ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Έλεγχος Πρόσβασης και Επίβλεψη Κάτω Όψης Οχημάτων με τη Χρήση Ευφυών Αλγορίθμων Επεξεργασίας Εικόνας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κοσιάρη Θ. Αλέξανδρου

Επιβλέπων: Λούμος Βασίλειος
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2006

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ελεγχος Πρόσβασης και Επίβλεψη Κάτω Οψης Οχημάτων με τη Χρήση Εφυών Αλγορίθμων Επεξεργασίας Εικόνας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κοσιάρη Θ. Αλέξανδρου

Επιβλέπων: Λούμος Βασίλειος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Μαρτίου 2006

.....	.....	.....
Λούμος Βασίλειος	Καγιάφας Ελευθέριος	Θεολόγου Μιχαήλ
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2006

.....

Κοσιάρης Θ. Αλέξανδρος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της διπλωματικής εργασίας ήταν η ανάπτυξη μίας μεθοδολογίας ανίχνευσης ξένων, ως προς ένα όχημα αντικειμένων, τοποθετημένων στην κάτω όψη του αμαξώματος. Έγινε χρήση προηγμένων μεθόδων ανάλυσης και επεξεργασίας εικόνων, ώστε η διαδικασία ανίχνευσης να είναι όσο το δυνατόν πιο ακριβής.

Συγκεκριμένα, έγινε μελέτη μεθόδων για την ταυτοποίηση οχημάτων μέσω της πινακίδας κυκλοφορίας, τους ακολουθούμενη από μεθόδους ταυτοποίησης εικόνων διαφοροποιούμενων μεταξύ τους, κατά γωνία και μετατόπιση, καθώς και μεθόδων ανίχνευσης και αναγνώρισης αλλαγών.

Η ανάπτυξη του συστήματος έγινε κυρίως στο πρόγραμμα Labview της εταιρείας National Instruments καθώς και με εγγραφή κώδικα για το περιβάλλον .NET της εταιρείας Microsoft. Η μεθοδολογία που αναπτύχθηκε είναι γενικευμένη και μπορεί να χρησιμοποιηθεί για ανίχνευση διαφορών και σε άλλες περιπτώσεις πέραν της αναφερόμενης. Η γενικότητας της έγκειται στο γεγονός ότι οι μέθοδοι που ακολουθήθηκαν είναι καλά τεκμηριωμένες και πλήρεις ενώ παρουσιάζουν μεγάλη ανοχή στο θόρυβο.

Λέξεις Κλειδιά

Ανίχνευση αλλαγών, αμάξωμα, ταυτοποίηση εικόνων, θόρυβος,
παρακολούθηση μέσω κάμερας.

Abstract

The scope of this thesis was the development of a methodology in order to detect foreign objects under the body of a car. Advanced methods for analyzing and processing images where used in order for the detection process to be as accurate as possible.

Specifically, methods of car identification through their license plate were studied followed by methods of registration between images different by an angle and distance as well as change detection methods.

The system was primarily developed on the Labview framework provided by National Instruments Corporation as well as the .NET framework provided by Microsoft Corporation. The developed methodology is generalized and can be used to detect differences in cases other than the stated one. The usefulness of the methodology is the well documented methods followed as well as the fact that they are resistant to noise.

Key-Words

Change detection, car body, image registration, noise, under-vehicle camera surveillance.

Ευχαριστίες

Στη διάρκεια της εκπόνησης της διπλωματικής μου εργασίας, μου προσφέρθηκε πολύτιμη βοήθεια από τους καθηγητές μου. Πρώτα από όλα, θα ήθελα να ευχαριστήσω τον επιβλέποντα τη διπλωματική μου εργασία Καθηγητή κ. Βασίλειο Λούμο για την ανεκτίμητη επιστημονική και ηθική υποστήριξή του, καθώς και τον Καθηγητή κ. Ελευθέριο Καγιάφα για την επιστημονική καθοδήγηση που μου προσέφερε. Τους ευχαριστώ επίσης θερμά γιατί μου προσέφεραν την ευκαιρία να κάνω χρήση του Εργαστήριου Τεχνολογίας Πολυμέσων. Θα ήθελα ακόμη να ευχαριστήσω τον καθηγητή κ. Μιχαήλ Θεολόγου ο οποίος ως μέλος της τριμελούς επιτροπής, στήριξε την προσπάθειά μου.

Επίσης θα ήθελα να ευχαριστήσω τον κ. Θεόδωρο Αλεξανδρόπουλο για την συνεχή βοήθεια και τις συμβουλές του σε κάθε στάδιο της εργασίας αυτής καθώς και τον κ. Χρήστο Αναγνωστόπουλο που με βοήθησε στην εξοικείωση μου με το εργαστήριο και την τεχνολογία εικόνας. Ευχαριστώ επίσης όλους εκείνους που με τον έναν ή τον άλλον τρόπο βοήθησαν να φέρω σε πέρας αυτή την εργασία.

ΠΕΡΙΕΧΟΜΕΝΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Εισαγωγή

Σκοπός αυτής της διπλωματικής εργασίας ήταν η εκπόνηση μίας μεθόδου ανίχνευσης ξένων αντικειμένων στην κάτω όψη του αμαξώματος ενός οχήματος. Με τον όρο «ξένα» εννοούμε αντικείμενα τα οποία, σε ένα προηγούμενο, σε ελεγχόμενο χώρο, έλεγχο του αμαξώματος, δεν υπήρχαν εκεί. Τέτοια αντικείμενα μπορεί να έχουν βρεθεί εκεί είτε από κάποια αλλαγή που ο ίδιος ο ιδιοκτήτης έχει ζητήσει, είτε από κάποιο ατύχημα είτε εν αγνοία του από κάποιον δολιοφθορέα. Σε κάθε μία περίπτωση από τις παραπάνω σκοπός είναι να ανιχνεύονται ώστε να λαμβάνει χώρα η κατάλληλη ενέργεια. Η εφαρμογή είναι δυνατόν να χρησιμοποιηθεί κατά κόρον σε χώρους ελεγχόμενης στάθμευσης όπως στην Βουλή των Ελλήνων , σε πρεσβείες , υπουργεία , αστυνομικά τμήματα και σε χώρους γενικών χρήσεων όπου πρόσβαση έχουν συγκεκριμένα διαπιστευμένα οχήματα. Κάθε όχημα το οποίο θα ελέγχεται από την εφαρμογή πρέπει να έχει περάσει από έναν αρχικό έλεγχο και να έχουν ληφθεί φωτογραφίες της κάτω όψης του καθώς και ο αριθμός κυκλοφορίας του.

Για το λόγο αυτό μελετήθηκαν αρχικώς αλγόριθμοι ταυτοποίησης οχημάτων μέσω της πινακίδας κυκλοφορίας. Ο σκοπός είναι να συνδέεται άρρηκτα μέσω της πινακίδας κυκλοφορίας που είναι μοναδική για κάθε όχημα, μία εικόνα της κάτω όψης του οχήματος με συγκεκριμένα δεδομένα στα αρχεία της εφαρμογής μας.

Μετά την ταυτοποίηση μελετήθηκαν αλγόριθμοι ταυτοποίησης μεταξύ της αρχικής εικόνας, που έχουμε στα αρχεία μας και έχει ληφθεί κατά τον αρχικό έλεγχο του οχήματος και μίας τρέχουσας εικόνας που λαμβάνεται κατά τη στιγμή του τρέχοντος ελέγχου. Δεδομένου ότι λόγω μετατόπισης του οχήματος είναι δυνατόν οι δύο φωτογραφίες να διαφέρουν κατά θέση και γωνία, κρίθηκε απαραίτητο να αναπτυχθεί αλγόριθμος που να ταυτοποιεί καταλλήλως τις δύο φωτογραφίες.

Τέλος, για την αναγνώριση αλλαγών μεταξύ των δύο εικόνων αναπτύχθηκε ένας τρίτος αλγόριθμος που βάσει της απόλυτης διαφοράς μεταξύ της αρχικής φωτογραφίας και της τρέχουσας ανακαλύπτει περιοχές του αμαξώματος που έχουν υποστεί μεταβολή.

Για τη σωστή λειτουργία του όλου συστήματος απαραίτητος είναι ο ανθρώπινος παράγοντας, ο οποίος θα πρέπει να ελέγχει κατά πόσο τα στοιχεία των αρχείων, όπως μοντέλο και μάρκα οχήματος, ανταποκρίνονται στην πραγματικότητα αλλά και για να ελέγχει τις πιθανές περιοχές ενδιαφέροντος και να ενημερώνει καταλλήλως τους αρμόδιους.

Η συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί με τις κατάλληλες τροποποιήσεις σε οποιαδήποτε περίπτωση χρειάζεται ταυτοποίηση και ανακάλυψη διαφορών μεταξύ εικόνων.

ΜΕΡΟΣ ΠΡΩΤΟ

ΤΕΧΝΙΚΗ ΑΝΙΧΝΕΥΣΗΣ ΘΕΣΗΣ ΠΙΝΑΚΙΔΑΣ ΚΥΚΛΟΦΟΡΙΑΣ ΟΧΗΜΑΤΟΣ

VEHICLE LICENSE PLATE PLACE RECOGNITION

Μέρος Πρώτο

Εισαγωγή

Σκοπός της επεξεργασίας εικόνας είναι η απόκτηση χρήσιμων πληροφοριών από αυτήν. Επειδή ο όγκος των δεδομένων που περιέχει μία εικόνα είναι τεράστιος, είναι αναγκαία η ανάπτυξη κάποιων διαδικασιών κατάτμησης και απαρίθμησης αντικειμένων. Στη συνέχεια η εξαγωγή χαρακτηριστικών από αυτά τα αντικείμενα επιτρέπει την περαιτέρω απόκτηση πληροφορίας.

Η πληροφορία που αρχικά αναζητούμε είναι ο αριθμός κυκλοφορίας του οχήματος, τον οποίο θα ανακτήσουμε μέσω της πινακίδας κυκλοφορίας. Αυτή αποτελεί την περιοχή ενδιαφέροντος (ROI - Region Of Interest).

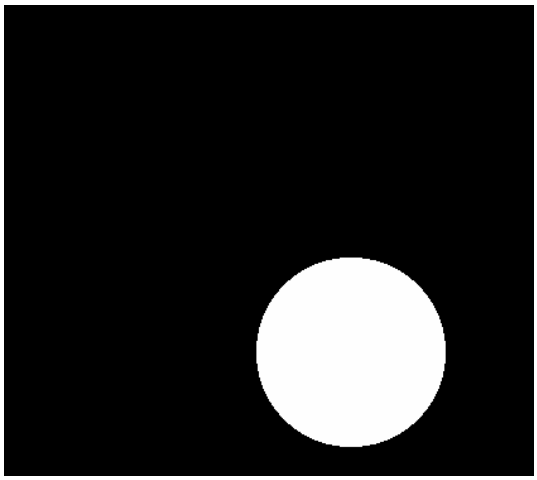
Πριν από το στάδιο της επεξεργασίας υπάρχει το στάδιο της προ-επεξεργασίας κατά το οποίο γίνεται προσπάθεια να απομονωθούν πιθανές περιοχές ενδιαφέροντος αλλά και να προσαρμοστούν οι πιθανές διαφορετικές συνθήκες φωτισμού που μπορεί να υπάρχουν. Η συμβολή αυτής της διπλωματικής εργασίας είναι κυρίως στο στάδιο της προεπεξεργασίας και συγκεκριμένα στο να καθοριστούν οι συντεταγμένες της πινακίδας στην αρχική εικόνα.

Η λογική πράξη ΚΑΙ (AND) χρησιμοποιείται στην επεξεργασία εικόνας για το συνδυασμό πληροφοριών από δύο ή περισσότερες εικόνες. Χρησιμοποιώντας τη λογική πράξη ΚΑΙ με κάποια κατάλληλη «εικόνα μάσκα» είναι δυνατόν να καθορίσουμε επακριβώς την περιοχή ενδιαφέροντος σε μία εικόνα. Ουσιαστικά, χρησιμοποιείται ο πολλαπλασιασμός κάθε στοιχείου εικόνας (pixel) με το αντίστοιχό τους στη μάσκα. Η μάσκα είναι μία δυαδική εικόνα που ουσιαστικά έχει μόνο άσπρες η μαύρες περιοχές. Έτσι κάθε στοιχείο εικόνας (pixel) στο αποτέλεσμα είτε θα έχει μηδενική τιμή είτε την

τιμή που είχε πριν. Η διαδικασία καλείται masking και ένα παράδειγμα χρήσης φαίνεται στις επόμενες εικόνες 1α,1β,1γ.

Η 1α είναι η αρχική ενώ η 1β είναι μια δυαδική εικόνα, που δημιουργήθηκε ειδικά για τις ανάγκες της επίδειξης. Η 1γ είναι το αποτέλεσμα της πράξης και παρατηρούμε ότι πλέον υπάρχει μόνο η συγκεκριμένη περιοχή στην εικόνα.

Εικόνα 1α



Εικόνα 1β

Εικόνα 1γ

Περιγραφή Αλγορίθμου

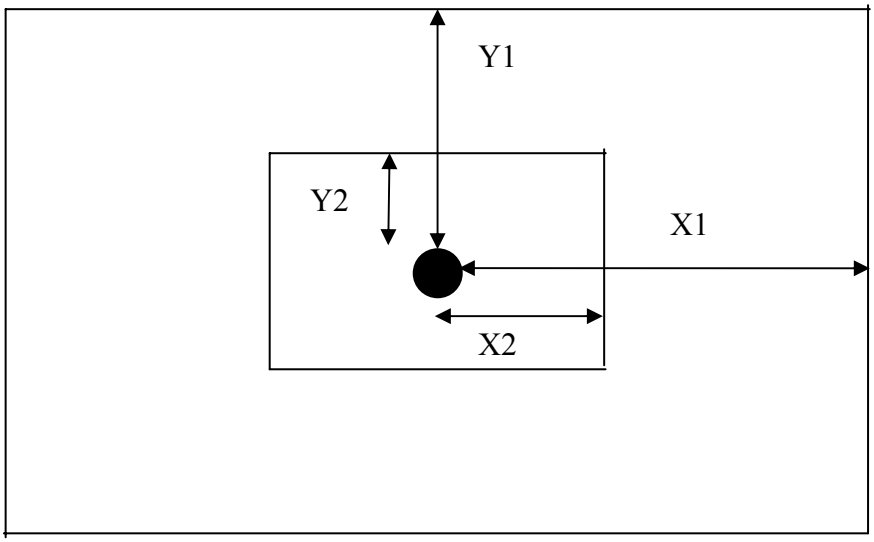
Γίνεται φανερό ότι η διαδικασία που πρέπει να ακολουθηθεί για τον εντοπισμό της πινακίδας κυκλοφορίας του οχήματος δεν θα πρέπει να εξαρτάται από παράγοντες όπως:

- Διαστάσεις της εικόνας
- Ανάλυση της κάμερας
- Απόσταση μεταξύ της εικόνας και του οχήματος

Για αυτόν το λόγο αναπτύχθηκε η ακόλουθη σκέψη. Η περιοχή της πινακίδας κυκλοφορίας είναι θεωρητικά περιοχή που παρουσιάζονται ανωμαλίες στην υφή της εικόνας. Κατά συνέπεια απότομες αλλαγές στα χαρακτηριστικά μεγέθη της εικόνας προδίδουν την πιθανή παρουσία πινακίδας κυκλοφορίας.

Συγκεκριμένα η τυπική απόκλιση των τιμών της κλίμακας του γκρι στην περιοχή της πινακίδας είναι μεγάλη. Αυτό όμως δεν είναι αρκετό για να την εντοπίσουμε επιτυχώς, αφού κάτι τέτοιο συμβαίνει και σε άλλα σημεία της εικόνας. Ένα πρόσθετο χαρακτηριστικό των πινακίδων κυκλοφορίας είναι ότι υπάρχει μεγάλη διαφορά των τιμών της κλίμακας του γκρι στα άκρα τους καθώς και στα νούμερα και επίσης ο λόγος πλάτους προς ύψος τους είναι μεγαλύτερος του δύο (2).

Έτσι χρησιμοποιήθηκε μία νέα μέθοδος κατάτμησης, ο αλγόριθμος κυλιόμενων ομόκεντρων παραθύρων. Αυτή η μέθοδος ορίζει μία χαρακτηριστική τιμή για κάθε στοιχείο εικόνας (pixel) ξεχωριστά (pixel-based). Εξετάζονται δύο γειτονικές περιοχές , ομόκεντρες, αλλά διαφορετικού μεγέθους και εάν πληρούν μία συγκεκριμένη συνθήκη, η τιμή του στοιχείου εικόνας αλλάζει. Στην περίπτωση μας οι δύο περιοχές είναι δύο ορθογώνια παραλληλόγραμμα, με μήκη και πλάτη X1, Y1 και X2,Y2 και κέντρο τους το υπό εξέταση στοιχείο εικόνας.[1]



Ο χρήστης στο συγκεκριμένο αλγόριθμο μπορεί να ορίσει τις διαστάσεις των παραθύρων κατά τη θέληση του. Στην περίπτωσή μας ορίστηκαν σε X1=10, Y1=4, X2=5, Y2=2. Με αυτό τον τρόπο πλησιάζουμε περισσότερο προς τις διαστάσεις μίας πινακίδας κυκλοφορίας.

Οι συνθήκες, βάσει των οποίων αλλάζει η τιμή ενός στοιχείου εικόνας , (pixel) ορίζονται ως εξής. [1]

$$\text{Αν } \left|(k_1 M_A + k_2 std_A) - (k_3 M_B + k_4 std_B)\right| > T_v \quad \tau\acute{o}\,\tau\varepsilon \quad I(x,y) = I_v$$

Όπου k_1, k_2, k_3, k_4 σταθερές και

$$M_A = \frac{1}{4X_1Y_1} \sum_{4X_1Y_1} I(x,y)$$

$$M_B = \frac{1}{4X_2Y_2} \sum_{4X_2Y_2} I(x,y)$$

$$std_A = \sqrt{\frac{1}{4X_1Y_1} \sum_{4X_1Y_1} (I(x,y) - M_A)^2}$$

$$std_B = \sqrt{\frac{1}{4X_2Y_2} \sum_{4X_2Y_2} (I(x,y) - M_B)^2}$$

Όπου I(x,y) η τιμή της φωτεινότητας του εν λόγω στοιχείου εικόνας(pixel) και ν ο αριθμός των κατωφλίων που θέλουμε να χρησιμοποιήσουμε. Προφανώς μπορούμε να έχουμε μόνο μία τιμή κατωφλίωσης οπότε ουσιαστικά η διαδικασία ισοδυναμεί με διαδικοποίηση και το αποτέλεσμα είναι μία μάσκα. Στην περίπτωση την οποία μελετάμε επιλέξαμε να έχουμε μόνο μία τιμή κατωφλίωσης, ενώ η τιμή κατωφλίου που επιλέχθηκε ήταν ίση με το ένα τέταρτο της τυπικής απόκλισης όλης της εικόνας. Τέλος δεδομένου ότι μας ενδιέφερε μόνο η τιμή της τυπικής απόκλισης, οι σταθερές k_1, k_3 ήταν ίσες με το μηδέν(0) ενώ οι k_2, k_4 ίσες με την μονάδα(1).

Το αποτέλεσμα της προηγούμενης μεθόδου είναι μία δυαδική εικόνα, η οποία χρησιμοποιείται ως μάσκα πάνω στην αρχική για την παραγωγή μίας νέας εικόνας.

Προφανώς η νέα αυτή εικόνα δεν είναι ακριβώς αυτό που θέλουμε. Για αυτό το επόμενο βήμα είναι η κατωφλίωση της εικόνας βάση κάποιας τιμής. Κατωφλίωση σημαίνει ότι επιλέγουμε μία τιμή και οποιοδήποτε στοιχείο μέσα στην εικόνα έχει φωτεινότητα χαμηλότερη αυτής της τιμής θεωρείται ίσο με το 0 ενώ οποιοδήποτε έχει μεγαλύτερη τίθεται ίσο με το 255 που είναι στην περίπτωση μας το μέγιστο. Το κατώφλι που επιλέχθηκε ήταν το 120 στην κλίμακα του γκρι δηλαδή στο σύνολο [0…255]. Το αποτέλεσμα είναι μία νέα δυαδική εικόνα.

Σε αυτή τη νέα εικόνα χρειάζεται να ανιχνεύσουμε όλα τα αντικείμενα των οποίων ο λόγος του μήκους προς το πλάτος είναι μεγαλύτερος του 2. Θεωρητικά αυτό μπορεί να γίνει με χρήση run codes. Αυτοί είναι αλγόριθμοι

που διατρέχουν την περιφέρεια ενός αντικειμένου που ανακαλύπτουν και μετρούν σε εικονοστοιχεία το μήκος των πλευρών. Η υλοποίηση που χρησιμοποιήσαμε ήταν ένα έτοιμο εργαλείο που προσέφερε το Labview, που λειτουργεί τελικώς με χρήση run codes τροποποιημένων για μέγιστη απόδοση. Με αυτό τον τρόπο μειώνουμε αισθητά τα πιθανά αντικείμενα που πρέπει να ελέγξουμε.

Το επόμενο βήμα είναι η διαστολή της προηγούμενης εικόνας. Διαστολή είναι μια διαδικασία που ορίζεται με τον εξής μαθηματικό τύπο. [2]

$$\mathrm{A} \oplus \mathrm{B} = \{c \mid c = a + b, a \in A, b \in B\}$$

Στην εφαρμογή μας ο πίνακας Β είναι ένα 3x3 στοιχείο γεμάτο με μονάδες. Το αποτέλεσμα της πράξης είναι ότι τα διάφορα αντικείμενα στην εικόνα, τα οποία μπορεί να περιέχουν μικρές οπές, θα απομείνουν ως επί το πλείστον συμπαγή. Η μέθοδος έχει την επίδραση να γεμίζει τις μικρές οπές σε ένα αντικείμενο. Αυτό είναι σημαντικό για το επόμενο βήμα.

Τέλος για τον εντοπισμό της πινακίδας κυκλοφορίας χρησιμοποιούμε τον αριθμό Euler. Αυτός ο αριθμός είναι ίσος με τον αριθμό των κυρτοτήτων μείον τον αριθμό των κοιλοτήτων σε ένα αντικείμενο. Θεωρώντας αυτό τον αριθμό μικρότερο του μείον τρία (-3) είμαστε σε θέση να ανιχνεύσουμε πινακίδες κυκλοφορίας που να περιέχουν τουλάχιστον τρείς (3) χαρακτήρες.[2]

Ο αριθμός Euler μπορεί να βρεθεί σαρώνοντας την εικόνα για τους παρακάτω πίνακες

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Κυρτότητες

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

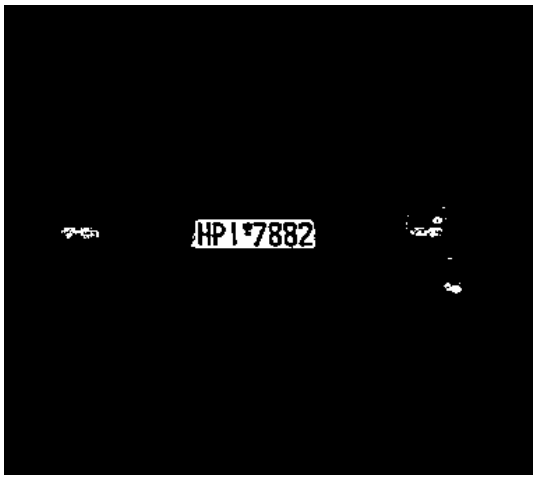
Κοιλότητες

Με το πέρας όλων αυτών των διαδικασιών έχουν απομείνει στην τελική εικόνα μόνο η πινακίδα κυκλοφορίας και αρκεί να υπολογίσουμε τα άκρα της για να βρούμε επακριβώς την σχετική της θέση στην εικόνα.

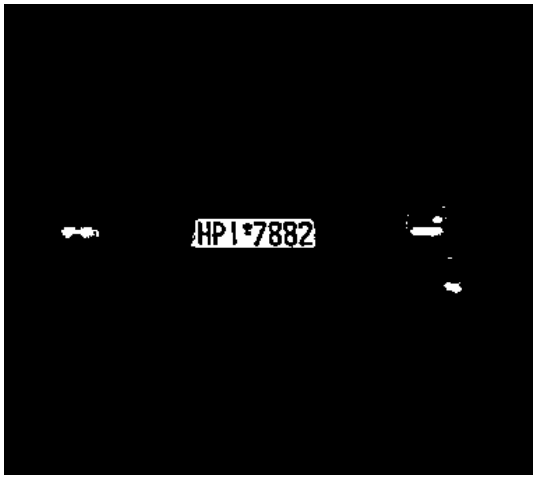
Να δηλώσουμε εδώ ότι ο αλγόριθμος δεν είναι δυνατόν να δώσει αποτέλεσμα στις περιπτώσεις όπου δεν είναι ευδιάκριτη η πινακίδα κυκλοφορίας του οχήματος, παραδείγματος χάριν λόγω αντανακλάσεων.
Παρακάτω παραθέτουμε μερικά παραδείγματα

Παράδειγμα 1^{ο}

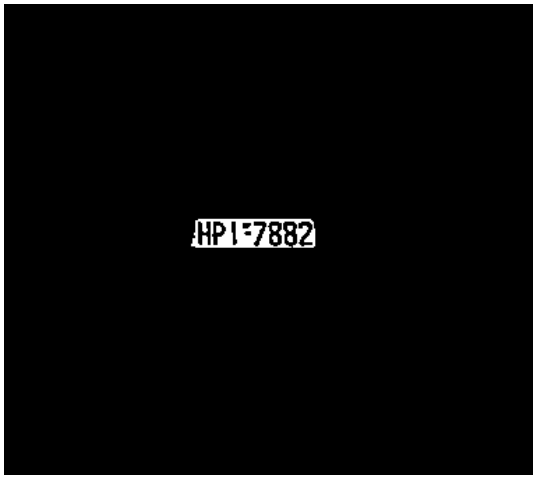
Αρχική Εικόνα



Δυαδική Μάσκα



Εικόνα κατόπιν διαστολής



Εικόνα κατόπιν Euler

Παράδειγμα 2^{ο}

Αρχική Εικόνα

Δυαδική Μάσκα

Εικόνα κατόπιν διαστολής

Εικόνα κατόπιν Euler

Στα παραπάνω δεν έχουμε επιδείξει το βήμα όπου γίνεται το λογικό ΚΑΙ της αρχικής εικόνας με τη μάσκα αλλά πάμε κατευθείαν στο στάδιο της διαστολής.

Παρατηρούμε ότι τελικώς ο αλγόριθμος δίνει ικανοποιητικά αποτελέσματα.

Παράθεση Υλοποίησης

Παρακάτω παρατίθεται ο κώδικας που αναπτύχθηκε για την υλοποίηση της μεθόδου των κυλιόμενων παραθύρων. Η αντιμετώπιση ήταν αντικειμενοστραφής και γι' αυτό το λόγο ορίστηκε μία κλάση SlidingWindows που περιείχε τις κατάλληλες συναρτήσεις για το σκοπό μας.

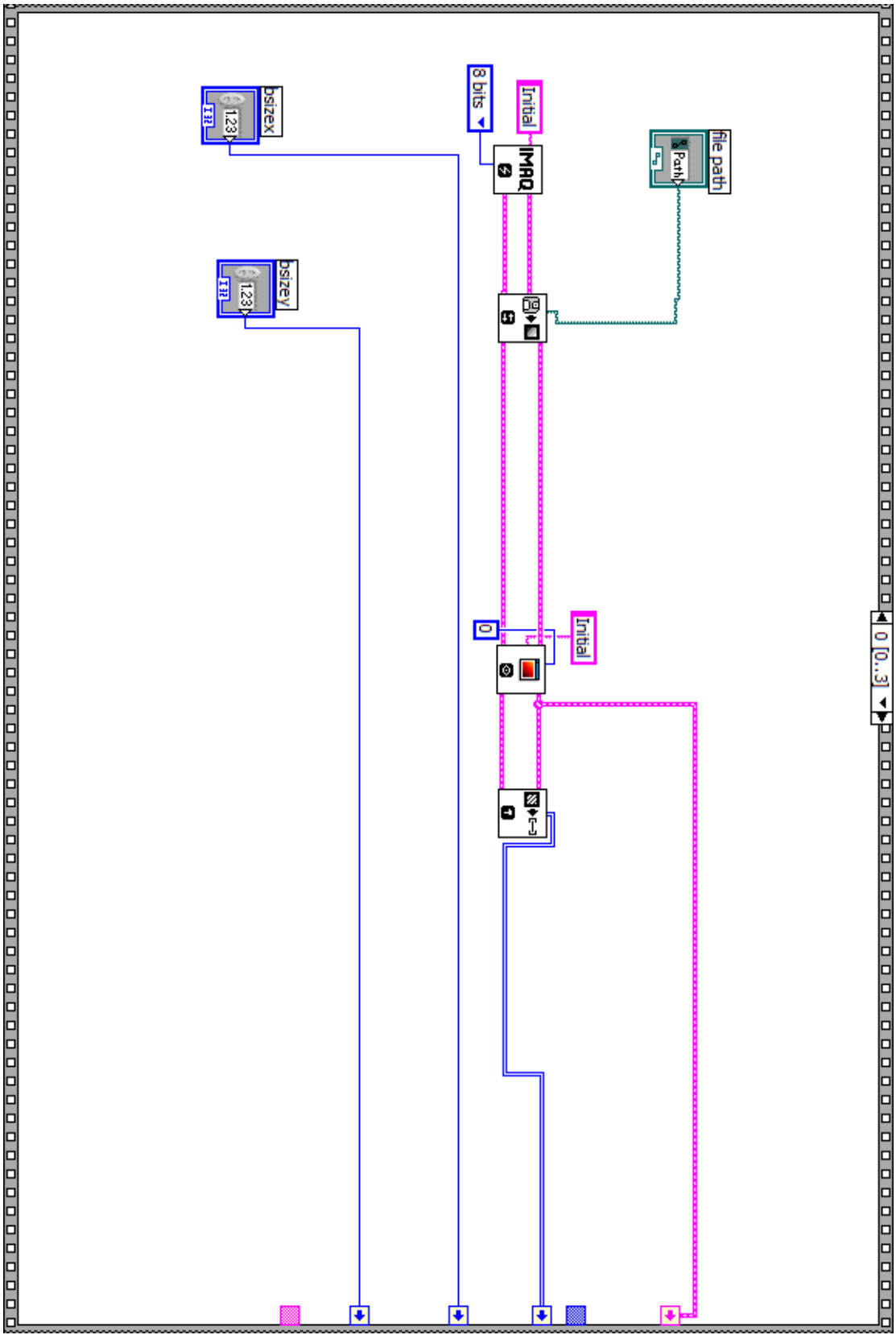
```
// SlidingWindows class declarations

#pragma once

using namespace System;

namespace UnderVehicleSurv
{
	public __gc class SlidingWindows
	{
	public:
		static int SLW(Byte image[,],double threshold,int bsizex,int
bsizey,Byte imageout[,]);
		static double std(Byte image[,]);
		static void dilation(Byte image[,]);
		static int euler(Byte image[,]);
	};
}



// Sliding Windows class implemenations

#include "stdafx.h"

#include "SlidingWindows.h"


double UnderVehicleSurv::SlidingWindows::std(Byte image[,])
{
	int dimx=image->GetLength(0),dimy=image->GetLength(1);
	register int i,j;
	double register sum;
	double meanval;
	double std;

	sum=0;
```

```
        for (i=0;i<dimx;i++)
                for (j=0;j<dimy;j++) {
                        //Calculate mean value
                        sum +=image[i,j];
                }
        meanval = sum/(dimx*dimy);

        sum=0;
        for (i=0;i<dimx;i++)
                for (j=0;j<dimy;j++) {
                        //Calculate standard deviation
                        sum +=(image[i,j]-meanval)*(image[i,j]-meanval);
                }
        std = sum/(dimx*dimy);
        std=System::Math::Sqrt(std);
        return std;
}

int UnderVehicleSurv::SlidingWindows::SLW(Byte image[,],double
threshold,int bsizex,int bsizey,Byte imageout[,])
{
        /*Calculating some really common variables*/
        int divider = bsizex * bsizey *4; //It is (bsizex*2)(bsizey*2)
        int divider2 = bsizex * bsizey; // It is (bsizex*2/2)(bsizey*2/2)
        int bsizex2 = bsizex/2, bsizey2=bsizey/2;
        int i,j;
        int dimx=image->GetLength(0),dimy=image->GetLength(1); //
Finding image's dimensions

        for (i=0+bsizex;i<dimx-bsizex;i++) // For each pixel in picture
                for (j=0+bsizey;j<dimy-bsizey;j++) {
                        double meanval1=0;
                        double std1=0;
                        int k,l;
                        for (k=-bsizex;k<bsizex;k++) //For a sliding window find
mean value
                                for (l=-bsizey;l<bsizey;l++)
                                        meanval1 = meanval1 + image[i+k,j+l];
                        meanval1 = meanval1/divider;
                        std1 = std1 + (image[i+k,j+l]-
meanval1)*(image[i+k,j+l]-meanval1); // And deviation
                        std1 /= divider;
                        std1 = System::Math::Sqrt(std1);

                        double meanval2=0;
                        double std2=0;
                        int g,h;
```

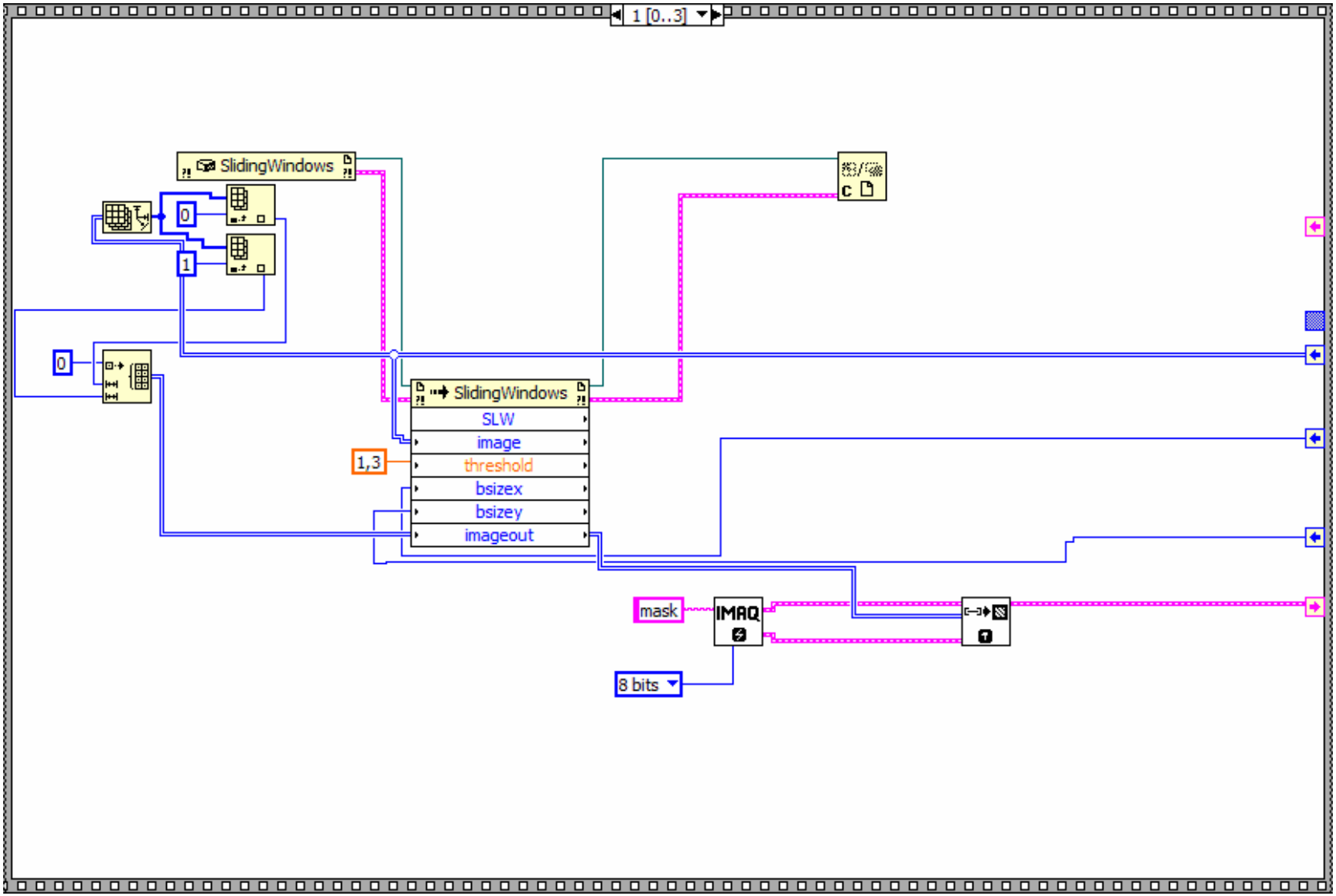
```
                        for (g=-bsizex2;g<bsizex2;g++) // Do the same for a
smaller sliding window(each dimension smaller by half)
                                for(h=-bsizey2;h<bsizey2;h++)
                                        meanval2 = meanval2 + image[i+g,j+h];
                        meanval2= meanval2/divider2;
                        std2 = std2 + (image[i+g,j+h]-
meanval2)*(image[i+g,j+h]-meanval2);
                        std2 /= divider2;
                        std2 = System::Math::Sqrt(std2);
                        if ( System::Math::Abs(std2-std1) >= threshold) // If
threshold exceeded
                                imageout[i,j]=255; //pixel marked
                        }
        return 0;
}

void UnderVehicleSurv::SlidingWindows::dilation(Byte image[,])
{
        int dimx=image->GetLength(0),dimy=image->GetLength(1); //
Finding image's dimensions

        int i,j;

        /* i=i+2,j=j+2 because we have a 3x3 matrix full of logical ones*/
        for (i=0;i<dimx;i=i+2)
                for (j=0;j<dimy;j=j+2) {
                        if (image[i,j]==255)
                                image[i-1,j-1]=255;
                                image[i-1,j]=255;
                                image[i-1,j+1]=255;
                                image[i,j-1]=255;
                                image[i,j+1]=255;
                                image[i+1,j-1]=255;
                                image[i+1,j]=255;
                                image[i+1,j+1]=255;
                }
                return;
}

int UnderVehicleSurv::SlidingWindows::euler(Byte image[,])
{
        int dimx=image->GetLength(0),dimy=image->GetLength(1); //
Finding image's dimensions
        int counter;

        int i,j;

        for (i=0;i<dimx;i++)
```

```
                for (j=0;j<dimy;j++) {
                        if (image[i,j]==0 && image[i,j+1]==0 &&
image[i+1,j]==0 && image[i+1,j+1]==1)
                                counter++;
                        if (image[i,j]==0 && image[i,j+1]==1 &&
image[i+1,j]==1 && image[i+1,j+1]==1)
                                counter--;
                }
        return counter;
}
```

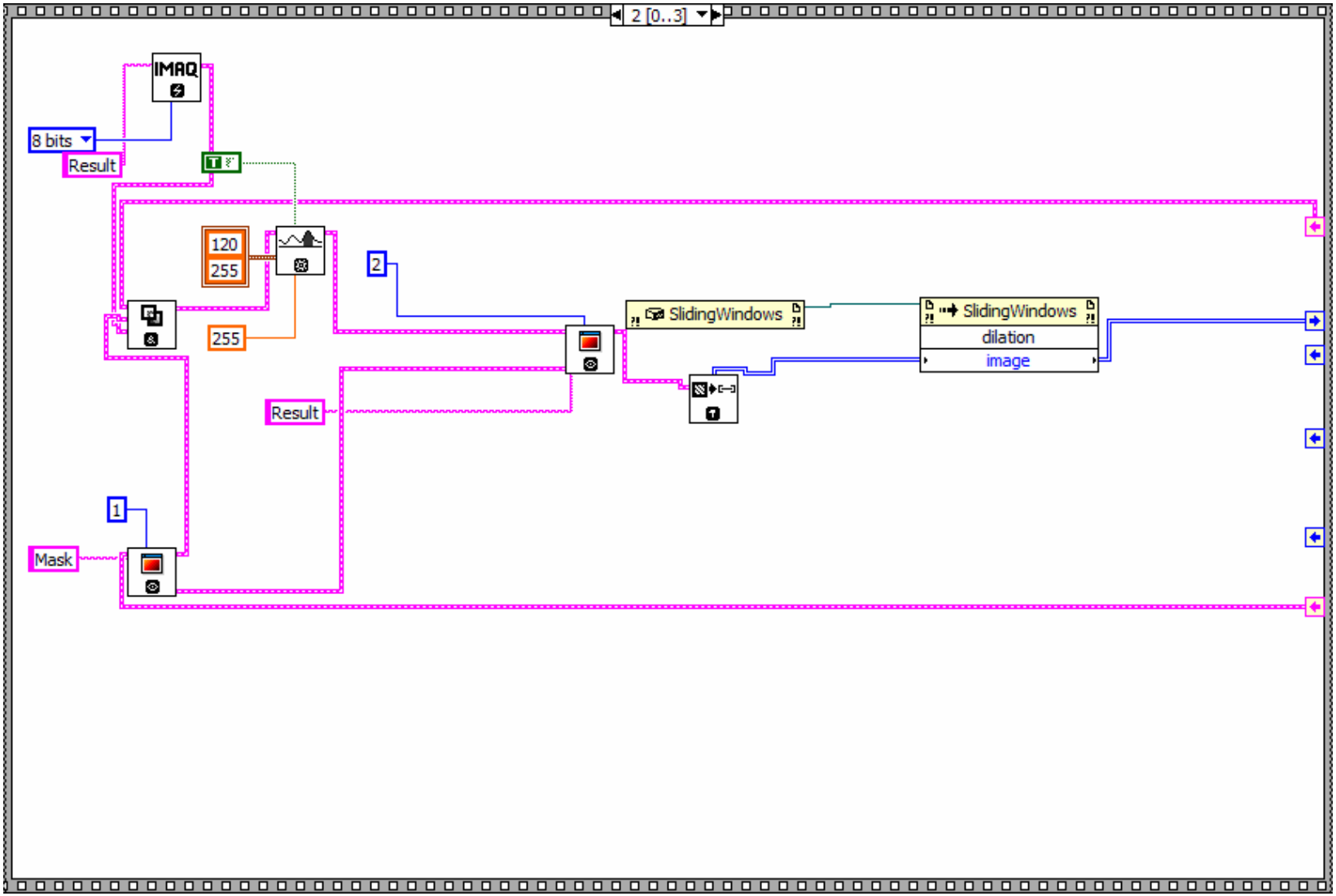
Παρακάτω παραθέτουμε τον σχηματικό κώδικα όπως υλοποιήθηκε στο πρόγραμμα Labview της National Instruments



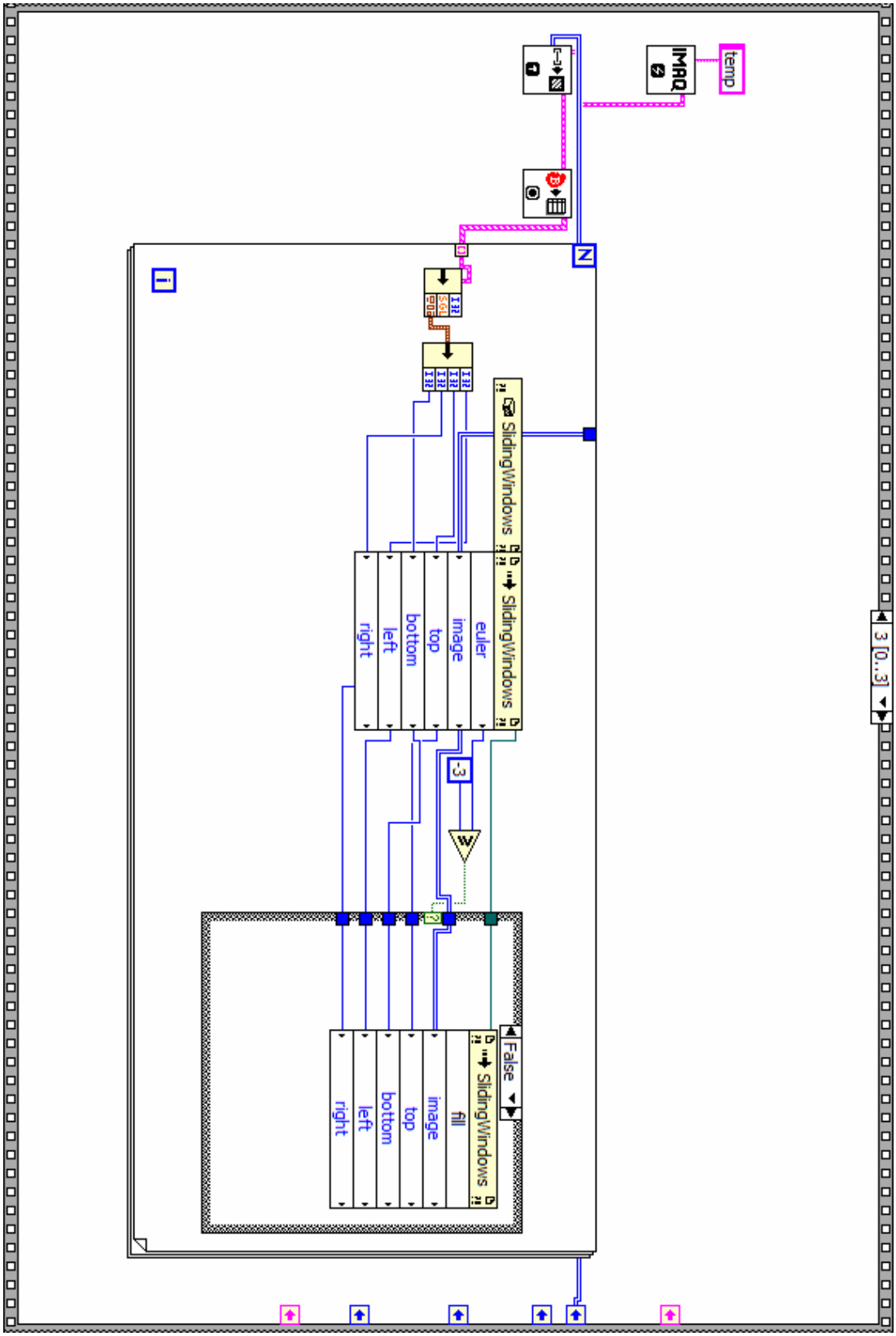
Πλαίσιο 1 του σχηματικού κώδικα Labview



Πλαίσιο 2 του σχηματικού κώδικα Labview



Πλαίσιο 3 του σχηματικού κώδικα Labview



Πλαίσιο 4 του σχηματικού κώδικα Labview

ΜΕΡΟΣ ΔΕΥΤΕΡΟ

ΤΕΧΝΙΚΗ ΤΑΥΤΟΠΟΙΗΣΗΣ ΕΙΚΟΝΩΝ ΜΕ ΧΡΗΣΗ WAVELETS

WAVELET BASED IMAGE REGISTRATION

Μέρος Δεύτερο

Εισαγωγή

Σε αυτό το μέρος της διπλωματικής εργασίας εξετάζουμε μία μέθοδο ταυτοποίησης εικόνων. Η ταυτοποίηση εικόνων στην περίπτωση μας χρειάζεται γιατι, όπως είναι εύκολο κανείς να φανταστεί, η πιθανότητα να περάσει ένα όχημα μπροστά από την κάμερα, με την ακριβώς ίδια γωνία και ταχύτητα, που πέρασε κατά τον αρχικό έλεγχο, είναι πολύ μικρή. Έτσι η σάρωση κάθε φορά από την κάμερα θα είναι διαφορετική και η τελική εικόνα, που θα πρέπει να συγκρίνουμε με την αρχική, θα διαφέρει τουλάχιστον κατά θέση και γωνία, χωρίς να λαμβάνουμε υπόψη μας πιθανές αλλαγές περιεχομένου.

Περιγραφή Αλγορίθμου

Σε ό,τι αφορά την ταυτοποίηση εικόνων υπάρχει μία πληθώρα μεθόδων. Αναφέρουμε την οικογένεια των στατιστικών μεθόδων που έχει ως κύριο σκοπό την μεγιστοποίηση στατιστικών μεγεθών μεταξύ των υπό σύγκριση εικόνων. Σημαντικός εκπρόσωπος αυτής της οικογένειας είναι η μετρική της αμοιβαίας πληροφορίας (Mutual Information) που έχει ως σκοπό τη μεγιστοποίηση του συγκεκριμένου μεγέθους που συνεπάγεται ελαχιστοποίηση της εντροπίας. Άλλη σημαντική οικογένεια μεθόδων είναι αυτή της μείωσης του σφάλματος σύγκρισης μεταξύ των εικόνων.

Η επιλογή γι' αυτήν τη διπλωματική εργασία ήταν η μέθοδος της κανονικοποιημένης ετεροσυσχέτισης.

Ο τύπος της κανονικοποιημένης ετεροσυσχέτισης δίνεται από το:

$$C(u,v) = \frac{\sum_x \sum_y T(x,y) I(x-u, y-v)}{\left[\sum_x \sum_y I^2(x-u, y-v)\right]^{\frac{1}{2}}}$$

Όπου T η εικόνα που θέλουμε να ταυτοποιήσουμε και I η αρχική εικόνα. Τα u,v ορίζουν το στοιχείο στην εικόνα I ως προς το οποίο υπολογίζουμε την ετεροσυσχέτιση.[3]

Θεωρητική Ανάλυση της Κανονικοποιημένης Ετεροσυσχέτισης

Μια σημαντική ερώτηση είναι γιατί η μέθοδος της κανονικοποιημένης ετεροσυσχέτισης χρησιμοποιείται τόσο αποτελεσματικά για την ταυτοποίηση δύο σημάτων. Για να απαντηθεί το ερώτημα παραθέτουμε τα παρακάτω.[4]

Θεωρείστε δύο σήματα f(n) , g(n) όπου το g είναι το σήμα το οποίο θέλουμε να βρούμε μέσα στο f. Μια τυπική προσέγγιση του θέματος είναι να βρούμε μία σταθερά κ η οποία να ελαχιστοποιεί το μέσο τετραγωνικό σφάλμα.

$$E_2(k) = \sum_{n \in W} \left[f(n+k) - g(n) \right]^2$$

Όπου το W είναι ένα υποσύνολο του πεδίου ορισμού των f,g. Είναι γνωστό ότι

$$(a-b)^2 = a^2 + b^2 - 2ab$$

Χρησιμοποιώντας αυτόν τον τύπο στην παραπάνω εξίσωση γίνεται εμφανές ότι η ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος είναι ταυτόσημη έννοια με την μεγιστοποίηση του όρου

$$L_{fg}(k) = \sum_{n \in W} f(n+k)g(n)$$

Ο οποίος είναι η γραμμική ετεροσυσχέτιση. Για τις δύο διαστάσεις και ο τύπος μετατρέπεται σε :

$$L_{fg}(k,l) = \sum_{x \in W} \sum_{y \in W} f(x+k, y+l)g(x,y)$$

Που είναι ο αριθμητής του τύπου της κανονικοποιημένης ετεροσυσχέτισης που παραθέσαμε πιο πριν. Ο παρονομαστής παρέχει την κανονικοποίηση στην εξίσωση, ώστε τυχόν τοπικά μέγιστα να μην επηρεάζουν το μέγεθος.[3]. Εάν η εικόνα Τ ταιριάζει απόλυτα με την εικόνα Ι σε ένα σημείο (x,y) τότε η κανονικοποιημένη ετεροσυσχέτιση σε αυτό το σημείο θα είναι μέγιστη. Έτσι ελέγχοντας ένα πλήθος από διαφορετικές ετεροσυσχετίσεις και λαμβάνοντας τη μέγιστη μπορούμε να βρούμε που ακριβώς οι δύο εικόνες ταυτοποιούνται. Για την πληρότητα της ανάλυσης αναφέρουμε ότι υπάρχουν και άλλα κριτήρια ταυτοποίησης σημάτων όπως το μέσο απόλυτο σφάλμα

$$E_1(k) = \sum_{n \in W} \left| f(n+k) - g(n) \right|$$

Το οποίο με αντίστοιχη λογική οδηγεί στη χρήση της μη-γραμμικής ετεροσυσχέτισης, η οποία όμως δεν χρησιμοποιείται στη συγκεκριμένη εργασία.

$$M_{fg}(k) = \sum_{n \in W} \min\left[f(n+k), g(n) \right]$$

Από τον τύπο της κανονικοποιημένης ετεροσυσχέτισης γίνεται προφανές ότι, για μεγάλες διαστάσεις εικόνας, ο υπολογισμός ετεροσυσχέτισης για κάθε σημείο (x,y) μπορεί να αποβεί εξαιρετικά χρονοβόρος. Σε αυτό το σημείο δεδομένου ότι η εκτέλεση ενός τέτοιου

αλγορίθμου ήταν απαγορευτική για το μέγεθος των εικόνων που μελετάμε καταφύγαμε σε κάποια υπολογιστικά τεχνάσματα.

Συγκεκριμένα αποφασίστηκε να χρησιμοποιηθεί η θεωρία των wavelets, ώστε να καταφύγουμε σε μία μείωση των διαστάσεων κάθε εικόνας.

Θεωρητική Ανάλυση των Wavelets

Τα wavelets είναι συναρτήσεις που ικανοποιούν συγκεκριμένα μαθηματικά κριτήρια και χρησιμοποιούνται για την αναπαράσταση δεδομένων ή και άλλων συναρτήσεων. Η ιδέα δεν είναι καινούρια. Η προσέγγιση συναρτήσεων βάσει άλλων υπήρξε από τις αρχές του 19^{ου} αιώνα, όταν ο Joseph Fourier ανακάλυψε ότι μπορούσε να τοποθετήσει συναρτήσεις ημιτόνων και συνημίτονων και να αναπαραστήσει άλλες συναρτήσεις. Παρόλα αυτά στην ανάλυση wavelet η κλίμακα που χρησιμοποιούμε για να εξετάζουμε τα δεδομένα έχει εξέχοντα ρόλο. Οι αλγόριθμοι wavelet επεξεργάζονται δεδομένα σε διαφορετικές αναλύσεις ή κλίμακες. Αυτό δίνει μεγάλη χρησιμότητα στα wavelets.[5]

Στην ανάλυση wavelet χρησιμοποιούμε μία δοσμένη συνάρτηση που ονομάζεται αναλυτικό (analyzing) ή μητρικό (mother) wavelet. Χρησιμοποιώντας μία κατάλληλη μορφή αυτού του αρχικού wavelet αναλύουμε τα δεδομένα μας είτε στο πεδίο του χρόνου είτε στο πεδίο της συχνότητας. Δεδομένου ότι τα αρχικά μας δεδομένα, σήμα ή συνάρτηση, μπορεί να αναπαρασταθεί χρησιμοποιώντας ένα γραμμικό συνδυασμό των συντελεστών του wavelet είναι δυνατόν να κάνουμε οποιαδήποτε πράξη πάνω στα δεδομένα χρησιμοποιώντας αυτούς τους συντελεστές.[5]

Οι μετασχηματισμοί wavelets αποσυνθέτουν ένα σήμα σε δύο υποσήματα του μισού μήκους. Ο μετασχηματισμός Haar, ο αρχικός όλων των μετασχηματισμών wavelets, λειτουργεί ως εξής. Το ένα υπόσημα είναι ένας

τρέχον μέσος όρος και το άλλο μία τρέχουσα διαφορά. Εξετάζουμε λίγο το πρώτο σήμα

Εστω $a^1=(a_1,a_2,...,a_{N/2})$ το πρώτο από τα δύο υποσήματα, όπου N το μήκος του αρχικού σήματος. Οι τιμές του υπολογίζονται παίρνοντας το μέσο όρο κάθε ζεύγους τιμών και πολλαπλασιάζοντας τον με $\sqrt{2}$. Ισχύει δηλαδή:

$$a_m = \frac{f_{2m-1} + f_{2m}}{\sqrt{2}}$$

Για το υποσήμα που φέρει τις τρέχουσες διαφορές ακολουθείται ο τύπος

$$d_m = \frac{f_{2m-1} - f_{2m}}{\sqrt{2}}$$

Η δυνατότητα να επιστρέψουμε στο αρχικό σήμα υπάρχει μέσω του ακόλουθου μετασχηματισμού:

$$f = (\frac{a_1 + d_1}{\sqrt{2}}, \frac{a_1 - d_1}{\sqrt{2}}, ..., \frac{a_{N/2} + d_{N/2}}{\sqrt{2}}, \frac{a_{N/2} - d_{N/2}}{\sqrt{2}})$$

Μία από τις πιο σημαντικές ιδιότητες του μετασχηματισμού Haar είναι ότι η τάξη μεγέθους των τιμών του υποσήματος διαφορών σε σχέση με αυτή του αρχικού σήματος είναι σημαντικά μικρότερη. Επίσης, εφόσον το αρχικό σήμα παρουσιάζει μία στοιχειώδη συνέχεια, οι τιμές του υποσήματος διαφορών τείνουν να βρίσκονται κοντά στο μηδέν(0). Το εκπληκτικό είναι ότι το υπόσημα μέσων παρουσιάζει γραφικά ομοιότητα με το αρχικό σήμα. Συνεπαγωγή των παραπάνω είναι ότι μπορούμε να χρησιμοποιήσουμε τα παραπάνω, ώστε να επιτύχουμε συμπίεση ενός σήματος.

Μια δεύτερη σημαντική ιδιότητα είναι ότι ο Haar μετασχηματισμός επιτυγχάνει να διατηρεί την ενέργεια των σημάτων.

Εστω η ενέργεια του αρχικού σήματος

$$\varepsilon_f = f_1^2 + f_2^2 + \ldots + f_N^2$$

Και η ενέργεια του μετασχηματισμένου κατά Haar σήματος

$$\varepsilon_{(a^1|d^1)} = \sum (a_m^2 + d_m^2)$$

Ισχύει:

$$a_m^2 + d_m^2 = \left[\frac{f_{2m-1} + f_{2m}}{\sqrt{2}}\right]^2 + \left[\frac{f_{2m-1} + f_{2m}}{\sqrt{2}}\right]^2 = f_{2m-1}^2 + f_{2m}^2$$

Αρα τελικά:

$$\varepsilon_{(a^1|d^1)} = \varepsilon_f$$

Όμως ακόμη πιο σημαντικό είναι ότι το μεγαλύτερο μέγεθος της ενέργειας τοποθετείται στο πρώτο υποσήμα ενώ μικρό μόνο ποσοστό στο δεύτερο. Αυτό δικαιολογείται από την προηγούμενη ιδιότητα, αφού τελικά οι τιμές του υποσήματος διαφορών θα του αποδώσουν πολύ μικρή ενέργεια συγκριτικά με το υποσήματα μέσων.[6]

Σημαντικό είναι το γεγονός ότι, εάν επιλέξουμε προσεκτικά το μητρικό wavelet και αποκόψουμε το υπόσημα διαφορών, μπορούμε να επιτύχουμε εξαιρετική συμπίεση δεδομένου ότι η μεγαλύτερη ενέργεια των δεδομένων ή του σήματος εμπεριέχεται σε μερικούς μόνο από τους συντελεστές του γραμμικού συνδυασμού.

Η πρώτη αναφορά στα wavelets στην ιστορία έγινε από τον A.Haar το 1909 που όρισε και την πρώτη οικογένεια τέτοιων συναρτήσεων. Δυστυχώς αυτή η οικογένεια wavelets δεν διαφοροποιείται συνεχώς, με αποτέλεσμα να είναι σχετικά μειωμένες οι εφαρμογές τους. Ο Haar όμως έδωσε το εναρκτήριο

λάκτισμα για μία συνεχόμενη μελέτη πάνω στο θέμα. Σύντομα υπήρχαν και άλλες οικογένειες από wavelets όπως των Meyer, Daubechie, Coifman αλλά και άλλων. Οι εφαρμογές τους ποικίλλουν ανάλογα την οικογένεια και το βαθμό πολυπλοκότητάς τους, αλλά χρησιμοποιούνται ήδη σε πολλούς τομείς όπως υπολογιστική όραση, συμπίεση, αφαίρεση θορύβου, ενώ καθημερινά εφευρίσκονται και καινούριες.

Σε αυτή την διπλωματική εργασία τα wavelets χρησιμοποιούνται με σκοπό την συμπίεση των δεδομένων που περιέχει μία εικόνα σε χαμηλότερες αναλύσεις, ώστε να μειωθεί ο χρόνος επεξεργασίας. Για να το επιτύχουμε αυτό χρησιμοποιούμε τον αλγόριθμο του Mallat. Αυτός αποτελείται από μία σειρά από συνδυασμούς βαθυπερατών και υψιπερατών φίλτρων κατά τις κάθετες και οριζόντιες διευθύνσεις. Η μέθοδος επιστρέφει τέσσερις(4) υποεικόνες στη μισή ανάλυση της αρχικής εικόνας. Αυτές αντιστοιχούν στις LL (βαθυπερατό - βαθυπερατό), HL (υψιπερατό - βαθυπερατό), LH (βαθυπερατό - υψιπερατό), HH (υψιπερατό -υψιπερατό). Η LL υποεικόνα έχει υποβληθεί σε 2 βαθυπερατά φίλτρα κατά τον οριζόντιο και κάθετο άξονα και περιέχει μία προσέγγιση της αρχικής εικόνας στη μισή ανάλυση, ενώ οι άλλες περιέχουν πληροφορίες για τις ακμές της αρχικής εικόνας ανάλογα το όνομα τους. Έτσι η LH έχει πληροφορίες για τις ακμές κατά την οριζόντια διεύθυνση, η HL για την κάθετη και η HH για την διαγώνια. Παρακάτω δείχνουμε σχηματικά τον αλγόριθμο του Mallat.[7]

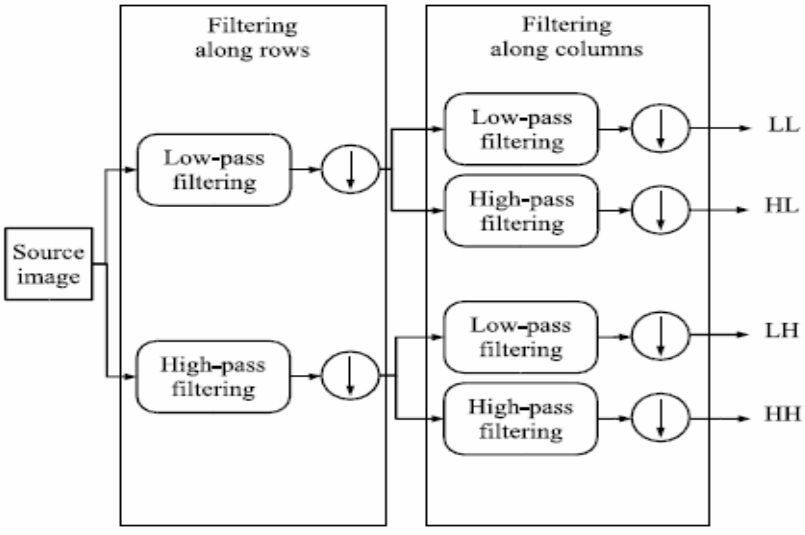


Fig. 1. The Mallat decomposition diagram

Είναι προφανές ότι η εικόνα LL,δεδομένου ότι είναι μία πολύ καλή προσέγγιση της αρχικής μας σε μικρότερη ανάλυση είναι ιδανική γι' αυτό που θέλουμε να επιτύχουμε. Έτσι επαναλαμβάνουμε τη διαδικασία του Mallat τέσσερις συνολικά φορές, με είσοδο κάθε φορά τη νέα εικόνα LL, που προέκυψε από το προηγούμενο βήμα. Το αποτέλεσμα είναι μία εικόνα 16 φορές μικρότερης ανάλυσης, η οποία όμως διατηρεί μεγάλο μέρος της ενέργειας της αρχικής.[7]

Τελικός σκοπός της προσέγγισης αυτής είναι να χρησιμοποιήσουμε έναν αλγόριθμο ταυτοποίησης εικόνας αρχικά σε αυτήν την επιπέδου τέσσερα εικόνα και λαμβάνοντας τα αποτελέσματα γι' αυτήν, να ανεβαίνουμε σταδιακά επίπεδα, βελτιώνοντας με την ίδια μέθοδο ταυτοποίησης κάθε φορά τα αποτελέσματα. Το κέρδος έγκειται στο ότι, ενώ στην αρχική εικόνα θα έχουμε ένα μεγάλο εύρος, στο οποίο θα πρέπει να ανιχνεύσουμε για την ταυτόσημη εικόνα, στα επόμενα βήματα αρκεί να περιοριστούμε σε στενές περιοχές γύρω από τα αποτελέσματα, που μας παρέχει το προηγούμενο. Στο τελικό βήμα θα αρκεί να κάνουμε μία μικρή σε εύρος ταυτοποίηση γεγονός που μας οδηγεί σε τεράστιο κέρδος απόδοσης.[8]

Χωρική Ταυτοποίηση

Σε ό,τι αφορά τον αλγόριθμο ταυτοποίησης, με βάση όσα αναφέραμε πιο πριν, ακολουθήθηκε η εξής λογική. Για κάθε ζεύγος υποεικόνων, που εξετάζαμε, λαμβάναμε την τρέχουσα υποεικόνα και την υποβάλαμε σε ένα ζεύγος μετασχηματισμών κατά γωνία και κατά θέση. Συγκεκριμένα είχαμε:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

Όπου θ η γωνιακή μετατόπιση που δοκιμάζαμε σε κάθε βήμα και x_0, y_0 η μετατόπιση. Για κάθε τέτοιο συνδυασμό υπολογίζαμε την κανονικοποιημένη ετεροσυσχέτιση με βάση τον τύπο, που παρουσιάσαμε παραπάνω για θέση τέτοια ώστε u=v=0. Σε ό,τι αφορά την περιστροφή χρησιμοποιήσαμε την τεχνική της διγραμμικής παρεμβολής(bilinear interpolation), ώστε να επιτύχουμε καλύτερη απόδοση των τιμών των στοιχείων εικόνας(pixel) και τόσο σε αυτήν όσο και στην μετακίνηση, όπου χρειαζόταν, χρησιμοποιήθηκε το γέμισμα πινάκων με μηδενικά(zero padding).

Σε κάθε στάδιο, η μέθοδος επιστρέφει τρεις τιμές. Τη γωνία, τη μετακίνηση κατά τον οριζόντιο άξονα και κατά τον κάθετο, ώστε οι δύο εικόνες να ταυτίζονται κατά το μέγιστο δυνατό. Στο τελικό στάδιο η τρέχουσα εικόνα, που έχει ληφθεί από την κάμερα, περιστρέφεται και μετακινείται κατά τα απαιτούμενα και κατόπιν υπολογίζεται η απόλυτος διαφορά της με την αρχική. Η απόλυτος διαφορά υπολογίζεται αφαιρώντας στοιχείο προς στοιχείο τις δύο εικόνες και λαμβάνοντας το απόλυτο κάθε αφαίρεσης. Το αποτέλεσμα είναι εικόνες παρόμοιες σαν αυτές που παρατίθενται παρακάτω και όπως θα δούμε χρησιμοποιούνται αυτούσιες στο τρίτο κομμάτι αυτής της διπλωματικής.

Σε ό,τι αφορά τα εύρη αναζήτησης, που χρησιμοποιήσαμε, έχουμε τον παρακάτω πίνακα

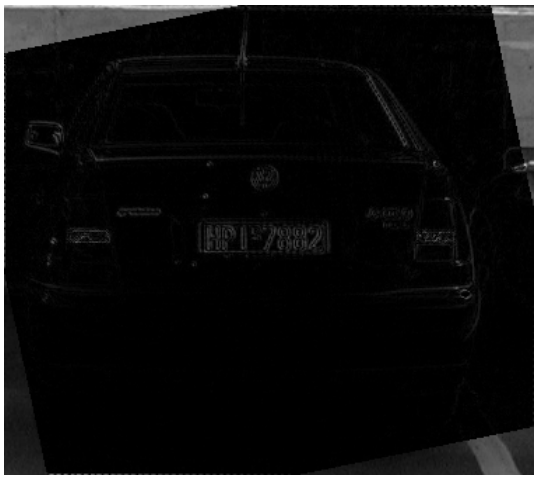
Βαθμός Ανάλυσης	Μέγιστη Γωνία	Βήμα Γωνίας	Μέγιστη μετατόπιση x,y
4	$\pm 16^\circ$	1°	$\pm$ διάσταση/2
3	$\pm 8^\circ$	1°	± 2
2	$\pm 4^\circ$	$0,5^\circ$	± 2
1	$\pm 2^\circ$	$0,2^\circ$	± 2
Πλήρης Ανάλυση	$\pm 1^\circ$	$0,2^\circ$	± 2

Παραθέτουμε μερικά παραδείγματα:

Παράδειγμα 1^{ο}

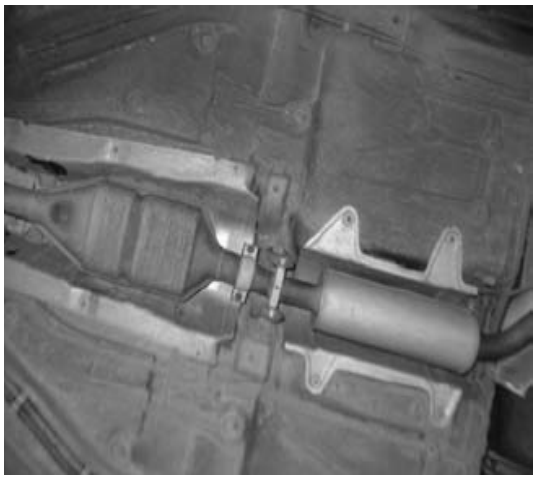
Αρχική Εικόνα

Τρέχουσα εικόνα

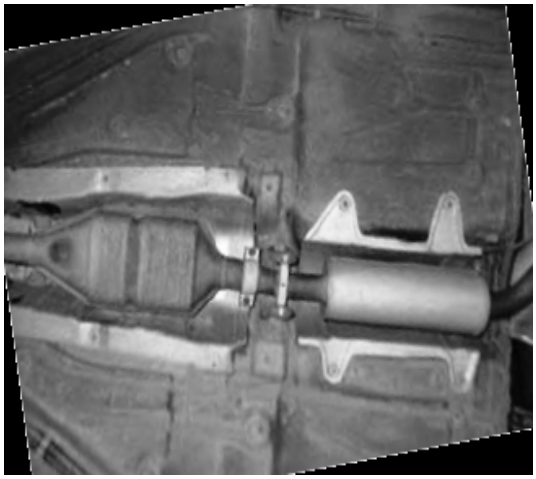


Ταυτοποίηση των δύο εικόνων

Παράδειγμα 2^{ο}



Αρχική



Τρέχουσα

Ταυτοποίηση των δύο εικόνων.

Συμπερασματικά η μέθοδος δουλεύει ικανοποιητικά και ταυτοποιεί εικόνες με επιτυχία.

Τροποποιημένη μέθοδος Κανονικοποιημένης Ετεροσυσχέτισης

Παρόλα αυτά παρατηρήθηκε ότι ενώ εργαστηριακά σε τεχνητά περιστρεμμένες και μετακινημένες εικόνες η μέθοδος είναι ικανοποιητική, στην πραγματικότητα, σε διαδοχικές εικόνες, που ελήφθησαν με κάμερες υψηλής ανάλυσης, αποτυγχάνει να παρέχει τα κατάλληλα αποτελέσματα. Το γεγονός αυτό στο γεγονός ότι υπάρχουν τοπικές διαστρεβλώσεις. Έχουν δε ως αποτέλεσμα η μείωση του σφάλματος, δηλαδή η μεγιστοποίηση της ετεροσυσχέτισης να μην εγγυάται την χωρική ταύτιση. Για την αντιμετώπιση του φαινομένου χρησιμοποιήθηκε ο παρακάτω αλγόριθμος:

Οι εικόνες χωρίστηκαν σε τετραγωνικές υποεικόνες και εκτελέστηκε για κάθε τέτοιο στοιχείο ο αλγόριθμος της Κανονικοποιημένης Ετεροσυσχέτισης. Μετά το πέρας του αλγορίθμου κάθε υποεικόνα αναφέρει για ποια γωνιακή και σχετική μετατόπιση παρουσιάζει τη μέγιστη τιμή της Ετεροσυσχέτισης και κατόπιν με ένα σύστημα ψηφοφορίας, όπου όλες οι υποεικόνες συμμετέχουν με το ίδιο βάρος, επιστρέφεται από το σύστημα η τιμή των μετατοπίσεων, που έχει την πλειοψηφία. Η πολυπλοκότητα του αλγορίθμου παραμένει η ίδια, αν και υπάρχει μία μικρή καθυστέρηση στις επιδόσεις στο τελικό στάδιο, λόγω του συστήματος ψηφοφορίας.

Όπως δείχνουν τα παρακάτω παραδείγματα ο αλγόριθμος πλέον είναι ανεκτικός τόσο στο θόρυβο όσο και στις τοπικές διαστρεβλώσεις και παρέχει ικανοποιητικά αποτελέσματα στο σύνολο των περιπτώσεων.

Παράδειγμα 1^{ο}

Αρχική Εικόνα

Τρέχουσα εικόνα

Ταυτοποίηση των δύο εικόνων

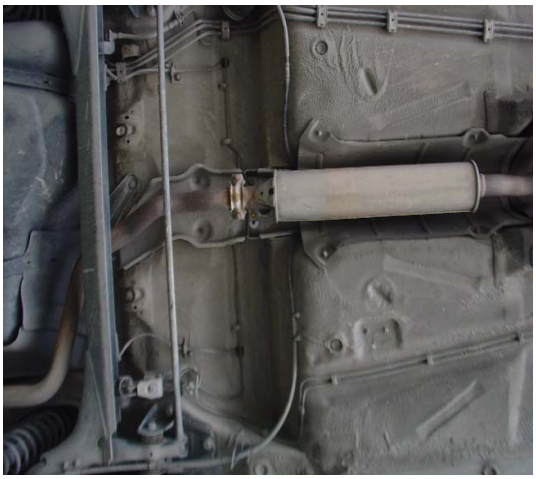
Παράδειγμα 2^{ο}

Αρχική Εικόνα

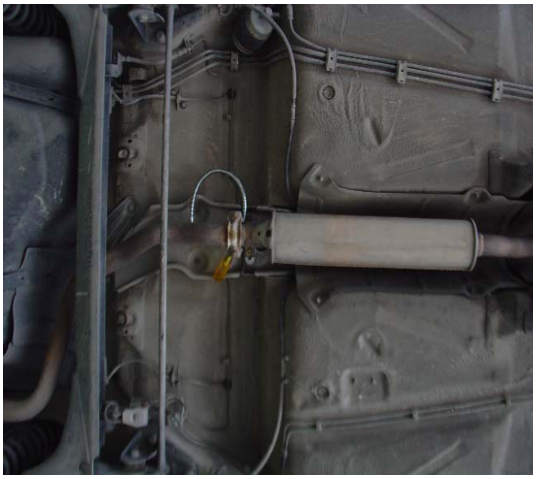
Τρέχουσα εικόνα

Ταυτοποίηση των δύο εικόνων

Παράδειγμα 3^{ο}



Αρχική Εικόνα



Τρέχουσα εικόνα

Ταυτοποίηση των δύο εικόνων

Παράθεση Υλοποίησης

Παρακάτω παραθέτουμε τον κώδικα C++ που υλοποιήθηκε σε αυτό το στάδιο καθώς και το σχηματικό κώδικα Labview. Σε ό,τι αφορά την παράθεση του σχηματικού κώδικα, η πρώτη παράθεση αναφέρεται στην υλοποίηση χωρίς το μηχανισμό ψηφοφορίας, ενώ η δεύτερη τον εμπεριέχει. Η παράθεση του κώδικα C++ περιέχει και τις δύο μορφές, μια και βασίζονται απλά σε διαφορετικές συναρτήσεις.

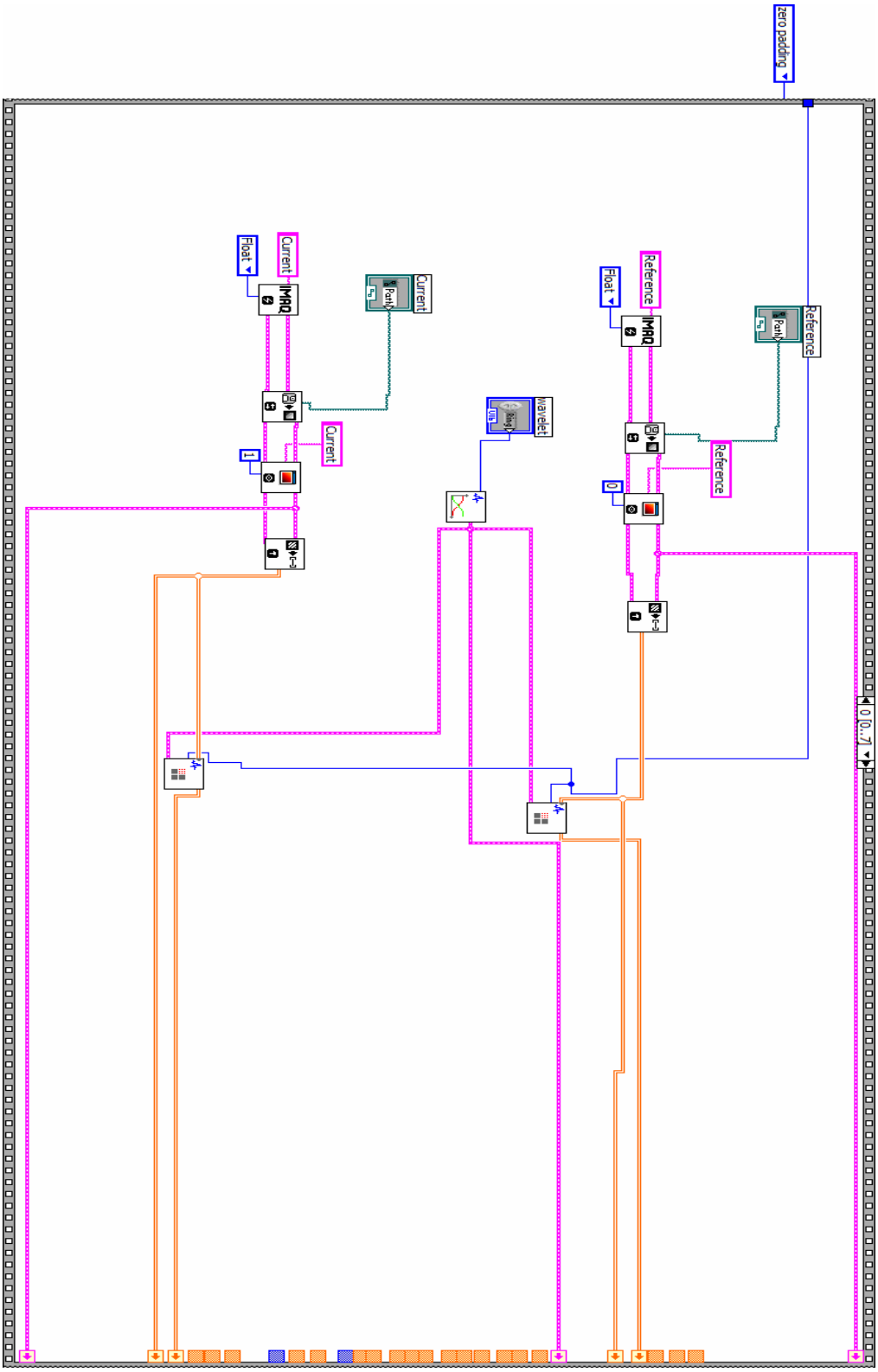
```
// Registration class header file

#pragma once

using namespace System;
using namespace System::Collections;

namespace UnderVehicleSurv
{
	public __gc class Relatplace {
	public:
		double angle;
		int metx;
		int mety;
		double nccvalue;
		Relatplace(double a,int x,int y,double nc);
		Relatplace();
		bool Equals(Object *);
	};

	public __gc class Registration
	{
	public:
		static void noisyncc(float image[,],float templ[,],int bsizex,int
bsizey,Relatplace *r,Relatplace *values[,]);
		static double ncc(float image[,],float templ[,]);
		static Relatplace* vote(Relatplace* values[,]);
	};


}
```

```
// Registration class implemantations

#include "stdafx.h"
#include "Registration.h"

using System::Math;
using System::Collections::ArrayList;

UnderVehicleSurv::Relatplace::Relatplace(double a,int x,int y,double nc)
{
	angle=a;
	metx=x;
	mety=y;
	nccvalue=nc;
}

UnderVehicleSurv::Relatplace::Relatplace()
{
	angle=0;
	metx=0;
	mety=0;
	nccvalue=0;
}

bool UnderVehicleSurv::Relatplace::Equals(Object *b)
{
	Relatplace *c=dynamic_cast<Relatplace *> (b);

	if ( c->angle==angle && c->metx==metx && c->mety==mety )
		return true;
	else
		return false;
}

void UnderVehicleSurv::Registration::noisyncc(float image[,],float templ[,],int
bsizex,int bsizey,Relatplace *r,Relatplace *values[,])
{
	int register x,y,bx,by;
	double register sum,sum2;
	int placex;
	int placey;

	int imagex=image->GetLength(0);
	int imagey=image->GetLength(1);

	int numbx=imagex/bsizex;
	int numby=imagey/bsizey;
```

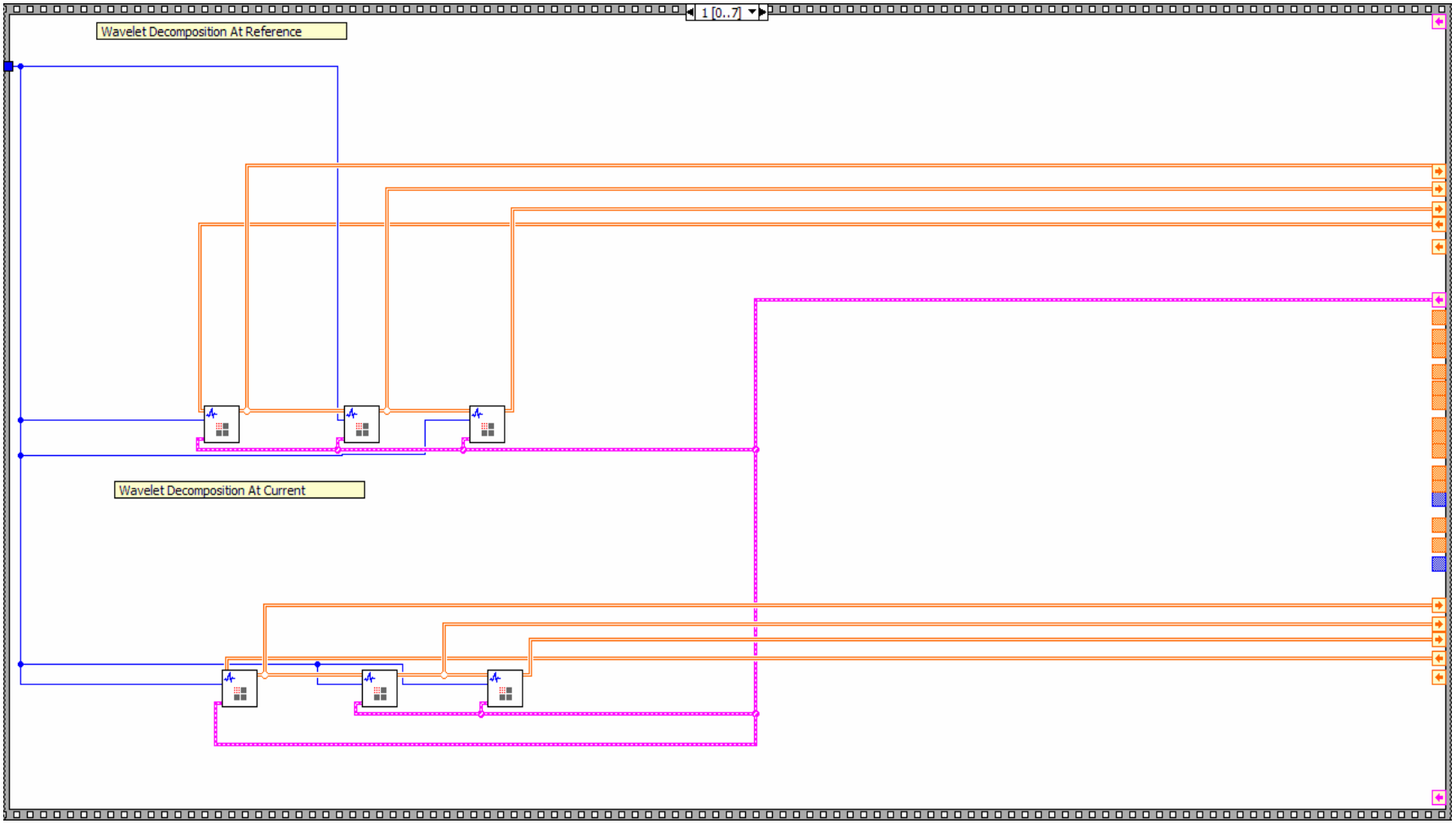
```
        for (bx=0;bx<numbx;bx++)
                for (by=0;by<numby;by++) {
                        sum=0;
                        sum2=0;
                        /* Some basic optimization */
                        placex=(bx+1)*bsizex;
                        placey=(by+1)*bsizey;
                        for (x=bx*bsizex;x<placex;x++)
                                for (y=by*bsizey;y<placey;y++) {
                                        sum = sum + templ[x,y]*image[x,y];
                                        sum2 = sum + image[x,y]*image[x,y];
                                }
                                if ( values[bx,by]->nccvalue <
(sum/Math::Sqrt(sum2)) ) {
                                        values[bx,by]->angle=r->angle;
                                        values[bx,by]->metx=r->metx;
                                        values[bx,by]->mety=r->mety;
                                        values[bx,by]-
>nccvalue=sum/Math::Sqrt(sum2);
                                }
                }
                return;
}


UnderVehicleSurv::Relatplace*
UnderVehicleSurv::Registration::vote(Relatplace* values[,])
{
        int dimx=values->GetLength(0);
        int dimy=values->GetLength(1);
        int i,j,k=0;
        ArrayList* r=new ArrayList();
        int a[50];

        for (i=0;i<50;i++)
                a[i]=0;

        for(i=0;i<dimx;i++)
                for (j=0;j<dimy;j++) {
                        if (!r->Contains(values[i,j])) {
                                r->Add(values[i,j]);
                                a[k++]++;
                        } else {
                                int l=r->IndexOf(values[i,j]);
                                a[l]++;
                        }
                }
```

```
        double pm=0;
        int pos=0;
        for (i=0;i<50;i++) {
                if (a[i]>pm) {
                        pm=a[i];
                        pos=i;
                }
        }
        return dynamic_cast<Relatplace*> (r->Item[pos]);
}

double UnderVehicleSurv::Registration::ncc(float image[,],float templ[,])
{
        int register x,y;
        double sum,sum2;
        int imagex=image->GetLength(0);
        int imagey=image->GetLength(1);
        sum=0;
        sum2=0;
        for (x=0;x<imagex;x++)
                for (y=0;y<imagey;y++) {
                        sum = sum + templ[x,y]*image[x,y];
                        sum2 = sum + image[x,y]*image[x,y];
                }
        return sum/Math::Sqrt(sum2);
        }
```

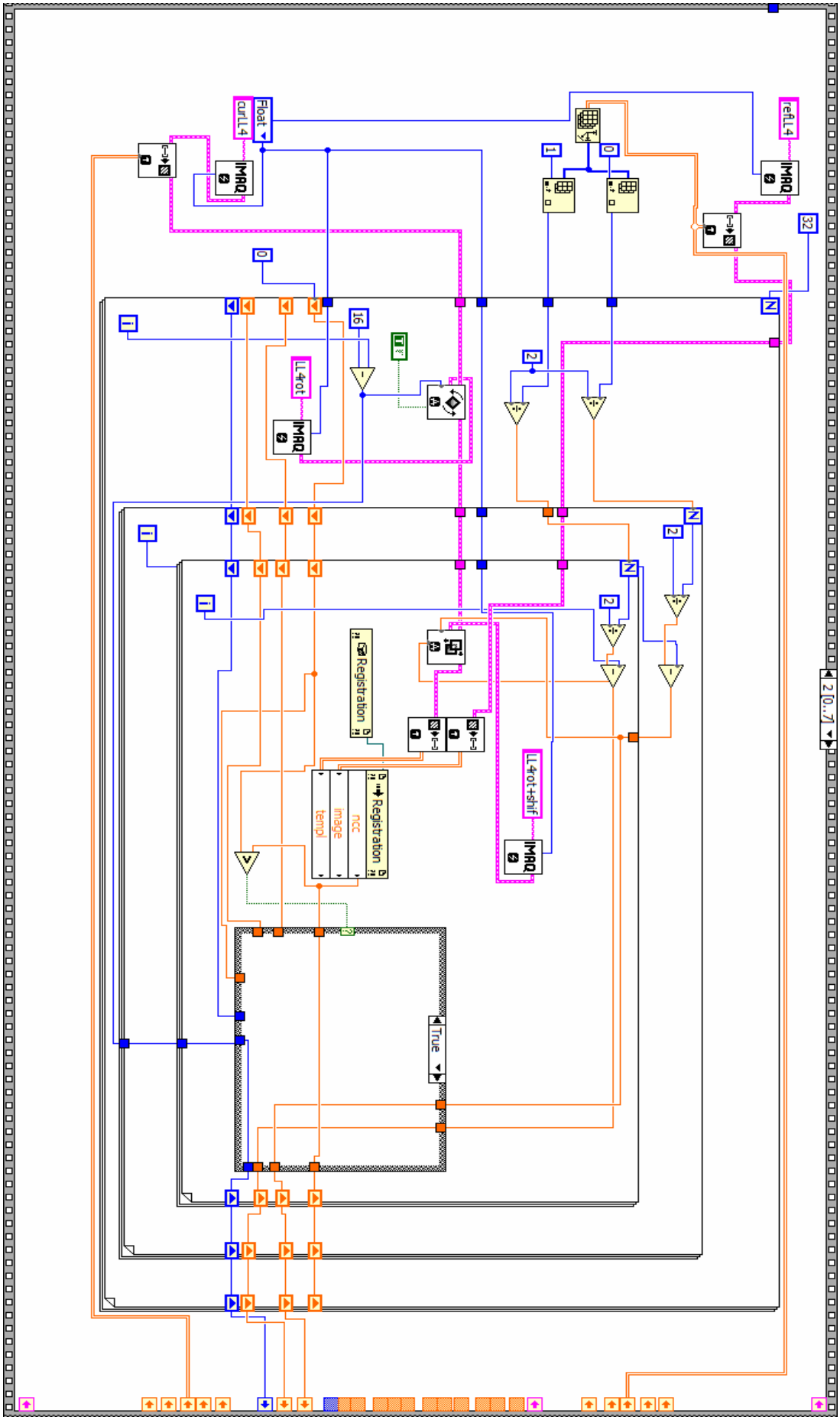
Ο σχηματικός κώδικας, που υλοποιήθηκε σε Labview, έχει ως εξής για την περίπτωση μη ελέγχου του θορύβου:



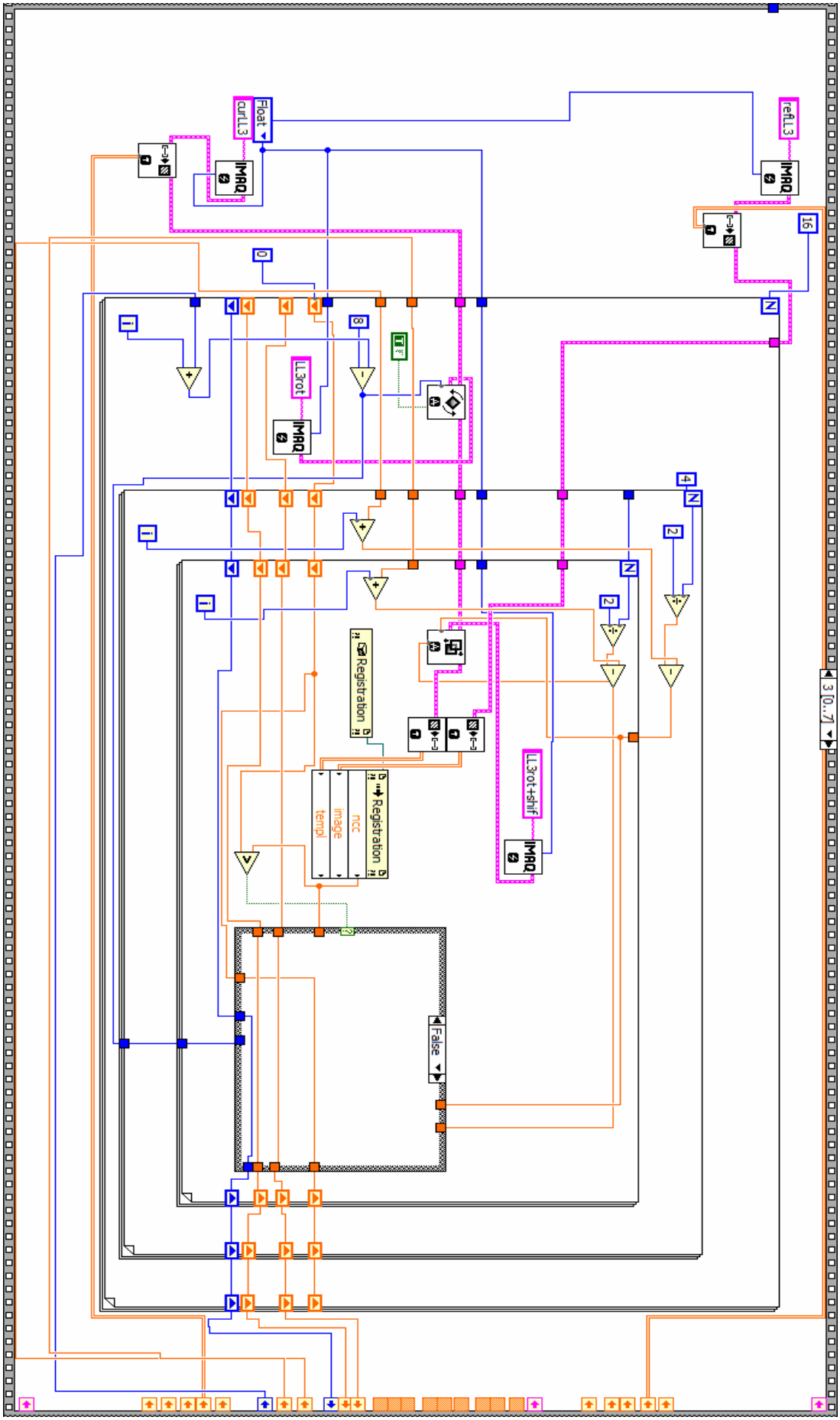
Πλαίσιο 1 του σχηματικού κώδικα Labview



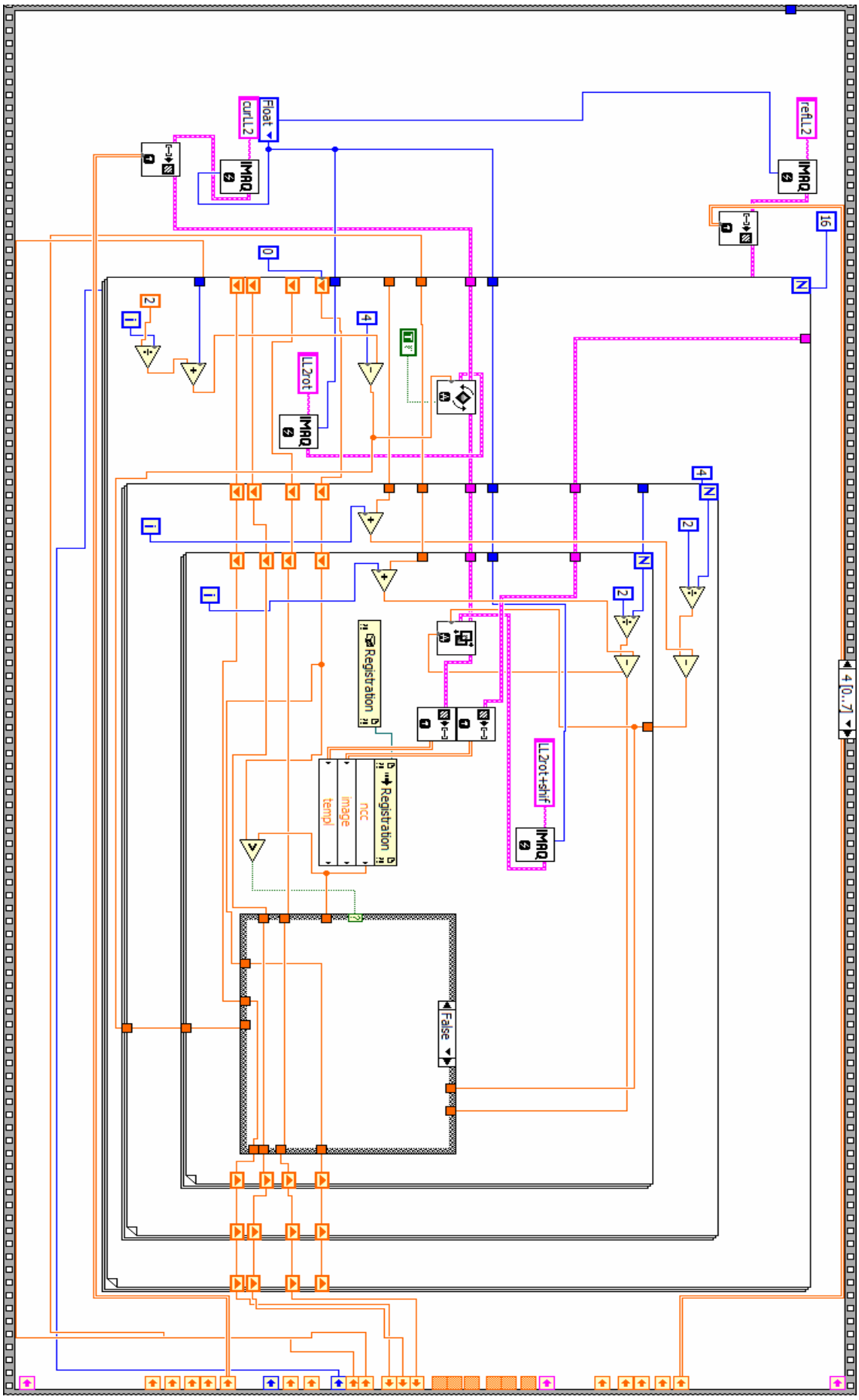
Πλαίσιο 2 του σχηματικού κώδικα Labview



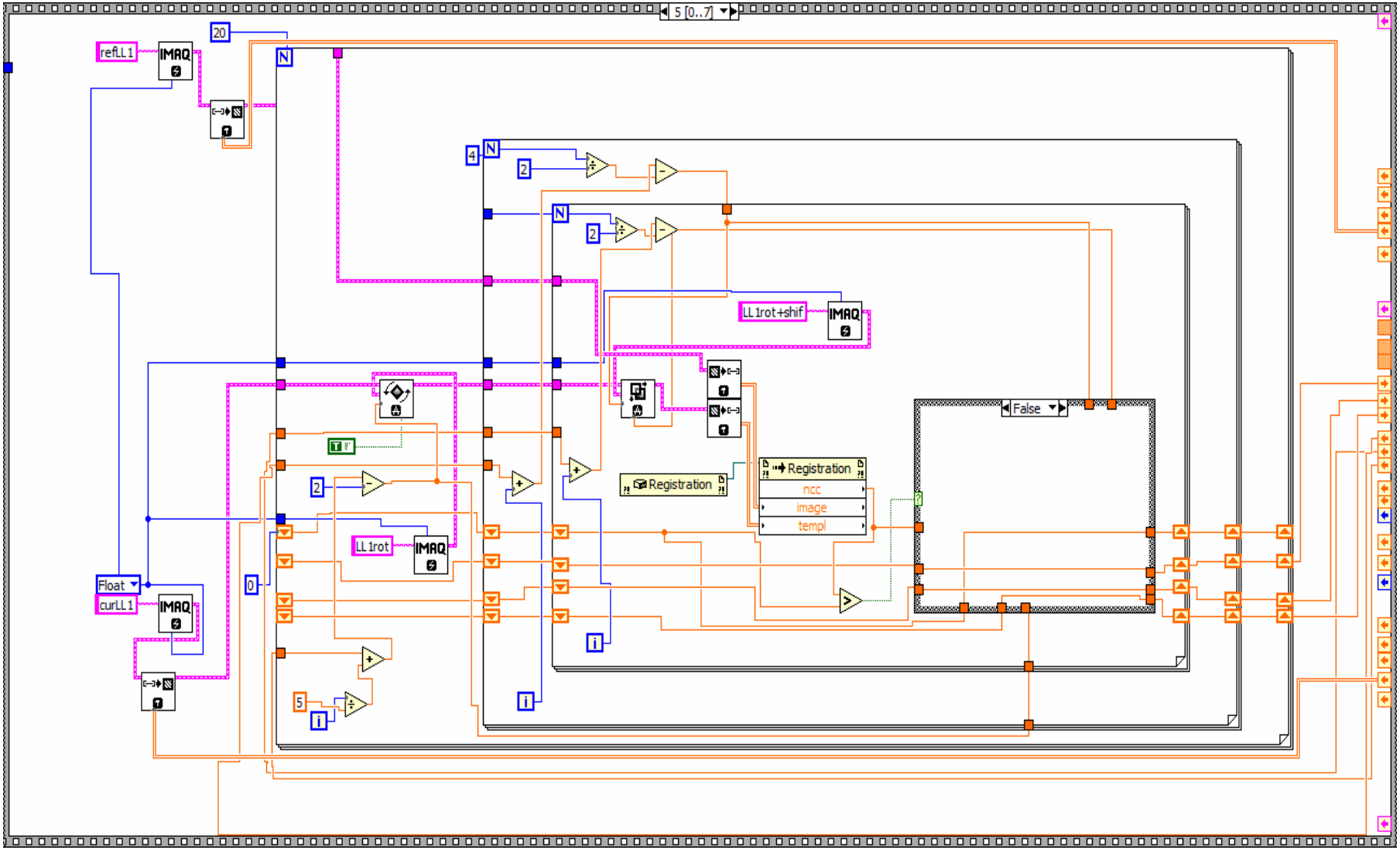
Πλαίσιο 3 του σχηματικού κώδικα Labview



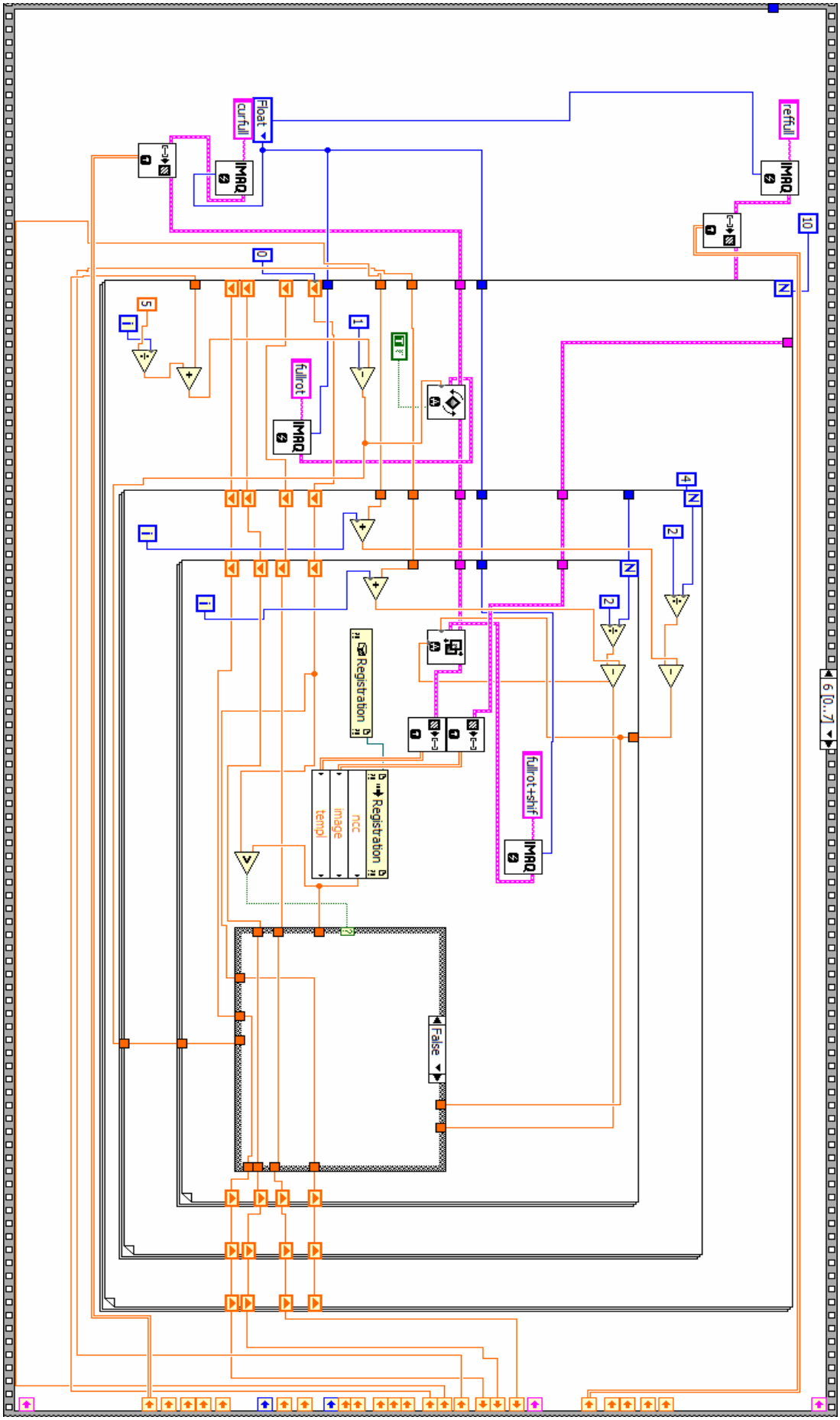
Πλαίσιο 4 του σχηματικού κώδικα Labview



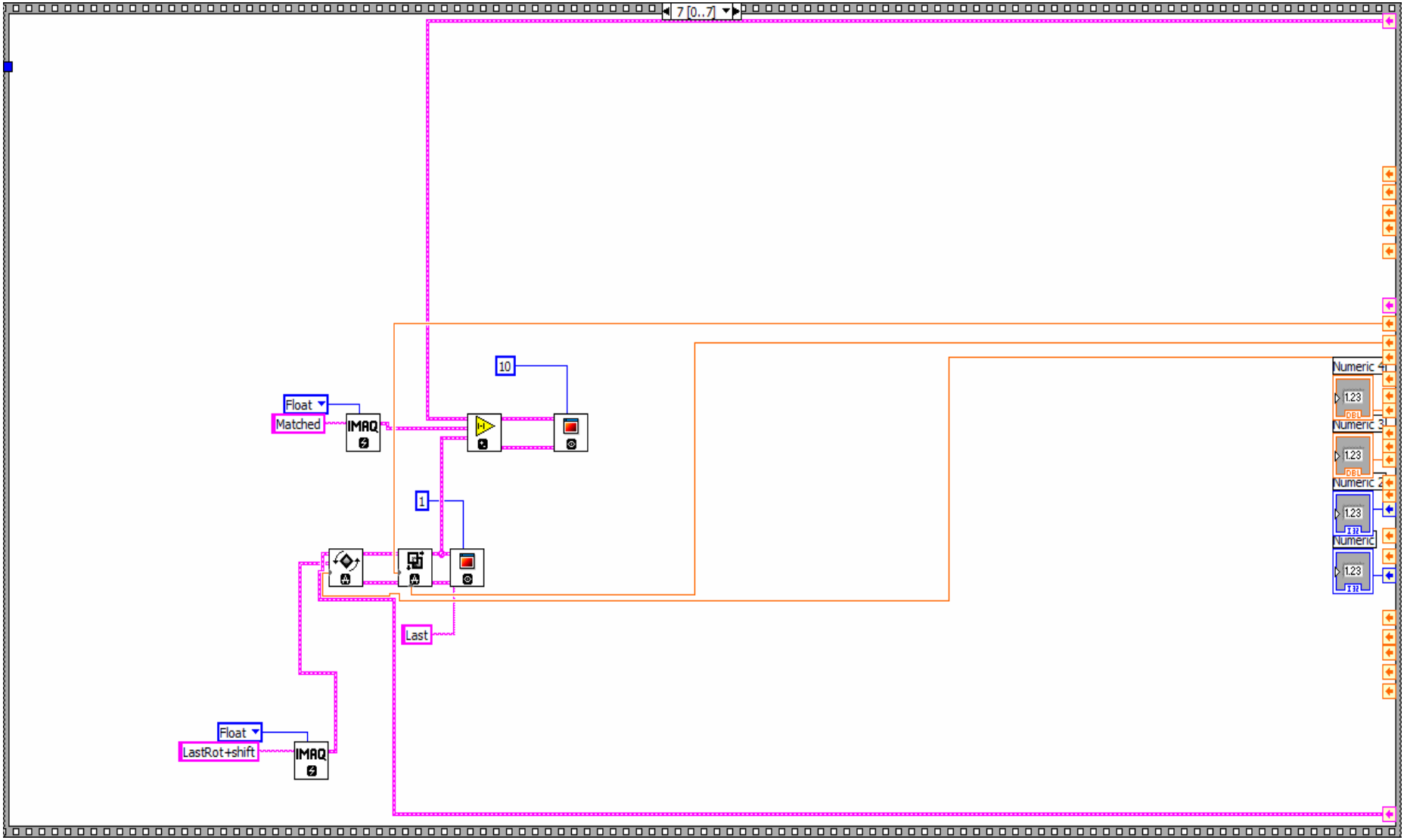
Πλαίσιο 5 του σχηματικού κώδικα Labview



Πλαίσιο 6 του σχηματικού κώδικα Labview

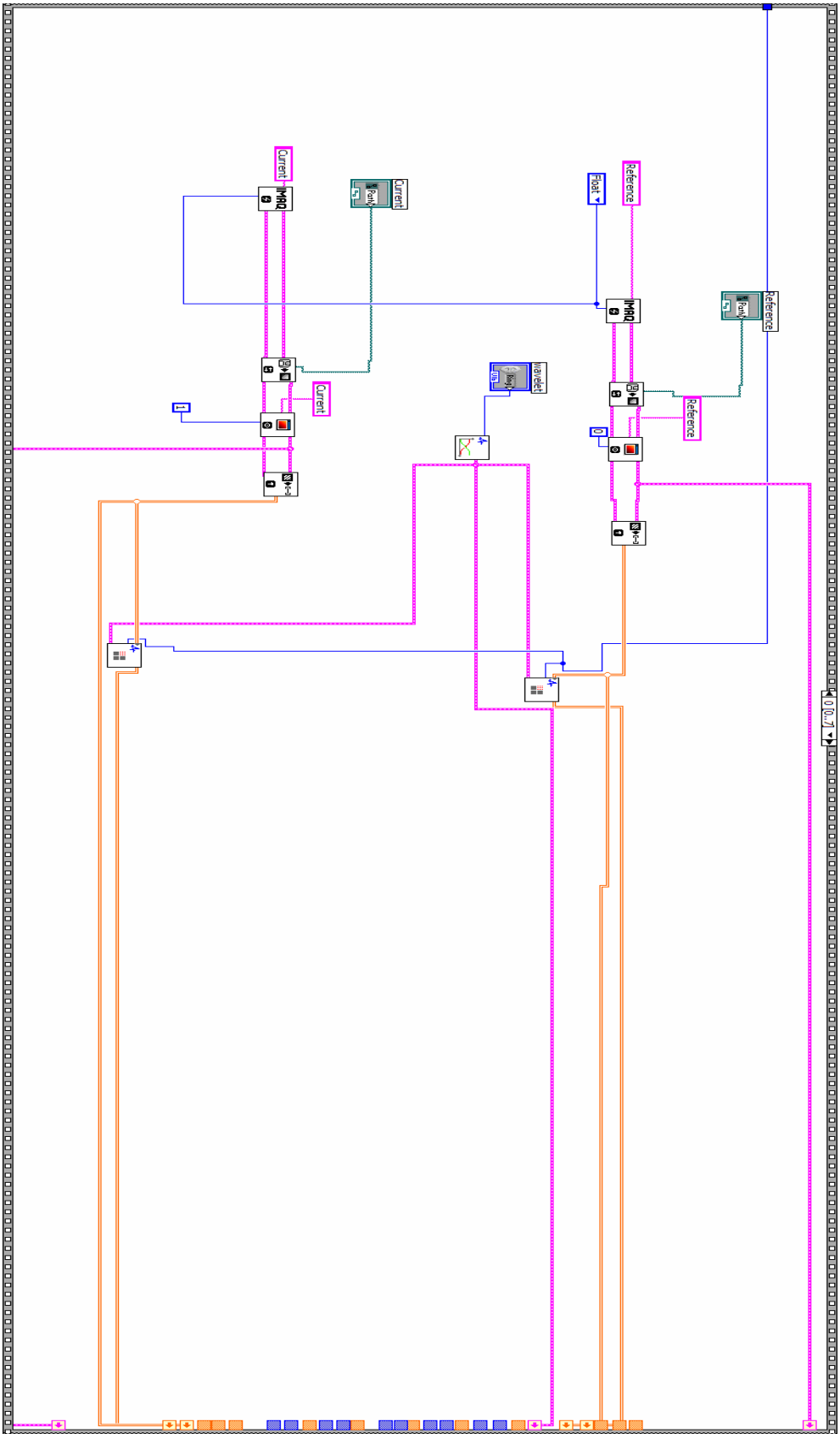


Πλαίσιο 7 του σχηματικού κώδικα Labview

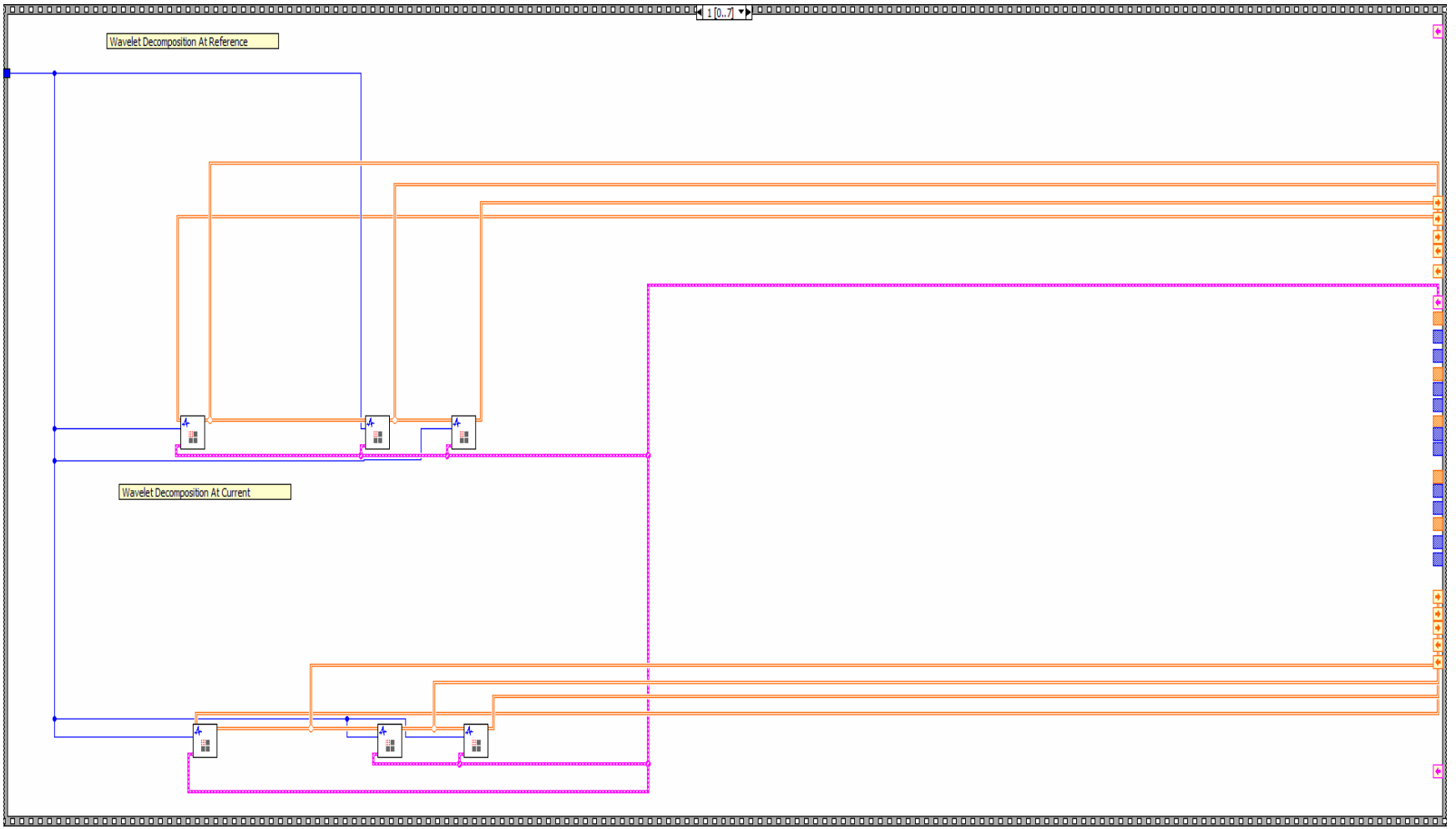


Πλαίσιο 8 του σχηματικού κώδικα Labview

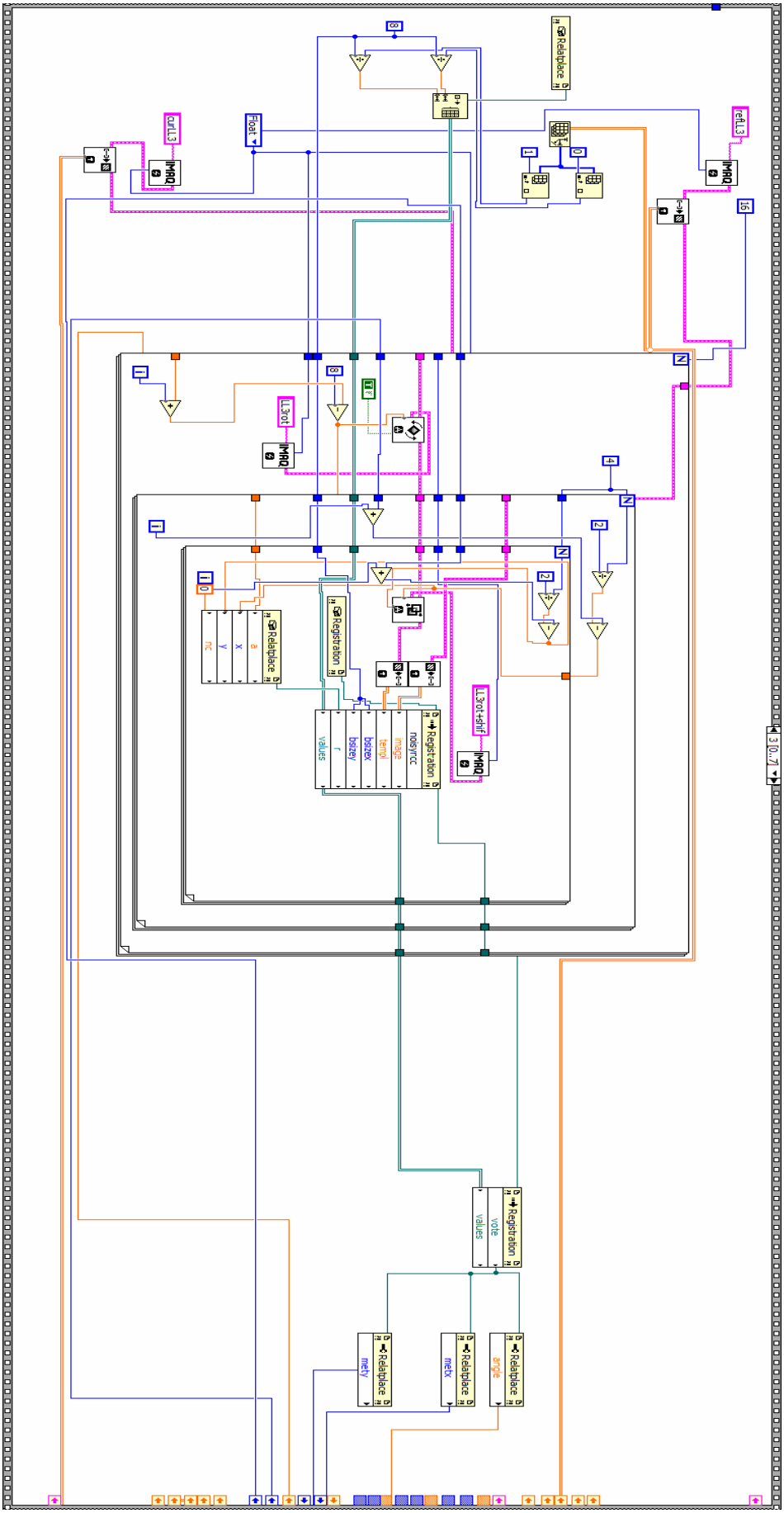
Ενώ για την περίπτωση ελέγχου του θορύβου τελικά ο σχηματικός κώδικας έχει ως εξής:



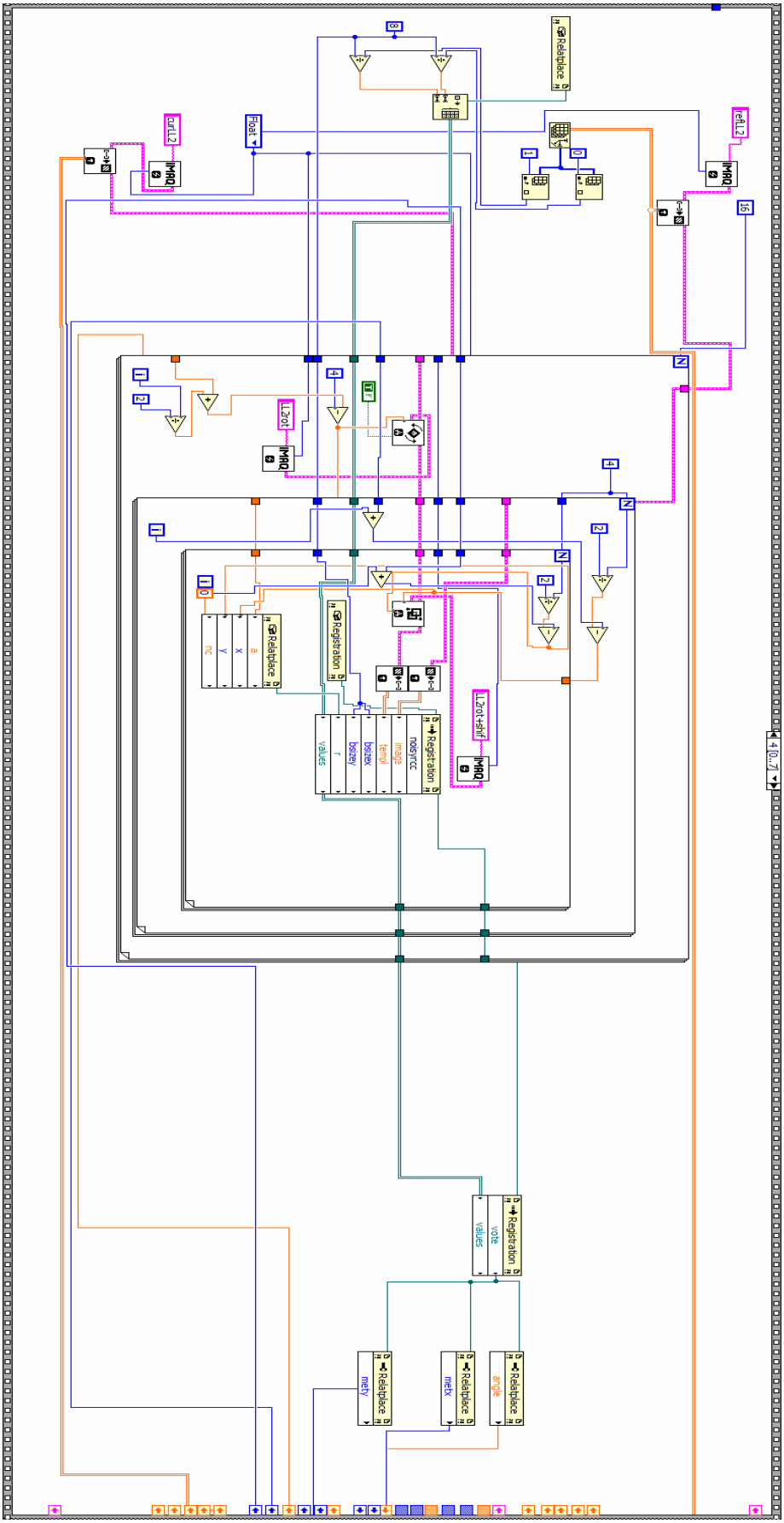
Πλαίσιο 1 του σχηματικού κώδικα Labview



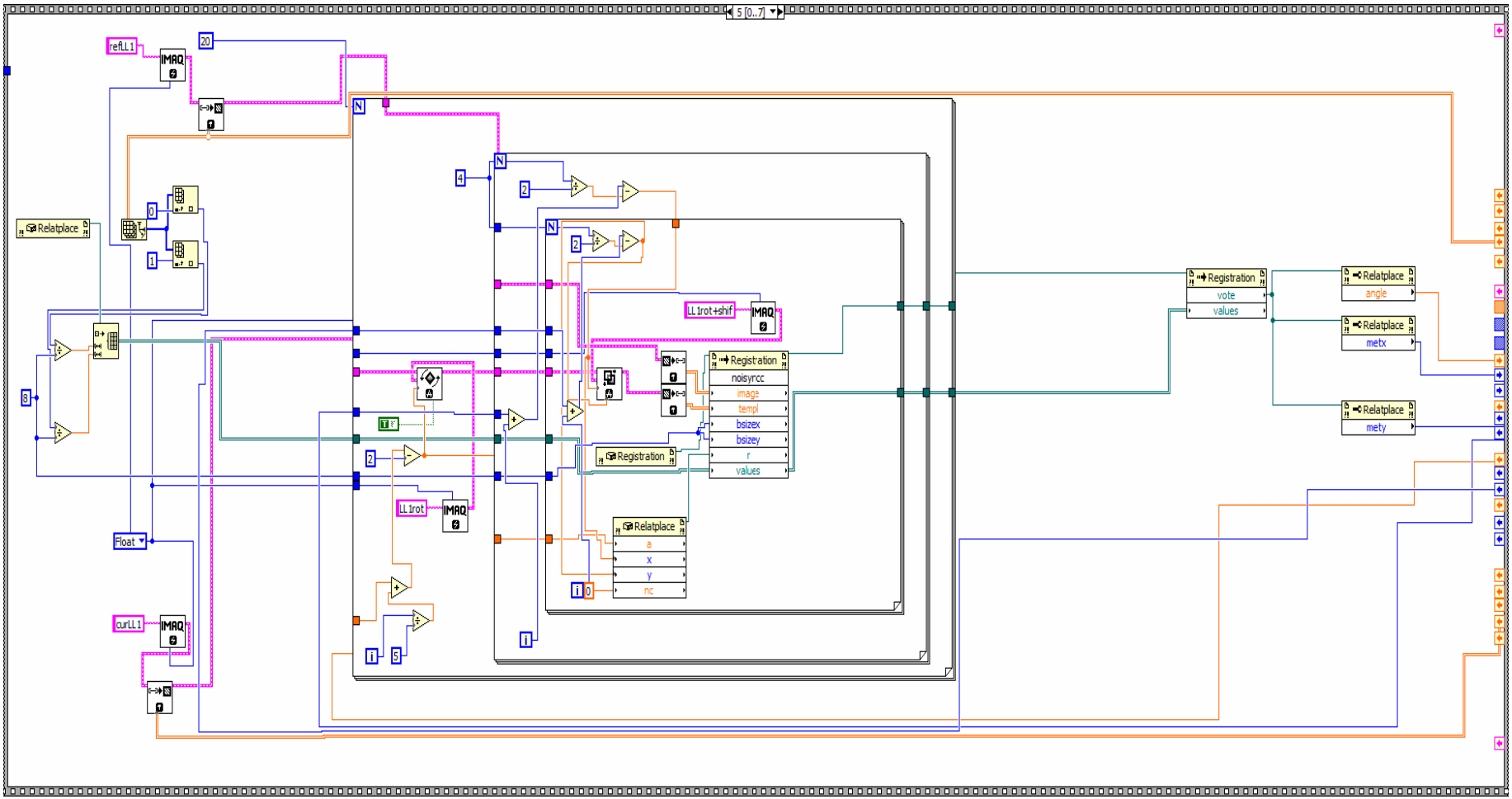
Πλαίσιο 2 του σχηματικού κώδικα Labview



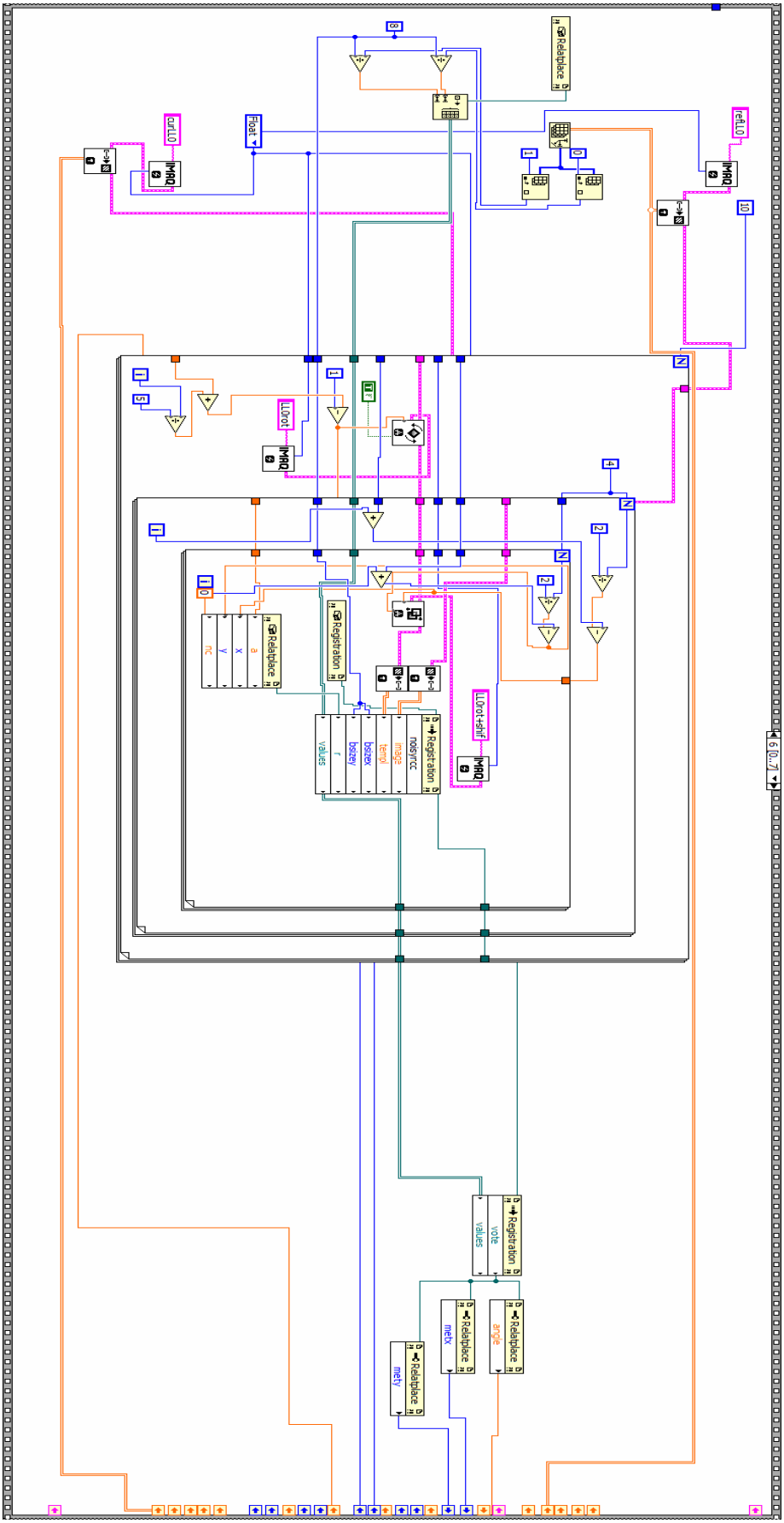
Πλαίσιο 3 του σχηματικού κώδικα Labview



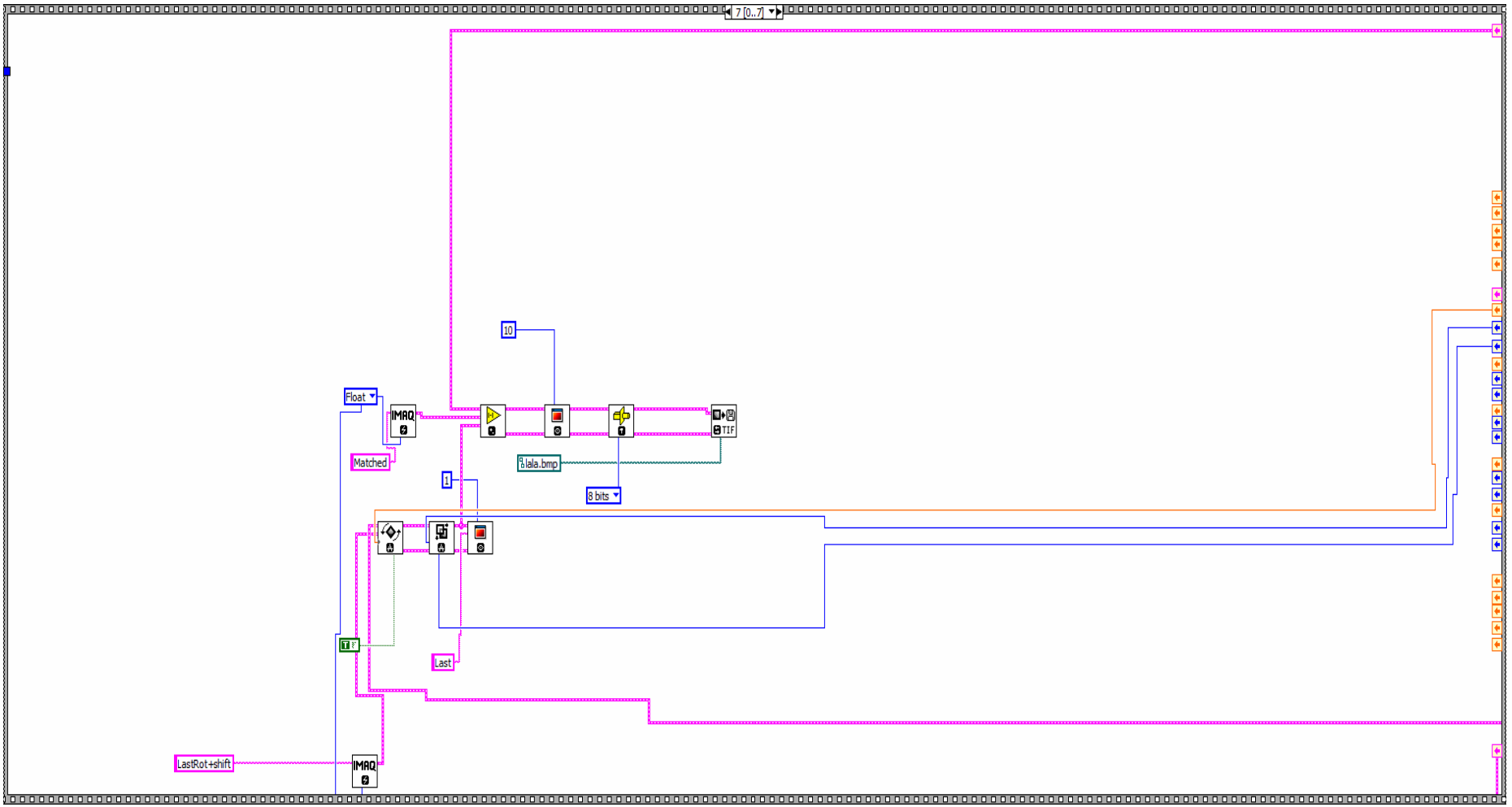
Πλαίσιο 4 του σχηματικού κώδικα Labview



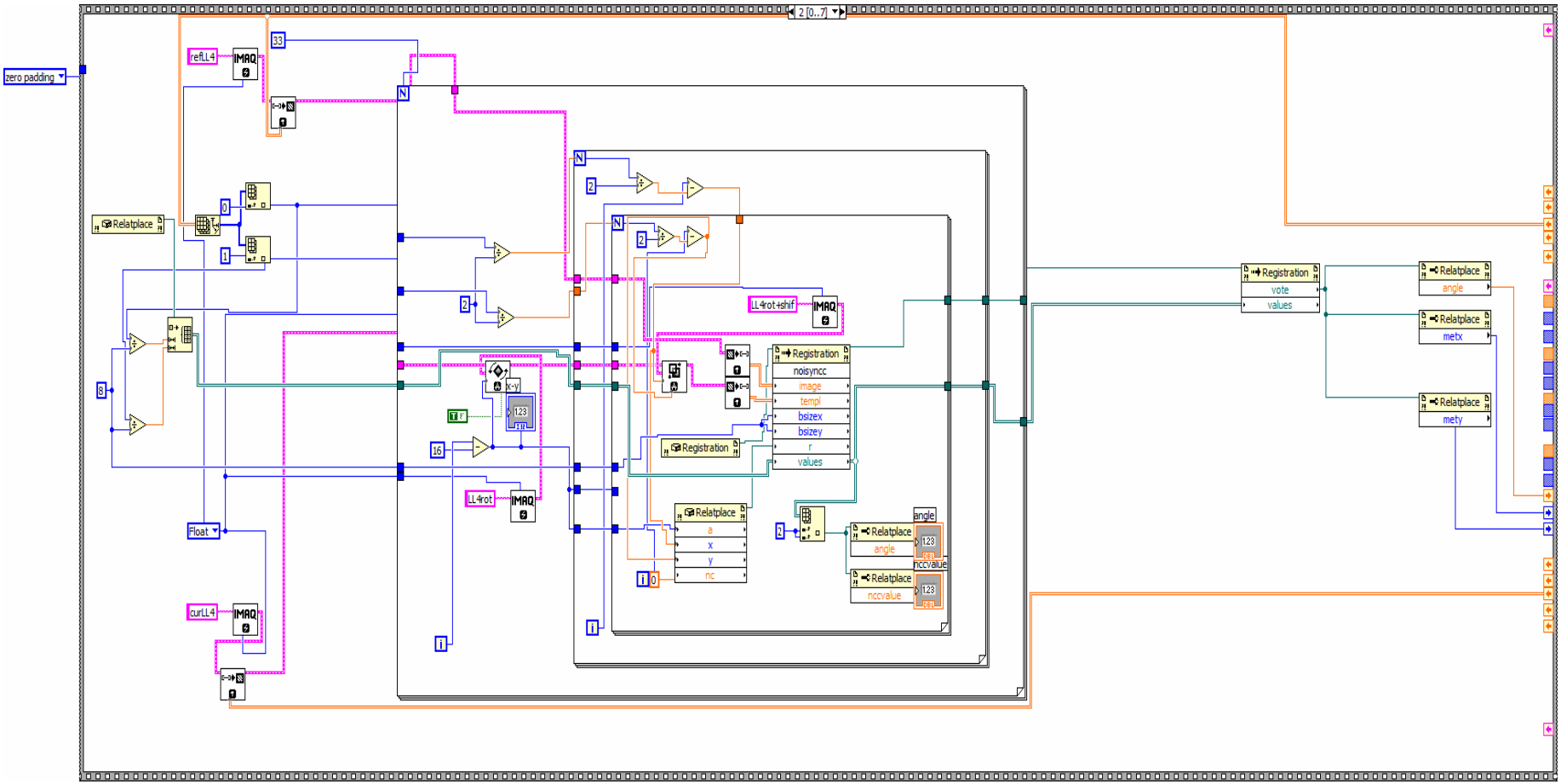
Πλαίσιο 5 του σχηματικού κώδικα Labview



Πλαίσιο 6 του σχηματικού κώδικα Labview



Πλαίσιο 7 του σχηματικού κώδικα Labview



Πλαίσιο 8 του σχηματικού κώδικα Labview

ΜΕΡΟΣ ΤΡΙΤΟ

ΤΕΧΝΙΚΗ ΑΝΙΧΝΕΥΣΗΣ ΑΛΛΑΓΩΝ ΒΑΣΙΣΜΕΝΗ ΣΕ ΟΜΑΔΕΣ ΣΤΟΙΧΕΙΩΝ ΕΙΚΟΝΑΣ

BLOCK-BASED CHANGE DETECTION

Μέρος Τρίτο

Εισαγωγή

Στο τρίτο μέρος της διπλωματικής εργασίας ασχολούμαστε με την ανεύρεση διαφορών μεταξύ των εικόνων που έχουμε λάβει από το δεύτερο μέρος, δηλαδή της αρχικής και της μεταβλημένης, κατά γωνία και θέση εικόνας, που λαμβάνουμε από το στάδιο 2.

Θεωρητικά για να βρεθούν οι αλλαγές μεταξύ δύο εικόνων αρκεί να γίνει η αφαίρεση τους. Στην περίπτωση που οι δύο εικόνες διαφέρουν μόνο κατά κάποια προσθήκη ή αφαίρεση αντικειμένων, αυτά θα πρέπει να είναι τα μόνα αντικείμενα στην τελική εικόνα. Αυτό συμβαίνει στην ιδεατή περίπτωση. Στην πραγματικότητα δύο εικόνες, ακόμη και εάν έχουν ληφθεί με ακριβώς τον ίδιο εξοπλισμό, χωρίς καμία μεταβολή σε οποιαδήποτε ελεγχόμενη, από τον ανθρώπινο παράγοντα, παράμετρο δεν πρόκειται να είναι ίδιες.

Ο λόγος για τον οποίο συμβαίνει αυτό είναι ο θόρυβος που εισαγάγει το περιβάλλον και που είναι πανταχού παρών και συνεχώς μεταβαλλόμενος. Μπορεί να οφείλεται σε μία πληθώρα λόγων, όπως παρεμβολές από ηλεκτρικά κυκλώματα, ανεπαίσθητες αλλαγές στο φωτισμό καθώς και σε άλλες αιτίες. Αν και υπάρχουν διάφορες μέθοδοι αφαίρεσης θορύβου με φίλτρα, στην περίπτωση μας δεν είναι δυνατόν να βοηθήσουν, καθώς δεν είναι δυνατόν να εξαλείψουν κάθε είδους θόρυβο, με συνέπεια το αποτέλεσμα τους να είναι μη ντετερμινιστικό και μη προβλέψιμο.

Παρακάτω γίνεται μία επίδειξη δύο εικόνων που, ενώ οπτικά διαφέρουν μόνο κατά ένα αντικείμενο, γίνεται εμφανής, μετά την αφαίρεση τους η παρουσία του θορύβου. Οι εικόνες 1α και 1β είναι οι εικόνες από την κάμερα ενώ οι 1γ και 1δ εικόνες είναι οι απόλυτος διαφορά και η απλή διαφορά των 2. Είναι εμφανής η παρουσία θορύβου που κάνει απαγορευτική την κατευθείαν χρήση αυτών των μεθόδων για την ανεύρεση διαφορών.

Εικόνα 1α

Εικόνα 1β

Εικόνα 1γ

Εικόνα 1δ

Μία παρατήρηση που είναι δυνατόν να γίνει σε αυτό το σημείο είναι ότι η απόλυτος διαφορά των δύο εικόνων, ενώ περιέχει το διαφορά θορύβων μεταξύ των δύο εικόνων, περιέχει επίσης και αυτούσιο το αντικείμενο το οποίο αναζητούμε. Η συγκεκριμένη διαπίστωση οδήγησε στην ανάπτυξη του παρακάτω αλγορίθμου ανίχνευσης αλλαγών.

Περιγραφή Αλγορίθμου

Όπως μπορούμε εύκολα να συμπεράνουμε σε μία τέτοια εικόνα, όπως η απόλυτος διαφορά δύο εικόνων, τα στοιχεία εικόνας (pixels) έχουν γενικώς σταθερή τυπική απόκλιση των τιμών της κλίμακας του γκρι. Στην περίπτωση που κάτι τέτοιο δεν ισχύει για κάποιο στοιχείο εικόνας(pixel), είναι πιθανόν αυτό το στοιχείο να ανήκει σε κάποιο από τα αντικείμενα της εικόνας, που προσπαθούμε να ανιχνεύσουμε.[9]

Η συγκεκριμένη ιδιότητα θα ισχύει και για τα γειτονικά του στοιχεία, με την εξαίρεση των ακραίων στοιχείων. Αυτό συνεπάγεται ότι η ίδια ιδιότητα θα ισχύει για γειτνιάσεις στοιχείων, γεγονός που σημαίνει ότι είναι δυνατόν να δουλέψουμε σε γειτνιάσεις στοιχείων, διευκολύνοντας αλλά και επιταχύνοντας σημαντικά τη διαδικασία. Συγκεκριμένα στην περίπτωσή μας, οι ομάδες, που αποφασίσαμε να χρησιμοποιήσουμε, ήταν ορθογώνια παραλληλόγραμμα από γειτονικά στοιχεία εικόνας, όπου προφανώς τα μεγέθη μήκος, πλάτος είναι μεταβλητά και μπορεί να αλλάζουν ανάλογα με την εφαρμογή.

Η διαδικασία που ακολουθήσαμε ήταν η εξής. Καταρχάς η εικόνα χωρίζεται σε ίσου εμβαδού τετράγωνα με πλευρά δεκαέξι (16) στοιχείων, που στο εξής θα ονομάζομε μπλοκ, εκφυλίζοντας τα ορθογώνια παραλληλόγραμμα που περιγράψαμε πριν, όπου για καθένα από τα οποία υπολογίζεται η μέση τιμή και η τυπική απόκλιση. Οι τύποι που χρησιμοποιούνται είναι οι:

$$m = \frac{1}{XY}\sum_{XY} I(x,y) \quad \text{μέση τιμή}$$

$$\sigma = \sqrt{\frac{1}{XY}\sum_{XY}(I(x,y)-m)^2} \quad \text{τυπική απόκλιση}$$

Όπου προφανώς X=Y=16.

Κατόπιν ακολουθείται ένας αλγόριθμος κατάταξης καθενός μπλοκ από τα προαναφερθέντα ανάλογα με τη μέση τιμή του που μόλις υπολογίσαμε. Δημιουργούμε συστάδες (cluster) που η κάθε μία χαρακτηρίζεται από τις μέσες τιμές των γνωρισμάτων των μπλοκ, που την απαρτίζουν. Κάθε μπλοκ τοποθετείται σε μία συστάδα (cluster) ανάλογα με την απόλυτο διαφορά της μέσης τιμής της κλίμακας του γκρι και της μέσης τιμής της συστάδας. Συγκεκριμένα έχουμε τη σχέση

$$d_{ij,G_k} = \left| m_{ij} - m_{G_k} \right|$$

Όπου i,j οι δείκτης που ορίζουν το αντίστοιχο μπλοκ και m_{ij} η μέση τιμή του μπλοκ. Αντίστοιχα το m_{G_k} είναι η μέση τιμή της τρέχουσας συστάδας(cluster). Το συγκεκριμένο κριτήριο επαναλαμβάνει για συστάδα που υπάρχει και για κάθε ένα που ισχύει

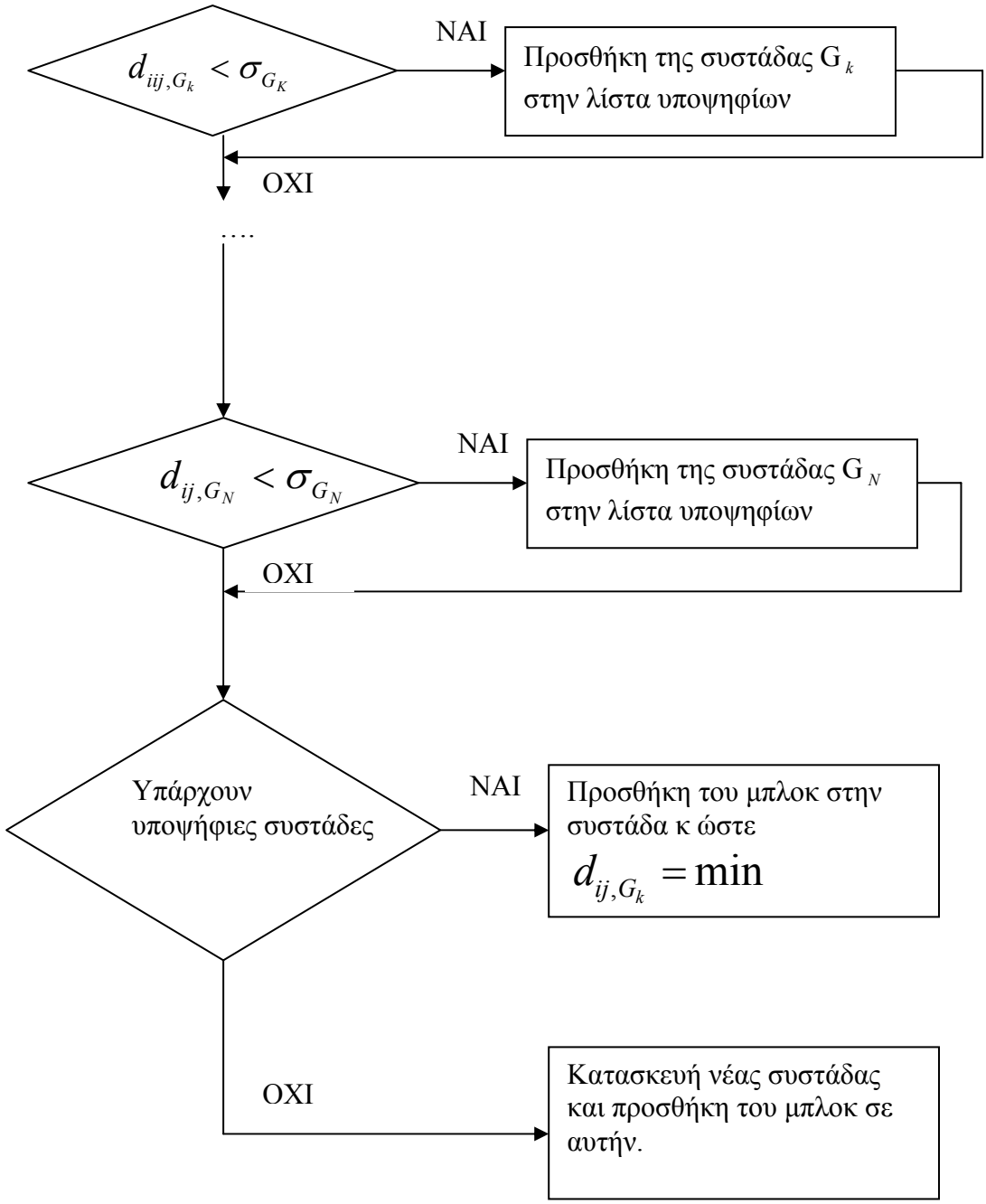
$$d_{ij,G_k} = \left| m_{ij} - m_{G_k} \right| < \sigma_{G_k}$$

Προσθέτουμε την αντίστοιχη συστάδα σε μία λίστα υποψηφίων συστάδων. Στο τέλος διατρέχουμε τη λίστα των υποψηφίων συστάδων και προσθέτουμε το μπλοκ στη συστάδα όπου

$$d_{ij,G_k} = \min$$

Προφανώς μετά την προσθήκη του νέου μπλοκ στη συστάδα επαναϋπολογίζουμε τα δύο χαρακτηριστικά μεγέθη της δηλαδή τη μέση τιμή και την τυπική απόκλιση.

Στην περίπτωση που δεν υπάρχει συστάδα στην οποία να μπορούμε να προσθέσουμε το μπλοκ, τότε ιδρύουμε μία νέα συστάδα και προσθέτουμε σε αυτή το νέο μπλοκ. Σχηματικά ο αλγόριθμος παρίσταται με το παρακάτω διάγραμμα ροής (flowchart). [9]


$d_{iij,G_k} < \sigma_{G_K}$
ΝΑΙ
Προσθήκη της συστάδας G_k στην λίστα υποψηφίων
ΟΧΙ
….
$d_{ij,G_N} < \sigma_{G_N}$
ΝΑΙ
Προσθήκη της συστάδας G_N στην λίστα υποψηφίων
ΟΧΙ
Υπάρχουν υποψήφιες συστάδες
ΝΑΙ
Προσθήκη του μπλοκ στην συστάδα κ ώστε $d_{ij,G_k} = \min$
ΟΧΙ
Κατασκευή νέας συστάδας και προσθήκη του μπλοκ σε αυτήν.

Μετά την εκτέλεση του παραπάνω αλγορίθμου, για όλα τα μπλοκ της απολύτου διαφοράς έχουμε έναν αριθμό από συστάδες, οι οποίες χαρακτηρίζονται από συγκεκριμένες τιμές των χαρακτηριστικών τους. Θεωρώντας ότι οι προσθήκες αντικειμένων στην εικόνα δεν είναι τόσο εκτεταμένες, ώστε να ξεπερνούν το ήμισυ της εικόνας, μπορούμε να κάνουμε την παραδοχή ότι η μεγαλύτερη σε μέγεθος συστάδα θα περιέχει μπλοκ τα οποία δεν θα ανήκουν σε κάποιο από τα αντικείμενα, άρα θα ανήκουν στον περιβάλλοντα χώρο και δεδομένου ότι ο περιβάλλοντας χώρος, στην περίπτωση της απολύτου διαφοράς εικόνων, είναι ο υπεισερχόμενος θόρυβος, τότε μπορούμε, με ασφάλεια, να πούμε ότι τα χαρακτηριστικά του θορύβου προσεγγίζονται από τα χαρακτηριστικά της μεγαλύτερης συστάδας.[9]

$$m_{G_{\max}} = m_{noise}$$

και

$$\sigma_{G_{\max}} = \sigma_{noise}$$

Δεδομένης αυτής της παραδοχής εξάγουμε το παρακάτω κριτήριο:

$$\left| m_{ij} - m_{Gnoise} \right| > \sigma_{G_{noise}}$$

Τότε το μπλοκ i,j έχει υποστεί αλλαγή και ανήκει σε κάποιο από τα αντικείμενα τα οποία θέλουμε να ανιχνεύσουμε. Συνεπώς, δημιουργούμε μία δυαδική μάσκα, όπου για κάθε μπλοκ για το οποίο ισχύει το παραπάνω κριτήριο θέτουμε όλα τα αντίστοιχα στοιχεία εικόνας (pixels) ίσα με ένα (1) και όλα τα υπόλοιπα ίσα με το μηδέν (0).

Το επόμενα βήμα είναι η χρήση ενός median φίλτρου πάνω στη δυαδική μάσκα ώστε να υπάρξει εξομάλυνση των άκρων των αντικειμένων και αφαίρεση πιθανών απομονωμένων μπλοκ, που τηρούσαν το παραπάνω κριτήριο.

Ένα median φίλτρο λειτουργεί σε ομάδες γειτονικών στοιχείων εικόνας(pixel blocks) όπου παίρνει όλα τα στοιχεία , τα τοποθετεί κατά αύξουσα αριθμητική σειρά και στην περίπτωση περιττού αριθμού στοιχείων παίρνει το ενδιάμεσο, στην περίπτωση άρτιου αριθμού στοιχείων, ανάλογα με την υλοποίηση, παίρνει κάποιο από τα δύο ενδιάμεσα στοιχεία ή ,το συνηθέστερο, το μέσο όρο αυτών.

Σε μία δυαδική εικόνα ένα median φίλτρο λειτουργεί με τον ίδιο τρόπο μόνο που δεδομένου ότι οι τιμές των στοιχείων της εικόνας θα είναι άσσοι ή μηδενικά, είναι αρκετό να βρεθεί, μέσα στην ομάδα των στοιχείων, ποια από τις δύο τιμές έχει μεγαλύτερο αριθμό εμφανίσεων. Θέτει τότε στην αντίστοιχη τιμή το στοιχείο.

Ένα τέτοιο φίλτρο μπορεί , όπως στην περίπτωση μας, να υλοποιηθεί και σε επίπεδο ομάδων (block-based). Συγκεκριμένα αντί να μελετάμε ένα-ένα τα στοιχεία μελετάμε ολόκληρες ομάδες αφού όλα τα στοιχεία στην ομάδα έχουν ακριβώς την ίδια τιμή. Σε αυτή την περίπτωση ελέγχονται όλα τα γειτονικά μπλοκ του υπό εξέταση μπλοκ και ανάλογα με τον αριθμό των μπλοκ που απαρτίζονται μόνο από μηδενικά ή μόνο από άσσους αποφασίζεται η τιμή του μπλοκ.

Στην περίπτωση που μελετάμε έγχρωμες εικόνες, ο αλγόριθμος είναι και πάλι λειτουργικός με την εξής απλή αλλαγή. Χωρίζουμε την απόλυτο διαφορά των εικόνων στις χρωματικές συνιστώσες της και ασκούμε τον αλγόριθμο σε κάθε μία από αυτές. Μετά το πέρας των υπολογισμών έχουμε τρείς δυαδικές εικόνες (μάσκες) στις οποίες εκτελούμε τη λογική πράξη Η (OR), η οποία μας επιστρέφει μία συνολική δυαδική εικόνα, την οποία χρησιμοποιούμε ως μάσκα. Συγκεκριμένα ο αλγόριθμος, αν και πιο αργός σε έγχρωμες εικόνες είναι πιο ακριβής αφού ανακαλύπτει αλλαγές τόσο στο επίπεδο της φωτεινότητας ώστε και της χρωματικότητας σε εικόνες , γεγονός το οποίο δεν είναι δυνατόν να συμβεί στις ασπρόμαυρες εικόνες.

Όπως αναφέραμε και παραπάνω, ο αλγόριθμος δουλεύει αξιόπιστα όταν οι αλλαγές περιεχομένου δεν ξεπερνούν το ήμισυ της εικόνας. Αυτό γίνεται γιατί η συστάδα, που ορίζει τα χαρακτηριστικά του θορύβου, δεν έχει πλέον την απόλυτη πλειοψηφία σε αριθμό μπλοκ. Παρόλα αυτά, εάν κρατηθεί η σχετική πλειοψηφία, γεγονός το οποίο συμβαίνει κατά την εισαγωγή περισσοτέρων του ενός ξεχωριστών αντικειμένων στην εικόνα, όπου κάθε αντικείμενο χωριστά θα πρέπει να αποτελείται τελικώς από λιγότερα μπλοκ από το θόρυβο, ο αλγόριθμος είναι λειτουργικός, αν και προφανώς δεν έχει την ίδια αξιοπιστία.

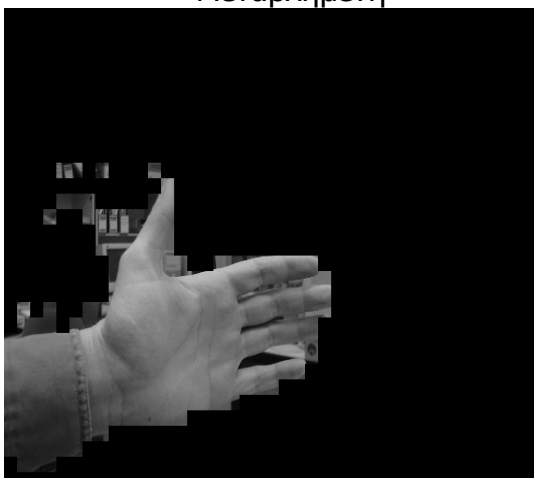
Τα αποτελέσματα του αλγορίθμου πάνω σε ένα αριθμό από εικόνες δίνονται ανά τετράδες παρακάτω. Για κάθε παράδειγμα δίνουμε την αρχική εικόνα, την εικόνα μεταβολής, τη μάσκα που υπολογίστηκε και τέλος το αποτέλεσμα. Θα παρατηρηθεί ότι ο αλγόριθμος λειτουργεί ανεξαρτήτως του αριθμού των αντικειμένων αρκεί να τηρείται ο κανόνας της σχετικής πλειοψηφίας.

Παράδειγμα 1ο

Αρχική

Μεταβλημένη

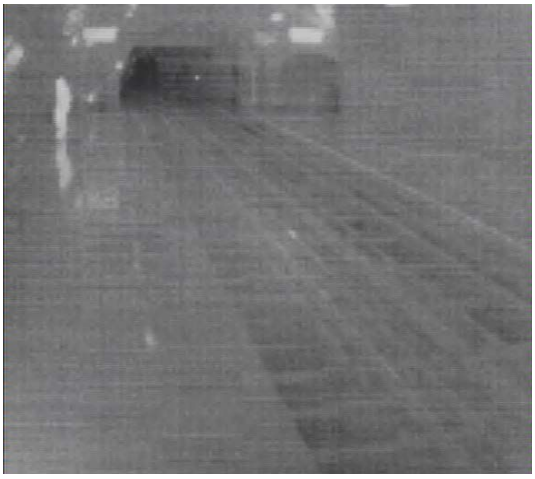
Μάσκα



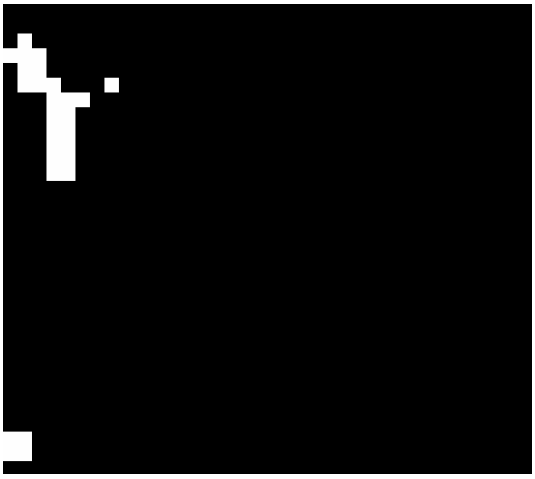
Αποτέλεσμα

Παράδειγμα 2ο

Αρχική



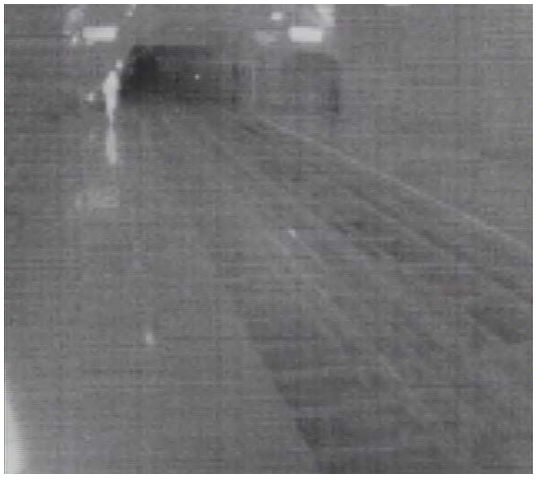
Μεταβλημένη



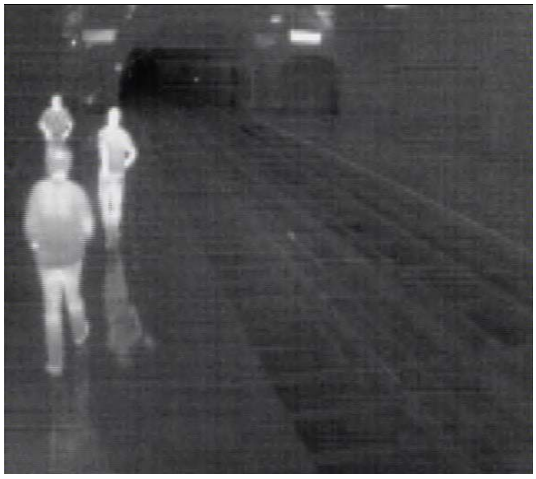
Μάσκα

Αποτέλεσμα

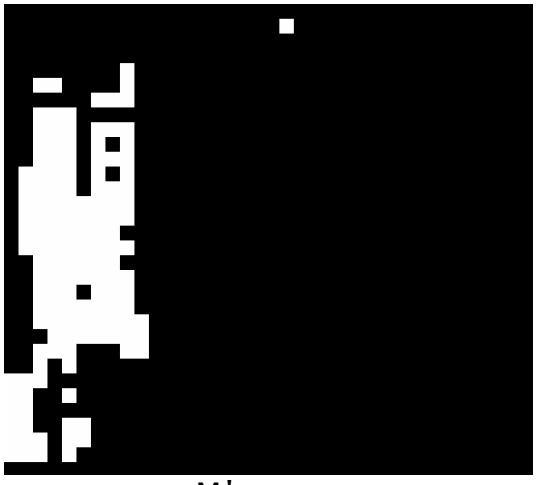
Παράδειγμα 3^{ο}



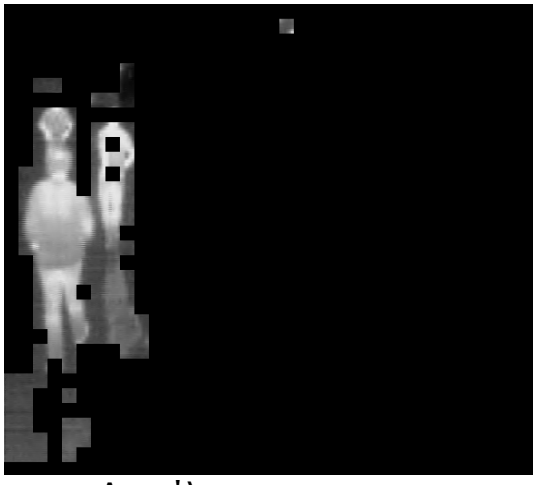
Αρχική



Μεταβλημένη



Μάσκα



Αποτέλεσμα

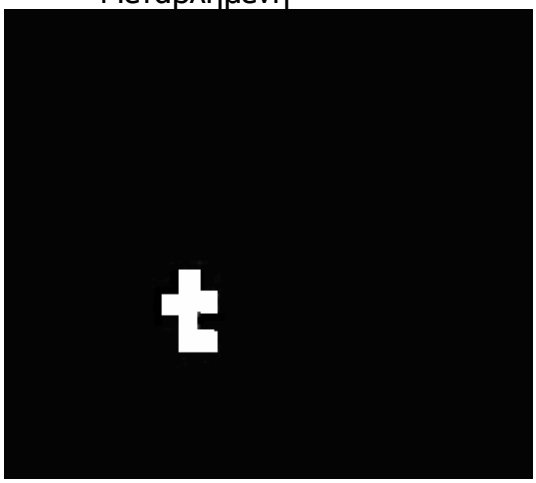
Τέλος παρουσιάζουμε και ένα παράδειγμα όπου η είσοδος του συστήματος προέρχεται από το προηγούμενο αλγόριθμο.

Παράδειγμα 4^{ο}

Αρχική

Μεταβλημένη

Αποτέλεσμα προηγούμενου σταδίου



Μάσκα

Τελικό Αποτέλεσμα

Παράθεση Υλοποίησης

Παρακάτω παραθέτουμε τον κώδικα που υλοποιήθηκε σε C++ στα πλαίσια αυτού του μέρους. Η υλοποίηση ήταν πλήρως αντικειμενοστραφής για αυτό υπάρχουν οι κλάσεις Block, Cluster, BbMedianFilter που υλοποιούν τα αντίστοιχα κομμάτια του προγράμματος.

```
// Block class declarations

#pragma once

using namespace System;

namespace UnderVehicleSurv
{
	public __gc class Block
	{
	private:
		int m_place_x; // Relative x coordinate
		int m_place_y; // Relative y coordinate
		int m_dimx; // x coordinate size
		int m_dimy; // y coordinate size
		double m_meanval; // Block's mean value
		double m_std; // Block's standard deviation
		double meanvalue(Byte image[,]);
		double deviation(Byte image[,]);
	public:
		__property int get_dimx();
		__property int get_dimy();
		__property int get_placex();
		__property int get_placey();
```

```
        __property double get_meanval();
        __property double get_std();

        Block(Byte block[,],int x,int y);
    };
}

// Block class implemantations

#include "stdafx.h"
#include "Block.h"



int UnderVehicleSurv::Block::get_dimx()
{
    return m_dimx;
}

int UnderVehicleSurv::Block::get_dimy()
{
    return m_dimy;
}

int UnderVehicleSurv::Block::get_placex()
{
    return m_place_x;
}

int UnderVehicleSurv::Block::get_placey()
{
    return m_place_y;
```

```
}

double UnderVehicleSurv::Block::get_meanval()
{
        return m_meanval;
}


double UnderVehicleSurv::Block::get_std()
{
        return m_std;
}
double UnderVehicleSurv::Block::meanvalue(Byte block[,])
{
        register int i,j;
        int size=dimx*dimy;
        unsigned int sum=0;

        for (i=0;i<dimx;i++)
                for (j=0;j<dimy;j++) {
                        sum += block[i,j];
                }
        this->m_meanval= sum/size;
        return m_meanval;
}
double UnderVehicleSurv::Block::deviation(Byte block[,])
{
        register int i,j;
    int size=dimx*dimy;
        double std=0;

        for (i=0;i<dimx;i++)
```

```
                for (j=0;j<dimy;j++) {
                        std += (block[i,j]-m_meanval)*(block[i,j]-m_meanval);
                }
        std /= size;
        std = System::Math::Sqrt(std);
        this->m_std=std;
        return m_std;
}
UnderVehicleSurv::Block::Block(Byte block[,],int x,int y)
{
        m_dimx=block->GetLength(0);
        m_dimy=block->GetLength(1);
        m_place_x=x;
        m_place_y=y;

        meanvalue(block);
        deviation(block);
}

// Cluster class declarations

#pragma once

using namespace System;
using namespace System::Collections;

#include "Block.h"

namespace UnderVehicleSurv
{
        public __gc class Cluster
        {
```

```
        private:
                double m_meanval; //Cluster's mean value of mean values
                double m_std; // Cluster's mean value of standard deviations
                ArrayList *c;

                double calcmeanval();
                double calcstd();
        public:
                __property double get_meanval();
                __property double get_std();
                __property int get_count();

                Block* get_Item(int index);

                Cluster(Block *block);
                void addBlock(Block *block);
        };
}

// Cluster Class implementations


#include "stdafx.h"
#include "Cluster.h"
#include "Block.h"

UnderVehicleSurv::Block* UnderVehicleSurv::Cluster::get_Item(int index)
{
        return dynamic_cast<Block *> (c->Item[index]);
}
double UnderVehicleSurv::Cluster::get_meanval()
{
```

```
        return m_meanval;
}

double UnderVehicleSurv::Cluster::get_std()
{
        return m_std;
}

int UnderVehicleSurv::Cluster::get_count()
{
        return c->Count;
}

double UnderVehicleSurv::Cluster::calcmeanval()
{
        double meanval=0;
        Block *b;
        for (int i=0; i< c->Count ; i++) {
                b = dynamic_cast<Block *> (c->Item[i]);
                meanval += b->meanval;
        }
        meanval = meanval/c->Count;
        this->m_meanval=meanval;
        return meanval;
}

double UnderVehicleSurv::Cluster::calcstd()
{
        double std=0;
        Block *b;
        for (int i=0; i<c->Count; i++) {
                b= dynamic_cast<Block *> (c->Item[i]);
```

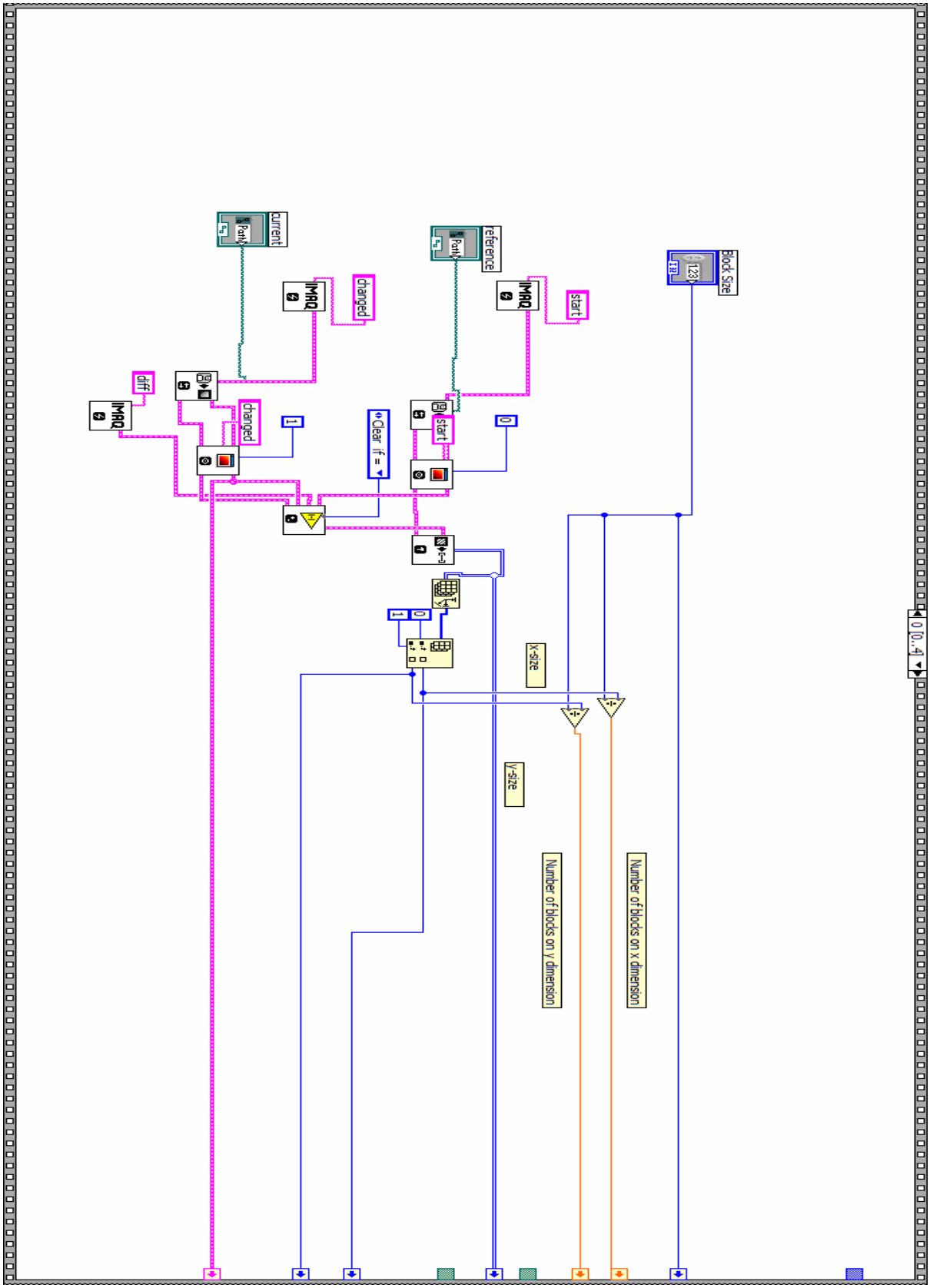
```
                std+=b->std;
        }
        std = std/c->Count;
        this->m_std=std;
        return std;
}

UnderVehicleSurv::Cluster::Cluster(Block *b)
{
        c = new ArrayList();
        c->Add(b);
        calcmeanval();
        calcstd();
}

void UnderVehicleSurv::Cluster::addBlock(Block *b)
{
        c->Add(b);
        calcmeanval();
        calcstd();
}

// Block based median filtering class declarations

#pragma once

using namespace System;

namespace UnderVehicleSurv
{
        public __gc class BbMedianFilter
        {
```

```
        private:
                Byte image[,];
                int sizex;
                int sizey;
        public:
                int median(int blocksize,Byte imageout[,]);
                BbMedianFilter(Byte image[,]);
                int neighbour(int x,int y);
        };
}

// BbMedianFilter class implemantations

#include "stdafx.h"
#include "BbMedianFilter.h"



UnderVehicleSurv::BbMedianFilter::BbMedianFilter(Byte image[,])
{
        this->image=image;
        sizex=image->GetLength(0);
        sizey=image->GetLength(1);
}

int UnderVehicleSurv::BbMedianFilter::neighbour(int x,int y)
{
        if ( (x >= 0) && (x < sizex) && (y >= 0) && (y < sizey) )
        {
                if (this->image[x,y]==255) {
                        return 1;
                } else {
```

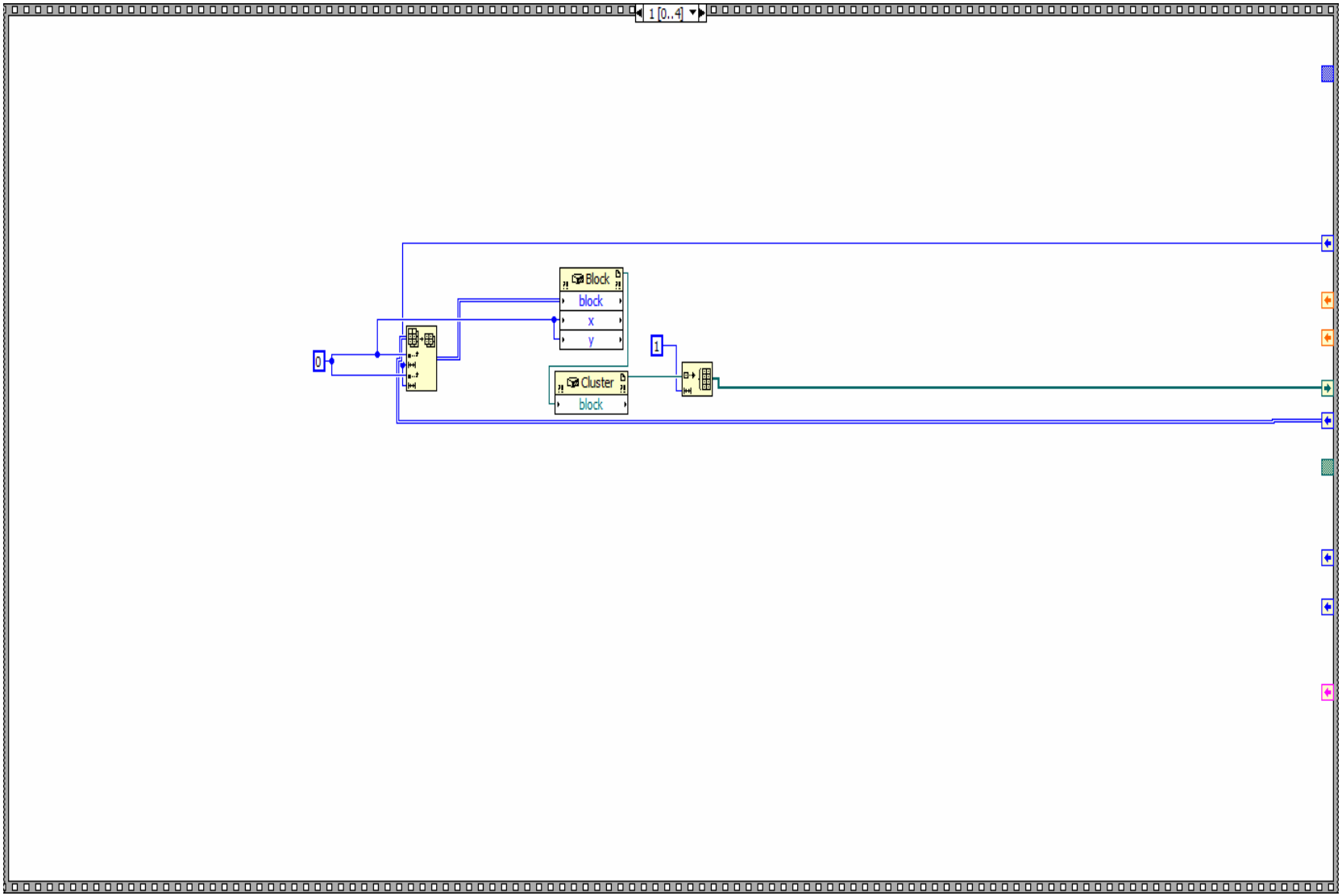
```
                return 0;
            }
        } else {
            return 0;
        }

}
int UnderVehicleSurv::BbMedianFilter::median(int blocksize,Byte imageout[,])
{
        register int i,j,k,l;

        for (i=0;i<sizex;i=i+blocksize)
            for (j=0;j<sizey;j=j+blocksize) {

                int count=0; //White blocks around

                count += neighbour(i-blocksize,j-blocksize);
                count += neighbour(i-blocksize,j);
                count += neighbour(i-blocksize,j+blocksize);
                count += neighbour(i,j-blocksize);
                count += neighbour(i,j+blocksize);
                count += neighbour(i+blocksize,j-blocksize);
                count += neighbour(i+blocksize,j);
                count += neighbour(i+blocksize,j+blocksize);


                for (k=i;k<i+blocksize;k++)
                    for (l=j;l<j+blocksize;l++) {
                        // in bounds... change image
                        if ( k < sizex && l < sizey ) {
                            // More than half neighbours are
white
```

```
                                        if ( count>=5 ) {
                                                imageout[k,l]=255;
                                        //or more than half are black
                                        } else {
                                                imageout[k,l]=0;
                                        }
                                }
                        }
                }
        return 0;
}
```

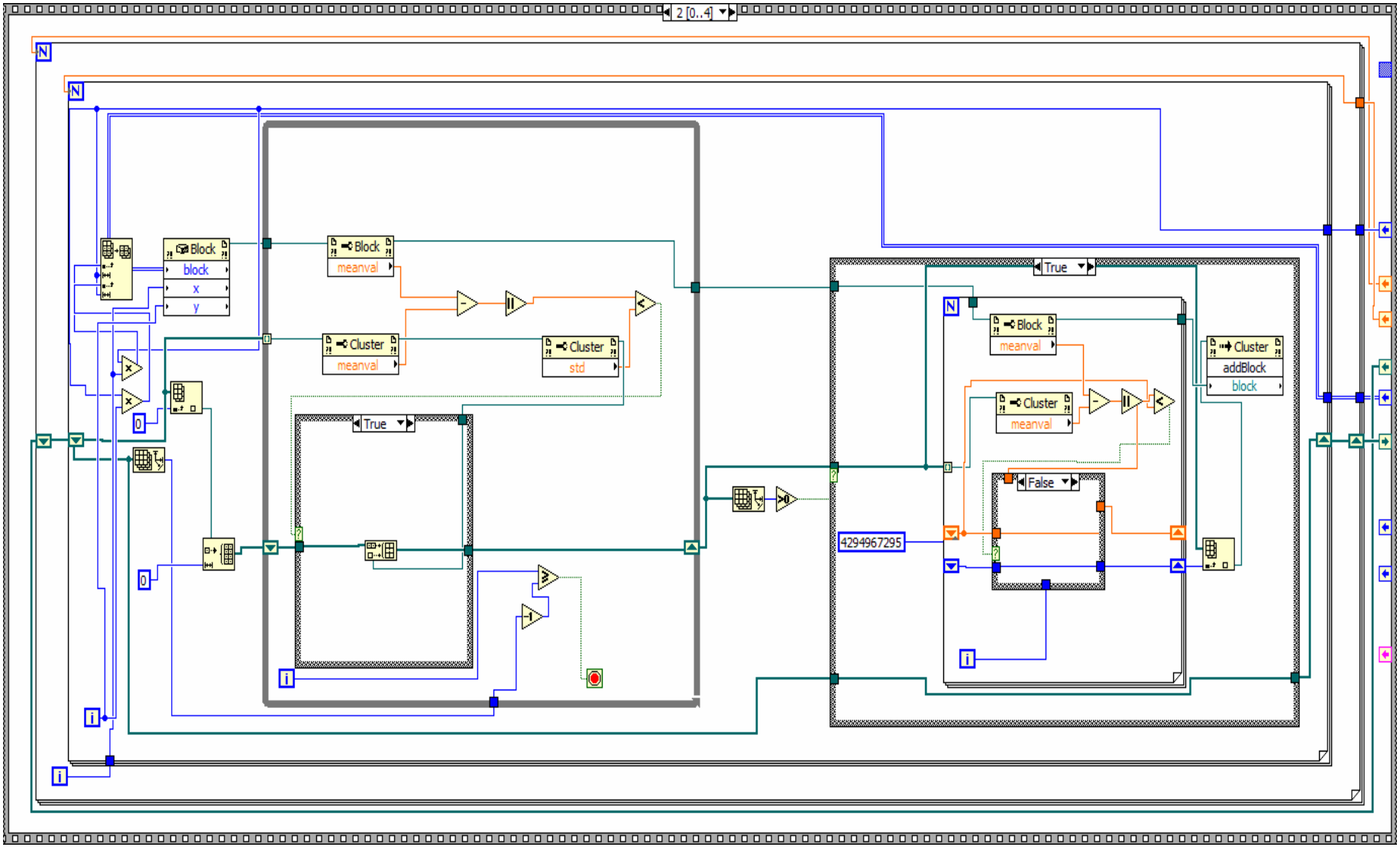
Παρακάτω παραθέτουμε τον σχηματικό κώδικα που υλοποιήσαμε στο πρόγραμμα Labview της National Instruments , με την ακολουθία πλαισίων που εκτελείται.



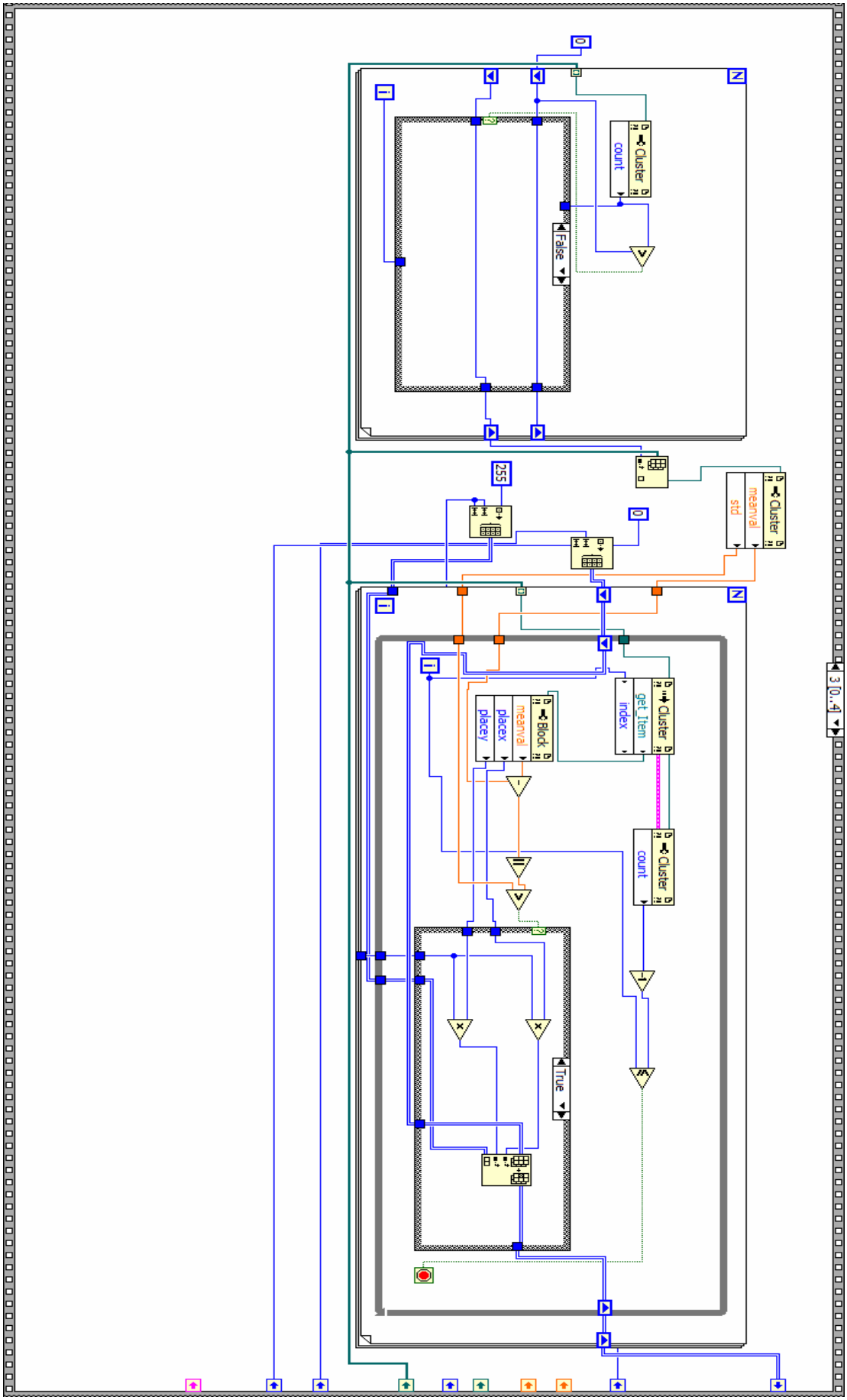
Πλαίσιο 1 του σχηματικού κώδικα Labview



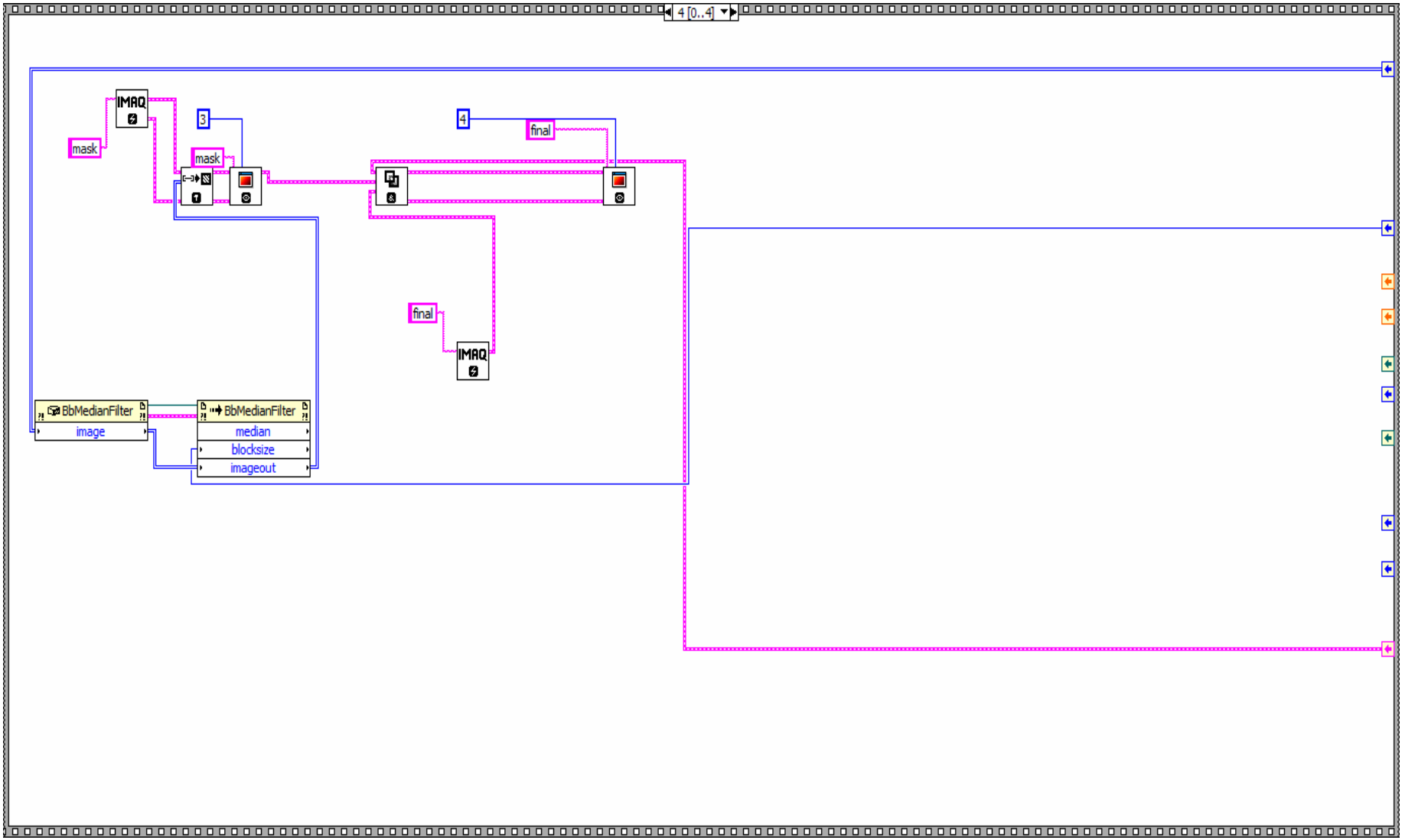
Πλαίσιο 2 του σχηματικού κώδικα Labview



Πλαίσιο 3 του σχηματικού κώδικα Labview



Πλαίσιο 4 του σχηματικού κώδικα Labview



Πλαίσιο 5 του σχηματικού κώδικα Labview

Μέρος Τέταρτο

Εισαγωγή

Σκοπός του τελευταίου μέρους της εργασίας είναι να παρουσιάσουμε την ενοποίηση των αλγορίθμων και να επιδείξουμε μία προτεινόμενη διάταξη υλοποίησης ενός συστήματος επίβλεψης κάτω όψης οχημάτων. Η διαδικασία προορίζεται για τον έλεγχο διαπιστευμένων οχημάτων σε χώρους ελεγχόμενης στάθμευσης όπως πρεσβείες, αστυνομικά τμήματα, κυβερνητικά κτήρια κ.α.

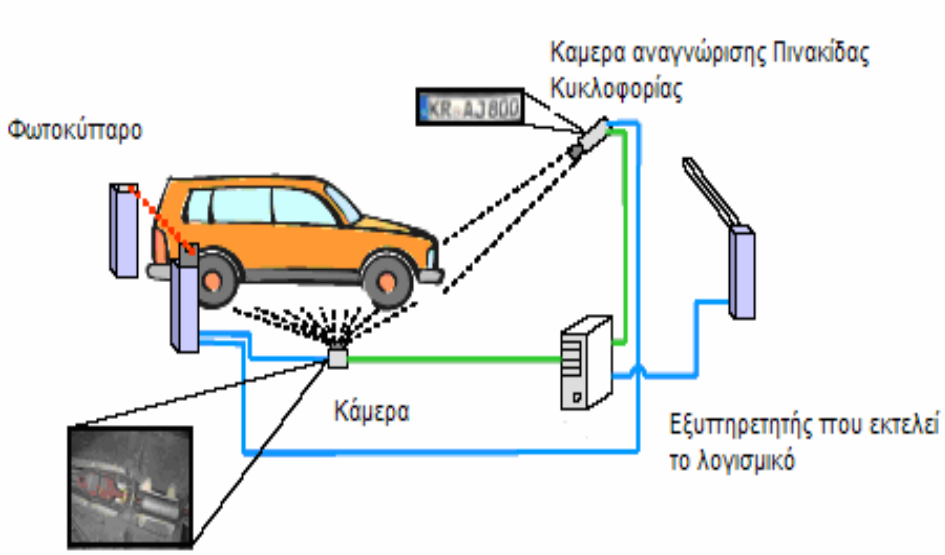
Ενοποίηση

Το πρώτο μέρος της εργασίας με την προσθήκη ενός αλγόριθμου αναγνώρισης χαρακτήρων (OCR) είναι χρησιμοποιείται για την ταυτοποίηση. Η υλοποίηση ενός τέτοιου αλγορίθμου δεν ήταν μέσα στα πλαίσια της εργασίας αλλά αναφέρουμε χάριν πληρότητας ότι υπάρχουν αρκετοί ήδη στο εμπόριο ενώ μία συχνή υλοποίηση τους είναι με χρήση νευρωνικών δικτύων ώστε να είναι σε θέση να μαθαίνουν νέους χαρακτήρες ή διαφορετικές μορφές παλιών και να αυξάνεται έτσι η χρηστικότητά τους.

Το επόμενο στάδιο περιλαμβάνει τη σάρωση της κάτω όψης του αμαξώματος με υψηλής ανάλυσης φωτογραφικό εξοπλισμό. Βάσει του αλγορίθμου ταυτοποίησης εικόνων που αναπτύξαμε στο δεύτερο μέρος της εργασίας παράγεται η απόλυτος διαφορά της εικόνας που λαμβάνεται από την τρέχουσα σάρωση, προφανώς μεταβλημένες όπως αναφέραμε, και της εικόνας που περιέχεται σε μία βάση δεδομένων και ανασύρεται από αυτήν μέσω της μοναδικής πινακίδας κυκλοφορίας. Η εν λόγω εικόνα έχει ληφθεί σε ελεγχόμενο χώρο, κατόπιν προσεκτικού ελέγχου του αμαξώματος του οχήματος

Κατόπιν της ταύτισης των δύο εικόνων, η απόλυτος διαφορά των ταυτοποιημένων εικόνων και η αρχική εικόνα που ανασύρουμε από την βάση δεδομένων τροφοδοτούνται στην υλοποίηση του τρίτου αλγόριθμου και τα αποτελέσματα εμφανίζονται σε μία οθόνη. Να επισημάνουμε εδώ η διαδικασία είναι σε θέση να ανιχνεύσει πιθανές αλλαγές στην κάτω όψη του αμαξώματος αλλά δεν είναι σε θέση να ξεχωρίσει την φύση τους. Κατά συνέπεια είναι απαραίτητη η ανθρώπινη παρουσία ώστε να αξιολογεί κατά περίπτωση τα αποτελέσματα του αλγορίθμου. Είναι παρόλα αυτά δυνατόν να αυτοματοποιηθεί η διαδικασία εισόδου στους ελεγχόμενους χώρους συνδέοντας το λογισμικό μας με κάποιο πιθανό εμπόδιο στο χώρο το οποίο εάν τα αποτελέσματα του αλγορίθμου είναι αρνητικά να αφαιρείται. Αντίθετα σε περίπτωση θετικών αποτελεσμάτων να ειδοποιείται άμεσα ο υπεύθυνος.

Παρακάτω παρουσιάζουμε μία ενδεικτική κατασκευή που μπορεί να υλοποιηθεί για την χρήση του συστήματος.



Το φωτοκύτταρο χρησιμοποιείται ώστε να γνωρίζει το σύστημα πότε να ξεκινήσει την εκτέλεση της διαδικασίας. Αυτό στέλνει το κατάλληλο σήμα στον κεντρικό εξυπηρετητή ο οποίος με την σειρά του θέτει σε λειτουργία τις δύο κάμερες ταυτόχρονα για εξοικονόμηση χρόνου. Η κάμερα αναγνώρισης

πινακίδας κυκλοφορίας επιστρέφει την εικόνα που χρησιμοποιείται για την ταυτοποίηση του οχήματος. Σε περίπτωση που δεν υπάρξει ταυτοποίηση ειδοποιεί τον χειριστή ειδάλλως ξεκινά τα επόμενα βήματα της διαδικασίας. Η δεύτερη κάμερα είναι μία γραμμική κάμερα υψηλής ανάλυσης που σαρώνει την κάτω όψη του οχήματος. Η έξοδος της τροφοδοτείται στον εξυπηρετητή ο οποίος εκτελεί το λογισμικό που περιγράψαμε και αναλόγως επιτρέπει την πρόσβαση στο όχημα ή ειδοποιεί τον χειριστή.

Συμπεράσματα

Παρουσιάστηκε μία μελέτη ενός αλγορίθμου ελέγχου κάτω όψης οχημάτων. Ο αλγόριθμος έχει τρία στάδια. Αρχικά υπάρχει ένας αλγόριθμος εντοπισμού της πινακίδας κυκλοφορίας ενός οχήματος από μία εικόνα που έχει ληφθεί με κάμερα. Κατόπιν παρουσιάζεται ένας αλγόριθμος ταυτοποίησης της αρχικής εικόνας, που έχουμε στα αρχεία μας, και της τρέχουσας εικόνας, που λαμβάνεται από την κάμερα. Αυτό επιτυγχάνεται μέσω μία τεχνικής χρησιμοποίησης της κανονικοποιημένης ετεροσυσχέτισης και της χρήσης wavelets, ώστε να μειωθεί το κόστος υπολογισμού. Το τελικό προϊόν της προηγούμενης μεθόδου τροφοδοτείται στο τρίτο μέρος σε ένα αλγόριθμο εύρεσης διαφορών, ο οποίος, όπως είδαμε, είναι αρκετά ανθεκτικός σε μεγάλες αλλαγές περιεχομένου καθώς και σε μεγάλες ποσότητες θορύβου. Η ανοσία στο θόρυβο επιτυγχάνεται μέσω μίας μεθόδου στατιστικής προσέγγισης των χαρακτηριστικών του θορύβου, με χρήση ομάδων στοιχείων εικόνας.

Το σύστημα ελέγχθηκε με διάφορες εικόνες δείγματα και απέδειξε ότι έχει καλή απόδοση τόσο στη μελετώμενη περίπτωση όσο και σε περιπτώσεις εισόδων διαφορετικών της υπό μελέτης περίπτωσής μας.

Βιβλιογραφία

- [1] Anagnostopoulos Christos, D. Vergados, E. Kayafas, V. Loumos, and G.Stassinopoulos, ”A computer vision approach for textile quality control”, Journal of Visualization and Computer Animation, 2001, Vol.12, No.1, 31-44
- [2] Αναγνωστόπουλος Χρήστος, Τεχνικές Υπολογιστικής Ορασης και Νοημοσύνης για Εξειδικευμένες Εφαρμογές και Ποιοτικό Ελεγχο στην Βιομηχανία, Ε.Μ.Π.,2002
- [3] Gottesfeld Brown Lisa, A survey of Image Registration Techniques, ACM Computing Surveys (CSUR), 1992, Volume 24, Issue 4, 325-376, ISSN: 0360-0300
- [4] Μαραγκός Πέτρος, Οραση Υπολογιστών, Ε.Μ.Π., 2002
- [5] Graps Amara, 1995, An Introduction to Wavelets, IEEE Computational Science and Engineering, Summer 1995, vol. 2, num. 2
- [6] Walker S. James, A primer on wavelets and their scientific applications,Chapman & Hall / CRC, 1999
- [7] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989, Vol. 2, No. 7, 674-693
- [8] Prachya Chalermwat, "HIGH PERFORMANCE AUTOMATIC IMAGE REGISTRATION FOR REMOTE SENSING", PhD Thesis, Fall 1999, George Mason University, Department of Computational Sciences and Computer Engineering
- [9] T. Alexandropoulos, S. Boutas, V. Loumos and E. Kayafas, "Real-time change detection for surveillance in public transportation", IEEE International Conference on Advanced Video and Signal-Based Surveillance /, Como, Italy, September 15-16, 2005, pp 58-63, IEEE Catalog Number: 05EX1166, ISBN: 0-7803-9385-6
- [10] Alexandropoulos Theodoros, Vassilis Loumos, Eleftherios Kayafas, “Block-based change detection in the presence of ambient

illumination variations", Journal of Advanced Computational Intelligence and Intelligent Informatics, 2005, Vol. 9

- [11] Alexandropoulos Theodoros, Sotirios Boutas, Vassilis Loumos and Eleutherios Kayafas, "Template-guided inspection of arbitrarily oriented targets", 2005
- [12] Alexandropoulos Theodoros, Vassilis Loumos and Eleutherios Kayafas , "A block clustering technique for real-time object detection on a static background", 2nd International IEEE Conference on Intelligent Systems, Vol. 1, pp. 169-173, 2004.