ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΝΣΩΜΑΤΩΜΕΝΟ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ ΣΥΣΤΗΜΑ ΓΙΑ ΑΝΙΧΝΕΥΣΗ ΟΠΩΝ ΣΕ ΔΙΧΤΥΑ ΙΧΘΥΟΚΑΛΛΙΕΡΓΕΙΩΝ

Νικόλαος Μπαδογιάννης

Επιτροπή:

Επιβλέπων: Καθηγητής Απόστολος Δόλλας
Μέλος Επιτροπής: Καθηγητής Μιχαήλ Ζερβάκης
Μέλος Επιτροπής: Δρ. Νικόλαος Παπανδρουλάκης (ΕΛΚΕΘΕ)

Χανιά,Μάρτιος 2019

Περίληψη

Τεχνικές επεξεργασίας εικόνας χρησιμοποιούνται συχνά από τη βιομηχανία για τον ποιοτικό έλεγχο προϊόντων. Η πολυπλοκότητα τους καθιστά απαραίτητη την αναζήτηση μεθόδων επιτάχυνσης των τεχνικών αυτών σε hardware ώστε να είναι εφικτή η χρήση τους σε ενσωματωμένα συστήματα πραγματικού χρόνου με μικρή κατανάλωση ενέργειας.

Ένα τέτοιο πρόβλημα μελετά η παρούσα διπλωματική, καθώς αναζητούνται τεχνικές οι οποίες θα οδηγήσουν στην ανίχνευση οπών σε δίχτυα ιχθυοκαλλιεργειών. Στόχος της παρούσας διπλωματικής είναι η σχεδίαση ενός ενσωματωμένου συστήματος πραγματικού χρόνο με χαμηλό χρόνο εκτέλεσης και κατανάλωση ενέργειας. Αφού μοντελοποιήθηκαν αλγόριθμοι που χρησιμοποιούν τις μεθόδους Template Matching και Edge Detection σε Matlab, έγιναν πειράματα για να διαπιστωθεί κατά πόσο είναι εφικτή η χρήση τους σε ένα τέτοιο πρόβλημα. Έπειτα υλοποιήθηκαν σε FPGA για την εξαγωγή συμπερασμάτων όσο αφορά την κατανάλωση ενέργειάς, την απαιτούμενη δέσμευση πόρων και το συνολικό χρόνο εκτέλεσης.

Abstract

Image processing techniques are used by the industry for the quality control of the products. Their complexity demands research on methods that can accelerate these techniques on hardware and make possible their use on real time embedded systems with low power consumption.

This thesis diploma studies image processing methods for the detection of holes on fish farming nets. The main purpose of this thesis is to design a real time embedded system with low execution time and power consumption. Once implemented algorithms that use Template Matching and Edge Detection techniques, we studied them experimentally using Matlab to detect if they can be used for this purpose. Then we implemented our algorithms on FPGA and checked their demands on resources, power consumption and total execution time.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κύριο Απόστολο Δόλλα για τις πολύτιμες γνώσεις που μου μετέδωσε κατά τη διάρκεια των σπουδών μου και την καθοδήγηση του κατά την εκπόνηση της διπλωματικής εργασίας. Επιπλέον θα ήθελα να ευχαριστήσω την κυρία Ντίνα Μοιρογιώργου για τις συμβουλές της όσον αφορά την επεξεργασία εικόνας και τον Παύλο Μαλακωνάκη για την βοήθεια του κατά την υλοποίηση του συστήματος.

Τέλος, οφείλω να ευχαριστήσω την οικογένειά μου για τις θυσίες τους και τη στήριξή τους όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Το πρόβλημα

Στη σύγχρονη βιομηχανική εποχή είναι απαραίτητη η χρήση μεθόδων για τον ποιοτικό έλεγχο των προϊόντων αλλά και των μηχανημάτων παραγωγής τους. Συχνά χρησιμοποιούνται αυτοματοποιημένες τεχνικές για την άμεση εξακρίβωση βλαβών, ώστε να μειωθεί η παραγωγή ελαττωματικών προϊόντων αλλά και να αποφευχθούν πιθανοί κίνδυνοι που μπορεί να προκαλέσουν αυτές.

Τεχνικές επεξεργασίας εικόνας χρησιμοποιούνται συχνά από βιομηχανία για την αυτοματοποίηση του ποιοτικού ελέγχου. Στόχος τους είναι η εξακρίβωση βλαβών και η ανίχνευση ελαττωμάτων στο παραγόμενο προϊόν με τη χρήση συστημάτων τα οποία διαθέτουν κάποιο τύπο κάμερας. Στο σύστημα ελέγχου δίνονται πληροφορίες που περιγράφουν το προϊόν με το επιθυμητό αποτέλεσμα και συγκρίνονται κάθε φορά με την εικόνα που έχει ληφθεί. Η αυτοματοποίηση είναι σημαντική διότι πολλές φορές είναι δύσκολος ή χρονοβόρος ο έλεγχος από τον άνθρωπο ή είναι μεγάλη η πιθανότητα λάθους συμπεράσματος από αυτόν.

Μία τέτοια περίπτωση αποτελεί και η εύρεση οπών σε δίχτυα ιχθυοκαλλιεργειών. Στις ιχθυοκαλλιέργειες χρησιμοποιούνται ειδικοί περίφρακτοι χώροι κατασκευασμένοι από πασσάλους και δίχτυα με στόχο εγκλωβισμό των ψαριών ώστε είναι δυνατή η εκτροφή και αλιεία τους. Η ύπαρξη οπών στα δίχτυα μπορεί να οδηγήσει στην απώλεια κάποιου αριθμού ψαριών. Ο συστηματικός τους έλεγχος είναι σημαντικός αλλά και ιδιαίτερα δύσκολος και χρονοβόρος. Αυτό οφείλεται στο γεγονός ότι τα δίχτυα βρίσκονται μέσα στη θάλασσα και για να ελεγχθούν θα πρέπει είτε να απομακρυνθούν από το νερό είτε να ελεγχθούν μέσα σε αυτό από κάποιο δύτη. Η αυτοματοποίηση αυτή της διαδικασίας θα βοηθήσει στη μείωση του χρόνου και κόστους συντήρησης των διχτυών στις ιχθυοκαλλιέργειες.

1.2 Συνεισφορά διπλωματικής

Στόχος αυτής της διπλωματικής εργασίας είναι η μελέτη μεθόδων επεξεργασίας εικόνας για την κατασκευή ενός ενσωματωμένου συστήματος το οποίο ανιχνεύει τις οπές στα δίχτυα ιχθυοκαλλιεργειών. Το σύστημα θα λαμβάνει εικόνες και θα ελέγχει αν το κομμάτι του διχτυού το οποίο απεικονίζεται περιέχει κάποια οπή.

Το ενσωματωμένο σύστημα που μελετάει η εργασία θα υλοποιηθεί σε FPGA για να είναι φορητό και ταυτόχρονα γρήγορο, ώστε να είναι δυνατή η τοποθέτηση του σε υποβρύχιο όχημα για την μερική ή και ολική αυτοματοποίηση της διαδικασίας ελέγχου των διχτυών. Οι βασικές προϋποθέσεις του συστήματος είναι η μειωμένη κατανάλωση ενέργειας και η εξαγωγή συμπερασμάτων σε όσο το δυνατόν λιγότερο χρόνο. Κατά την ολοκλήρωση της διαδικασίας επεξεργασίας θα εξάγεται εικόνα που θα απεικονίζει τα σημείο όπου εντοπίστηκε οπή. Ακόμα και στην περίπτωση όπου τα αποτελέσματα δεν είναι απολύτως ακριβή ,θα μειώσει αρκετά το χρόνο απασχόλησης και τον αριθμό του ανθρώπινου δυναμικού που απαιτείται για τη διαδικασία ελέγχου των διχτυών.

1.3 Δομή διπλωματικής

Η παρούσα διπλωματική ακολουθεί την παρακάτω δομή:
Κεφάλαιο 2:Γίνεται αναφορά σε μεθόδους επεξεργασίας εικόνας καθώς και σε δημοσιεύσεις οι οποίες λύνουν παρόμοια προβλήματα.
Κεφάλαιο 3:Απαρτίζεται από τη μελέτη μοντελοποίησης του προβλήματος, συγκρίνοντας αλγορίθμους εντοπισμού οπών που υλοποιήθηκαν.
Κεφάλαιο 4:Περιγράφεται η σχεδίαση και υλοποίηση του συστήματος σε FPGA.
Κεφάλαιο 5:Γίνεται πειραματικός έλεγχος του τελικού συστήματος και αξιολογούνται τα αποτελέσματα του.
Κεφάλαιο 6:Αποτελείται από τη σύνοψη της διπλωματικής και αναφέρονται τα συμπεράσματα που προέκυψαν από αυτή.

Κεφάλαιο 2

Σχετική Έρευνα

2.1 Επεξεργασία εικόνας

Λόγω της φύσης του προβλήματος είναι απαραίτητη η μελέτη τεχνικών επεξεργασίας εικόνας οι οποίες θα μας δώσουν τη δυνατότητα να εντοπίσουμε τις οπές στα δίχτυα. Αρχικά αναλύονται τεχνικές συσχέτισης προτύπου (Template Matching) και μέθοδοί κατάτμησης της εικόνας με Edge Detection. Έπειτα μελετούνται τεχνικές εντοπισμού ανομοιομορφιών σε επαναλαμβανόμενα μοτίβα (Regular/Near-Regular Textures).

2.1.1 Template Matching

Μια τεχνική επεξεργασίας εικόνας η οποία χρησιμοποιείται συχνά από τη βιομηχανία για τον ποιοτικό έλεγχο των αντικειμένων είναι η Template Matching [9].

Οι τεχνικές Template Matching λαμβάνουν ως δεδομένα μια πηγαία εικόνα (Source Image (I)), δηλαδή την εικόνα που μελετάται και την εικόνα πρότυπο (Template Image (T)) η οποία περιέχει την πληροφορία που επιθυμούμε να εντοπίσουμε στην πηγαία εικόνα.

Η βασική ιδέα της μεθόδου είναι ότι δίνουμε στο σύστημα τις παραπάνω εικόνες με στόχο τον εντοπισμό των σημείων της πηγαίας εικόνας που ταιριάζουν με το πρότυπο. Για να το επιτύχουμε αυτό, μεταφέρεται το πρότυπο πάνω σε όλες τις πιθανές θέσεις της εικόνας προς επεξεργασία, υπολογίζοντας κάθε φορά το βαθμό συσχέτισης ή διαφοροποίησης μεταξύ των pixel των δύο εικόνων. Στο τέλος αναζητούμε τις θέσεις όπου το αποτέλεσμα του μέτρου σύγκρισης είναι μεγαλύτερο από την τιμή που έχουμε ορίσει ως κατώφλι.

Τα βήματα που ακολουθούνται κατά την εφαρμογή Template Matching είναι τα εξής:

1. Επιλέγεται η εικόνα προς επεξεργασία.

2. Μετατρέπεται από έγχρωμη σε εικόνα με αποχρώσεις του γρι.

3. Επιλέγεται ένα πρότυπο,το οποίο πιθανόν να είναι ένα κομμάτι της αρχικής εικόνας.

4. Εφαρμόζεται μία από τις τεχνικές συσχέτισης που περιγράφονται παρακάτω.

5. Εντοπίζεται το σημείο που παρουσιάζει μέγιστη συσχέτιση.

6. Εμφανίζονται τα αποτελέσματα.

Ακολουθούν βασικές τεχνικές συσχέτισης που χρησιμοποιούνται στο Template Matching[4]. Στις συναρτήσεις που παρουσιάζονται παρακάτω, με T συμβολίζεται το πρότυπο και με I η εικόνα προς επεξεργασία.

Sum of Absolute Differences:
Σε αυτή τη μέθοδο συσχέτισης υπολογίζεται το σύνολο της απόλυτης διαφοράς μεταξύ των pixel του προτύπου και των pixel της εικόνας που επεξεργαζόμαστε. Η μέθοδος αυτή απαιτεί απλές μαθηματικές πράξεις και είναι η πιο φτηνή σε υπολογιστικό κόστος συγκριτικά με τις μεθόδους που ακολουθούν.

$$R(x,y) = \sum_{x',y'} abs(T(x',y') - I(x+x',y+y'))$$

Sum of Squared Differences:
Μία παραλλαγή της παραπάνω μεθόδου είναι η Sum of Squared Differences όπου εκεί υπολογίζεται το σύνολο διαφοράς μεταξύ των pixel του προτύπου και των pixel της εικόνας που επεξεργαζόμαστε, υψωμένη στο τετράγωνο.

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

Στα αποτελέσματα των παραπάνω μεθόδων αναζητούμε τις μικρότερες δυνατές τιμές, διότι αυτές υποδηλώνουν την ελάχιστη διαφοροποίηση μεταξύ του template και της εικόνας προς επεξεργασία.

Cross-Correlation:
Άλλη μία μέθοδος σύγκρισης του προτύπου με την εικόνα είναι η Cross-Correlation η οποία υπολογίζει το άθροισμα του γινομένου όλων των pixel των δύο εικόνων.

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

Σε αυτή την περίπτωση, αντίθετα από τις δύο προηγούμενες μεθόδους, μεγαλύτερη συσχέτιση παρουσιάζουν τα σημεία όπου το αποτέλεσμα της συνάρτησης είναι μεγαλύτερο.

Οι παραπάνω μέθοδοι επηρεάζονται σημαντικά από τις αλλαγές της φωτεινότητας και την ύπαρξη θορύβου. Για την αντιμετώπιση αυτών των προβλημάτων είναι απαραίτητη η ομαλοποίηση των αποτελεσμάτων με την χρήση ομαλοποιημένων μεθόδων [7].

Normalized Sum of Squared Differences:
Η μέθοδος Normalized Sum of Squared Differences κάνει ομαλοποίηση διαιρώντας τα αποτελέσματα της Sum of Squared Differences με την τετραγωνική ρίζα του γινομένου των αθροισμάτων των τετραγωνικών τιμών κάθε pixel των δύο εικόνων.

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

Normalized Cross-Correlation:
Η Normalized Cross-Correlation είναι μία βελτιωμένη έκδοση της τεχνικής Cross-Correlation η οποία δεν επηρεάζεται από τις διαφορές στην φωτεινότητα. Αυτό οφείλεται στο γεγονός ότι κατά τον υπολογισμό συμπεριλαμβάνονται και οι μέσες τιμές του τμήματος της εικόνας και του προτύπου. Οι τιμές που υπολογίζονται από την τεχνική αυτή είναι στο διάστημα [-1,1]. Το 1 συμβολίζει ότι οι δύο εικόνες ταιριάζουν απόλυτα ενώ το -1 ότι είναι εντελώς διαφορετικές.

$$R(x,y) = \frac{\sum_{x',y'}[T(x',y') - \overline{T}] \cdot [I(x+x',y+y') - \overline{I_{x,y}}]}{\sqrt{\sum_{x',y'}[T(x',y') - \overline{T}]^2 \cdot \sum_{x',y'}[I(x+x',y+y') - \overline{I_{x,y}}]^2}}$$

Υπάρχουν και άλλες μέθοδοι σύγκρισης όπως για παράδειγμα η Cross-Correlation Coefficient και η Normalized Cross-Correlation Coefficient οι οποίες όμως είναι υπολογιστικά πιο ακριβές από τις μεθόδους που μελετήθηκαν παραπάνω.

2.1.2 Τεχνικές Edge Detection

Η κατάτμηση της εικόνας σε τμήματα είναι το αρχικό βήμα για την εξαγωγή χαρακτηριστικών της εικόνας. Αποτελεί μία διαδικασία χωρίσματος της εικόνας σε περιοχές που έχουν διαφορετικά χαρακτηριστικά, όπως είναι το χρώμα, τα επίπεδα του γκρι και το texture. Μια τεχνική που χρησιμοποιείται συχνά για αυτόν το σκοπό είναι η Edge Detection, η οποία λαμβάνει μέρος σε πληθώρα εφαρμογών όπως τον εντοπισμό και την αναγνώριση αντικειμένων, την βελτίωση εικόνων κ.α. [11].

Οι τεχνικές Edge Detection μετατρέπουν τις εικόνες που επεξεργάζονται σε εικόνες με ακμές εκμεταλλευόμενες τις διαφορές στα επίπεδα του γκρι μέσα σε αυτές. Οι ακμές αποτελούν δείγμα έλλειψης συνοχής στην εικόνα, είναι δηλαδή σημεία όπου η τιμή της εικόνας αλλάζει απότομα[13].
Η διαδικασία εντοπισμού ακμών αποτελείται από τρία βασικά μέρη:

1. Το φιλτράρισμα της εικόνας για την αφαίρεση πιθανού θορύβου. Η υπερβολική χρήση όμως μεθόδων αφαίρεσης θορύβου με φίλτρα εξομάλυνσης έχει σαν αποτέλεσμα το θόλωμα της εικόνας και τη μείωση της διαφοροποίησης μεταξύ των τιμών της, με αποτέλεσμα να είναι πιο δύσκολος ο εντοπισμός των ακμών.

2. Τη βελτίωση της εικόνας, όπου τονίζονται τα pixel που παρουσιάζουν σημαντικές τοπικές διαφοροποιήσεις στην ένταση με χρήση μεθόδων υπολογισμού του μεγέθους της διαβάθμισης.

3. Την αναγνώριση των ακμών, ανιχνεύοντας τις απότομες τοπικές μεταβολές έντασης στην εικόνα. Η ύπαρξη διαφοροποίησης δεν αποτελεί πάντοτε ακμή, επομένως συγκρίνονται οι τιμές με κάποιο κατώφλι.

Η ανίχνευση ακμών πραγματοποιείται χρησιμοποιώντας παραγώγους πρώτης ή δεύτερης τάξεως[3]. Με τη χρήση παραγώγων πρώτης τάξεως υπολογίζεται η κλίση της εικόνας από τον τύπο:

$$\bigtriangledown f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

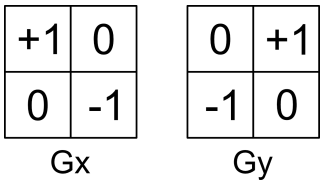
Το διάνυσμα που υπολογίζεται δείχνει πάντοτε προς τη διεύθυνση του μεγαλύτερου ρυθμού μεταβολής της f στη θέση (x,y). Το μέτρο του παραπάνω διανύσματος εκφράζει την τιμή του ρυθμού μεταβολής κατά τη διεύθυνση του διανύσματος κλίσης και υπολογίζεται από τον τύπο:

$$M(x, y) = mag(\bigtriangledown f) = \sqrt{g_x^2 + g_y^2}$$

Για κάθε pixel της εικόνας υπολογίζεται ο ρυθμός μεταβολής, λαμβάνοντας υπόψιν τα γειτονικά pixel. Κάθε τεχνική ανίχνευσης διαθέτει τους δικούς τις τελεστές οι οποίοι εκφράζουν τις πράξεις που θα γίνουν στην περιοχή γύρω από κάθε pixel για τον υπολογισμό της κλίσης της εικόνας. Ο τελεστής g_x είναι υπεύθυνος για τις πράξεις στον άξονα x ενώ ο g_y για τον y.

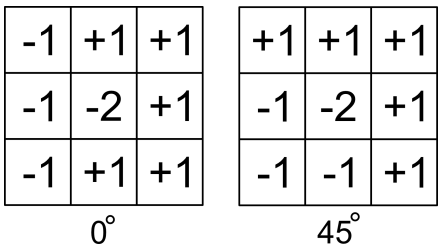
Παρακάτω περιγράφονται τρεις από τις βασικότερες τεχνικές εντοπισμού ακμών:

Roberts Detection:
Η τεχνική Roberts είναι απλή και γρήγορη και χρησιμοποιεί μάσκες διαστάσεων 2x2 οι οποίες όμως δεν θεωρούνται πολύ χρήσιμες για τον υπολογισμό διευθύνσεων ακμών, συγκριτικά με άλλες μεθόδους που χρησιμοποιούν μεγαλύτερες μάσκες.



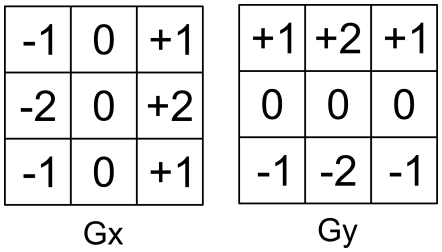
Σχήμα 2.1: Robert Masks

Prewitt Detection:
Η μέθοδος Prewitt είναι πιο πολύπλοκη, διότι χρησιμοποιεί μάσκες διαστάσεων 3x3 και μπορεί να υπολογίσει διευθύνσεις ακμών προς οχτώ κατευθύνσεις.



Σχήμα 2.2: Prewitt Masks

Sobel Detection:
Οι υπολογισμοί με τη χρήση του Sobel γίνονται με τη χρήση μασκών διαστάσεων 3x3 και υπολογίζονται διευθύνσεις ακμών κάθετα και οριζόντια.

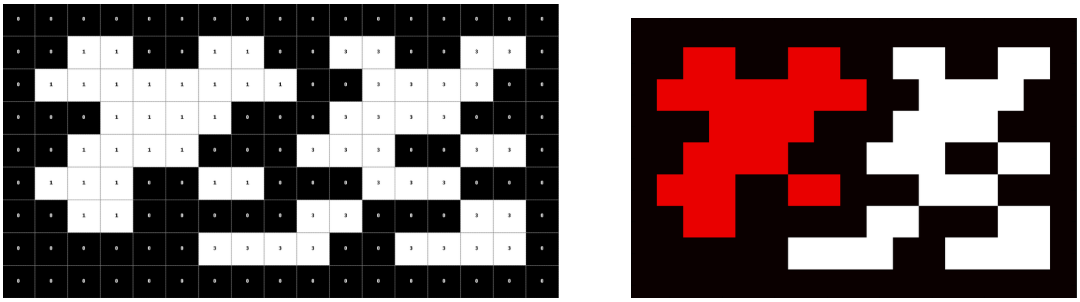


Σχήμα 2.3: Sobel Masks

Υπάρχουν και πιο προηγμένες μέθοδοι ανίχνευσης ακμών, όπως η Marr-Hildreth και Canny οι οποίες λαμβάνουν υπόψη παράγοντες όπως το θόρυβος της εικόνας και τη φύση των ακμών [3]. Αυτές οι μέθοδοι όμως απαιτούν πιο πολύπλοκες πράξεις και περισσότερο χρόνο εκτέλεσης, σε σχέση με αυτές που μελετήθηκαν παραπάνω.

2.1.3 Connected Component Labeling

Ο Connected Component Labeling είναι ένας βασικός αλγόριθμος επεξεργασίας εικόνας που χρησιμοποιείται σε εφαρμογές αναγνώρισης αντικειμένων [12]. Στόχος του είναι η ομαδοποίηση των pixel που είναι συνδεδεμένα μεταξύ τους και ανήκουν στο ίδιο αντικείμενο. Για κάθε pixel της εικόνας, ελέγχονται τα γειτονικά του για τον έλεγχο της σύνδεσης τους. Σε μία δυαδική εικόνα, δύο pixel θεωρούνται συνδεδεμένα όταν έχουν την ίδια τιμή, δηλαδή είναι και τα δύο μαύρα ή άσπρα. Οι βασικές



Σχήμα 2.4: Παράδειγμα Connected Component Labeling

μέθοδοι για Connected Component Labeling είναι οι εξής:

Multipass methods:
Σε αυτή τη μέθοδο χρησιμοποιείται μία μάσκα, το μέγεθος της οποίας καθορίζει ποια pixel γύρω από το κεντρικό θα ελεγχθούν. Η εικόνα αρχικά σαρώνεται από αριστερά προς τα δεξιά ξεκινώντας από την κορυφή. Έπειτα επαναλαμβάνεται η ίδια διαδικασία από κάτω και προς την αντίθετη κατεύθυνση. Κατά το πρώτο πέρασμα, αν κανένα σημείο δεν είναι συνδεδεμένο με το κέντρο της μάσκας, ορίζεται μία νέα ταμπέλα, διαφορετικά λαμβάνει την μικρότερη τιμή των σημείων μέσα στη μάσκα. Η σάρωση προς τα μπροστά και προς τα πίσω επαναλαμβάνεται έως ότου σταματήσουν να συμβαίνουν αλλαγές. Τότε η ανάθεση ταμπελών έχει ολοκληρωθεί. Εικόνες με πολύπλοκη δομή απαιτούν μεγάλο αριθμό περασμάτων.

Two pass methods:
Σε αυτή την περίπτωση απαιτούνται δύο σαρώσεις της εικόνας. Το κεντρικό σημείο της μάσκας λαμβάνει την ελάχιστη τιμή ταμπέλας από τα υπόλοιπα σημεία που

βρίσκονται μέσα σε αυτή. Αν κανένα pixel δεν συνδέεται με το κεντρικό, τότε λαμβάνει μία νέα ταμπέλα. Στην περίπτωση όπου δύο σημεία στη μάσκα έχουν διαφορετική ταμπέλα, τότε το κεντρικό λαμβάνει την μικρότερη τιμή και σε ένα πίνακα (look up table) αποθηκεύεται η πληροφορία ότι οι δύο διαφορετικές ταμπέλες ανήκουν στο ίδιο αντικείμενο. Κατά τη δεύτερη σάρωση της εικόνας, αντικαθίστανται οι τιμές χρησιμοποιώντας την πληροφορία που υπάρχει στον LUT.

One pass methods:
Αυτού του είδους οι μέθοδοι εξάγουν απευθείας χαρακτηριστικά των συνδεδεμένων αντικειμένων ως ένα κουτί πλαίσιο ή ως κέντρο βαρύτητας, χωρίς την παραγωγή label mask. Εκμεταλλεύονται το γεγονός ότι, όταν απαιτείται η ανάθεση ταμπέλας σε λίγα αντικείμενα, τότε είναι εφικτό να αποθηκευτούν οι πληροφορίες για αυτά σε μικρό κομμάτι μνήμης.

Υπάρχουν και άλλες μέθοδοι Connected Component Labeling οι οποίες είναι βασισμένες σε τεχνικές region-growing ή contour-tracing. Αυτές αναθέτουν ταμπέλα σε κάθε αντικείμενο ξεχωριστά, το ένα μετά το άλλο και έχουν πολύ δυναμική ροή προγράμματος αλλά δεν είναι ιδανικές για χρήση σε hardware.

2.1.4 Regular texture images

Με τον όρο Regular texture images περιγράφουμε εικόνες οι οποίες αποτελούνται από επαναλήψεις του ίδιου σχήματος ή αντικειμένου. Πολλά αντικείμενα παρουσιάζουν μία περιοδικότητα και ένα από αυτά είναι και τα δίχτυα τα οποία αποτελούνται από επαναλαμβανόμενα εξάγωνα συνδεδεμένα μεταξύ τους. Στη βιομηχανία υπάρχουν αυτοματοποιημένοι μέθοδοι για τον ποιοτικό έλεγχο τέτοιων αντικειμένων όπως για παράδειγμα υφάσματα, χαλιά και ξύλο.

Στη δημοσίευση [1] παρουσιάζεται ένας αλγόριθμος για την αναγνώριση ελαττωμάτων σε εικόνες με επαναλαμβανόμενα μοτίβα. Η δημοσίευση εφαρμόζει ανάλυση προεξοχών με φασματικό υπόλοιπο, όπου υποθετικά η εικόνα αποτελείται από ένα καινοτόμο κομμάτι (ελαττώματα) και ένα περιττό (επαναλαμβανόμενο μοτίβο). Δεδομένης μίας εικόνας I(x) ο εντοπισμός των ελαττωμάτων γίνεται ως εξής:

$$A(f) = \Re\left(F(I(x))\right) \quad (1)$$
$$P(f) = \Im\left(F(I(x))\right) \quad (2)$$
$$L(f) = log\left(A(f)\right) \quad (3)$$
$$R(f) = L(f) - h_n(f) * L(f) \quad (4)$$
$$EM(x) = g(x) * \|F^{-1}[e^{R(f)+i\cdot P(f)}]\|^2 \quad (5)$$

Σχήμα 2.5: Πράξεις για τον εντοπισμό ελαττωματικών τμημάτων σε εικόνα τύπου Regular texture [1]

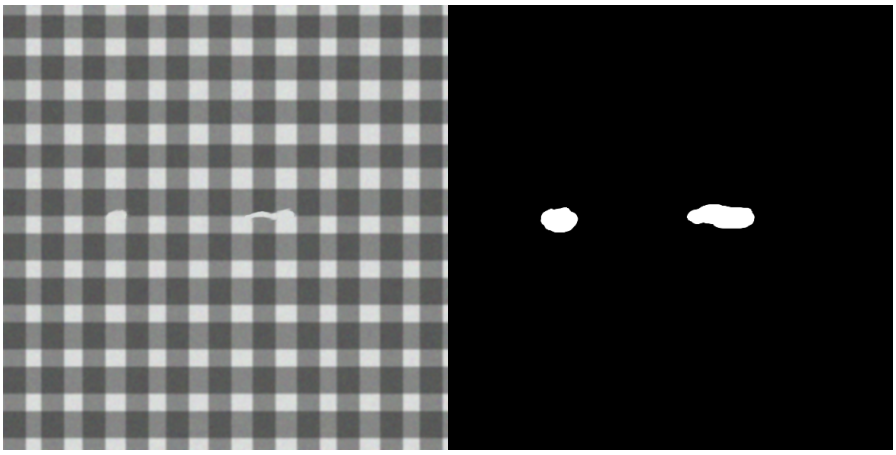
Με F συμβολίζεται ο μετασχηματισμός Fourier. Η περιττή πληροφορία εξομοιώνεται χρησιμοποιώντας το φίλτρο τοπικής μέσης τιμής στην εικόνα 2.6 όπου για n χρησιμοποιείται η τιμή τρία.

$$h_n(f) = \frac{1}{n^2}\begin{pmatrix} 1 & 1 & \dots \\ 1 & 1 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

Σχήμα 2.6: Τύπος μέσης τιμής [1]

Έπειτα ορίζεται μία τιμή κατώφλι και με βάση αυτή εντοπίζονται τα ελλειμματικά σημεία. Το κατώφλι εξαρτάται από τα χαρακτηριστικά της εικόνας. Γενικά ο παραπάνω αλγόριθμος είναι αποδοτικός υπολογιστικά και αξιόπιστος. Στην εικόνα 2.7 φαίνονται

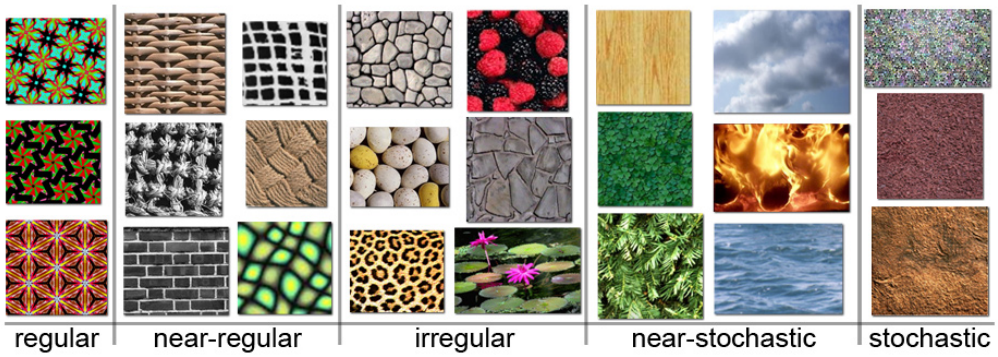
τα αποτελέσματα του παραπάνω αλγορίθμου.



Σχήμα 2.7: Αποτελέσματα δημοσίευσης [1]

2.1.5 Near-Regular texture images

Εικόνες Near-Regular texture [8] ονομάζονται αυτές που αποτελούνται από επαναλήψεις του ίδιου αντικειμένου ή σχήματος αλλά δεν είναι απόλυτα περιοδικές. Μακροσκοπικά φαίνονται κανονικοποιημένες αλλά τοπικά παρουσιάζουν τυχαία χαρακτηριστικά, δηλαδή σημεία της εικόνας παρουσιάζουν διαφοροποιήσεις ως προς τη γεωμετρία ή τα αντικείμενα πιθανόν να μην είναι απολύτως ίδια. Στη εικόνα 2.8 φαίνονται οι διάφοροι τύποι Texture.



Σχήμα 2.8: Texture Spectrum [8]

Η δημοσίευση [10] μελετά την αυτόματη αναγνώριση επαναλαμβανόμενων μοτίβων σε τέτοιες εικόνες. Στόχος της είναι η κατασκευή ενός παραμορφωμένου πλέγματος που συνδέει τα επαναλαμβανόμενα σημεία μέσα σε μία εικόνα χρησιμοποιώντας την μέθοδο Mean-Shift Belief Propagation. Ο τρόπος προσέγγισης του προβλήματος έχει πολλά πλεονεκτήματα συγκριτικά με προηγούμενες δουλειές. Κάποια από αυτά είναι ότι:

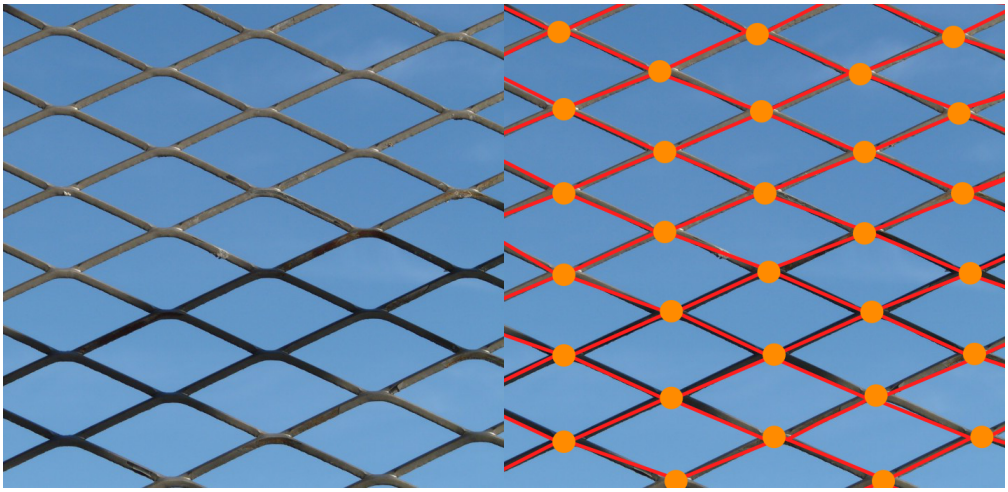
1. Η ενσωμάτωση περιορισμών υψηλότερης τάξης από νωρίς επιτρέπει την κατασκευή αξιόπιστων πλεγμάτων.

2. Είναι εφικτή η ανάπτυξη του πλέγματος προς πολλές κατευθύνσεις ταυτόχρονα.

3. Επιτυγχάνονται πιο ακριβή αποτελέσματα από άλλους αλγορίθμους (π.χ. state-of-the-art).

Η μέθοδος κατασκευής του πλέγματος αποτελείται από τρεις φάσεις:

1. Αρχικά εξάγονται τα σημεία που παρουσιάζουν ενδιαφέρον χρησιμοποιώντας τη μέθοδο Kanade–Lucas–Tomasi feature tracker και στη συνέχεια υπολογίζεται η δομή του πλέγματος λαμβάνοντας υπόψη σε κάθε σημείο δύο γειτονικά σημεία. Με αυτόν το τρόπο κατασκευάζεται ένα ζεύγος διανυσμάτων που αποτελούν



Σχήμα 2.9: Δείγμα αποτελεσμάτων δημοσίευσης [10]

την κατεύθυνση των γειτονικών σημείων. Η ομοιότητα τους υπολογίζεται με τη χρήση Normalized Cross Correlation.

2. Τα διανύσματα που υπολογίστηκαν, χρησιμοποιούνται για την κατασκευή του πλέγματος με τη βοήθεια του αλγορίθμου Mean-Shift Belief Propagation, ο οποίος χρησιμοποιείται για τον καλύτερο υπολογισμό της θέσης των στοιχείων πάνω στο texture.

3. Τέλος χρησιμοποιούνται παραμορφωμένες εκδοχές του πλέγματος και μέθοδοι template matching για την εξέταση όλης της εικόνας, ώστε να εντοπιστούν τμήματα τα οποία δεν είχαν εντοπιστεί πριν, μέχρι το πλέγμα φτάσει στην άκρη της εικόνας.

2.2 Image Processing On FPGA

Στις μέρες μας συνεχώς εμφανίζονται νέες εφαρμογές επεξεργασίας εικόνας που απαιτούν περισσότερη επεξεργαστική ισχύ από αυτή που είναι διαθέσιμη [5]. Παρόλο που οι υπολογιστές γίνονται όλο και γρηγορότεροι, συχνά δεν μπορούν να αντεπεξέλθουν σε τέτοιου είδους εφαρμογές και υπάρχει η ανάγκη για την ανάπτυξη τεχνικών επιτάχυνσης των εφαρμογών επεξεργασίας εικόνας στο hardware.

Σε πολλές εφαρμογές επεξεργασίας εικόνας απαιτείται η επεξεργασία μεγάλου όγκου δεδομένων και η πραγματοποίηση πολύπλοκων εργασιών. Για κάθε pixel της εικόνας χρειάζεται να γίνουν αρκετοί υπολογισμοί και όσο μεγαλύτερη είναι η εικόνα τόσο μεγαλώνει ο όγκος τους. Για αυτό το λόγο η επεξεργασία εικόνας σε πραγματικό χρόνο είναι δύσκολο να επιτευχθεί από ένα σειριακό επεξεργαστή [2]. Συχνά

για τέτοιου είδους εφαρμογές χρησιμοποιούνται Field-Programmable Gate Arrays. Οι FPGA αποτελούνται από πλέγματα κομματιών λογικής, συνδεδεμένα με προγραμματιζόμενα καλώδια. Κάθε κομμάτι έχει έναν οι περισσότερους πίνακες αληθείας και μερικά bits μνήμης. Οι FPGA μπορούν να χρησιμοποιηθούν για την εφαρμογή κυκλωμάτων συνδέοντας πύλες και καταχωρητές.

Στο κομμάτι που υπερτερούν οι FPGA από τους επεξεργαστές είναι ο παραλληλισμός. Μας δίνεται η δυνατότητα δηλαδή να κάνουμε πολλούς υπολογισμούς ταυτόχρονα σε όλη την εικόνα ή σε κομμάτια της. Επίσης χρησιμοποιούνται ευρέως σε εφαρμογές επεξεργασίας ψηφιακών σημάτων διότι είναι αποδοτικές ως προς το κόστος λειτουργίας, την κατανάλωση ενέργειας και την ευελιξία. Οι βασικές γλώσσες που χρησιμοποιούνται για τον προγραμματισμό των FPGA είναι η Verilog και η VHDL. Η χρήση των παραπάνω γλωσσών είναι αρκετά πολύπλοκη και χρονοβόρα, αλλά είναι εφικτή και η χρήση τεχνικών προγραμματισμού υψηλού επιπέδου σε γλώσσα C ή Matlab.

Ένα σύστημα επεξεργασίας εικόνας αποτελείται από δύο βασικά μέρη. Την παραλαβή της εικόνας και την επεξεργασία της. Μία FPGA μπορεί να λαμβάνει την εικόνα είτε μέσω μίας κάμερας που είναι συνδεδεμένη σε αυτή είτε μέσω κάποια άλλης πηγής, όπως ο υπολογιστής, χρησιμοποιώντας τις διαθέσιμες εξόδους/εισόδους.

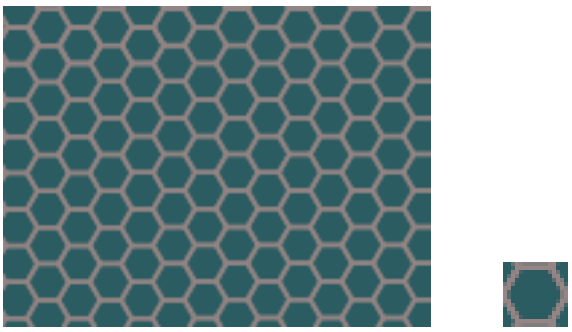
Κεφάλαιο 3

Μοντελοποίηση

3.1 Μοντελοποίηση με χρήση μεθόδων Template Matching

3.1.1 Σύγκριση απόδοσης μεθόδων Template Matching

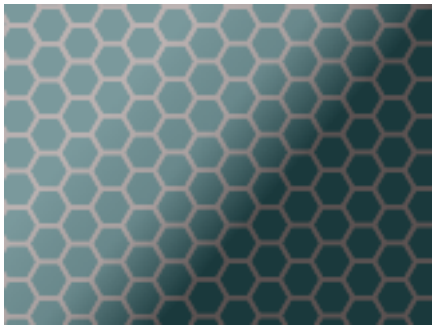
Με βάση τις πληροφορίες που λάβαμε από τη σχετική έρευνα, ξεκινήσαμε την μοντελοποίηση του προβλήματος. Κάθε μέθοδος Template Matching απαιτεί διαφορετικό αριθμό πράξεων, γεγονός το οποίο καθιστά απαραίτητη τη μελέτη τους ώστε να επιλεγεί ο λιγότερο πολύπλοκος αλγόριθμος με τα καλύτερα αποτελέσματα. Για τα πειράματα που ακολουθούν χρησιμοποιήθηκε εικόνα που εξομοιώνει το πρόβλημα, δηλαδή μοιάζει με δίχτυα ιχθυοκαλλιεργειών. Σε όλα τα πειράματα το πρότυπο (template) παραμένει ίδιο.



Σχήμα 3.1: Εικόνα και Template

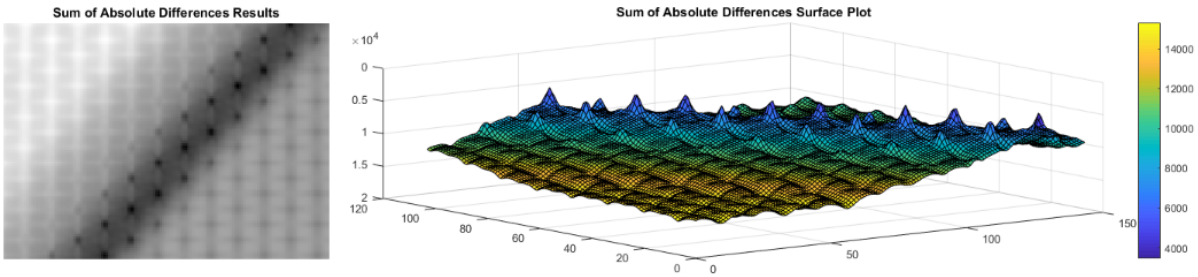
Η υλοποίηση και πειραματική εξέταση των αλγορίθμων έγινε σε Matlab. Σε ιδανικές συνθήκες όλες οι τεχνικές λειτουργούν εξίσου καλά. Οι εικόνες όμως που λαμβάνονται με τη χρήση κάμερας σε μη ελεγχόμενες συνθήκες δεν είναι ιδανικές.

Συχνό φαινόμενο στις υποβρύχιες εικόνες είναι η ύπαρξη διαφοροποιήσεων στο φωτισμό και την αντίθεση της εικόνας. Μετά από επεξεργασία της εικόνας 3.1 ώστε να παρουσιάζει διαφορετικό φωτισμό από τη μία άκρη στην άλλη, προέκυψε η παρακάτω εικόνα 3.2.



Σχήμα 3.2: Εικόνα με διαφοροποιήσεις στο φωτισμό

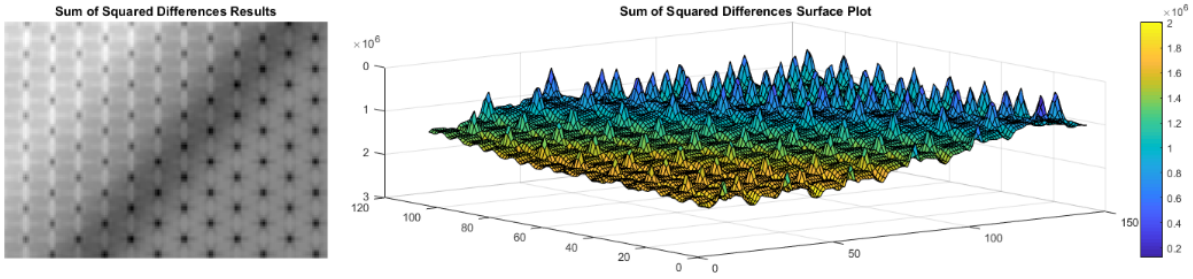
Στην εικόνα 3.3 φαίνονται τα αποτελέσματα της τεχνικής Sum Of Absolute Differences. Τα σημεία με μικρότερη τιμή είναι αυτά στα οποία το πρότυπο ταιριάζει περισσότερο με την εικόνα. Τα σημεία αυτά εμφανίζονται με μαύρο χρώμα, αλλά παρατηρούμε ότι δεν μπορούμε να βγάλουμε κάποιο σαφές συμπέρασμα. Αυτό οφείλεται στο γεγονός ότι η μέθοδος παράγει τα αποτελέσματα αφαιρώντας τα pixel της εικόνας με τα αντίστοιχα pixel του προτύπου. Επομένως στα σημεία όπου η διαφορά της φωτεινότητας της εικόνας με το πρότυπο είναι μεγάλη, το αποτέλεσμα τις διαφοράς είναι εξίσου μεγάλο.



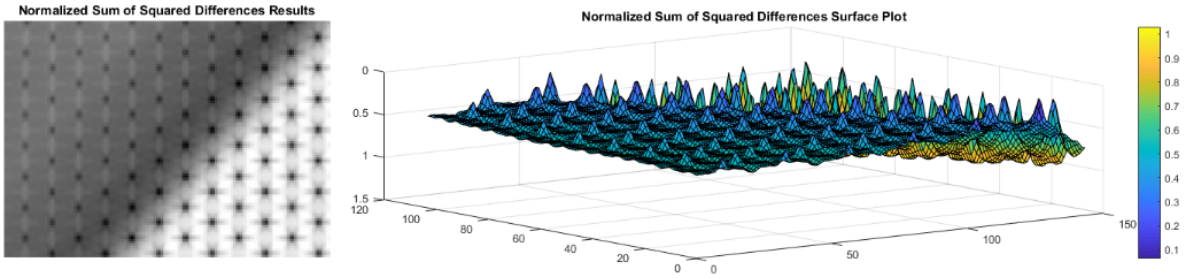
Σχήμα 3.3: Αποτελέσματα SAD (διαφοροποιήσεις στο φωτισμό)

Όμοια είναι τα αποτελέσματα και των μεθόδων Sum Of Squared Differences και Normalized Sum Of Squared Differences, που φαίνονται στις εικόνες 3.4 και 3.5 αντίστοιχα. Τα μαύρα σημεία είναι αυτά με τη μικρότερη διαφορά. Η μέθοδος Normalized Sum Of Squared Differences δίνει λίγο πιο ξεκάθαρα αποτελέσματα ,διότι

ομαλοποιεί τα αποτελέσματα της SSD και επηρεάζεται λιγότερο από τις διαφοροποιήσεις.

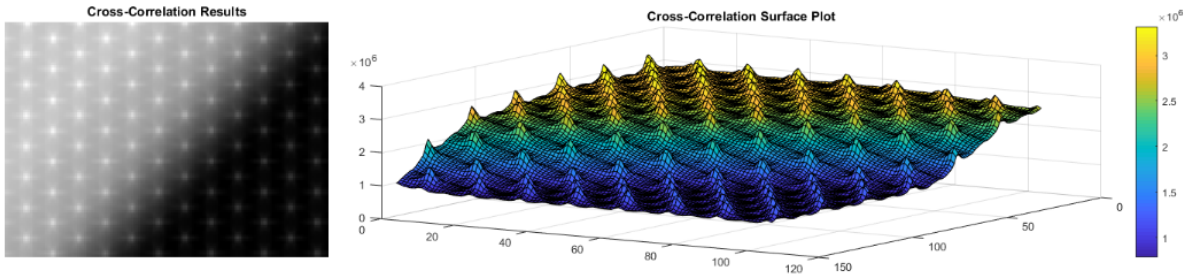


Σχήμα 3.4: Αποτελέσματα SSD (διαφοροποιήσεις στο φωτισμό)

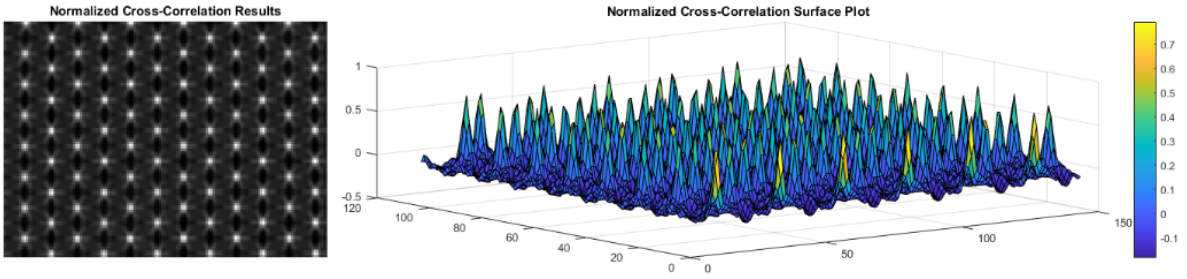


Σχήμα 3.5: Αποτελέσματα NSSD (διαφοροποιήσεις στο φωτισμό)

Συγκριτικά με τις παραπάνω τεχνικές, ο τρόπος υπολογισμού της συσχέτισης μεταξύ εικόνας και προτύπου διαφέρει στις μεθόδους Cross-Correlation και Normalized Cross-Correlation. Σε αυτές δεν υπολογίζεται η διαφορά, αλλά το το γινόμενο μεταξύ των pixel της εικόνας και του προτύπου. Όπως παρατηρούμε και στις εικόνες 3.6 και 3.7, όπου τα λευκά σημεία υποδηλώνουν την μεγαλύτερη συσχέτιση εικόνας-προτύπου, τα αποτελέσματα της NCC είναι εξαιρετικά. Αυτό οφείλεται στο γεγονός ο υπολογισμός της τιμής συσχέτισης συμπεριλαμβάνει τη μέση τιμή της εικόνας και του προτύπου. Όπως φαίνεται και στο διάγραμμα της NCC, τα σημεία που ταιριάζουν έχουν πολύ υψηλότερες τιμές από τα υπόλοιπα και γενικά δεν έχει επηρεαστεί σχεδόν καθόλου από τις διαφοροποιήσεις στα χρώματα της εικόνας που χρησιμοποιήθηκε.

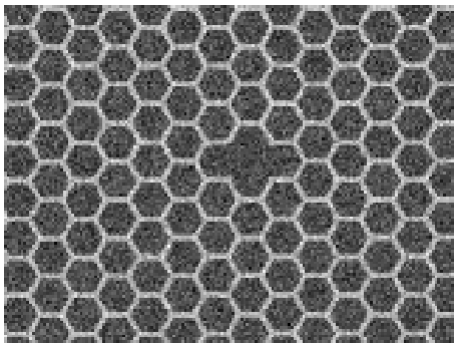


Σχήμα 3.6: Αποτελέσματα CC (διαφοροποιήσεις στο φωτισμό)

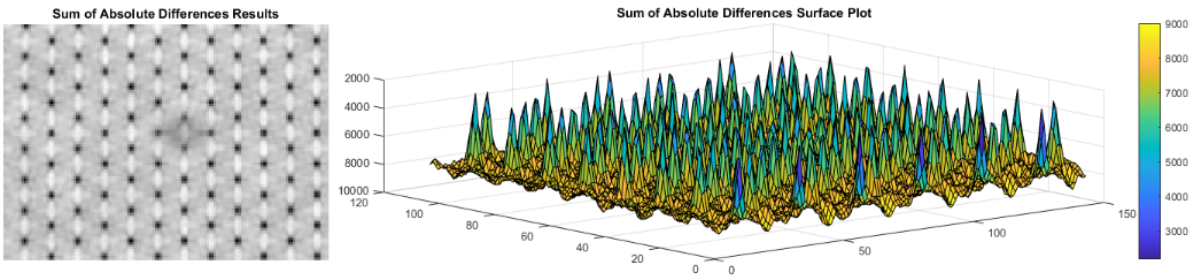


Σχήμα 3.7: Αποτελέσματα NCC (διαφοροποιήσεις στο φωτισμό)

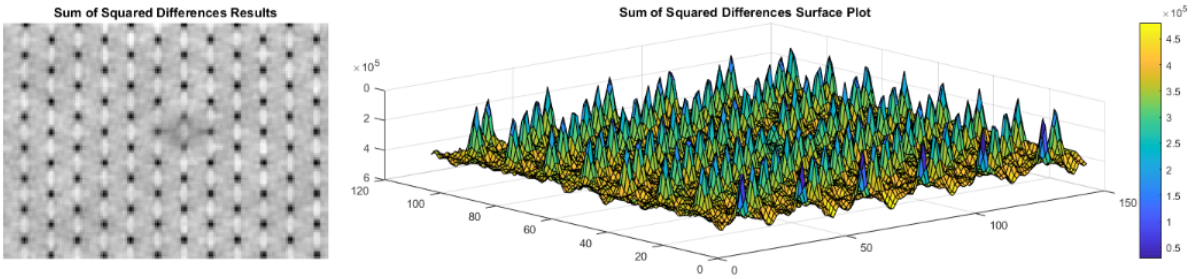
Από το προηγούμενο πείραμα συμπεράναμε ότι η μέθοδος NCC δίνει τα καλύτερα αποτελέσματα. Το γεγονός όμως ότι απαιτεί πολύ μεγάλο αριθμό πράξεων, καθιστά απαραίτητη τη περαιτέρω μελέτη των πιο απλών μεθόδων. Σε συνθήκες χαμηλού φωτισμού είναι πιθανή η ύπαρξη θορύβου και για αυτό το λόγο μελετήσαμε την επίδρασή του στα αποτελέσματα των μεθόδων Sum Of Absolute Differences, Sum Of Squared Differences και Normalized Sum Of Squared Differences. Μετά την προσθήκη θορύβου Gaussian με μέση τιμή 0.004 και variance 0.01 στην εικόνα 3.1, προέκυψε η 3.8, στην οποία δημιουργήσαμε μία οπή για να παρατηρήσουμε πόσο ξεκάθαρα είναι τα αποτελέσματα στις συνθήκες αυτές.



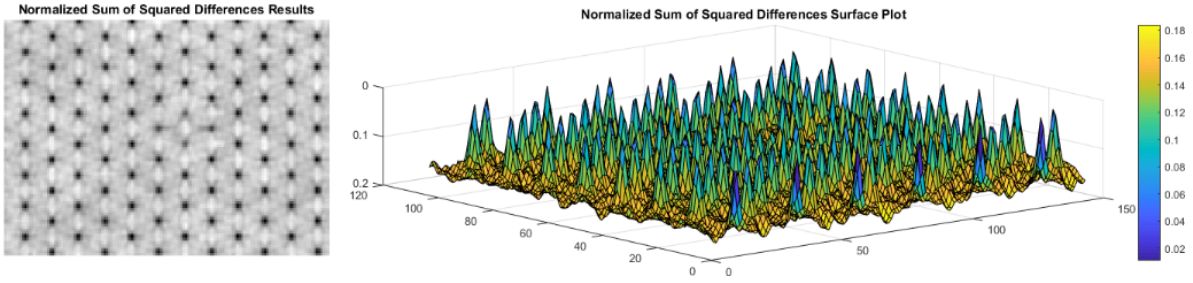
Σχήμα 3.8: Εικόνα με οπή και θόρυβο



Σχήμα 3.9: Αποτελέσματα SAD σε εικόνα με οπή και θόρυβο



Σχήμα 3.10: Αποτελέσματα SSD σε εικόνα με οπή και θόρυβο



Σχήμα 3.11: Αποτελέσματα NSSD σε εικόνα με οπή και θόρυβο

Τα αποτελέσματα όλων των μεθόδων είναι όμοια και παρατηρούμε ότι δεν επηρεάζονται σημαντικά από την ύπαρξη θορύβου και τα σημεία όπου υπάρχει οπή είναι ευδιάκριτα.

3.1.2 Σύγκριση πολυπλοκότητας μεθόδων Template Matching

Κάθε μέθοδος Template Matching έχει διαφορετικό τύπο υπολογισμού και απαιτεί διαφορετικές πράξεις. Η μέθοδος NCC φαίνεται να είναι η πολυπλοκότερη διότι απαιτεί τους περισσότερους υπολογισμούς. Στον Πίνακα 3.1 παρουσιάζεται ο αριθμός των πράξεων κάθε μεθόδου, για τον υπολογισμό μίας τιμής της, χωρίς όμως τις πράξεις που απαιτούνται για τον υπολογισμό των τιμών που αφορούν το Template,

καθώς υπολογίζονται μία φορά στην αρχή κάθε αλγορίθμου. Επίσης η ύψωση στο τετράγωνο υπολογίζεται ως πολλαπλασιασμός.

Πίνακας 3.1: Αριθμός πράξεων για τον υπολογισμό μίας τιμής

Μέθοδος	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί/Διαιρέσεις**
SAD	$2 \cdot (T_w \cdot T_h)$	0
SSD	$2 \cdot (T_w \cdot T_h)$	$T_w \cdot T_h$
NSSD	$3 \cdot (T_w \cdot T_h)$	$2 \cdot (T_w \cdot T_h) + 3$
CC	$T_w \cdot T_h$	$T_w \cdot T_h$
NCC	$4 \cdot (T_w \cdot T_h)$	$2 \cdot (T_w \cdot T_h) + 4$

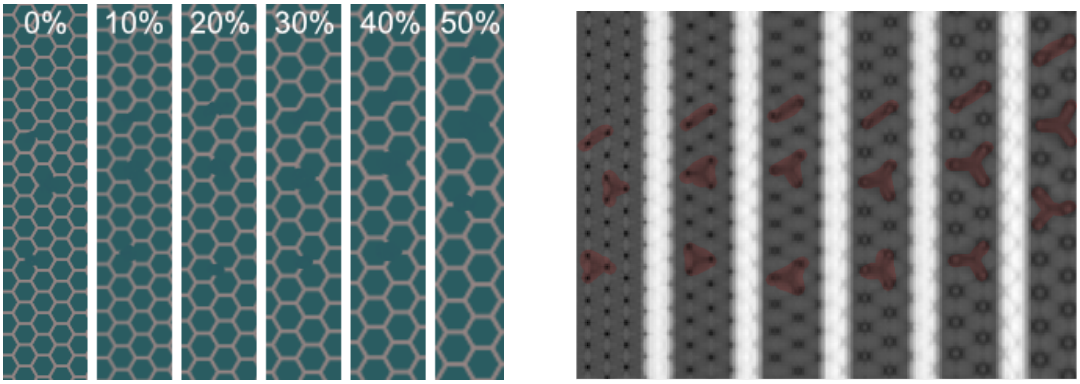
Παρατηρούμε ότι οι μέθοδοι Normalized Sum of Squared Differences(NSSD) και Normalized Cross Correlation(NCC) είναι αρκετά πιο πολύπλοκες από τις υπόλοιπες μεθόδους. Από τις τεχνικές που μελετήθηκαν ξεχωρίζουν οι SAD και NCC, διότι η πρώτη απαιτεί τις λιγότερες πράξεις ενώ η δεύτερη δίνει τα πιο ξεκάθαρα αποτελέσματα.

3.1.3 Πειραματική μελέτη μεθόδου SAD

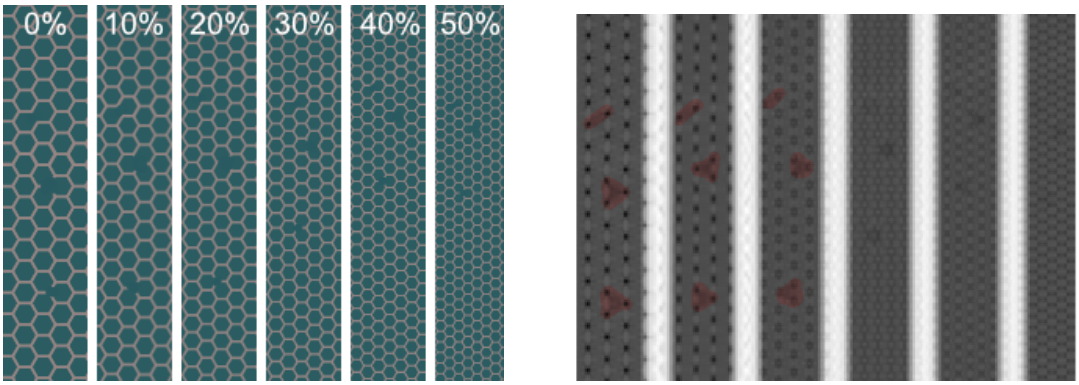
Σε αυτό το κομμάτι έγινε μελέτη των αποτελεσμάτων της μεθόδου SAD σε διάφορες συνθήκες για να ελέγξουμε κατά πόσο είναι εφικτός ο εντοπισμός των οπών σε αυτές. Γνωρίζουμε ότι γενικά οι μέθοδοί Template Matching είναι αδύναμες σε περιπτώσεις όπου το αντικείμενο που αναζητούμε στην εικόνα έχει υποστεί κάποια παραμόρφωση. Στην περίπτωση μας όμως οι εικόνες που επεξεργαζόμαστε αποτελούνται από επαναλαμβανόμενα τμήματα όμοια μεταξύ τους και πιθανόν να είναι εφικτή η αναγνώριση ατελειών στην εικόνα ακόμα και αν τα αποτελέσματα δεν είναι ακριβή. Αυτό το βασίζουμε στο γεγονός ότι τα τμήματα με οπές θα διαφέρουν αρκετά από τη γύρω περιοχή.

Αρχικά μελετήθηκαν τα αποτελέσματα, από εικόνες με οπές, όπου το δίχτυ στην εικόνα είναι μεγαλύτερο ή μικρότερο από αυτό της εικόνας από την οποία λήφθηκε το πρότυπο. Η μελέτη έγινε σε εικόνες που είναι κατά 10, 20, 30, 40 και 50% μεγαλύτερες ή μικρότερες. Από τις εικόνες 3.12 και 3.13 παρατηρούμε ότι οπτικά είναι εφικτός ο εντοπισμός τον οπών όταν το δίχτυ είναι κατά 10% μεγαλύτερο ή μικρότερο αλλά στις άλλες περιπτώσεις είναι αρκετά πιο δύσκολο και για τον ακριβή εντοπισμό τους θα χρειαστεί διαφορετικό πρότυπο.

Έπειτα γίνανε πειράματα σε εικόνες όπου το δίχτυ έχει περιστραφεί κατά 5, 10, 15, 20 και 25 μοίρες.Όπως φαίνεται και στην εικόνα 3.14 τα σημεία όπου υπάρχουν οπές



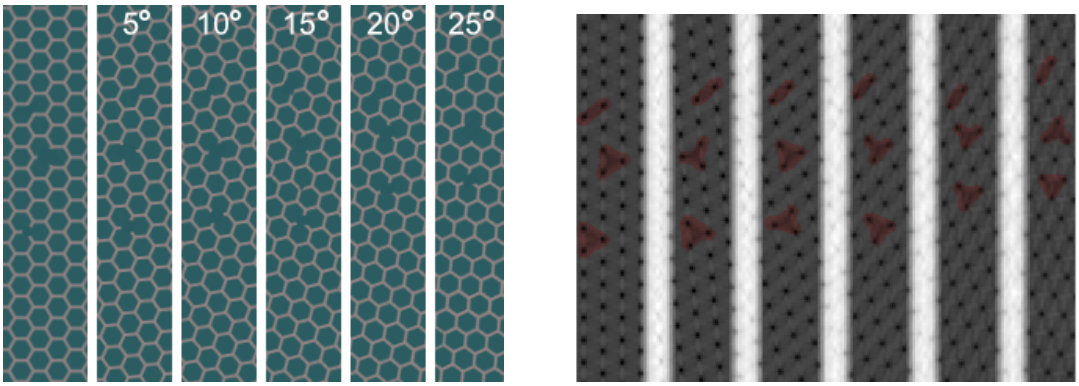
Σχήμα 3.12: Αποτελέσματα σε μεγαλύτερα δίχτυα



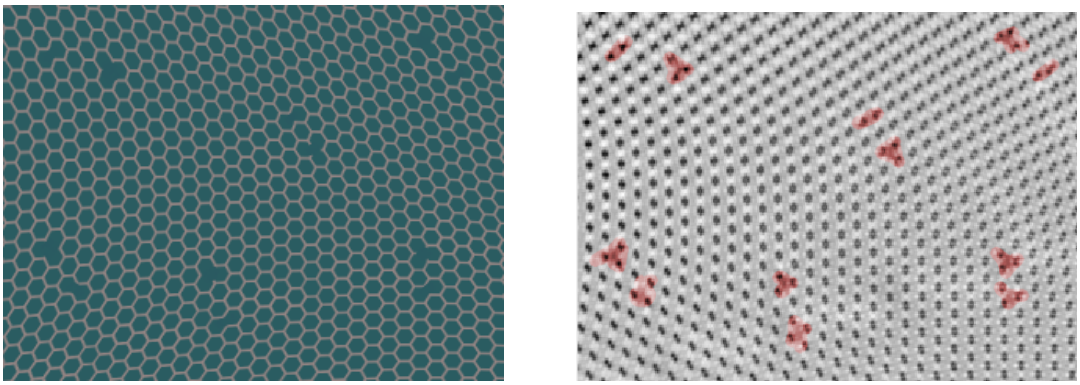
Σχήμα 3.13: Αποτελέσματα σε μικρότερα δίχτυα

είναι εμφανή και διαφέρουν αρκετά από τα γειτονικά τους.

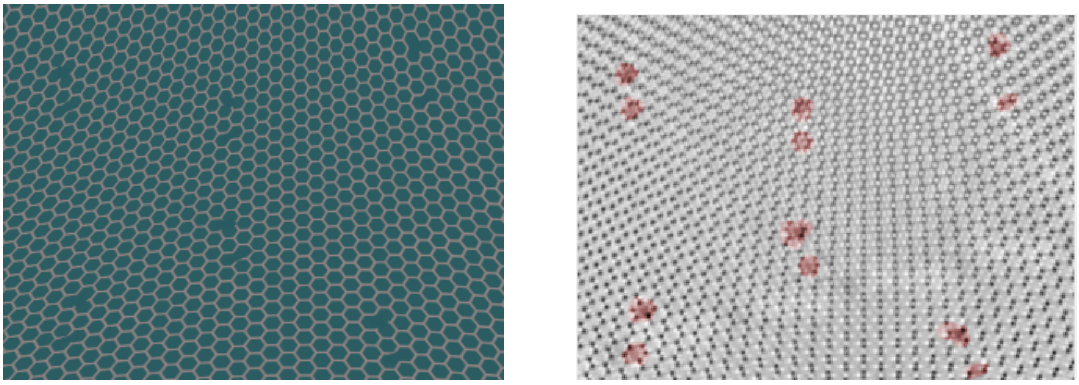
Τέλος έγιναν δοκιμές σε δίχτυα τα οποία δεν είναι τεντωμένα και παρουσιάζουν γεωμετρικές ανομοιομορφίες. Στις εικόνες 3.15 και 3.16 τα σημεία όπου υπάρχουν οπές είναι πιο θολά από τα γειτονικά τους και πιθανόν να είναι εφικτός ο εντοπισμός τους.



Σχήμα 3.14: Αποτελέσματα σε περιστραμμένα δίχτυα

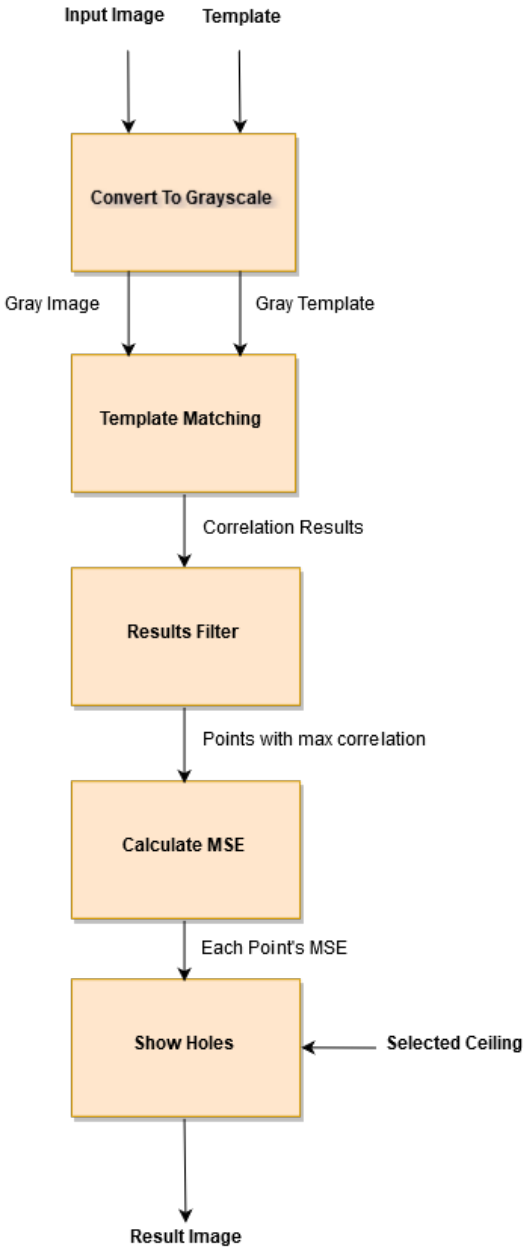


Σχήμα 3.15: Παραμορφωμένα δίχτυα (1)



Σχήμα 3.16: Παραμορφωμένα δίχτυα (2)

3.1.4 Αλγόριθμος εντοπισμού οπών με χρήση μεθόδων Template Matching

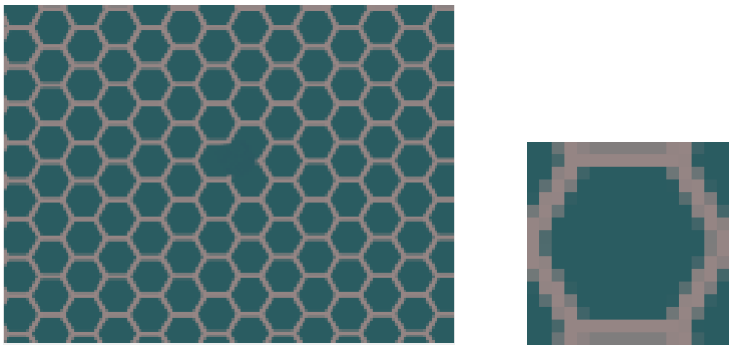


Σχήμα 3.17: Αλγόριθμος Template Matching

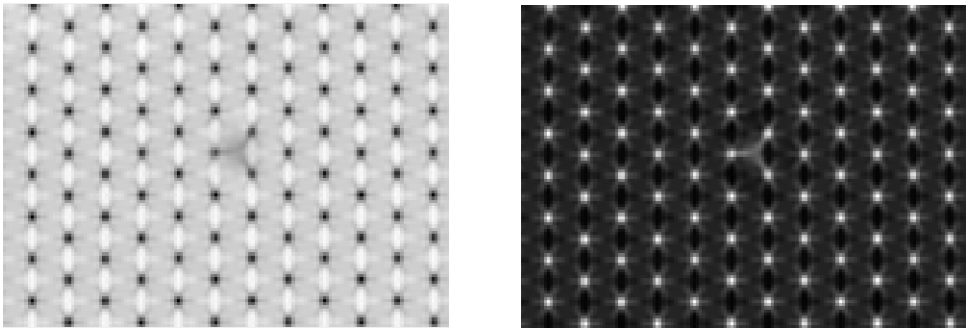
Ο αλγόριθμος που υλοποιήθηκε για τον εντοπισμό τον οπών στα δίχτυα αποτελείται από πέντε μέρη:

1. Το πρώτο βήμα είναι η λήψη της εικόνας προς επεξεργασία και του template και η μετατροπή τους σε αποχρώσεις του γκρι.

2. Έπειτα γίνεται ο υπολογισμός των τιμών συσχέτισης. Ο σωστός υπολογισμός γίνεται μεταφέροντας το πρότυπο σε ολόκληρη την εικόνα αλλά και έξω από αυτή. Για την αποφυγή της χρήσης διακλαδώσεων (if-statements) αποφασίσαμε να υπολογίζουμε μόνο τις τιμές για τα κομμάτια όπου το πρότυπο χωράει στην εικόνα. Αν υλοποιηθεί σωστά η μέθοδος,τότε σαν αποτέλεσμα δίνεται ένα πίνακας διαστάσεων $(I_w + T_w - 1) \cdot (I_h + T_h - 1)$ ενώ στη δική μας υλοποίηση το αποτέλεσμα είναι $(I_w - T_w) \cdot (I_h - T_h)$, όπου I_w και T_w τα πλάτη της εικόνας και του προτύπου αντίστοιχα και I_h, T_h τα ύψη.



Σχήμα 3.18: Εικόνα και πρότυπο

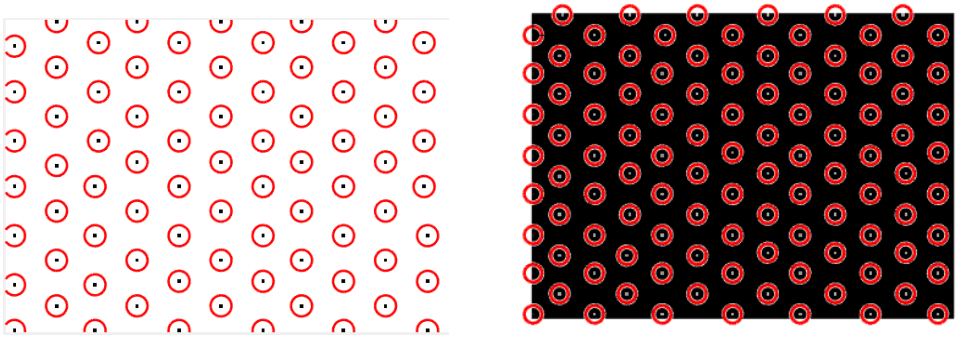


(α′) SAD (β′) NCC

Σχήμα 3.19: Αποτελέσματα μετά την εφαρμογή των μεθόδων Template Matching

3. Αν η μέθοδος Template Matching χρησιμοποιούνταν για την αναζήτηση ενός αντικειμένου μέσα στην εικόνα θα ήταν αρκετό να αναζητήσουμε στον πίνακα που υπολογίστηκε το σημείο που παρουσιάζει μέγιστη συσχέτιση. Στην

περίπτωση μας όμως αυτό δεν είναι αρκετό, διότι το δίχτυ παρουσιάζει παραμορφώσεις και σε πολλά σημεία δεν ταιριάζει απόλυτα με το πρότυπο. Για αυτό το λόγο υλοποιήσαμε μία μέθοδο φιλτραρίσματος τύπου Running Window Min/Max των αποτελεσμάτων για να λαμβάνουμε τα σημεία που παρουσιάζουν υψηλή συσχέτιση. Ο αλγόριθμος φιλτραρίσματος μεταφέρει ένα παράθυρο σε όλο το ύψος και πλάτος της εικόνας, το οποίο κρατάει την τιμή που υποδηλώνει τη μέγιστη συσχέτιση και αντικαθιστά τις υπόλοιπες τιμές με 0 στην περίπτωση της NCC και 65,200 για την SAD. Η τιμή 0 επιλέχτηκε διότι τα αποτελέσματα της NCC βρίσκονται στο διάστημα [-1,1] και η τιμή 65,200 είναι η μέγιστη τιμή που μπορεί να δώσει ως αποτέλεσμα η SAD στην περίπτωση όπου η διαφορά όλων των pixel της εικόνας και του προτύπου είναι 255 και χρησιμοποιείται πρότυπο διαστάσεων 16x16. Το παράθυρο επιλέχτηκε να έχει διαστάσεις $(T_w/3) \cdot (T_h/3)$ ώστε να μην υπάρχει ο κίνδυνος να βρεθούν μέσα σε αυτό δύο σημεία με μέγιστη συσχέτιση.



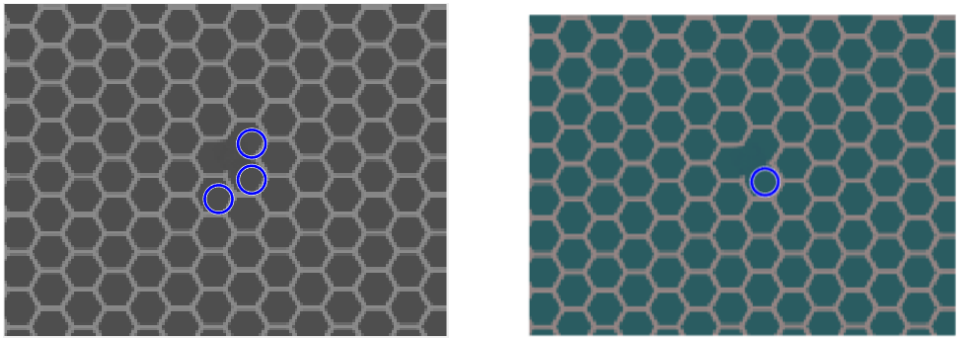
(α′) SAD (β′) NCC

Σχήμα 3.20: Αποτέλεσμα φιλτραρίσματος

4. Έπειτα για κάθε σημείο με τιμή μικρότερη του 65,200 στην περίπτωση της SAD ή μεγαλύτερη του 0 για την NCC υπολογίζεται το μέσο τετραγωνικό σφάλμα της τιμής του με τα τέσσερα κοντινότερα γειτονικά του σημεία. Τα σημεία που βρίσκονται κοντά στα όρια τις εικόνας δεν υπολογίζονται διότι δεν υπάρχουν αρκετά γειτονικά σημεία για τον σωστό υπολογισμό. Η τιμή του τετραγωνικού σφάλματος εξάγεται από τον παρακάτω τύπο, όπου P τα γειτονικά σημεία και C η τιμή του σημείου που γίνεται ο υπολογισμός.

$$MSE = \frac{1}{4}\sum_{4}^{i=1}(P_i - C)^2$$

5. Τέλος επιλέγονται τα σημεία που έχουν μεγαλύτερο μέσο τετραγωνικό σφάλμα από την τιμή που έχουμε ορίσει ως ανώφλι και παρουσιάζονται σαν οπές. Για τη μέθοδο NCC επιλέχτηκε η μέγιστη τιμή σφάλματος που εμφανίζεται σε δίχτυα χωρίς οπές. Στην περίπτωση της SAD, λόγω του μεγάλου εύρους τιμών των αποτελεσμάτων δεν ήταν εφικτή η χρήση μία συγκεκριμένης τιμής. Μετά από τον υπολογισμό των τιμών του μέσου τετραγωνικού σφάλματος, υπολογίζεται η μέση τιμή τους και σαν ανώφλι ορίζεται ένα πολλαπλάσιό της. Για το δικό μας dataset η μέση τιμή των MSE που υπολογίστηκαν για κάθε σημείο πολλαπλασιασμένη με το τρία έδινε καλά αποτελέσματα.



(α′) SAD (β′) NCC

Σχήμα 3.21: Σημείο όπου εντοπίστηκε οπή

3.2 Μοντελοποίηση με χρήση μεθόδων Edge Detection

3.2.1 Σύγκριση απόδοσης μεθόδων Edge Detection

Όπως είδαμε και στην ενότητα της σχετικής έρευνας, κάθε μέθοδος ανίχνευσης ακμών χρησιμοποιεί διαφορετική μάσκα. Στη περίπτωση της μεθόδου Roberts η μάσκα έχει διαστάσεις 2x2, ενώ οι μέθοδοι Sobel και Prewitt χρησιμοποιούν μάσκα διαστάσεων 3x3. Ο υπολογισμός της τιμής κάθε pixel χρησιμοποιώντας την τεχνική Roberts γίνεται με βάση τον παρακάτω τύπο:

$$mag_{x,y} = \sqrt{G_x^2 + G_y^2}$$

όπου

$$Gx = I_{x,y} - I_{x+1,y-1}$$

$$Gy = I_{x,y} - I_{x-1,y-1}$$

και $I_{x,y}$ το σημείο της εικόνας που γίνεται ο υπολογισμός.
Στην περίπτωση της μεθόδου Sobel:

$$Gy = (I_{x-1,y+1} + 2I_{x,y+1} + I_{x+1,y+1}) - (I_{x-1,y+1} + 2I_{x,y-1} + I_{x+1,y-1})$$

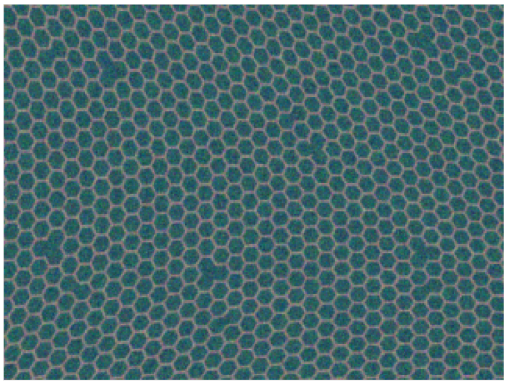
$$Gx = (I_{x+1,y-1} + 2I_{x+1,y} + I_{x+1,y+1}) - (I_{x-1,y-1} + 2I_{x-1,y} + I_{x-1,y+1})$$

Οι τύποι για Prewitt είναι:

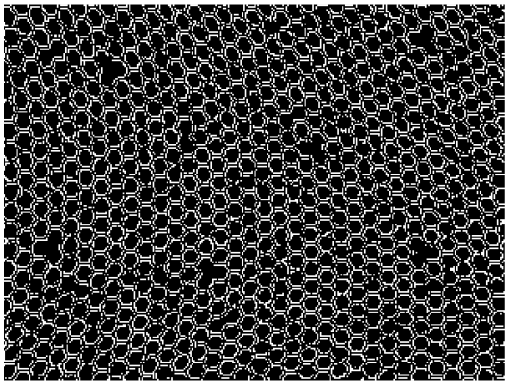
$$Gy = (I_{x-1,y+1} + I_{x,y+1} + I_{x+1,y+1}) - (I_{x-1,y+1} + I_{x,y-1} + I_{x+1,y-1})$$

$$Gx = (I_{x+1,y-1} + I_{x+1,y} + I_{x+1,y+1}) - (I_{x-1,y-1} + I_{x-1,y} + I_{x-1,y+1})$$

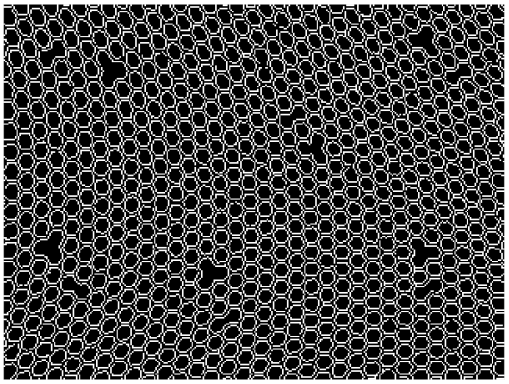
Όπως φαίνεται από τους παραπάνω τύπους, η μέθοδος Roberts απαιτεί λιγότερες πράξεις, αλλά υπολογίζει μόνο τις διαγώνιες διαφοροποιήσεις στην ένταση, ενώ οι άλλες μέθοδοι υπολογίζουν οριζόντιες και κάθετες διαφοροποιήσεις. Εφαρμόζοντας τις μεθόδους σε εικόνα με παραμορφωμένα δίχτυα και Gaussian θόρυβο μέσης τιμής 0.004 και variance 0.01 ,παρατηρήθηκε ότι δεν δίνει καλά αποτελέσματα (Εικόνα 3.22), ενώ οι άλλες δύο μέθοδοι δίνουν σχεδόν τα ίδια αποτελέσματα με τον ίδιο αριθμό προσθέσεων. Στην εικόνα προς επεξεργασία εφαρμόστηκε αρχικά φίλτρο Gaussian με sigma 1 ώστε να μειωθεί ο θόρυβος και έπειτα μετατράπηκε σε grayscale. Για κάθε μέθοδο χρησιμοποιήθηκε η αντίστοιχη συνάρτηση στη Matlab και το threshold υπολογίστηκε από τις ίδιες συναρτήσεις.



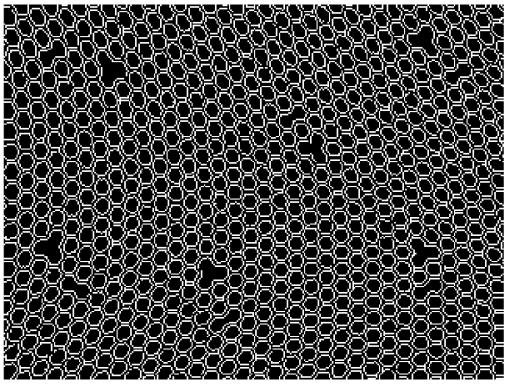
(α΄) Αρχική Εικόνα



(β΄) Αποτέλεσμα Roberts



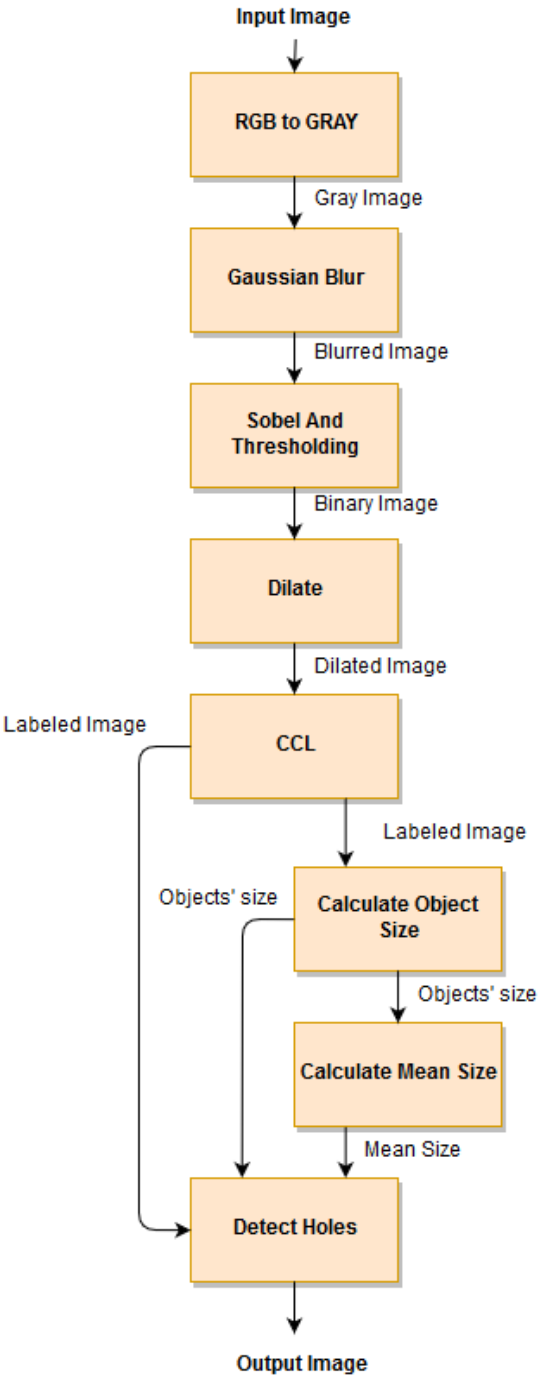
(γ΄) Αποτέλεσμα Sobel



(δ΄) Αποτέλεσμα Prewitt

Σχήμα 3.22: Αποτελέσματα μεθόδων Edge Detection

3.2.2 Αλγόριθμος εντοπισμού οπών με χρήση της μεθόδου Sobel



Σχήμα 3.23: Αλγόριθμος Edge Detection

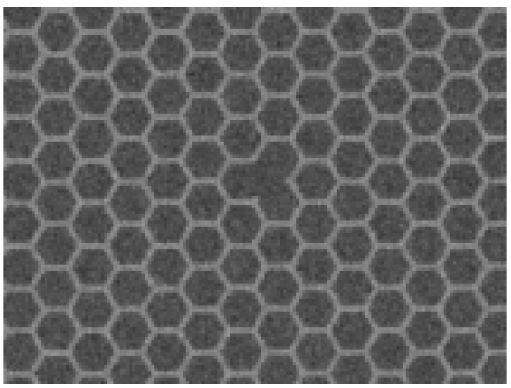
Σε αυτό το κομμάτι περιγράφεται ο αλγόριθμος ανίχνευσης οπών σε δίχτυα. Για την υλοποίηση, με τεχνικές Edge Detection, χρησιμοποιήθηκε η μέθοδος Sobel, διότι παρόλο που απαιτεί περισσότερες πράξεις, δίνει καλύτερα αποτελέσματα από τη μέθοδο Roberts. Η υλοποίηση έγινε σε Matlab και ακολουθούνται τα παρακάτω βήματα:

1. Πρώτο βήμα είναι η μετατροπή της εικόνας από έγχρωμη σε αποχρώσεις του γκρι. Ο προκαθορισμένος τύπος που χρησιμοποιείται στη Matlab είναι:

$$gray = 0.299R + 0.587G + 0.114B$$

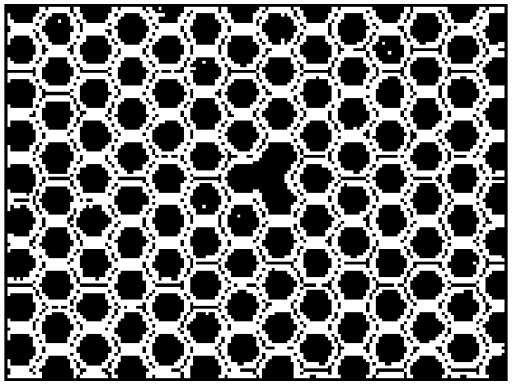
όπου R,G,B είναι τα κανάλια κόκκινο,πράσινο και μπλε αντίστοιχα της έγχρωμης εικόνας. Ο τύπος αυτός μπορεί να ρυθμιστεί ώστε να η τιμή της απόχρωσης του γρι να επηρεάζεται περισσότερο από το κόκκινο κανάλι για παράδειγμα. Στις υποθαλάσσιες εικόνες το φόντο πιθανόν να έχει χαμηλότερες τιμές σε κάποιο κανάλι από τα δίχτυα. Εμείς επιλέξαμε να χρησιμοποιήσουμε την έτοιμη συνάρτηση rgb2gray της Matlab. Σαν αποτέλεσμα δίνεται μία εικόνα που αποτελείται από ένα κανάλι με τιμές ακέραιους αριθμούς των 8-bit.

2. Επειδή οι τεχνικές εντοπισμού ακμών που μελετήσαμε επηρεάζονται αρκετά από την ύπαρξη θορύβου στην εικόνα, χρησιμοποιήθηκε φίλτρο Gaussian με τυπική απόκλιση 0.5. Η τιμή την τυπικής απόκλισης εξαρτάται από την ένταση του θορύβου και πρέπει να ρυθμιστεί ανάλογα με τις συνθήκες της εικόνας.



Σχήμα 3.24: Grayscale εικόνα

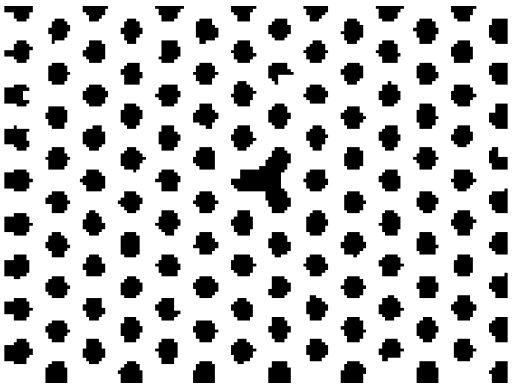
3. Έπειτα εφαρμόζεται η μέθοδος Sobel και στις τιμές που προκύπτουν γίνεται κατωφλίωση για την μετατροπή της εικόνας σε δυαδική. Όπου οι τιμές είναι μικρότερες του κατωφλιού ορίζεται η τιμή 0, ενώ στις υπόλοιπες η τιμή 1. Η τιμή του κατωφλιού εξαρτάται από τις συνθήκες της εικόνας. Εμείς επιλέξαμε



Σχήμα 3.25: Αποτέλεσμα Sobel

την τιμή 130, διότι έδινε καλά αποτελέσματα στο δικό μας dataset.

4. Στη συνέχεια γίνεται διαστολή των μονάδων του παραπάνω αποτελέσματος για να καλυφτούν τα κενά που εμφανίζονται. Χρησιμοποιήθηκε η συνάρτηση imdilate της Matlab η οποία αντικαθιστά τις τιμές 0, γύρω από τις τιμές 1 του προηγούμενου αποτελέσματος, με 1.



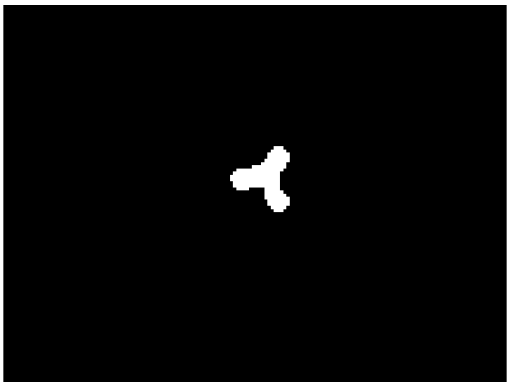
Σχήμα 3.26: Αποτέλεσμα διαστολής

5. Όπως φαίνεται στην εικόνα 3.26, το δίχτυ αναπαριστάται πλέον από την τιμή 1 και τα υπόλοιπα μέρη που αποτελούν το φόντο έχουν την τιμή 0. Σε αυτό το σημείο, στόχος μας είναι να εντοπίσουμε όλα τα σημεία με τιμή 0 που συνδέονται μεταξύ τους και να υπολογίσουμε το μέγεθός τους. Για αυτό το σκοπό υλοποιήθηκε αλγόριθμος Connected-component labeling, ο οποίος ομαδοποιεί τα pixel που ανήκουν στο ίδιο αντικείμενο. Επιλέξαμε να χρησιμοποιήσουμε

την μέθοδο δύο περασμάτων, διότι η εικόνα μας περιέχει πολλά αντικείμενα και θεωρήσαμε ότι θα είναι πιο αποδοτική από τις υπόλοιπες μεθόδους. Εν συντομία ο αλγόριθμος λειτουργεί ως εξής. Αρχικά για κάθε pixel της εικόνας, που έχει την τιμή 0, ελέγχονται τα γειτονικά pixel αριστερά και πάνω από αυτό, όπου υπάρχουν. Ταυτόχρονα κατασκευάζεται και ένα πίνακας ο οποίος περιέχει όλες τις διαθέσιμες τιμές labels. Αν κανένα από τα γειτονικά pixel δεν έχει τιμή (ταμπέλα) τότε ορίζεται μία σε αυτό. Αν κάποιο από τα γειτονικά pixel έχει ταμπέλα, τότε η τιμή του ορίζετε ίση με αυτή του γείτονα. Σε περίπτωση που διαθέτουν και τα δύο ταμπέλα ορίζεται η μικρότερη τιμή και αποθηκεύουμε σε πίνακα την πληροφορία ότι οι δύο ταμπέλες ανήκουν στο ίδιο αντικείμενο. Αφού ολοκληρωθεί αυτή η διαδικασία ελέγχονται όλες οι τιμές του παραπάνω αποτελέσματος και αντικαθίστανται από αυτές που έχουν αποθηκευτεί στον πίνακα με τις ταμπέλες. Σε αυτό το στάδιο όλα τα συνδεδεμένα αντικείμενα έχουν πλέον την ίδια τιμή.

6. Στη συνέχεια ελέγχονται όλες οι τιμές (labels) του παραπάνω αποτελέσματος και υπολογίζεται πόσες φορές εμφανίζονται στην εικόνα. Με αυτό τον τρόπο είναι εφικτό να υπολογιστεί το μέγεθος κάθε αντικειμένου.

7. Τέλος υπολογίζεται ο μέσος όρος του μεγέθους όλων των αντικειμένων στην εικόνα και εντοπίζονται τα αντικείμενα που έχουν διπλάσιο μέγεθος από αυτόν. Τα αντικείμενα αυτά αποτελούν οπές.

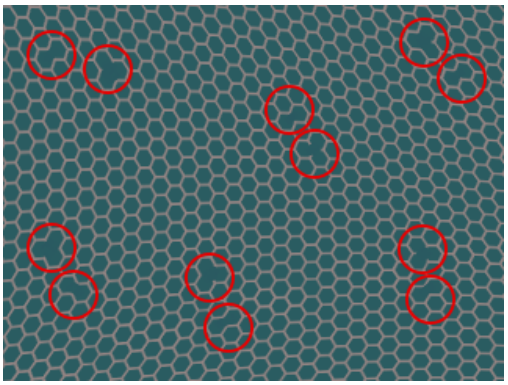


Σχήμα 3.27: Τελικό αποτέλεσμα

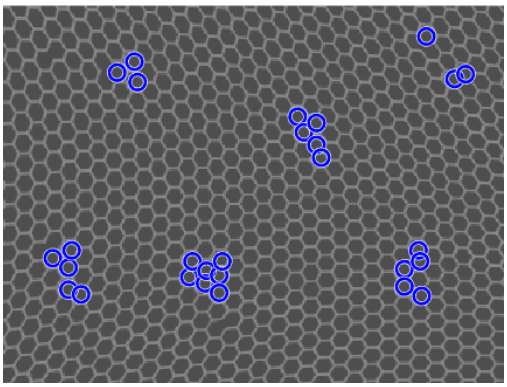
3.3 Σύγκριση των δύο αλγορίθμων που υλοποιήθηκαν

3.3.1 Πειραματική μελέτη

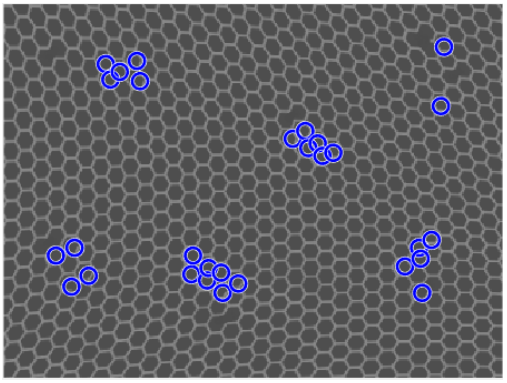
Σε αυτό το στάδιο ελέγχουμε πειραματικά τις αδυναμίες κάθε αλγορίθμου. Η υλοποίηση με χρήση μεθόδων Template Matching παρουσιάζει μη ακριβή αποτελέσματα στις περιπτώσεις όπου το αντικειμένου εμφανίζεται στην εικόνα προς επεξεργασία παραμορφωμένο, ενώ επηρεάζεται λιγότερο από την ύπαρξη θορύβου σε σχέση με το αλγόριθμο που χρησιμοποιεί τη μέθοδο Edge Detection.



(α′) Εικόνα προς επεξεργασία



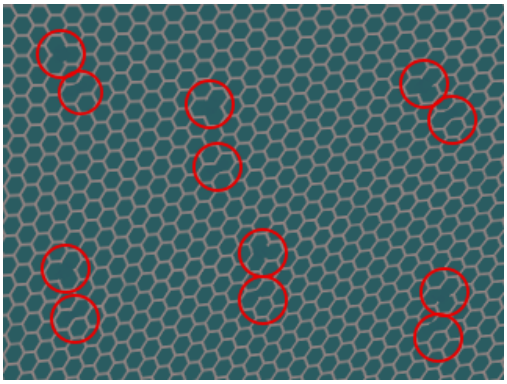
(β′) Αποτελέσματα NCC



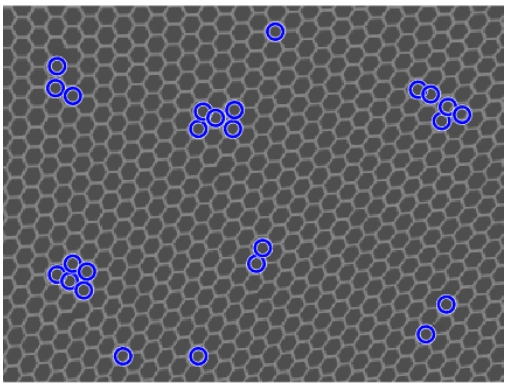
(γ′) Αποτελέσματα SAD

(δ′) Αποτελέσματα Edge Detection

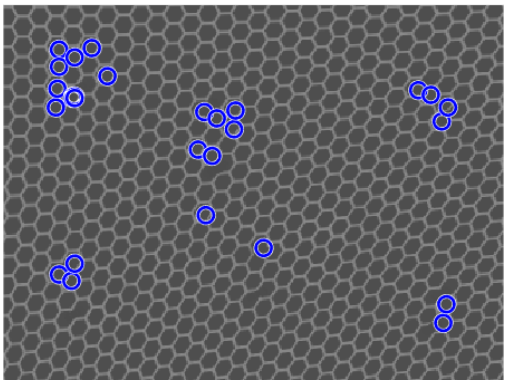
Σχήμα 3.28: Παραμορφωμένα δίχτυα (1)



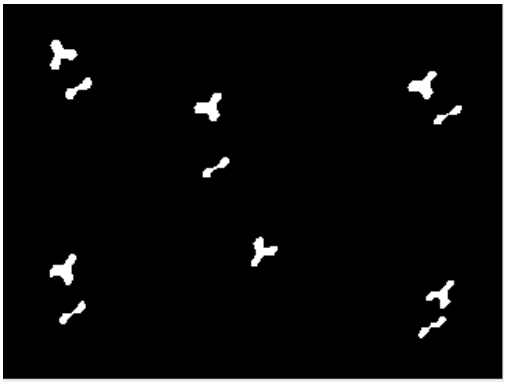
(α′) Εικόνα προς επεξεργασία



(β′) Αποτελέσματα NCC

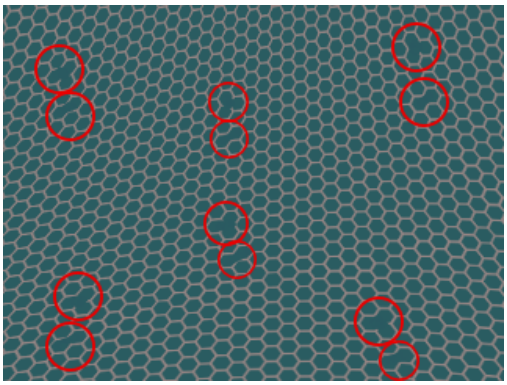


(γ′) Αποτελέσματα SAD

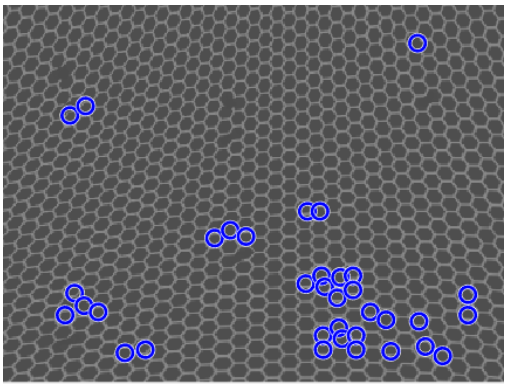


(δ′) Αποτελέσματα Edge Detection

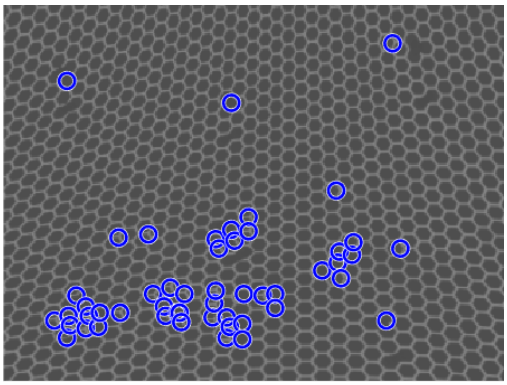
Σχήμα 3.29: Παραμορφωμένα δίχτυα (2)



(α′) Εικόνα προς επεξεργασία



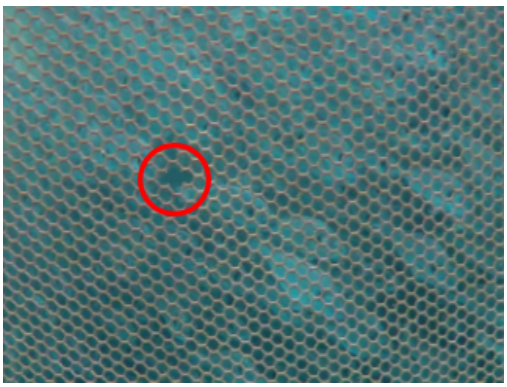
(β′) Αποτελέσματα NCC



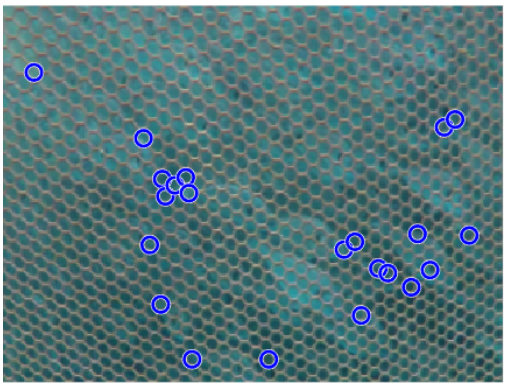
(γ′) Αποτελέσματα SAD

(δ′) Αποτελέσματα Edge Detection

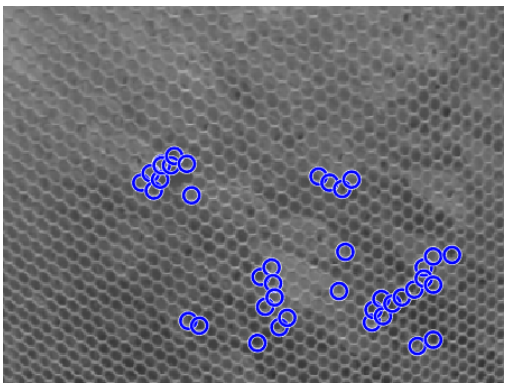
Σχήμα 3.30: Παραμορφωμένα δίχτυα (3)



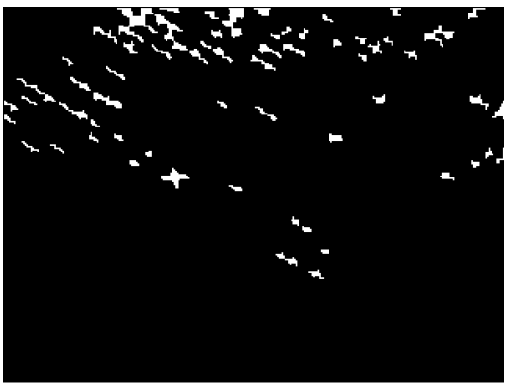
(α′) Εικόνα προς επεξεργασία



(β′) Αποτελέσματα NCC



(γ′) Αποτελέσματα SAD



(δ′) Αποτελέσματα Edge Detection

Σχήμα 3.31: Πραγματική εικόνα με ψάρια

Από τις παραπάνω εικόνες παρατηρούμε ότι οι αλγόριθμοι που χρησιμοποιούν τη μέθοδο Template Matching επηρεάζονται σημαντικά από τις παραμορφώσεις του διχτυού και εντοπίζουν οπές σε σημεία που δεν υπάρχουν. Ο αλγόριθμος με τη μέθοδο Edge Detection εμφανίζει κάποια λάθος αποτελέσματα στα όρια της εικόνας 3.30 και επηρεάζεται σημαντικά από τον θόρυβο στην εικόνα 3.31. Στον πίνακα 3.2 παρουσιάζεται το σύνολο των οπών που εντοπίστηκαν από κάθε αλγόριθμο.

Πίνακας 3.2: Σύνολο οπών που εντοπίστηκαν σε κάθε εικόνα

	T.M.(SAD)	T.M.(NCC)	Edge Detection
Εικόνα 3.28	8	8	12
Εικόνα 3.29	9	9	10
Εικόνα 3.30	8	6	11
Εικόνα 3.31	1	1	1
Οπές που εντοπίστηκαν	26/37	24/37	34/37
Ποσοστό εντοπισμένων οπών	70.2%	64.9%	91.9%

Η εξάρτηση των αλγορίθμων που χρησιμοποιούν τεχνικές Template Matching από το πρότυπο μειώνει την ευελιξία τους και απαιτεί πιο ελεγχόμενες συνθήκες, δηλαδή συγκεκριμένη απόσταση από το δίχτυ και σταθερή κλίση κατά τη λήψη της εικόνας. Αντίθετα ο αλγόριθμος που χρησιμοποιεί τη μέθοδο Edge Detection είναι πιο ευέλικτος και ακριβής όταν δεν υπάρχει έντονος θόρυβος.

3.3.2 Μελέτη πολυπλοκότητας

Σε αυτή την υποενότητα θα μελετήσουμε την πολυπλοκότητα των αλγορίθμων και το κατά πόσο είναι εφικτή η υλοποίηση τους σε FPGA. Οι αλγόριθμοι που χρησιμοποιούν την τεχνική Template Matching διαφέρουν σημαντικά στον αριθμό των πράξεων. Η χρήση pipeline είναι εφικτή αλλά στην περίπτωση της μεθόδου NCC απαιτούνται πολύ περισσότεροι πόροι. Επειδή η πολυπλοκότητα του προβλήματος αυξάνεται σε πολύ μεγάλο βαθμό όταν υπάρχουν διαφοροποιήσεις στη φωτεινότητα, θα υποθέσουμε ότι δεν είναι έντονες, ώστε να αρκεστούμε στη χρήση της μεθόδου SAD για τον υπολογισμό της συσχέτισης. Έτσι η τελική υλοποίηση θα μπορεί να γίνει σε μικρότερη FPGA και με λιγότερη κατανάλωση ενέργειας. Στους πίνακες 3.3 και 3.4 παρουσιάζονται οι απαιτούμενες πράξεις του αλγορίθμου με Template Matching(SAD) και με Edge Detection(Sobel) αντίστοιχα.

Πίνακας 3.3: Πίνακας συνολικών πράξεων αλγορίθμου Template Matching (SAD)

Κομμάτι αλγορίθμου	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί/Διαιρέσεις**	**Έλεγχος τιμών**
SAD	$2 \cdot (I_w \cdot I_h) \cdot (T_w \cdot T_h)$		
Results Filter			$(T_w/3 \cdot T_h/3) \cdot (I_w - T_w) \cdot (I_h - T_h)$
Υπολογισμός MSE	4·Points	5·Points	$(I_w - T_w) \cdot (I_h - T_h)$+$2 \cdot (T_w \cdot T_h)$·Points
Μέση τιμή MSE	Points	1	
Εντοπισμός Οπών			Points
Σύνολο	**2**·$(I_w \cdot I_h) \cdot (T_w \cdot T_h)$+5·**Points**	5·**Points**+**1**	$(T_w/3 \cdot T_h/3 + 1) \cdot (I_w - T_w) \cdot (I_h - T_h)$ +$2 \cdot (T_w \cdot T_h + 1)$·**Points**

Στον πίνακα 3.5 παρουσιάζεται ένα αριθμητικό παράδειγμα των απαιτούμενων υπολογισμών για την καλύτερη εκτίμησή τους. Χρησιμοποιήθηκε εικόνα διαστάσεων

Πίνακας 3.4: Πίνακας συνολικών πράξεων αλγορίθμου Edge Detection

Κομμάτι αλγορίθμου	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί/Διαιρέσεις**	**Έλεγχος τιμών**
Sobel	$10(I_w \cdot I_h)$	$7(I_w \cdot I_h)$	
Thresholding			$(I_w \cdot I_h)$
Binary Dilate			$(I_w \cdot I_h)$
CCL			$4(I_w \cdot I_h)$
Size Count	$(I_w \cdot I_h)$		$(I_w \cdot I_h)$
Mean Size	Objects		1
Detect Holes			Objects
Σύνολο	**11**$(I_w \cdot I_h)$**+Objects**	**7**$(I_w \cdot I_h)$	**7**$(I_w \cdot I_h)$**+1+Objects**

320x240 και πρότυπο διαστάσεων 16x16 και οι τιμές (Points=387,Objects=491).

Πίνακας 3.5: Αριθμητικό παράδειγμα υπολογισμών/πράξεων

Αλγόριθμος	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί/Διαιρέσεις**	**Έλεγχος τιμών**
Template Matching	39323535	1936	1905795
Edge Detection	845291	537600	538092

Παρατηρούμε ότι ο πρώτος αλγόριθμος απαιτεί πολύ μεγαλύτερο αριθμό προσθέσεων/αφαιρέσεων και παραπάνω από τον τριπλάσιο αριθμό ελέγχου τιμών. Επίσης ο τρόπος με τον οποίο φιλτράρονται τα δεδομένα δεν επιτρέπει την αποδοτική χρήση παραλληλισμού, γεγονός το οποίο καθιστά πολύ δύσκολή την υλοποίηση και επιτάχυνση του φίλτρου σε FGPA. Η αποφυγή της χρήσης Running Max/Min Filter και η χρήση μία τιμής κατώφλι για τον εντοπισμό των οπών θα οδηγήσει στην υλοποίηση ενός συστήματος το οποίο θα λειτουργούσε σωστά σε πολύ περιορισμένες συνθήκες.

Αντίθετα ο δεύτερος αλγόριθμος απαιτεί μικρότερο αριθμό πράξεων οι οποίες είναι εφικτό να γίνουν παράλληλα με μικρό αριθμό πόρων και το κομμάτι το οποίο δεν επιτρέπει παραλληλισμό είναι πολύ μικρότερο. Η μέθοδος Sobel,το Thresholding,το Dilate καθώς και οι τεχνικές φιλτραρίσματος για την μείωση του θορύβου είναι εφικτό να υλοποιηθούν με τη χρήση line buffer και να επιταχυνθούν με παραλληλισμό μέσω pipeline. Επιπλέον ο αλγόριθμος Connected Component Labeling που χρησιμοποιούμε απαιτεί πολύ μικρότερο αριθμό ελέγχου τιμών για κάθε pixel συγκριτικά με ένα φίλτρο μέγιστης τιμής που χρησιμοποιεί παράθυρο διαστάσεων 5x5.

Μετά την υλοποίηση και των δύο αλγορίθμων σε FPGA Nexys 4 DDR προέκυψαν οι παρακάτω πίνακες Latency με ρολόι περιόδου 12ns:

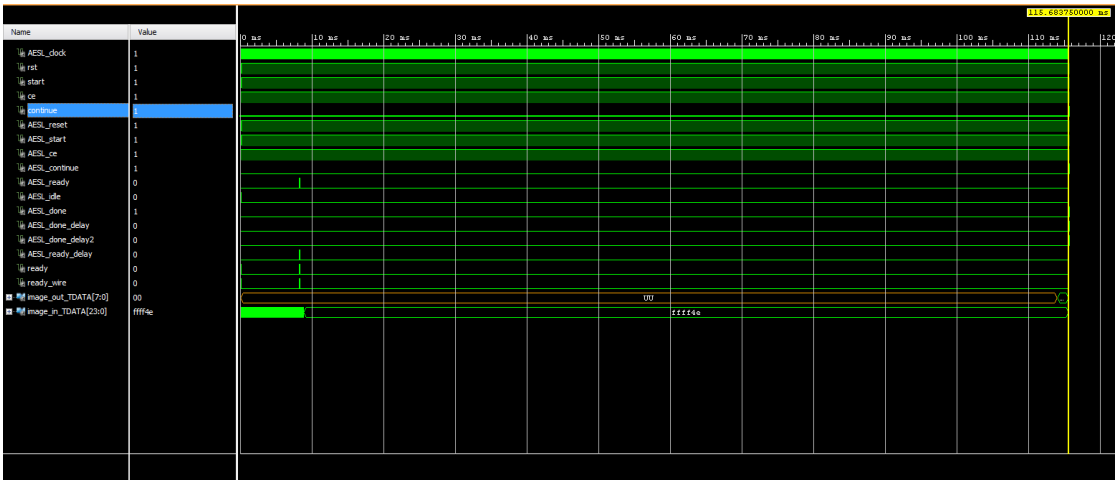
Πίνακας 3.6: Template Matching Algorithm Latency (clock cycles)

Latency		**Interval**		**Pipeline Type**
min	max	min	max	Type
9421978	9916978	8731069	9226069	dataflow

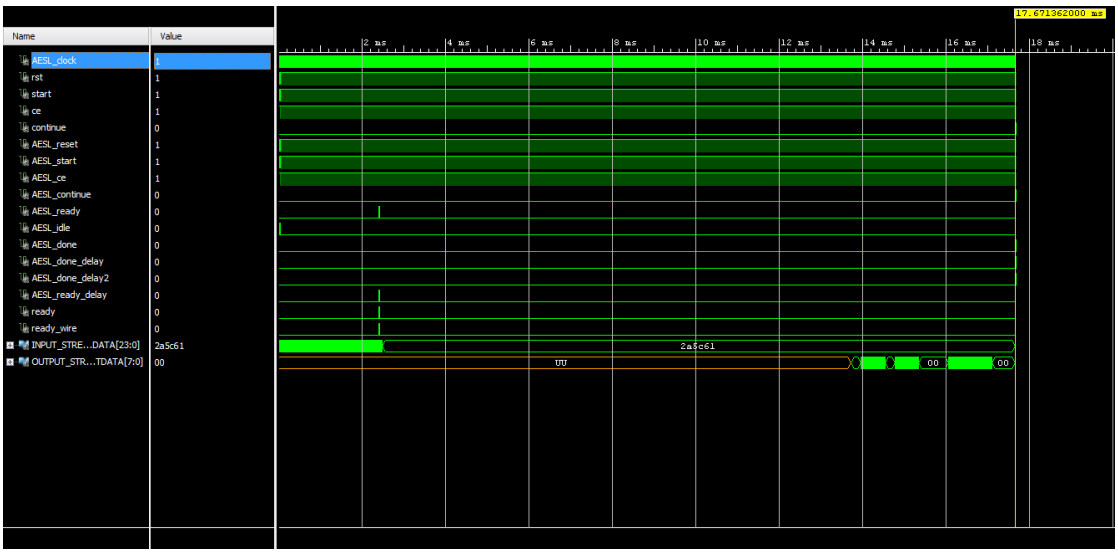
Πίνακας 3.7: Edge Detection Algorithm Latency (clock cycles)

Latency		**Interval**		**Pipeline Type**
min	max	min	max	Type
1304063	1993027	1303696	1992660	dataflow

Από τους παραπάνω πίνακες επιβεβαιώνονται οι παρατηρήσεις που κάναμε προηγουμένως. Ο αλγόριθμος που βασίζεται στη χρήση της μεθόδου Template Matching απαιτεί πολύ περισσότερους κύκλους ρολογιού. Αυτό οφείλεται σε μεγαλύτερο βαθμό στον τρόπο με τον οποίο φιλτράρονται τα αποτελέσματα. Όπως αναφέραμε και προηγουμένως δεν είναι εφικτή η χρήση παραλληλισμού και απαιτεί πολύ μεγάλο αριθμό υπολογισμών. Ο συνολικός χρόνος εκτέλεσης του πρώτου αλγορίθμου είναι 115.68ms όπως φαίνεται και στην εικόνα 3.32 ενώ του δεύτερου 17.67ms (Εικόνα 3.33). Οι αντίστοιχοι χρόνοι που λάβαμε κατά την εκτέλεση των αλγορίθμων σε υπολογιστή με επεξεργαστή Intel Core i5-4670 συχνότητας 3.4Ghz είναι 534ms και 266ms. Οι τιμές αυτές αποδεικνύουν ότι η χρήση FPGA έχει νόημα , καθώς λαμβάνουμε καλύτερους χρόνους εκτέλεσης. Πρέπει όμως να τονιστεί ότι δεν έγινε προσπάθεια επιτάχυνσης των αλγορίθμων σε software και πιθανότατα μπορούν να επιτευχθούν καλύτεροι χρόνοι εκτέλεσης.



Σχήμα 3.32: Simulation πρώτου αλγορίθμου για εικόνα διαστάσεων 320x240 με δώδεκα οπές.



Σχήμα 3.33: Simulation δεύτερου αλγορίθμου για εικόνα διαστάσεων 320x240

Από τους πίνακες 3.8 και 3.9 παρατηρούμε ότι οι συνολικές απαιτήσεις σε πόρους είναι αρκετά μικρές και για τους δύο αλγορίθμους. Ο πρώτος αλγόριθμος όμως δεσμεύει λιγότερους πόρους. Όμοια είναι και τα αποτελέσματα της ισχύς που φαίνονται στους πίνακες 3.10 και 3.11. Οι απαιτήσεις σε ισχύ είναι αρκετά χαμηλές και διαφέρουν πολύ λίγο. Συνδυάζοντας τα δεδομένα της ισχύς και του χρόνου εκτέλεσης κάθε αλγορίθμου, έχουμε ως αποτέλεσμα ο πρώτος αλγόριθμος να απαιτεί περίπου 30mJ ενέργειας για την επεξεργασία μίας εικόνας ενώ ο δεύτερος 5.4mJ. Για να είναι πιο συμφέρουσα μία υλοποίηση χρησιμοποιώντας τον επεξεργαστή Intel Core i5-4670

Πίνακας 3.8: Template Matching Algorithm Utilization Estimates

Name	**BRAM18K**	**DSP48E**	**FF**	**LUT**
DSP	-	-	-	-
Expression	-	-	0	1
FIFO	0	-	50	210
Instance	158	10	8010	14890
Memory	-	-	-	-
Multiplexer	-	-	-	-
Register	-	-	-	-
Total	**158**	**10**	**8060**	**15101**
Available	270	240	126800	63400
Utilization (%)	**58**	**4**	**6**	**23**

Πίνακας 3.9: Edge Detection Algorithm Utilization Estimates

Name	**BRAM18K**	**DSP48E**	**FF**	**LUT**
DSP	-	-	-	-
Expression	-	-	0	4
FIFO	0	-	85	312
Instance	204	44	7197	13507
Memory	-	-	-	-
Multiplexer	-	-	-	-
Register	-	-	-	-
Total	**204**	**44**	**7282**	**13823**
Available	270	240	126800	63400
Utilization (%)	**75**	**18**	**5**	**21**

θα πρέπει να επιτευχθούν χρόνοι εκτέλεσης μικρότεροι των 0.357ms και 0.06ms αντίστοιχα, αν υποθέσουμε ότι η απαιτούμενη ισχύς του επεξεργαστή είναι αυτή που δίνεται από τον κατασκευαστή (84W). Όμως, η πραγματική απαιτούμενη ισχύς από έναν υπολογιστή είναι αρκετά μεγαλύτερη καθώς απαρτίζεται και από άλλα μέρη με αποτέλεσμα οι παραπάνω τιμές να πρέπει να είναι πολύ μικρότερες.

Πίνακας 3.10: Ισχύς πρώτου αλγορίθμου

Total On-Chip Power	0.265W
Junction Temperature	26.2^oC
Thermal Margin	58.8^oC (12.7W)

Πίνακας 3.11: Ισχύς δεύτερου αλγορίθμου

Total On-Chip Power	0.309W
Junction Temperature	26.4^oC
Thermal Margin	58.6^oC (12.7W)

3.3.3 Συμπεράσματα

Βασιζόμενοι στα στοιχεία που συλλέξαμε για κάθε αλγόριθμο αποφασίσαμε το τελικό μας σύστημα να υλοποιηθεί με τη χρήση του αλγορίθμου που χρησιμοποιεί τη μέθοδο Edge Detection. Ο συνολικός χρόνος εκτέλεσης είναι πολύ μικρότερος από τον άλλο αλγόριθμο και οι απαιτήσεις σε πόρους και ισχύ δεν είναι μεγάλες. Τα παραπάνω χαρακτηριστικά τον καθιστούν ιδανικό για τη χρήση σε ένα φορητό ενσωματωμένο σύστημα πραγματικού χρόνου. Επιπλέον τα αποτελέσματα κατά τον εντοπισμό των οπών είναι πολύ καλύτερα και το τελικό μας σύστημα θα λειτουργεί σωστά με λιγότερους περιορισμούς. Επίσης παρατηρήσαμε ότι επηρεάζεται σημαντικά από την ύπαρξη θορύβου, όμως οι εικόνες που καλούμαστε να επεξεργαστούμε δεν θα περιέχουν τόσο θόρυβο όπως η εικόνα 3.31, διότι οι εικόνες θα λαμβάνονται μέσα από το κλουβί και το background θα έχει πολύ λιγότερο θόρυβο.

Κεφάλαιο 4

Σχεδίαση και υλοποίηση

4.1 Εισαγωγή

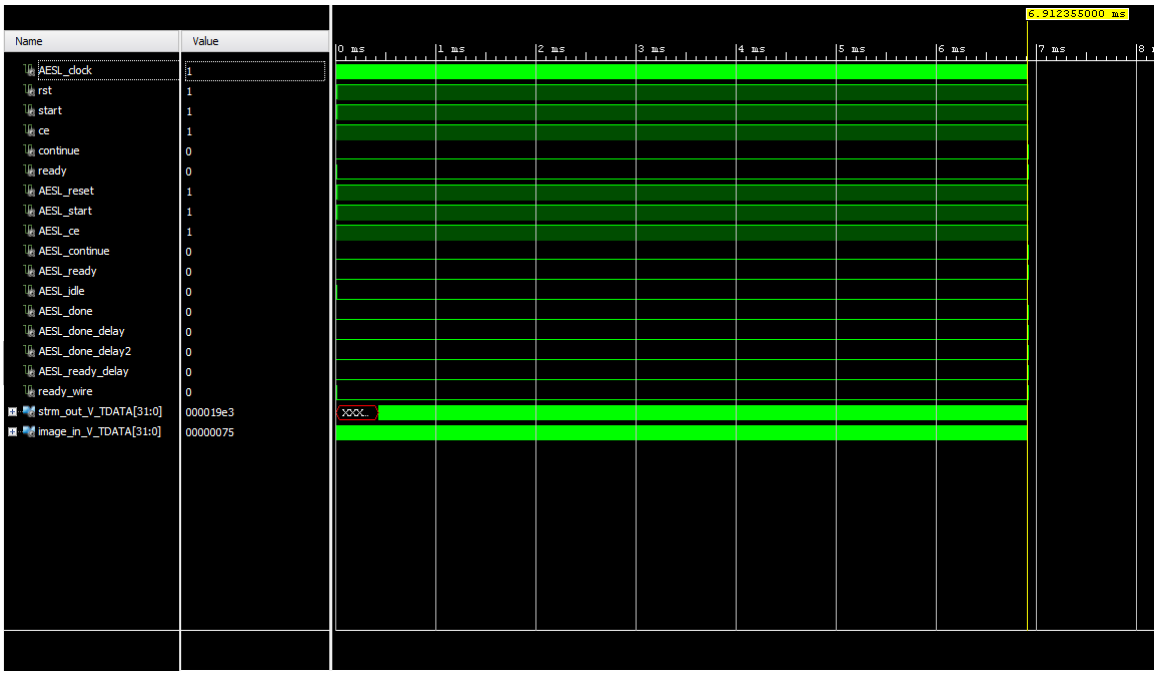
Για την υλοποίηση του αλγορίθμου σε FPGA χρησιμοποιήθηκε η γλώσσα προγραμματισμού C και μέσω του εργαλείου Xilinx Vivado High-Level Synthesis έγινε η μετατροπή σε γλώσσα χαμηλότερου επιπέδου (VHDL) ώστε να εξάγουμε το IP Core για την τελική σχεδίαση. Η χρήση γλώσσας υψηλού επιπέδου μας επιτρέπει να υλοποιήσουμε ευκολότερα και γρηγορότερα πολύπλοκους αλγορίθμους επεξεργασίας εικόνας. Επίσης μας επιτρέπει να επιβεβαιώσουμε τη σωστή λειτουργία της σχεδίασης χρησιμοποιώντας τη γλώσσα προγραμματισμού C, καθώς και να μετατρέψουμε εικόνες σε data streams μέσω της βιβλιοθήκης OpenCV. Κάθε συνάρτηση αποτελεί ένα block στην RTL ιεραρχία και τα ορίσματα της Top-level συνάρτησης καθορίζουν τις εισόδους και εξόδους του συστήματος. Με τη χρήση οδηγιών μπορούμε να κάνουμε βελτιστοποιήσεις, να χρησιμοποιήσουμε παραλληλισμό και να καθορίσουμε ποιοι πόροι της FPGA θα δεσμευτούν για την αποθήκευση των δεδομένων.

4.2 Επιτάχυνση μεθόδου Sum of Absolute Differences

Παρόλο που αποφασίσαμε να μην χρησιμοποιήσουμε τον αλγόριθμο που βασίζεται στη χρήση μεθόδων Template Matching, η επιτάχυνση της τεχνικής υπολογισμού συσχέτισης με τη χρήση του Sum of Absolute Differences παρουσίασε ιδιαίτερο ενδιαφέρον. Ο τρόπος υλοποίησης είναι όμοιος με αυτόν των φίλτρων που χρησιμοποιήθηκαν στην τελική σχεδίαση.

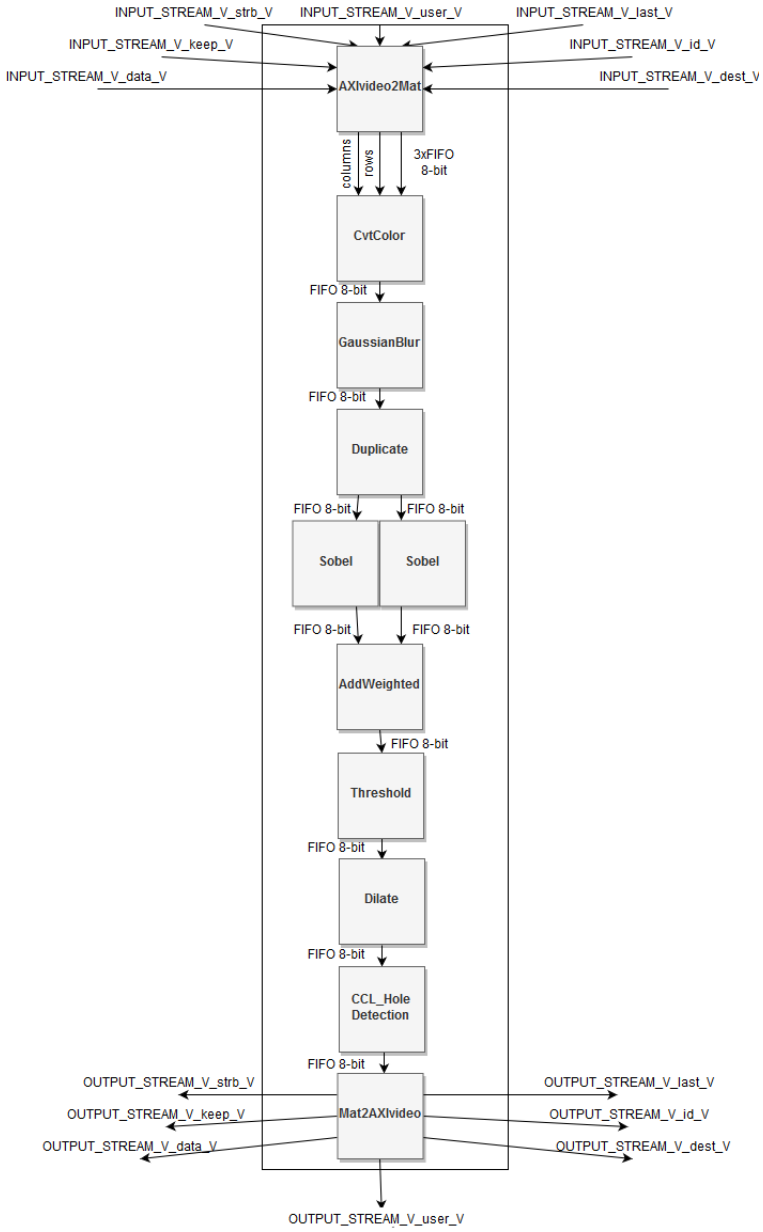
Το module που υλοποιήθηκε δέχεται σαν είσοδο την εικόνα προς επεξεργασία και το πρότυπο και δίνει σαν έξοδο τα αποτελέσματα της συσχέτισης. Αρχικά αποθηκεύεται το πρότυπο στη BRAM. Έπειτα ορίστηκε ένα Memory Line Buffer, με αριθμό γραμμών ίσο με το ύψος του προτύπου και αριθμό στηλών ίσο με το μήκος της εικόνας, το οποίο δέχεται τιμές των 8-bit. Επιπλέον ορίστηκε ένα Memory W-

indow Buffer το οποίο δέχεται τον ίδιο τύπο τιμών και έχει διαστάσεις ίσες με αυτές του προτύπου. Χρησιμοποιώντας ένα for-loop το οποίο εκτελείται για κάθε pixel της εικόνας, διαβάζονται ένα προς ένα τα δεδομένα της εικόνας και αποθηκεύονται στο Line Buffer.Όταν γεμίσει ο Line Buffer, μεταφέρονται τα δεδομένα στο Window Buffer και υπολογίζεται η συσχέτιση μεταξύ των pixel της εικόνας και του προτύπου για κάθε στήλη της εικόνας. Έπειτα μεταφέρεται στο Line Buffer η επόμενη γραμμή της εικόνας ενώ ταυτόχρονα διαγράφεται η πρώτη γραμμή που είναι αποθηκευμένη σε αυτό και επαναλαμβάνεται η ίδια διαδικασία μέχρι να υπολογιστούν όλες οι απαραίτητες τιμές. Για τη χρήση παραλληλισμού χρησιμοποιήθηκε η οδηγία HLS PIPELINE. Ο συνολικός χρόνος εκτέλεσης με ρολόι συχνότητας 100MHz είναι 6.91217ms όπως φαίνεται και στο παρακάτω simulation.



Σχήμα 4.1: Simulation Sum of Absolute Differences για εικόνα διαστάσεων 320x240 και πρότυπο 16x16

4.3 Σχεδιασμός συστήματος



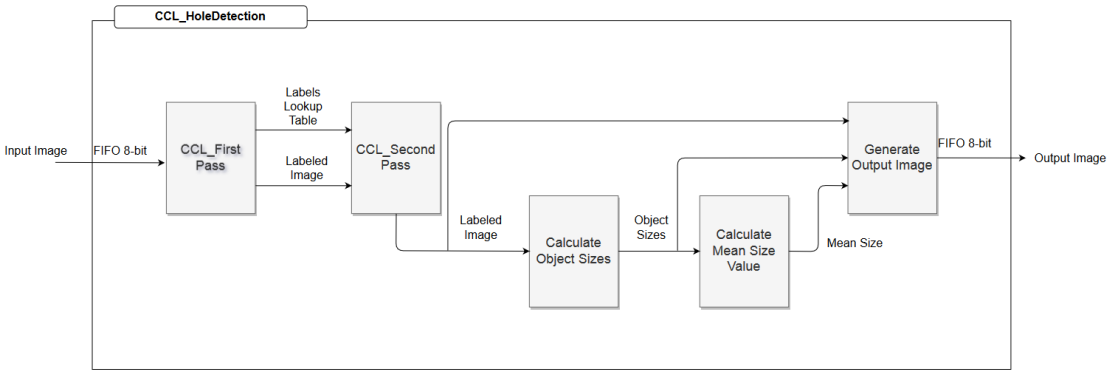
Σχήμα 4.2: Διάγραμμα σχεδίασης

Το σύστημα μας δέχεται σαν είσοδο μία μη κωδικοποιημένη (bmp) έγχρωμη εικόνα,η οποία αποτελείται από τρία κανάλια των 8-bit και δίνει σαν έξοδο μία εικόνα

η οποία παρουσιάζει τα σημεία στα οποία εντοπίστηκε οπή και αποτελείται από ένα μόνο κανάλι (grayscale). Οι εικόνες αποθηκεύονται και διαβάζονται από τη DDR μνήμη της FPGA σειριακά μέσω της χρήσης AXI4-Stream. Η συνολική σχεδίαση παρουσιάζεται στην εικόνα 4.2.

Ακολουθεί η αναλυτική περιγραφή κάθε block:

- AXIvideo2Mat:Αυτό το block είναι υπεύθυνο για τη μετατροπή του stream που λαμβάνεται από την μνήμη σε μεταβλητή τύπου Mat διαστάσεων 320x240 για την περαιτέρω επεξεργασία της εικόνας από τα παρακάτω block.
- CvtColor:Μετατρέπει την έγχρωμη εικόνα σε αποχρώσεις του γκρι.
- GaussianBlurr:Εξομαλύνει την εικόνα, για τη μείωση του θορύβου, χρησιμοποιώντας τη μάσκα:
$$\frac{1}{16}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix}$$
- Duplicate:Δημιουργεί δύο αντίγραφα της εικόνας στη μνήμη. Ουσιαστικά αντιγράφει τις τιμές του FIFO που λαμβάνει ως είσοδο σε δύο νέα FIFO.
- Sobel:Υπολογίζει τη συνέλιξη Sobel χρησιμοποιώντας την μάσκα που περιγράφηκε στην ενότητα 2.1.2. Το ένα block είναι υπεύθυνο για τον υπολογισμό με χρήση της οριζόντιας παραγώγου και το άλλο για την κάθετη. Έχοντας δύο αντίγραφα της εικόνας στη μνήμη οι υπολογισμοί μπορούν να γίνουν παράλληλα και για τις δύο διευθύνσεις.
- AddWeighted:Προσθέτει τα παραπάνω αποτελέσματα στοιχείο προς στοιχείο δίνοντας 0.5 βάρος σε κάθε ένα από αυτά.
- Threshold:Αυτό το block είναι υπεύθυνο για την κατωφλίωση της εικόνας. Τα αποτέλεσμα που έχουν τιμή μεγαλύτερη της τιμής που ορίστηκε ως κατώφλι, αντικαθίστανται από την τιμή 255, ενώ τα υπόλοιπα από την τιμή 0.
- Dilate:Πραγματοποιεί την διαστολή των μη μηδενικών στοιχείων χρησιμοποιώντας kernel διαστάσεων 3x3. Σε αυτό το σημείο όλα τα pixel που αποτελούν κομμάτι του διχτυού είναι λευκά ενώ τα υπόλοιπα είναι μαύρα, δηλαδή έχουν την τιμή 0.
- CCL Hole Detection:Είναι υπεύθυνο για το Connected-component labeling και τον εντοπισμό των οπών. Σαν αποτέλεσμα δίνει μία εικόνα στην οποία φαίνονται τα σημεία όπου εντοπίστηκε οπή.
- Mat2AXIvideo:Μετατρέπει το τελικό αποτέλεσμα σε AXI-Stream για την αποθήκευση στη μνήμη.



Σχήμα 4.3: Διάγραμμα CCL Hole Detection Module

Το module CCL Hole Detection που φαίνεται στην εικόνα 4.3 αποτελείται από τα ακόλουθα block:

- CCL First Pass:Αρχικά διαβάζονται όλα τα pixel της εικόνας που βρίσκονται στην πρώτη γραμμή. Όταν η τιμή του pixel είναι 0, ελέγχεται αν στο προηγούμενο pixel έχει δοθεί τιμή. Αν όχι τότε δίνεται μία νέα τιμή σε αυτό, διαφορετικά λαμβάνει την ίδια τιμή με το προηγούμενο. Ο υπολογισμός της πρώτης γραμμής γίνεται ξεχωριστά από τις υπόλοιπες διότι είναι γνωστό ότι δεν υπάρχει προηγούμενη σειρά. Αυτό έχει ως αποτέλεσμα στις υπόλοιπες σειρές της εικόνας να ελέγχονται και τα pixel που βρίσκονται πάνω από το τρέχων pixel, χωρίς να προηγείται έλεγχος ορίων του πίνακα. Στην περίπτωση όπου ένα γειτονικό pixel έχει τιμή, το τρέχον pixel λαμβάνει την ίδια ταμπέλα. Αν και τα δύο γειτονικά pixel έχουν τιμή, τότε δίνεται σε αυτό η μικρότερη και στον πίνακα αληθείας Labels Lookup Table αποθηκεύεται ότι η τιμή του μεγαλύτερου ισούται με αυτή του μικρότερου. Κάθε pixel της εικόνας που έχει τιμή διαφορετική του μηδέν, παίρνει την τιμή μηδέν στον πίνακα των αποτελεσμάτων. Σαν έξοδος δίνεται το αποτέλεσμα του πρώτου labeling της εικόνας Labeled Image και ένα πίνακας Labels Lookup Table ο οποίος περιέχει την πληροφορία για το ποιες τιμές-ταμπέλες ισούνται μεταξύ τους. Στον Labels Lookup Table κάθε θέση του πίνακα έχει αποθηκευμένη την τιμή της ταμπέλας στην οποία αντιστοιχεί. Για παράδειγμα αν η ταμπέλα 2 ανήκει στο ίδιο αντικείμενο με την ταμπέλα 1 τότε η τιμή του πίνακα στη θέση 1 (2-1) θα είναι 1. Αν όμως η παραπάνω ταμπέλα είναι η μικρότερη που ορίστηκε για κάποιο αντικείμενο τότε θα έχει την τιμή 2.

- CCL Second Pass:Είναι υπεύθυνο για τη διόρθωση των ταμπελών χρησιμοποιώντας τον πίνακα αληθείας Labels Lookup Table. Ελέγχονται όλες οι τιμές του πίνακα Labeled Image και αντικαθίστανται από αυτές που υπάρχουν στον Labels Lookup Table.

- Calculate Object Sizes:Υπολογίζεται το μέγεθος κάθε αντικειμένου τις εικόνας. Αντικείμενα ονομάζονται τα pixel που έχουν την ίδια ταμπέλα. Ελέγχονται σειριακά όλες οι τιμές του πίνακα Labeled Image και κάθε φορά που η τιμή είναι διαφορετική του μηδέν αυξάνεται ο μετρητής για την συγκεκριμένη ταμπέλα κατά ένα.

- Calculate Mean Size Value:Λαμβάνοντας ως είσοδο τον πίνακα Object Sizes υπολογίζει τη μέση τιμή του μεγέθους όλων των αντικειμένων που βρίσκονται στην εικόνα.

- Generate Output Image:Κατασκευάζει μία εικόνα η οποία παρουσιάζει τα σημεία όπου εντοπίστηκε οπή. Αυτό το κάνει ελέγχοντας τις ταμπέλες του πίνακα Labeled Image. Αν κάποια τιμή αντιστοιχεί σε αντικείμενο με διπλάσιο μέγεθος της μέσης τιμής που υπολογίστηκε προηγουμένως τότε ορίζει την τιμή τις εικόνας ίση με 255. Διαφορετικά σε αυτή την θέση το αντίστοιχο pixel της εικόνας θα έχει την τιμή 0.

Στην ανώτατη συνάρτηση η οποία αποτελεί και το Top Module χρησιμοποιήθηκε η οδηγία Dataflow ώστε να εκτελεστούν όσο το δυνατόν περισσότερες συναρτήσεις παράλληλα. Αυτός είναι και ο λόγος που δημιουργήσαμε δύο αντίγραφα της εικόνας πριν τον υπολογισμό Sobel. Κάθε Sobel module είναι υπεύθυνο για μία μόνο διεύθυνση και έχοντας δύο αντίγραφα καταφέραμε να γίνει ο υπολογισμός και προς τις δύο διευθύνσεις ταυτόχρονα.

Current Module : net_holes_detection

	Operation\Control Step	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
1	Block_proc(function)																					
2	AXIvideo2Mat(function)																					
3	CvtColor(function)																					
4	GaussianBlur(function)																					
5	Duplicate(function)																					
6	Sobel_1(function)																					
7	Sobel(function)																					
8	AddWeighted(function)																					
9	Threshold(function)																					
10	Dilate(function)																					
11	CCL(function)																					
12	Mat2AXIvideo(function)																					

Σχήμα 4.4: Ανάλυση εκτέλεσης των module

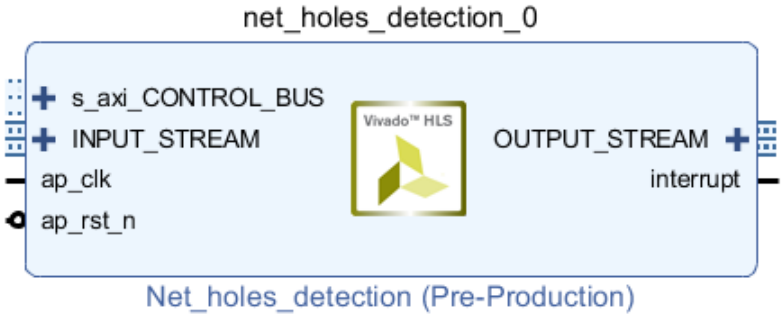
Για την αποθήκευση των δεδομένων Labeled Image και Labels Lookup Table χρησιμοποιήθηκε BRAM. Οι παραπάνω πίνακες αποτελούνται από τιμές των 16bit. Η τιμή αυτή επιλέχτηκε διότι στην χειρότερη περίπτωση όπου κάθε pixel απαιτεί διαφορετική ταμπέλα, τότε σε μία εικόνα διαστάσεων 320x240, η τιμή της μεγαλύτερης ταμπέλας θα είναι 38,400. Η τιμή αυτή μπορεί να αποθηκευτεί σε ένα unsigned integer των 16bit. Ο πίνακας Labels Lookup Table χρησιμοποιήθηκε και για την αποθήκευση του αριθμού εμφάνισης κάθε ταμπέλας κατά τον υπολογισμό του μεγέθους κάθε αντικειμένου. Κατανοούμε ότι σε περίπτωση όπου μία ταμπέλα εμφανίζεται περισσότερες από 65,536 φορές στην εικόνα θα προκύψει overflow, αλλά σε αυτή την περίπτωση το σύστημα μας δεν θα λειτουργεί σωστά διότι σε μία εικόνα που αποτελείτε από

δίχτυα και έχουν ρυθμιστεί σωστά οι απαραίτητοι παράμετροι (Threshold,Dilate) δεν θα υπάρχουν τόσο μεγάλα αντικείμενα.

Σε περίπτωση όπου η αποθήκευση των δεδομένων δεν μπορεί να καλυφθεί από την BRAM θα μπορούσε να χρησιμοποιηθεί η DDR, εφόσον είναι διαθέσιμη, για την αποθήκευση των δεδομένων, πληρώνοντας σε κατανάλωση ενέργειας. Όσο αυξάνονται οι διαστάσεις της εικόνας, η σχεδίαση μας θα γίνεται πιο αργή και θα απαιτεί περισσότερους πόρους. Εμείς επιλέξαμε να χρησιμοποιήσουμε εικόνες διαστάσεων 320x240 ώστε να είναι εφικτή η αποθήκευση των δεδομένων στην BRAM. Στη χειρότερη περίπτωση όπου το latency είναι 1993027 κύκλοι ρολογιού το σύστημά μας μπορεί να εξάγει αποτελέσματα για περίπου 41 εικόνες ανά δευτερόλεπτο, με ρολόι περιόδου 12ns.Ο αριθμός αυτός είναι αρκετά μεγάλος, οπότε είναι εφικτή η επεξεργασία μεγαλύτερων εικόνων σε αρκετά μικρό χρόνο.

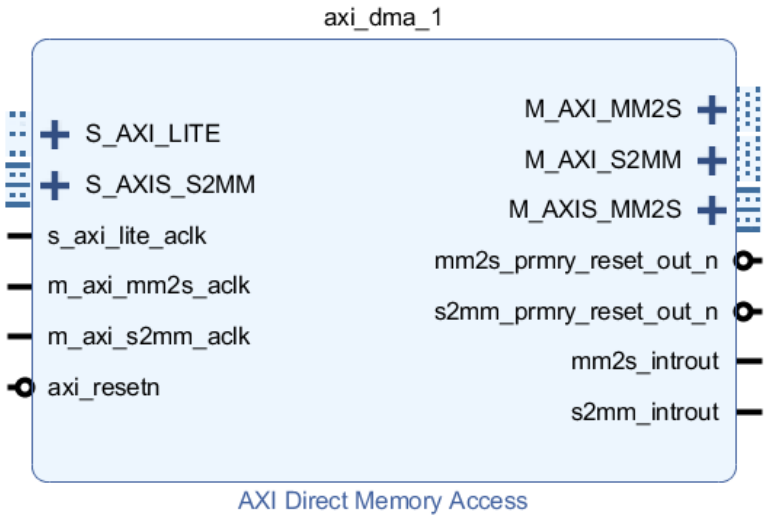
4.4 Τελικό σύστημα

Το τελικό σύστημα χρησιμοποιεί MicroBlaze Soft Processor για τον έλεγχο λειτουργίας της σχεδίασης και υλοποιήθηκε στην Nexys 4 DDR FPGA. Ο επεξεργαστής χρησιμοποιεί για μνήμη ένα μικρό κομμάτι της Block RAM και την DDR. Η μεταφορά των εικόνων γίνεται μέσω UART που επιτρέπει τη σειριακή επικοινωνία του συστήματος με τον υπολογιστή. Μετά την υλοποίηση του αλγορίθμου στο Vivado HLS εξάγαμε το παρακάτω IP Core.



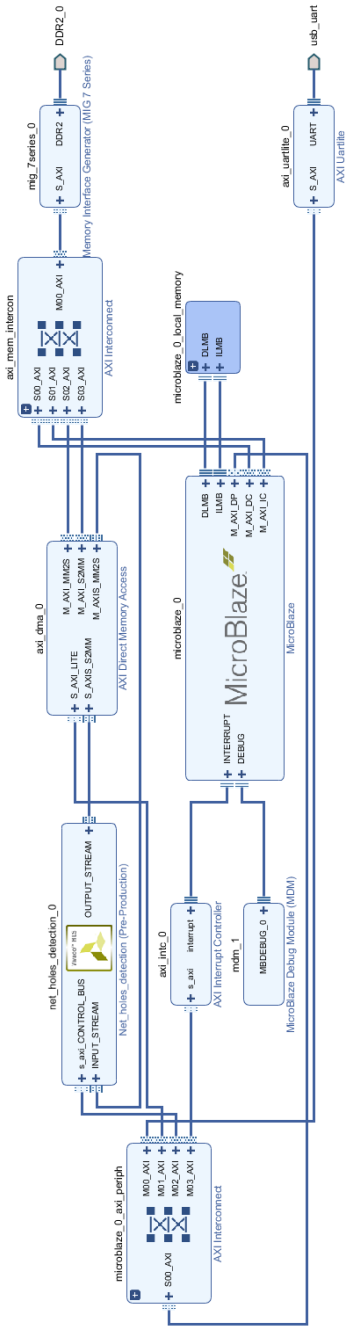
Σχήμα 4.5: IP Core εντοπισμού οπών σε δίχτυα

Μέσω του s_axi_CONTROL_BUS ελέγχεται η λειτουργία του υποσυστήματος εντοπισμού οπών από τον επεξεργαστή. Τα σήματα INPUT_STREAM και OUTPUT_STREAM συνδέονται στο AXI Direct Memory Access (AXI DMA) το οποίο επιτρέπει τη μετακίνηση των δεδομένων μεταξύ της DDR και του IP Core μέσω AXI4 Stream.
Στην εικόνα 4.7 φαίνεται η συνολική σχεδίαση εμφανίζοντας μόνο τις συνδέσεις διεπαφής.



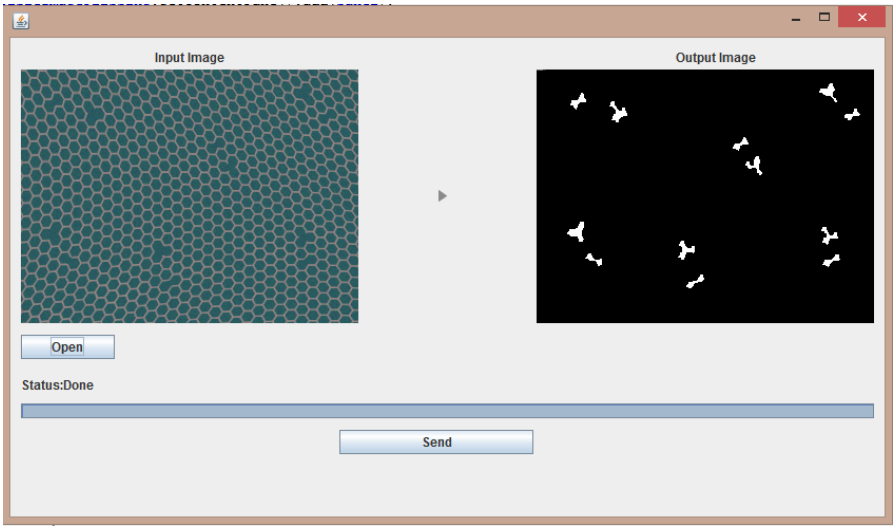
Σχήμα 4.6: DMA

Μετά τη υλοποίηση της συνολική σχεδίασης πραγματοποιήσαμε τον προγραμματισμό του Microblaze χρησιμοποιώντας το Xilinx SDK και τμήματα κώδικα από το παράδειγμα A Zynq Accelerator for Floating Point Matrix Multiplication Designed with Vivado HLS[6].



Σχήμα 4.7: Τελικό Design

Επειδή οι εικόνες μεταφέρονται μέσω σειριακής θύρας και η μεταφορά μεγάλου όγκου δεδομένων απαιτεί πολύ χρόνο, επιλέξαμε το σύστημα μας να δέχεται σαν είσοδο μια grayscale εικόνα.Η αλλαγή αυτή δεν επηρεάζει σημαντικά το συνολικό latency ούτε τη λειτουργία του συστήματος, ενώ ο χρόνος μεταφοράς των δεδομένων μειώθηκε στο ένα τρίτο. Ο Microblaze προγραμματίστηκε ώστε αρχικά να περιμένει να λάβει δεδομένα για εικόνα διαστάσεων 320x240 τα οποία αποθηκεύει στη DDR. Έπειτα ενεργοποιεί το IP Core επεξεργασίας της εικόνας και μόλις ολοκληρωθεί η επεξεργασία, το αποτέλεσμα αποστέλλεται σειριακά στον υπολογιστή. Τα δεδομένα της εικόνας μεταφέρονται byte προς byte με τη χρήση εφαρμογής που υλοποιήθηκε σε Java. Από την εφαρμογή επιλέγεται η εικόνα η οποία θέλουμε να μεταφέρουμε και στην συνέχεια μετατρέπεται σε αποχρώσεις του γκρι και αποστέλλεται στο σύστημα. Μόλις ολοκληρωθεί η αποστολή, περιμένει μέχρι να λάβει τα αποτελέσματα. Μετά την ολοκλήρωση της λήψης εμφανίζεται η εικόνα όπου παρουσιάζονται τα σημεία που εντοπίστηκαν οπές και αποθηκεύεται στον υπολογιστή.



Σχήμα 4.8: Εφαρμογή για την μεταφορά εικόνων μέσω της σειριακής θύρας

Τέλος στους πίνακες 4.1 και 4.2 παρουσιάζεται η συνολική απαίτηση του τελικού συστήματος σε ισχύ και πόρους.

Πίνακας 4.1: Ισχύς τελικού συστήματος

Total On-Chip Power	1.4W
Junction Temperature	31.4^oC
Thermal Margin	53.6^oC (11.6W)
Power supplied to off-chip devices	0.633 W

Πίνακας 4.2: Χρήση πόρων τελικού συστήματος

Resource	***Utilization***	***Available***	***Utilization %***
LUT	17002	63400	26.82
LUTRAM	1401	19000	7.37
FF	16131	126800	12.72
BRAM	106	135	78.52
DSP	41	240	17.08
IO	50	210	23.81
BUFG	5	32	15.63
MMCM	2	6	33.33
PLL	1	6	16.66

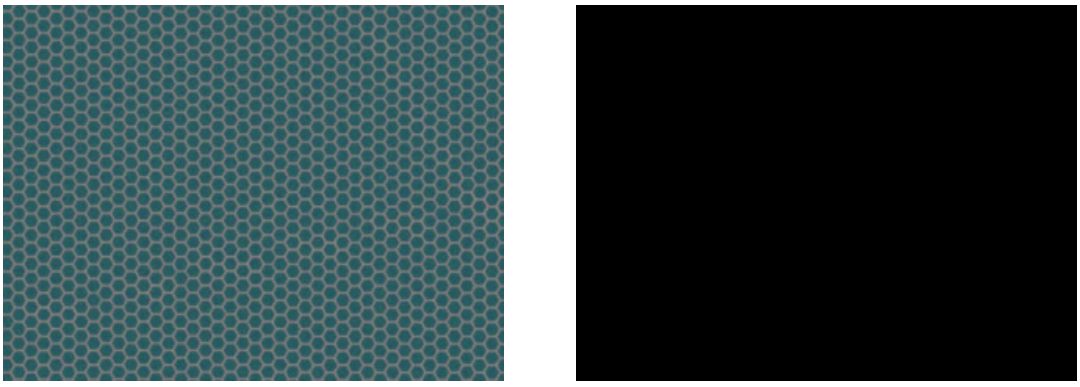
Κεφάλαιο 5

Αποτελέσματα και αξιολόγηση

5.1 Αποτελέσματα

Σε αυτό το κεφάλαιο θα μελετήσουμε τα αποτελέσματα του συστήματος. Αρχικά συγκρίθηκαν τα αποτελέσματα από το C/RTL Cosimulation του αρχικού συστήματος στο Vivado HLS με αυτά του τελικού συστήματος ώστε να διαπιστωθεί ότι οι αλλαγές που κάναμε δεν τα επηρέασαν. Για την αξιολόγηση χρησιμοποιήσαμε τα αποτελέσματα του τελικού συστήματος.

Το σύστημα μας ρυθμίστηκε ώστε να λειτουργεί σωστά σε ιδανικές συνθήκες. Έπειτα δίνοντας ως είσοδο την εικόνα α΄ του σχήματος 5.1 λάβαμε ως αποτέλεσμα την εικόνα β΄. Το γεγονός ότι είναι μαύρη υποδηλώνει ότι δεν εντοπίστηκε καμία οπή.

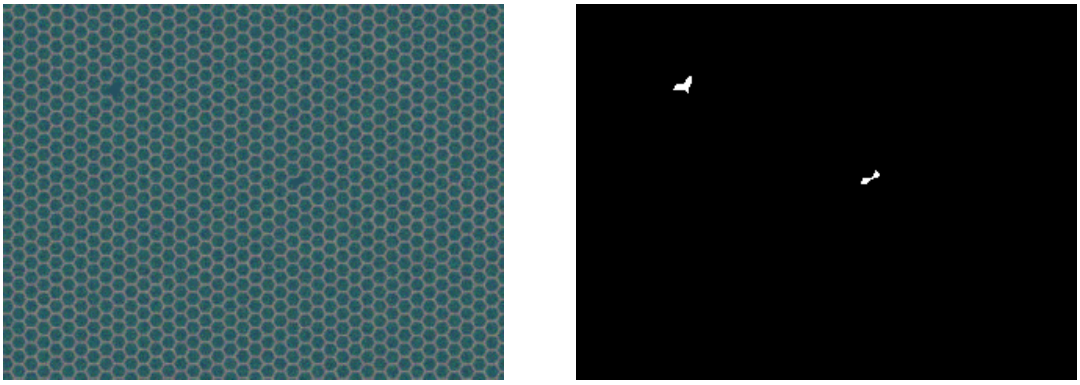


(α΄) Εικόνα προς επεξεργασία (β΄) Αποτελέσματα

Σχήμα 5.1: Δίχτυ χωρίς οπή

Μετά την προσθήκη περισσότερου θορύβου και τη δημιουργία δύο οπών λάβαμε τα

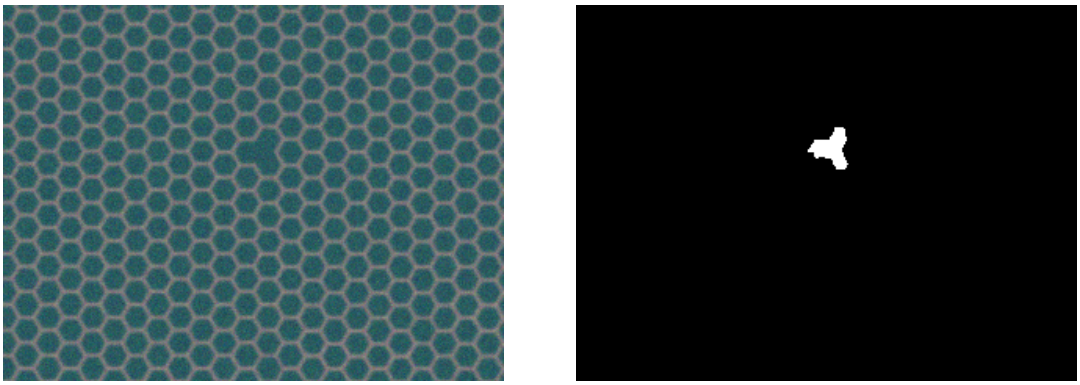
αποτελέσματα του σχήματος 5.2. Παρατηρούμε ότι το φίλτρο θορύβου που χρησιμοποιήσαμε αποδίδει αρκετά καλά ακόμα και σε πιο δυσχερείς συνθήκες.



(α′) Εικόνα προς επεξεργασία (β′) Αποτελέσματα

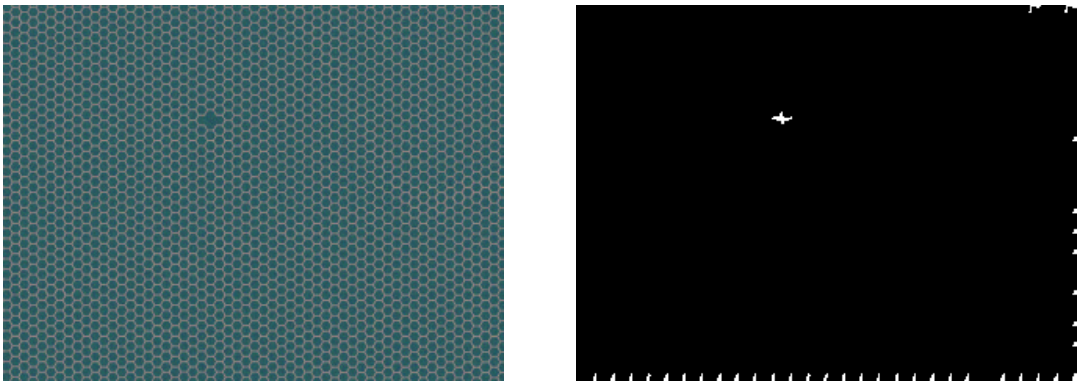
Σχήμα 5.2: Δίχτυ με οπές και υψηλό θόρυβο

Για τη μελέτη της συμπεριφοράς του συστήματος όταν οι εικόνες λαμβάνονται σε διαφορετικές αποστάσεις, προσομοιώσαμε τις περιπτώσεις όπου η εικόνα λαμβάνεται από πολύ κοντά και από πολύ μακριά. Στο σχήμα 5.3 τα αποτελέσματα είναι σωστά, ενώ στο σχήμα 5.4, παρόλο που εντοπίστηκε η οπή έχουν γίνει και λάθη.



(α′) Εικόνα προς επεξεργασία (β′) Αποτελέσματα

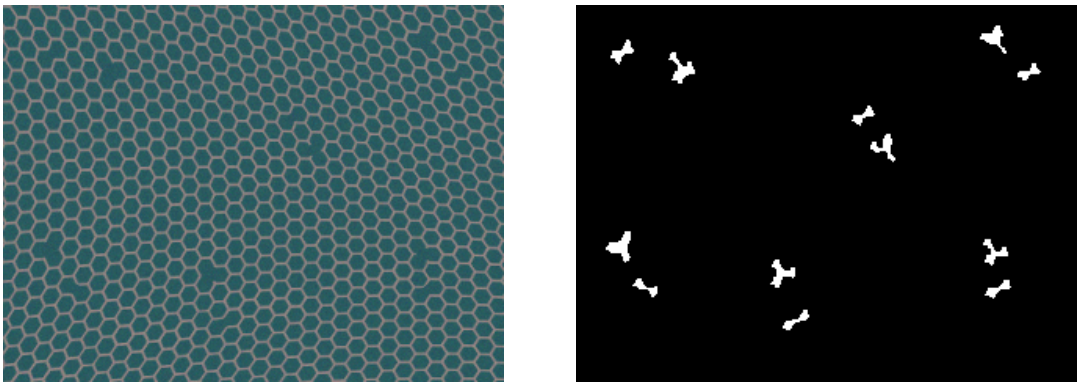
Σχήμα 5.3: Λήψη από μικρή απόσταση



(α΄) Εικόνα προς επεξεργασία (β΄) Αποτελέσματα

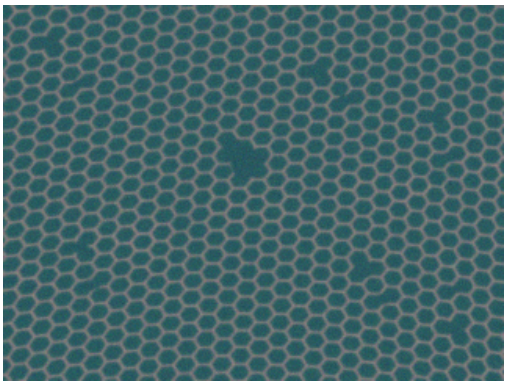
Σχήμα 5.4: Λήψη από μεγάλη απόσταση

Έπειτα ελέγξαμε τα αποτελέσματα του συστήματος σε εικόνες όπου οι μέθοδοι Template Matching είναι αδύναμες. Κατασκευάσαμε μία σειρά εικόνων όπου το δίχτυ είναι παραμορφωμένο (Σχήματα 5.5,5.6,5.7) και έχει κλίση (Σχήματα 5.8,5.9).Έχουν εντοπιστεί όλες οι οπές αλλά έχουν γίνει και κάποια λάθη κυρίως στα όρια της εικόνας. Αυτά οφείλονται στον τρόπο με τον οποίο υπολογίζεται το κατώφλι για το μέγεθος των αντικείμενων. Όπως βλέπουμε πιο ξεκάθαρα στην εικόνα β΄ του σχήματος 5.9, το γεγονός ότι τα αντικείμενα στην πάνω αριστερά άκρη είναι πολύ μικρότερα από αυτά στην κάτω δεξιά, οδηγεί στον υπολογισμό μέσου όρο μεγέθους των αντικειμένων, που έχει σαν αποτέλεσμα να αναγνωρίζονται τα μεγάλα αντικείμενα σαν οπές.



(α΄) Εικόνα προς επεξεργασία (β΄) Αποτελέσματα

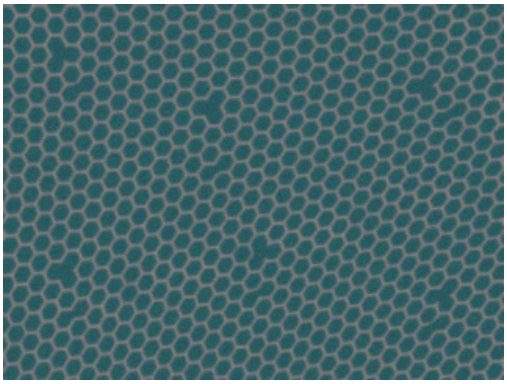
Σχήμα 5.5: Παραμορφωμένο δίχτυ (1)



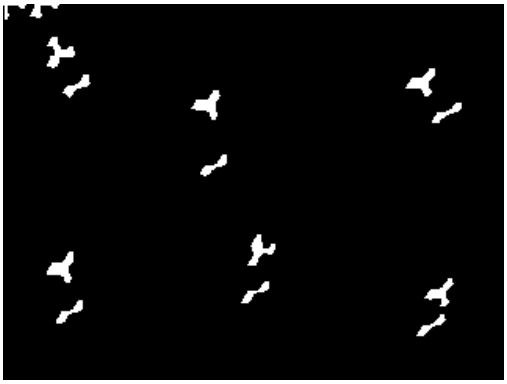
(α′) Εικόνα προς επεξεργασία

(β′) Αποτελέσματα

Σχήμα 5.6: Παραμορφωμένο δίχτυ (2)



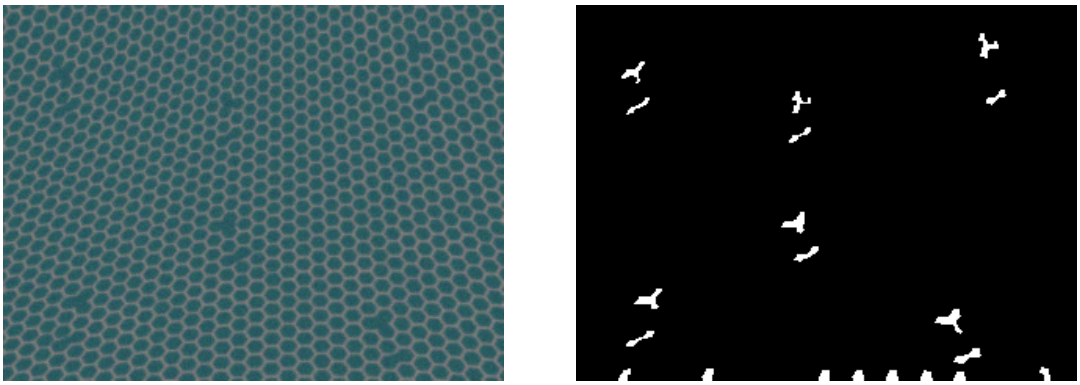
(α′) Εικόνα προς επεξεργασία



(β′) Αποτελέσματα

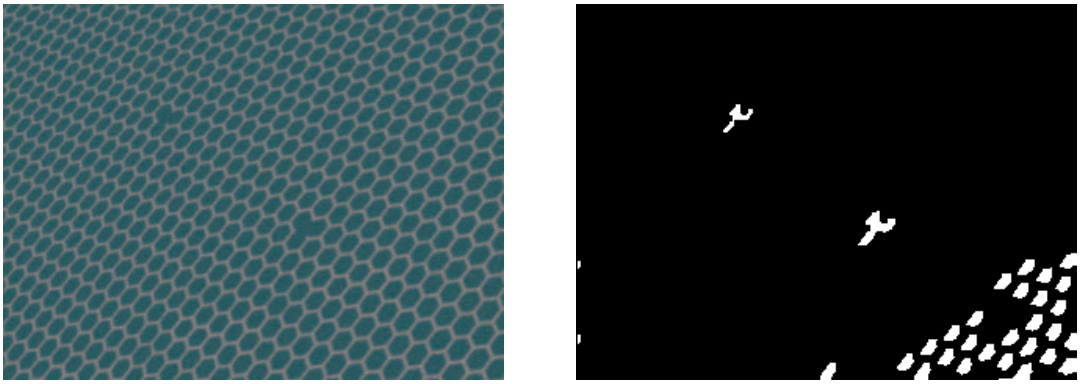
Σχήμα 5.7: Παραμορφωμένο δίχτυ (3)

Όμοια είναι και τα αποτελέσματα στην περίπτωση όπου το δίχτυ στην εικόνα εμφανίζεται περιστραμμένο 5.10. Η οπή έχει εντοπιστεί αλλά στα όρια της εικόνας έχουν γίνει κάποια λάθη.



(α′) Εικόνα προς επεξεργασία (β′) Αποτελέσματα

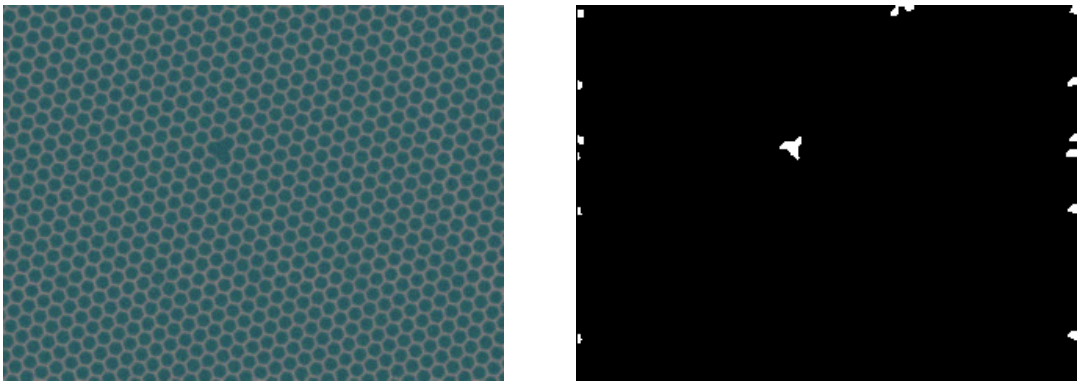
Σχήμα 5.8: Δίχτυ με έντονη παραμόρφωση (4)



(α′) Εικόνα προς επεξεργασία (β′) Αποτελέσματα

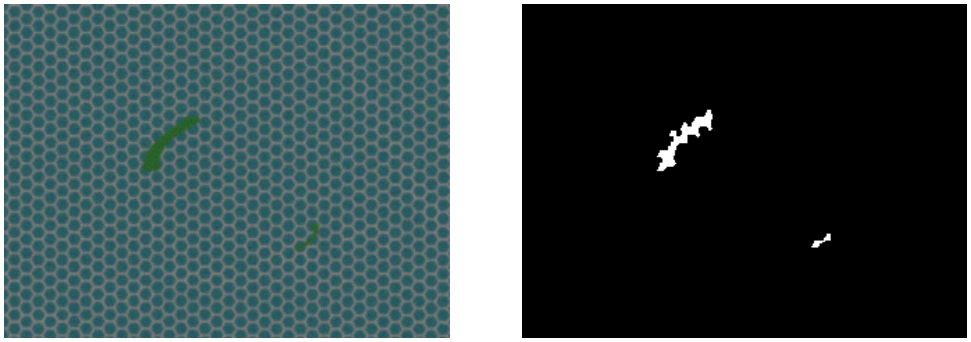
Σχήμα 5.9: Δίχτυ με κλίση (5)

Τέλος μελετήσαμε τη συμπεριφορά του συστήματος στην παρουσία αντικειμένων μπροστά (5.11) και πίσω (5.12) από το δίχτυ. Παρατηρούμε ότι δεν εμφανίζονται λάθη στην περίπτωση όπου τα αντικείμενα βρίσκονται πίσω από το δίχτυ. Αντίθετα όταν τα αντικείμενα είναι μπροστά από αυτό, αναγνωρίζονται σαν οπές.



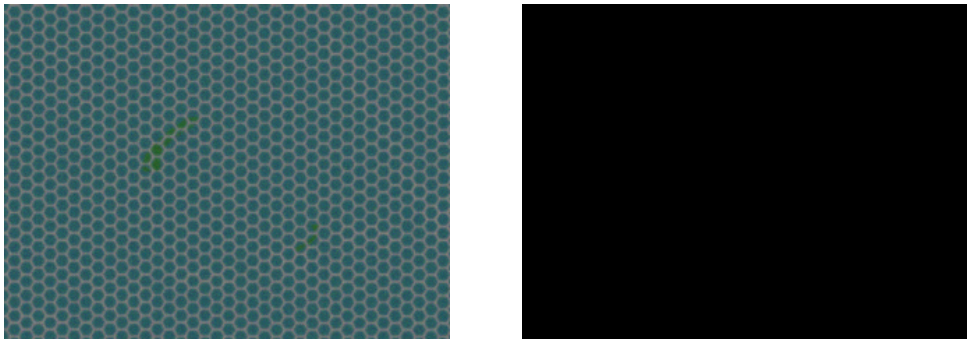
(α′) Εικόνα προς επεξεργασία (β′) Αποτελέσματα

Σχήμα 5.10: Περιστραμμένο δίχτυ



(α′) Εικόνα προς επεξεργασία (β′) Αποτελέσματα

Σχήμα 5.11: Αντικείμενα μπροστά από το δίχτυ



(α′) Εικόνα προς επεξεργασία (β′) Αποτελέσματα

Σχήμα 5.12: Αντικείμενα πίσω από το δίχτυ

5.2 Αξιολόγηση

Από τα παραπάνω αποτελέσματα συμπεραίνουμε ότι το σύστημα μας είναι ικανό να εντοπίσει τις οπές ακόμα και σε αντίξοες συνθήκες.Στο 50% του dataset που εξετάσαμε παρουσιάζονται λάθη γεγονός το οποίο υποδεικνύει ότι το σύστημα μας δεν είναι άψογο.

Η ύπαρξη υψηλού θορύβου μπορεί να επηρεάσει τα αποτελέσματα. Στην εικόνα 5.2 παρατηρούμε ότι παρόλο που εντοπίστηκαν και οι δύο οπές,το σχήμα τους στην εικόνα β΄ είναι παραμορφωμένο. Το Gaussian φίλτρο που χρησιμοποιήθηκε μείωσε το θόρυβο στις υπόλοιπες εικόνες, σε περίπτωση όμως που σε πραγματικές συνθήκες είναι υψηλότερος πρέπει να ρυθμιστεί κατάλληλα.

Ο τρόπος με τον οποίο υπολογίζεται το κατώφλι, για το μέγεθος των αντικειμένων, δίνει ικανοποιητικά αποτελέσματα αλλά σε ακραίες συνθήκες είναι αδύναμος. Τα λάθη που γίνονται στα όρια τις εικόνας μπορούν να παραληφθούν, όμως σε περιπτώσεις όμως που η λήψη της εικόνας γίνεται υπό μεγάλη γωνία σε σχέση με το δίχτυ τότε τα αποτελέσματα θα είναι λάθος.

Στην εικόνα 5.4, οπού εξομοιώνεται η λήψη της εικόνας από μεγαλύτερη απόσταση, παρατηρούμε ότι έχουν γίνει λάθη. Αυτό οφείλεται στο μέγεθος του kernel που χρησιμοποιήθηκε κατά τη διαστολή. Αν σε πραγματικές συνθήκες οι εικόνες λαμβάνονται από μεγάλη απόσταση τότε πρέπει να χρησιμοποιηθεί kernel μικρότερων διαστάσεων για την εξαγωγή καλύτερων αποτελεσμάτων.

Στην περίπτωση όπου υπάρχουν αντικείμενα πίσω από το δίχτυ δεν παρουσιάστηκε κάποιο σφάλμα. Στην εικόνα 5.11 όμως, παρατηρούμε ότι το σύστημα μας αδυνατεί να ξεχωρίσει τα αντικείμενα από τις οπές,όταν βρίσκονται μπροστά από το δίχτυ. Θα μπορούσαμε να ορίσουμε ένα ανώφλι, ώστε αντικείμενα που έχουν πολύ μεγάλο μέγεθος να μην αναγνωρίζονται ως οπές. Σε αυτή την περίπτωση όμως η μεγάλη τρύπα στην εικόνα 5.6 δεν θα είχε εντοπιστεί. Παρατηρούμε όμως ότι και το μικρό αντικείμενο στην εικόνα έχει εντοπιστεί σαν οπή, οπότε ακόμα και αν ρυθμιστεί καλύτερα το σύστημα γνωρίζοντας τα πιθανά μεγέθη οπών, τα αντικείμενα μπροστά από το δίχτυ που έχουν αρκετά μικρό μέγεθος θα ανιχνεύονται ως τρύπες.

Κεφάλαιο 6

Συμπεράσματα και μελλοντικές επεκτάσεις

6.1 Ανασκόπηση

Σε αυτή την διπλωματική εργασία μελετήσαμε τεχνικές επεξεργασίας εικόνας με τις οποίες μπορούμε να εντοπίσουμε οπές σε δίχτυα ιχθυοκαλλιεργειών. Στόχος μας ήταν η κατασκευή ενός ενσωματωμένου συστήματος με τη χρήση FPGA το οποίο θα ήταν ικανό να εξάγει αποτελέσματα σε όσο το δυνατόν λιγότερο χρόνο, με ελάχιστη κατανάλωση ενέργειας και μικρή απαίτηση σε πόρους της FPGA.

Σε πρώτο στάδιο, μελετήσαμε μεθόδους Template Matching και υλοποιήσαμε αλγόριθμο ο οποίος είναι ικανός να εντοπίσει τις οπές. Από όλες τις μεθόδους που μελετήθηκαν ιδιαίτερο ενδιαφέρον παρουσίασαν η Sum of Absolute Difference λόγω της χαμηλής πολυπλοκότητας της και η Normalized Cross-Corellation λόγω της ακρίβειας των αποτελεσμάτων της. Η δεύτερη μέθοδος αποδείχτηκε ιδιαίτερα πολύπλοκη και για αυτό αρκεστήκαμε στη χρήση της πρώτης για την εξαγωγή συμπερασμάτων. Για τον εντοπισμό των οπών χρησιμοποιήθηκε μια μορφή Running Min/Max φίλτρου για το φιλτράρισμα των αποτελεσμάτων κάθε μεθόδου, ώστε να εντοπιστούν τα σημεία που παρουσιάζουν μέγιστη συσχέτιση. Έπειτα τα σημεία αυτά συγκρίνονται με τα γειτονικά τους, υπολογίζοντας το μέσο τετραφωνικό σφάλμα, ώστε να εξακριβωθεί αν αποτελούν οπές ή όχι.

Έπειτα μελετήθηκαν μέθοδοι Edge Detection και υλοποιήθηκε αλγόριθμος ο οποίος είναι ικανός να εντοπίσει τις οπές στην εικόνα με χρήση τεχνικών κατάτμησης της εικόνας και αναγνώρισης αντικειμένων. Ο παραπάνω αλγόριθμος βασίστηκε στη χρήση την μεθόδου Sobel για τον εντοπισμό των ακμών στην εικόνα και έπειτα χρησιμοποιήθηκε αλγόριθμος Connected Component Labeling για τον υπολογισμό του μεγέθους κάθε αντικειμένου.

6.2 Συμπεράσματα

Η παρούσα διπλωματική δεν λύνει το πρόβλημα εντοπισμού των οπών, αλλά αποτελεί ένα πρώτο στάδιο έρευνας πάνω σε αυτό. Λάβαμε όμως πολλές γνώσεις στον τομέα επεξεργασίας εικόνας και στην υλοποίηση τέτοιων μεθόδων σε FPGA.

Όπως είδαμε οι μέθοδοι Template Matching είναι ικανές να εντοπίσουν τις οπές, αλλά σε περιορισμένες συνθήκες. Η χρήση σταθερού προτύπου τις περιορίζει σε μεγάλο βαθμό, διότι όταν το αντικείμενο βρίσκεται παραμορφωμένο στην εικόνα προς επεξεργασία, μειώνεται ο βαθμός συσχέτισης. Ο αλγόριθμος που υλοποιήσαμε βασίστηκε στο γεγονός ότι οι οπές θα συσχετίζονται με το πρότυπο λιγότερο από τα υπόλοιπα σημεία ακόμα και αν αυτά είναι παραμορφωμένα. Το αποτέλεσμα ήταν να εντοπίζονται κάποιες οπές αλλά με μεγάλη πιθανότητα λανθασμένου συμπεράσματος. Επιπλέον η μέθοδος φιλτραρίσματος που χρησιμοποιήθηκε για τον εντοπισμό των σημείων που παρουσιάζουν μέγιστη συσχέτιση αποδείχτηκε ότι είναι πολύ δύσκολο να επιταχυνθεί σε FPGA. Σε εικόνες όπου παρουσιάζονται διαφοροποιήσεις στο φωτισμό η μέθοδος Normalized Cross-Correlation υπερτερεί μακράν σε σύγκριση με τις άλλες μεθόδους, αλλά η αποδοτική υλοποίησή της σε FPGA είναι ιδιαίτερα πολύπλοκη. Μεγάλο ενδιαφέρον παρουσίασε το γεγονός ότι σε εικόνες με υψηλό θόρυβο, όπως αυτές με τα ψάρια στο background που μελετήσαμε, έδιναν αρκετά καλά αποτελέσματα.

Το τελικό σύστημα που υλοποιήθηκε δεν μπορούσε να ξεχωρίσει τα αντικείμενα από τις οπές.Σε αυτή την περίπτωση, υποθέτουμε ότι μία διαφορετική προσέγγιση του αλγόριθμου με τη χρήση μεθόδων Template Matching, θα έδινε τη δυνατότητα διαχωρισμού των αντικειμένων από τις οπές, δεδομένου ότι διατίθεται η πληροφορία για το αντικείμενου που αναζητείται. Στα σημεία της εικόνας όπου εμφανίζεται το αντικείμενο, η συσχέτιση με το πρότυπο θα είναι πολύ μικρότερη συγκριτικά με τα άλλα σημεία ακόμα και αν σε αυτά υπάρχουν οπές.

Για το τελικό σύστημα χρησιμοποιήθηκε η μέθοδος Sobel για τον εντοπισμό των ακμών στη εικόνα. Γενικά οι μέθοδοι Edge Detection επηρεάζονται σημαντικά από την ύπαρξη θορύβου. Τα φίλτρα για τη μείωση του όμως, είναι εφικτό να επιταχυνθούν σε μεγάλο βαθμό σε FPGA, καθώς και όλες οι μέθοδοι επεξεργασίας εικόνας που χρησιμοποιήθηκαν. Παρόλο που ο δεύτερος αλγόριθμος απαιτεί τη χρήση περισσότερων μεθόδων επεξεργασίας εικόνας, επιτεύχθηκε πολύ μικρότερος χρόνος εκτέλεσης από τον πρώτο. Αυτό οφείλεται κυρίως στο γεγονός ότι ο αλγόριθμος Connected Component Labeling που χρησιμοποιήσαμε, ενώ δεν μας επέτρεψε τη χρήση παραλληλισμού, χρησιμοποιεί απλές πράξεις, με αποτέλεσμα να μην απαιτεί πολύ μεγάλο αριθμό κύκλων ρολογιού για την εκτέλεση του. Η μέθοδος που χρησιμοποιήθηκε για τον διαχωρισμό των οπών από τα υπόλοιπα σημεία είναι ιδιαίτερα αδύναμη σε περιπτώσεις όπου το δίχτυ στην εικόνα εμφανιζόταν έντονα παραμορφωμένο. Πιθανότατα ένα τρόπος σύγκρισης των μεγεθών με τα υπόλοιπα σημεία να έδινε καλύτερα αποτελέσματα. Τέλος, όσον αφορά την αδυναμία διαφοροποίησης των αντικειμένων

μπροστά από το δίχτυ, σε περιπτώσεις όπου το αντικείμενο είναι πολύ μεγαλύτερο από το μέγεθός των πιθανών οπών, θα ήταν εφικτό να εντοπιστεί και να παραληφθεί από τα τελικά αποτελέσματα. Σε αυτή την περίπτωση όμως δεν θα ήταν εφικτός ο εντοπισμός μεγάλων οπών.

6.3 Μελλοντική Εργασία

Λόγω της υψηλής πολυπλοκότητας του προβλήματος και έλλειψης γνώσεων και χρόνου δεν ήταν εφικτή η δοκιμή πολυπλοκότερων μεθόδων οι οποίες θα έλυναν το πρόβλημα με μεγαλύτερο βαθμό επιτυχίας και λιγότερους περιορισμούς.

Όπως είδαμε από την παραπάνω μελέτη, καμία τεχνική δεν είναι ικανή να μας δώσει καλά αποτελέσματα από μόνη της. Οι μέθοδοι Template Matching εξαρτώνται από το πρότυπο και δεν δίνουν καλά αποτελέσματα όταν αυτό εμφανίζεται παραμορφωμένο στην εικόνα προς επεξεργασία, ενώ οι τεχνικές Edge Detection επηρεάζονται σημαντικά από το θόρυβο.

Η εξάρτηση των μεθόδων Template Matching από το πρότυπο μπορεί να αντιμετωπιστεί υπολογίζοντας τη συσχέτιση τμημάτων της εικόνας λαμβάνοντας το template μέσα από αυτή. Αυτό πιθανότατα μπορεί να γίνει χρησιμοποιώντας μεθόδους εντοπισμού σημείων που παρουσιάζουν ενδιαφέρον. Μία τέτοια μέθοδος μπορεί να αποτελέσει ο αλγόριθμος που υλοποιήσαμε με Edge Detection, ο οποίος μπορεί να αναβαθμιστεί με τη χρήση της τεχνικής Canny και την πιο αποδοτική υλοποίηση του αλγορίθμου Connected Component Labeling. Άλλος ένας τρόπος ίσως είναι ο εντοπισμός γωνιών με τη μέθοδο Harris Edge Detection.

Τέλος είναι απαραίτητη η υλοποίηση του συστήματος με χρήση κάμερας και η μελέτη σε πραγματικά δεδομένα, ώστε να είναι πιο ακριβής η τελική σχεδίαση και να λειτουργεί σε πραγματικές συνθήκες.

Βιβλιογραφία

[1] Ilkoo Ahn and Changick Kim, *Finding defects in regular-texture images*, Korea Advanced Institute of Science and Technology (2009).

[2] D. G. Bailey C. T. Johnston, K. T. Gribbon, *Implementing image processing algorithms on fpgas*, IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (2013).

[3] Rafael C.Gonzalez and Richard E.Woods, *Digital image processing*, Chapter 10.2.

[4] OpenCV Documentation, *Template matching* *docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html* (english).

[5] Bruce A. Draper, J. Ross Beveridge, A.P. Willem Böhm, Charles Ross, Monica Chawathe, and Jeffrey Hammes, *Accelerated image processing on fpgas*, IEEE Transactions on Image Processing (2003).

[6] A Zynq Accelerator for Floating Point Matrix Multiplication Designed with Vivado HLS, *www.xilinx.com/support/documentation/application_notes/xapp1170-zynq-hls.pdf*.

[7] James L Crowley Jerome Martin, *Comparison of correlation techniques*, Conference on Intelligent Autonomous Systems (1995).

[8] Wen-Chieh Lin, J. Hays, Chenyu Wu, V. Kwatra, and Yanxi Liu, *Quantitative evaluation of near regular texture synthesis algorithms*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2006).

[9] Neelam Sharma Paridhi Swaroop, *An overview of various template matching methodologies in image processing*, International Journal of Computer Applications (2016).

[10] M. Park, K. Brocklehurst, Robert Collins, and Yanxi Liu, *Deformed lattice detection in real-world images using mean-shift belief propagation*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2009) (english).

[11] Vaibhav Gadewar Radhika Chandwadkar, Saurabh Dhole, Deepika Raut, and Prof. S. A. Tiwaskar, *Comparison of edge detection techniques*, 6th Annual Conference of IRAJ (2013).

[12] Kurt Schwenk and Felix Huber, *Connected component labeling algorithm for very complex and high resolution images on an fpga platform*, SPIE Remote Sensing. International Society for Optics and Photonics (2015).

[13] N. Senthilkumaran and R. Rajesh, *Edge detection techniques for image segmentation – a survey of soft computing approaches*, International Journal of Recent Trends in Engineering, Vol. 1, No. 2 (2009).