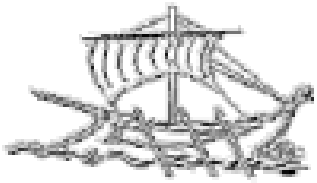


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επεξεργασία εικόνας με Visual Studio

Μαρία Βιντράσκο

Εισηγητής: Ιωάννης Ν. Έλληνας, Καθηγητής

ΑΘΗΝΑ
ΙΟΥΛΙΟΣ 2017

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επεξεργασία εικόνας με Visual Studio

Μαρία Βιντράσκο
Α.Μ. 42866

Εισηγητής:

Ιωάννης Ν. Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος/η Μαρία Βιντράσκο, του Αντρέι, με αριθμό μητρώου 42866 φοιτητής/τρια του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της επεξεργασίας εικόνας. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, Ιωάννης Ν. Έλληνας, τον οποίο θα ήθελα να ευχαριστήσω, για την πολύτιμη βοήθεια και καθοδήγηση του, για την επίλυση διάφορων θεμάτων.

Ακόμα θα ήθελα να ευχαριστήσω τη μητέρα μου για όλα όσα μου έχει προσφέρει στη διάρκεια των μαθητικών και φοιτητικών μου χρόνων, την στήριξη και την εμπιστοσύνη της σε όλα όσα κάνω.

ΠΕΡΙΛΗΨΗ

Η συγκεκριμένη πτυχιακή εργασία έχει στόχο να κάνει κατανοητή την έννοια και τη λειτουργία του περιβάλλοντος προγραμματισμού Visual Studio και της βιβλιοθήκης OpenCV στον κλάδο της επεξεργασίας εικόνας. Αρχικά, παρουσιάζεται μια εισαγωγή που περιλαμβάνει πληροφορίες για το τι είναι αυτό το περιβάλλον και η βιβλιοθήκη, αλλά παρουσιάζεται και η δομή, τα πλεονεκτήματα, τα πεδία εφαρμογής της δεύτερης. Στη συνέχεια, γίνεται αναφορά στους τύπους των εικόνων και σε κάποιες βασικές εντολές και συναρτήσεις που χρησιμοποιεί η C++ για λειτουργίες όπως ανάγνωση κ.ά. Παρόμοια αναφορά πραγματοποιείται και για τα εικονοστοιχεία. Έπειτα στην ίδια ενότητα επεξεργασίας εικόνας με χρήση της βιβλιοθήκης OpenCV, παρατίθεται ο ορισμός του ιστογράμματος και οι τρόποι υλοποίησης διάφορων λειτουργιών της. Επιπλέον, αναλύονται οι μέθοδοι με τις οποίες πραγματοποιείται η βελτιστοποίηση της εικόνας, που είναι οι τεχνικές σημείου (π.χ. επέκταση αντίθεσης) και οι τεχνικές χώρου (π.χ. φίλτρο μέσης τιμής). Επιπροσθέτως, υπάρχει και η υποενότητα που επικεντρώνεται στην αποκατάσταση εικόνας (π.χ. απαλοιφή θορύβου). Τέλος, είναι σημαντικό να αναφερθεί πως σε κάθε υποενότητα, όπου κρίνεται αναγκαίο, αναγράφονται παραδείγματα για κάθε περίπτωση ξεχωριστά. Όλοι οι κώδικες είναι γραμμένοι στη γλώσσα C++, η οποία γλώσσα χρησιμοποιείται στις σύγχρονες εκδόσεις της OpenCV.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Επεξεργασία Εικόνας

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: OpenCV, τεχνικές σημείου, τεχνικές χώρου, αποκατάσταση εικόνας, C++

ABSTRACT

The specific dissertation aims at understanding the meaning and operation of Visual Studio programming environment and OpenCV library in the digital image processing. Firstly, there is a presentation of an introduction that includes information what this environment and library are, but also presentation of the structure, the advantages, and the fields of application of this library. Then, there is a reference to the types of images and to basic commands and functions, which C ++ uses for functions such as reading, etc. A similar reference is made for pixels. Furthermore, in the same section about image processing by using OpenCV library, histogram definition and ways of implement its various functions are listed. In addition, the methods through which it accomplishes optimization, that are point operations (for example contrast stretching) and spatial operations (for example averaging filter). Additionally, there is a sub-section that focuses on image “recovery” (for example, noise abatement). Finally, it is important to mention that in each subsection, where necessary, examples are given about each case separately. All the codes are written in C++, language which is used in latest versions of OpenCV.

SCIENTIFIC AREA: Digital image processing
KEYWORDS: OpenCV, point operations, spatial operations, image “recovery”, C++

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Σκοπός της παρούσας πτυχιακής εργασίας είναι η υλοποίηση και η εφαρμογή αλγορίθμων επεξεργασίας εικόνας με τη βοήθεια της πλατφόρμας Visual Studio και με χρήση της βιβλιοθήκης OpenCV. Συγκεκριμένα, θα αναπτυχθούν αλγόριθμοι:

- Χειρισμού εικόνων, δηλαδή ανάγνωσης, εμφάνισης και αποθήκευσης εικόνων, χειρισμού εικονοστοιχείων, εμφάνισης χαρακτηριστικών εικόνων και υπολογισμού ιστογραμμάτων.
- Βελτιστοποίησης εικόνων, δηλαδή μεθόδους τροποποίησης της αντίθεσης και εφαρμογή φίλτρων με σκοπό τον εμπλουτισμό και την ανάδειξη λεπτομερειών.
- Αποκατάστασης εικόνων, δηλαδή εφαρμογή φίλτρων για την εξάλειψη του ανεπιθύμητου θορύβου.

Παρακάτω παρατίθεται μια σύντομη περιγραφή του περιεχομένου των κεφαλαίων της πτυχιακής εργασίας:

Στο **Κεφάλαιο 2** παρουσιάζεται η επιστήμη της επεξεργασίας εικόνας. Γίνεται αναφορά στην ψηφιακή εικόνα και στα είδη της, στον ορισμό της επεξεργασίας εικόνας, σε ορισμένα ιστορικά γεγονότα και στα πεδία εφαρμογής της.

Στο **Κεφάλαιο 3** παρουσιάζεται αναλυτικά η βιβλιοθήκη OpenCV και συνοπτικά το περιβάλλον του Visual Studio.

Στο **Κεφάλαιο 4** παρουσιάζονται και εφαρμόζονται οι αλγόριθμοι επεξεργασίας εικόνων με χρήση της OpenCV.

Στο Κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα της παρούσας πτυχιακής εργασίας.

Επίσης, στο **Παράρτημα** παρουσιάζεται η εγκατάσταση της OpenCV και η σύνδεση της με το Visual Studio 2012 Express.

ΚΕΦΑΛΑΙΟ 2

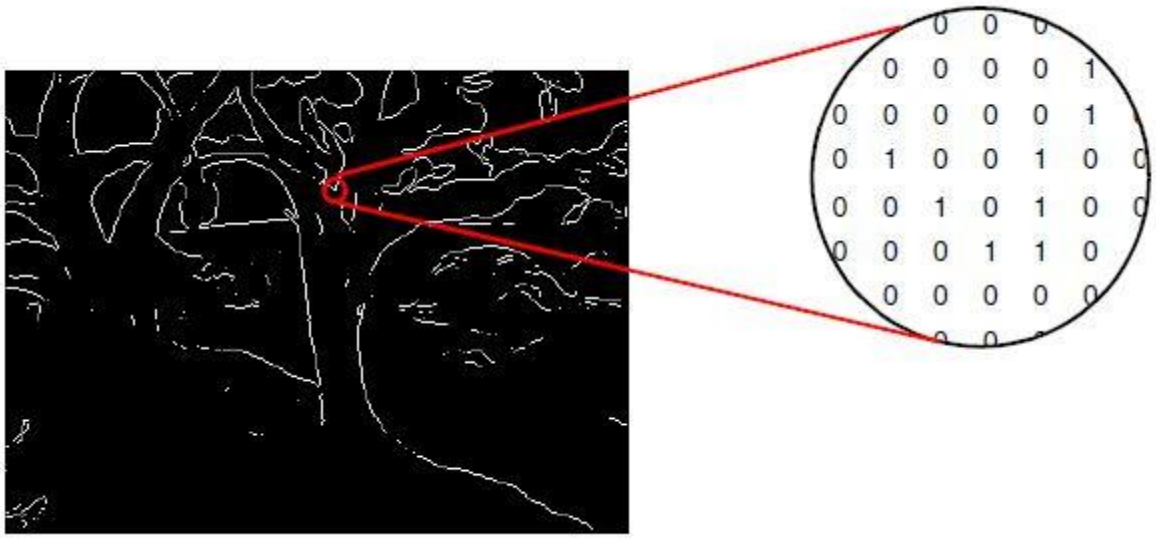
ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της πτυχιακής εργασίας, το οποίο είναι η επεξεργασία εικόνας. Συγκεκριμένα γίνεται αναφορά στον ορισμό της, στην ιστορική αναδρομή και στα πεδία εφαρμογής της.

2.1 Ορισμός της ψηφιακής εικόνας

Ψηφιακή εικόνα είναι η αριθμητική αναπαράσταση μιας δυαδικής εικόνας (binary image), η οποία ουσιαστικά είναι ένας δισδιάστατος πίνακας I(i,j). Ο πίνακας αυτός εικονοστοιχείων έχει διαστάσεις M στήλες × N γραμμές, με τιμές του i και j να κυμαίνονται από το 1 μέχρι το M και από το 1 μέχρι το N αντίστοιχα. Αυτές, αλλάζουν με βάση το είδος της εικόνας και αντιστοιχούν στη φωτεινότητα του φωτιζόμενου αντικειμένου. **[6] [12]**

Υπάρχουν τρεις τύποι ψηφιακής εικόνας : δυαδική (binary), μονοχρωματική αποχρώσεων του γκρι (grayscale) και έγχρωμη (color). Αναλυτικότερα, η **δυαδική** μορφή θεωρείται η πιο απλή, επειδή έχει μόνο δύο τιμές για κάθε εικονοστοιχείο (pixel), συνεπώς μικρότερη μνήμη και κόστος. Συνήθως αυτές οι τιμές αντιστοιχούν στα χρώματα μαύρο (τιμή 0) και άσπρο (τιμή 1). **[1]**

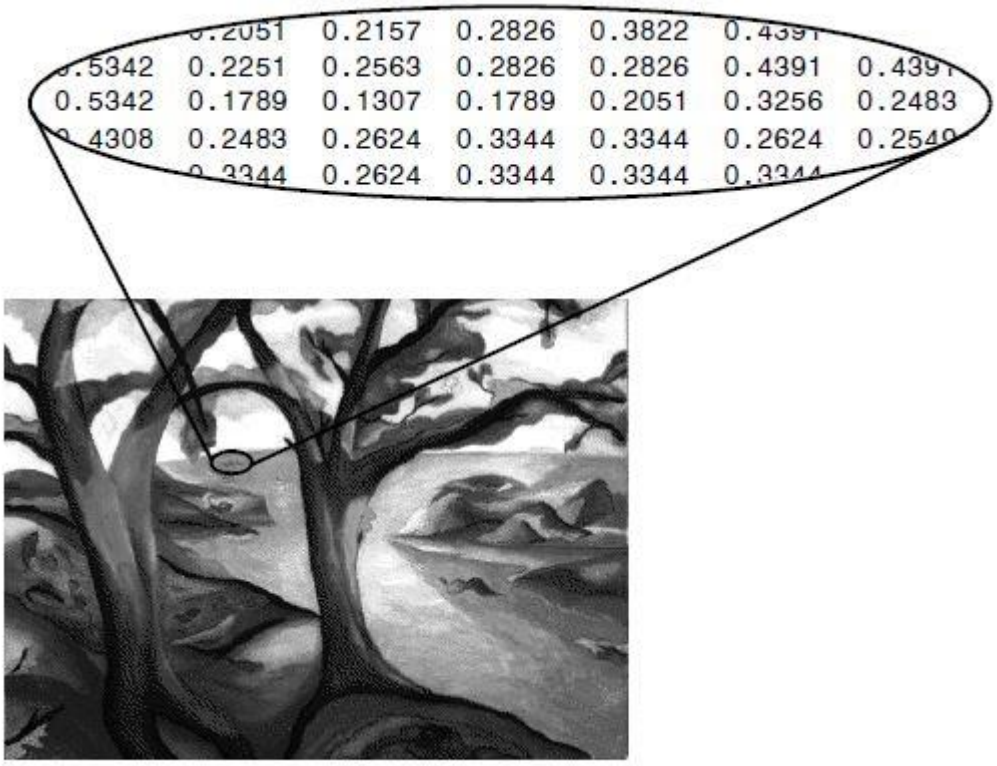


Εικόνα 2.1: Η δυαδική εικόνα (binary image) **[14]**

Η **μονοχρωματική** εικόνα είναι ένας δισδιάστατος πίνακας ακεραίων (M×N) I(i,j) με

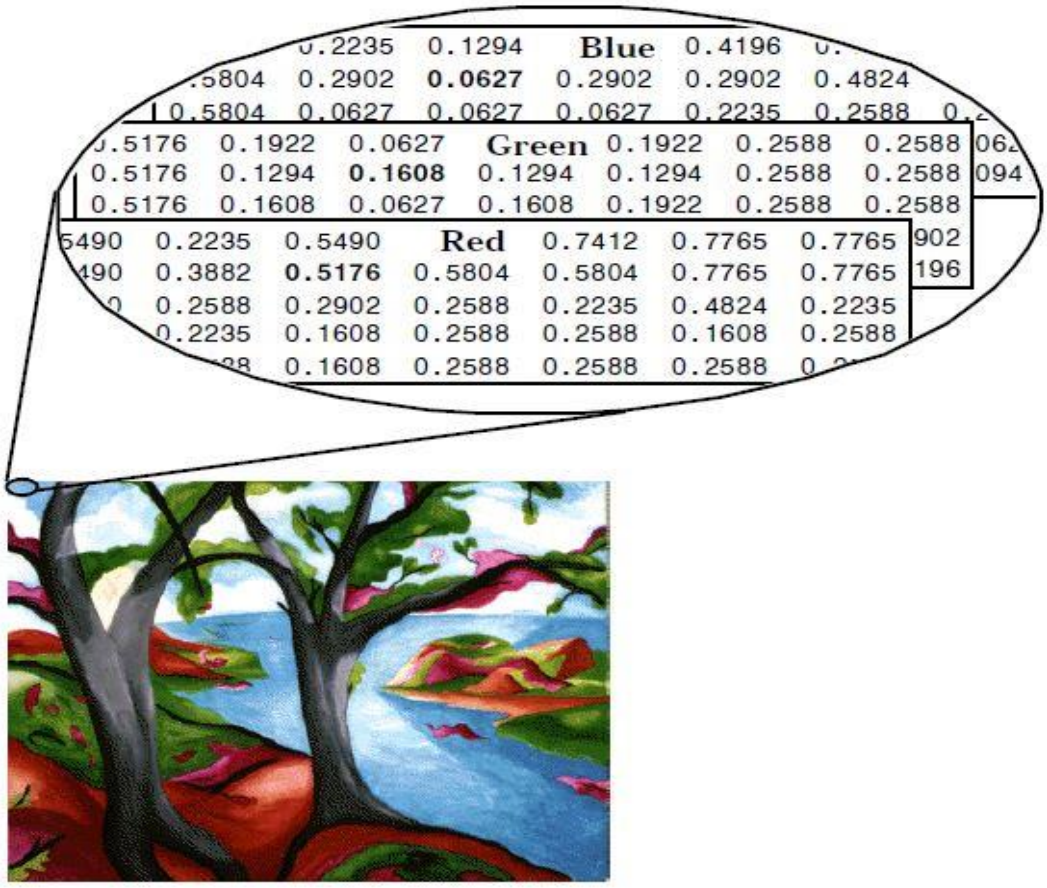
i = 1,….,M και j = 1,…,N όπου, 0≤ I(i,j)≤G-1 (G είναι συνήθως μια δύναμη του 2)

Για παράδειγμα για μια εικόνα με 256 αποχρώσεις, οι τιμές που θα πάρει είναι από το 0-255. **[1] [6]**



Εικόνα 2.2: Η εικόνα φωτεινότητας (grayscale image) **[14]**

Η **έγχρωμη**, είναι αυτός ο τύπος της εικόνας που περιέχει πληροφορίες χρωμάτων για κάθε ένα από τα pixels, αυτό μεταφράζεται σε 3 διαφορετικές grayscale εικόνες-πίνακες οι οποίες βρίσκονται σε ένα χρωματικό σύστημα που ονομάζεται RGB και έχουν ίδιες διαστάσεις με αυτές της εικόνας, οπότε είναι λογικό να θεωρείται ως ο καλύτερος τρόπος αναπαράστασης της πραγματικότητας. Οπότε, αν η τελική εικόνα-πίνακας είναι I(i,j), τότε οι 3 πίνακες είναι $I(i,j)_1$, $I(i,j)_2$, $I(i,j)_3$ όπου, $0\leq I(i,j)_1, I(i,j)_2, I(i,j)_3 \leq G-1$. **[6] [13]**



Εικόνα 2.3: Η έγχρωμη εικόνα (color image) **[14]**

2.2 Ορισμός της επεξεργασίας εικόνας

Η **επεξεργασία εικόνας** είναι ο κλάδος της πληροφορικής ο οποίος ασχολείται με την επεξεργασία, την ανάλυση, και τον χειρισμό εικόνων με χρήση ηλεκτρονικού υπολογιστή. Υπάρχουν διάφοροι τύποι επεξεργασίας, όπως οι χρωματικές μετατροπές (αλλαγή χρωματικού μοντέλου, ρύθμιση φωτεινότητας και αντίθεσης), γεωμετρικές μετατροπές (αλλαγή μεγέθους, ανάλυσης εικόνας, περιστροφή κτλ.), εφαρμογή φίλτρων (μείωση θορύβου, βελτίωση της ποιότητας της εικόνας), συμπίεση εικόνας, αποκατάσταση εικόνας και άλλοι. **[15]**

2.3 Ιστορική αναδρομή

Στις αρχές της δεκαετίας του 1920, παρατηρείται η πρώτη εφαρμογή της επεξεργασίας εικόνας στη βιομηχανία των εφημερίδων. Εκείνη την εποχή, η μεταφορά των εικόνων πραγματοποιούταν μέσω υποθαλάσσιου καλωδίου, που συνέδεε το Λονδίνο με την Νέα Υόρκη, και διαρκούσε περισσότερο από μια

εβδομάδα . Το 1921 έχουμε τη μετάδοση εικόνας (βλέπε εικόνα 2.) με χρήση του συστήματος Bartlane, μειώνοντας τη διάρκεια μεταφοράς σε 3 ώρες. Η εικόνα αυτή αποτελούταν από 5 αποχρώσεις του γκρι. Το 1929 έχουμε τη μετάδοση εικόνας (βλέπε εικόνα 2.), αυτή τη φορά με χρήση τεχνικών φωτογραφικής αναπαραγωγής, αυξάνοντας τις αποχρώσεις του γκρι σε 15. **[2]**

Εικόνα 2.4: Η εικόνα που μεταδόθηκε με το σύστημα Bartlane **[2]**

Εικόνα 2.5: Η εικόνα που μεταδόθηκε με τεχνικές φωτογραφικής αναπαραγωγής **[2]**

Από τη δεκαετία του 1960, η επεξεργασία εικόνας ξεκινά να χρησιμοποιείται σε εφαρμογές που σχετίζονται με το διάστημα, την ιατρική και την αστρονομία. Συγκεκριμένα το 1964 χρησιμοποιούνται υπολογιστές για την επεξεργασία φωτογραφιών του φεγγαριού που είχαν ληφθεί μέσω φωτογραφικής μηχανής, ενώ το 1970 έχουμε την εφεύρεση του αξονικού τομογράφου. **[2]**

Το 1994, το Space Foundation του Κολοράντο εγκαθιστά τεχνολογία επεξεργασίας εικόνας για χρήση σε ιατρικές εφαρμογές. Η δεκαετία του 2000, χαρακτηρίζεται από τη ψηφιακή επεξεργασία εικόνας ως τη πιο κοινή μορφή επεξεργασίας εικόνας και γενικά χρησιμοποιείται επειδή δεν είναι μόνο η πιο ευέλικτη μέθοδος αλλά και η φθηνότερη. **[17]**

2.4 Πεδία εφαρμογής της επεξεργασίας εικόνας

Η επεξεργασία εικόνας βρίσκει εφαρμογή σε αρκετούς κλάδους και επιστημονικά πεδία. Συγκεκριμένα, στη γραφιστική, στην ιατρική, στη ρομποτική, στη φωτογραφία, στην τεχνητή νοημοσύνη, στα συστήματα ασφαλείας, στην αναγνώριση προτύπων, στις αεροφωτογραφίες, στη μετάδοση εικόνων κτλ. **[1]**

ΚΕΦΑΛΑΙΟ 3

ΒΙΒΛΙΟΘΗΚΗ OPENCV – ΠΕΡΙΒΑΛΛΟΝ VISUAL STUDIO

Στο συγκεκριμένο κεφάλαιο αναλύεται η βιβλιοθήκη και το περιβάλλον, που θα χρησιμοποιηθεί για τις εφαρμογές στην επεξεργασία εικόνας. Στις παρακάτω ενότητες γίνεται αναφορά στην έννοια της OpenCV, στην ιστορική αναδρομή, στη δομή της, στα πεδία εφαρμογής της, στα πλεονεκτήματα της και τι είναι το Visual Studio.

3.1 Τι είναι βιβλιοθήκη (στον τομέα της πληροφορικής)

Βιβλιοθήκη (library) ονομάζεται μια συλλογή από έτοιμα υποπρογράμματα (όπως δεδομένα βοήθειας, πρότυπα μηνυμάτων, προγραμμένο κώδικα, κλάσεις) που χρησιμοποιούνται από τους ηλεκτρονικούς υπολογιστές με σκοπό την ανάπτυξη λογισμικού. Ακόμη θεωρείται μια συλλογή εφαρμογών συμπεριφοράς, η οποία χρησιμοποιείται ξανά και ξανά από πολλαπλά διαμοιρασμένα προγράμματα ή υποπρογράμματα, ο χρήστης χρειάζεται να γνωρίζει μόνο τη διεπαφή και έτσι μπορεί να χειρίζεται τον κώδικα με αρθρωτό τρόπο. Υπάρχουν 2 είδη βιβλιοθηκών, η στατική (ο κώδικας είναι προσπελάσιμος κατά τη διάρκεια τη δημιουργία του προγράμματος) και η δυναμική (η συμπεριφορά αυτής αποτελεί ή μέσο εκτέλεσης ή μέρος της διαδικασίας έναρξης της εκτέλεσης). Συνεπώς, οι περισσότεροι κώδικες που χρησιμοποιούνται από σύγχρονες εφαρμογές υπάρχουν σε αυτές τις βιβλιοθήκες συστήματος και έτσι δεν χρειάζεται να δημιουργηθεί για κάθε νέο πρόγραμμα. **[7] [18]**

3.2 Τι είναι η βιβλιοθήκη OpenCV

Η **OpenCV** (Open Source Computer Vision) είναι πιο δημοφιλής βιβλιοθήκη ανοιχτού κώδικα στον κόσμο με περισσότερους από 2500 βελτιστοποιημένους αλγόριθμους, που ασχολείται κυρίως με την υπολογιστική όραση σε πραγματικό χρόνο. Ακόμα, διαθέτει βιβλιοθήκη μηχανικής μάθησης που αποτελούν εργαλεία για ομαδοποίηση και στατιστική ταξινόμηση, καθώς και συναρτήσεις για τη προβολή και την αποθήκευση των βίντεο και γενικά ότι έχει να κάνει με τη διεπαφή χρήστη-εφαρμογής. Το ερευνητικό κέντρο της Intel στη Ρωσία είναι αυτό

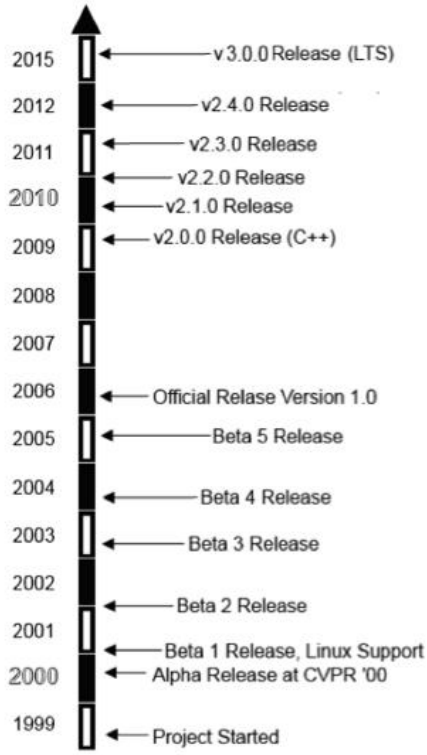
που την ανέπτυξε πρώτο. Έπειτα υποστηρίχθηκε από το Willow Garage και τώρα από το μη κερδοσκοπικό ίδρυμα OpenCV.org. Η OpenCV είναι πολλαπλής πλατφόρμας και διανέμεται δωρεάν με άδεια BSD για εμπορικούς ή ερευνητικούς σκοπούς. Αυτό σημαίνει πως διευκολύνει τις επιχειρήσεις, τις ερευνητικές ομάδες και τους κυβερνητικούς φορείς να χρησιμοποιούν και να τροποποιούν τον κώδικα χωρίς να τον αποκαλύψουν. Είναι σημαντικό να αναφερθεί ότι υπάρχουν μερικοί αλγόριθμοι με πλήρες πηγαίο κώδικα, όπως για παράδειγμα ο SURF, οι οποίοι είναι ελεύθεροι μόνο για ακαδημαϊκή και όχι για εμπορική χρήση. **[7] [8] [9] [19] [20] [22]**

Εικόνα 3.1: OpenCV Logo **[7]**

3.3 Ιστορική αναδρομή

Όλα ξεκίνησαν το 1999, όταν με πρωτοβουλία της Intel Research και με επικεφαλής τον Gary Bradski (μιας ομάδας εμπειρογνωμόνων της Intel Ρωσία και της Intel Performance Library) διεξήχθη μια έρευνα, που αφορούσε την προώθηση εφαρμογών με υψηλές απαιτήσεις από τον επεξεργαστή του συστήματος. Έτσι, αναπτύχθηκε η OpenCV με συναρτήσεις σε C και διάφορες κλάσεις σε C++. Το 2000 ανακοινώθηκε η πρώτη «alpha» έκδοση στο συνέδριο Computer Vision and Pattern Recognition και στο χρονικό διάστημα 2001-2005 πέντε «beta». Το 2006 κυκλοφόρησε η πρώτη 1.0 έκδοση και το 2008 η έκδοση 1.1. , στα μέσα της ίδιας χρονιάς την υποστήριξη της OpenCV ανέλαβαν οι εταιρίες WillowGarage και Itseez. Τον Οκτώβριο του 2009, έγινε η δεύτερη μεγαλύτερη αλλαγή και κυκλοφορία στη συγκεκριμένη βιβλιοθήκη, αφού βελτιστοποιήθηκαν διάφορες υλοποιήσεις και λειτουργίες. Τον Αύγουστο του

2012, το μη κερδοσκοπικό ίδρυμα OpenCV.org ανέλαβε την υποστήριξη και ανάπτυξη της OpenCV. Πλέον μετράει πάνω από 47.000 χρήστες και ο εκτιμώμενος αριθμός λήψεων υπερβαίνει τα 14 εκατομμύρια. Η τελευταία της έκδοση, 3.2.0, μπορεί να βρεθεί στο SourceForge. **[9] [10] [19] [22]**



Σχήμα 3.2: Χρονοδιάγραμμα της OpenCV **[24]**

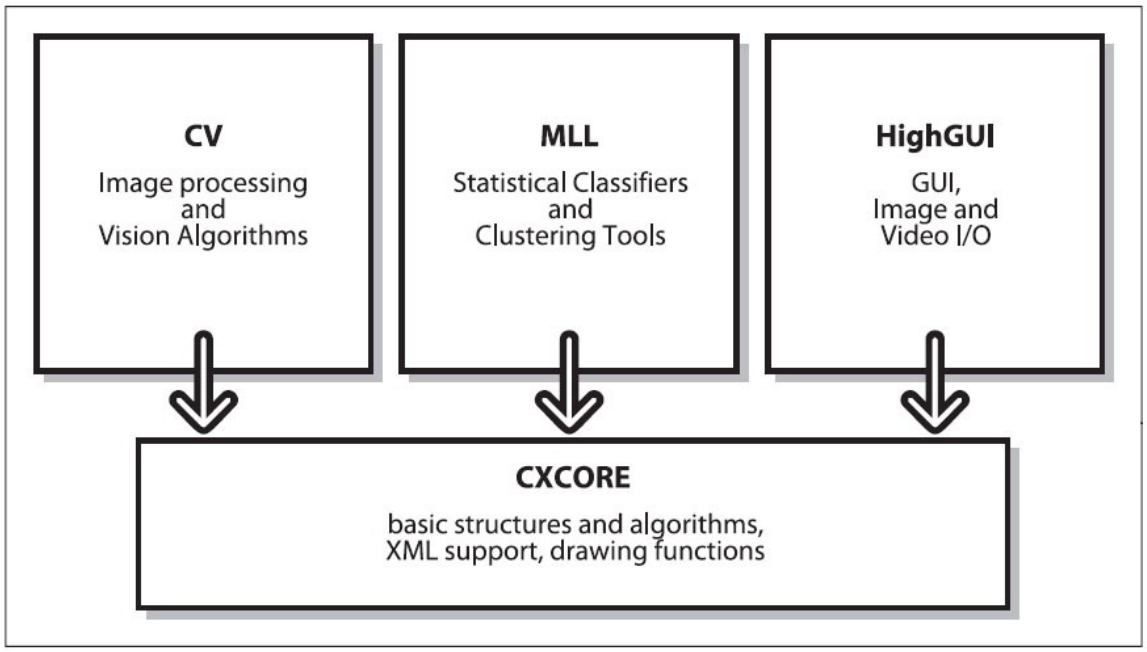
3.4 Διασυνδέσεις και υποστηριζόμενα λειτουργικά συστήματα της OpenCV

Η OpenCV διαθέτει τις διεπαφές C ++, C, Python, Ruby, Java, Matlab / Octave και είναι διαθέσιμη στους υπολογιστές σε Windows, Linux, Mac OS, FreeBSD, NetBSD, OpenBSD και σε κινητά σε iOS, Maemo, BlackBerry. Το API για αυτές τις διεπαφές μπορεί να βρεθεί στην ηλεκτρονική τεκμηρίωση. Κάθε καινούρια έκδοση κυκλοφορεί κάθε 6 μήνες και υπάρχει συνεχής ενημέρωση. Όλες αυτές οι εκδόσεις και οι αλγόριθμοι τους είναι γραμμένοι σε C ++ και αναπτύσσονται κυρίως στη διασύνδεση C ++. **[10] [11] [19] [20]**

3.5 Δομή της OpenCV

Όπως απεικονίζεται στο παρακάτω σχήμα η βιβλιοθήκη OpenCV αποτελείται κυρίως από τέσσερα συστατικά μέρη, το CV, το MLL, το HighGUI και το CxCore.

Αναλυτικότερα, το πρώτο συστατικό μέρος (CV), αναπαριστά τους βασικούς αλγορίθμους για επεξεργασία εικόνας και τους υψηλού επιπέδου για τη μηχανική όραση. Αυτό σημαίνει, όπως συναρτήσεις ομαδοποίησης, βαθμονόμησης κάμερας. Το δεύτερο, που ονομάζεται MLL, περιέχει τη εκτάσιμη βιβλιοθήκη ρουτινών μηχανικής μάθησης, δηλαδή είναι εργαλεία για ομαδοποίηση και για στατιστική ταξινόμηση, εύκολα και αποδοτικά στη χρήση. Αυτό που τα χαρακτηρίζει είναι η παράλληλη εκτέλεση σε ένα σύμπλεγμα. Το τρίτο, το οποίο έχει την ονομασία, HighGUI, περιλαμβάνει τις συναρτήσεις, τις ρουτίνες εισόδου-εξόδου, τα οποία αποσκοπούν για την προβολή και την αποθήκευση των εικόνων και ότι έχει να κάνει με τη δημιουργία γραφικών διεπαφών χρήστη για ανάγνωση και εγγραφή εικόνων. Τέλος, το CxCore, παρέχει ένα σύνολο βασικών δομών δεδομένων και τις λειτουργίες σχεδίασης γραμμών, κύκλων κ.λπ. στις εικόνες. Μια από τις δομές αυτές είναι η λεγόμενη IplImage, θεωρείται βασική δομή στην OpenCV και περιγράφει με λεπτομέρεια το κάθε καρέ προς επεξεργασία. **[10] [25]**



Σχήμα 3.3: Συστατικά μέρη της OpenCV **[10]**

3.6 Πλεονεκτήματα της OpenCV

Παρακάτω γίνεται η αρίθμηση των πλεονεκτημάτων της OpenCV :

1. Ταχύτητα. Αποτελεί το βασικότερο συν, αφού αυτός είναι και ο λόγος που αναπτύχθηκε, επειδή η εταιρεία Intel ήθελε να δείξει πόσο προχωρημένοι ήταν οι επεξεργαστές της, οι οποίοι επεξεργάζονταν βίντεο σε πραγματικό χρόνο (πχ. Σε ένα μικρό πρόγραμμα για να ανίχνευση των χαμόγελων σε μια σειρά καρέ βίντεο, χάρη στην OpenCV, θα υπάρχουν 30 καρέ ανά δευτερόλεπτο). Ουσιαστικά, είναι μια βιβλιοθήκη λειτουργιών γραμμένη σε C / C ++ , η οποία παρέχει άμεσα έναν κώδικα σε γλώσσας μηχανής για εκτέλεση. Συνεπώς γίνεται περισσότερη επεξεργασία εικόνας παρά διερμηνεία.
2. Είναι δωρεάν. Συγκεκριμένα, είναι δυνατή η χρήση της OpenCV ελεύθερα σε κάποια εμπορική εφαρμογή, η προβολή της και η διόρθωση - επεξεργασίας της σε περίπτωση ανάγκης. Επίσης, δεν χρειάζεται να ανοίξετε το έργο σας εάν χρησιμοποιείτε αυτή τη βιβλιοθήκη.
3. Υποστήριξη και συνεχής ανάπτυξη. Παρόλο λοιπόν που είναι ελεύθερο λογισμικό υποστηρίζεται και χρηματοδοτείται από μεγάλες εταιρίες (όπως η IBM, η Intel, AMD, η Google, Microsoft, SONY, Siemens) , πανεπιστήμια (όπως το MIT, το Standfort, CMU, Cambridge, INRIA), οργανισμούς και εργαστήρια που βοηθούν συνεχώς στην ανάπτυξή της.
4. Τεράστια βελτιστοποιημένη βιβλιοθήκη. Συγκεκριμένα, υπάρχει μια μεγάλη συλλογή έτοιμων μεθόδων και αλγορίθμων στην OpenCV. Αρκετά από αυτά είναι κατάλληλα για το ξεκίνημα κάθε νέου προγραμματιστή.
5. Μεγάλη απήχηση στους χρήστες. Πλέον η OpenCV έχει πάνω από 47.000 χρήστες και περίπου πάνω από 14 εκατομμύρια λήψεις. Η κοινότητα αυτή, είναι ένας συνδυασμός ανθρώπων από διαφορετικούς τομείς και βιομηχανίες. Επίσης, σημαντικό είναι να αναφερθεί ότι σε αυτή τη κοινότητα υπάρχει η αλληλοβοήθεια.
6. Ελάχιστος και απλός εξοπλισμός. Τα βασικά που χρειάζεται κάποιος είναι μια απλή web κάμερα και έναν απλό προσωπικό υπολογιστή, για την είσοδο και την επεξεργασία εικόνας αντίστοιχα.
7. Χρησιμότητα της σε πολλές εφαρμογές όρασης και εφαρμογές στο κινητό, λόγω του γεγονότος ότι επικεντρώνεται στην απόδοση. Υπάρχει δυνατότητα χρήσης της OpenCV στην εφαρμογή της επιφάνειας εργασίας ή ως το backend της εφαρμογής σας στο διαδίκτυο. Οποιαδήποτε συσκευή που μπορεί να τρέξει C, μπορεί, κατά πάσα πιθανότητα, να τρέξει και την OpenCV.

8. Μικρή χωρητικότητα απαιτούμενων πόρων. Τα τυπικά OpenCV προγράμματα απαιτούν περίπου 70MB RAM για να εκτελεστούν σε πραγματικό χρόνο. **[7] [11] [21] [26] [27]**

3.7 Πεδία εφαρμογής της OpenCV

Από το 1999 μέχρι σήμερα, τα πεδία εφαρμογής αυτής της βιβλιοθήκης έχουν γίνει πολλά και έχει χρησιμοποιηθεί σε προϊόντα και έρευνες. Στην συγκεκριμένη ενότητα θα αναφερθούν τα βασικότερα. Οι προγραμματιστές, οι επιστήμονες της πληροφορικής, και οι ακαδημαϊκοί είναι αυτοί που γνωρίζουν καλύτερα κάθε δυνατότητα και κάθε πτυχή στη χρήση της υπολογιστικής όρασης. Οι εφαρμογές λοιπόν που γνωρίζουν είναι :

1. Συστήματα παρακολούθησης, ελέγχου και ασφαλείας
2. Βιοϊατρική ανάλυση. Μείωση του θορύβου σε ιατρικές εικόνες
3. Αναγνώριση και ανίχνευση προσώπων/ χειρονομιών / συναισθημάτων/ αντικειμένων
4. Συγκερασμός εικόνων και βαθμονόμηση κάμερας από δορυφόρους και διαδικτυακούς (πχ. Google Street View) - εναέριους χάρτες
5. Επικοινωνία ανθρώπου-υπολογιστή (HCI – Human Computer Interaction)
6. Συστήματα όρασης στη ρομποτική. Εφαρμόστηκε στο ρομπότ “Stanley” του Stanford. Το σύστημα αυτό κέρδισε βραβείο δυο εκατομμυρίων δολαρίων στο διαγωνισμό DARPA Grand Challenge Desert Robot Race. Επίσης, εφαρμόζεται στη βιομηχανία για την επιθεώρηση της παραγωγής διάφορων προϊόντων που παράγονται μαζικά.
7. Στερεοφωνική όραση Stereopsis, δηλαδή αντίληψη βάθους με 2 κάμερες.
8. Ευθυγράμμιση και διόρθωση σαρωμένων εικόνων από παραμορφώσεις λόγω του φακού.
9. 2D και 3D εργαλειοθήκη χαρακτηριστικών
10. Αναγνώριση ήχων και μουσικής. Το σύστημα αναγνώρισης εικόνων εφαρμόζονται στο φασματογράφημα του ήχου.
11. Διεπαφές παιχνιδιών
12. Στρατιωτικές εφαρμογές

[7] [10] [11] [19]

3.8 Τι είναι το Microsoft Visual Studio

Εικόνα 3.4: Microsoft Visual Studio 2012 logo **[29]**

Το **Visual Studio** είναι ένα ολοκληρωμένο και ισχυρό περιβάλλον ανάπτυξης (Integrated Development Environment ή IDE) και εργασίας από τη Microsoft, το οποίο κυκλοφόρησε τον Φεβρουάριο του 1997. Περιλαμβάνει εργαλεία που χρησιμοποιούνται στη δημιουργία και την ανάπτυξη εφαρμογών και στον εντοπισμό σφαλμάτων (που λειτουργεί και σε επίπεδο πηγής αλλά, και σε επίπεδο μηχανής) για τα Microsoft Windows και τον Ιστό. Οι πλατφόρμες ανάπτυξης λογισμικού: Windows Forms, Windows API, Windows Store και άλλα βοηθούν στο «έργο» της, δηλαδή την ανάπτυξη εφαρμογών κονσόλας, υπηρεσιών διαδικτύου κτλ.. Αυτό το περιβάλλον ανάπτυξης, περιλαμβάνει διάφορες γλώσσες προγραμματισμού όπως C, C ++, C #, C ++ / CLI ,VB.NET, TypeScript και υποστηρίζει άλλες όπως Python, Ruby, HTML / XHTML, XML / XSLT, JavaScript, CSS. Υπάρχει και η έκδοση Visual Studio Express, που σημαίνει ότι είναι πιο ελαφριά από την κανονική. Επιπλέον, η Microsoft παρέχει μια δωρεάν έκδοση του Visual Studio, που ονομάζεται “κοινοτική έκδοση” (Community edition). **[3] [7] [28]**

Στην συγκεκριμένη πτυχιακή εργασία χρησιμοποιήθηκε το Microsoft Visual Studio 2012 Express.

ΚΕΦΑΛΑΙΟ 4

ΕΠΕΞΕΡΑΣΙΑ ΕΙΚΟΝΑΣ ΜΕ ΧΡΗΣΗ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ OPENCV

Στο συγκεκριμένο κεφάλαιο αναπτύσσονται και παρουσιάζονται αλγόριθμοι επεξεργασίας εικόνας. Συγκεκριμένα θα δούμε πως μπορούμε να εφαρμόσουμε βασικούς χειρισμούς, τεχνικές βελτιστοποίησης και τεχνικές αποκατάστασης σε εικόνες με χρήση της βιβλιοθήκης OpenCV.

4.1 Αριθμητικοί τύποι

Η OpenCV είναι μια βιβλιοθήκη πολλαπλών χρήσεων και λειτουργιών. Μία από τις βασικές δομές δεδομένων που επεξεργάζεται είναι ο πίνακας τύπου cv::Mat, μέσω του οποίου αναπαρίστανται και αποθηκεύονται εικόνες, αραιές, πυκνές αναπαραστάσεις, διαφορετικοί χρωματικοί χώροι, εικόνες 2D με 1-4 κανάλια δεδομένων κτλ. Υπάρχουν πολλοί τύποι δεδομένων που μπορούν να αποθηκευτούν σε αυτόν τον πίνακα. Για να γνωρίζει ο χρήστης τον τύπο, ύστερα από τη χρήση της βιβλιοθήκης, η κλάση Mat (), χάρη στις 2 μεθόδους της depth() και type(), επιστρέφει έναν απλός ακέραιος αριθμό, που αντιστοιχεί στο συγκεκριμένο τύπο. Η depth() είναι ο τύπος δεδομένων κάθε μεμονωμένου στοιχείου στα δεδομένα εικόνας και έχει 8 διαφορετικές τιμές. Η type() συνδυάζει τον τύπο δεδομένων των στοιχείων μαζί με τον αριθμό (1-4) των καναλιών στην εικόνα, επομένως μπορεί να πάρει μία από τις 28 τιμές .Παρακάτω παρουσιάζεται ο πίνακας αντιστοιχιών και αναλυτικότερα οι αριθμητικοί τύποι που υποστηρίζονται από την OpenCV.

CV_8U	0	8-bit unsigned integer (uchar)
CV_8S	1	8-bit signed integer (schar)
CV_16U	2	16-bit unsigned integer (ushort)
CV_16S	3	16-bit signed integer (short)
CV_32S	4	32-bit signed integer (int)
CV_32F	5	32-bit floating point number (float)
CV_64F	6	64-bit floating point number (double)
CV_USRTYPE1	7	User-defined type

Πίνακας 4.1: Αριθμητικοί τύποι που υποστηρίζονται από την OpenCV και τα αντίστοιχα ονόματα τους στη c++ **[30]**

	C1	C2	C3	C4
CV_8U	0	8	16	24
CV_8S	1	9	17	25
CV_16U	2	10	18	26
CV_16S	3	11	19	27
CV_32S	4	12	20	28
CV_32F	5	13	21	29
CV_64F	6	14	22	30

Πίνακας 4.2: Αριθμητικοί τύποι που υποστηρίζονται από την OpenCV και ο αντίστοιχος αριθμός καναλιών της εικόνας **[31]**

8-bit unsigned integer (uchar) : Είναι ένας μη προσημασμένος ακέραιος αριθμός, που έχει μήκος 1 byte. Τα στοιχεία του ανήκουν στο διάστημα [0, 255]. Το 0 δηλώνει το μαύρο χρώμα και το 255 το λευκό.

```
#define CV_8UC1 CV_MAKETYPE(CV_8U,1)
#define CV_8UC2 CV_MAKETYPE(CV_8U,2)
#define CV_8UC3 CV_MAKETYPE(CV_8U,3)
#define CV_8UC4 CV_MAKETYPE(CV_8U,4)
```

Εικόνα 4.1: Οι τιμές της type() για τον τύπο uchar **[32]**

8-bit signed integer (schar) : Είναι ένας προσημασμένος ακέραιος αριθμός, που έχει μήκος 1 byte. Τα στοιχεία του ανήκουν στο διάστημα [-128, 127].

```
#define CV_8SC1 CV_MAKETYPE(CV_8S,1)
#define CV_8SC2 CV_MAKETYPE(CV_8S,2)
#define CV_8SC3 CV_MAKETYPE(CV_8S,3)
#define CV_8SC4 CV_MAKETYPE(CV_8S,4)
```

Εικόνα 4.2: Οι τιμές της type() για τον τύπο schar **[32]**

16-bit unsigned integer (ushort) : Είναι ένας μη προσημασμένος ακέραιος αριθμός, που έχει μήκος 2 bytes. Τα στοιχεία του ανήκουν στο διάστημα [0, 65535]. Χρησιμοποιείται στις συναρτήσεις μετατροπής χώρου χρώματος Το 0 δηλώνει το μαύρο χρώμα και το 65535 το λευκό.

```
#define CV_16UC1 CV_MAKETYPE(CV_16U,1)
#define CV_16UC2 CV_MAKETYPE(CV_16U,2)
#define CV_16UC3 CV_MAKETYPE(CV_16U,3)
#define CV_16UC4 CV_MAKETYPE(CV_16U,4)
```

Εικόνα 4.3: Οι τιμές της type() για τον τύπο ushort **[32]**

16-bit signed integer (short) : Είναι ένας προσημασμένος ακέραιος αριθμός, που έχει μήκος 2 bytes. Τα στοιχεία του ανήκουν στο διάστημα [-32768, 32767].

```
#define CV_16SC1 CV_MAKETYPE(CV_16S,1)
#define CV_16SC2 CV_MAKETYPE(CV_16S,2)
#define CV_16SC3 CV_MAKETYPE(CV_16S,3)
#define CV_16SC4 CV_MAKETYPE(CV_16S,4)
```

Εικόνα 4.4: Οι τιμές της type() για τον τύπο short **[32]**

32-bit signed integer (int) : Είναι ένας προσημασμένος ακέραιος αριθμός, που έχει μήκος 4 bytes. Τα στοιχεία του ανήκουν στο διάστημα [-2147483648, 2147483647].

```
#define CV_32SC1 CV_MAKETYPE(CV_32S,1)
#define CV_32SC2 CV_MAKETYPE(CV_32S,2)
#define CV_32SC3 CV_MAKETYPE(CV_32S,3)
#define CV_32SC4 CV_MAKETYPE(CV_32S,4)
```

Εικόνα 4.5: Οι τιμές της type() για τον τύπο int **[32]**

32-bit floating point number (float) : Είναι ένας πραγματικός αριθμός ακρίβειας (κινητής υποδιαστολής) και τα στοιχεία του συνήθως ανήκουν στο διάστημα [-1.18*10-38, 3.40*10-38]. Χρησιμοποιείται για τον ορισμό αριθμητικών τιμών με

δεκαδικά ψηφία, στις συναρτήσεις της γραμμικής άλγεβρας, στις συναρτήσεις μετατροπής χώρου χρώματος και στους αλγόριθμους μηχανικής μάθησης.

```
#define CV_32FC1 CV_MAKETYPE(CV_32F,1)
#define CV_32FC2 CV_MAKETYPE(CV_32F,2)
#define CV_32FC3 CV_MAKETYPE(CV_32F,3)
#define CV_32FC4 CV_MAKETYPE(CV_32F,4)
```

Εικόνα 4.6: Οι τιμές της type() για τον τύπο float **[32]**

64-bit floating point number (double) : Είναι ένας πραγματικός αριθμός διπλής ακρίβειας και τα στοιχεία του συνήθως ανήκουν στο διάστημα [0, 1], εκτός αν μετά από κάποιο μαθηματικό χειρισμό βρίσκονται στο [0, 255]. Το 0 δηλώνει το μαύρο χρώμα και το 1 το λευκό. Χρησιμοποιείται στην απεικόνιση εικόνων και στην εφαρμογή της γραμμικής άλγεβρας (αρκεί να είναι διάστημα [0, 1]). Επίσης, και στους αλγόριθμους μηχανικής μάθησης.

```
#define CV_64FC1 CV_MAKETYPE(CV_64F,1)
#define CV_64FC2 CV_MAKETYPE(CV_64F,2)
#define CV_64FC3 CV_MAKETYPE(CV_64F,3)
#define CV_64FC4 CV_MAKETYPE(CV_64F,4)
```

Εικόνα 4.7: Οι τιμές της type() για τον τύπο double **[32]**

User defined type : Η πλειάδα πολλών στοιχείων που έχουν τον ίδιο τύπο και αποτελούν και αυτά με τη σειρά τους στοιχεία ενός πίνακα, θεωρούνται πίνακες πολλών καναλιών. Έτσι λοιπόν εξαρτάται από τον χρήστη πως θα χρησιμοποιήσει αυτούς τους τύπους για να δημιουργήσει κάποιον δικό του.

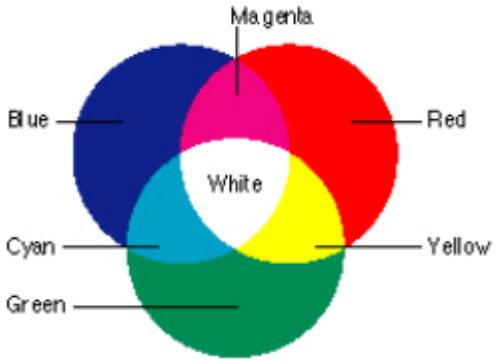
4.2 Τύποι εικόνων

Η βιβλιοθήκη OpenCV υποστηρίζει τους παρακάτω τύπους εικόνας:

- **Εικόνες φωτεινότητας (Intensity images)**
  Οι αριθμητικοί τύποι που εκφράζουν αυτού του είδους εικόνες είναι 8-bit unsigned integer [διάστημα [0, 255], 16-bit unsigned integer [0, 65535] και 64-bit floating point number (double) [0, 1].
- **Εικόνες δυαδικές (Binary images)**
  Ο αριθμητικός τύπος που εκφράζει κάθε εικονοστοιχείο αυτού του είδους εικόνων είναι συνήθως 8-bit unsigned integer [0 ή 255] και 64-bit floating point number [0 ή 1].

- **Εικόνες έγχρωμες (RGB)**

  Υπάρχουν τρεις πίνακες, ένα για κάθε χρώμα (Red , Green, Blue), τα οποία συνθέτουν τις έγχρωμες εικόνες και συνδυάζονται με διάφορους τρόπους για να αναπαραχθούν άλλα χρώματα.. Οι αριθμητικοί τύποι που εκφράζουν κάθε πίνακα αυτού του είδους εικόνων είναι 8-bit signed integer [διάστημα [0, 255], 16-bit unsigned integer [0, 65535] και 64-bit floating point number (double) [0, 1].



Εικόνα 4.8: Ανάμειξη βασικών χρωμάτων **[35]**

4.3 Μετατροπή τύπων εικόνων

Για την μετατροπή μιας εικόνας από ένα χρωματικό χώρο σε έναν άλλο, γίνεται χρήση της συνάρτησης **cvtColor**. Είναι απαραίτητο για την μετατροπή μιας έγχρωμης εικόνας να αποσαφηνιστεί η σειρά των καναλιών (RGB ή BGR). Η OpenCv ως default έχει τη σειρά RGB, αλλά στην πραγματικότητα είναι BGR (πχ για μια 24-bit έγχρωμη εικόνα, τα κάθε 8-bit θα παίρνουν το αντίστοιχο χρώμα της σειράς), επειδή υπάρχει αντιστροφή των bytes.

Συντάσσεται ως εξής :

cvtColor(InputArray src, OutputArray dst, int code, int dstCn=0) όπου

- src : Εικόνα εισόδου (8-bit unsigned,16-bit unsigned,)
- dst : Εικόνα εξόδου (ίδιο μέγεθος και βάθος με εικόνα εισόδου)
- code : Κώδικας μετατροπής μιας εικόνας από ένα χρωματικό χώρο σε έναν άλλο

- dstCn :Πλήθος καναλιών της εικόνας εξόδου (αν ο αριθμός των καναλιών είναι 0, αυτό σημαίνει πως όλα τα κανάλια προέρχονται από την εικόνα εισόδου και τον κώδικα μετατροπής)

Η σύνταξη μετασχηματισμού μιας εικόνας RGB (έγχρωμη) σε grayscale (μονοχρωματική) είναι αυτή:

cvtColor(src, dst, CV_RGB2GRAY)

4.4 Ανάγνωση εικόνων

Για την ανάγνωση εικόνων διαφόρων τύπων στην OpenCV γίνεται χρήση της συνάρτησης **imread**. Ο τύπος καθορίζεται από το περιεχόμενο της εικόνας και όχι από την επέκταση αρχείου.

Συντάσσεται ως εξής :

imread(const string& filename, int flags=1) όπου

- filename : Όνομα του αρχείου προς φόρτωση
- flags: Καθορίζουν τον αριθμητικό τύπο της εικόνας, δηλαδή ελέγχουν τον τρόπο που διαβάζεται. Υπάρχουν οι εξής τιμές:
  - CV_LOAD_IMAGE_COLOR, μετατρέπει οποιαδήποτε εικόνα σε έγχρωμη, τα κανάλια αποθήκευσης της οποίας θα έχουν τη σειρά B G R.
  - CV_LOAD_IMAGE_ANYDEPTH, επιστρέφει 16-bit ή 32-bit εικόνα όταν η είσοδος έχει το αντίστοιχο βάθος, αλλιώς επιστρέφει 8-bit εικόνα.
  - CV_LOAD_IMAGE_GRAYSCALE, μετατρέπει οποιαδήποτε εικόνα σε μονοχρωματική αποχρώσεων του γκρι (grayscale)
  - CV_LOAD_IMAGE_UNCHANGED, επιστρέφει την εικόνα ως έχει
  - >0, επιστρέφει μια έγχρωμη εικόνα τριών καναλιών, τα κανάλια αποθήκευσης της οποίας θα έχουν τη σειρά B G R.
  - =0, επιστρέφει μια μονοχρωματική εικόνα αποχρώσεων του γκρι (grayscale)
  - <0, επιστρέφει την εικόνα ως έχει

Οι τύποι εικόνων - μορφές αρχείων που διαβάζει η imread :

- JPEG (Joint Photographic Experts Group) : filenname.jpeg, filename.jpg, filename.jpe, filename.jp2 (JPEG 2000)
- PNG (Portable Network Graphics) : filename.png
- BMP (Microsoft Windows Bitmaps) : filename.bmp, filename.dib
- WebP : filename.web
- Portable Image Format : filename.pbm (portable bitmap format), filename.pgm (portable graymap format), filename.ppm (portable pixmap format)
- Sun Rasters : filename.sr, filename.ras
- TIFF (Tagged Image File Format) : filename.tiff, filename.tif

Αξίζει να σημειωθεί πως αν μια εικόνα δεν μπορεί να διαβαστεί από την imread (πχ. λόγω έλλειψης αρχείου, δικαιωμάτων), η συνάρτηση αυτή επιστρέφει έναν κενό πίνακα (Mat :: data == NULL).

Παράδειγμα

```
Mat img = imread ("pic.JPG", CV_LOAD_IMAGE_UNCHANGED);
```

Διαβάζει τα δεδομένα της εικόνας στο αρχείο "pic.JPG" και τα αποθηκεύει στο 'img', τύπου mat, που είναι μια δομή δεδομένων για την αποθήκευση εικόνων σε έναν πίνακα.

4.5 Εγγραφή εικόνων

Για την εγγραφή εικόνων διαφόρων τύπων στην OpenCV, γίνεται χρήση της συνάρτησης **imwrite**. Η μορφή της εικόνας καθορίζεται από την επέκταση αρχείου (που αναφέρθηκε παραπάνω).

Συντάσσεται ως εξής :

```
imwrite(const String& filename, InputArray img, const vector<int>&
                params=vector<int>() ) όπου
```

- filename : Όνομα του αρχείου

- img : Εικόνα που θα αποθηκευτεί
- params : Αυτές οι ειδικές παράμετροι αποθήκευσης και μορφοποίησης κωδικοποιούνται σε ζευγάρια πχ. paramId_1, paramValue_1, paramId_2, paramValue_2

Οι παράμετροι και οι τύποι εικόνων - μορφές αρχείων που υποστηρίζει η imwrite :

- CV_IMWRITE_JPEG_QUALITY, χρησιμοποιείται για τον τύπο JPEG, με τιμές από 0 έως 100 (όσο μεγαλύτερος είναι ο αριθμός τόσο το καλύτερο, γι'αυτό και η default τιμή είναι 95)
- CV_IMWRITE_PNG_COMPRESSION, χρησιμοποιείται για τον τύπο PNG, με τιμές από 0 έως 9 (όσο μικρότερος είναι ο αριθμός τόσο μεγαλύτερο είναι το μέγεθος και μικρότερος ο χρόνος συμπίεσης, γι'αυτό και η default τιμή είναι 3)
- CV_IMWRITE_PXM_BINARY, χρησιμοποιείται για τους τύπους PPM, PGM, PBM, με τιμές από 0 ή 1 (η default τιμή είναι 1)

4.6 Απεικόνιση εικόνων

Για την απεικόνιση εικόνων διαφόρων τύπων στην OpenCV, γίνεται χρήση της συνάρτησης **imshow**. Για να γίνει αυτό, πρώτα πρέπει να δημιουργηθεί ένα παράθυρο OpenCV με τη βοήθεια της συνάρτησης namedWindow. Μόλις γίνει αυτό, η βιβλιοθήκη αυτόματα το διαχειρίζεται. Είναι απαραίτητο να δοθεί ένα όνομα και ο τρόπος χειρισμού της αλλαγής της εικόνας με γνώμονα το μέγεθος της.

Υπάρχουν 2 τρόποι χειρισμού :

- WINDOW_AUTOSIZE, το μέγεθος του παραθύρου (που δεν μπορεί να αλλάξει) θα καταλαμβάνει το μέγεθος της εικόνας που απεικονίζει (default, δηλαδή αν ο χρήστης δεν ορίσει δεύτερο όρισμα, τότε θεωρείται WINDOW_AUTOSIZE)
- WINDOW_NORMAL, το μέγεθος του παραθύρου θα είναι με βάση το οποίο θα αλλάξει το μέγεθος της εικόνας. Είναι απαραίτητο να καθοριστεί αν η εικόνα θα παραμείνει με τις διαστάσεις που έχει (WINDOW_KEEPRATIO) ή όχι (WINDOW_FREERATIO), κάνοντας χρήση του | .

Συντάσσεται ως εξής :

namedWindow("window name", WINDOW_AUTOSIZE)

Η συνάρτηση imshow εξυπηρετεί στην ενημέρωση του παραθύρου με μια καινούρια εικόνα.

Συντάσσεται ως εξής :

imshow("window name", img) όπου

- window name : Όνομα του επιθυμητού παραθύρου στην OpenCV
- img : Εικόνα που θα χρησιμοποιηθεί

4.7 Παραδείγματα χειρισμού εικόνων

4.7.1 Χαρακτηριστικά εικόνας

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <Windows.h>

using namespace std;
using namespace cv;

int main()
{
	string imageName = "pout.tif";
	string imagePath = "C:/images/pout.tif";

	Mat img = imread(imagePath,0);

	// Ypologismos tou megethous tis eikonas se arithmo pixels
	int size = img.size().width * img.size().height;

	// Euresi tou tupou dedomenon tis eikonas.
	// H depth() epistrefei enan arithmo pou antistoixei se enan tupo dedomenon
	int type = img.depth();

	string dtype;
	if (type == 0)
		dtype = "CV_8U";
	else if (type == 1)
		dtype = "CV_8S";
	else if (type == 2)
		dtype = "CV_16U";
	else if (type == 3)
		dtype = "CV_16S";
	else if (type == 4)
		dtype = "CV_32S";
	else if (type == 5)
```

```
            dtype = "CV_32F";
        else if (type == 6)
            dtype = "CV_64F";

        cout << "Image Name: " << imageName << endl;
        Sleep(1000);
        cout << "Image Path: " << imagePath << endl;
        Sleep(1000);
        cout << "Image Width: " << img.size().width << endl; // Mikos h arithmos
        sthlwn tis eikonas
        Sleep(1000);
        cout << "Image Height: " << img.size().height << endl; // Ypsos h arithmos
        gramwn tis eikonas
        Sleep(1000);
        cout << "Image Dimensions: " << img.size() << endl; // Diastaseis tis
        eikonas
        Sleep(1000);
        cout << "Number of channels: " << img.channels() << endl; // Arithmos
        kanaliwn tis eikonas
        Sleep(1000);
        cout << "Image Size: " << size << " pixels" << endl;
        Sleep(1000);
        cout << "Image Data type: " << dtype << endl;
        Sleep(10000);

return 0;
}
```

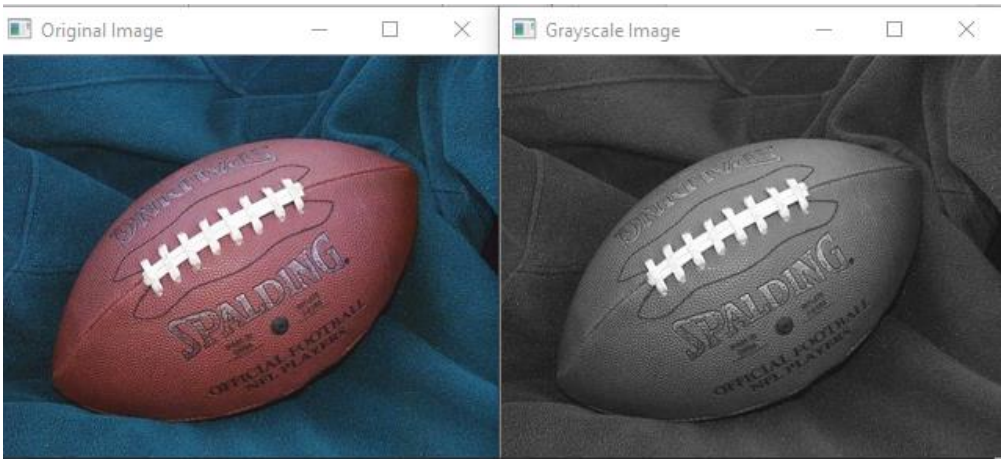
```
Image Name: pout.tif
Image Path: G:/images/pout.tif
Image Width: 240
Image Height: 291
Image Dimensions: [240 x 291]
Number of channels: 1
Image Size: 69840 pixels
Image Data type: CV_8U
```

Εικόνα 4.9: Αποτέλεσμα παραδείγματος 4.7.1

4.7.2 Ανάγνωση, αποθήκευση και απεικόνιση εικόνας

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main()
{
    Mat img = imread("C:/images/football.jpg"); // Anagnosi egxromis eikonas

       Mat img_gray;
       cvtColor(img, img_gray, CV_BGR2GRAY); // Metatropi se eikona foteinotitas

       // Emfanisi ton eikonon
       namedWindow("Original Image", CV_WINDOW_AUTOSIZE);
       namedWindow("Grayscale Image", CV_WINDOW_AUTOSIZE);
```

```
        imshow("Original Image", img);
        imshow("Grayscale Image", img_gray);
        waitKey(0);

        imwrite("C:/images/football_gray.jpg", img_gray); // Apothikeusi tis eikonas
        foteinotitas
        destroyAllWindows();
return 0;
}
```



Εικόνα 4.10: Αποτέλεσμα παραδείγματος 4.7.2

4.8 Χειρισμός εικονοστοιχείων

4.8.1 Τι είναι το εικονοστοιχείο

Εικονοστοιχείο (Pixel) ονομάζεται ένα "σημείο" μιας εικόνας, που αποτελεί ουσιαστικά ένα μικρό δείγμα πληροφορίας. Στον τομέα της πληροφορικής θεωρείται μια ψηφίδα του ψηφιδωτού, με το οποίο απεικονίζεται μια εικόνα. Συγκεκριμένα, στα υπολογιστικά συστήματα ένα pixel είναι ένα "κελί" σε έναν δισδιάστατο πίνακα που αναπαριστά κάποια εικόνα. Είναι τόσο μεγάλο το πλήθος των υποδιαιρέσεων σε στήλες και γραμμές αυτού του πίνακα, που είναι αδύνατον για το ανθρώπινο μάτι να παρατηρήσει κάθε pixel ξεχωριστά. **[16]**

4.8.2 Ανάγνωση τιμής ενός εικονοστοιχείου

Για την ανάγνωση τιμής ενός εικονοστοιχείου σε έναν πίνακα απαιτείται η γνώση του τύπου της και του αριθμό των καναλιών.

- Για μια εικόνα ενός καναλιού αριθμητικού τύπου **uchar** :
  Scalar img.at<uchar>(y, x) ή
  Scalar intensity = img.at<uchar>(Point(x, y)

- Για μια εικόνα τριών καναλιών (BGR) αριθμητικού τύπου **uchar** :
  Vec3b intensity = img.at<Vec3b>(y, x);
  uchar blue = intensity.val[0];
  uchar green = intensity.val[1];
  uchar red = intensity.val[2];
- Για μια εικόνα τριών καναλιών αριθμητικού τύπου **float** :
  Vec3f intensity = img.at<Vec3f>(y, x);
  float blue = intensity.val[0];
  float green = intensity.val[1];
  float red = intensity.val[2];

όπου:

- img : εικόνα που θα χρησιμοποιηθεί
- y, x : συντεταγμένες εικονοστοιχείου (y : γραμμές, x: στήλες)

4.8.3 Τροποποίηση τιμής ενός εικονοστοιχείου

Η τροποποίηση τιμής φωτεινότητας ενός εικονοστοιχείου σε έναν πίνακα επιτυγχάνεται με μέθοδο

img.at<uchar>(y, x) = z;

όπου z ο καινούριος επιθυμητός αριθμός φωτεινότητας, ο οποίος πρέπει να είναι σε κάποιο συγκεκριμένο διάστημα ανάλογα με τον αριθμητικό τύπο της εικόνας (πχ. για 8-bit unsigned integer, το διάστημα είναι [0,255])

4.8.4 Τροποποίηση τιμής περισσότερων του ενός εικονοστοιχείων

Προηγουμένως είδαμε πως μπορούμε να τροποποιήσουμε τη τιμή ενός εικονοστοιχείου. Αν όμως θέλαμε να τροποποιήσουμε τα pixels που έχουν τιμή από 0 έως 50 δίνοντας τους τη τιμή 0? Θα έπρεπε να διαβάσουμε τις τιμές όλων των pixels, να ελέγξουμε αν έχουν τιμή από 0 έως 50 και αν ναι τότε να τους δώσουμε τη τιμή 0. Αντί αυτού, η OpenCV διαθέτει τη συνάρτηση **inRange()** για την εύρεση των pixels που ανήκουν σε ένα εύρος τιμών. Συντάσσεται ως εξής:

inRange(inputArray src, InputArray lowerb, InputArray upperb, OutputArray dst)

όπου

- src: Εικόνα εισόδου

- lowerb: Κατώτερο όριο του εύρους τιμών
- upperb: Ανώτερο όριο του εύρους τιμών
- dst: Εικόνα εξόδου ίδιου μεγέθους με τη src και τύπο δεδομένων CV_8U

Με την συνάρτηση inRange(), όταν η τιμή ενός pixel της src ανήκει στο εύρος [lowerb, upperb], η τιμή του αντίστοιχου pixel τις dst γίνεται 255. Μέτα από αυτό μπορούμε να χρησιμοποιήσουμε τη μέθοδο setTo() για τη τροποποίηση των pixels αυτών δίνοντας ως ορίσματα τη νέα τιμή και τη εικόνα εξόδου της συνάρτησης inRange() πχ:
img.seTo(0, dst);

4.8.5 Εύρεση ελάχιστης και μέγιστης τιμής

Για την εύρεση ελάχιστης και μέγιστης τιμής ενός εικονοστοιχείου και τις θέσεις αυτών σε έναν πίνακα, γίνεται χρήση της συνάρτησης **minMaxLoc()** (σε 2D πίνακες) και **minMaxIdx()** (σε nD πίνακες). Συντάσσονται ως εξής :

minMaxLoc(InputArray src, double* minVal, double* maxVal=0, Point* minLoc=0, Point* maxLoc=0, InputArray mask=noArray()) όπου:

- src: Είσοδος στο πίνακα ενός καναλιού
- minVal: Δείκτης στην επιστρεφόμενη ελάχιστη τιμή
- maxVal: Δείκτης στην επιστρεφόμενη μέγιστη τιμή
- minLoc: Δείκτης στην επιστρεφόμενη ελάχιστη θέση
- maxLoc: Δείκτης στην επιστρεφόμενη μέγιστη θέση
- mask : δεν απαιτείται η χρήση της, εκτός αν χρειάζεται η επιλογή υποπίνακα

minMaxIdx(InputArray src, double* minVal, double* maxVal=0, int* minLoc=0, int* maxLoc=0, InputArray mask=noArray()) όπου:

- src: Είσοδος στο πίνακα ενός καναλιού
- minVal: Δείκτης στην επιστρεφόμενη ελάχιστη τιμή
- maxVal: Δείκτης στην επιστρεφόμενη μέγιστη τιμή
- minIdx: Δείκτης στην επιστρεφόμενη ελάχιστη θέση
- maxIdx: Δείκτης στην επιστρεφόμενη μέγιστη θέση

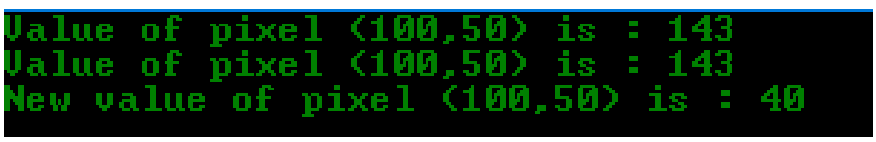
- mask : δεν απαιτείται η χρήση της, εκτός αν χρειάζεται η επιλογή υποπίνακα

Αν την ελάχιστη και τη μέγιστη τιμή την έχουν περισσότερα από ένα pixels, τότε οι παράμετροι minLoc, maxLoc, minIdx και maxIdx έχουν τις πρώτες θέσεις στις όποιες βρέθηκαν η ελάχιστη και μέγιστη τιμή αντίστοιχα.

4.8.6 Παραδείγματα χειρισμού εικονοστοιχείων

4.8.6.1 Ανάγνωση και τροποποίηση τιμής φωτεινότητας ενός pixel μονοχρωματικής εικόνας

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <Windows.h>

using namespace std;
using namespace cv;

int main()
{
	// Anagnosi monoxromatikis eikonas
	Mat img = imread("C:/images/pentagon.png",0);

	// Prosvasi sti timi tou pixel (100,50) kai apothikeush sth metavliti px
	uchar px = img.at<uchar>(50, 100);
	// allios: uchar px = img.at<uchar>(Point(100, 50));

	// Emfanisi tis timis
	cout<< "Value of pixel (100,50) is : "<<(int)px<<endl;
	Sleep(2000);

	// Emfanisi tis timis xoris na tin exoume apothikeusi se kapoia metavlith
	cout<<"Value of pixel (100,50) is : "<<(int)img.at<uchar>(50,100)<<endl;
	Sleep(2000);

	// Tropopoihsi tis timis tou pixel (10, 50)
	img.at<uchar>(50, 100) = 40;

	// Emfanisi tis neas timis
	cout<<"New value of pixel (100,50) is : "<<(int)img.at<uchar>(50, 100);
	Sleep(10000);
 return 0;
 }
```



Εικόνα 4.11: Αποτέλεσμα παραδείγματος 4.8.6.1

4.8.6.2 Ανάγνωση και τροποποίηση τιμής φωτεινότητας ενός pixel έγχρωμης εικόνας

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <Windows.h>

using namespace std;
using namespace cv;

int main()
{
	// Anagnosi egxromis eikonas
	Mat img = imread("C:/images/clown.bmp");

	// Prosvasi sti timi tou pixel (100,50) kai apothikeush sth metavliti px
	Vec3b px = img.at<Vec3b>(50,100);

	// Emfanisi tis timis
	cout << "Value of pixel (100,50) is: " << img.at<Vec3b>(50,100) << endl;
	Sleep(2000);

	// Tropopoihsi tis timis tou pixel (10, 50)
	img.at<Vec3b>(50,100)[0] = 255;
	img.at<Vec3b>(50,100)[1] = 0;
	img.at<Vec3b>(50,100)[2] = 0;

	// Emfanisi tis neas timis
	cout << "New value of pixel (100,50) is: " << img.at<Vec3b>(50,100) << endl;
	Sleep(10000);
return 0;
}
```

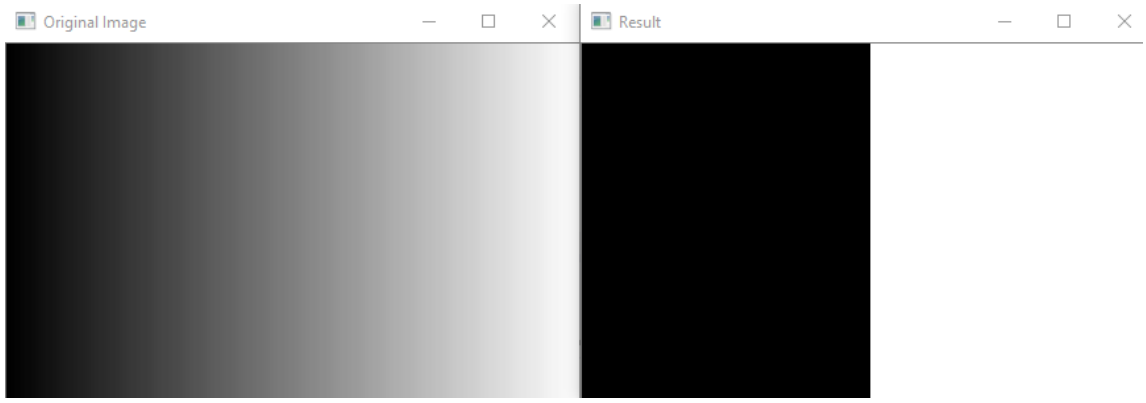
```
Value of pixel (100,50) is: [74, 106, 147]
New value of pixel (100,50) is: [255, 0, 0]
```

Εικόνα 4.12: Αποτέλεσμα παραδείγματος 4.8.6.2

4.8.6.3 Τροποποίηση τιμής φωτεινότητας περισσοτέρων του ενός pixels

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main()
{
	// Anagnosi monoxromatikis eikonas
	Mat img = imread("C:/images/gradient.jpg", 0);
	int width = img.cols; int height = img.rows;

	// Dimiourgia telikis eikonas
```

```
        Mat res; img.copyTo(res);

        // Dimiourgia ton 2 maskon
        Mat mask1(width, height, CV_8UC1, Scalar(0));
        Mat mask2(width, height, CV_8UC1, Scalar(0));

        // Euresi ton pixels me timi sto euros [0, 127] kai apodosi twn ti timi 0
        inRange(img, 0, 127, mask1);
        img.setTo(0, mask1);

        // Euresi twn pixels me timi sto euros [128, 255] kai apodosi twn ti timi
        255
        inRange(img, 128, 255, mask2);
        img.setTo(255, mask2);

        // Emfanisi twn eikonwn
        namedWindow("Original Image", CV_WINDOW_AUTOSIZE);
        namedWindow("Result", CV_WINDOW_AUTOSIZE);

        imshow("Original Image", img);
        imshow("Result", res);

        waitKey(0);
        destroyAllWindows();
return 0;
}
```



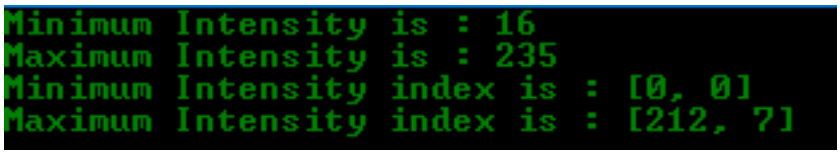
Εικόνα 4.13: Αποτέλεσμα παραδείγματος 4.8.6.3

4.8.6.4 Εύρεση ελάχιστης και μέγιστης τιμής φωτεινότητας

Στη συνέχεια παρουσιάζεται μια εφαρμογή της συνάρτησης minMaxLoc για την εικόνα pic, ώστε να βρεθεί η ελάχιστη και μέγιστη τιμή φωτεινότητας μαζί με τις θέσεις τους σε μία μονόχρωμη εικόνα (grayscale).

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <Windows.h>

using namespace std;
using namespace cv;

int main()
```

```
{
	Mat img = imread("C:/images/street.tif",0);
	double min,max;
	Point min_loc, max_loc;

	minMaxLoc(img, &min, &max, &min_loc, &max_loc);

	cout << "Minimum Intensity is : " << min << endl;
	Sleep(2000);
	cout << "Maximum Intensity is : " << max << endl;
	Sleep(2000);
	cout << "Minimum Intensity index is : " << min_loc << endl;
	Sleep(2000);
	cout << "Maximum Intensity index is : " << max_loc << endl;
	Sleep(10000);
return 0;
}
```



Εικόνα 4.14: Αποτέλεσμα παραδείγματος 4.8.6.4

4.9 Ιστόγραμμα

4.9.1 Τι είναι ιστόγραμμα

Ιστόγραμμα (Histogram) είναι η γραφική αναπαράσταση της κατανομής φωτεινότητας μιας εικόνας, δηλαδή δείχνει τον αριθμό των εικονοστοιχείων για κάθε επιθυμητή τιμή φωτεινότητας. Ουσιαστικά είναι μια συλλογή δεδομένων, που είναι απαραίτητα για να περιγράψουν μια εικόνα και να μετρήσουν για παράδειγμα εκτός από τη φωτεινότητα, τις κατευθύνσεις και τις κλίσεις της.

4.9.2 Υπολογισμός και κανονικοποίηση ιστογράμματος

4.9.2.1 Υπολογισμός

Η βιβλιοθήκη OpenCV παρέχει τη συνάρτηση **calcHist** για αυτό τον υπολογισμό, η οποία μπορεί να εφαρμοστεί σε εικόνες ή επίπεδα εικόνας με έως 32 διαστάσεις.

Συντάσσεται ως εξής :

calcHist(const Mat* images, int nimages, const int* channels, InputArray mask, OutputArray hist, int dims, const int* histSize, const float** ranges, bool uniform=true, bool accumulate=false) όπου

- images : Πηγές των πινάκων που θα χρησιμοποιηθούν, οι οποίοι πρέπει να έχουν ίδιο μέγεθος και βάθος - depth (CV_8U ή CV_32F). Το πλήθος των καναλιών μπορεί να είναι διαφορετικός.
- nimages : Αριθμός των εικόνων προς εξέταση
- channels : Πλήθος καναλιών dims που συμμετέχουν στον υπολογισμό. Τα κανάλια του πρώτου πίνακα είναι αριθμημένα από το 0 μέχρι το images[0].channels()-1, τα κανάλια του δεύτερου από το images[0].channels() έως το images[0].channels() + images[1].channels()-1 κτλ.
- mask : Δεν είναι απαραίτητη η χρήση της. Τα στοιχεία της μη μηδενικής μάσκας «μαρκάρουν» αυτά του πίνακα, που μετρήθηκαν στο ιστόγραμμα. Σε περίπτωση που ο πίνακας δεν είναι κενός, τότε πρέπει να είναι 8-bit ίδιου μεγέθους με το images[i].
- hist : Ιστόγραμμα εξόδου
- dims : Διαστάσεις του ιστογράμματος, που πρέπει να είναι θετικές τιμές μικρότερες ή ίσες του 32
- histSize : Πίνακας με τα μεγέθη του ιστογράμματος σε κάθε διάσταση
- ranges : Εύρος του ιστογράμματος που θα σχεδιαστεί. Για να σχεδιαστεί όλο το ιστόγραμμα, πρέπει να γραφτεί το διάστημα [0,255].
- uniform : Flag που δείχνει αν το ιστόγραμμα είναι ομοιόμορφο ή όχι
- accumulate : Flag, με τη βοήθεια του οποίου γίνεται ο υπολογισμός ενός ιστογράμματος από πολλαπλά σύνολα των πινάκων ή ενημέρωση της τη σωστή στιγμή

4.9.2.2 Κανονικοποίηση

Για την κανονικοποίηση ενός ιστογράμματος, αλλά και μιας εικόνας, η βιβλιοθήκη διαθέτει τη συνάρτηση **normalize()**. Όμως υπάρχει και μια σχέση που εξυπηρετεί σε αυτό τον σκοπό, η οποία είναι :

hist / (rows * cols)

όπου hist είναι το ιστόγραμμα και τα rows, cols οι διαστάσεις της εικόνας
Το γινόμενο τους δίνει το συνολικό αριθμό των pixels αυτής.

Η normalize συντάσσεται ως εξής :

normalize(InputArray src, OutputArray dst, double alpha=1, double beta=0, int norm_type=NORM_L2, int dtype=-1, InputArray mask=noArray()) όπου

- src: Πίνακας εισόδου (ιστόγραμμα ή εικόνα)
- dst: Πίνακας εξόδου, ίδιου μεγέθους με τον πίνακα εισόδου
- alpha: Δηλώνει το κατώτερο όριο φωτεινοτήτων, δηλαδή την αρχή του ιστογράμματος
- beta: Δηλώνει το ανώτερο όριο φωτεινοτήτων, δηλαδή το τέλος του ιστογράμματος
- normType: Τύπος κανονικοποίησης
- dtype: Σε περίπτωση αρνητικής τιμής, ο πίνακας εξόδου έχει τον ίδιο τύπο με τον πίνακα εισόδου. Σε περίπτωση μη αρνητικής τιμής, έχει τον ίδιο πλήθος καναλιών με το src και το βάθος είναι ίσο με τη συνάρτηση CV_MAT_DEPTH(dtype)
- mask: Προαιρετική η χρήση της

Ο υπολογισμός και η κανονικοποίηση του ιστογράμματος γίνεται με τις εξής εντολές:

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main()
{
	Mat img = imread("C:/images/street.tif",0);
	int cols = img.cols; int rows = img.rows;

	// Arxikopoihsh parametrwn
	int histSize = 256;
	float range[] = { 0, 255 };
	const float *ranges[] = { range };

	// Ypologismos istogrammatos
	MatND hist;
	calcHist( &img, 1, 0, Mat(), hist, 1, &histSize, ranges, true, false );

	// Ypologismos kanonikopoihmenou istogrammatos me tin normalize
	MatND norm_hist;
	normalize(hist, hist, 0, 255, NORM_MINMAX, -1, Mat());

	// Ypologismos kanonikopoihmenou istogrammatos
	MatND norm_hist2 = hist / (rows * cols);
```

```
return 0;
}
```

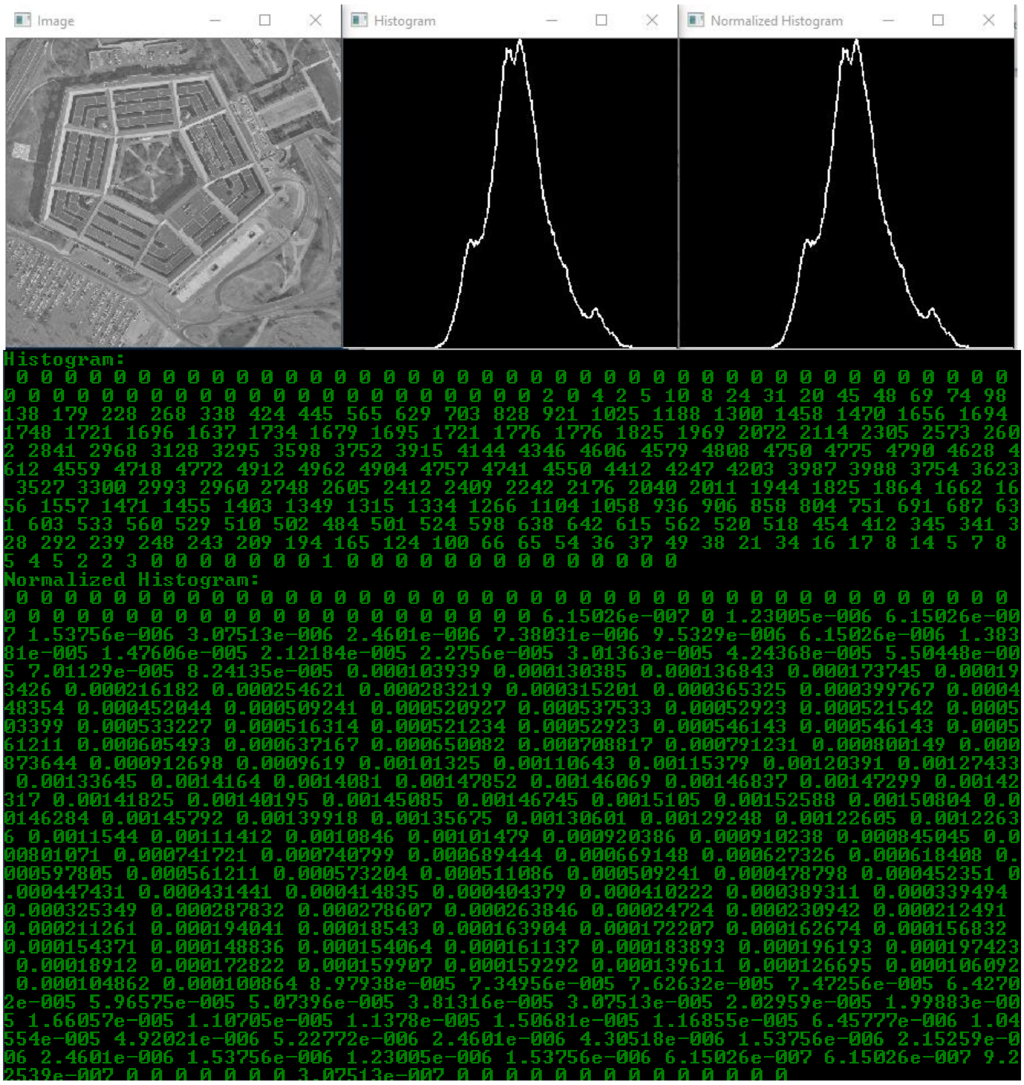
4.9.3 Απεικόνιση ιστογράμματος

Επειδή η OpenCV δεν διαθέτει συνάρτηση plot για την απεικόνιση ιστογραμμάτων, αυτά εμφανίζονται χωρίς τους άξονες x και y. Γι' αυτό τον λόγο γίνεται χρήση της συνάρτησης imshow.

Παράδειγμα υπολογισμού και εμφάνισης ιστογραμμάτων

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(int, char**)
{
	Mat img = imread("C:/images/pentagon.png",0);
	 // Arxikopoihsh parametrwn
	int histSize = 256;
	float range[] = { 0, 255 };
	 const float *ranges[] = { range };

	// Ypologismos Istogrammatos
	MatND hist, norm_hist;
	calcHist( &img, 1, 0, Mat(), hist, 1, &histSize, ranges, true, false );

	// Emfanisi tou Istogrammatos sto cmd
	double total;
	total = img.rows * img.cols;
	cout<< "Histogram: "<< endl;

	for( int h = 0; h < histSize; h++ ){
	     float binVal = hist.at<float>(h);
	     cout<<" "<<binVal;
	  }
	cout<<""<< endl;

	// Sxediasi tou Istogrammatos
	int hist_w = 512; int hist_h = 400;
	int bin_w = cvRound( (double) hist_w/histSize );
	// Dimiourgia eikonas gia th sxediash tou istogrammatos
	Mat histImage( hist_h, hist_w, CV_8UC1, Scalar( 0,0,0) );
	// Kanonikopoihsh tou gia thn orthi sxediash tou
	normalize(hist, hist, 0, histImage.rows, NORM_MINMAX, -1, Mat() );

	for( int i = 1; i < histSize; i++ ){
	line(histImage, Point(bin_w*(i-1), hist_h - cvRound(hist.at<float>(i-1))),
	                Point( bin_w*(i), hist_h - cvRound(hist.at<float>(i))),
	                Scalar( 255, 0, 0), 2, 8, 0  );

}
	// Ypologismos kai emfanisi tou kanonikopoihmenou Istogrammatos sto cmd
```

```
        // Den apaiteitai sxediash tou afou san grafima einai idio me to Istogramma
        cout<< "Normalized Histogram: "<< endl;
        norm_hist = hist / (img.size().width * img.size().height);
        for( int h = 0; h < histSize; h++ ){
             float binVal = norm_hist.at<float>(h);
             cout<<" "<<binVal;
        }
        cout<<""<< endl;

        // Emfanish ths eikonas kai twn istogrammatwn
        namedWindow("Image", CV_WINDOW_NORMAL);
        namedWindow("Histogram", CV_WINDOW_NORMAL);
        namedWindow("Normalized Histogram", CV_WINDOW_NORMAL);

        imshow("Image", img);
        imshow("Histogram", histImage);
        imshow("Normalized Histogram", histImage);

        waitKey(0);
        destroyAllWindows();
return 0;
}
```



Εικόνα 4.15: Ιστόγραμμα και κανονικοποιημένο ιστόγραμμα της εικόνας pentagon.png

4.10 Τεχνικές σημείου

Ουσιαστικά οι **τεχνικές σημείου**, όπως δηλώνουν και το όνομα τους εφαρμόζονται σε διάφορα σημεία ή εικονοστοιχεία μιας εικόνας. Αυτά τα pixels, έχουν 256 διαφορετικές τιμές ή αποχρώσεις του γκρι, στο διάστημα [0,255]. Υπάρχει μια σχέση, η οποία δηλώσει το γεγονός πως αν εφαρμοστεί μια τεχνική σημείου σε μια εικόνα, στην έξοδο εμφανίζεται η ιδία εικόνα με διαφορετικές φωτεινότητες. **[1]**

$$I´(x,y) = f[I(x,y)] \text{ όπου}$$

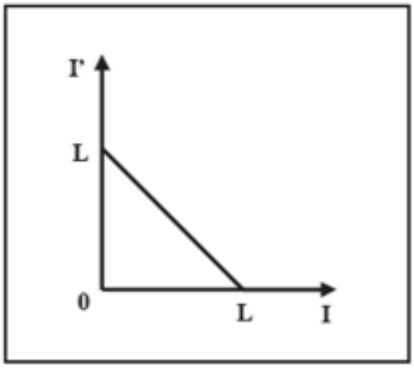
- x,y: συντεταγμένες σημείου
- I (x,y): η φωτεινότητά του στο διάστημα [0, 255]
- I´ (x,y): η φωτεινότητα του μετά την επεξεργασία στο διάστημα [0, 255]
- f: συνάρτηση που εφαρμόζεται σε αυτό

4.10.1 Αντιστροφή φωτεινότητας

Η διαδικασία αυτή συνήθως είναι χρήσιμη σε περίπτωση που οι λεπτομέρειες της αρχικής εικόνας δεν είναι τόσο εμφανείς. Η αντιστροφή εικόνας δίνει το αρνητικό της στην τελική μορφή της. Υπάρχουν δύο τρόποι που μπορεί να επιτευχθεί αυτό. Ο πρώτος είναι με τη χρήση της συνάρτησης **bitwise_not()**, η οποία αντιστρέφει κάθε bit στον πίνακα. Ο δεύτερος είναι η αφαίρεση μέγιστης τιμής φωτεινότητας μείων τη φωτεινότητα ενός σημείο, με χρήση της συνάρτησης **subtract()**. Πριν την αναλυτικότερη σύνταξη και εφαρμογή των συναρτήσεων, είναι σημαντικό να αναφερθεί η σχέση :

$$I´(x,y) = L - I(x,y)$$

όπου L η μέγιστη τιμής φωτεινότητας. **[1]**



Σχήμα 4.9: Αντιστροφή φωτεινότητας **[36]**

Η συνάρτηση **bitwise_not()** συντάσσεται ως εξής :
bitwise_not(InputArray src, OutputArray dst, InputArray mask=noArray()) όπου

- src : πίνακας εισόδου
- dst : πίνακας εξόδου που έχει ίδιο μέγεθος και τύπο με το src
- mask : μη απαραίτητη η χρήση της. Είναι ένας πίνακας ενός καναλιού, που έχει 8-bit στοιχεία, που στοιχεία του πίνακα εξόδου που πρέπει να αλλάξουν

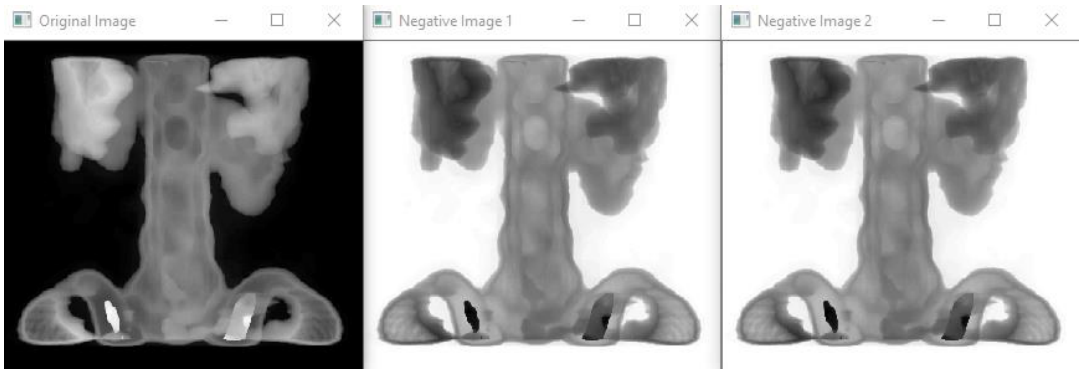
Η συνάρτηση subtract συντάσσεται ως εξής :
subtract(InputArray src1, InputArray src2, OutputArray dst, InputArray mask=noArray(), int dtype=-1) όπου

- src1: Πρώτος πίνακας εισόδου ή σταθερή τιμή
- src2: Δεύτερος πίνακας εισόδου ή σταθερή τιμή
- dst: Πίνακας εξόδου που έχει ίδιο μέγεθος και τύπο με το src1 και src2
- mask: Μη απαραίτητη η χρήση της. Είναι ένας πίνακας ενός καναλιού, που έχει 8-bit στοιχεία, που στοιχεία του πίνακα εξόδου που πρέπει να αλλάξουν
- dtype: Μη απαραίτητη η χρήση της. Δηλώνει το βάθος της εικόνας εξόδου

Παράδειγμα αντιστροφής φωτεινότητας

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(int, char**)
{
        Mat img = imread("C:/images/spine.tif",0);
        Mat img_inv, img_inv2;
        img.copyTo(img_inv); img.copyTo(img_inv);
        subtract(255, img, img_inv); // Antistrofh fwteinothtas me th subtract
        bitwise_not(img, img_inv2); // Antistrofh fwteinothtas me th bitwise_not

        // Emfanish twn eikonwn
        namedWindow("Original Image", CV_WINDOW_NORMAL);
        namedWindow("Negative Image 1", CV_WINDOW_NORMAL);
        namedWindow("Negative Image 2", CV_WINDOW_NORMAL);
        imshow("Original Image", img);
        imshow("Negative Image 1", img_inv);
```

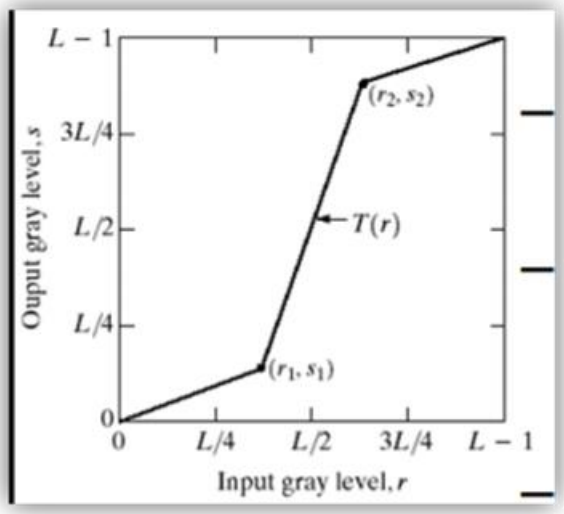
```
        imshow("Negative Image 2", img_inv2);

        waitKey(0);
        destroyAllWindows();
return 0;
}
```



Εικόνα 4.16: Αντιστροφή φωτεινότητας της εικόνας spine.tif

4.10.2 Επέκταση αντίθεσης

Η **επέκταση αντίθεσης** (ή κανονικοποίηση εικόνας) είναι μια τεχνική βελτίωσης μιας εικόνας και χρησιμοποιείται για την αύξηση του δυναμικό εύρος του γκρίζου επιπέδου σε αυτή, λόγω κακού φωτισμού. Συγκεκριμένα, επεκτείνει το εύρος τιμών φωτεινότητας που περιέχει, ώστε να καλύψει ένα επιθυμητό εύρος τιμών. Συνήθως δέχεται και επιστρέφει εικόνες γκρίζου επιπέδου. Αφού λοιπόν θεωρείται κανονικοποίηση θα χρησιμοποιηθεί η συνάρτηση normalize, της οποίας η σύνταξη αναφέρθηκε στην υποενότητα για την κανονικοποίηση ιστογράμματος (4.9.2.2). **[1] [38] [39]**



Σχήμα 4.10: Επέκταση αντίθεσης **[37]**

Τα σημεία (r1, s1) και (r2, s2) στο γράφημα, είναι υπεύθυνα για τον έλεγχο του σχήματος μεταμόρφωσης. Αν ισχύει ότι r1 = s1 και r2 = s2, τότε η επέκταση είναι γραμμική και δεν υπάρχει κάποια αλλαγή στην εικόνα. **[39]**

Παράδειγμα επέκτασης της αντίθεσης

Στο παρακάτω παράδειγμα διαβάζουμε την εικόνα pentagon.png και υπολογίζουμε το ιστόγραμμά της. Στη συνέχεια, εφαρμόζουμε επέκταση αντίθεσης και υπολογίζουμε εκ' νέου το ιστογράμμα. Τέλος, σχεδιάζουμε τα δυο ιστογράμματα και τα εμφανίζουμε μαζί με την αρχική και την τελική εικόνα.

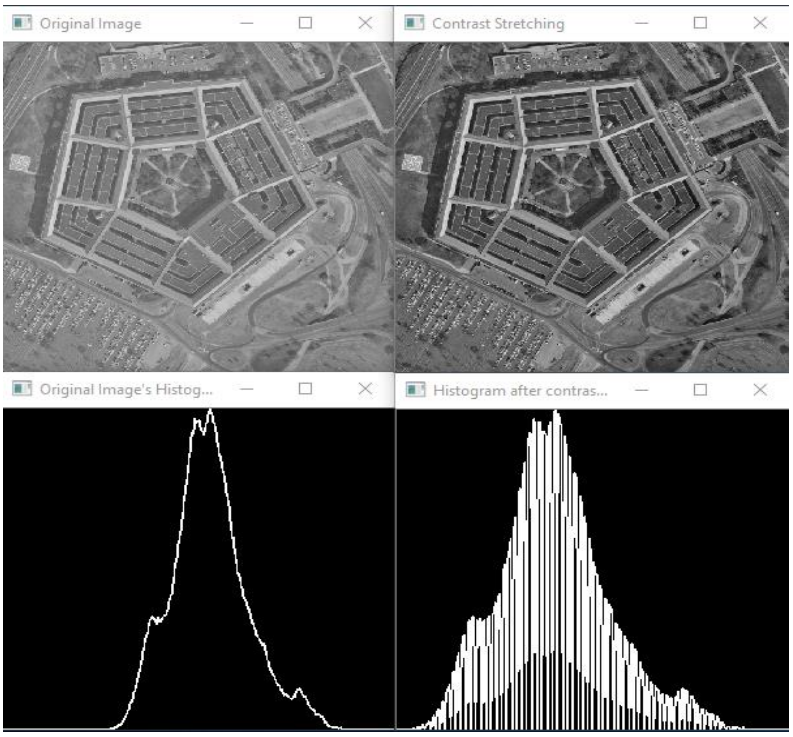
Σημείωση

Το bin size παίρνει τη τιμή 256, όταν υπάρχει η επιθυμία υπολογισμού του αριθμού των pixels για όλες τις φωτεινότητες.

Όταν όμως υπάρχει η επιθυμία υπολογισμού του αριθμού των pixels που έχουν φωτεινότητες από 0 έως 15, 16 έως 31, ... , 240 ως 255, το bin size παίρνει τη τιμή 16, όσο δηλαδή είναι ο αριθμός αυτών των ομάδων.

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(int, char**)
{
	Mat img = imread("C:/images/pentagon.png",0);
	Mat adj;
	img.copyTo(adj);

	// Arxikopoihsh parametrwn tou istogrammatos
	int histSize = 256;
	float range[] = { 0, 255 };
	const float *ranges[] = { range };

	// Ypologismos tou istogrammatos ths arxikhs eikonas
	MatND hist;
	calcHist( &img, 1, 0, Mat(), hist, 1, &histSize, ranges, true, false );

	// Epektash ths antitheshs
	normalize(adj, adj, 0, 255, NORM_MINMAX, -1, Mat());

	// Ypologismos tou istogrammatos ths telikis eikonas
	MatND hist_adj;
	calcHist( &adj, 1, 0, Mat(), hist_adj, 1, &histSize, ranges, true, false );

	// Sxediash twn istogrammatwn
	int hist_w = 512; int hist_h = 400;
	int bin_w = cvRound( (double) hist_w/histSize );
```

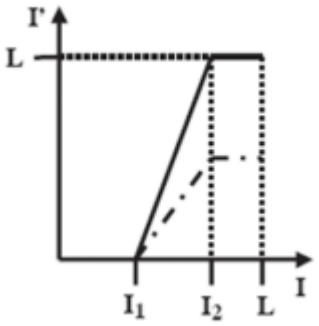
```
        Mat histImage( hist_h, hist_w, CV_8UC1, Scalar( 0,0,0) );
        Mat hist_adjImage( hist_h, hist_w, CV_8UC1, Scalar( 0,0,0) );
        normalize(hist, hist, 0, histImage.rows, NORM_MINMAX, -1, Mat() );
        normalize(hist_adj, hist_adj, 0, histImage.rows, NORM_MINMAX, -1, Mat() );

        for( int i = 1; i < histSize; i++ )
        {
        line(histImage, Point(bin_w*(i-1), hist_h - cvRound(hist.at<float>(i-1))),
                        Point(bin_w*(i), hist_h - cvRound(hist.at<float>(i))),
                        Scalar( 255, 0, 0), 2, 8, 0  );
        line(hist_adjImage,          Point(bin_w*(i-1),          hist_h          -
        cvRound(hist_adj.at<float>(i-1))),
                              Point(          bin_w*(i),          hist_h          -
        cvRound(hist_adj.at<float>(i)) ),
                              Scalar( 255, 0, 0), 2, 8, 0  );
        }

        // Emfanisi twn eikonwn kai twn istogrammatwn tous
        namedWindow("Original Image", CV_WINDOW_NORMAL);
        namedWindow("Contrast Stretching", CV_WINDOW_NORMAL);
        namedWindow("Original Image's Histogram", CV_WINDOW_NORMAL);
        namedWindow("Histogram after contrast stretching", CV_WINDOW_NORMAL);
        imshow("Original Image", img);
        imshow("Contrast Stretching", adj);
        imshow("Original Image's Histogram", histImage);
        imshow("Histogram after contrast stretching", hist_adjImage);

        waitKey(0);
        destroyAllWindows();
return 0;
}
```



Εικόνα 4.17: Επέκταση της αντίθεσης στην εικόνα pentagon.png

4.10.3 Κόψιμο φωτεινότητας

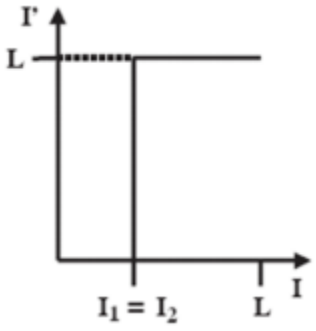
Το **κόψιμο φωτεινότητας** (Intensity clipping) θεωρείται μια μερική περίπτωση της επέκτασης αντίθεσης. Με βάση την παρακάτω εικόνα, αυτή η διαδικασία πραγματοποιείται, αν $I'_1 = 0$ και $I'_2 > I_2$. Ένα όμως, $I'_2 = L$, τότε αυτή η διαδικασία έχει παρόμοια χαρακτηριστικά με την επέκταση αντίθεσης χωρίς τη κλιμάκωση τιμών φωτεινότητας στο διάστημα $[I_1, I_2]$. Αν όμως υπάρχει αυτή η κλιμάκωση ενδιάμεσων τιμών, τότε υπάρχει και η επέκταση αντίθεσης. **[1] [36]**



Σχήμα 4.11: Κόψιμο επιπέδου φωτεινότητας **[36]**

4.10.3.1 Κατωφλίωση φωτεινότητας

Η **κατωφλίωση επιπέδων φωτεινότητας** (thresholding) είναι η πιο απλή μέθοδος τμηματοποίησης εικόνας. Θεωρείται κόψιμο φωτεινότητας κάτω από το επίπεδο κατωφλίου, που ορίζουμε εμείς. Χάρη σε αυτή τη διαδικασία, από μια μονοχρωματική εικόνα (grayscale) μπορούν να δημιουργηθούν δυαδικές εικόνες (binary images). Ουσιαστικά, για να ξεχωρίσουμε τα εικονοστοιχεία που θέλουμε από τα υπόλοιπα, γίνεται μια σύγκριση τιμής φωτεινότητας κάθε εικονοστοιχείου σε σχέση με ένα κατώφλι. Αφού γίνει αυτό, για την καλύτερη ανάγνωση των εικονοστοιχείων του ενδιαφέροντος μας, ορίζουμε μια τιμή (πχ. τιμή 0 για μαύρο χρώμα, τιμή 255 για λευκό). Στην παρακάτω εικόνα για τη κατωφλίωση ισχύει, $I'_1 = 0$ για $I < I_1 = I_2$ και $I'_2 = L$ για $I \geq I_1 = I_2$ (όπου L, μέγιστη δυνατή φωτεινότητα). **[36] [40]**



Σχήμα 4.12: Κατωφλίωση επιπέδου φωτεινότητας **[36]**

Για αυτό το σκοπό η OpenCV έχει τη συνάρτηση **threshold**, η οποία εφαρμόζει ένα σταθερό όριο κατωφλίου σε κάθε στοιχείο ενός πίνακα. Συντάσσεται ως εξής :

threshold(InputArray src, OutputArray dst, double thresh, double maxval, int type)
όπου:

- src: Πίνακας εισόδου (ενός καναλιού, κινητού σημείου των 8-bit ή 32-bit)
- dst: Πίνακας εξόδου που έχει ίδιο μέγεθος και τύπο με το src
- thresh: Τιμή κατωφλίωσης
- maxval: Μέγιστη τιμή φωτεινότητας που ορίζεται (χρήση των τύπων THRESH_BINARY και THRESH_BINARY INV)
- type : Τύπος κατωφλίωσης (Σχήμα 4.13)

- **THRESH_BINARY**

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- **THRESH_BINARY_INV**

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$$

- **THRESH_TRUNC**

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

- **THRESH_TOZERO**

$$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

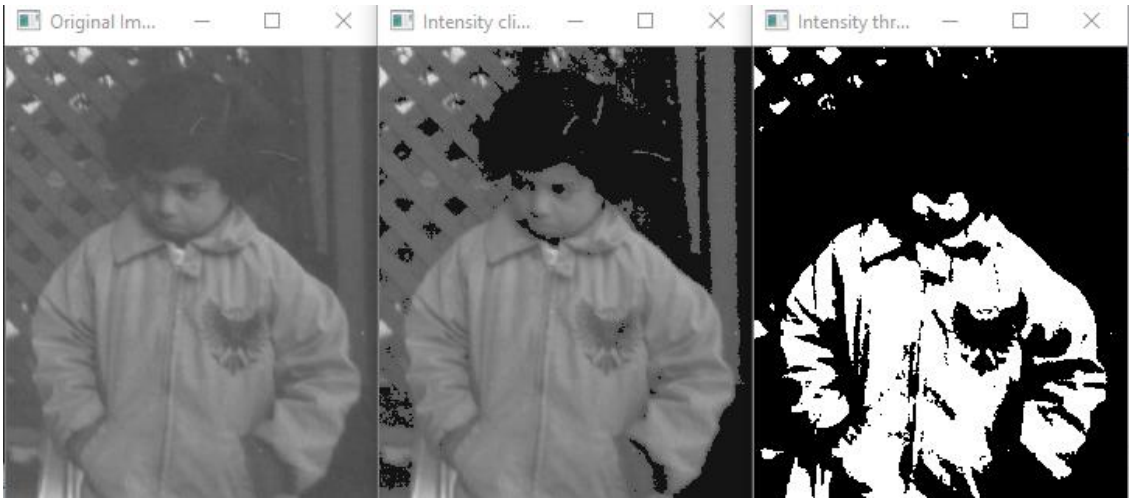
- **THRESH_TOZERO_INV**

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

Σχήμα 4.13: Τύποι κατωφλίωσης **[23]**

Παράδειγμα (κόψιμο και κατωφλίωση επιπέδων φωτεινότητας)

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(int, char**)
{
	Mat img = imread("C:/images/pout.tif",0);
	Mat clp,thresh;
	img.copyTo(clp);

	Mat mask;
	inRange(img, 0, 90, mask); // Euresh twn suntetagmenwn opou oi fwteinothtes
	einai <= 90

	clp.setTo(20, mask); // Kopsimo fwteinothtas
	threshold(img,thresh,127,255,THRESH_BINARY); // Katwfliwsh fwteinothtas

	// Emfanish twn eikonwn
	namedWindow("Original Image", CV_WINDOW_AUTOSIZE);
	namedWindow("Intensity clipping", CV_WINDOW_AUTOSIZE);
	namedWindow("Intensity thresholding", CV_WINDOW_AUTOSIZE);

	imshow("Original Image", img);
	imshow("Intensity clipping", clp);
	imshow("Intensity thresholding", thresh);

	waitKey(0);
	destroyAllWindows();
return 0;
}
```



Εικόνα 4.18: Κόψιμο και κατωφλίωση φωτεινότητας της εικόνας pout.tif

4.10.4 Τεμαχισμός κλίμακας φωτεινότητας

Τεμαχισμός κλίμακας φωτεινότητας (intensity scale slicing) ονομάζεται η απεικόνιση μιας επιθυμητής περιοχής φωτεινότητας σε μια εικόνα. Στη παρακάτω εικόνα φαίνονται οι δύο τρόποι για να επιτευχθεί αυτή η διαδικασία. Ο πρώτος (αριστερά), παίρνει όλα τα στοιχεία στο διάστημα [a,b] και τους ορίζει τη μέγιστη τιμή φωτεινότητας. Τα στοιχειά που δε βρίσκονται σε αυτό το διάστημα παίρνουν την τιμή μηδέν, δηλαδή:

Αν $a \leq I \leq b$, τότε $I'(x,y) = L$

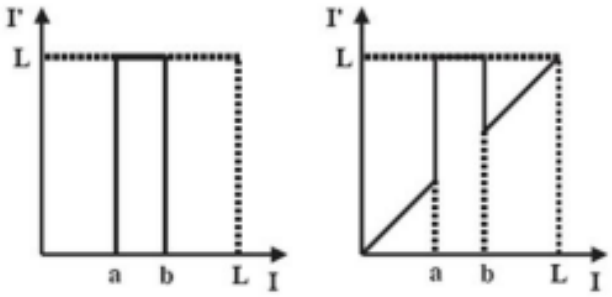
Αλλιώς $I'(x,y) = 0$

Ο δεύτερος (δεξιά), παίρνει μόνο μια περιοχή στο διάστημα [a,b] και της ορίζει τη μέγιστη τιμή φωτεινότητας. Τα στοιχειά που δε βρίσκονται σε αυτή τη περιοχή μένουν ίδια, δηλαδή:

Αν $a \leq I \leq b$, τότε $I'(x,y) = L$

Αλλιώς $I'(x,y) = I$

[1]

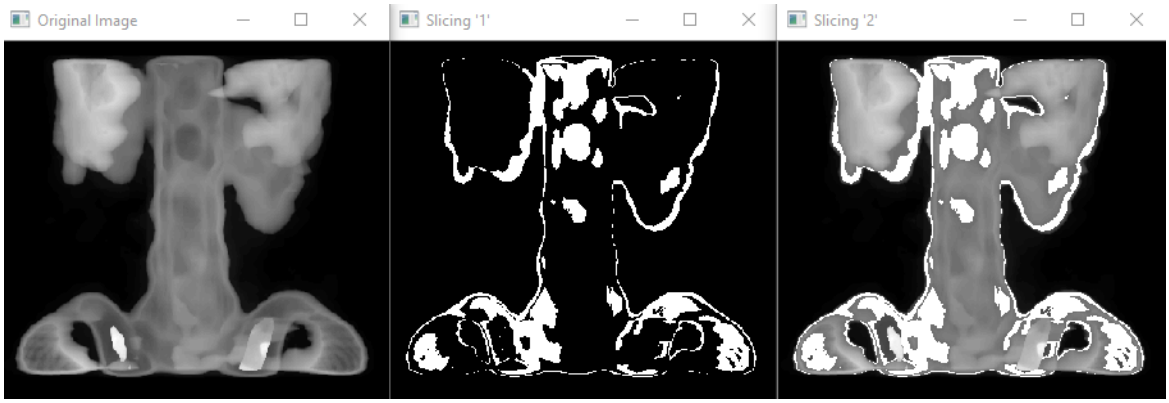


Σχήμα 4.14: Τεμαχισμός φωτεινότητας εικόνας **[36]**

Παράδειγμα τεμαχισμού της κλίμακας φωτεινότητας

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main()
{
	Mat img = imread("C:/images/spine.tif",0);
	Mat sl1, sl2;
	img.copyTo(sl1);
	img.copyTo(sl2);

	Mat mask;
	// Evresh suntetagmenwn twn pixels me fwteinothtes sto euros [a,b]
```

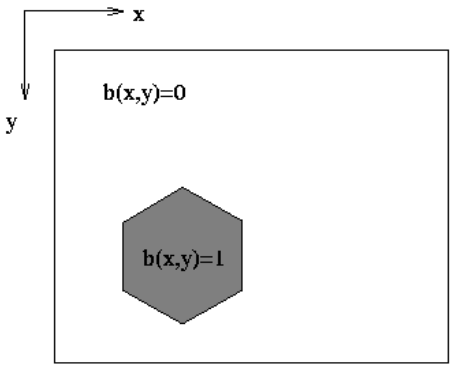
```
        inRange(img, 80, 110, mask);

        // Slicing '1'
        sl1.setTo(0); // Oi fwteinothtes ektos tou [a,b] ginontai 0
        sl1.setTo(255, mask); // Oi fwteinothtes entos tou [a,b] ginontai 255

        // Slicing '2'
        sl2.setTo(255, mask); // Oi fwteinothtes entos tou [a,b] ginontai 255
                              // Oi upoloipes menoun ws exoun

        // Emfanish twn eikonwn
        namedWindow("Original Image", CV_WINDOW_NORMAL);
        namedWindow("Slicing '1'", CV_WINDOW_NORMAL);
        namedWindow("Slicing '2'", CV_WINDOW_NORMAL);

        imshow("Original Image", img);
        imshow("Slicing '1'", sl1);
        imshow("Slicing '2'", sl2);

        waitKey(0);
        destroyAllWindows();
return 0;
}
```



Εικόνα 4.19: Τεμαχισμός της κλίμακας φωτεινότητας της εικόνας spine.tif

4.10.5 Ανάλυση σε δυαδικές εικόνες

Λαμβάνοντας υπόψη πως μια μονοχρωματική εικόνα (grayscale), αποτελείται από εικονοστοιχεία των 8 bits, καταλαβαίνουμε πως αυτή η εικόνα με τη σειρά της μπορεί να αποτελείται από 8 δυαδικές εικόνες (binary planes). Όπως φαίνεται παρακάτω μια δυαδική εικόνα έχει δύο τιμές : 0 (για τα σημεία πίσω από την εικόνα, μαύρο χρώμα) ή 1(για τα σημεία της εικόνας, λευκό χρώμα). **[41]**

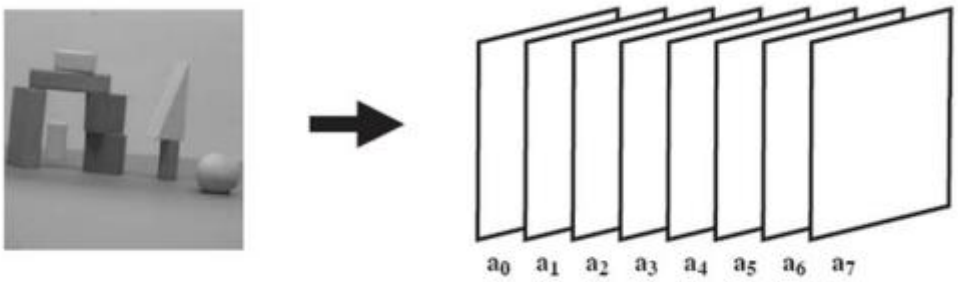


Σχήμα 4.15: Δυαδική εικόνα **[41]**

Κάθε pixel μπορεί να αναπαρασταθεί, χάρη στη σχέση :

$$I(x,y) : a_7 2^7 + a_6 2^6 + a_5 2^5 + a_4 2^4 + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0$$

όπου $a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7$: δυαδικά βάρη, τα οποία περιέχουν τα bit planes **[1]**



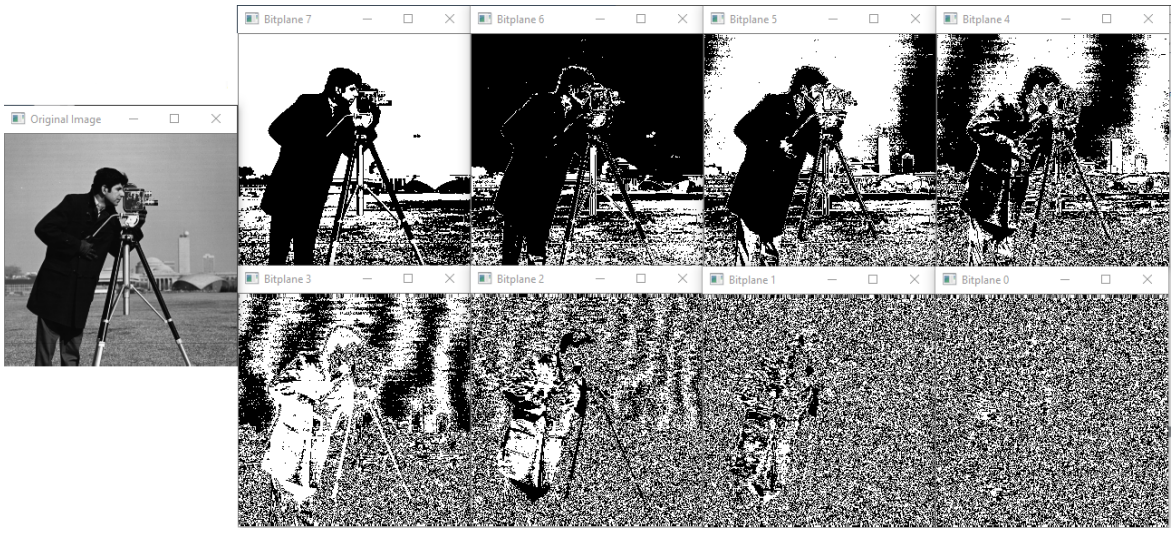
Σχήμα 4.16: Ανάλυση μονοχρωματικής εικόνας σε 8 δυαδικές εικόνες, η λιγότερο σημαντική θεωρείται αυτή που έχει βάρος a_0 **[36]**

Παράδειγμα ανάλυσης εικόνας σε δυαδικές

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>


using namespace std;
using namespace cv;


// Sunarthsh gia thn analush monoxrwmatikhs eikonas se duadikes
// To x dhlwnei ta duadika varh kai exei tis exhs times:
// x = 1 -> bit0, x = 2 -> bit1, x = 4 -> bit2, x = 8 -> bit3
// x = 16 -> bit4, x = 32 -> bit5, x = 64 -> bit6, x = 128 -> bit7
// Efarmozoume se kathe pixel bitwise_and me to x.
// An to apotelesma einai iso me x tote tote to bit tou pixel pou antiproswpeyei to
// x einai 1 kai etsi thetoume sto antistoixo pixel tou bitplane th timh 255
```

```
// An to apotelesma einai iso me 0 tote tote to bit tou pixel pou antiproswpeyei to
// x einai 0 kai etsi thetoume sto antistoixo pixel tou bitplane th timh 0


Mat bitplane(Mat image, Mat bitplane, int x)
{

Mat mask;

bitwise_and(image,x,bitplane);    // Euresh twn fwteinothtwn pou exoun timh >= x
                                  // kai apodosh timhs ish me x
inRange(bitplane, x, x, mask);  // Euresh twn suntetagmenwn twn pixels me timh = x
bitplane.setTo(255, mask);     // kai apodosh timhs ish me 255

return bitplane;

}
int main(int, char**)
{
        Mat img = imread("C:/images/cameraman.tif",0);
        int width = img.cols; int height = img.rows;
        int b0 = 1; // varos tou bit0
        int b1 = 2; // varos tou bit1
        int b2 = 4; // varos tou bit2
        int b3 = 8; // varos tou bit3
        int b4 = 16; // varos tou bit4
        int b5 = 32; // varos tou bit5
        int b6 = 64; // varos tou bit6
        int b7 = 128; // varos tou bit7

        // Dhmiourgia 8 mavrwn eikonwn gia tis duadikes eikones
        Mat bp7(width, height, CV_8UC1, Scalar(0));
        Mat bp6(width, height, CV_8UC1, Scalar(0));
        Mat bp5(width, height, CV_8UC1, Scalar(0));
        Mat bp4(width, height, CV_8UC1, Scalar(0));
        Mat bp3(width, height, CV_8UC1, Scalar(0));
        Mat bp2(width, height, CV_8UC1, Scalar(0));
        Mat bp1(width, height, CV_8UC1, Scalar(0));
        Mat bp0(width, height, CV_8UC1, Scalar(0));

        // Klish ths sunartishs gia th dhmiourgia twn duadikwn eikonwn
        bitplane(img,bp0,b0); bitplane(img,bp1,b1); bitplane(img,bp2,b2);
        bitplane(img,bp3,b3); bitplane(img,bp4,b4); bitplane(img,bp5,b5);
        bitplane(img,bp6,b6); bitplane(img,bp7,b7);

        // Emfanish twn duadikwn eikonwn
        namedWindow("Bitplane 7", CV_WINDOW_AUTOSIZE);
        namedWindow("Bitplane 6", CV_WINDOW_AUTOSIZE);
        namedWindow("Bitplane 5", CV_WINDOW_AUTOSIZE);
        namedWindow("Bitplane 4", CV_WINDOW_AUTOSIZE);
        namedWindow("Bitplane 3", CV_WINDOW_AUTOSIZE);
        namedWindow("Bitplane 2", CV_WINDOW_AUTOSIZE);
        namedWindow("Bitplane 1", CV_WINDOW_AUTOSIZE);
        namedWindow("Bitplane 0", CV_WINDOW_AUTOSIZE);

        imshow("Bitplane 7", bp7); imshow("Bitplane 6", bp6);
        imshow("Bitplane 5", bp5); imshow("Bitplane 4", bp4);
        imshow("Bitplane 3", bp3); imshow("Bitplane 2", bp2);
        imshow("Bitplane 1", bp1); imshow("Bitplane 0", bp0);

        waitKey(0);
        destroyAllWindows();
```

```
return 0;
}
```



Εικόνα 4.20: Ανάλυση της εικόνας cameraman.tif σε δυαδικές

4.10.6 Πράξεις επί των εικόνων

Οι πράξεις επί των εικόνων είναι χρήσιμες σε αρκετές εφαρμογές. Στις παρακάτω υποενότητες παρουσιάζονται οι πιο βασικές από αυτές.

4.10.6.1 Πρόσθεση

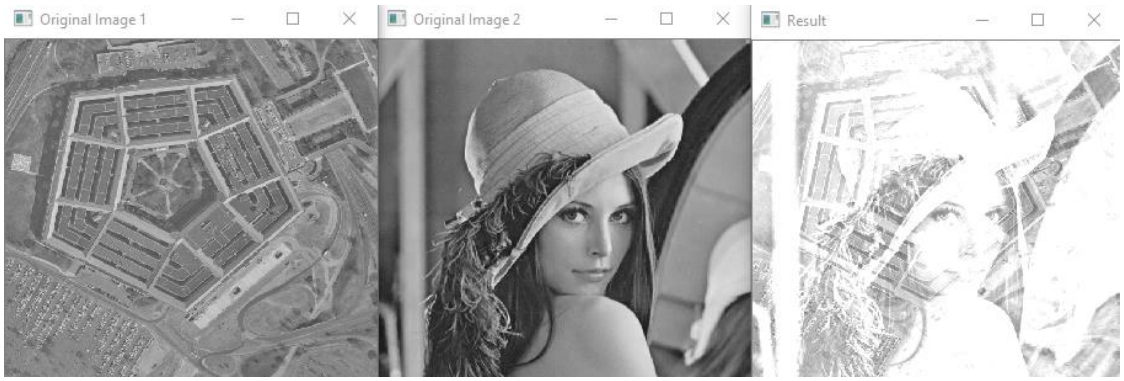
Η OpenCV διαθέτει τη συνάρτηση **add()**, για την υπολογισμό αθροίσματος δύο εικόνων. Συντάσσεται ως εξής :

add(InputArray src1, InputArray src2, OutputArray dst, InputArray mask=noArray(), int dtype=-1) όπου

- src1: Πρώτος πίνακας εισόδου
- src2: Δεύτερος πίνακας εισόδου
- dst: Πίνακας εξόδου που έχει ίδιο μέγεθος και αριθμό καναλιών με τους src1 και src2
- mask: Μη απαραίτητη η χρήση της. Είναι ένας πίνακας ενός καναλιού, που έχει 8-bit στοιχεία, που στοιχεία του πίνακα εξόδου που πρέπει να αλλάξουν
- dtype: Μη απαραίτητη η χρήση του. Είναι αυτό που ορίζει το βάθος, σε περίπτωση που δε το ορίζει το src1 και το src2 για τον πίνακα εξόδου

Παράδειγμα

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(int, char**)
{
        Mat img1 = imread("C:/images/pentagon.png",0);
        Mat img2 = imread("C:/images/Lenna.jpg",0);
        Mat res;

        add(img1, img2, res);

        namedWindow("Original Image 1", CV_WINDOW_NORMAL);
        namedWindow("Original Image 2", CV_WINDOW_NORMAL);
        namedWindow("Result", CV_WINDOW_NORMAL);

        imshow("Original Image 1", img1);
        imshow("Original Image 2", img2);
        imshow("Result", res);

        waitKey(0);
        destroyAllWindows();
return 0;
}
```



Εικόνα 4.21: Πρόσθεση δυο εικόνων

4.10.6.2 Πρόσθεση με ειδικά βάρη

Η συνάρτηση **addWeighted** είναι αυτή που χρησιμοποιείται στο άθροισμα δύο εικόνων με ειδικά βάρη. Συντάσσεται ως εξής :

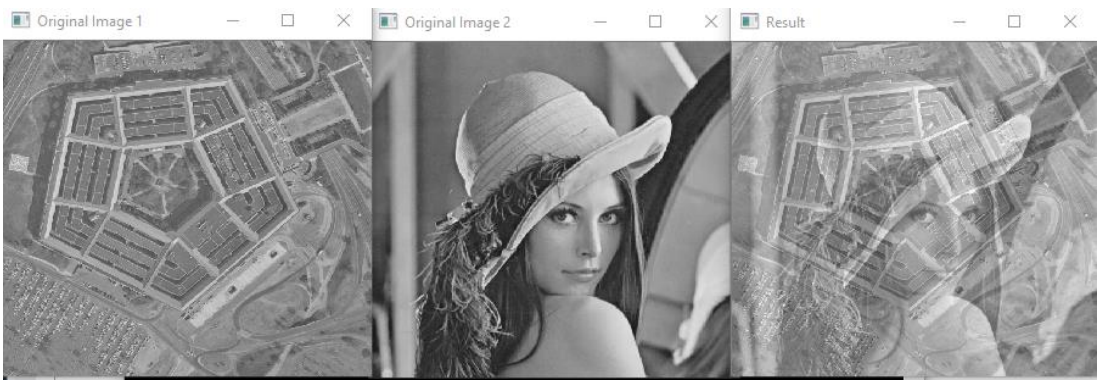
addWeighted(InputArray src1, double alpha, InputArray src2, double beta, double gamma, OutputArray dst, int dtype=-1) όπου:

- src1: Πρώτος πίνακας εισόδου
- alpha: Το βάρος των στοιχείων του πρώτου πίνακα

- src2: Δεύτερος πίνακας εισόδου που έχει ίδιο μέγεθος και αριθμό καναλιών με τους src1
- beta: Το βάρος των στοιχείων του δεύτερου πίνακα
- dst : Πίνακας εξόδου που έχει ίδιο μέγεθος και αριθμό καναλιών με τους src1 και src2
- gamma: Κλιμακωτή τιμή που υπολογίζεται μαζί με τη πρόσθεση
- dtype: Μη απαραίτητη η χρήση του. Σε περίπτωση που το src1 και το src2 έχουν ίδιο βάθος, το dtype παίρνει συνήθως τη τιμή -1.

Παράδειγμα

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(int, char**)
{
        Mat img1 = imread("C:/images/pentagon.png",0);
        Mat img2 = imread("C:/images/Lenna.jpg",0);
        Mat res;

        addWeighted(img1, 0.7, img2, 0.5, 0, res);

        namedWindow("Original Image 1", CV_WINDOW_NORMAL);
        namedWindow("Original Image 2", CV_WINDOW_NORMAL);
        namedWindow("Result", CV_WINDOW_NORMAL);

        imshow("Original Image 1", img1);
        imshow("Original Image 2", img2);
        imshow("Result", res);

        waitKey(0);
        destroyAllWindows();
return 0;
}
```



Εικόνα 4.22: Πρόσθεση δυο εικόνων με ειδικά βάρη

4.10.6.3 Αφαίρεση

Για την διαφορά δύο πινάκων υπάρχει η συνάρτηση **subtract()**. Συντάσσεται ως εξής :

subtract(InputArray src1, InputArray src2, OutputArray dst, InputArray mask=noArray(), int dtype=-1) όπου

- src1: Πρώτος πίνακας εισόδου
- src2: Δεύτερος πίνακας εισόδου
- dst: Πίνακας εξόδου που έχει ίδιο μέγεθος και αριθμό καναλιών με τους src1 και src2
- mask: Μη απαραίτητη η χρήση της. Είναι ένας πίνακας ενός καναλιού, που έχει 8-bit στοιχεία, που στοιχεία του πίνακα εξόδου που πρέπει ν αλλάξουν
- dtype: Μη απαραίτητη η χρήση του. Είναι αυτό που ορίζει το βάθος στον πίνακα εξόδου

Παράδειγμα

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(int, char**)
{
        Mat img1 = imread("C:/images/caman.tif",0);
        Mat img2 = imread("C:/images/cameraman.tif",0);
        Mat res;

        subtract(img1, img2, res);

        namedWindow("Original Image 1", CV_WINDOW_NORMAL);
        namedWindow("Original Image 2", CV_WINDOW_NORMAL);
        namedWindow("Result", CV_WINDOW_NORMAL);

        imshow("Original Image 1", img1);
        imshow("Original Image 2", img2);
        imshow("Result", res);

        waitKey(0);
        destroyAllWindows();
return 0;
}
```

Εικόνα 4.23: Αφαίρεση δυο εικόνων

4.10.6.4 Λογική πράξη AND

Η λογική πράξη AND στην OpenCV, γίνεται με τη συνάρτηση **bitwise_and()**, και εφαρμόζεται κυρίως στο κομμάτι του roi (region of interest). Συγκερκιμένα, αυτή η πράξη χρησιμοποιείται και ως μάσκα, για την να απομονωθεί μια περιοχή της εικόνας ή των συντελεστών μετασχηματισμού. Συντάσσεται ως εξής:

bitwise_and(InputArray src1, InputArray src2, OutputArray dst, InputArray mask=noArray()) όπου

- src1 : Πρώτος πίνακας εισόδου
- src2 : Δεύτερος πίνακας εισόδου
- dst : Πίνακας εξόδου που έχει ίδιο μέγεθος και αριθμό καναλιών με τους src1 και src2
- mask : Μη απαραίτητη η χρήση της. Είναι ένας πίνακας ενός καναλιού, που έχει 8-bit στοιχεία, που στοιχεία του πίνακα εξόδου που πρέπει να αλλάξουν

Παράδειγμα

Στον παρακάτω κώδικα υπάρχουν μία εικόνα και μία μάσκα ιδίων διαστάσεων με την αρχική εικόνα και με όλα τα pixels μαύρα εκτός από το κομμάτι που αντιστοιχεί στις συντεταγμένες του roi που είναι άσπρο.

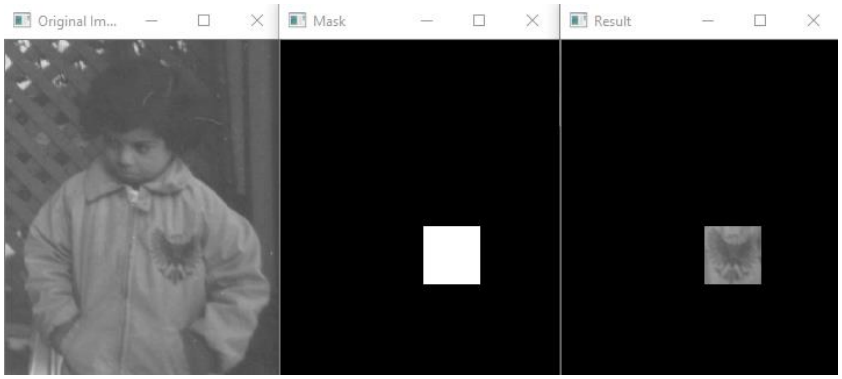
Όταν εφαρμοστεί AND στις δύο αυτές εικόνες ισχύει το εξής:

0 and x = 0

1 and x = x

Επομένως το αποτέλεσμα που θα προκύψει θα ναι η μάσκα όπου αντί του λευκού κομματιού θα υπάρχει το επιθυμητό roi.

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main()
{
    Mat img = imread("C:/images/pout.tif",0);
       cout<< img.depth();
       int w = img.cols; int h = img.rows;
       int i, j;
       Mat mask, res;
       img.copyTo(mask); img.copyTo(res);
       mask.setTo(0); res.setTo(0);
       for (i = 123; i < 172; i++){
              for (j = 160; j < 210; j++){
                     mask.at<uchar>(j,i) = 255;
              }
       }

       bitwise_and(img, mask, res);

       namedWindow("Original Image", CV_WINDOW_NORMAL);
       namedWindow("Mask", CV_WINDOW_NORMAL);
       namedWindow("Result", CV_WINDOW_NORMAL);

       imshow("Original Image", img);
       imshow("Mask", mask);
       imshow("Result", res);

       waitKey(0);
       destroyAllWindows();
return 0;
}
```



Εικόνα 4.24: Πράξη AND μεταξύ δυο εικόνων

4.10.6.5 Λογική πράξη OR

Η λογική πράξη OR στην OpenCV, γίνεται με τη συνάρτηση **bitwise_or()**, και εφαρμόζεται κυρίως στο κομμάτι του roi (region of interest). Αυτή η πράξη

χρησιμοποιείται και ως μάσκα, για την να εξαιρεθεί μια περιοχή της εικόνας ή των συντελεστών μετασχηματισμού. Συντάσσεται ως εξής:

bitwise_or(InputArray src1, InputArray src2, OutputArray dst, InputArray mask=noArray()) όπου

- src1: Πρώτος πίνακας εισόδου
- src2: Δεύτερος πίνακας εισόδου
- dst: Πίνακας εξόδου που έχει ίδιο μέγεθος και αριθμό καναλιών με τους src1 και src2
- mask: Μη απαραίτητη η χρήση της. Είναι ένας πίνακας ενός καναλιού, που έχει 8-bit στοιχεία, που στοιχεία του πίνακα εξόδου που πρέπει να αλλάξουν

Παράδειγμα

Στον παρακάτω κώδικα υπάρχουν μία εικόνα και μία μάσκα ιδίων διαστάσεων με την αρχική εικόνα και με όλα τα pixels μαύρα εκτός από το κομμάτι που αντιστοιχεί στις συντεταγμένες του roi που είναι άσπρο.

Όταν εφαρμοστεί OR στις δύο αυτές εικόνες ισχύει το εξής:

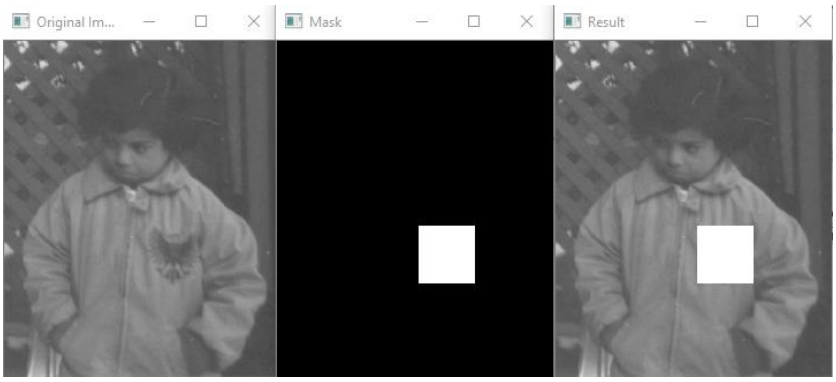
0 or x = x

1 or x = 1

Επομένως το αποτέλεσμα που θα προκύψει θα ναι η αρχική εικόνα όπου αντί του η περιοχή του roi θα χει αντικατασταθεί από λεύκα pixels.

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main()
{
    Mat img = imread("C:/images/pout.tif",0);
       cout<< img.depth();
       int w = img.cols; int h = img.rows;
       int i, j;
       Mat mask, res;
       img.copyTo(mask); img.copyTo(res);
       mask.setTo(0); res.setTo(0);
       for (i = 123; i < 172; i++){
              for (j = 160; j < 210; j++){
                     mask.at<uchar>(j,i) = 255;
```

```
            }
        }

        bitwise_or(img, mask, res);

        namedWindow("Original Image", CV_WINDOW_NORMAL);
        namedWindow("Mask", CV_WINDOW_NORMAL);
        namedWindow("Result", CV_WINDOW_NORMAL);

        imshow("Original Image", img);
        imshow("Mask", mask);
        imshow("Result", res);

        waitKey(0);
        destroyAllWindows();
return 0;
}
```



Εικόνα 4.25: Πράξη OR μεταξύ δυο εικόνων

4.10.7 Εξισορρόπηση ιστογράμματος

Η εξισορρόπηση ιστογράμματος (Histogram equalization) είναι υπεύθυνη για τη βελτίωση της αντίθεσης μιας εικόνας, με σκοπό να γίνει επέκταση του εύρους φωτεινότητας. Η βιβλιοθήκη παρέχει τη συνάρτηση για αυτή τη διαδικασία (για μονοχρωματική εικόνα), που ονομάζεται **equalizeHist()**. Συντάσσεται ως εξής :

equalizeHist(InputArray src, OutputArray dst) όπου

- src : Εικόνα εισόδου (ενός καναλιού, των 8-bit)
- dst : Εικόνα εξόδου που έχει ίδιο μέγεθος και τύπο με το src

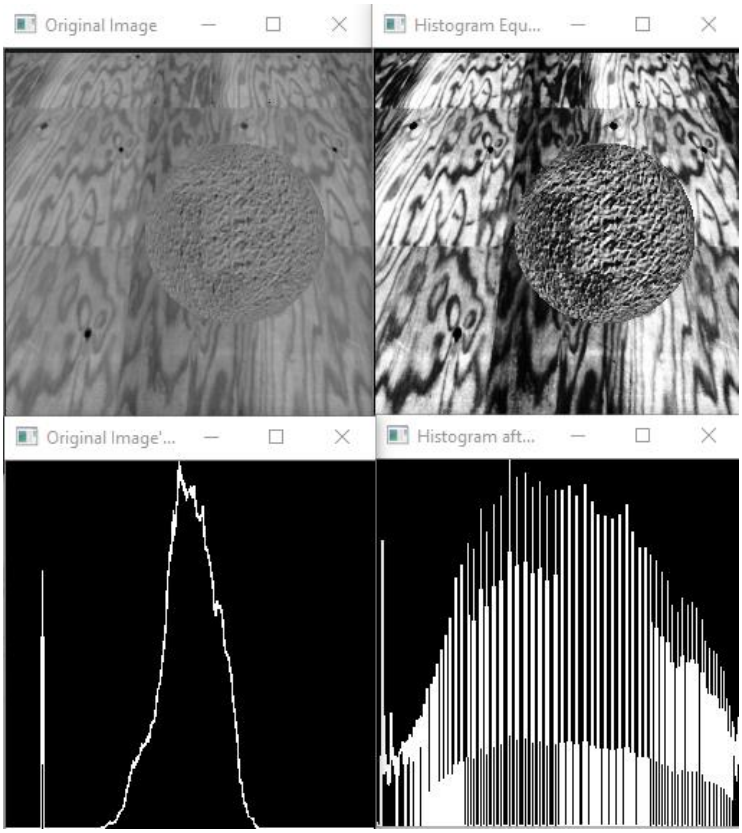
Παράδειγμα

Στο παρακάτω παράδειγμα διαβάζουμε την εικόνα sphere.png και υπολογίζουμε το ιστόγραμμά της. Στη συνέχεια, εφαρμόζουμε εξισορρόπηση ιστογράμματος και

υπολογίζουμε εκ' νέου το ιστογράμμα. Τέλος, σχεδιάζουμε τα δυο ιστογράμματα και τα εμφανίζουμε μαζί με την αρχική και την τελική εικόνα

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace std;
using namespace cv;

int main(int, char**)
{
	Mat img = imread("C:/images/sphere.png",0);
	Mat equ;
	img.copyTo(equ);

	// Arxikopoihsh parametrwn
	int histSize = 256;
	float range[] = { 0, 255 };
	const float *ranges[] = { range };

	// Ypologismos tou istogrammatos ths arxikhs eikonas
	MatND hist;
	calcHist( &img, 1, 0, Mat(), hist, 1, &histSize, ranges, true, false );

	// Exisorropish istogrammatos
	equalizeHist(img, equ);

	// Ypologismos tou istogrammatos ths neas eikonas
	MatND hist_equ;
	calcHist( &equ, 1, 0, Mat(), hist_equ, 1, &histSize, ranges, true, false );

	// Sxediash twn istogrammatwn
	int hist_w = 512; int hist_h = 400;
	int bin_w = cvRound( (double) hist_w/histSize );

	Mat histImage( hist_h, hist_w, CV_8UC1, Scalar( 0,0,0) );
	Mat hist_equImage( hist_h, hist_w, CV_8UC1, Scalar( 0,0,0) );
	normalize(hist, hist, 0, histImage.rows, NORM_MINMAX, -1, Mat() );
	normalize(hist_equ, hist_equ, 0, histImage.rows, NORM_MINMAX, -1, Mat() );

	for( int i = 1; i < histSize; i++ )
	{
	line( histImage, Point(bin_w*(i-1), hist_h - cvRound(hist.at<float>(i-1))),
	                 Point( bin_w*(i), hist_h - cvRound(hist.at<float>(i)) ),
	                 Scalar( 255, 0, 0), 2, 8, 0  );
	line( hist_equImage, Point(bin_w*(i-1), hist_h -
cvRound(hist_equ.at<float>(i-1)) ), Point( bin_w*(i), hist_h -
cvRound(hist_equ.at<float>(i)) ), Scalar( 255, 0, 0), 2, 8, 0  );
    }
	// Emfanish twn eikonwn kai twn istogrammatwn tous
	namedWindow("Original Image", CV_WINDOW_NORMAL);
	namedWindow("Histogram Equalization", CV_WINDOW_NORMAL);
	namedWindow("Original Image's Histogram", CV_WINDOW_NORMAL);
	namedWindow("Histogram after histogram equalization", CV_WINDOW_NORMAL);

	imshow("Original Image", img);
	imshow("Histogram Equalization", equ);
	imshow("Original Image's Histogram", histImage);
	imshow("Histogram after histogram equalization", hist_equImage);
```

```
        waitKey(0);
        destroyAllWindows();
return 0;
}
```



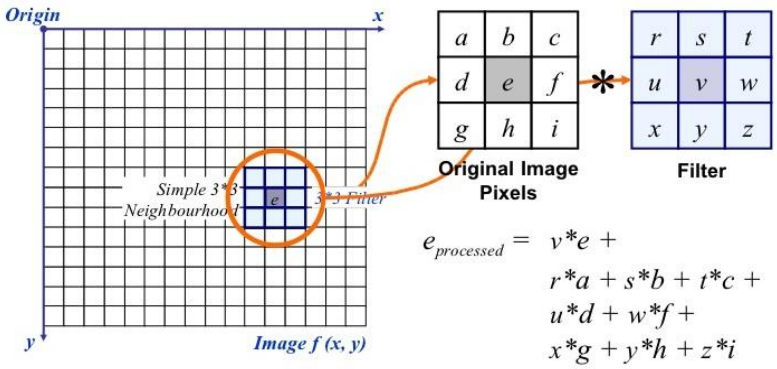
Εικόνα 4.26: Εξισορρόπηση ιστογράμματος στην εικόνα sphere.png

4.11 Τεχνικές χώρου

Μια **τεχνική χώρου**, ή αλλιώς φιλτράρισμα, μεταβάλλει τις τιμές των εικονοστοιχείων μιας εικόνας χρησιμοποιώντας έναν αλγόριθμο (φίλτρο) που επενεργεί στις γειτονίες των. Τα φίλτρα χωρίζονται σε δυο κατηγορίες: τα γραμμικά και τη μη γραμμικά. Τόσο στα γραμμικά όσο και στα μη γραμμικά φίλτρα, οι τιμές των εικονοστοιχείων μιας εικόνας μεταβάλλονται με χρήση μιας μάσκας, όπου στο κέντρο της τοποθετείται το εικονοστοιχείο του οποίου η τιμή πρόκειται να μεταβληθεί. Η διαφορά τους έχει να κάνει με τη γραμμικότητα ή μη του χρησιμοποιούμενου αλγορίθμου. Χρησιμοποιούνται για θόλωμα μιας εικόνας (blurring), ανάγλυφη εκτύπωση (embossing), ανίχνευση ακμών (edge detection), απαλοιφή θορύβου (denoising) κλπ. **[1] [42]**

4.11.1 Εφαρμογή γραμμικών φίλτρων

Κατά την εφαρμογή ενός γραμμικού φίλτρου σε μια εικόνα, η μεταβολή κάθε εικονοστοιχείου είναι ίση με το άθροισμα των γινομένων των συντελεστών της μάσκας με τα αντίστοιχα εικονοστοιχεία της εικόνας. **[1]**



Σχήμα 4.17: Εφαρμογή γραμμικού φίλτρου **[43]**

Η συνάρτηση με την οποία γίνεται εφαρμογή ενός γραμμικού φίλτρου σε μια εικόνα είναι η **filter2D()**. Συντάσσεται ως εξής:

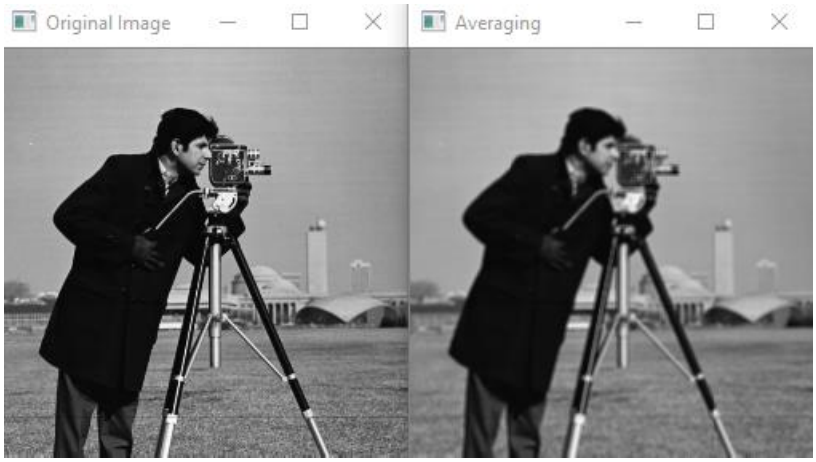
filter2D(InputArray src, OutputArray dst, int ddepth, InputArray kernel, Point anchor=Point(-1,-1), double delta=0, int borderType=BORDER_DEFAULT) όπου

- src: Εικόνα εισόδου.
- dst: Εικόνα εξόδου με ίδιο μέγεθος και τύπο δεδομένων με την εικόνα εισόδου
- ddepth: Τύπος δεδομένων της εικόνας εξόδου. Με τιμή -1 είναι ίδιος με το τύπο δεδομένων της εικόνας εισόδου (src.depth()). Ισχύουν τα εξής:
    - src.depth() = CV_8U, ddepth = -1/CV_16S/CV_32F/CV_64F
    - src.depth() = CV_16U/CV_16S, ddepth = -1/CV_32F/CV_64F
    - src.depth() = CV_32F, ddepth = -1/CV_32F/CV_64F
    - src.depth() = CV_64F, ddepth = -1/CV_64F
- kernel: Μάσκα του γραμμικού φίλτρου (μέσης τιμής, πρώτης/ δεύτερης παραγώγου κλπ)
- anchor: Θέση του pixel στο οποίο θα εφαρμοστεί το φίλτρο. Η εξ' ορισμού τιμή (-1,-1) δηλώνει ότι η θέση του pixel βρίσκεται στο κέντρο της μάσκας
- delta: Προαιρετική σταθερά που προστίθεται στο αποτέλεσμα της εφαρμογής του φίλτρου.
- borderType: Τύπος επέκτασης της εικόνας. Δέχεται τις εξής τιμές:
    - BORDER_REPLICATE

- BORDER_REFLECT
- BORDER_REFLECT_101
- BORDER_WRAP
- BORDER_CONSTANT

Παράδειγμα εφαρμογής της συνάρτησης filter2D()

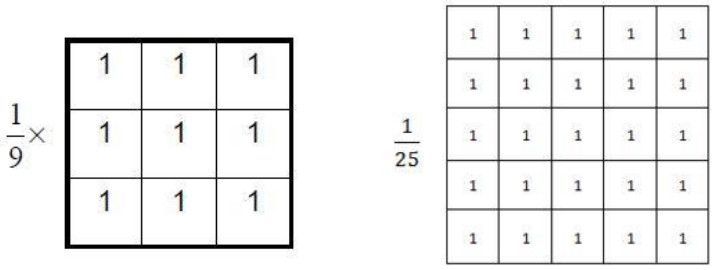
```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{

	Mat img = imread("C:/images/cameraman.tif", 0);
	Mat res1;
	// Dimiourgia tou filtrou mesis timis 3x3
	Mat averaging = Mat::ones(3,3,CV_32F)/9;

	// Efarmogi tou filtrou stin eikona
	filter2D(img,res1,-1,averaging);

	// Emfanisi ton eikonon
	namedWindow("Original Image" , CV_WINDOW_NORMAL);
	namedWindow("Averaging" , CV_WINDOW_NORMAL);

	imshow("Original Image", img);
	imshow("Averaging", res1);

	waitKey(0);
	destroyAllWindows();
return 0;
}
```



Εικόνα 4.27: Εφαρμογή της συνάρτησης filter2D()

4.11.2 Φίλτρο μέσης τιμής

Το φίλτρο μέσης τιμής (averaging or mean filter) αποτελεί ένα από τα γραμμικά φίλτρα και χρησιμοποιείται κατά κύριο λόγο για την εξομάλυνση μιας εικόνας, δηλαδή τη μείωση της διαφοράς φωτεινότητας ενός pixel και του επόμενου του. Κατά την εφαρμογή του φίλτρου μέσης τιμής σε μια εικόνα, το pixel στο οποίο εφαρμόζεται το φίλτρο αντικαθιστάται από το μέσο όρο των γειτονικών pixels. **[1] [44]**



Σχήμα 4.18: Φίλτρο μέσης τιμής 3x3 και 5x5 **[45] [46]**

Η συνάρτηση με την οποία γίνεται εφαρμογή του φίλτρου μέσης τιμής σε μια εικόνα είναι η **blur()** και συντάσσεται ως εξής:

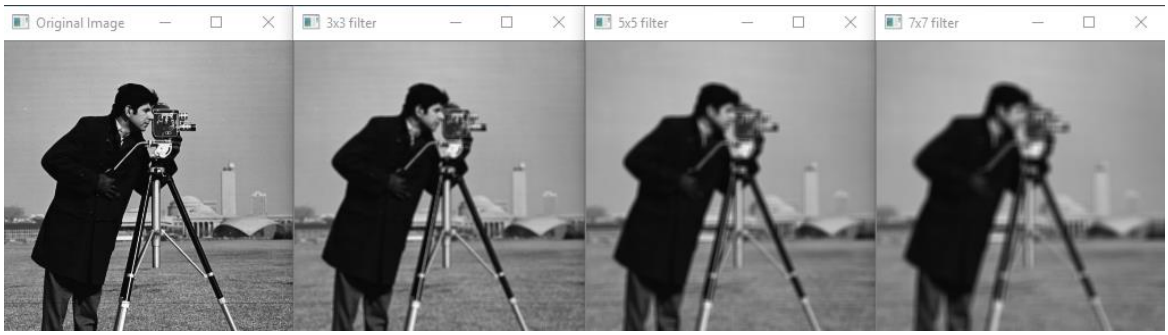
blur(InputArray src, OutputArray dst, Size ksize, Point anchor=Point(-1,-1), int borderType=BORDER_DEFAULT) όπου

- src: Εικόνα εισόδου, όπου ο τύπος δεδομένων των στοιχείων της θα πρέπει να είναι ένας εκ' των CV_8U, CV_16U, CV_16S, CV_32F ή CV_64F.
- dst: Εικόνα εξόδου με ίδιο μέγεθος και τύπο δεδομένων με την εικόνα εισόδου
- ksize: Μέγεθος της μάσκας του φίλτρου (πχ Size(5,5))
- anchor: Θέση του pixel στο οποίο θα εφαρμοστεί το φίλτρο. Η εξ' ορισμού τιμή (-1,-1) δηλώνει ότι η θέση του pixel βρίσκεται στο κέντρο της μάσκας
- borderType: Τύπος επέκτασης της εικόνας (βλέπε filter2D).

Παράδειγμα εφαρμογής φίλτρου μέσης τιμής

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
```

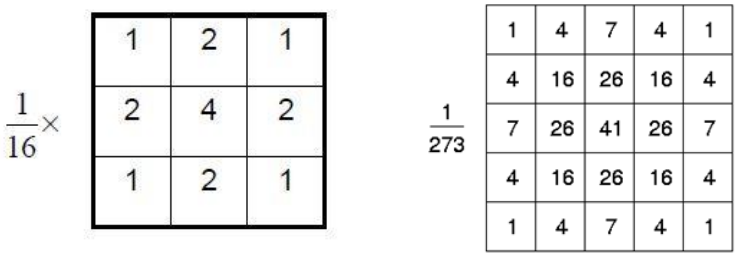
```
using namespace std;

int main(int argc, char** argv)
{

	Mat img = imread("C:/images/cameraman.tif", 0);
	Mat res1, res2, res3;
	blur(img, res1, Size(3, 3)); // Efarmogi filtrou mesis timis 3x3
	blur(img, res2, Size(5, 5)); // Efarmogi filtrou mesis timis 5x5
	blur(img, res3, Size(7, 7)); // Efarmogi filtrou mesis timis 7x7

	// Emfanisi ton eikonon
	namedWindow("Original Image" , CV_WINDOW_AUTOSIZE);
	namedWindow("3x3 filter" , CV_WINDOW_AUTOSIZE);
	namedWindow("5x5 filter" , CV_WINDOW_AUTOSIZE);
	namedWindow("7x7 filter" , CV_WINDOW_AUTOSIZE);

	imshow("Original Image", img);
	imshow("3x3 filter", res1);
	imshow("5x5 filter", res2);
	imshow("7x7 filter", res3);

	waitKey(0);
	destroyAllWindows();
 return 0;
}
```



Εικόνα 4.28: Εφαρμογή φίλτρου μέσης τιμής στην εικόνα cameraman.tif

4.11.3 Φίλτρο μέσης τιμής ειδικών βαρών

Το φίλτρο μέσης τιμής ειδικών βαρών (weighted averaging or weighted mean filter), ή αλλιώς Γκαουσσιανό φίλτρο (Gaussian filter) αποτελεί μια παραλλαγή του φίλτρου μέσης τιμής. Η διαφορά του έχει να κάνει με τις τιμές της μάσκας, οι οποίες δεν είναι όλες ίσες με 1 αλλά υπάρχουν και τιμές μεγαλύτερες του 1. Οι τιμές αυτές αποτελούν τα ειδικά βάρη (weights).



Σχήμα 4.19: Φίλτρο μέσης τιμής ειδικών βαρών 3x3 και 5x5 **[45] [46]**

Η συνάρτηση με την οποία γίνεται εφαρμογή του φίλτρου μέσης τιμής ειδικών βαρών σε μια εικόνα είναι η **GaussianBlur()**. Συντάσσεται ως εξής:

GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT) όπου

- src: εικόνα εισόδου, όπου ο τύπος δεδομένων των στοιχείων της θα πρέπει να είναι ένας εκ' των CV_8U, CV_16U, CV_16S, CV_32F ή CV_64F.
- dst: εικόνα εξόδου με ίδιο μέγεθος και τύπο δεδομένων με την εικόνα εισόδου
- ksize: μέγεθος της μάσκας του φίλτρου (πχ (5,5))
- sigmaX: Τυπική απόκλιση στον άξονα X
- sigmaY: Τυπική απόκλιση στον άξονα Y
- borderType: Τύπος επέκτασης της εικόνας (βλέπε filter2D).

Παράδειγμα εφαρμογής φίλτρου μέσης τιμής ειδικών βαρών

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, char** argv )
{

	Mat img = imread("C:/images/cameraman.tif", 0);
	Mat res1, res2, res3;
	// Efarmogh filtrou meshs timhs eidikwn varwn 3x3
	GaussianBlur(img, res1, Size(3, 3), 0);
	// Efarmogh filtrou meshs timhs eidikwn varwn 5x5
	GaussianBlur(img, res2, Size(5, 5), 0);
	// Efarmogh filtrou meshs timhs eidikwn varwn 7x7
	GaussianBlur(img, res3, Size(7, 7), 0);

	// Emfanisi ton eikonon
```

```
        namedWindow("Original Image" , CV_WINDOW_AUTOSIZE);
        namedWindow("3x3 filter" , CV_WINDOW_AUTOSIZE);
        namedWindow("5x5 filter" , CV_WINDOW_AUTOSIZE);
        namedWindow("7x7 filter" , CV_WINDOW_AUTOSIZE);

        imshow("Original Image", img);
        imshow("3x3 filter", res1);
        imshow("5x5 filter", res2);
        imshow("7x7 filter", res3);

        waitKey(0);
        destroyAllWindows();
return 0;
```

Εικόνα 4.29: Εφαρμογή φίλτρου μέσης τιμής ειδικών βαρών στην εικόνα cameraman.tif

4.11.4 Φίλτρο πρώτης παραγώγου

Το φίλτρο πρώτης παραγώγου, ή αλλιώς φίλτρο Sobel (Sobel filter), αποτελεί ένα ακόμα γραμμικό φίλτρο και χρησιμοποιείται κυρίως για την ανάδειξη ακμών (edge detection). Υπάρχουν δυο είδη φίλτρων πρώτης παραγώγου: το φίλτρο οριζόντιας πρώτης παραγώγου, το οποίο προσδιορίζει τις κάθετες ακμές και το φίλτρο κάθετης πρώτης παραγώγου, το οποίο προσδιορίζει τις οριζόντιες ακμές μιας εικόνας. **[1]**

-1	0	+1
-2	0	+2
-1	0	+1

x filter

+1	+2	+1
0	0	0
-1	-2	-1

y filter

Σχήμα 4.20: Φίλτρο πρώτης παραγώγου 3x3
(x filter) οριζόντια πρώτη παράγωγος
(y filter) κάθετη πρώτη παράγωγος **[47]**

Η συνάρτηση με την οποία γίνεται εφαρμογή του φίλτρου πρώτης παραγώγου σε μια εικόνα είναι η **Sobel()**. Συντάσσεται ως εξής:

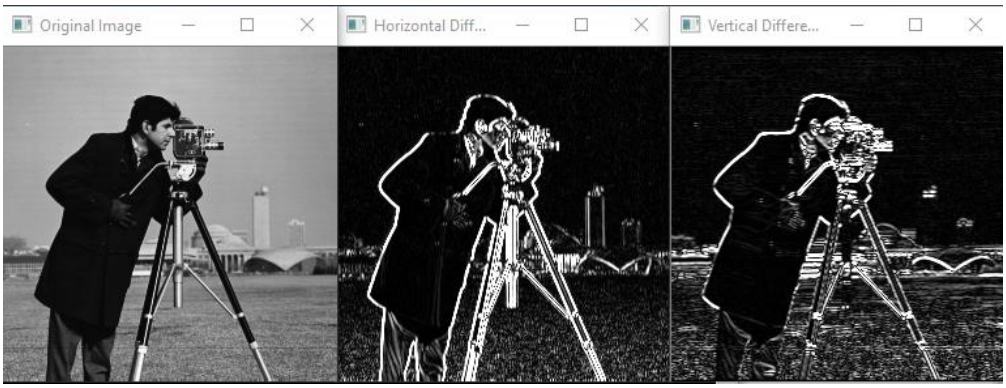
Sobel(InputArray src, OutputArray dst, int ddepth, int dx, int dy, int ksize=3, double scale=1, double delta=0, int borderType=BORDER_DEFAULT) όπου

- src: Εικόνα εισόδου
- dst: Εικόνα εξόδου με ίδιο μέγεθος και τύπο δεδομένων με την εικόνα εισόδου
- ddepth: Τύπος δεδομένων της εικόνας εξόδου. Με τιμή -1 είναι ίδιος με το τύπο δεδομένων της εικόνας εισόδου (src.depth()). Ισχύουν τα εξής:
    - src.depth() = CV_8U, ddepth = -1/CV_16S/CV_32F/CV_64F
    - src.depth() = CV_16U/CV_16S, ddepth = -1/CV_32F/CV_64F
    - src.depth() = CV_32F, ddepth = -1/CV_32F/CV_64F
    - src.depth() = CV_64F, ddepth = -1/CV_64F
- dx: Ορισμός οριζόντιας πρώτης παράγωγου (dx = 1, dy = 0)
- dy: Ορισμός κάθετης πρώτης παράγωγου (dx = 0, dy = 1)
- ksize: Μέγεθος της μάσκας. Ισχύουν τα εξής:
    - Με ksize = 1, χρησιμοποιείται μάσκα 3x1 ή 1x3 (εξαρτάται από τις τιμές dx και dy).
    - Με ksize = 3, 5 ή 7, η μάσκα έχει μέγεθος ksize x ksize.
    - Με ksize = 3 χρησιμοποιείται μια από τις δυο μάσκες της Εικόνας -. (εξαρτάται από τις τιμές dx και dy)
    - Με ksize = -1 χρησιμοποιείται το φίλτρο Scharr 3x3
- scale: προαιρετικός συντελεστής κλίμακας για τον υπολογισμό των πρώτων παραγώγων. Εξ' ορισμού δεν εφαρμόζεται συντελεστής κλίμακας..
- delta: Προαιρετική σταθερά που προστίθεται στο αποτέλεσμα της εφαρμογής του φίλτρου.
- borderType: τύπος επέκτασης της εικόνας (βλέπε filter2D).

Παράδειγμα εφαρμογής φίλτρου πρώτης παραγώγου

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;
```

```
int main(int argc, char** argv)
{

	Mat img = imread( "C:/images/cameraman.tif", 0 );
	Mat sobelx,sobely, abs_sobelx, abs_sobely;

	// Efarmogh orizontias prwths paragwgou.
	//  Thetoume typo dedomenwn megalutero apo CV_8U gia na apofugoume twn uperxulish
	Sobel(img, sobelx, CV_16S, 1, 0, 3);
	convertScaleAbs(sobelx, abs_sobelx); // Metatropi se CV_8U
	// Efarmogh katheths prwths paragwgou.
	//  Thetoume typo dedomenwn megalutero apo CV_8U gia na apofugoume twn uperxulish
	Sobel(img, sobely, CV_16S, 0, 1, 3);
	convertScaleAbs(sobely, abs_sobely); // Metatropi se CV_8U

	// Emfanisi ton eikonon
	namedWindow( "Original Image" , CV_WINDOW_AUTOSIZE );
	namedWindow( "Horizontal Differentiation" , CV_WINDOW_AUTOSIZE );
	namedWindow( "Vertical Differentiation" , CV_WINDOW_AUTOSIZE );

	imshow( "Original Image", img );
	imshow( "Horizontal Differentiation", abs_sobelx );
	imshow("Vertical Differentiation", abs_sobely);

	waitKey(0);
	destroyAllWindows();
 return 0;
}
```



Εικόνα 4.30: Εφαρμογή φίλτρου πρώτης παραγώγου στην εικόνα cameraman.tif

4.11.5 Φίλτρο Scharr

Το φίλτρο Scharr (Scharr filter), αποτελεί ένα γραμμικό φίλτρο, παρόμοιο με το φίλτρο Sobel, και χρησιμοποιείται κυρίως για την ανάδειξη ακμών (edge detection). Το φίλτρο αυτό, πολλές φορές, μπορεί να δώσει πιο ακριβή αποτελέσματα από το φίλτρο Sobel. Όπως το φίλτρο Sobel, έτσι και το φίλτρο Scharr έχει φίλτρο οριζόντιας και κάθετης παραγώγου.

$$\begin{bmatrix} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{bmatrix} \qquad \begin{bmatrix} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$$

x filter **y filter**

Σχήμα 4.21: Φίλτρο Scharr 3x3
(x filter) οριζόντια παράγωγος
(y filter) κάθετη παράγωγος **[48]**

Η συνάρτηση με την οποία γίνεται εφαρμογή του φίλτρου Scharr σε μια εικόνα είναι η **Scharr().** Συντάσσεται ως εξής:

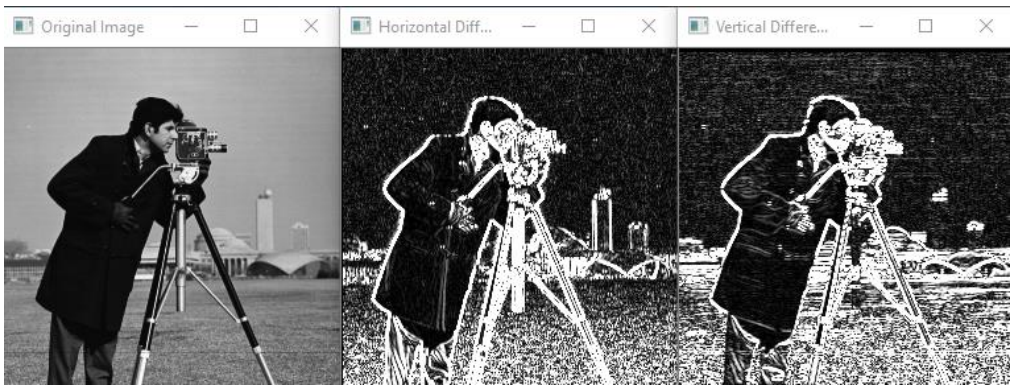
Scharr(InputArray src, OutputArray dst, int ddepth, int dx, int dy, double scale=1, double delta=0, int borderType=BORDER_DEFAULT) όπου

- src: Εικόνα εισόδου
- dst: Εικόνα εξόδου με ίδιο μέγεθος και τύπο δεδομένων με την εικόνα εισόδου
- ddepth: Τύπος δεδομένων της εικόνας εξόδου. Με τιμή -1 είναι ίδιος με το τύπο δεδομένων της εικόνας εισόδου (src.depth()). Ισχύουν τα εξής:
  - src.depth() = CV_8U, ddepth = -1/CV_16S/CV_32F/CV_64F
  - src.depth() = CV_16U/CV_16S, ddepth = -1/CV_32F/CV_64F
  - src.depth() = CV_32F, ddepth = -1/CV_32F/CV_64F
  - src.depth() = CV_64F, ddepth = -1/CV_64F
- dx: Ορισμός οριζόντιας πρώτης παράγωγου (dx = 1, dy = 0)
- dy: Ορισμός κάθετης πρώτης παράγωγου (dx = 0, dy = 1)
- scale: προαιρετικός συντελεστής κλίμακας για τον υπολογισμό των πρώτων παραγώγων. Εξ' ορισμού δεν εφαρμόζεται συντελεστής κλίμακας.
- delta: Προαιρετική σταθερά που προστίθεται στο αποτέλεσμα της εφαρμογής του φίλτρου.
- borderType: τύπος επέκτασης της εικόνας (βλέπε filter2D).

Παράδειγμα εφαρμογής φίλτρου Scharr

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
```

```
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{

	Mat img = imread("C:/images/cameraman.tif", 0);
	Mat scharrx, scharry, abs_scharrx, abs_scharry;
	// Efarmogh orizontias paragwgou.
	//  Thetoume typo dedomenwn megalutero apo CV_8U gia na apofugoume tin uperxulish
	Scharr(img, scharrx, CV_16S, 1, 0);
	convertScaleAbs(scharrx, abs_scharrx); // Metatropi se CV_8U

	// Efarmogh katheths paragwgou.
	//  Thetoume typo dedomenwn megalutero apo CV_8U gia na apofugoume tin uperxulish
	Scharr(img, scharry, CV_16S, 0, 1);
	convertScaleAbs(scharry, abs_scharry); // Metatropi se CV_8U

	// emfanisi ton eikonon
	namedWindow("Original Image" , CV_WINDOW_AUTOSIZE);
	namedWindow("Horizontal Differentiation" , CV_WINDOW_AUTOSIZE);
	namedWindow("Vertical Differentiation" , CV_WINDOW_AUTOSIZE);

	imshow("Original Image", img);
	imshow("Horizontal Differentiation", abs_scharrx);
	imshow("Vertical Differentiation", abs_scharry);

	waitKey(0);
	destroyAllWindows();
 return 0;
}
```



Εικόνα 4.31: Εφαρμογή φίλτρου Scharr στην εικόνα cameraman.tif

4.11.6 Φίλτρο δεύτερης παραγωγού

Το φίλτρο δεύτερης παραγώγου, ή αλλιώς φίλτρο Laplacian (Laplacian filter), αποτελεί ένα ακόμα γραμμικό φίλτρο και χρησιμοποιείται για τον εμπλουτισμό μιας εικόνας. Αυτό επιτυγχάνεται με την ανάδειξη των απότομων μεταβολών

φωτεινότητας και τη ταυτόχρονη εξομάλυνση των ομαλών μεταβολών φωτεινότητας της εικόνας (sharpening). **[1]**

0	1	0
1	−4	1
0	1	0

Σχήμα 4.22: Φίλτρο δεύτερης παραγώγου 3x3 **[49]**

Η συνάρτηση με την οποία γίνεται εφαρμογή του φίλτρου δεύτερης παραγώγου σε μια εικόνα είναι η **laplacian()** και συντάσσεται ως εξής:

Laplacian(InputArray src, OutputArray dst, int ddepth, int ksize=1, double scale=1, double delta=0, int borderType=BORDER_DEFAULT) όπου

- src: Εικόνα εισόδου.
- dst: Εικόνα εξόδου με ίδιο μέγεθος και τύπο δεδομένων με την εικόνα εισόδου
- ddepth: Τύπος δεδομένων της εικόνας εξόδου
- ksize: Το μέγεθος που θα χρησιμοποιηθεί για τον υπολογισμό του φίλτρου δεύτερης παραγώγου. Δέχεται θετικές και περιττές τιμές. Ισχύουν τα εξής:
    - Με ksize > 1 ο υπολογισμός του φίλτρου γίνεται σύμφωνα με τον τελεστή Laplacian:

$$\nabla^2 src = \frac{\partial^2 src}{\partial x^2} + \frac{\partial^2 src}{\partial y^2}$$

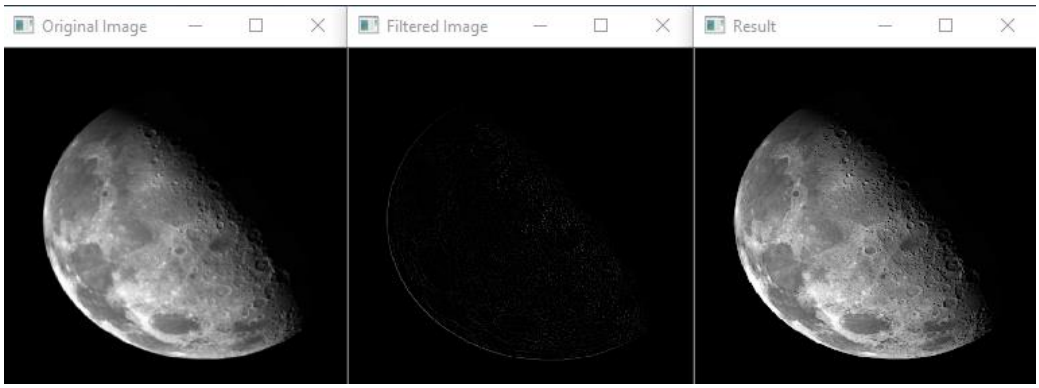
    - Με ksize = 1 χρησιμοποιείται το φίλτρο του σχήματος 4.22.
- scale: Προαιρετικός συντελεστής κλίμακας για τον υπολογισμό των πρώτων παραγώγων. Εξ' ορισμού δεν εφαρμόζεται συντελεστής κλίμακας..
- delta: Προαιρετική σταθερά που προστίθεται στο αποτέλεσμα της εφαρμογής του φίλτρου.
- borderType: Τύπος επέκτασης της εικόνας (βλέπε filter2D).

Παράδειγμα εφαρμογής φίλτρου πρώτης παραγώγου

Σημείωση

Για να πάρουμε τη νέα εμπλουτισμένη εικόνα αφαιρούμε από την αρχική την εικόνα που προέκυψε μετά την εφαρμογή του φίλτρου.

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{

	Mat img = imread("C:/images/moon.tif", 0);
	Mat lap, res;

	// Efarmogh filtrou deuterhs paragwgou stin eikona
	Laplacian(img, lap, CV_8U, 1);

	subtract(img, lap, res);

	//show the images
	namedWindow("Original Image" , CV_WINDOW_NORMAL);
	namedWindow("Filtered Image" , CV_WINDOW_NORMAL);
	namedWindow("Result", CV_WINDOW_NORMAL);

	imshow("Original Image", img);
	imshow("Filtered Image", lap);
	imshow("Result", res);

	waitKey(0);
	destroyAllWindows();
 return 0;
}
```

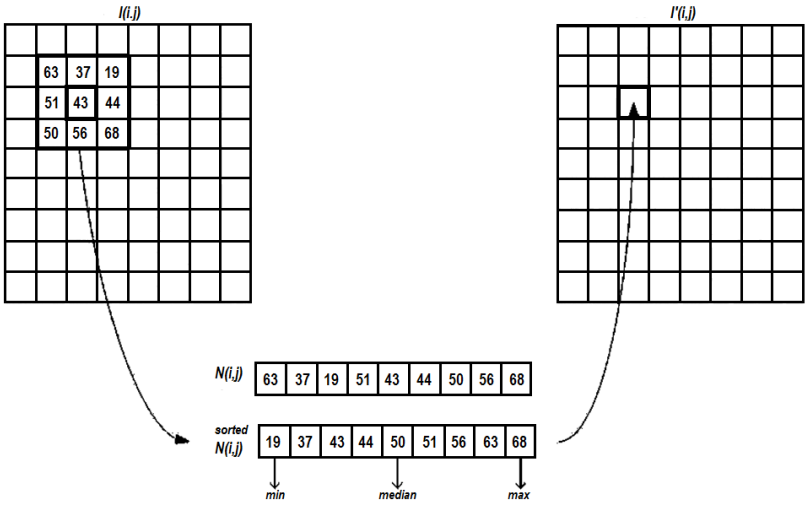


Εικόνα 4.32: Εφαρμογή φίλτρου δεύτερης παραγώγου στην εικόνα moon.tif

4.11.7 Μη γραμμικά φίλτρα

Κατά την εφαρμογή ενός μη γραμμικού φίλτρου σε μια εικόνα, η μεταβολή κάθε εικονοστοιχείου γίνεται ως εξής:
Οι τιμές των εικονοστοιχείων της γειτονίας ταξινομούνται κατά αύξουσα σειρά και με βάση τον χρησιμοποιούμενο αλγόριθμο επιλέγεται η τιμή που θα αντικαταστήσει τη αρχική τιμή του εκάστοτε εικονοστοιχείου.

Στη κατηγορία αυτή υπάρχουν τα φίλτρα μεσαίας, ελάχιστης και μέγιστης τιμής.



Σχήμα 4.23: Εφαρμογή μη γραμμικού φίλτρου **[50]**

Το φίλτρο μεσαίας τιμής αποτελεί ένα από τα πιο γνωστά μη γραμμικά φίλτρα και χρησιμοποιείται κατά κύριο λόγο για την απαλοιφή θορύβου salt & pepper. Κατά την εφαρμογή του φίλτρου μεσαίας τιμής σε μια εικόνα, το pixel στο οποίο εφαρμόζεται το φίλτρο αντικαθιστάται από τη μεσαία τιμή των pixels της γειτονίας του. Αντίστοιχα, στα φίλτρα ελάχιστης και μέγιστης τιμής επιλέγονται η ελάχιστη και η μέγιστη τιμή των pixels της γειτονίας του αντίστοιχα. Θα παρουσιαστούν αναλυτικά τα φίλτρα ελάχιστης και μέγιστης τιμής στην ενότητα 4.12.3. **[1]**

Η συνάρτηση με την οποία γίνεται εφαρμογή του φίλτρου μέσης τιμής σε μια εικόνα είναι η **medianBlur**, η οποία συντάσσεται ως εξής:

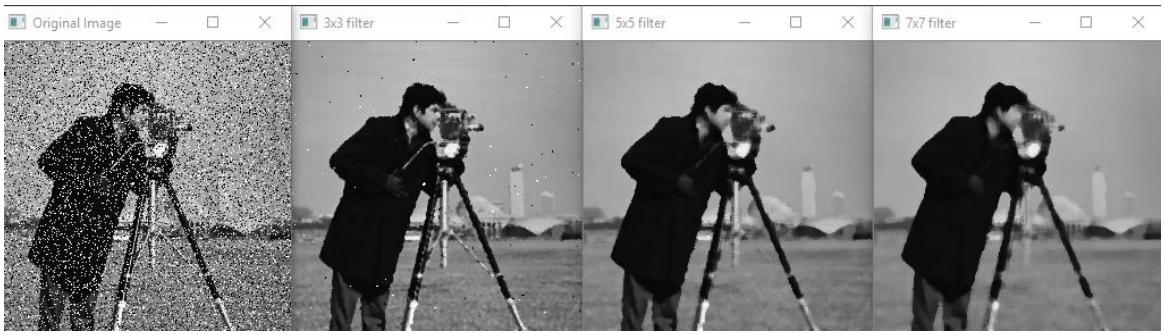
medianBlur(InputArray src, OutputArray dst, int ksize) όπου

- src: Εικόνα εισόδου. Όταν το ksize είναι 3 ή 5, ο τύπος δεδομένων της εικόνας θα πρέπει να είναι ένας εκ' των CV_8U, CV_16U, ή CV_32F, ενώ για μεγαλύτερο ksize, ο τύπος δεδομένων θα πρέπει να είναι CV_8U.

- dst: εικόνα εξόδου με ίδιο μέγεθος και τύπο δεδομένων με την εικόνα εισόδου
- ksize: μέγεθος της μάσκας του φίλτρου. Θα πρέπει να είναι αριθμός περιττός και μεγαλύτερος από 1, για παράδειγμα 3 (μάσκα 3x3), 5 (μάσκα 5x5) κλπ.

Παράδειγμα εφαρμογής φίλτρου μεσαίας τιμής

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{

	Mat img = imread("C:/images/cameraman_noise.tif", 0);
	Mat res1, res2, res3;
	medianBlur(img, res1, 3); // Efarmogi filtrou mesaias timis 3x3
	medianBlur(img, res2, 5); // Efarmogi filtrou mesaias timis 5x5
	medianBlur(img, res3, 7); // Efarmogi filtrou mesaias timis 7x7

	// Emfanisi ton eikonon
	namedWindow("Original Image" , CV_WINDOW_AUTOSIZE);
	namedWindow("3x3 filter" , CV_WINDOW_AUTOSIZE);
	namedWindow("5x5 filter" , CV_WINDOW_AUTOSIZE);
	namedWindow("7x7 filter" , CV_WINDOW_AUTOSIZE);

	imshow("Original Image", img);
	imshow("3x3 filter", res1);
	imshow("5x5 filter", res2);
	imshow("7x7 filter", res3);

	waitKey(0);
	destroyAllWindows();
 return 0;
}
```



Εικόνα 4.33: Εφαρμογή φίλτρου μεσαίας τιμής στην εικόνα cameraman.tif

4.12 Αποκατάσταση εικόνας

Στην αποκατάσταση εικόνας εφαρμόζονται τεχνικές (φίλτρα) τα οποία μπορούν να διορθώσουν σε μεγάλο βαθμό μια ενθόρυβη, θολωμένη ή παραμορφωμένη εικόνα. Εμείς θα ασχοληθούμε με το κομμάτι της απαλοιφής θορύβου.

4.12.1 Τι είναι ο θόρυβος

Ο θόρυβος (noise) είναι ένα ανεπιθύμητο σήμα, το οποίο επηρεάζει τη φωτεινότητα μιας εικόνας και μπορεί να προκύψει είτε κατά τη διαδικασία ψηφιοποίησης είτε κατά τη διαδικασία μετάδοσης της. **[1]**

Στην ενότητα αυτή θα παρουσιάσουμε τους πιο συνηθισμένους τύπους θορύβου και τους τρόπους προσθήκης και απαλοιφής θορύβου.

4.12.2 Τύποι θορύβου

Υπάρχουν αρκετοί τύποι θορύβου στις εικόνες. Εμείς παραθέτουμε επιγραμματικά τους εξής:

- Ομοιόμορφος θόρυβος (Uniform noise)
- Κανονικός θόρυβος (Normal or Gaussian noise)
- Σημειακός θόρυβος (Impulse or Salt & Pepper noise)
- Θόρυβος Rayleigh (Rayleigh noise)
- Εκθετικός θόρυβος (Exponential noise)
- Θόρυβος gamma (Gamma noise)

Εμείς θα ασχοληθούμε με τους πρώτους τρείς τύπους θορύβου. **[1]**

4.12.3 Δημιουργία και προσθήκη θορύβου

Για τη δημιουργία ομοιόμορφου και κανονικού θορύβου μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις της OpenCV για παραγωγή τυχαίων αριθμών **randu()** και **randn()** αντίστοιχα. Η συνάρτηση randu() συντάσσεται ως εξής:

randu(InputOutputArray dst, InputArray low, InputArray high) όπου:

- dst: Πίνακας θορύβου
- low: Κατώτερο όριο του εύρους των τιμών που θα παραχθούν
- high: Ανώτερο όριο του εύρους των τιμών που θα παραχθούν

Οι τιμές που παράγει η συνάρτηση randu() ανήκουν στο εύρος:

low <= dst(I) < high

Η συνάρτηση randn() συντάσσεται ως εξής:

randn(InputOutputArray dst, InputArray mean, InputArray stddev) όπου:

- dst: Πίνακας θορύβου
- mean: Μέση τιμή
- stddev: Διασπορά

Η προσθήκη ομοιόμορφου θορύβου σε μια εικόνα γίνεται με τη σχέση:

Noise Image = image + σ * noise όπου

- image: Αρχική εικόνα
- noise: Θόρυβος
- σ: Διασπορά

Με βάση τα παραπάνω γράφουμε την εξής συνάρτηση για τη προσθήκη ομοιόμορφου θορύβου σε μια εικόνα:

```
// Sunartisi gia tin prosthiki omiomorfou thorivou
// H eikona eisodou prepei na exei times fotinotiton sto diastima [0,255]
// Ta low kai high dhlonoun ta oria tou eurous ton tuxaion timon
// To sigma einai h diaspora

Mat add_uniform(Mat image, int low, int high, double sigma)
{
	int w = image.cols; int h = image.rows;

	// Dimiourgia tis eikonas me thorivo
	Mat noise_image(w, h, CV_8U, Scalar(0));

	Mat noise(w, h, CV_8U); // Dimiourgia tou pinaka thorivou

	randu(noise, low, high); // Dimiourgia tou thorivou


	// Prosthiki thorivou stin eikona
	noise_image = image + (sigma * noise);

return noise_image;
}
```

Η προσθήκη κανονικού θορύβου σε μια εικόνα γίνεται ως εξής:

- Αν stddev = 0, τότε Noise Image = image + σ * noise
- Αν stddev > 0, τότε Noise Image = image + noise

Με βάση τα παραπάνω γράφουμε την εξής συνάρτηση για τη προσθήκη κανονικού θορύβου σε μια εικόνα:

```
// Sunartisi gia tin prosthiki omiomorfou thorivou
// H eikona eisodou prepei na exei times fotinotiton sto diastima [0,255]
// To mean einai i mesi timi
// To sigma einai h diaspora

Mat add_gaussian(Mat image, double mean, double sigma)
{
	int w = image.cols; int h = image.rows;

	// Dimiourgia tis eikonas me thorivo
	Mat noise_image(w, h, CV_8U, Scalar(0));

	Mat noise(w, h, CV_8U); // Dimiourgia tou pinaka thorivou

	randn(noise, mean, sigma); // Dimiourgia tou thorivou

	// Prosthiki thorivou stin eikona
	noise_image = image + noise;

return noise_image;
}
```

Η δημιουργία και προσθήκη θορύβου salt & pepper μπορεί να πραγματοποιηθεί γράφοντας την εξής συνάρτηση:

```
// Sunartisi gia tin prosthiki thorivou salt & pepper
// H eikona eisodou prepei na exei times fotinotitwn sto diastima [0,255]
// To p einai h piknotita thorivou

Mat add_salt_and_pepper(Mat image, double p)
{
	int width = image.cols; int height = image.rows;
	// Dimiourgia tis eikonas me thorivo
	Mat noise_image; image.copyTo(noise_image);
	Mat rnd(w, h, CV_8U, Scalar(0)); // Dimiourgia tou pinaka thorivou
	Mat mask1, mask2;
	double min, max;
	double thr = 255 - p;

	randu(rnd, 0, 256);
	minMaxLoc(rnd, &min, &max);

	// Evresi ton suntetagmenwn ton fotinotitwn pou einai < p
	inRange(rnd, min, p, mask1);
	// Evresi ton suntetagmenwn ton fotinotitwn pou einai > thr
	inRange(rnd, thr, max, mask2);

	// Prosthikh thorivou stin eikona
	noise_image.setTo(0, mask1);
	noise_image.setTo(255, mask2);

return noise_image; }
```

4.12.4 Απαλοιφή θορύβου

Στην ενότητα αυτή θα παρουσιάσουμε ορισμένα φίλτρα απαλοιφής θορύβου. Μερικά από αυτά έχουν αναφερθεί στις τεχνικές χώρου.

- **Φίλτρο μέσης τιμής (βλέπε ενότητα 4.11.2)**

  Το φίλτρο μέσης τιμής χρησιμοποιείται κυρίως για τη απαλοιφή κανονικού θορύβου. **[1]**

Παράδειγμα απαλοιφής κανονικού θορύβου με το φίλτρο μέσης τιμής

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, char** argv )
{
	Mat img = imread("C:/images/cameraman.tif", 0);
	Mat noise_img = add_gaussian(img, 45, 30); // mean = 45 kai sigma =
	30
	Mat res;
	blur(noise_img, res, Size(5, 5)); // Efarmogi filtrou mesis timis

	// Emfanisi ton eikonon
	imshow("Original Image", img );
	imshow("Noisy Image", noise_img);
	imshow("Result", res);

	waitKey(0);
	destroyAllWindows();
return 0;
}
```

Εικόνα 4.34: Απαλοιφή κανονικού θορύβου με το φίλτρο μέσης τιμής

- **Φίλτρο μεσαίας τιμής (βλέπε ενότητα 4.11.7)**

  Το φίλτρο μεσαίας τιμής χρησιμοποιείται κυρίως για τη απαλοιφή θορύβου salt & pepper. **[1]**

  Παράδειγμα απαλοιφής θορύβου salt & pepper με το φίλτρο μεσαίας τιμής

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, char** argv )
{
	Mat img = imread("C:/images/cameraman.tif", 0);
	int width = img.cols; int height = img.rows;
	Mat noise_img = add_salt_and_pepper(img, 26); // p = 0.1 * 255 = 25.5
	= 26
	Mat res;
	medianBlur(noise_img, res, 3); // Efarmogi filtrou mesaias timis

	// Emfanisi ton eikonon
	imshow("Original Image", img );
	imshow("Noisy Image", noise_img);
	imshow("Result", res);

	waitKey(0);
	destroyAllWindows();
return 0;
}
```

Εικόνα 4.35: Απαλοιφή θορύβου salt & pepper με το φίλτρο μέσης τιμής

- **Φίλτρο ελάχιστης τιμής**

  Το φίλτρο ελάχιστης τιμής λειτουργεί όπως και το φίλτρο μεσαίας τιμής με τη διαφορά ότι επιλέγεται η μικρότερη τιμή φωτεινότητας της γειτονιάς. Η συνάρτηση με την οποία γίνεται εφαρμογή του φίλτρου ελάχιστης τιμής σε μια εικόνα είναι η **erode()**. Συντάσσεται ως εξής:

void erode(InputArray src, OutputArray dst, InputArray kernel, Point anchor=Point(-1,-1), int iterations=1, int borderType=BORDER_CONSTANT, const Scalar& borderValue=morphologyDefaultBorderValue()) όπου:

- src: Εικόνα εισόδου, όπου ο τύπος δεδομένων των στοιχείων της θα πρέπει να είναι ένας εκ' των CV_8U, CV_16U, CV_16S, CV_32F ή CV_64F.
- dst: Εικόνα εξόδου με ίδιο μέγεθος και τύπο δεδομένων με την εικόνα εισόδου
- kernel: Μάσκα του φίλτρου
- anchor: Θέση του pixel στο οποίο θα εφαρμοστεί το φίλτρο. Η εξ' ορισμού τιμή (-1,-1) δηλώνει ότι η θέση του pixel βρίσκεται στο κέντρο της μάσκας
- iterations: Αριθμός επαναλήψεων εφαρμογής του φίλτρου. Εξ' ορισμού επαναλαμβάνεται μια φορά
- borderType: Τύπος επέκτασης της εικόνας (βλέπε filter2D).
- borderValue: Τιμή σε περίπτωση που το borderValue έχει τιμή BORDER_CONSTANT

Το φίλτρο ελάχιστης τιμής χρησιμοποιείται κυρίως για τη απαλοιφή θορύβου salt. **[1]**

Παράδειγμα απαλοιφής θορύβου salt με το φίλτρο ελάχιστης τιμής

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, char** argv )
{
	Mat img = imread("C:/images/cameraman.tif", 0);

	// p = 0.1 * 255 = 25.5 = 26
	Mat noise_img = add_salt_and_pepper(img, 26);
	Mat res;
	Mat kernel = Mat::ones(3,3,CV_8U); // Dimiourgia tis maskas

	erode(noise_img, res, kernel);  // Efarmogi filtrou elaxistis timis

	// Emfanisi ton eikonon
```

```
        imshow("Original Image", img );
        imshow("Noisy Image", noise_img);
        imshow("Result", res);

        waitKey(0);
        destroyAllWindows();
return 0;
}
```

Εικόνα 4.36: Απαλοιφή θορύβου salt με το φίλτρο ελάχιστης τιμής

- **Φίλτρο μέγιστης τιμής**

  Το φίλτρο μέγιστης τιμής λειτουργεί όπως και το φίλτρο μεσαίας τιμής με τη διαφορά ότι επιλέγεται η μεγαλύτερη τιμή φωτεινότητας της γειτονιάς. Η συνάρτηση με την οποία γίνεται εφαρμογή του φίλτρου μέγιστης τιμής σε μια εικόνα είναι η **dilate()**. Συντάσσεται ως εξής:

  void dilate(InputArray src, OutputArray dst, InputArray kernel, Point anchor=Point(-1,-1), int iterations=1, int borderType=BORDER_CONSTANT, const Scalar& borderValue=morphologyDefaultBorderValue()) όπου:

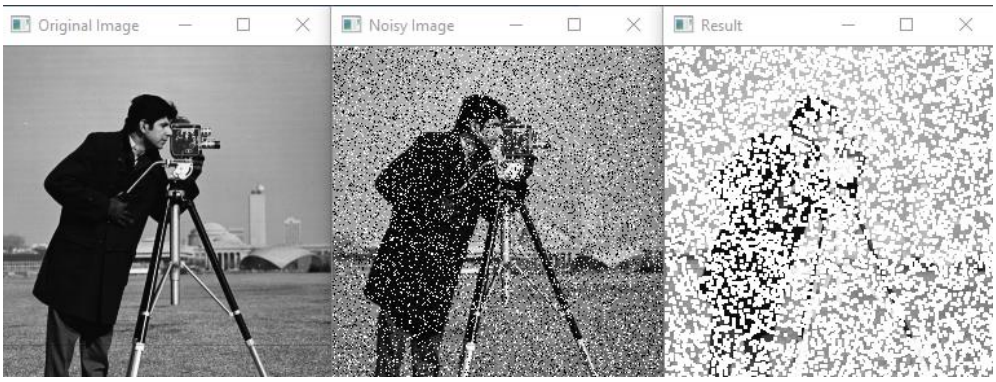
  - src: Εικόνα εισόδου, όπου ο τύπος δεδομένων των στοιχείων της θα πρέπει να είναι ένας εκ' των CV_8U, CV_16U, CV_16S, CV_32F ή CV_64F.
  - dst: Εικόνα εξόδου με ίδιο μέγεθος και τύπο δεδομένων με την εικόνα εισόδου
  - kernel: Μάσκα του φίλτρου
  - anchor: Θέση του pixel στο οποίο θα εφαρμοστεί το φίλτρο. Η εξ' ορισμού τιμή (-1,-1) δηλώνει ότι η θέση του pixel βρίσκεται στο κέντρο της μάσκας

- iterations: Αριθμός επαναλήψεων εφαρμογής του φίλτρου. Εξ' ορισμού επαναλαμβάνεται μια φορά
- borderType: Τύπος επέκτασης της εικόνας (βλέπε filter2D).
- borderValue: Τιμή σε περίπτωση που το borderValue έχει τιμή BORDER_CONSTANT

Το φίλτρο μέγιστης τιμής χρησιμοποιείται κυρίως για τη απαλοιφή θορύβου pepper. **[1]**

Παράδειγμα απαλοιφής θορύβου pepper με το φίλτρο μέγιστης τιμής

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, char** argv )
{
	Mat img = imread("C:/images/cameraman.tif", 0);
	Mat noise_img = add_salt_and_pepper(img, 26); // p = 0.1 * 255 = 25.5
	= 26
	Mat res;
	Mat kernel = Mat::ones(3,3,CV_8U); // Dimiourgia tis maskas

	dilate(noise_img, res, kernel);  // Efarmogi filtrou megistis timis

	// Emfanisi ton eikonon
	imshow("Original Image", img );
	imshow("Noisy Image", noise_img);
	imshow("Result", res);

	waitKey(0);
	destroyAllWindows();
return 0;
}
```



Εικόνα 4.37: Απαλοιφή θορύβου pepper με το φίλτρο μέγιστης τιμής

- **Φίλτρο ενδιάμεσου σημείου**

  Το φίλτρο ενδιάμεσου σημείου αποτελεί ένα ακόμα μη γραμμικό φίλτρο. Κατά την εφαρμογή του σε μια εικόνα, το pixel στο οποίο εφαρμόζεται το φίλτρο αντικαθιστάται από τη μέση τιμή της μέγιστης και της ελάχιστης τιμής των pixels της γειτονιάς του. Χρησιμοποιείται κυρίως για τη απαλοιφή κανονικού και ομοιόμορφου θορύβου. **[1]**

  Παράδειγμα απαλοιφής κανονικού θορύβου με το φίλτρο ενδιάμεσου σημείου

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, char** argv )
{
	Mat img = imread("C:/images/cameraman.tif", 0);

	// mean = 45 kai sigma = 20
	Mat noise_img = add_gaussian(img, 45, 20);
	Mat max, min, sum, medp;

	// Dimiourgia maskas gia ta filtra megistis kai elaxistis timis
	Mat kernel = Mat::ones(3,3,CV_8U);

	dilate(noise_img, max, kernel);  // Efarmogi filtrou megistis timis
	erode(noise_img, min, kernel);  // Efarmogi filtrou elaxistis timis

	// Ypologismos tou filtrou endiamesou simeiou
	medp = (max + min) / 2;

	// Emfanisi ton eikonon
	imshow("Original Image", img );
	imshow("Noisy Image", noise_img);
	imshow("Result", medp);

	waitKey(0);
	destroyAllWindows();
return 0;
}
```

Εικόνα 4.38: Απαλοιφή κανονικού θορύβου με το φίλτρο ενδιάμεσου σημείου

ΚΕΦΑΛΑΙΟ 5

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ

5.1 Συμπεράσματα

Στη παρούσα πτυχιακή εργασία ασχοληθήκαμε με τις παρακάτω θεματικές ενότητες που σχετίζονται άμεσα με διάφορες περιοχές επεξεργασίας της εικόνας:

- Βασικός χειρισμός εικόνων (μετατροπή, ανάγνωση, εγγραφή, απεικόνιση, ιστόγραμμα)
- Βασικός χειρισμός εικονοστοιχείων (ανάγνωση, τροποποίηση τιμής εύρεση ελάχιστης και μέγιστης τιμής)
- Βελτιστοποίηση εικόνας
    - τεχνικές σημείου (αντιστροφή φωτεινότητας, επέκταση αντίθεσης, κόψιμο φωτεινότητας, τεμαχισμός κλίμακας φωτεινότητας, ανάλυση σε δυαδικές εικόνες, πράξεις επί των εικόνων, εξισορρόπηση ιστογράμματος)
    - τεχνικές χώρου (φίλτρο μέσης τιμής, φίλτρο μέσης τιμής ειδικών βαρών, φίλτρο πρώτης παραγώγου, φίλτρο Scharr, φίλτρο δεύτερης παραγωγού, μη γραμμικά φίλτρα)

- Αποκατάσταση εικόνας (φίλτρα απαλοιφής θορύβου)

Όπως φαίνεται, τα φίλτρα εκτός από την αποκατάσταση εικόνας, είναι αρκετά χρήσιμα και για την βελτιστοποίηση της. Για αυτό το σκοπό υπάρχει μια πληθώρα συναρτήσεων, που εξυπηρετεί οποιοδήποτε πρόβλημα, ώστε να υπάρχει μεγαλύτερη ευκρίνεια και η εικόνα να είναι περισσότερο εκμεταλλεύσιμη.

Συνεπώς, με βάση αυτά που αναφέρθηκαν, η επεξεργασία εικόνας είναι ικανή να επεκταθεί και σε άλλους τομείς που μπορούν στο μέλλον να καλυφθούν όπως για παράδειγμα είναι η τεχνητή όραση και αναγνώριση προτύπων. Συγκεκριμένα, υπολογιστική όραση ονομάζεται ο επιστημονικός κλάδος της τεχνητής νοημοσύνης, δηλαδή εξετάζει αλγορίθμους οι οποίοι δέχονται ως είσοδο ψηφιακές εικόνες, πολυδιάστατες εικόνες ή βίντεο και παράγουν συμβολικές περιγραφές των εν λόγω οπτικών σκηνών. ΄Οσον αφορά την αναγνώριση προτύπων, είναι και

αυτός ένας επιστημονικός κλάδος, όπου τα δεδομένα αυτόματα ταξινομούνται σε κατηγορίες ή διαχωρίζονται σε ομάδες με βάση κάποια κριτήρια.

Αξίζει να σημειωθεί πως η βιβλιοθήκη OpenCV θεωρείται μία από τις καλύτερες στον τομέα της επεξεργασίας εικόνας, αφού περιέχει κατανοητές συναρτήσεις και μεγάλη ταχύτητα εκτέλεσης προγραμμάτων. Αν γράφετε στη C/ C++ παρέχετε απευθείας τον κώδικα γλώσσας μηχανής στον υπολογιστή, έχοντας έτσι περισσότερη επεξεργασία εικόνας, και όχι περισσότερη διερμηνεία. Ως αποτέλεσμα, μεγαλύτερη αποτελεσματικότητα. Είναι σύνηθες, για αυτή την εργασία να προτιμάται αυτή αντί του Matlab λογισμικού.

Παρόλα τα αρκετά πλεονεκτήματα που χαρακτηρίζουν αυτή τη βιβλιοθήκη, ένα από τα μεγαλύτερα μειονεκτήματα της είναι η απουσία συνάρτησης plot, για τη σχεδίαση και απεικόνιση γραφημάτων. Επομένως, θα πρέπει ο χρήστης να σχεδιάσει πρώτα το γράφημα σε μια εικόνα και ύστερα να την εμφανίσει την ίδια την εικόνα με την κατάλληλη συνάρτηση.

ΠΑΡΑΡΤΗΜΑ Α’

Εγκατάσταση της OpenCV 2.4.9 στο Microsoft Visual Studio 2012 για Windows 10 64-bit

1. Κατεβάζουμε την έκδοση 2.4.9 της OpenCV για Windows από τον ιστότοπο:

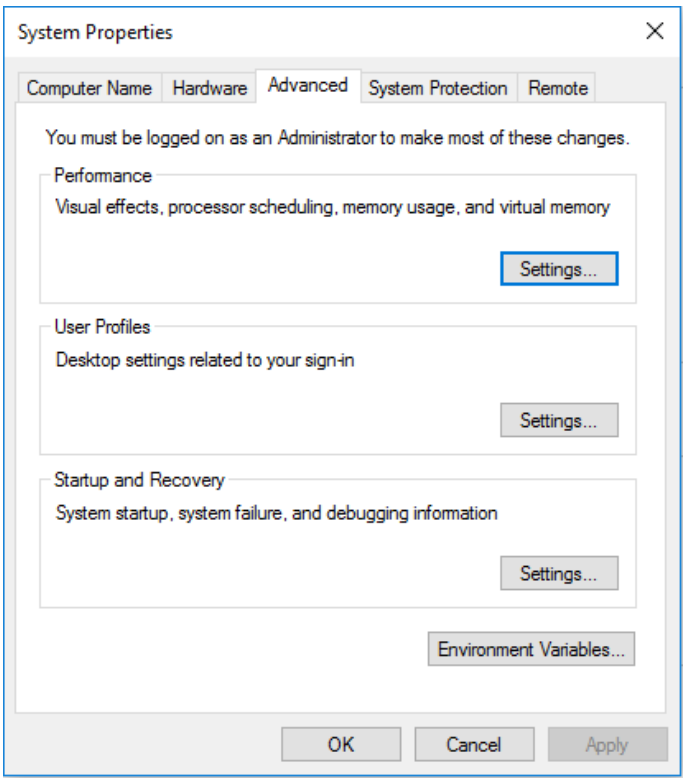
http://opencv.org/downloads.html

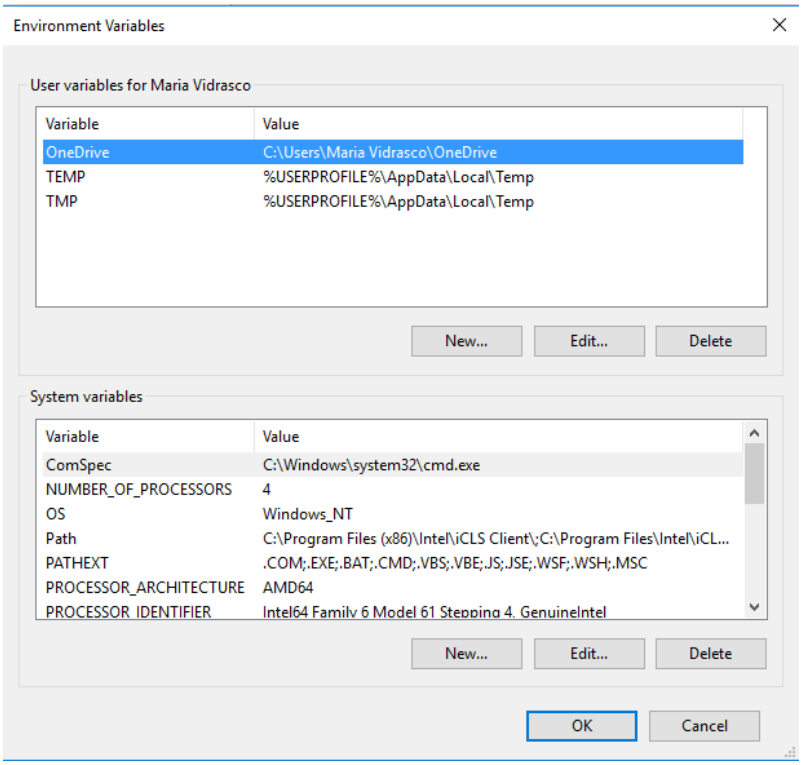
2. Αποσυμπιέζουμε το αρχείο *opencv-2.4.9.exe* στον φάκελο *C:/opencv*.

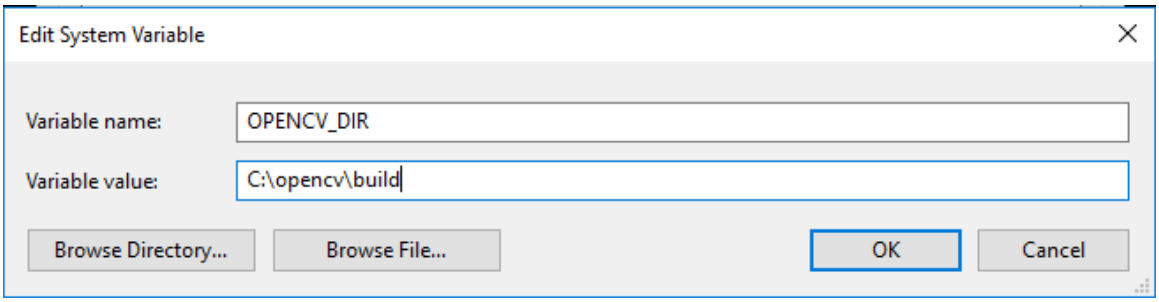
3. Για την τοποθέτηση του path, επιλέγουμε:

(a)

Control Panel ⇨ System and Security ⇨ System ⇨ Advanced system settings ⇨ Environment Variables ⇨ System variables ⇨ New ⇨ Variable name=OPENCV_DIR, Variable value=C:\opencv\build ⇨ OK

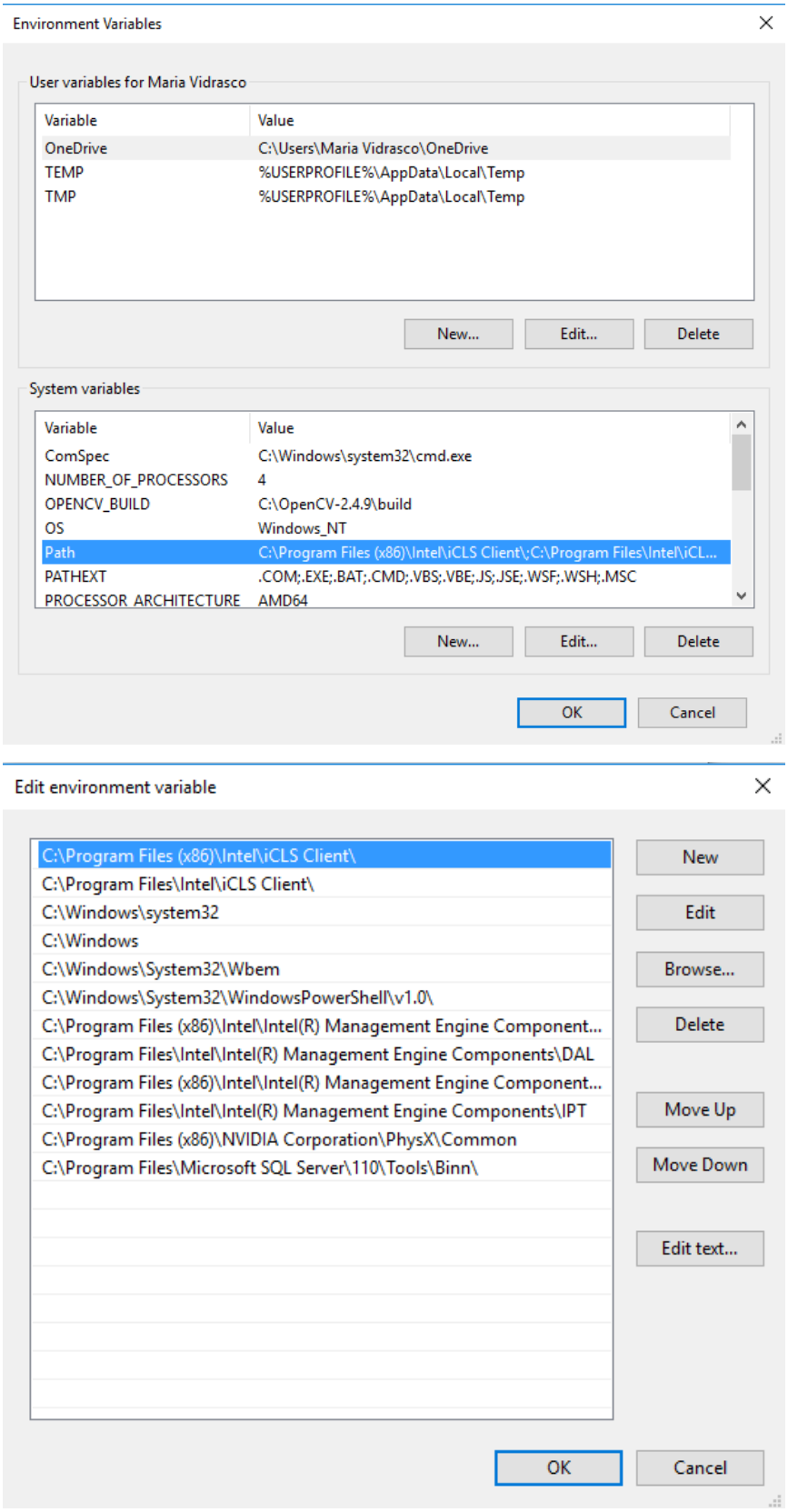


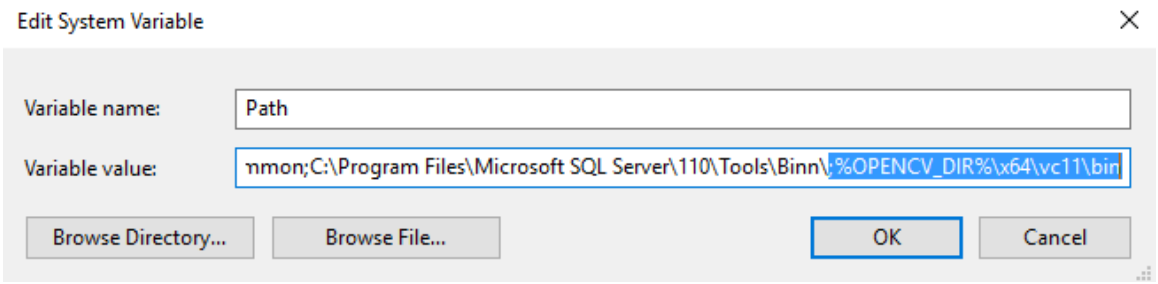




(b)

System variables ⇨ Path ⇨ Edit ⇨ Edit Text… προσθέτουμε στο τέλος του Variable value (αφού βάλουμε ένα διαχωριστικό ;) %OPENCV_DIR%\x64\vc11\bin ⇨ OK

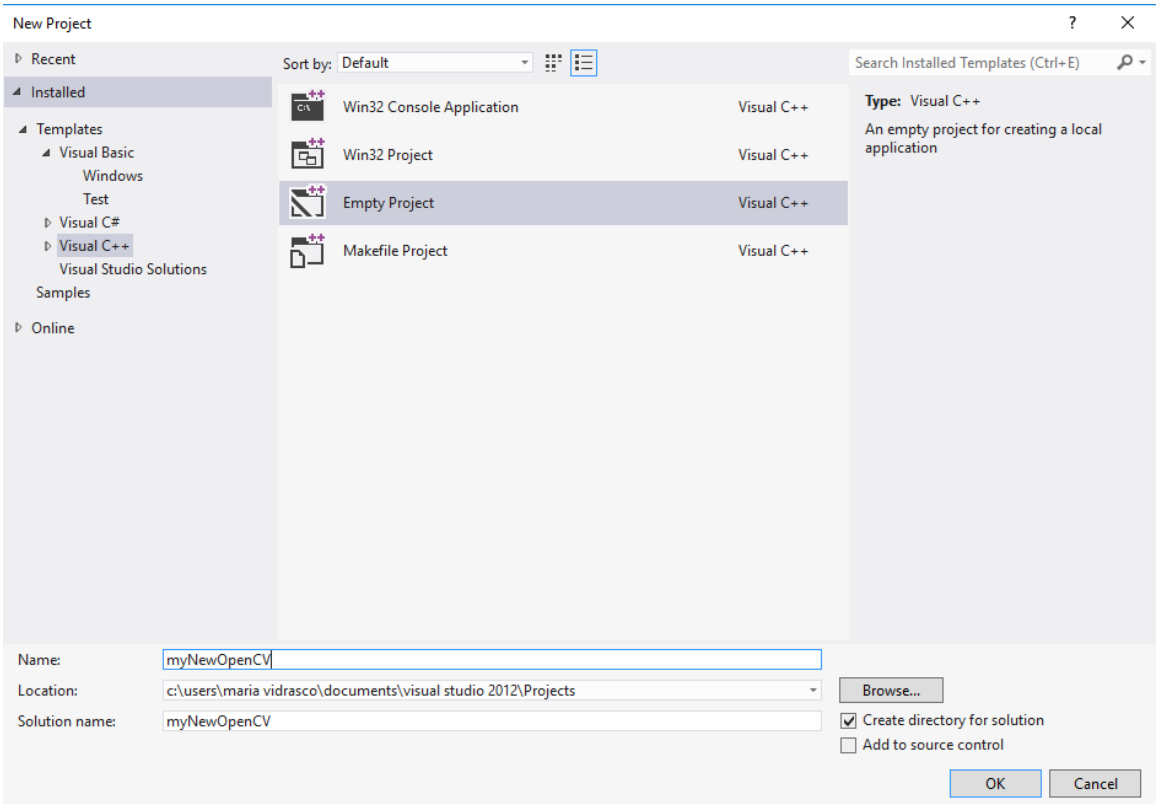

Environment Variables
User variables for Maria Vidrasco
Variable
Value
OneDrive
C:\Users\Maria Vidrasco\OneDrive
TEMP
%USERPROFILE%\AppData\Local\Temp
TMP
%USERPROFILE%\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
ComSpec
C:\Windows\system32\cmd.exe
NUMBER_OF_PROCESSORS
4
OPENCV_BUILD
C:\OpenCV-2.4.9\build
OS
Windows_NT
Path
C:\Program Files (x86)\Intel\iCLS Client\;C:\Program Files\Intel\iCL...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE
AMD64
New...
Edit...
Delete
OK
Cancel
Edit environment variable
C:\Program Files (x86)\Intel\iCLS Client\
C:\Program Files\Intel\iCLS Client\
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Program Files (x86)\Intel\Intel(R) Management Engine Component...
C:\Program Files\Intel\Intel(R) Management Engine Components\DAL
C:\Program Files (x86)\Intel\Intel(R) Management Engine Component...
C:\Program Files\Intel\Intel(R) Management Engine Components\IPT
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
C:\Program Files\Microsoft SQL Server\110\Tools\Binn\
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel



Πατάμε τέλος το OK και κάνουμε Restart τον υπολογιστή.

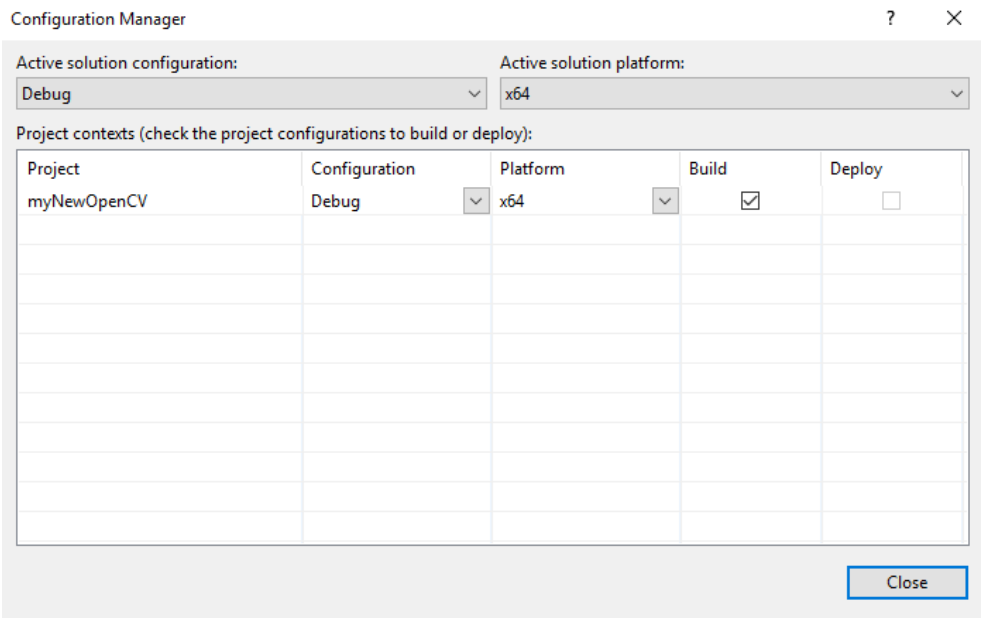
Εάν τα Windows είναι 32 bit: %OPENCV_DIR%\x86\vc11\bin

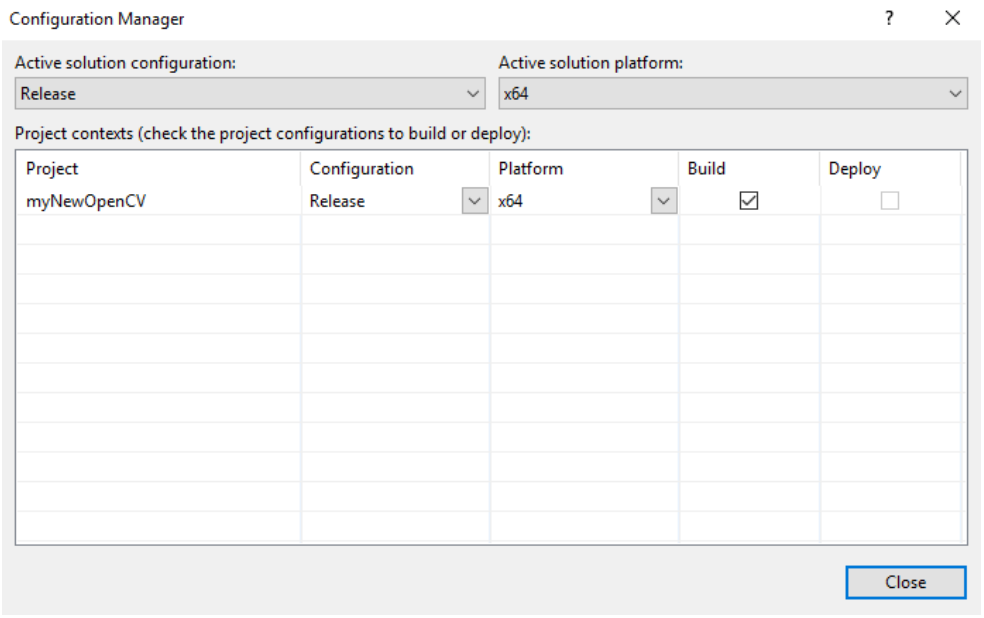
4. Ανοίγουμε το Visual Studio και επιλέγουμε File ⇨ New ⇨ Project. Επιλέγουμε Empty Project, τοποθετούμε σαν όνομα αρχείου το myNewOpenCV και κλείνουμε με OK.

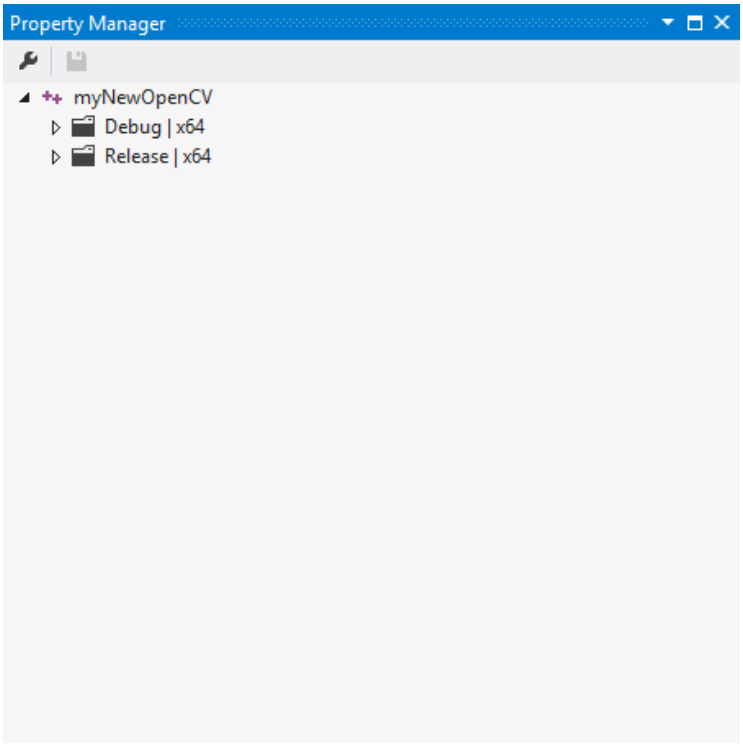


5. Επιλέγουμε τον Property Manager (στο κάτω μέρος της οθόνης ή στο Other Windows) και στην οθόνη εμφανίζονται τα δύο μέρη του project (Debug|Win32 και Release|Win32).

Για Windows 64 bit επιλέγουμε BUILD ⇨ Configuration Manager και με Configuration=Debug κάνουμε το Platform=x64 επιλέγοντας New ⇨ x64. Έτσι δημιουργούνται τα Debug|x64 και Release|x64, ενώ μπορούμε να διαγράψουμε τα Debug|Win32 και Release|Win32.







Για κάθε ένα από αυτά θα πραγματοποιήσουμε την ακόλουθη διαδικασία:

Debug ⇨ δεξί κλικ ⇨ Add New Project Property Sheet ⇨ Name: OPENCV_DEBUG ⇨ Add.

Επιλέγουμε:

OPENCV_DEBUG ⇨ Properties ⇨ C/C++ ⇨ Additional Include Directories ⇨ Edit ⇨ Τοποθετούμε το: $(OPENCV_DIR)\include ⇨ OK

Linker ⇨ General ⇨ Additional Library Directories ⇨ Edit ⇨ Τοποθετούμε το: $(OPENCV_DIR)**x64**\vc11\lib ⇨ OK

Για Windows 32 bit τοποθετούμε $(OPENCV_DIR)**x86**\vc11\lib ⇨ OK

Linker ⇨ Input ⇨ Additional Dependencies ⇨ Edit ⇨ Τοποθετούμε τα:

opencv_calib3d249d.lib

opencv_contrib249d.lib

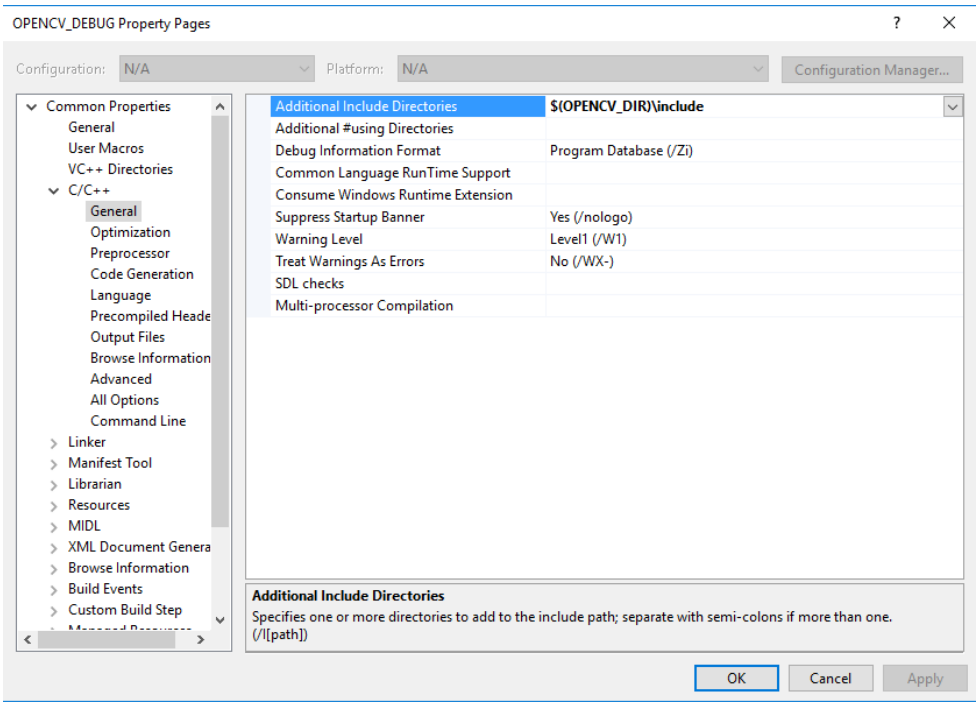
opencv_core249d.lib

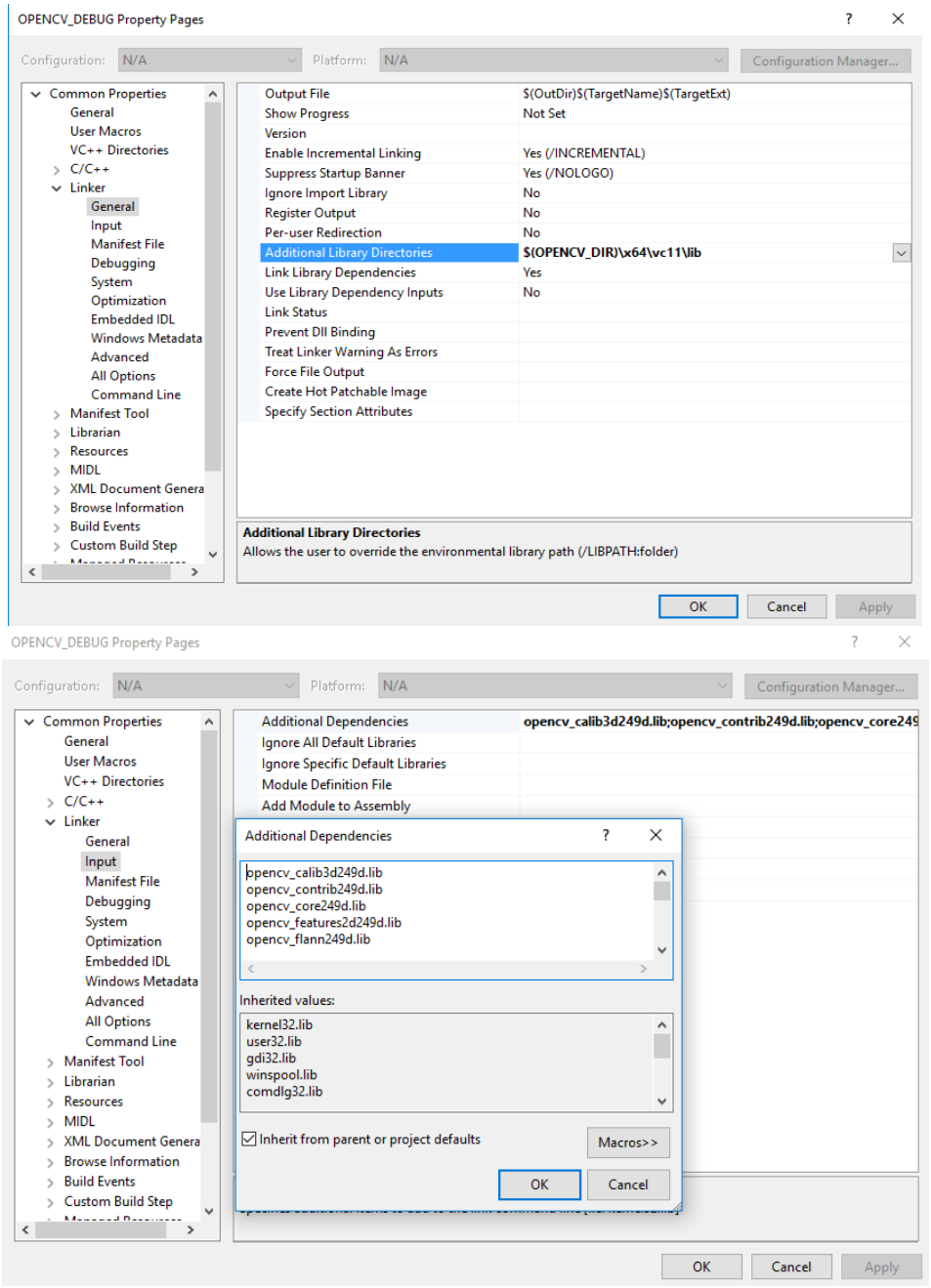
opencv_features2d249d.lib

opencv_flann249d.lib

opencv_gpu249d.lib
opencv_highgui249d.lib
opencv_imgproc249d.lib
opencv_legacy249d.lib
opencv_ml249d.lib
opencv_nonfree249d.lib
opencv_objdetect249d.lib
opencv_ocl249d.lib
opencv_photo249d.lib
opencv_stitching249d.lib
opencv_superres249d.lib
opencv_ts249d.lib
opencv_video249d.lib
opencv_videostab249d.lib

και πατάμε OK. Το *d* υποδεικνύει ότι πρόκειται για debugging.





6. Επαναλαμβάνουμε ακριβώς τα ίδια βήματα της προηγούμενης διαδικασίας για το Release|Win32, με τη μόνη διαφορά ότι στο τελευταίο βήμα οι βιβλιοθήκες δεν θα έχουν το γράμμα *d*.

7. Οι δύο αυτές ρυθμίσεις μένουν στον φάκελο του project που δημιουργήσαμε σαν:

Projects/myNewOpenCV/myNewOpenCV/OPENCV_DEBUG.props και *Projects/myNewOpenCV/myNewOpenCV/OPENCV_RELEASE.props*.

Αντιγράφουμε αυτά τα δύο αρχεία σε έναν ξεχωριστό φάκελο και όταν δημιουργούμε ένα νέο project αντί να κάνουμε την παραπάνω διαδικασία απλά επιλέγουμε το “Add Existing Property Sheet...” για να προσθέσουμε αυτές τις ιδιότητες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία:

[1] Ι.Ν. Έλληνας, "ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ & ΒΙΝΤΕΟ: Από τη Θεωρία στη Πράξη", 2010

[2] Rafael C. Gonzales, Richard E. Woods, "Digital Image Processing 2nd edition", 2002

[3] Michael Halvorson, "Microsoft Visual Basic 2008: Βήμα Βήμα", 2008

[4] Robert Laganiere, "OpenCV 2 Computer Vision Application Programming Cookbook", 2011

[5] Samarth Brahmbhatt, "Practical OpenCV", 2013

Πτυχιακές και Διπλωματικές εργασίες:

[6] Καργοπούλου Ωραιοζήλη, "Τεχνικές επεξεργασίας για βελτιστοποίηση υπερηχογραφικών εικόνων και εξαγωγή χαρακτηριστικών, με χρήση του Matlab", Διπλωματική εργασία, Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας, 2011

[7] Παπαρήγα Στεφανία, "ΣΥΣΤΗΜΑ ΑΝΑΓΝΩΡΙΣΗΣ ΚΑΙ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ ΚΙΝΗΣΗΣ ΣΕ ΕΛΕΓΧΟΜΕΝΟ ΧΩΡΟ ΜΕ ΧΡΗΣΗ WEBCAMERA ΚΑΙ OPENCV", Πτυχιακή εργασία, Τμήμα Μηχανικών Πληροφορικής ΤΕ, ΤΕΙ Κεντρικής Μακεδονίας, 2013

[8] Γιαννόπουλος Ευθύμιος "Σχεδιασμός - Υλοποίηση Ενσωματωμένου Συστήματος για Κίνηση και Έλεγχο Οχήματος με τη βοήθεια Ασύρματης Επικοινωνίας", Διπλωματική εργασία, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών, Πανεπιστήμιο Πατρών, 2014

[9] Βερύκοκου Στυλιανή, "ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΕΠΑΥΞΗΜΕΝΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ ΒΑΣΕΙ ΕΠΙΠΕΔΟΥ ΠΡΟΤΥΠΟΥ", Διπλωματική εργασία, Σχολή Αγρονόμων και Τοπογράφων Μηχανικών, Εθνικό Μετσόβιο Πολυτεχνείο, 2013

[10] Πρωτονοτάριος Ιωάννης, "Ανάπτυξη συστήματος ενσωματωμένων αισθητήρων για ασύρματη μετάδοση εικόνας και δεδομένων", Πτυχιακή εργασία, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών, Πανεπιστήμιο Πατρών, 2011

[11] Ρήγος Στέφανος - Μάρκος "Βελτιστοποίηση της βιβλιοθήκης OpenCV με την χρήση εξειδικευμένου hardware", Πτυχιακή εργασία, Τμήμα Τηλεπ/κων Συστημάτων και Δικτύων, ΤΕΙ Μεσολογγίου, 2011

Ιστότοποι:

[12] https://en.wikipedia.org/wiki/Binary_image

[13] https://en.wikipedia.org/wiki/Color_image

[14] https://www.robotix.in/tutorial/imageprocessing/imagetypes/

[15] https://el.wikipedia.org/wiki/Επεξεργασία_εικόνας

[16] https://el.wikipedia.org/wiki/Εικονοστοιχείο

[17] https://en.wikipedia.org/wiki/Digital_image_processing

[18] https://en.wikipedia.org/wiki/Library_(computing)

[19] https://en.wikipedia.org/wiki/OpenCV

[20] https://www.packtpub.com/mapt/book/Application+Development/9781782168812/1/ch01lvl1sec03/so,-what-is-opencv%3F

[21] http://opencv.org/

[22] http://opencv.org/about.html

[23] http://docs.opencv.org/2.4/

[24] https://www.safaribooksonline.com/library/view/learning-opencv-3/9781491937983/ch04.html

[25] http://image.diku.dk/mediawiki/index.php/OpenCV

[26] https://www.learnopencv.com/opencv-c-vs-python-vs-matlab-for-computer-vision/

[27] https://karanjthakkar.wordpress.com/2012/11/21/what-is-opencv-opencv-vs-matlab/

[28] https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[29] https://www.seeklogo.net/technology-logos/visual-studio-2012-9864.html

[30] http://dovgalecs.com/blog/opencv-matrix-types/

[31] http://ninghang.blogspot.gr/2012/11/list-of-mat-type-in-opencv.html

[32] https://codeyarns.com/2015/08/27/depth-and-type-of-matrix-in-opencv/

[33] https://www.thoughtco.com/definition-of-float-958293

[34] https://el.wikipedia.org/wiki/RGB

[35] http://www.intelligence.tuc.gr/~petrakis/courses/computervision/color.pdf

[36] http://eclass.teipir.gr/openeclass/modules/document/file.php/HYS107/I.%20Διαφάνειες%20Θεωρίας/II.%20Βελτιστοποίηση%20Εικόνας.pdf

[37] https://devendrapratapyadav.github.io/Image_transform/

[38] http://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm

[39] http://www.programming-techniques.com/2013/01/contrast-stretching-using-c-and-opencv.html

[40] https://en.wikipedia.org/wiki/Thresholding_(image_processing)

[41] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT2/node3.html

[42] https://en.wikipedia.org/wiki/Kernel_(image_processing)

[43] https://www.slideshare.net/gichelleamon/6-spatial-filtering-p2

[44] http://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm

[45] http://opencv-srf.blogspot.gr/2013/10/smooth-images.html

[46] http://people.ucalgary.ca/~dasaid/CPSC535/lab7.html

[47] https://blog.saush.com/2011/04/20/edge-detection-with-the-sobel-operator-in-ruby/

[48] https://en.wikipedia.org/wiki/Sobel_operator

[49] http://www.cse.dmu.ac.uk/~sexton/WWWPages/HIPR/html/log.html

[50] http://what-when-how.com/embedded-image-processing-on-the-tms320c6000-dsp/non-linear-filtering-of-images-image-processing-part-1/