά κύτταρα ενώθηκαν μεταξύ τους και υπάρχει κίνδυνος να αναγνωριστούν σαν ένα ενιαίο.

Άρα, για το διαχωρισμό αυτών των κυττάρων εφαρμόζεται ξανά η πράξη opening.



Προηγούμενη επεξεργασία Μετά από εκ νέου opening

Πλέον μπορούμε να παρατηρήσουμε ότι δεν υπάρχουν τρύπες και τα κύτταρα που πριν ήτα διαχωρισμένα, διαχωρίστηκαν και πάλι. Βέβαια, τα κύτταρα τα οποία ήταν από πριν ενωμένα δεν διαχωρίστηκαν, καθώς δεν είναι δυνατόν να διαχωριστούν με opening και πιθανότατα στο τέλος να εντοπιστούν σαν ένα ενιαίο κύτταρο.

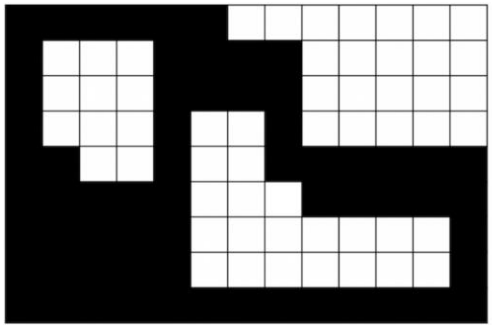
Μπορούμε επομένως να προχωρήσουμε στο επόμενο ζητούμενο της εργασίας, καθώς η επεξεργασία που έγινε στην εικόνα ήταν η καλύτερη δυνατή.

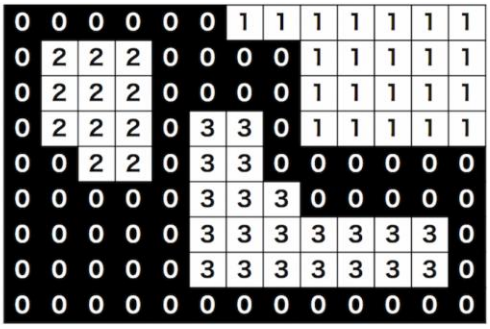
5. Ανίχνευση όλων των κυττάρων της εικόνας

Για την ανίχνευση των κυττάρων της εικόνας χρησιμοποιήθηκε η συνάρτηση **cv2.connectedComponents**. Η συνάρτηση αυτή χρησιμοποιείται για τον εντοπισμό και τη μέτρηση του αριθμού των συνδεδεμένων περιοχών σε μια δυαδική εικόνα. Επιστρέφει τον αριθμό των συνδεδεμένων στοιχείων που υπάρχουν στην εικόνα και έναν πίνακα που έχει μέγεθος ίσο με το μέγεθος της εικόνας και τα εικονοστοιχεία που αναφέρονται σε κάθε συνδεδεμένο στοιχείο επισημαίνονται με την ίδια μοναδική ετικέτα.

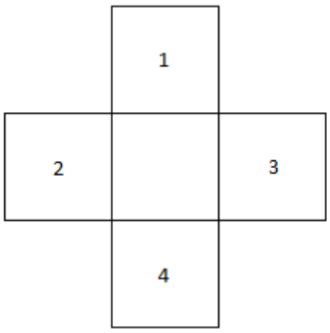
Για παράδειγμα, αν υπάρχουν τρία διασυνδεδεμένα στοιχεία, όπως φαίνεται και στην παραπάνω εικόνα, τα αντίστοιχα εικονοστοιχεία τους θα επισημανθούν με τον αριθμό της κάθε ετικέτας (τα εικονοστοιχεία του πρώτου αντικειμένου θα λάβουν την τιμή 1 κτλ.). Όλα τα υπόλοιπα εικονοστοιχεία θα έχουν την τιμή μηδέν. Η συνάρτηση θα επιστρέψει τον αριθμό 3 και τον πίνακα που φαίνεται δεξιά στην παρακάτω εικόνα.

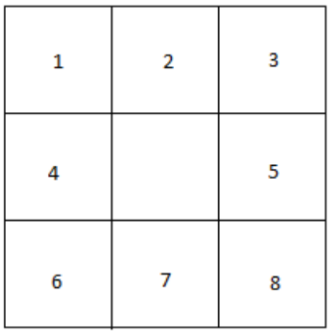
```
n_labels, labels = cv2.connectedComponents(binary_img)
```





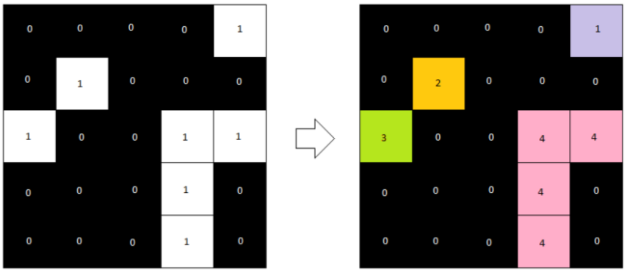
Υπάρχουν δύο συνήθεις τρόποι για να καθοριστεί εάν ένα στοιχείο είναι συνδεδεμένο ή όχι. Το ένα δηλώνει ότι ένα εικονοστοιχείο έχει μόνο 4 γείτονες (μερικές φορές ονομάζεται συνδεσιμότητα 4). Το άλλο δηλώνει ότι ένα εικονοστοιχείο έχει 8 γείτονες.



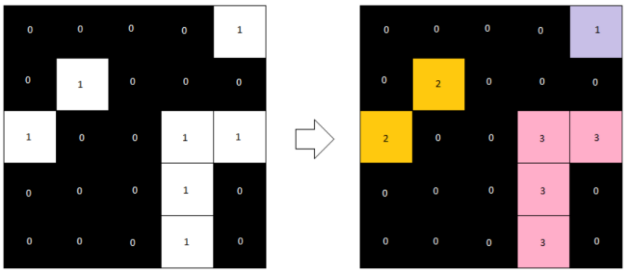


4 Γείτονες 8 Γείτονες

Στη συνδεσιμότητα 8 γειτόνων θα υπάρχουν λιγότερα συ δεδομένα αντικείμενα, σε σύγκριση με τους 4 γείτονες.



4 Γείτονες



8 Γείτονες

Αρχικά, ο αλγόριθμος για κάθε μη μηδενικό εικονοστοιχείο, ελέγχει τους γείτονές του. Αν δεν έχει μη μηδενικούς γείτονες, τότε πρόκειται για ένα νέο στοιχείο, το οποίο λαμβάνει μια νέα ετικέτα. Αν έχει έναν μη μηδενικό γείτονα, αυτό σημαίνει ότι αυτά τα δύο εικονοστοιχεία είναι συνδεδεμένα, οπότε λαμβάνει την ίδια ετικέτα με τον γείτονα. Αν έχει περισσότερους από έναν μη μηδενικούς γείτονες, υπάρχουν δύο περιπτώσεις:

- Οι γείτονες έχουν όλοι την ίδια ετικέτα. Οπότε, λαμβάνει την ίδια ετικέτα με τους γείτονες.
- Οι γείτονες έχουν διαφορετικές ετικέτες. Εφόσον, όλα αυτά τα εικονοστοιχεία είναι συνδεδεμένα, οι ετικέτες θα πρέπει να είναι όλες ίδιες. Το τρέχον εικονοστοιχείο λαμβάνει τη χαμηλότερη ετικέτα των γειτόνων. Κάθε εικονοστοιχείο που δεν είναι μηδενικό θα έχει πλέον μια ετικέτα. Ωστόσο, ορισμένες συνδεδεμένες περιοχές θα έχουν διαφορετικές ετικέτες. Τέλος, όλες οι ετικέτες που είναι ισοδύναμες (δηλαδή αναφέρονται στο ίδιο συνδεδεμένο αντικείμενο) θα λάβουν την ίδια τιμή, που θα είναι η τιμή της χαμηλότερης ετικέτας από αυτές.

Επομένως, όλα τα εικονοστοιχεία που αναφέρονται στο ίδιο συνδεδεμένο αντικείμενο θα έχουν την ίδια, μοναδική ετικέτα.

Στη συνέχεια, είναι αναγκαίο να επεξεργαστούμε τον πίνακα με τις ετικέτες και χρησιμοποιώντας τη συνάρτηση **cv2.boundingRect** να βρεθούν τα περιβάλλοντα κουτιά του κάθε συνδεδεμένου αντικειμένου. Η συνάρτηση αυτή, δέχεται σαν είσοδο έναν πίνακα που περιέχει μόνο 0 και 255 (πίνακας στο μέγεθος της εικόνας) και επιστρέφει 4 τιμές (τις συντεταγμένες της επάνω αριστερής γωνίας του κουτιού, το πλάτος του και το μήκος του).

Με τη χρήση μιας επαναληπτικής δομής for, επιλέγουμε σε κάθε επανάληψη μία ετικέτα, δηλαδή ένα συνδεδεμένο στοιχείο. Η μεταβλητή label παίρνει τιμές από 1 μέχρι το πλήθος των συνδεδεμένων στοιχείων που υπάρχουν στην εικόνα (πλήθος ετικετών).

```
for label in range(1, n_labels):
```

Σε ένα νέο πίνακα που έχει το ίδιο μέγεθος με τον πίνακα των ετικετών, θέτουμε 255 στα εικονοστοιχεία που η τιμή τους στον πίνακα των ετικετών είναι ίση με τον αύξοντα αριθμό της ετικέτας. Για παράδειγμα, στην πρώτη επανάληψη που label = 1, στον πίνακα mask θέτουμε 255 στα εικονοστοιχεία που στον πίνακα labels έχουν την τιμή 1. Σε όλα τα υπόλοιπα θέτουμε την τιμή 0. Τέλος, εφαρμόζουμε τη συνάρτηση **cv2.boundingRect**, αφού πλέον ο πίνακας mask περιέχει μόνο 0 και 255 και με βάση τις τιμές που μας επιστρέφει σχεδιάζουμε το περιβάλλον κουτί με τη συνάρτηση **cv2.rectangle**. Η συγκεκριμένη συνάρτηση παίρνει σαν όρισμα την επάνω αριστερή και την κάτω δεξιά γωνία του κουτιού που θα σχεδιαστεί που στην περίπτωσή μας είναι οι συντεταγμένες x, y (συντεταγμένες επάνω αριστερής γωνίας) και x+w, y+h (συντεταγμένες κάνω δεξιάς γωνίας) που μας επιστρέφει η **cv2.boundingRect**. Επίσης, τοποθετούμε μέσα σε κάθε περιβάλλον κουτί τον

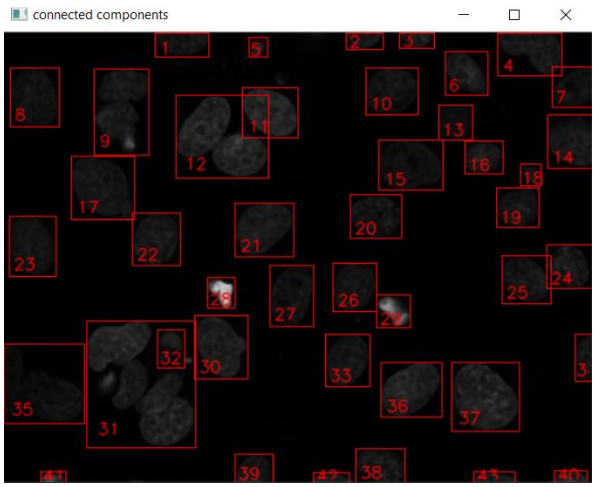
αύξοντα αριθμό της αντίστοιχης ετικέτας, με τη συνάρτηση **cv2.putText**. Μετά από δοκιμές το κείμενο τοποθετήθηκε κοντά στην κάτω αριστερή γωνία του κάθε περιβάλλοντος κουτιού, για να φαίνεται όσο το δυνατόν καλύτερα σε όλα τα κύτταρα. Η **cv2.putText** δέχεται σαν 3ο όρισμα τις συντεταγμένες της κάτω αριστερής γωνίας του κειμένου.

Η παραπάνω διαδικασία θα πραγματοποιηθεί για κάθε συνδεδεμένο στοιχείο που έχει εντοπιστεί στην εικόνα.

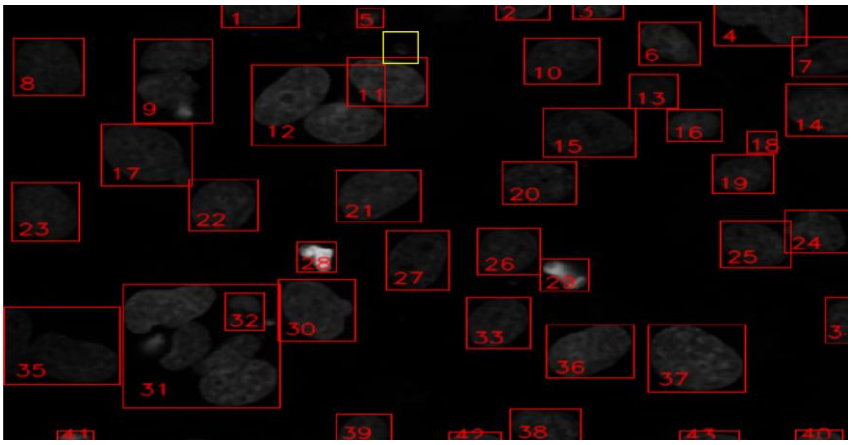
```
filtered_img = cv2.cvtColor(filtered_img, cv2.COLOR_GRAY2BGR)

for label in range(1, n_labels):
    mask = np.array(labels, dtype=np.uint8)
    mask[labels == label] = 255
    mask[labels != label] = 0
    x, y, w, h = cv2.boundingRect(mask)
    filtered_img = cv2.rectangle(filtered_img, (x, y),(x+w, y+h),
color, thickness)
    org = (int(x+0.1*w), int(y+h*0.9))
    filtered_img = cv2.putText(filtered_img, str(label), org, font,
fontScale, color, thickness, cv2.LINE_AA)

    area = np.count_nonzero(mask == 255)
    bounding_box_area = w*h
    mean_graylevel_value = (int_img[y, x] - int_img[h+y, x] -
int_img[y, w+x] + int_img[h+y, w+x])/bounding_box_area

    print("---- Region " + str(label) + ": ----")
    print("Area (px):", area)
    print("Bounding Box Area (px):", bounding_box_area)
    print("Mean graylevel value in bounding box:",
mean_graylevel_value)
```

Αξιοσημείωτο, είναι επίσης το γεγονός ότι σαν χρώμα του κειμένου και των περιβάλλοντων κουτιών επιλέχθηκε το κόκκινο, όμως εφόσον η εικόνα αρχικά είχε διαβαστεί σαν εικόνα σε κλίμακα του γκρι, πρέπει πρώτα να γίνει η μετατροπή της σε BGR με τη συνάρτηση **cv2.cvtColor**, διαφορετικά δε θα φανεί κάποιο αποτέλεσμα στην εικόνα.

```
filtered_img = cv2.cvtColor(filtered_img, cv2.COLOR_GRAY2BGR)
```



Το αποτέλεσμα είναι σχετικά καλό, αφού καταφέραμε να εντοπίσουμε όλα τα κύτταρα της εικόνας (43 κύτταρα συνολικά εντοπίστηκαν). Βέβαια, κάποια κύτταρα που και στην αρχική εικόνα ήταν ενωμένα, ανιχνεύθηκαν σαν ένα ενιαίο κύτταρο, παρά τις προσπάθειές μας να τα διαχωρίσουμε (χαρακτηριστικά παραδείγματα τα κύτταρα 9, 12, 31, 35). Επίσης, δεν εντοπίστηκαν κάποιες ανεπαίσθητες σκιάσεις που ακόμα και με το ανθρώπινο μάτι φαίνονται με δυσκολία, οπότε μπορεί να θεωρήσουμε ότι είναι ασφαλές να τις αγνοήσουμε. Μία τέτοια σκίαση φαίνεται στο κίτρινο κουτάκι στην παρακάτω εικόνα.



Οπότε, επειδή όλα τα κύτταρα εντοπίζονται, παρά τις δύο αστοχίες που αναφέρουμε παραπάνω το αποτέλεσμα μας ικανοποιεί για τη συγκεκριμένη εργασία. Ανάλογα βέβαια με την εφαρμογή στην οποία θέλουμε να χρησιμοποιήσουμε το πρόγραμμα μας, πρέπει κάθε φορά να αναλογιζόμαστε τις αστοχίες που προκύπτουν για να μπορούμε με ασφάλεια να κρίνουμε αν έχουμε επιτύχει το επιθυμητό αποτέλεσμα.

Βέβαια, δοκιμάστηκε το ίδιο πρόγραμμα και σε άλλες εικόνες και διαπιστώθηκε ότι παρ' όλο που κανένα κύτταρο δεν χάνεται και δεν παραμένουν τρύπες στα κύτταρα κατά τη μετατροπή της εικόνας σε δυαδική, πολλές φορές διαφορετικά κύτταρα εντοπίζονται σαν ένα ενιαίο (ακόμα και αν πριν δεν είναι ενωμένα). Οπότε, για τη γενίκευση του αλγορίθμου καλούμαστε να αναρωτηθούμε αν μας ενδιαφέρει να έχουμε τα κύτταρα διαχωρισμένα και να τροποποιήσουμε τον αλγόριθμο (λίγο υψηλότερο κατώφλι στο thresholding και ίσως λιγότερο closing).

6. Υπολογισμός επιφάνειας κυττάρων και επιφάνειας περιβάλλοντων κουτιών

Ο υπολογισμός της επιφάνειας του περιβάλλοντος κουτιού του κάθε κυττάρου, ως αριθμός εικονοστοιχείων προκύπτει πολύ εύκολα από την πράξη w*h (πλάτος κουτιού * ύψος κουτιού), τα w, h είναι γνωστά από τη συνάρτηση **cv2.boundingRect** που αναφέρθηκε και παραπάνω.

```
bounding_box_area = w*h
```

Ο υπολογισμός της επιφάνειας του κάθε κυττάρου, που ορίζεται ως ο αριθμός των εικονοστοιχείων που ανήκουν σε αυτό μπορεί να υπολογιστεί από τον πίνακα mask που αναφέρθηκε παραπάνω. Σε κάθε επανάληψη, ο πίνακας αυτός θα έχει την τιμή 255 στα εικονοστοιχεία που αναφέρονται στο συγκεκριμένο κύτταρο. Οπότε, για τον υπολογισμό των εικονοστοιχείων που ανήκουν σε αυτό το κύτταρο αρκεί να υπολογιστεί πόσα εικονοστοιχεία έχουν την τιμή 255 στον πίνακα mask.

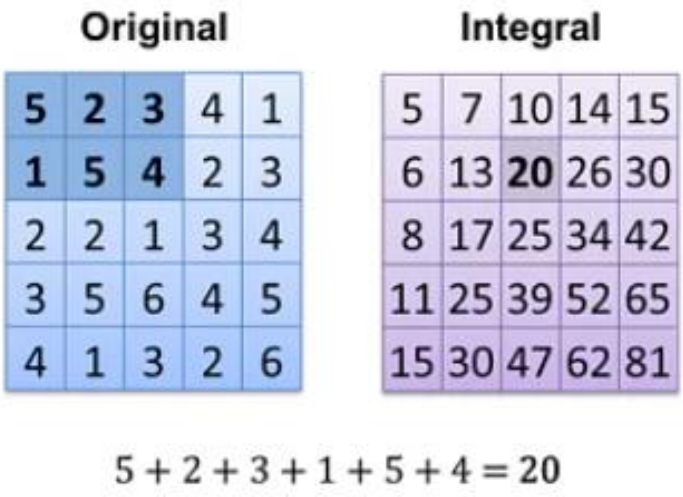
```
area = np.count_nonzero(mask == 255)
```

Οι δύο παραπάνω υπολογισμοί γίνονται για κάθε κύτταρο που εντοπίστηκε, για αυτό το λόγο βρίσκονται μέσα στη δομή επανάληψης for.

7. Υπολογισμός μέσης τιμής διαβάθμισης του γκρι των περιβάλλοντων κουτιών

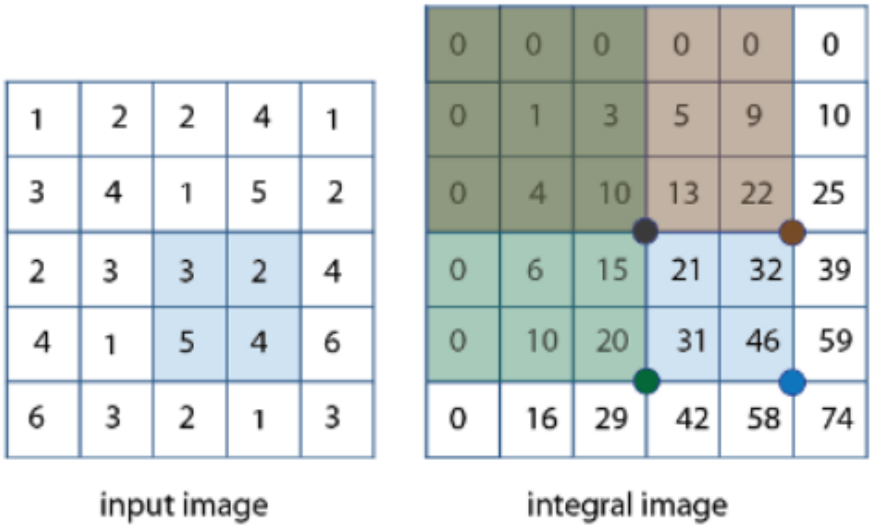
Για τον υπολογισμό της μέσης τιμής διαβάθμισης του γκρι των εικονοστοιχείων που περιέχονται σε κάθε περιβάλλον κουτί είναι απαραίτητος ο υπολογισμός της εικόνας ολοκλήρωσης (integral image).

Η εικόνα ολοκλήρωσης ή αλλιώς πίνακας αθροισμάτων (summed area table), είναι ένας πίνακας που οι τιμές του είναι το άθροισμα των πάνω και αριστερά εικονοστοιχείων της αρχικής εικόνας. Η πράξη γίνεται καλύτερα κατανοητή με το παρακάτω παράδειγμα.

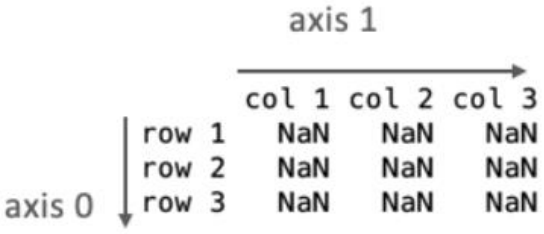


Το στοιχείο με τιμή 81 στην εικόνα ολοκλήρωσης είναι στην ουσία το άθροισμα όλων των τιμών του αρχικού πίνακα.

Παρακάτω φαίνεται η υλοποίηση της συνάρτησης my_integral για τον υπολογισμό της εικόνας ολοκλήρωσης. Αρχικά, ο πίνακας ολοκλήρωσης έχει τις ίδιες διαστάσεις με την αρχική εικόνα και στη συνέχεια προστίθενται μία γραμμή και μία στήλη με μηδενικά στην αρχή του πίνακα αυτού. Οπότε, ο πίνακας ολοκλήρωσης έχει μία παραπάνω γραμμή και στήλη από την αρχική εικόνα.



Στη συνέχεια, χρησιμοποιήθηκε η συνάρτηση **np.cumsum**, η οποία επιστρέφει το αθροιστικό άθροισμα (cumulative sum) των στοιχείων κατά μήκος ενός δεδομένου άξονα. Η **np.cumsum** εφαρμόζεται αρχικά κατά στήλες και στη συνέχεια κατά γραμμές.




```
# row cumsum
arr = np.array([[1,2,3],
                [4,5,6]])
np.cumsum(arr, axis = 1)
```

```
array([[ 1,  3,  6],
       [ 4,  9, 15]])
```

```
# column cumsum
arr = np.array([[1,2,3],
                [4,5,6]])
np.cumsum(arr, axis = 0)
```

```
array([[1, 2, 3],
       [5, 7, 9]])
```

Αν εφαρμοστεί και κατά στήλες και κατά γραμμές (η σειρά δεν παίζει ρόλο), τότε επιστρέφει στην πραγματικότητα τον πίνακα ολοκλήρωσης της εικόνας.

```
# row cumsum
arr = np.array([[1,2,3],
                [4,5,6]])
arr = np.cumsum(arr, axis = 1)
arr = np.cumsum(arr, axis = 0)
arr
```

```
array([[ 1,  3,  6],
       [ 5, 12, 21]])
```

Ο υπολογισμός θα μπορούσε να γίνει με τη χρήση δύο επαναληπτικών δομών for αυτός όμως ο τρόπος που φαίνεται και παρακάτω είναι πολύ πιο αργός.

```
def my_integral(image):
    integral_image = image.copy()
    integral_image = np.concatenate((np.zeros((1, image.shape[1])),
integral_image), axis=0)
    integral_image = np.concatenate((np.zeros((image.shape[0]+1, 1)),
integral_image), axis=1)
    integral_image = integral_image.astype(int)
    integral_image = np.cumsum(integral_image, axis = 1)
    integral_image = np.cumsum(integral_image, axis = 0)
    return integral_image
```

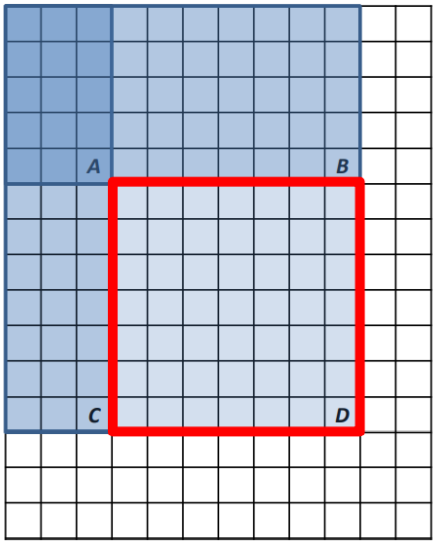
Με for:

```
def my_integral(image):
    integral_image = image.copy()

    integral_image = np.concatenate((np.zeros((1, image.shape[1])),
integral_image), axis=0)
    integral_image = np.concatenate((np.zeros((image.shape[0]+1, 1)),
integral_image), axis=1)
    integral_image = integral_image.astype(int)

    for i in range(1, integral_image.shape[0]):
        for j in range(1, integral_image.shape[1]):
            integral_image[i, j] = np.sum(image[0:i, 0:j])
    return integral_image
```

Με την παρακάτω εντολή γίνεται ο υπολογισμός της εικόνας ολοκλήρωσης ολόκληρης της εικόνας. Πλέον έχουμε τον πίνακα αθροισμάτων για όλη την εικόνα.

```
int_img = my_integral(filtered_img)
```

Ο υπολογισμός του integral ενός παραθύρου μέσα στην εικόνα γίνεται με τον τύπο integral = A + D - B - C, όπου τα A, B, C, D φαίνονται στην παρακάτω εικόνα.



Ο παραπάνω υπολογισμός της μέσης τιμής του integral (μέση τιμή διαβάθμισης του γκρι) για οποιοδήποτε παράθυρο μέσα στην εικόνα γίνεται με την παρακάτω εντολή.

```
mean_graylevel_value = (int_img[y, x] - int_img[h+y, x] - int_img[y, w+x] + int_img[h+y, w+x])/bounding_box_area
```

Με το παρακάτω παράδειγμα μπορεί να γίνει πιο κατανοητό το integral για ένα παράθυρο μέσα στην εικόνα.

Original

5	2	3	4	1
1	**5**	**4**	**2**	3
2	**2**	**1**	**3**	4
3	5	6	4	5
4	1	3	2	6

$5 + 4 + 2 + 2 + 1 + 3 = 17$

Integral

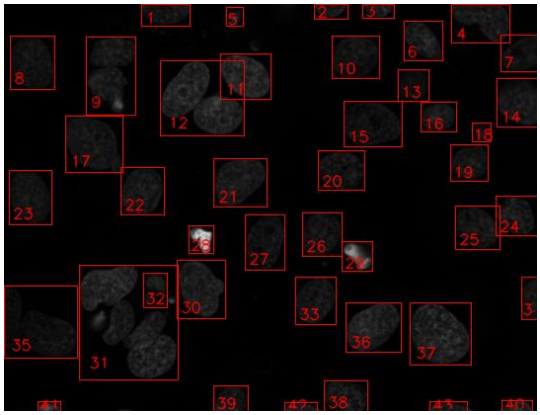
5	7	10	**14**	15
6	13	20	26	30
8	17	25	**34**	42
11	25	39	52	65
15	30	47	62	81

$34 - 14 - 8 + 5 = 17$

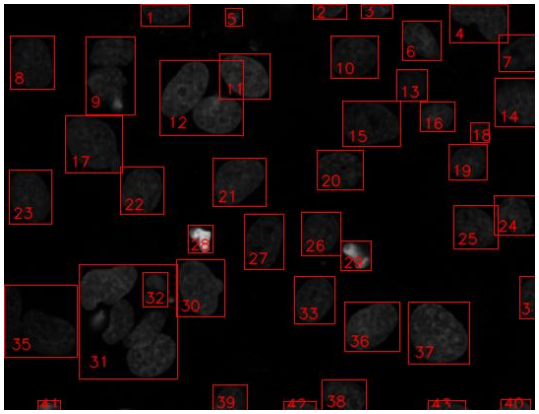
Το πλεονέκτημα του να παράγεται ο πίνακας ολοκλήρωσης για ολόκληρη την εικόνα είναι ότι εφόσον έχουμε έτοιμο τον πίνακα μπορούμε σε σταθερό χρόνο να υπολογίσουμε το integral οποιουδήποτε παραθύρου μέσα στην εικόνα (σε κάθε περίπτωση απαιτούνται 4 αριθμοί). Οπότε, με τον τρόπο αυτό η ταχύτητα εκτέλεσης του υπολογισμού είναι ανεξάρτητη του μεγέθους της υποπεριοχής.

8. Αποτελέσματα εκτύπωσης και σύγκριση αρχικής εικόνας και εικόνας με θόρυβο

Παρακάτω παρατηρούμε ότι το αποτέλεσμα του προγράμματος στην αρχική εικόνα και στην εικόνα στην οποία εφαρμόστηκε φίλτρο median είναι ακριβώς το ίδιο και οι υπολογισμοί που έγιναν είχαν πολύ μικρές διαφορές.

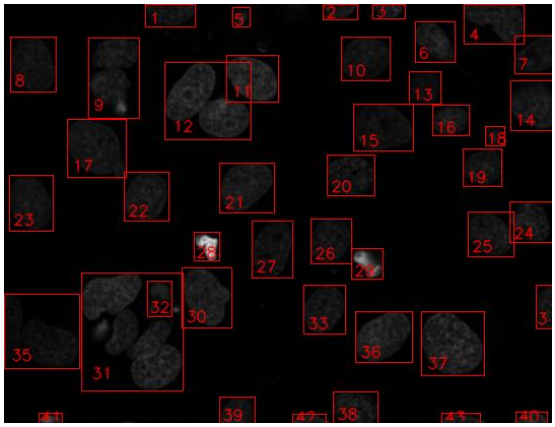


Αρχική εικόνα

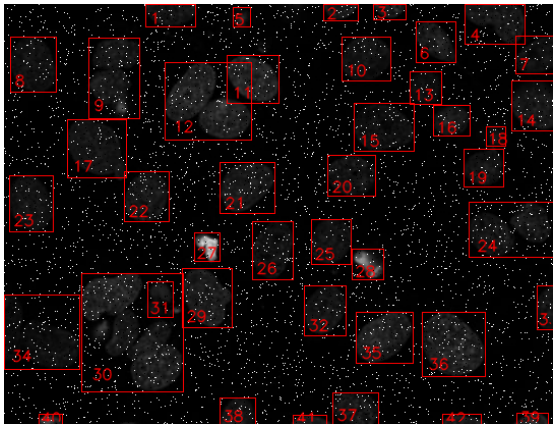


Εικόνα μετά από median

Όσον αφορά την εικόνα με θόρυβο ο αλγόριθμος εντόπισε λανθασμένα τα κύτταρα 24 και 25 που φαίνονται στην αρχική εικόνα, σαν ένα ενιαίο. Οπότε εντόπισε ένα λιγότερο κύτταρο (42 κύτταρα συνολικά αντί για 43). Επομένως, ο θόρυβος μπορεί να οδηγήσει τον αλγόριθμο σε σφάλματα. Επίσης, οι υπολογισμοί που έγιναν έχουν κάποιες διαφορές με τους υπολογισμούς για την αρχική εικόνα, ειδικά στα κύτταρα 24 και 25 η διαφορά θα είναι πολύ μεγάλη.



Αρχική εικόνα



Εικόνα με θόρυβο

Αρχική εικόνα

```
---- Region 1: ----
Area (px): 869
Bounding Box Area (px): 1078
Mean graylevel value in bounding box: 15.275510204081632
---- Region 2: ----
Area (px): 394
Bounding Box Area (px): 510
Mean graylevel value in bounding box: 16.784313725490197
---- Region 3: ----
Area (px): 340
Bounding Box Area (px): 448
Mean graylevel value in bounding box: 18.973214285714285
---- Region 4: ----
Area (px): 1576
Bounding Box Area (px): 2301
Mean graylevel value in bounding box: 15.553237722729248
---- Region 5: ----
Area (px): 245
Bounding Box Area (px): 323
Mean graylevel value in bounding box: 14.356037151702786
```

```
---- Region 6: ----
Area (px): 1025
Bounding Box Area (px): 1560
Mean graylevel value in bounding box: 16.13653846153846
---- Region 7: ----
Area (px): 1042
Bounding Box Area (px): 1369
Mean graylevel value in bounding box: 12.510591672753835
---- Region 8: ----
Area (px): 1884
Bounding Box Area (px): 2430
Mean graylevel value in bounding box: 12.65679012345679
---- Region 9: ----
Area (px): 2549
Bounding Box Area (px): 3950
Mean graylevel value in bounding box: 14.966075949367088
---- Region 10: ----
Area (px): 1645
Bounding Box Area (px): 2064
Mean graylevel value in bounding box: 13.505329457364342
```

Εικόνα με θόρυβο

```
---- Region 1: ----
Area (px): 865
Bounding Box Area (px): 1078
Mean graylevel value in bounding box: 15.275510204081632
---- Region 2: ----
Area (px): 394
Bounding Box Area (px): 544
Mean graylevel value in bounding box: 15.847426470588236
---- Region 3: ----
Area (px): 347
Bounding Box Area (px): 480
Mean graylevel value in bounding box: 17.789583333333333
---- Region 4: ----
Area (px): 1577
Bounding Box Area (px): 2301
Mean graylevel value in bounding box: 15.553237722729248
---- Region 5: ----
Area (px): 245
Bounding Box Area (px): 323
Mean graylevel value in bounding box: 14.356037151702786
```

```
---- Region 6: ----
Area (px): 1024
Bounding Box Area (px): 1560
Mean graylevel value in bounding box: 16.13653846153846
---- Region 7: ----
Area (px): 1041
Bounding Box Area (px): 1369
Mean graylevel value in bounding box: 12.510591672753835
---- Region 8: ----
Area (px): 1887
Bounding Box Area (px): 2430
Mean graylevel value in bounding box: 12.65679012345679
---- Region 9: ----
Area (px): 2549
Bounding Box Area (px): 3950
Mean graylevel value in bounding box: 14.966075949367088
---- Region 10: ----
Area (px): 1647
Bounding Box Area (px): 2064
Mean graylevel value in bounding box: 13.505329457364342
```

Αρχική εικόνα

```
---- Region 1: ----
Area (px): 869
Bounding Box Area (px): 1078
Mean graylevel value in bounding box: 15.275510204081632
---- Region 2: ----
Area (px): 394
Bounding Box Area (px): 510
Mean graylevel value in bounding box: 16.784313725490197
---- Region 3: ----
Area (px): 340
Bounding Box Area (px): 448
Mean graylevel value in bounding box: 18.973214285714285
---- Region 4: ----
Area (px): 1576
Bounding Box Area (px): 2301
Mean graylevel value in bounding box: 15.553237722729248
---- Region 5: ----
Area (px): 245
Bounding Box Area (px): 323
Mean graylevel value in bounding box: 14.356037151702786
---- Region 6: ----
Area (px): 1025
Bounding Box Area (px): 1560
Mean graylevel value in bounding box: 16.13653846153846
---- Region 7: ----
Area (px): 1042
Bounding Box Area (px): 1369
Mean graylevel value in bounding box: 12.510591672753835
---- Region 8: ----
Area (px): 1884
Bounding Box Area (px): 2430
Mean graylevel value in bounding box: 12.65679012345679
---- Region 9: ----
Area (px): 2549
Bounding Box Area (px): 3950
Mean graylevel value in bounding box: 14.966075949367088
---- Region 10: ----
Area (px): 1645
Bounding Box Area (px): 2064
Mean graylevel value in bounding box: 13.505329457364342
```

Εικόνα μετά από median

```
---- Region 1: ----
Area (px): 866
Bounding Box Area (px): 1078
Mean graylevel value in bounding box: 15.275510204081632
---- Region 2: ----
Area (px): 392
Bounding Box Area (px): 510
Mean graylevel value in bounding box: 16.784313725490197
---- Region 3: ----
Area (px): 346
Bounding Box Area (px): 448
Mean graylevel value in bounding box: 18.973214285714285
---- Region 4: ----
Area (px): 1574
Bounding Box Area (px): 2301
Mean graylevel value in bounding box: 15.553237722729248
---- Region 5: ----
Area (px): 234
Bounding Box Area (px): 306
Mean graylevel value in bounding box: 14.823529411764707
---- Region 6: ----
Area (px): 1023
Bounding Box Area (px): 1560
Mean graylevel value in bounding box: 16.13653846153846
---- Region 7: ----
Area (px): 1032
Bounding Box Area (px): 1369
Mean graylevel value in bounding box: 12.510591672753835
---- Region 8: ----
Area (px): 1878
Bounding Box Area (px): 2430
Mean graylevel value in bounding box: 12.65679012345679
---- Region 9: ----
Area (px): 2548
Bounding Box Area (px): 3950
Mean graylevel value in bounding box: 14.966075949367088
---- Region 10: ----
Area (px): 1642
Bounding Box Area (px): 2064
Mean graylevel value in bounding box: 13.505329457364342
```

9.Βιβλιογραφία

1. https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Image%20Filtering.pdf
2. https://www.researchgate.net/figure/Concept-of-median-filtering-in-image-processing_fig17_281534044
3. https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html
4. https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT2/node3.html
5. https://towardsdatascience.com/a-visual-introduction-to-binary-image-processing-part-1-d2fba9f102a4
6. https://towardsdatascience.com/implementing-a-connected-component-labeling-algorithm-from-scratch-94e1636554f
7. https://docs.opencv.org/4.x/modules.html
8. https://www.projectpro.io/article/computer-vision-algorithms-and-applications/514
9. https://levelup.gitconnected.com/the-integral-image-4df3df5dce35
10. https://numpy.org/doc/stable/reference/generated/numpy.cumsum.html
11. https://www.mathworks.com/help/images/integral-image.html
12. https://blog.finxter.com/numpy-cumsum/
13. Διαφάνειες μαθήματος “ Όραση Υπολογιστών”, Δημοκρίτειο Πανεπιστήμιο Θράκης