Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Τίτλος: "Ανάπτυξη ενός αποδοτικού αλγόριθμου για την αναγνώριση των αποτελεσμάτων ενός ή περισσοτέρων τεστ πολλαπλής επιλογής."

Μάντζιος Χρήστος (Α.Μ 2334)

Επιβλέπων καθηγητής: Νικόλαος Παπαδάκης

Abstract

The multiple choice questions are a popular way to review, which has prevailed in educational practice. The advantages obtained by the above method makes it ideal than others. More specifically, allowing simultaneous examination of a large number test a large volume of the syllabus, ensure objectivity and impartiality while their creation is relatively easy and quick affair. However, there are also disadvantages of this method with this important graduation from their teachers, which is certainly time consuming.

Already by the 1890s, originated the punched cards bringing a revolution in data collection and storage. The punched cards were the precursor technology optical mark recognition (OMR). Technology evolution punch cards, was the creation of appliances from IBM, which swept the surface of the paper with metal brushes and through conductivity graphite recognizes the user responses.

Later, with the new digital revolution and the emergence of the first personal computer, created optical scanners, which swept preprinted forms that were on special paper, allowing the passage of light and measured the amount of light that ran the paper, so that marks found on both sides of the paper to reduce the amount of light that passes through the document.

In other way optical scanners, the OMR software does not need specialized hardware nor preprinted forms, as for result in a very low cost, making the automatic recognition test multiple choice are very affordable to educational institutions, small organizations etc.

For this reason, we created an automated correction system multiple choice test with the use of OpenCV, where the verification of user options, ie whether the user has marked a choice be made with the use of support vector machines (support vector machines or SVM), which belongs to the field of machine learning.

Σύνοψη

Οι ερωτήσεις πολλαπλής επιλογής αποτελούν έναν δημοφιλή τρόπο εξέτασης, που έχει επικρατήσει στην εκπαιδευτική πρακτική. Τα πλεονεκτήματα που συγκεντρώνει η παραπάνω μέθοδος την καθιστούν ιδανική σε σχέση με άλλες. Πιο συγκεκριμένα, επιτρέπουν την ταυτόχρονη εξέταση μεγάλου αριθμού εξεταζομένων σε μεγάλο όγκο της εξεταστέας ύλης, εξασφαλίζουν αντικειμενικότητα και αμεροληψία ενώ η δημιουργία τους είναι σχετικώς εύκολη και ταχεία υπόθεση. Ωστόσο, δεν λείπουν και τα μειονεκτήματα της μεθόδου αυτής με σημαντικότερο αυτό της βαθμολόγησης τους από τους από τους διδάσκοντες, η οποία είναι οπωσδήποτε χρονοβόρα.

Ήδη από τη δεκαετία του 1890, πρωτοεμφανιστήκαν οι διάτρητες κάρτες φέρνοντας μια επανάσταση στον τρόπο συλλογής και αποθήκευσης δεδομένων. Οι διάτρητες κάρτες αποτέλεσαν πρόδρομο της τεχνολογίας της οπτικής αναγνώρισης σημαδιών (optical mark recognition ή OMR).

Εξέλιξη της τεχνολογίας διάτρητων καρτών, υπήρξε η δημιουργία συσκευών από την IBM, οι οποίες σάρωναν την επιφάνεια του χαρτιού με μεταλλικές βούρτσες και μέσω της αγωγιμότητας του γραφίτη αναγνώριζαν τις απαντήσεις του χρήστη.

Αργότερα, με τη νέα ψηφιακή επανάσταση και την δημιουργία των πρώτων προσωπικών υπολογιστών, δημιουργήθηκαν οπτικοί σαρωτές, οι οποίοι σάρωναν φόρμες που ήταν προτυπωμένες σε ειδικό χαρτί, που επέτρεπε την διέλευση του φωτός και μετρούσαν την ποσότητα του φωτός που διέτρεχε το χαρτί, έτσι ώστε τα σημάδια που βρίσκονται και στις δύο πλευρές του εγγράφου να μειώνουν την ποσότητα του φωτός που περνά μέσα από το έγγραφο.

Σε αντίθεση με τους οπτικούς σαρωτές, το OMR λογισμικό δεν χρειάζεται εξειδικευμένο υλικό αλλά ούτε και προτυπωμένες φόρμες, με αποτέλεσμα το πολύ χαμηλό κόστος, κάνοντας έτσι την αυτόματη αναγνώριση τεστ πολλαπλής επιλογής πολύ προσιτή σε εκπαιδευτικά ιδρύματα, μικρούς οργανισμούς κ.α.

Για το λόγο αυτό, δημιουργήσαμε ένα αυτοματοποιημένο σύστημα διόρθωσης τεστ πολλαπλής επιλογής με την χρήση της OpenCV, όπου η επαλήθευση των επιλογών του χρήστη, δηλαδή το αν ο χρήστης έχει μαρκάρει μια επιλογή, να γίνεται με τη χρήση των μηχανών διανυσμάτων υποστήριξης (support vector machines ή SVM), οι οποίες ανήκουν στον κλάδο της μηχανικής μάθησης.

Ευρετήριο περιεχομένων

Ευρετήριο εικόνων

Ευρετήριο πινάκων

1. Εισαγωγή

Η παρούσα πτυχιακή εργασία πραγματεύεται την επεξεργασία εικόνας σε περιβάλλον Visual Studio 2012 και τη χρήση της βιβλιοθήκης OpenCV σε γλώσσα προγραμματισμού C++ και ειδικότερα έχει ως αντικείμενο την οπτική αναγνώριση σημαδιών (OMR), η οποία αποτελεί μια ειδίκευση της οπτικής αναγνώρισης χαρακτήρων .

Η διαφορά μεταξύ των δύο αυτών τεχνικών, αν και έχουν πολλά κοινά μεταξύ τους, εντοπίζεται στον τρόπο με τον οποίο γίνεται η αναγνώριση. Στην περίπτωση της οπτικής αναγνώρισης χαρακτήρων, τα πρότυπα που πρέπει να αναγνωριστούν είναι άγνωστα σε εμάς και για το λόγο αυτό συχνά η ανάγνωση χαρακτήρων, ειδικά όταν είναι χειρόγραφη, είναι πολύ δύσκολη έως και αδύνατη. Αντίθετα, στην περίπτωση της οπτικής αναγνώρισης σημαδιών τα πρότυπα είναι γνωστά και τα σημάδια που πρέπει να αναγνωριστούν είναι αυστηρώς προκαθορισμένα, τόσο ως προς τη θέση τους στο έγγραφο, όσο και ως προς το σχήμα τους.

Η οπτική αναγνώριση σημαδιών (OMR) ξεκίνησε στα τέλη του 19^{ου} αιώνα με τη μορφή διάτρητων καρτών που χρησιμοποιούνταν για τον προγραμματισμό μηχανών παραγωγής (π.χ. πλεκτικές μηχανές στην υφαντουργία). Έπειτα, με την δημιουργία των πρώτων υπολογιστών χρησιμοποιήθηκαν για τον προγραμματισμό αυτών, όπως και για την εισαγωγή δεδομένων. Εξέλιξη της τεχνολογίας διάτρητων καρτών, υπήρξε η δημιουργία συσκευών από την IBM, οι οποίες σάρωναν την επιφάνεια του χαρτιού με μεταλλικές βούρτσες και μέσω της αγωγιμότητας του γραφίτη αναγνώριζαν τις απαντήσεις του χρήστη.

Τέλος, στα τέλη της δεκαετίας του 1970 με την δημιουργία των προσωπικών υπολογιστών η οπτική αναγνώριση σημαδιών χρησιμοποιήθηκε σε πληθώρα εφαρμογών, όπως στην διεξαγωγή τεστ, δημοσκοπήσεων, ψηφοφοριών, καθώς και σε εφαρμογές, όπως οι συσκευές διπλώματος εγγράφων για την εισαγωγή τους σε φάκελο κ.α..

1.1 Κίνητρο για την διεξαγωγή της εργασίας

Οι κλασσικές μέθοδοι εξέτασης εμφανίζουν συγκεκριμένα χαρακτηριστικά. Ως τέτοιες νοούνται οι γραπτές ή προφορικές εξετάσεις από τον διδάσκοντα, ο οποίος μέσω αυτών αποπειράται να αξιολογήσει τον εξεταζόμενο και την εκπαιδευτική του πορεία εν γένει. Ωστόσο, η εξαγωγή δίκαιων και ορθών συμπερασμάτων καθίσταται ιδιαίτερα δυσχερής για τον διδάσκοντα. Κατ' αρχάς, δεν είναι δυνατή η συχνή εξέταση του εξεταζομένου, καθώς η διεξαγωγή ενός γραπτού διαγωνίσματος ή προφορικής εξέτασης απαιτεί την ύπαρξη χρόνου, ο οποίος σε όλο το φάσμα της εκπαιδευτικής διαδικασίας σπανίζει ή οι προτεραιότητες για τον διδάσκοντα είναι άλλες (π.χ. η κάλυψη της διδακτικής ύλης). Έπειτα η γραπτή και κυρίως η προφορική εξέταση κλασσικού τύπου απευθύνεται σε έναν στενότερο κύκλο προσώπων - εξεταζομένων, πράγμα που δεν παρέχει ικανοποιητικά επίπεδα αντικειμενικότητας. Τέλος, σημαντικό μέρος της εξεταστέας ύλης μένει εκτός της εξεταστικής διαδικασίας, αφού ο διδάσκων υποχρεώνεται να αξιολογήσει τον εξεταζόμενο σε ένα μόνο μέρος της διδακτέας ύλης και όχι στο σύνολο της.

Οι ως άνω αδυναμίες οδήγησαν στην ανάπτυξη σύγχρονων και καινοτόμων μεθόδων εξέτασης, αιχμή των οποίων αποτελεί η μέθοδος των ερωτήσεων πολλαπλής επιλογής. Με τη μέθοδο αυτή αντιμετωπίζονται οι ως άνω πραγματικές δυσχέρειες εξασφαλίζοντας αντικειμενικότητα, καθολικότητα και συχνότητα στην διεξαγωγή εξετάσεων σε βαθμό μείζονα συγκριτικά με τις κλασσικές εξεταστικές μεθόδους.

Επίσης, σημαντικό κίνητρο στάθηκε και ότι τα τελευταία χρόνια ο κλάδος της επεξεργασίας εικόνας έχει αναπτυχθεί εντυπωσιακά και έχει διεισδύσει σε πολλούς τομείς της καθημερινότητας.

Επιπλέον, η δημιουργία της OpenCV από την Intel, η διανομή της ως ανοιχτό λογισμικό, καθώς και η εξάπλωση των Smartphone's έκαναν την επεξεργασία εικόνας έναν δυναμικά αναπτυσσόμενο κλάδο.

Τέλος, κατά τη διάρκεια της φοίτησης μου είχα την ευκαιρία να έρθω σε επαφή, μέσω των μαθημάτων του προγράμματος σπουδών, με την επεξεργασία εικόνας όπως και την αναγνώριση

προτύπων. που μου έδωσαν ένα επιπλέον κίνητρο για την συγγραφή της παρούσας πτυχιακής εργασίας.

1.2 Σκοπός και στόχοι εργασίας

Σκοπός της εργασίας αυτής είναι η ανάπτυξη ενός ολοκληρωμένου συστήματος αυτόματης αναγνώρισης των απαντημένων ερωτήσεων για τεστ πολλαπλής επιλογής, το οποίο θα είναι απλό στη χρήση του και δεν θα έχει μεγάλες απαιτήσεις σε πόρους, ώστε να είναι συμβατό και με παλαιότερης τεχνολογίας υπολογιστές.

Επίσης, η λειτουργιά του θα είναι πολύ απλή. Η διόρθωση θα ξεκινά αυτόματα, μόλις ο χρήστης σύρει (drag n drop) πάνω στην κονσόλα τον φάκελο με τις φωτογραφίες. Αρχικά το πρόγραμμα θα αναγνωρίζει τις εικόνες και έπειτα θα αποθηκεύει τα αποτελέσματα σε μορφή csv, η οποία είναι συμβατή με το excel.

Αναλυτικότερα, το σύστημα μας θα είναι σε θέση να αναγνωρίσει της απαντήσεις που έδωσε ο εξεταζόμενος σε μια πρότυπη φόρμα απαντήσεων σταθερού μεγέθους ερωτήσεων, το οποίο ορίστηκε να είναι 80 ερωτήσεις. Αφού ο διδάσκων συγκεντρώσει το σύνολο των OMR εγγράφων θα μπορεί να τα εισαγάγει σε έναν σαρωτή γενικού σκοπού ή κατά προτίμηση σε ένα πολυμηχάνημα με αυτόματο τροφοδότη εγγράφων, έτσι ώστε η διαδικασία σάρωσης να γίνει απλούστερη και ευχερέστερη.

1.3 Δομή εργασίας

Στο κεφάλαιο 1 γίνεται μια εισαγωγή σχετικά με την εργασία, καθώς επίσης καθορίζονται οι στόχοι και ο σκοπός της.

Στο κεφάλαιο 2 γίνεται αναφορά σε προηγούμενες εργασίες σχετικά με την οπτική αναγνώριση σημαδιών.

Στο κεφάλαιο 3 αναλύεται το υπόβαθρο της εργασίας που έχει να κάνει με την οπτική αναγνώριση σημαδιών (OMR), την τεχνολογία των QR code, καθώς και των μηχανών υποστήριξης διανυσμάτων (SVM).

Στο κεφάλαιο 4 παρατίθενται όλες οι απαραίτητες πληροφορίες σχετικά με τα εργαλεία που χρησιμοποιήσαμε.

Στο κεφάλαιο 5 γίνεται αναλυτική περιγραφή του αλγορίθμου που αναπτύξαμε για την αυτοματοποιημένη διόρθωση των τεστ.

Στο κεφάλαιο 6 παρατίθενται όλες οι μετρήσεις σχετικά με την αποτελεσματικότητα του συστήματος μας.

Τέλος στο κεφάλαιο 7 γίνεται μια αναφορά σε μελλοντικές επεκτάσεις και τροποποιήσεις που θα μπορούσαν να γίνουν μελλοντικά στο σύστημα.

2. Σχετικές εργασίες

Dillman, 2000: Στην εργασία με τίτλο "Optical Mark Recognition (OMR)" παρουσιάζονται οι επιπτώσεις των OMR φορμών στο ρυθμό απόκρισης. Ένα πιθανό μειονέκτημα των ερευνών που γίνονται με την χρήση της τεχνολογίας OMR είναι ότι μπορεί να δημιουργήσουν πρόβλημα στο ρυθμό απόκρισης. Αυτό μπορεί να συμβεί για διάφορους λόγους. Οι OMR έρευνες χρησιμοποιούνται συχνά σε συνδυασμό με άλλα μέτρα μείωσης του κόστους, ώστε τα τόσο χαμηλά ποσοστά απόκρισης τους μπορεί να είναι απλώς αποτέλεσμα που σχετίζεται με επιλογές που αφορούν την διαχείριση της έρευνας (Dillman, 2000).

Symons, 2001: Δηλώνει ότι οι MCQ αναγνωρίζονται γενικά ως μια ερώτηση με κάποιες εναλλακτικές απαντήσεις, όπου η μια είναι σωστή και οι άλλες λανθασμένες. Αυτό σημαίνει πολλαπλής επιλογής αλλά υπάρχουν και άλλες παραλλαγές, όπως πολλαπλής επιλογής, όπου επιλέγεται οποιοσδήποτε αριθμός από τις σωστές απαντήσεις, αριθμητικές, πολλαπλής με την ένδειξη Σωστό-Λάθος (όπου η απάντηση είναι αληθής ή ψευδής ή ο ερωτώμενος μπορεί και να μην απαντήσει), ή ερωτήσεις σχόλια (χώρος για ποιοτικές απαντήσεις).

Hussmann S. et al, 2005: Παρουσιάζεται η εργασία με τίτλο "Υψηλής ταχύτητας οπτικός αναγνώστης σημαδιών που υλοποιείται σε χαμηλού κόστους υλικό βασισμένο σε FPGA". Στο περιεχόμενο της περιγράφεται η ανάπτυξη ενός πρωτοτύπου χαμηλού κόστους και ενός υψηλής ταχύτητας συστήματος OMR για την ανάγνωση ερωτήσεων πολλαπλής επιλογής. Η καινοτομία της προσέγγισης αυτής είναι η εφαρμογή του ολοκληρωμένου συστήματος σε ένα ενιαίο χαμηλού κόστους Field Programmable Gate Array (FPGA), προκειμένου να επιτευχθεί υψηλή ταχύτητα επεξεργασίας. Αποτελεσματικοί αλγόριθμοι ανίχνευσης σημαδιών και επαλήθευσης έχουν αναπτυχθεί και εφαρμοστεί για την επίτευξη των επιδόσεων σε πραγματικό χρόνο με χαμηλό υπολογιστικό κόστος. Το σύστημα αυτό μπορεί να επεξεργαστεί δεδομένα από ένα υψηλής ανάλυσης CCD γραμμικό αισθητήρα με 3456 pixels στα 5000 καρέ/s με τον πραγματικό μέγιστο ρυθμό του ρολογιού του αισθητήρα 20 MHz (4 × 5 MHz). Η απόδοση του πρωτοτύπου συστήματος ελέγχθηκε για διαφορετικά χρώματα στυλό και για διαφορετικές μεθόδους συμπλήρωσης (Hussmann & Weiping Deng, 2005).

Sabyasachi Das, 2010: Στην εργασία με τίτλο "Οπτική αναγνώριση σημαδιών για συλλογή δεδομένων αγροτικής υγείας" παρουσιάζεται η OMR τεχνολογία που έχει χρησιμοποιηθεί για διάφορες μεγάλης κλίμακας έρευνες από κυβερνητικούς οργανισμούς μέχρι εταιρείες μικροχρηματοδοτήσεων, όπως η Equitas για να συλλέγουν δεδομένα από νέους πελάτες. Η Equitas έχει διαπιστώσει τριπλάσια αύξηση της απόδοσης σε σχέση με τη χειροκίνητη εισαγωγή δεδομένων (Das, 2010).

Akash, Tewari, & Jain, 2012: Η εργασία με τίτλο "Computer vision based omr sheet evaluation using OpenCV" εστιάζει στην δημιουργία μια εφαρμογής διόρθωσης τεστ πολλαπλής επιλογής, η οποία θα βασίζεται σε εικόνες που προέρχονται από μια web camera αντί ενός σαρωτή. Αυτό έχει ως αποτέλεσμα την εξεύρεση μιας χαμηλού κόστους OMR λύσης, που θα δύναται να εφαρμοστεί από μικρές επιχειρήσεις παροχής υπηρεσιών ή προπονητικά κέντρα κ.α (Akash, Tewari, & Jain, 2012).

Chidrewar, Yang, & Moon, 2013: Στην εργασία τους με τίτλο "Mobile Based Auto Grading Of Answersheets" ερευνάται ένας αλγόριθμος αυτόματης διόρθωσης τεστ πολλαπλής επιλογής που θα υλοποιείται μέσω της χρήσης κινητού τηλεφώνου με λειτουργικό σύστημα Android αντί της χρήσεως σαρωτή. Το βασικό πλεονέκτημα αυτής της μεθόδου είναι η μείωση του κόστους, καθώς τα κινητά τηλέφωνα που βασίζονται στο λειτουργικό σύστημα Android είναι πλέον πολύ προσιτά σε οικονομικό επίπεδο ενώ ταυτόχρονα καταλαμβάνουν το μεγαλύτερο μερίδιο της αγοράς (Chidrewar, Yang, & Moon, 2013).

Accusoft: Η Accusoft παρουσίασε ένα Software Development Kit (SDK) για OMR αναγνώριση από εικόνες εγγράφων. Το SDK υποστηρίζει λειτουργία αναγνώρισης πρότυπου και λειτουργία ελεύθερης αναγνώρισης. Ένα OMR τεστ αποτελείται από μια παραλληλόγραμμη περιοχή που περιέχει προκαθορισμένο αριθμό στηλών και γραμμών από κυκλικά σημεία στα οποία σημειώνεται η απάντηση. Το SDK μπορεί να σαρώσει την περιοχή οριζόντια και στη συνέχεια κάθετα για να εντοπίσει τις κυκλικές περιοχές εκτός από τα κενά μεταξύ τους. Και στην συνέχεια αφού αναγνωρίσει τις κυκλικές περιοχές, καταμετρά τα πίξελ μαύρου χρώματος και κατ' αυτόν τον τρόπο να καθορίσει ποια είναι συμπληρωμένα και ποια όχι. Η τεχνική της Accusoft μπορεί να υποστηρίξει τον σχεδιασμό και την εκτύπωση σε απλό χαρτί, αλλά κατά την εφαρμογή στο σχολείο το ποσοστό επιτυχίας των απαντήσεων πολλαπλής επιλογής δεν μπορεί να επιτύχει τις απαιτήσεις της εξέτασης (accusoft,FormSuite).

3. Υπόβαθρο

3.1. Οπτική αναγνώριση σημαδιών(OMR)

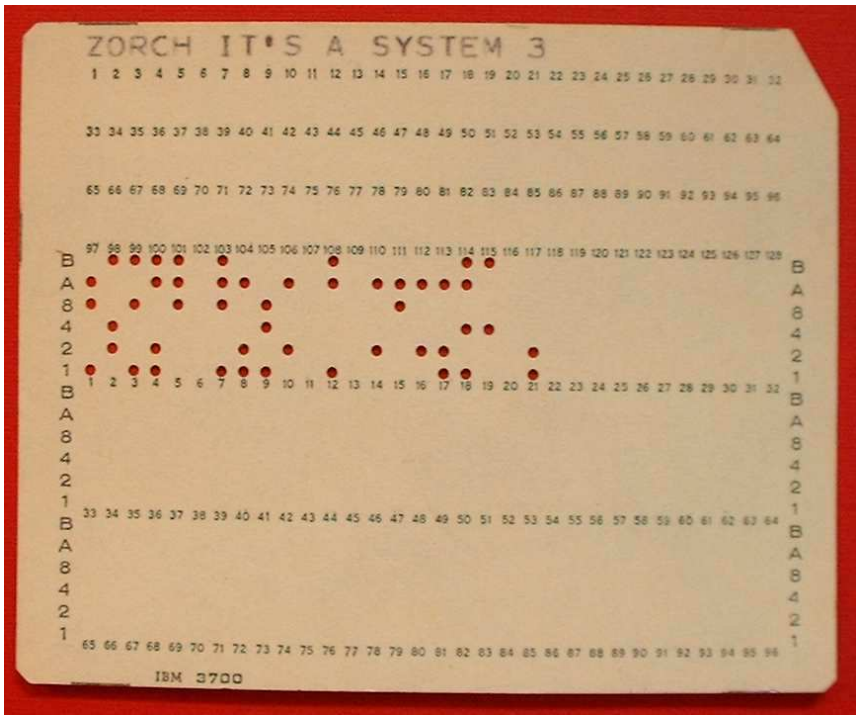
3.1.1 Εισαγωγή

Η οπτική αναγνώριση σημαδιών ή Optical mark recognition (εφεξής OMR) είναι μια αυτοματοποιημένη διαδικασία αναγνώρισης εγγράφων που έχουν συμπληρωθεί από άνθρωπο, όπως ερωτηματολόγια ή διαγωνίσματα πολλαπλής επιλογής .

3.1.2 Ιστορία της οπτικής αναγνώρισης σημαδιών (OMR)

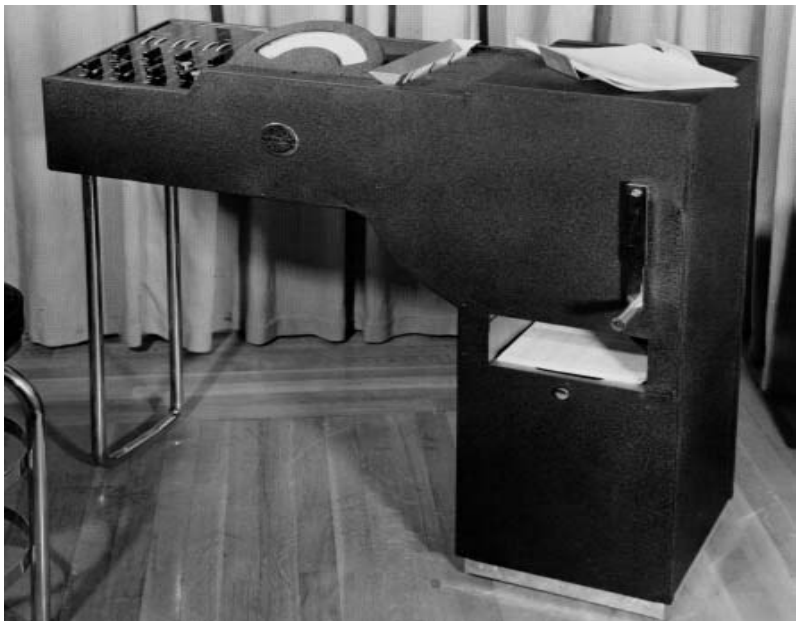
Η οπτική αναγνώριση σημαδιών (OMR) είναι η σάρωση μιας σελίδας χαρτιού και η ανίχνευση της παρουσίας ή της απουσίας σημαδιών σε μια προκαθορισμένη θέση. Στην εξέλιξη της οπτικής αναγνώρισης σημαδιών έχουν συμβάλει διάφορες τεχνολογίες. Στις αρχές του 19^{ου} και 20^{ού} αιώνα τα διπλώματα ευρεσιτεχνίας δόθηκαν για μηχανές που βοηθούσαν τους τυφλούς.

Η OMR σήμερα ευρύτατα χρησιμοποιείται ως μια συσκευή για την εισαγωγή δεδομένων. Δυο παλαιότεροι τύποι φόρμας ήταν η ταινία χαρτιού και οι διάτρητες κάρτες (Εικόνα 1) που χρησιμοποιούσαν τρύπες αντί μολύβι. Οι ταινίες χαρτιού χρησιμοποιήθηκαν στις αρχές του 1857 ως συσκευή εισόδου για τον τηλέγραφο. Οι διάτρητες κάρτες δημιουργηθήκαν το 1890 και χρησιμοποιήθηκαν ως συσκευή εισόδου για τους πρώτους ηλεκτρονικούς υπολογιστές. Η χρήση των διάτρητων καρτών μειώθηκε αισθητά στις αρχές του 1970 με την εισαγωγή των προσωπικών ηλεκτρονικών υπολογιστών. Που στις μέρες μας με την σύγχρονη OMR όπου η αναγνώριση του σημαδιού γίνεται από ένα οπτικό σαρωτή.



Εικόνα 1 IBM 96 στηλών διάτρητη κάρτα

Ο πρώτος σαρωτής αναγνώρισης σημαδιών ήταν ο IBM 805 Test Scoring Machine **(**Εικόνα 2), ο οποίος αναγνώριζε τα σημάδια μέσω της ηλεκτρικής αγωγιμότητας του γραφίτη του μολυβιού χρησιμοποιώντας ζεύγη από συρμάτινες βούρτσες που σάρωναν την επιφάνεια του χαρτιού. Κατά την δεκαετία του 1930, ο Richard Warren της IBM πειραματίστηκε με συστήματα οπτικής αναγνώρισης για τη βαθμολόγηση διαγωνισμάτων, με τις υπ' αριθμούς 2.150.256



Εικόνα 2 IBM 805 test scoring machine

και 2.010.653 Όπως κατοχυρώθηκε στα διπλώματα ευρεσιτεχνίας των Η.Π.Α. 2.150.256 (που κατατέθηκε το 1932, χορηγήθηκε το 1939) και 2.010.653 (που κατατέθηκε το 1933, χορηγήθηκε το 1935). Το πρώτο επιτυχημένο σύστημα οπτικής αναγνώρισης κατασκευάστηκε από τον Everett Franklin Lindquist και κατοχυρώθηκε στα διπλώματα ευρεσιτεχνίας των ΗΠΑ 3.050.248 (που κατατέθηκε το 1955, χορηγήθηκε το 1962).

Ο Lindquist ανέπτυξε πολυάριθμα πρότυπα εκπαιδευτικά διαγωνίσματα και χρειάστηκε μια καλύτερη μηχανή βαθμολόγησης από το τότε πρότυπο της IBM 805. Τα δικαιώματα της ευρεσιτεχνίας του Lindquist κατέχοντο από το Measurement Research Center μέχρι το 1968, όταν το πανεπιστήμιο της Αϊόβα πώλησε τα δικαιώματα στην Westinghouse Corporation.

Την ίδια χρονική περίοδο, η IBM ανέπτυξε με επιτυχία μια μηχανή οπτικής αναγνώρισης, που κατοχυρώθηκε ως ευρεσιτεχνία με αριθμό καταχώρισης 2.944.734 στις Η.Π.Α. (κατατέθηκε το 1957, χορηγήθηκε το 1960), η οποία κυκλοφόρησε με την επωνυμία IBM 1230 Optical mark scoring reader το 1962.

Αυτή και μια ποικιλία από σχετικές μηχανές επέτρεψε στην IBM να επεκταθεί σε μια ποικιλία εφαρμογών που έχουν αναπτυχτεί για τις μηχανές οπτικής αναγνώρισης σημαδιών μέχρι την νέα οπτική τεχνολογία. Περιλαμβάνουν μια ποικιλία από εφαρμογές, όπως η διαχείριση αποθεμάτων και οι φόρμες αναφοράς προβλημάτων, οι περισσότερες από τις όποιες έχουν τις διαστάσεις μιας τυπικής διάτρητης κάρτας.

Ενώ οι άλλες εταιρείες στόχευαν στην πώληση υπηρεσιών σάρωσης, η Scantron Corporation που ιδρύθηκε το 1972, υιοθέτησε ένα διαφορετικό μοντέλο προμηθεύοντας με φτηνούς σαρωτές τα σχολεία με σκοπό την αύξηση της κερδοφορίας από την πώληση φορμών διαγωνισμάτων. Η πρακτική αυτή είχε ως αποτέλεσμα, πολλοί άνθρωποι να θεωρούν ότι όλες οι φόρμες διαγωνισμάτων ήταν φόρμες Scantron. Η Scantron λειτουργεί ως θυγατρική της M&F Worldwide(MFW) και παρέχει συστήματα διεξαγωγής και διόρθωσης διαγωνισμάτων, καθώς και συστήματα συλλογής δεδομένων και υπηρεσίες ανάλυσης σε εκπαιδευτικά ινστιτούτα, επιχειρήσεις και κυβερνητικούς οργανισμούς.

Το 1983, η Westinghouse Learning Corporation εξαγοράστηκε από την National Computer Systems (NCS). Το 2000, η NCS εξαγοράστηκε από την Pearson Education, όπου η τεχνολογία OMR αποτέλεσε τον πυρήνα της Pearson's Data Management group. Τον Φεβρουάριο του 2008, M&F Worldwide εξαγόρασε την Data Management group από την Pearson, ο όμιλος τώρα είναι μέρος της Scantron.

Η χρήση της OMR σε συστήματα απογραφής αναπτύχθηκε ως μεταβατική τεχνολογία μεταξύ των διάτρητων καρτών και της τεχνολογίας των barcode. Η OMR χρησιμοποιείται σήμερα εκτενώς για ερευνητικούς και εκπαιδευτικούς σκοπούς.

3.1.3 Background οπτικής αναγνώρισης σημαδιών (OMR)

Πολλές παραδοσιακές συσκευές OMR λειτουργούν με τη βοήθεια ενός ειδικού σαρωτή (Εικόνα 3), όπου μια δέσμη φωτός πέφτει πάνω στο έγγραφο που θέλουμε να αναγνωρίσουμε.

Εικόνα 3 OMR scanner

Η ανακλαστικότητα που προκαλείται λόγω της αντίθεσης (contrasting reflectivity) σε προκαθορισμένες θέσεις του εγγράφου, χρησιμοποιείται για την αναγνώριση των σημαδεμένων περιοχών, επειδή ανακλούν λιγότερο φώς από τις κενές περιοχές του εγγράφου.

Μερικές OMR συσκευές χρησιμοποιούν φόρμες εγγράφων που είναι προτυπωμένες σε ειδικό χαρτί, το οποίο επιτρέπει την διέλευση του φωτός και μετρούν την ποσότητα του φωτός που περνά μέσα από το χαρτί, έτσι τα σημάδια που βρίσκονται και στις δύο πλευρές του εγγράφου μειώνουν την ποσότητα του φωτός που διαπερνά το έγγραφο.

Σε αντίθεση με την ειδική συσκευή OMR, το λογισμικό OMR επιτρέπει στον χρήστη να δημιουργήσει τα δικά του έγγραφα, προσαρμοσμένα σε ένα επεξεργαστή κειμένου και να τα τυπώσει σε έναν κοινό εκτυπωτή. Το OMR λογισμικό λειτουργεί με έναν κοινό επιτραπέζιο σαρωτή εικόνων με τροφοδότη εγγράφων για την επεξεργασία των συμπληρωμένων φορμών.

Η OMR γενικά διακρίνεται από την οπτική αναγνώριση χαρακτήρων (OCR), καθώς και από το γεγονός ότι δεν απαιτούνται πολύπλοκες τεχνικές αναγνώρισης προτύπων. Δηλαδή, τα σημάδια αποτυπώνονται με τέτοιον τρόπο, ώστε να υπάρχει μικρή πιθανότητα να μην διαβαστεί σωστά ένα σημάδι. Στην προκειμένη περίπτωση, απαιτείται εικόνα υψηλής αντίθεσης και εύκολα αναγνωρίσιμο σχήμα. Ένα σχετικό πεδίο με OMR και OCR, είναι η αναγνώριση των barcodes, όπως τα UPC barcode, που βρίσκονται στις συσκευασίες των προϊόντων.

Μια από της πιο συνήθεις εφαρμογές της OMR, είναι αυτή που σχετίζεται με τη χρήση μολυβιού σκληρότητας HB για τη συμπλήρωση διαγωνισμάτων. Οι μαθητές σημειώνουν τις απαντήσεις ή άλλες προσωπικές πληροφορίες, μουτζουρώνοντας κύκλους σε ένα προτυπωμένο φύλλο χαρτί. Στη συνέχεια το φύλλο βαθμολογείται αυτόματα από ένα μηχάνημα σάρωσης.

Στις Ηνωμένες Πολιτείες όπως και στις περισσότερες ευρωπαϊκές χώρες, οι τύποι που χρησιμοποιούνται συνηθέστερα για OMR έγγραφα είναι οριζόντια ή κάθετα ‘τικ’ ή ορθογώνιοι ‘ρόμβοι’, ενώ η πιο οικεία εφαρμογή είναι στα συστήματα λοταρίας. Το πλεονέκτημα των ορθογώνιων ρόμβων είναι ότι σημειώνονται και διαγράφονται εύκολα. Οι μεγάλοι κύκλοι για την καταχώριση των απαντήσεων είναι μια κληρονομιά από τα παλιά OMR συστήματα που δεν ήταν τόσο ευαίσθητα και γι' αυτό ήταν αναπόφευκτη η χρήση μεγάλων κύκλων.

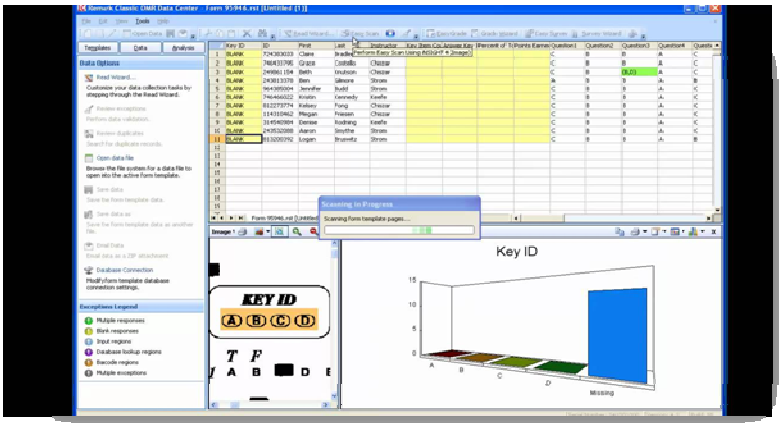
Στις μέρες μας, για την υλοποίηση πολλών OMR εφαρμογών, χρησιμοποιείται ανθρώπινο δυναμικό που συμπληρώνει εξειδικευμένες φόρμες. Αυτές οι φόρμες έχουν βελτιστοποιηθεί για τη σάρωση μέσω υπολογιστή, με προσεκτική εγγραφή στην εκτύπωση και προσεκτικό σχεδιασμό φόρμας , έτσι ώστε η ασάφεια να μειώνεται στο ελάχιστο. Το εξαιρετικά χαμηλό ποσοστό σφάλματος, το χαμηλό κόστος και ευκολία στη χρήση, κάνει την OMR μια πολύ δημοφιλή μέθοδο για την καταμέτρηση ψήφων.

Σημάδια OMR προστίθενται και στην αλληλογραφία ώστε να μπορέσει να χρησιμοποιηθεί από ειδικές συσκευές που εισάγουν την αλληλογραφία σε φάκελο. Τα σημάδια προστίθενται σε κάθε σελίδα της αλληλογραφίας και αποτελούνται από μια ακολουθία από μαύρες παύλες, που καθορίζουν το σημείο όπου θα διπλωθεί η αλληλογραφία για την εισαγωγή της στον φάκελο.

3.1.4 Λογισμικό οπτικής αναγνώρισης σημαδιών (OMR)

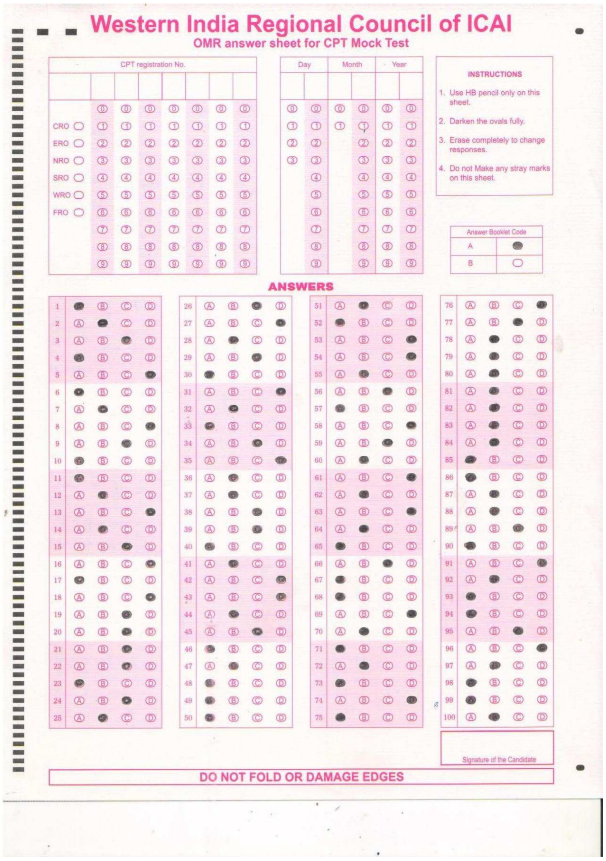
Το λογισμικό OMR είναι μια εφαρμογή, που κάνει εφικτή τη χρήση OMR σε επιτραπέζιο υπολογιστή, με τη χρήση ενός σαρωτή εικόνων για την επεξεργασία ερωτηματολόγιων, διαγωνισμάτων πολλαπλής επιλογής, απουσιολογίων, λιστών έλεγχου και άλλων απλών φορμών, που μπορούν να εκτυπωθούν από έναν laser εκτυπωτή.

Το λογισμικό OMR χρησιμοποιείται για να συλλέξει τα δεδομένα από OMR φύλλα. Όσο τα δεδομένα συλλέγονται ο σαρωτής εστιάζει σε πολλούς παράγοντες, όπως το πάχος του χαρτιού, οι διαστάσεις του OMR φύλλου και το πρότυπο σχέδιο. Ένα από τα πρώτα πακέτα λογισμικού που χρησιμοποιεί εικόνες από έναν κοινό σαρωτή εικόνας είναι το Remark Office OMR (Εικόνα 4), το οποίο κατασκευάζεται από την Gravic, Inc. (Originally named Principia Products, Inc.), Remark Office OMR 1.0 που κυκλοφόρησε το 1991.



Εικόνα 4 Remark Office OMR

Η ανάγκη για OMR λογισμικό ανέκυψε επειδή τα συστήματα OMR σχεδιαστήκαν για τη χρήση από ειδικούς σαρωτές και ειδικές προτυπωμένες φόρμες (Εικόνα 5), με ειδικά σημάδια ελέγχου. Τέτοιες φόρμες κοστίζουν περίπου 0.07€ με 0.13€ ευρώ ανά σελίδα. Αντιθέτως, οι χρήστες μπορούν να δημιουργήσουν τις δίκες τους φόρμες χρησιμοποιώντας OMR λογισμικό, με έναν επεξεργαστή κειμένου ή με αυτοματοποιημένο τρόπο μέσω ειδικού λογισμικού και να τις τυπώσουν τοπικά σε έναν κοινό εκτυπωτή, εξοικονομώντας πολλά χρήματα.



Εικόνα 5 OMR sheet

Η αναγνώριση οπτικών σημαδιών σε μια φόρμα, όπως οι φόρμες απογράφης, παρέχεται ήδη από τα τέλη της δεκαετίας του 1980 από πολλές εταιρείες. Τα περισσότερα συστήματα βασίζονται σε δυαδικές εικόνες, όπου γίνεται απαρίθμηση των πίξελ και υπάρχουν όρια ελαχίστης και μεγίστης τιμής. Τα πίξελ καταμετρώνται και εξαλείφονται τα ανεπιθύμητα σημάδια, όπως αυτά που αφήνει μια λερωμένη σβήστρα, που όμως όταν θα μετατρέψουμε την εικόνα σε ασπρόμαυρη, θα αναγνωριστούν ως επιθυμητά σημάδια. Έτσι, αυτή η μέθοδος μπορεί να δημιουργήσει προβλήματα, αν ο χρηστής αλλάξει γνώμη. Για το λόγο αυτό, πολλά προϊόντα λογισμικού άρχισαν να χρησιμοποιούν την κλίμακα του γκρι, ώστε να αναγνωρίζεται καλύτερα η πρόθεση του χρήστη.

Το OMR λογισμικό χρησιμοποιείται επίσης για την προσθήκη OMR σημαδιών στην παραδοσιακή αλληλογραφία, ώστε να μπορούν αυτά να σαρωθούν από μηχανές αυτόματης εισαγωγής σε φάκελο. Για παράδειγμα, το OMR λογισμικό Mail Markup, από την αγγλική Funasset Limited. Αυτό το λογισμικό επιτρέπει στον χρήστη να επιλέξει και να ρυθμίσει μια OMR ακολουθία, ώστε να προσθέσει OMR σημάδια στην αλληλογραφία πριν την εκτύπωση.

3.1.5 Εφαρμογές

- Στην ακαδημαϊκή ερευνά
- Σε κοινοτικές έρευνες
- Σε καταναλωτικές έρευνες
- Σε διαγωνίσματα
- Στη συλλογή δεδομένων
- Στην αξιολόγηση προϊόντων
- Σε φύλλα παρουσίας
- Στην απογραφή
- Σε έντυπα εγγραφής νέων μελών

3.1.6 Δυνατότητες/απαιτήσεις

Στο παρελθόν αλλά και στο παρόν, μερικά συστήματα OMR απαιτούν ειδικό χαρτί, ειδικό μελάνι και ειδικό αναγνώστη εισόδου (Bergeron & Bryan, 1998). Αυτό περιορίζει τους τύπους των ερωτήσεων που μπορούν να δημιουργηθούν και δεν επιτρέπει τη μεταβλητότητα των φορμών.

Η πρόοδος στην OMR επιτρέπει πλέον στους χρήστες να δημιουργούν και να εκτυπώνουν τις δικές τους φόρμες με τη χρήση ενός σαρωτή (κατά προτίμηση με τροφοδότη εγγράφων) για να διαβάζονται οι πληροφορίες (Bergeron & Bryan, 1998). Ο χρήστης έχει τη δυνατότητα να οργανώσει τις ερωτήσεις σε μορφή που να καλύπτει τις ανάγκες του, ενώ είναι σε θέση να εισάγει με ευκολία τα δεδομένα (LoPresti, Zvia Segal, & Zvia Segal, 1996).

Συστήματα OMR πλησιάζουν ακρίβεια 100% και χρειάζονται μόνο 0,005 δευτερόλεπτα κατά μέσον όρο για να αναγνωριστεί η φόρμα (Bergeron & Bryan, 1998). Οι χρήστες μπορούν να χρησιμοποιούν τετράγωνα, κύκλους, ελλείψεις και εξάγωνα για να σημειώνουν τις απαντήσεις. Το λογισμικό μπορεί στη συνέχεια να ρυθμιστεί, ώστε να αναγνωρίζει κύκλους(bubbles), x ή σημάδια ελέγχου.

Η OMR επίσης διατίθεται και για προσωπική χρήση. Υπάρχουν all-in-one εκτυπωτές στην αγορά, που επιτρέπουν στον χρήστη να εκτυπώσει τις φόρμες που θέλει. Μόλις η φόρμα συμπληρωθεί, ο χρήστης τοποθετεί το φύλλο στον σαρωτή για να σαρωθεί και να πάρει τα αποτελέσματα.

3.1.7 Μειονεκτήματα

Υπάρχουν, όμως, ορισμένα μειονεκτήματα και περιορισμοί στη χρήση της OMR. Αν ο χρήστης θέλει να συγκεντρώσει μεγάλες ποσότητες κειμένου, τότε η OMR δυσχεραίνει τη συλλογή των δεδομένων (Green & Phil, 2000). Υπάρχει, επίσης, η πιθανότητα ελλειπόντων στοιχείων, κατά τη διαδικασία σάρωσης, όπως και η πιθανότητα για εσφαλμένη αρίθμηση σελίδων, που μπορεί να οδηγήσει σε λάθος σειρά σάρωσης. Η έλλειψη δικλείδων ασφαλείας, θα μπορούσε να κάνει μια σελίδα να σαρωθεί ξανά, ενώ θα ήταν πιθανό να παραχθούν διπλότυπα δεδομένα, αλλά και να στρεβλωθούν αποτελέσματα (Bergeron & Bryan, 1998). Το αποτέλεσμα της ευρείας υιοθέτησης και χρήσης της OMR, είναι οι τυποποιημένες εξετάσεις να αποτελούνται κατά κύριο λόγο από ερωτήσεις πολλαπλής επιλογής, επειδή όμως αυτό αλλάζει τη φύση της, είναι υπό δοκιμή.

3.1.8 OMR Ανοιχτού λογισμικού

Εδώ παρουσιάζονται μερικά προϊόντα OMR λογισμικού που αναπτύχθηκαν και διανέμονται με την άδεια ανοιχτού λογισμικού.

- queXF which, το οποίο μπορεί να χρησιμοποιηθεί μόνο του ή σε συνδυασμό με το LimeSurvey.
- Udai OMR
- Shared Questionnaire System (SQS)
- Auto Multiple Choice for class tests, με μορφοποίηση LaTeX.[13]
- Moodle
- TCExam
- SDAPS για έρευνες, υποστηρίζει το LaTeX και ODT για μορφοποίηση εγγράφων.
- Smartshoot OMR, υποστηρίζει κάθε omr form, με opensource OCR engine.

3.2 QR code

Τα QR Codes (Εικόνα 6) είναι σύγχρονοι γραμμωτοί κώδικες δύο διαστάσεων (2D), ταχείας αποκωδικοποίησης (γνωστοί και ως matrix code) και αποτελούν μια σύγχρονη μετεξέλιξη των γνωστών σε όλους μας barcodes, μιας διάστασης, τα οποία εφηύρε το 1952, κατοχυρώνοντας την ευρεσιτεχνία, ο Αμερικάνος από το Atlantic City του New Jersey, Norman Joseph Woodland, μαζί με τον συνάδελφο του Bernard Silver και τα οποία χρησιμοποιούνται μέχρι και σήμερα στην τυποποίηση κυρίως των προϊόντων και πλέον τα συναντάμε παντού, συνήθως επάνω στις συσκευασίες προϊόντων. Κάθε μέρα σαρώνονται πάνω από 5 δισεκατομμύρια προϊόντα με τη βοήθεια των γραμμωτών κωδικών (Wikipedia QR code).

Εικόνα 6 Διάφορα QRcodes

Τα μονοδιάστατα barcodes δημιουργήθηκαν από την ανάγκη αποθήκευσης κάποιων πληροφοριών, που ήταν σχετικές με το προϊόν, την προέλευση και την συσκευασία του. Η σάρωση γίνεται μηχανικά με μια στενή δέσμη φωτός από τους σαρωτές barcode . Οι ανάγκες, όμως, για την αποθήκευση όλο και περισσότερων δεδομένων με την μορφή γραμμωτού κώδικα ολοένα και αυξάνονταν, προσπαθώντας να βρεθούν απλές λύσεις αποθήκευσης δεδομένων, αποφεύγοντας τα παραδοσιακά μέσα των μαγνητικών καρτών.

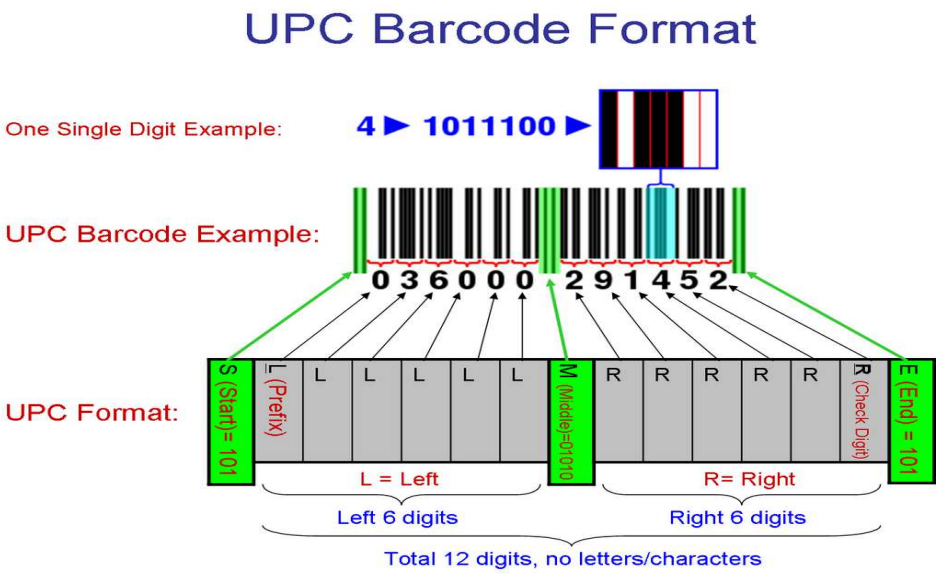
Αυτό είχε σαν αποτέλεσμα να οδηγήσουν στην επινόηση και δημιουργία, από την ιαπωνική εταιρεία Denso-Wave, θυγατρική της Toyota, ενός νέου γραμμωτού κώδικα και πρότυπου ISO, ο οποίος πλέον είναι δισδιάστατος, για να έχει την δυνατότητα αποθήκευσης περισσοτέρων δεδομένων.

Αυτό το νέο είδος barcode, ονομάστηκε QR Code και προέρχεται από τα αρχικά των αγγλικών λέξεων "Quick Response", που σημαίνουν Γρήγορη Ανταπόκριση, διότι οι γιαπωνέζοι δημιουργοί του, είχαν σαν σκοπό την ταχύτατη αποκωδικοποίηση του πλήθους των πληροφοριών, μέσα από το σκανάρισμα αυτών των σχημάτων δύο διαστάσεων, με τη χρήση τεχνολογικού εξοπλισμού σκάνερ ή τη χρήση των κινητών τηλεφώνων, τύπου smartphones και με την εγκατάσταση ανάλογου λογισμικού. Αυτά έχουν την δυνατότητα να ανιχνεύουν την δύο διαστάσεων ψηφιακή εικόνα, από έναν αισθητήρα εικόνας CCD και στη συνέχεια αναλύονται ψηφιακά από τον επεξεργαστή.

3.2.1 Ιστορία

Στα πρώτα στάδια επινόησης και εφαρμογής (από το 1994), εφαρμόστηκαν αρχικά μόνον στη βιομηχανία κατασκευής αυτοκινήτων, για τον εντοπισμό των ανταλλακτικών σε διάφορα στάδια της παραγωγής, διότι η Denso-Wave, ειδική στις εφαρμογές barcode, η οποία και τα επινόησε, δούλευε για λογαριασμό της αυτοκινητοβιομηχανίας Toyota (Furht, 2011).

Σύντομα όμως η χρήση τους ξεπέρασε τις βιομηχανικές εφαρμογές και έγιναν δημοφιλή και σε άλλες εφαρμογές καθημερινής χρήσης, κατακτώντας τον χώρο της διαφήμισης και της τυποποίησης, προσφέροντας τεράστιες δυνατότητες πληροφόρησης στους καταναλωτές, λόγω της ταχύτητας ανάγνωσης του μεγάλου όγκου πληροφοριών σε σύγκριση με τα παλιά παραδοσιακά UPC barcodes (Εικόνα 7).



Εικόνα 7 UPC barcode

Ευρεία διάδοση και χρήση των QR Code έγινε αρχικά κυρίως στην Ιαπωνία και την Αμερική, αλλά τελευταία η χρήση τους διαδόθηκε παντού, με τις στατιστικές να δείχνουν ότι κατά τον μήνα Ιούνιο 2011, 14 εκατομμύρια χρήστες έχουν σκανάρει ένα QR Code ή ένα barcode. Ένα ποσοστό 58% από αυτούς τους χρήστες έχουν σκανάρει ένα QR ή bar code από το σπίτι τους, ενώ το 39% σκάναρε από καταστήματα λιανικής πώλησης. Το 53% από τα 14 εκατομμύρια χρηστών ήταν άνδρες, μεταξύ 18 και 34 ετών.

3.2.2 Πρότυπα

Υπάρχουν πολλά πρότυπα που καλύπτουν την κωδικοποίηση των δεδομένων QR codes:

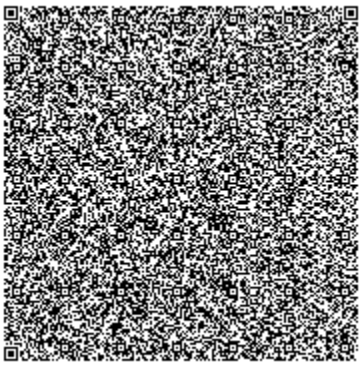
• Οκτώβριος 1997 – AIM (Association for Automatic Identification and Mobility)
• Ιανουάριος 1999 – JIS X 0510
• Ιούνιος 2000 – ISO/IEC 18004:2000)
• 1 Σεπτεμβρίου 2006 – ISO/IEC 18004:2006

3.2.3 Χωρητικότητα

Η ποσότητα των δεδομένων που μπορεί να αποθηκευτεί στο QR code εξαρτάται από τον τύπο δεδομένων (mode ή σύνολο χαρακτήρων εισόδου), έκδοση (1, ..., 40, αναφέροντας τις συνολικές διαστάσεις του συμβόλου) και από το επίπεδο διόρθωσης σφάλματος. Οι μέγιστες χωρητικότητες αποθήκευσης αφορούν τα σύμβολα 40-L (έκδοση 40, επίπεδο διόρθωσης σφάλματος L- Εικόνα 8):

Λειτουργία εισόδου	Μεγ. Αρ. χαρακτήρων	bits/char	Πιθανοί χαρακτήρες
Αριθμιτικα μονο	7,089	3⅓	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Αλφαριθμιτικα	4,296	5½	0–9, A–Z (upper-case only), space, $, %, *, +, -, ., /,
Binary/byte	2,953	8	ISO 8859-1
Kanji/kana	1,817	13	Shift JIS X 0208

Πίνακας 1 Ποσότητα των δεδομένων που μπορεί να αποθηκευτεί στο QR code



Εικόνα 8 QRcode ver 40

3.2.3 Χρήση

Τα QR Codes βοηθούν στην ενημέρωση των καταναλωτών και αποτελούν την ιδανική γέφυρα διασύνδεσης μεταξύ των offline και online media. Μπορούν να προστεθούν σε οποιαδήποτε έντυπη διαφήμιση, φυλλάδια, αφίσες, προσκλητήρια, τηλεοπτικές διαφημίσεις, πινακίδες, μπλουζάκια t-shirts, κούπες, οχήματα κλπ και μπορούν να περιέχουν:

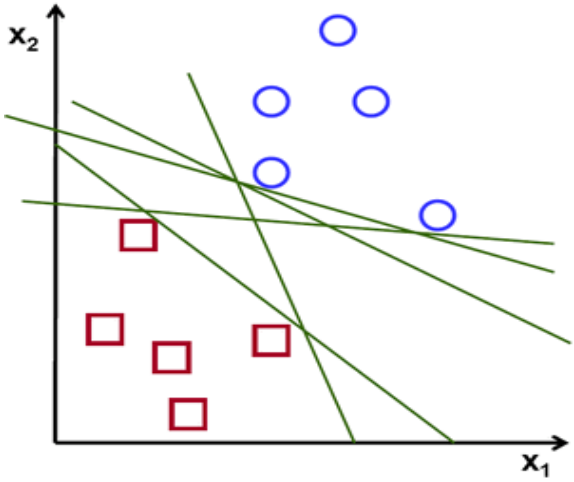
- Λεπτομέρειες για το προϊόν
- Στοιχεία επικοινωνίας (vCard)
- Λεπτομέρειες προσφοράς
- Λεπτομέρειες για ένα γεγονός
- Στοιχεία ανταγωνισμού
- Εκπτωτικό κουπόνι
- Ταυτότητες Twitter, Facebook, MySpace
- Ένα σύνδεσμο για το YouTube
- Επαγγελματικές κάρτες
- Μηνύματα ηλεκτρονικού ταχυδρομείου
- Μηνύματα SMS
- Χάρτες
- Κείμενο ή σύνδεσμο (Url)

3.3 Suport vectors machines

3.3.1 Εισαγωγή

Οι μηχανές διανυσμάτων υποστήριξης (Support Vector Machines, SVM), αποτελούν έναν αλγόριθμο ταξινόμησης προτύπων, ο οποίος βασίζεται στη στατιστική Θεωρία μάθησης και αναπτύχθηκε απο τον Vapnik. Για την ταξινόμηση, οι μηχανές διανυσμάτων υποστήριξης αναζητούν ένα υπερεπίπεδο (hyperplane) στον χώρο των πιθανών εισόδων. Αυτό το υπερεπίπεδο δημιουργείται ώστε να διαχωρίσει τα θετικά παραδείγματα από τα αρνητικά, επιδιώκοντας την μέγιστη απόσταση από το κοντινότερο θετικό και αρνητικό παράδειγμα, όπως παρουσιάζεται στην (Εικόνα 9).

Έτσι, μπορεί κανείς να διαπιστώσει ότι υπάρχει μεγάλο πλήθος υπερεπιπέδων που μπορούν να διαχωρίσουν τα θετικά παραδείγματα από τα αρνητικά. Αυτό, βέβαια, αντιμετωπίζεται με το γεγονός ότι επιλέγεται τελικά εκείνο το υπερεπίπεδο σ_t που έχει το μέγιστο περιθώριο μεταξύ του κοντινότερού θετικού και αρνητικού παραδείγματος (Εικόνα 9) (Corinna & Vapnik, 1995).



Εικόνα 9 Διαχωριστικά υπερεπίπεδα για το SVM

3.3.2 Γραμμικές μηχανές διανυσμάτων υποστήριξης (SVM)

Για δεδομένα εκπαίδευσης $\mathcal{D}$,(Εικόνα 10) ένα σύνολο από *n* σημεία της μορφής

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\}\}_{i=1}^{n}$$

όπου y_i είναι είτε 1 ή −1, που υποδεικνύει την κατηγορία στην οποία ανήκει το σημείο $\mathbf{x}_i$. Κάθε $\mathbf{x}_i$ είναι ένα *p*-διάστατο, πραγματικό διάνυσμα. Θέλουμε να βρούμε το μέγιστο περιθώριο του υπερεπίπεδου που χωρίζει τα σημεία που έχουν $y_i = 1$ από εκείνα που έχουν $y_i = -1$. Κάθε υπερεπίπεδο μπορεί να γραφτεί ως το σύνολο των σημείων $\mathbf{x}$ που ικανοποιεί την

$$\mathbf{w} \cdot \mathbf{x} - b = 0,$$

όπου . υποδηλώνει το εσωτερικό γινόμενο και το $\mathbf{w}$ (δεν είναι απαραίτητο να είναι κανονικοποιημένο) κανονικό διάνυσμα στο υπερεπίπεδο. Η παράμετρος $\frac{b}{\|\mathbf{w}\|}$ καθορίζει τη μετατόπιση του υπερεπίπεδου από την αρχή κατά μήκος του κανονικού διανύσματος $\mathbf{w}$.

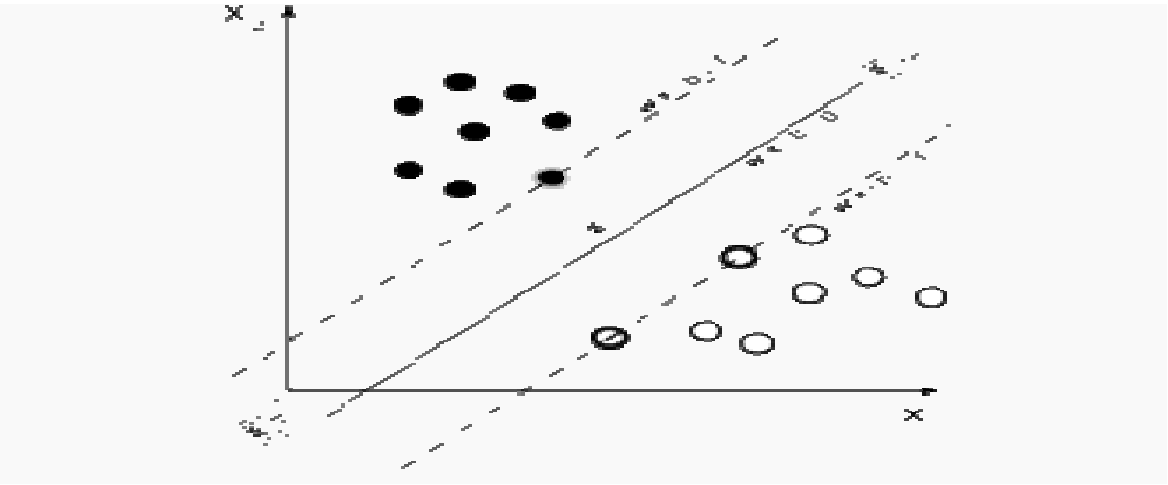
Εάν τα δεδομένα εκπαίδευσης είναι γραμμικά διαχωρίσιμα, μπορούμε να επιλέξουμε δύο υπερεπίπεδα με τέτοιο τρόπο, ώστε ο διαχωρισμός των δεδομένων να γίνεται χωρίς να υπάρχουν σημεία μεταξύ τους και στη συνέχεια να προσπαθούν να μεγιστοποιήσουν την απόστασή τους. Η περιοχή που περικλείεται από αυτά ονομάζεται 'περιθώριο'. Αυτά τα υπερεπίπεδα μπορεί να περιγραφούν από τις εξισώσεις

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$

και

$$\mathbf{w} \cdot \mathbf{x} - b = -1.$$

Με τη χρήση γεωμετρίας, βρίσκουμε την απόσταση μεταξύ αυτών των δύο υπερεπιπέδων $\frac{2}{\|\mathbf{w}\|}$, έτσι θέλουμε να ελαχιστοποιήσουμε $\|\mathbf{w}\|$.για να αποτρέψουμε τα σημεία να πέσουν στο περιθώριο, προσθέτουμε, λοιπόν, τον ακόλουθο περιορισμό: για κάθε i είτε



Εικόνα 10 Μέγιστο περιθώριο υπερεπίπεδου και περιθώρια για ένα εκπαιδευμένο SVM με δείγματα από δύο κατηγορίες. Τα δείγματα στο περιθώριο ονομάζονται διανύσματα υποστήριξης.

$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1$ for $\mathbf{x}_i$ για την πρώτη κλάση

ή

$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1$ for $\mathbf{x}_i$ για την δεύτερη κλάση.

Αυτό μπορεί να γραφεί ως:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n. \qquad (1)$$

Μπορούμε να τα βάλουμε όλα αυτά μαζί για να πάρει το πρόβλημα βελτιστοποίησης:

Minimize (in $\mathbf{w}, b$)

$$\|\mathbf{w}\|$$

υπόκεινται σε (για κάθε $i = 1, \ldots, n$)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1.$$ (Corinna & Vapnik, 1995)

4. Εργαλεία

4.1 OepenCv

4.1. 1 Εισαγωγή στην OpenCv

Η OpenCv είναι μια βιβλιοθήκη μηχανικής όρασης (computer vision), ανοικτού λογισμικού που αναπτύχθηκε αρχικά από την Intel και είναι ελεύθερη για εμπορική και ερευνητική χρήση υπό την άδεια ανοικτού λογισμικού BSD. Αυτό σημαίνει ότι μπορεί να γίνει ελεύθερη χρήση ολόκληρης ή μέρους της βιβλιοθήκης για την ανάπτυξη ενός εμπορικού η ερευνητικού προϊόντος από τον οποιοδήποτε.

Η βιβλιοθήκη είναι ανεξάρτητη πλατφόρμας και είναι γραμμένη σε optimized C. Αποτελείται από περισσότερες από 500 συναρτήσεις σε C και διάφορες κλάσεις σε C++. Έχει σχεδιαστεί με στόχο την αποτελεσματική εκμετάλλευση των υπολογιστικών πόρων και με έμφαση στις εφαρμογές πραγματικού χρόνου. Η Intel έχει, επίσης, αναπτύξει ένα σύνολο ρουτινών χαμηλού προγραμματιστικού επιπέδου, το Integraded Performance Primitives(IPP), το οποίο χρησιμοποιείται σε πολλές αλγοριθμικές περιοχές και αν είναι εγκατεστημένο, η OpenCv θα το χρησιμοποιήσει για να επιταχύνει τα προγράμματα που τη χρησιμοποιούν. Πρέπει να σημειωθεί ότι αυτό το σύνολο ρουτινών δεν είναι απαραίτητο για τη χρήση της OpenCv.

4.1.2 Διαθέσιμες εκδόσεις

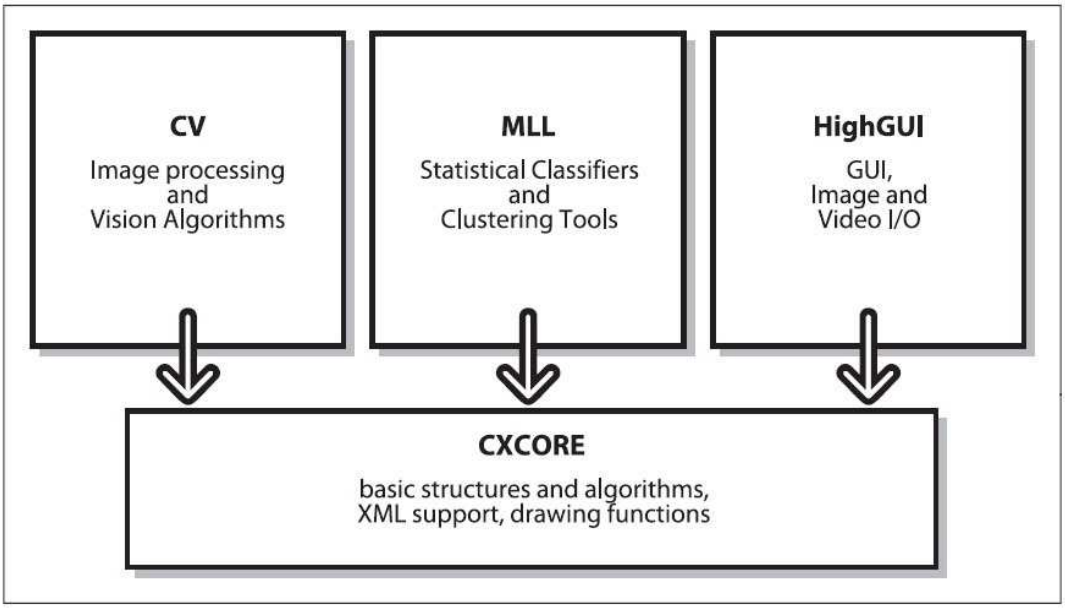
Η OpenCv είναι διαθέσιμη σε Linux, Mac OS και Windows. Η τελευταία της έκδοση, 2.4.9, μπορεί να βρεθεί στο Sourceforge. Η πρώτη έκδοση ανακοινώθηκε το 2000 στο συνέδριο Computer Vision and Pattern Recognition και από τότε ενημερώνεται συνεχώς.

4.1.3 Τα δομικά στοιχεία της OpenCv

Η OpenCv περιέχει πάνω από 500 συναρτήσεις, κατάλληλες για πολλές διαφορετικές εφαρμογές στους τομείς της οπτικής επεξεργασίας, όπως για ιατρικές εφαρμογές που αφορούν εικόνα και βίντεο, κλειστό κύκλωμα παρακολούθησης, ασφάλεια, ρομποτική, ρύθμιση κάμερας, διεπαφή χρήστη και υπολογιστή.

Επειδή υπάρχει ισχυρή συσχέτιση μεταξύ μηχανικής όρασης και μηχανικής εκμάθησης, η OpenCv περιέχει και μια βιβλιοθήκη μηχανικής εκμάθησης γενικού σκοπού (MLL - Machine Learning Library). Αυτή η ειδική βιβλιοθήκη εστιάζει στην αναγνώριση προτύπων με στατιστικές μεθόδους και στην ομαδοποίηση. Χρησιμοποιείται σε οποιοδήποτε πρόβλημα μηχανικής όρασης, λόγω της γενικής μορφής της.

Η OpenCv δομείται κυρίως από 5 συστατικά μέρη, 4 από τα οποία παρουσιάζονται στην (Εικόνα 11). Το πρώτο στο σχήμα με την ονομασία CV, αναπαριστά τους βασικούς αλγορίθμους για επεξεργασία εικόνας, αλλά και τους υψηλού επιπέδου για τη μηχανική όραση. Το δεύτερο, ML, αναπαριστά τη βιβλιοθήκη μηχανικής μάθησης, που περιλαμβάνει εργαλεία για ομαδοποίηση και στατιστική ταξινόμηση. Το τρίτο, HighGUI, αναπαριστά τις ρουτίνες εισόδου-εξόδου, τις συναρτήσεις για την προβολή και την αποθήκευση των βίντεο και γενικά ό,τι έχει να κάνει με τη διεπαφή χρήστη-εφαρμογής. Τέλος, το CxCore περιέχει τις βασικές δομές δεδομένων, όπως το IplImage, τη βασική δομή στην OpenCv που περιγράφει με λεπτομέρεια το κάθε καρέ προς επεξεργασία. Το τελευταίο δομικό μέρος της OpenCv είναι το CvAux, που περιέχει αλγορίθμους που δεν χρησιμοποιούνται πολύ, γιατί είναι σε πειραματικό στάδιο ή που έχουν απλά εγκαταλειφθεί.



Εικόνα 11 Τα συστατικά μέρη της OpenCv

4.1.4 Τα πλεονεκτήματα της OpenCv

Το βασικότερο πλεονέκτημα της OpenCv είναι η ταχύτητα. Αυτός είναι και ο βασικός λόγος που δημιουργήθηκε εξ αρχής, καθώς η Intel ήθελε να δείξει πόσο προχωρημένοι ήταν οι επεξεργαστές της, που μπορούσαν να επεξεργάζονται βίντεο σε πραγματικό χρόνο.

Το δεύτερο είναι ότι ενώ είναι ελεύθερο λογισμικό, υποστηρίζεται από εταιρίες όπως η Intel και η IBM, πανεπιστήμια όπως το Stanford, το MIT και πλήθος άλλων οργανισμών και εργαστηριών που βοηθούν συνεχώς στην ανάπτυξή της.

Ένα άλλο πλεονέκτημα είναι ότι δεν απαιτεί περίπλοκα μηχανήματα και εξοπλισμούς, αφού και η πιο απλή webcam, είναι αρκετή για είσοδο εικόνας και ο πιο απλός προσωπικός υπολογιστής, είναι ικανός για την επεξεργασία της.

4.1.5 Εφαρμογές της OpenCv

Οι επιστήμονες και οι ακαδημαϊκοί, που ασχολούνται με τη μηχανική όραση, γνωρίζουν πραγματικά τις δυνατότητες της και το ευρύ φάσμα στο οποίο χρησιμοποιείται. Οι περισσότεροι γνωρίζουν κάποια από τα πλεονεκτήματα της μηχανικής όρασης και λίγες εφαρμογές της. Οι πιο γνωστές εφαρμογές της είναι στα συστήματα παρακολούθησης, σε εικόνες και βίντεο στο διαδίκτυο.

Η πραγματική δύναμή της φαίνεται σε περιοχές που δεν είναι ευρέως γνωστό ότι χρησιμοποιείται, όπως σε διεπαφές παιχνιδιών και σε άλλες εντυπωσιακές, όπως στους περισσότερους εναέριους χάρτες και χάρτες δρόμων (όπως στο Google Street View), όπου χρησιμοποιούν κατά κόρον τεχνικές συγκόλλησης εικόνων, ευθυγράμμισης και διόρθωσης της εικόνας από παραμορφώσεις που εισάγει ο φακός. Άλλες χρήσεις της είναι σε μη-επανδρωμένα οχήματα (γενικά στη ρομποτική) ή βιοϊατρικές αναλύσεις.

Ένας σημαντικός κλάδος στον οποίο κερδίζει έδαφος, είναι η βιομηχανία. Σχεδόν όλα τα προϊόντα που παράγονται μαζικά, έχουν επιθεωρηθεί σε κάποιο στάδιο της παραγωγής από συστήματα μηχανικής όρασης. Από την πρώτη της έκδοση, τον Ιανουάριο του 1999, η OpenCv έχει χρησιμοποιηθεί σε πολλές εφαρμογές, προϊόντα και ερευνητικές μελέτες.

Αυτές οι εφαρμογές περιλαμβάνουν τον συγκερασμό εικόνων σε δορυφορικούς και διαδικτυακούς χάρτες, τη μείωση του θορύβου σε ιατρικές εικόνες, τα αυτόματα συστήματα ελέγχου και ασφαλείας, στρατιωτικές εφαρμογές και μη-επανδρωμένα οχήματα (εναέρια, υποβρύχια, επίγεια). Έχει χρησιμοποιηθεί, ακόμα, σε αναγνώριση ήχου και μουσικής.

Η OpenCv ήταν ένα βασικό συστατικό του συστήματος όρασης του ρομπότ από το πανεπιστήμιο Stanford, του "Stanley", το οποίο κέρδισε τον DARPA Grand Challenge, αγώνα δρόμου ρομπότ στην έρημο.

4.1.6 Συναρτήσεις και δομές της OpenCv που χρησιμοποίησα

Δομές της OpenCV :

- **Mat :** Η κλάση Mat αντιπροσωπεύει ένα ν-διαστατό πυκνό αριθμητικό μονού καναλιού ή πολυκάναλου πίνακα. Μπορεί να χρησιμοποιηθεί για την αποθήκευση πραγματικών ή μιγαδικών διανυσμάτων και πινάκων, σε κλίμακα του γκρι ή έγχρωμες εικόνες, τρισδιάστατους όγκους, διανυσματικά πεδία, σμήνη σημείων, τανύστες και ιστογράμματα.
- **Point_ :** Πρότυπη κλάση για δισδιάστατα σημεία που καθορίζονται από τις συντεταγμένες x και y της. Ένα στιγμιότυπο της κλάσης είναι συμβατό με δομές της C, CvPoint και CvPoint2D32f. Υπάρχει επίσης ένας τελεστής Cast για την μετατροπή από συντεταγμένες τύπου point, σε έναν καθορισμένο τύπο. Η μετατροπή από συντεταγμένες κινητής υποδιαστολής σε ακέραιες συντεταγμένες, γίνεται με στρογγυλοποίηση.
- **Size_ :** Η πρότυπη κλάση Size χρησιμοποιείται για τον καθορισμό μιας εικόνας η ενός ορθογωνίου. Η κλάση περιλαμβάνει δύο μέλη, τα ποια ονομάζονται πλάτος (width) και ύψος (height). Αυτή η δομή μπορεί να μετατραπεί και σε παλιές δομές της OpenCV, όπως CvSize και CvSize2D32f. Το ίδιο σετ από αριθμητικούς και συγκριτικούς τελεστές είναι διαθέσιμο και για την Point_.
- **Rect_:** Πρότυπη κλάση για δισδιάστατα ορθογώνια, περιγράφεται από τις παρακάτω παραμέτρους:
    - Συντεταγμένες της πάνω αριστερής γωνίας tl(). Με προκαθορισμένους παραμέτρους Rect_::x και Rect_::y στην OpenCV. Αν και σε αλγόριθμους, υπάρχει η δυνατότητα χρήσης των x και y συντεταγμένων της κάτω δεξιάς γωνίας br().
    - Ορθογώνιο(Rectangle) με πλάτος(width) και ύψος (height).

  Η OpenCV υποθέτει ότι το πάνω αριστερό όριο του ορθογωνίου, περιλαμβάνεται ενώ το κάτω αριστερό όχι. Για παράδειγμα η μέθοδος Rect_::contains επιστέφει αληθές(true) αν

  $$x \leq pt.x < x + width, y \leq pt.y < y + height$$

- **RotatedRect :** Η κλάση αντιπροσωπεύει περιστρεφόμενα ορθογώνια σε ένα επίπεδο. Κάθε ορθογώνιο προσδιορίζεται από ένα κεντρικό σημείο, το μήκος της κάθε πλευράς και τη γωνία περιστροφής σε μοίρες.

  C++: RotatedRect::**RotatedRect**()

  C++: RotatedRect::**RotatedRect**(const Point2f&center,constSize2f& size,float angle)

  C++: RotatedRect::**RotatedRect**(const CvBox2D& **box**)

  C++: void RotatedRect::**points**(Point2f **pts**[]) const

  C++: Rect RotatedRect::**boundingRect**() const

  C++: RotatedRect::**operator** CvBox2D() const

- **Scalar_** Πρότυπη κλάση για 4-στοιχεία διάνυσμα, η οποία προέρχεται από το Vec <_Tp, 4>, Scalar_ και Scalar μπορεί να χρησιμοποιηθεί ως τυπικό 4-στοιχεία διάνυσμα. Επιπλέον, μπορούν να μετατραπούν σε / από CvScalar. Ο τύπος Scalar χρησιμοποιείται ευρέως στην OpenCV για να περάσει τιμές εικονοστοιχείων.
- **CvSVMParams** Η δομή πρέπει να αρχικοποιηθεί και να περάσει στην μέθοδο εκπαίδευσης της CvSVM.

  Ο κατασκευαστής.

  - **C++:** CvSVMParams::**CvSVMParams**()
  - **C++:** CvSVMParams::**CvSVMParams**(int **svm_type**, int **kernel_type**, double **degree**, double **gamma**, double **coef0**, double **Cvalue**, double**nu**, double **p**, CvMat* **class_weights**, CvTermCriteria **term_crit**)

  Παράμετροι: **svm_type** –

  Τύποι παραμέτρων SVM:

  - **CvSVM::C_SVC** C-support ταξινόμηση διανυσμάτων. n-class ταξινόμηση ($n \geq 2$) επιτρέπει τον ατελή διαχωρισμό των κατηγοριών με ποινή πολλαπλασιαστή C για ακραίες τιμές.
  - **CvSVM::NU_SVC** ν-Support ταξινόμηση διανυσμάτων. n-κλάση ταξινόμησης με πιθανή ατελή διαχωρισμό. Παράμετρος (με εύρος 0 .. 1, όσο μεγαλύτερη είναι η τιμή, τόσο πιο ομαλό το όριο αποφάσεως) χρησιμοποιείται αντί του C.

  Τύποι SVM πύρινα: **kernel_type** –

  - **CvSVM::LINEAR** Γραμμικός πυρήνας.

    $$K(x_i, x_j) = x_i^T x_j$$

  - **CvSVM::POLY** Πολυωνυμικός πυρήνας:

    $$K(x_i, x_j) = (\gamma x_i^T x_j + coef0)^{degree}, \gamma > 0$$

  - **CvSVM::RBF** Συνάρτηση ακτινικής βάσης(RBF)

    $$K(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2}, \gamma > 0$$

  - **CvSVM::SIGMOID** Σιγμοειδής πυρήνας:

    $$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + coef0)$$

- **CvSVM::train** Εκπαιδεύει ένα SVM:

  - **C++:** bool CvSVM::**train**(const Mat& **trainData**, const Mat& **responses**, const Mat& **varIdx**=Mat(), const Mat& **sampleIdx**=Mat(), CvSVMParams **params**=CvSVMParams())

  - **C++:** bool CvSVM::**train**(const CvMat* **trainData**, const CvMat* **responses**, const CvMat* **varIdx**=0, const CvMat* **sampleIdx**=0, CvSVMParams **params**=CvSVMParams())

Συναρτήσεις της OpenCV που χρησιμοποίησα:

- **Rectangle :** Η συνάρτηση Rectangle σχεδιάζει ένα ορθογώνιο περίγραμμα ή ένα γεμάτο ορθογώνιο του οποίου οι δυο απέναντι γωνίες είναι pt1 και pt2, ή r.tl () και r.br ()-Point (1,1).
- **Threshold :** Η συνάρτηση εφαρμόζει σταθερού επιπέδου κατωφλίωση σε μονοκάναλη εικόνα. Η συνάρτηση χρησιμοποιείται για την μετατροπή μιας εικόνας σε δυαδική από κλίμακα του γκρι ή για την αφαίρεση θορύβου, δηλαδή, φιλτράρονται και απορρίπτονται πολύ μικρές ή πολύ μεγάλες τιμές των εικονοστοιχείων. Υπάρχουν πολύ τύποι κατωφλίωσης (Εικόνα 12) που υποστηρίζονται από τη συνάρτηση όπως φαίνεται παρακάτω:

  - **THRESH_BINARY**

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

  - **THRESH_BINARY_INV**

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

  - **THRESH_TRUNC**

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

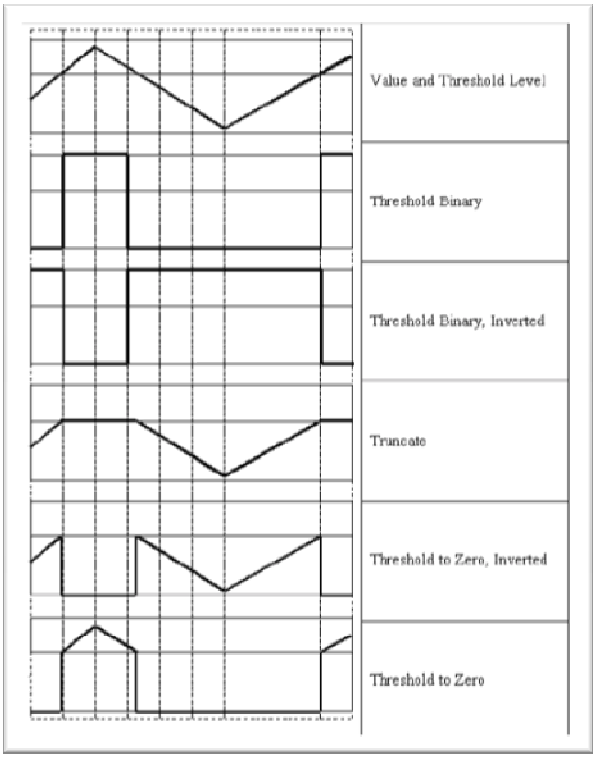
  - **THRESH_TOZERO**

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

  - **THRESH_TOZERO_INV**

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

Επίσης ειδική τιμή THRESH_OTSU μπορεί να συνδυαστεί με μια από τις παραπάνω τιμές. Σε αυτήν την περίπτωση, η λειτουργία καθορίζει τη βέλτιστη τιμή κατωφλίου χρησιμοποιώντας τον αλγόριθμο του Otsu και χρησιμοποιείται αντί για καθορισμένη τιμή κατωφλιού. Η συνάρτηση επιστρέφει την υπολογισμένη τιμή κατωφλιού. Επί του παρόντος, η μέθοδος Otsu εφαρμόζεται μόνο για εικόνες 8-bit.



Εικόνα 12 Επιλογές κατωφλίωσης

- **Resize :** Η συνάρτηση resize αλλάζει το μέγεθος μιας εικόνας μικραίνοντας ή μεγαλώνοντας τις διαστάσεις ανάλογα με το καθορισμένο μέγεθος.
- **cvtColor :** Η συνάρτηση μετατρέπει μια εικόνα εισόδου από έναν χρωματικό χώρο σε έναν άλλο. Σε περίπτωση μετατροπής από ή σε RGB χρωματικό χώρο, η σειρά των καναλιών θα πρέπει να προσδιορίζεται ρητά (RGB ή BGR). Σημειώστε ότι η προεπιλεγμένη μορφή για το χρώμα στο OpenCV συχνά αναφέρεται ως RGB, αλλά στην πραγματικότητα είναι BGR (τα bytes αντιστρέφονται). Έτσι, το πρώτο byte σε μια (24-bit) έγχρωμη εικόνα θα είναι το πρώτο byte 8-bit μπλε, το δεύτερο byte θα είναι πράσινο, και το τρίτο byte θα είναι κόκκινο.

  0 to 255 for CV_8U images
  0 to 65535 for CV_16U images
  0 to 1.0 for CV_32F images

- **warpAffine :** Εφαρμόζει τον affine μετασχηματισμό σε μια εικόνα.

4.2 Zbar

4.2.1 Εισαγωγή στην Zbar

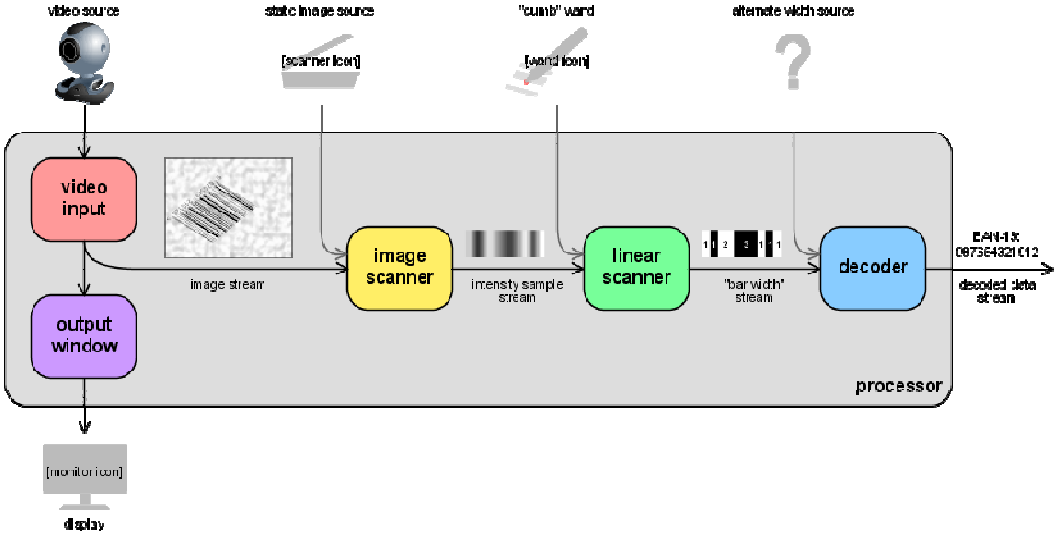
Η Zbar είναι μια σουίτα λογισμικού ανοιχτού κώδικα για την ανάγνωση barcode από διάφορες πηγές όπως ροές βίντεο, αρχεία εικόνας και ειδικούς αισθητήρες έντασης φωτός. Υποστηρίζει πολλούς δημοφιλείς τύπους bar code συμπεριλαμβανομένων των EAN-13/UPC-A, UPC-E, EAN-8, Code 128, Code 39, Interleaved 2 of 5 και QR Code. Η ευέλικτη, πολυεπίπεδη εφαρμογή διευκολύνει την ανάγνωση και αποκωδικοποίηση bar code για κάθε εφαρμογή. Η Zbar είναι υπό την GNU LGPL 2.1 άδεια, για να καταστεί δυνατή η ανάπτυξη ανοιχτού λογισμικού αλλά και εμπορικών εφαρμογών.

4.2.2 Τρόπος λειτουργιάς της Zbar

Η βιβλιοθήκη Zbar χρησιμοποίει μια προσέγγιση πιο κοντά σε αυτή που χρησιμοποιείται σε σαρωτές τύπου ράβδος και λέιζερ. Τα γραμμικά(1D) bar codes σχεδιαστήκαν για να αποκωδικοποιούνται από έναν απλό αισθητήρα που περνά πάνω από τις φωτεινές και σκοτεινές περιοχές του bar code.

Αξιοποιώντας αυτό, η υλοποίηση της Zbar κάνει γραμμική σάρωση πάνω από μια εικόνα, αντιμετωπίζοντας κάθε εικονοστοιχείο, ως δείγμα από έναν ενιαίο αισθητήρα φωτός. Τα δεδομένα σαρώνονται, αποκωδικοποιούνται και συναρμολογούνται αμέσως. Παίρνοντας ως σημείο έμπνευσης μοντέρνα παραδείγματα επεξεργασίας, η Zbar μεταφέρει την ιδέα αυτή σε ένα πολυεπίπεδο μοντέλο συνεχούς ροής.

Η επεξεργασία διαχωρίζεται σε ανεξάρτητα επίπεδα με σαφώς καθορισμένες διασυνδέσεις, οι οποίες μπορούν να χρησιμοποιηθούν μαζί ή ξεχωριστά να συνδεθούν σε ένα οπουδήποτε σύστημα. Μια υψηλού επιπέδου περιγραφή από τα modules φαίνεται στην παρακάτω εικόνα (Εικόνα 13).



Εικόνα 13 Υψηλού επιπέδου περιγραφή από τα modules

4.2.3 Χαρακτηριστικά της Zbar

- Υποστηρίζει πολλές πλατφόρμες Linux/Unix, Windows, iPhone®, embedded...
- Υψηλή ταχύτητα - σάρωση σε πραγματικό χρόνο από ροές βίντεο.
- Μικρό ίχνος μνήμης.
- Μικρό μέγεθος κώδικα - ο πυρήνας του σαρωτή και αποκωδικοποιητή EAN είναι κάτω από 1K γραμμές σε κώδικα C.
- Δεν περιορίζεται σε εικόνες.
- Κατάλληλο για ενσωματωμένες εφαρμογές που χρησιμοποιούν χαμηλού κόστους υλικό.
- Αρθρωτά στοιχεία μπορούν να χρησιμοποιηθούν μαζί ή χωριστά

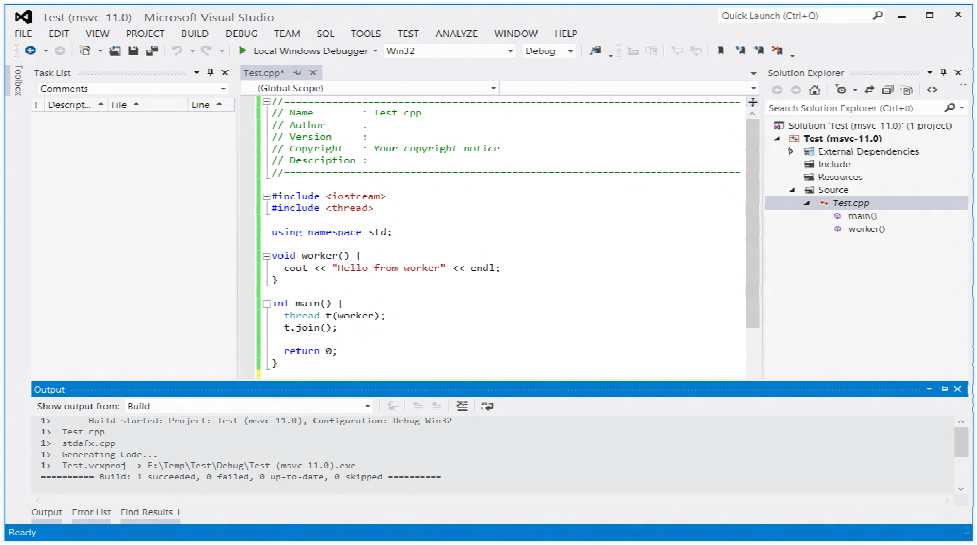
4.3 Microsoft visual studio 2012

4.3.1 Εισαγωγή στο visual studio 2012

Το Visual Studio 2012 (Εικόνα 14), είναι ένα εύχρηστο περιβάλλον ανάπτυξης εφαρμογών, που αναπτύχθηκε από την εταιρεία Microsoft Corporation. Υποστηρίζει την ανάπτυξη προγραμμάτων σε κονσόλα, οπτικές εφαρμογές, ιστοσελίδες, υπηρεσίες WEB και άλλα.

Το περιβάλλον ανάπτυξης Visual Studio 2012, βοηθά τον προγραμματιστή να αναπτύξει τα προγράμματά του με σχετική ευκολία, καθώς η τεχνολογία Microsoft IntelliSense βοηθά τον προγραμματιστή να κατανοήσει με ευκολία πιθανά λάθη του κώδικά του, υπογραμμίζοντας τα με κόκκινη γραμμή δυναμικά κατά το χρόνο συγγραφής του προγράμματος.

Αυτή η τεχνολογία είναι ικανή να εντοπίσει λάθη τα οποία μπορεί να είναι είτε συντακτικά, όπως για παράδειγμα η χρήση μιας εντολής με εσφαλμένο τρόπο, είτε λογικά, όπως για παράδειγμα η δήλωση ενός αντικειμένου χωρίς αυτό να χρησιμοποιείται. Το Visual Studio 2010 υποστηρίζει την ανάπτυξη προγραμμάτων στις C++, C#, Visual Basic, F# και τη μεταφορά προγραμμάτων από τη μία γλώσσα στην άλλη. Με άλλα λόγια συγγράφει ένα πρόγραμμα σε γλώσσα C++ μπορεί απλά και εύκολα να μετατραπεί αυτόματα σε κάποια εκ των γλωσσών που υποστηρίζει το Visual Studio.



Εικόνα 14 Visual Stusio 2012 SDK

4.3.1 Εκδόσεις του visual studio

Product name	Codename	Internal version	Supported .NET Framework versions	Release date
Visual Studio	N/A	4.0	N/A	April 1995
Visual Studio 97	*Boston*	5.0	N/A	February 1997
Visual Studio 6.0	*Aspen*	6.0	N/A	June 1998
Visual Studio .NET (2002)	*Rainier*	7.0	1.0	February 13, 2002
Visual Studio .NET 2003	*Everett*	7.1	1.1	April 24, 2003
Visual Studio 2005	*Whidbey*	8.0	2.0, 3.0	November 7, 2005
Visual Studio 2008	*Orcas*	9.0	2.0, 3.0, 3.5	November 19, 2007
Visual Studio 2010	*Dev10/Rosario*	10.0	2.0, 3.0, 3.5, 4.0	April 12, 2010
Visual Studio 2012	*Dev11*	11.0	2.0, 3.0, 3.5, 4.0, 4.5	September 12, 2012
Visual Studio 2013	*Dev12*	12.0	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1	October 17, 2013

Πίνακας 2 Εκδόσεις του visual studio

4.4 Η βιβλιοθήκη dirent.h

Η Dirent είναι μια διεπαφή προγραμματισμού εφαρμογών (API) που επιτρέπει στους προγραμματιστές να απαριθμήσουν τα αρχεία και τους καταλόγους. Η Dirent API είναι διαθέσιμη σε όλα τα συστήματα UNIX, αλλά δεν είναι διαθέσιμη για όλους τους compilers των Windows.

5. Ανάλυση συστήματος

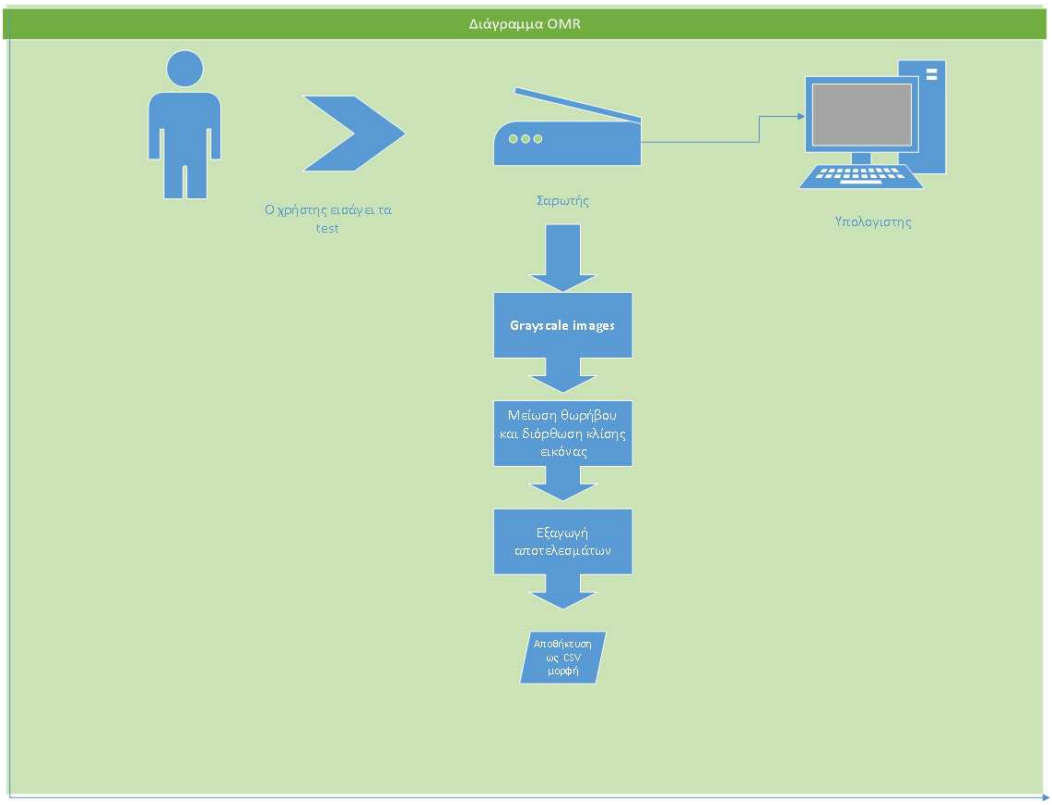
Στο κεφάλαιο αυτό γίνεται αναλυτική περιγραφή του αλγορίθμου αυτόματης διόρθωσης τεστ πολλαπλής επιλογής, του οποίου η ροή εργασίας φαίνεται στην (Εικόνα 15).

Αρχικά, ο χρήστης αφού συγκεντρώσει τα τεστ, τα εισάγει σε έναν σαρωτή με αυτόματο τροφοδότη εγγράφων, η σάρωση γίνεται σε κλίμακα του γκρι και έπειτα τα σαρωμένα έγγραφα των τεστ εισάγονται στα συστήματα μας.

Επόμενο στάδιο της επεξεργασίας, είναι η μείωση θορύβου από τις εικόνες και οι τυχών παραμορφώσεις στην κλίση του εγγράφου που ενδεχομένως να δημιουργήθηκαν κατά την διαδικασία της σάρωσης.

Έπειτα, πλέον απαλλαγμένες από θόρυβο και προβλήματα στην κλίση τους οι εικόνες εισάγονται στον αλγόριθμο για αναγνώριση των απαντήσεων και εξαγωγή των αποτελεσμάτων .

Τέλος, αφού τα αποτελέσματα εξαχθούν για κάθε τεστ που έχει εισάγει ο χρήστης τα αποτελέσματα αποθηκεύονται σε ένα αρχείο csv.



Εικόνα 15 Διάγραμμα OMR

5.1 Ανάλυση προβλήματος

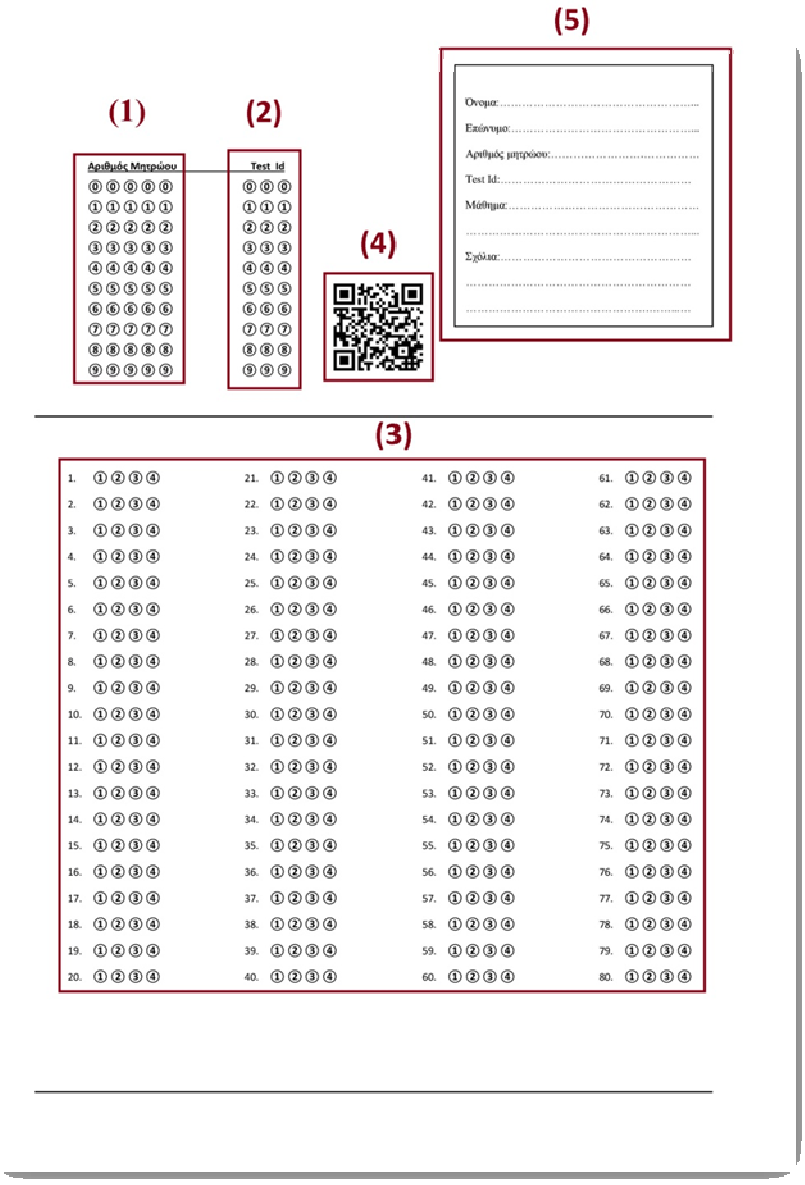
Σε αυτό το υποκεφάλαιο γίνεται ανάλυση του προβλήματος. Για την ανάλυση του προβλήματος θα πρέπει αρχικά να καθορίσουμε τα δεδομένα και τα ζητούμενα της εφαρμογής μας.

- **Δεδομένα.**

  Η εφαρμογή που θα σχεδιάσαμε αποτελείται από μια φόρμα μεγέθους Α4 (Εικόνα 16) η όποια είναι προσχεδιασμένη από εμάς και δεν υπάρχει δυνατότητα παραμετροποίησης της.
  Αναλυτικότερα, η φόρμα που θα σχεδιάσαμε θα πρέπει να αποτελείται από τα έξης πεδία:

  - **Αριθμός μητρώου :** Αυτό το πεδίο θα πρέπει να έχει μέγεθος 5, δηλαδή ο μέγιστος αριθμός μητρώου που θα μπορεί να συμπληρώσει ο φοιτητής θα είναι πενταψήφιος, ο όποιος είναι και ο μέγιστος αριθμός στις σχόλες του ΤΕΙ Κρήτης [Εικόνα 16 (1)].
  - **Αριθμός διαγωνίσματος :** Αυτό το πεδίο θα πρέπει να έχει μέγεθος 3, ο σκοπός αυτού του πεδίου είναι να έχει ο καθηγητής τη δυνατότητα να μπορεί να δημιουργεί διαφορετικά διαγωνίσματα για κάθε φοιτητή. Δηλαδή, σε κάθε εξέταση θα μπορούν να δημιουργηθούν μέχρι και 1000 διαφορετικά διαγωνίσματα [Εικόνα 16 (2)].
  - **Περιοχή απαντήσεων :** Αυτή η περιοχή θα αποτελείται από 80 απαντήσεις, με 4 δυνατές επιλογές η κάθε απάντηση. Πιο συγκεκριμένα, θα αποτελείται από 4 στήλες των 20 απαντήσεων [Εικόνα 16 (3)].
  - **Περιοχή QRcode :** Σε αυτή την περιοχή θα υπάρχει χώρος για την τοποθέτηση του QRcode [Εικόνα 16 (4)].
  - **Περιοχή πληροφοριών φοιτητή :** Αυτή η περιοχή δεν θα λαμβάνεται υπόψη από την εφαρμογή μας και θα περιέχει όλες τις απαραίτητες πληροφορίες που θα χρειάζεται να ξέρει ο καθηγητής.

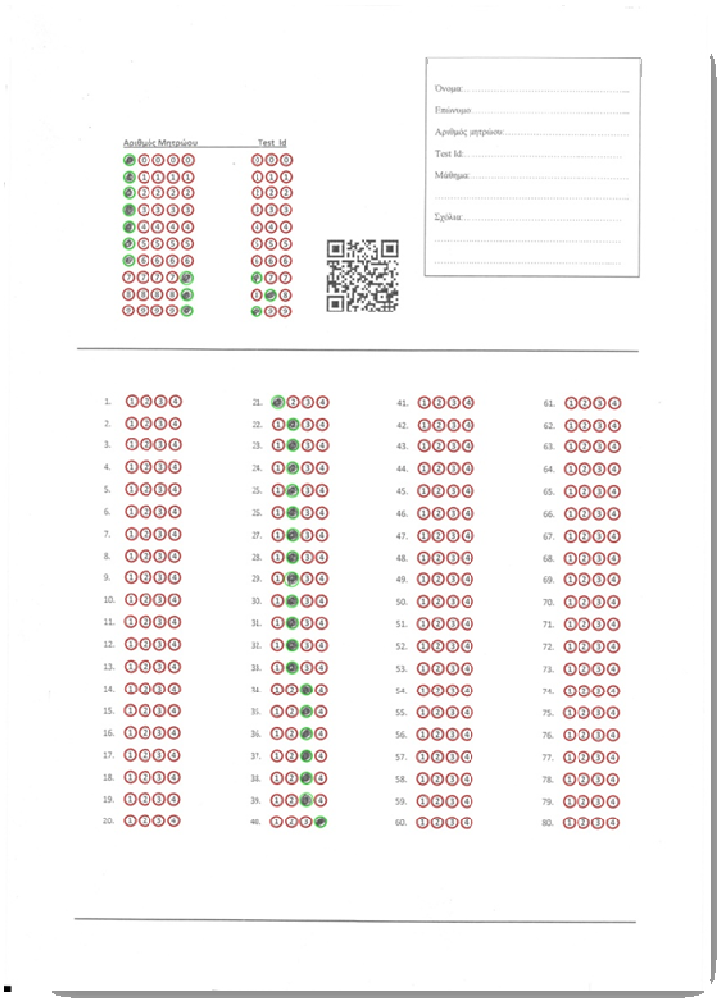
    Αυτές οι πληροφορίες ορίστηκαν να είναι : το όνομα του φοιτητή, το επώνυμο, ο αριθμός μητρώου ,ο αριθμός διαγωνίσματος, ο τίτλος του μαθήματος και μια περιοχή για σχόλια του φοιτητή [Εικόνα 16 (5)].



Εικόνα 16 Πρότυπη φόρμα απαντήσεων

- **Ζητούμενα**

  Η εφαρμογή μας θα πρέπει να είναι σε θέση να εντοπίζει τις κυκλικές περιοχές(bubbles), στις οποίες ο φοιτητής σημειώνει τη απάντηση, να αποθηκεύει τις συντεταγμένες τους και έπειτα να εισάγονται στην μηχανή υποστήριξης διανυσμάτων (SVM), την οποία έχουμε εκπαιδεύσει προηγουμένως με γνωστά πρότυπα, για την επαλήθευση τους. Δηλαδή, αν η απάντηση έχει επιλεγεί ή όχι από τον φοιτητή . Τέλος, τα αποτελέσματα θα εξάγονται σε μορφή CSV.



Εικόνα 17 Έξοδος του συστήματος. Με πράσινο χρώμα επιλεγμένες απαντήσεις με κόκκινο απαντήσεις που δεν έχουν επιλεγεί

5.2 Υλοποίηση

Στο υποκεφάλαιο αυτό αναλύουμε τη διαδικασία εξαγωγής των απαντήσεων. Η διαδικασία αποτελείται από 5 βασικά βήματα όπως φαίνεται και στην (Εικόνα 18). Αρχικά, οι σαρωμένες εικόνες εισάγονται στο σύστημα. Το σύστημα μας δέχεται μόνο αρχεία JPEG, για κάθε αρχείο εικόνας ελέγχεται η ορθότητα του και έπειτα, αποθηκεύεται για να συνεχίσει στο επόμενο βήμα όπου είναι η διόρθωση της κλίσης της εικόνας. Στο βήμα 3 αναλαμβάνει ο αλγόριθμος εξαγωγής των αποτελεσμάτων και στο βήμα 4 η μηχανή διανυσμάτων υποστήριξης αναγνωρίζει, αν ένα bubble είναι επιλεγμένο ή όχι και τέλος, τα αποτελέσματα αποθηκεύονται σε ένα CSV αρχείο. Όπως και οι διορθωμένες εικόνες, όπου πάνω με πράσινο κύκλο θα είναι οι επιλεγμένες απαντήσεις από τον χρηστή και με κόκκινο αυτές που δεν έχει επιλέξει (Εικόνα 17).

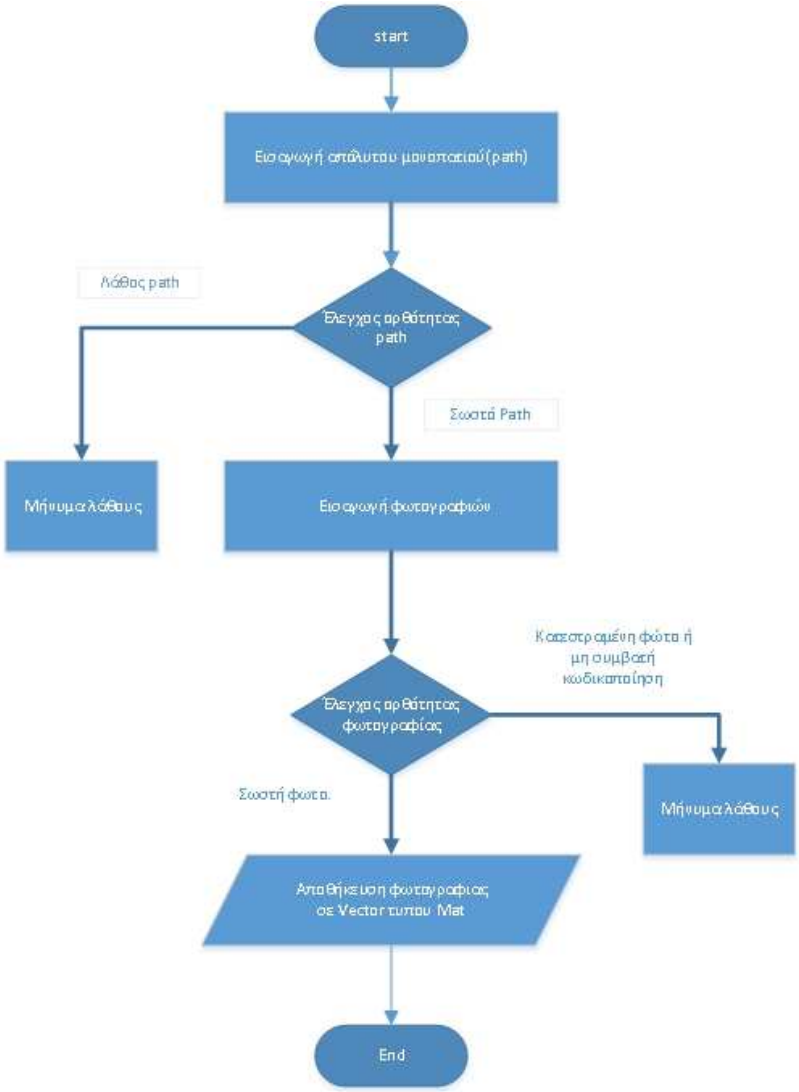
Εικόνα 18 Διάγραμμα ροής OMR

Η αποθήκευση των εικόνων των τεστ θα γίνεται σε ξεχωριστό φάκελο και ως όνομα αρχείου θα είναι ο αριθμός μητρώου και ο αριθμός τεστ, χωρισμένοι με κάτω παύλα. Θα έχει τη μορφή, δηλαδή, A.M_A.T.jpg (π.χ. 2334_123.jpg). Με αυτόν τον τρόπο ο καθηγητής θα μπορεί να ελέγχει τα τεστ που έχουν κάποιο πρόβλημα στην αναγνώριση ή ακόμα θα μπορεί να στείλει μέσω e-mail το τεστ στον φοιτητή για να επιβεβαιώσει τα λάθη του.

5.2.1 Εισαγωγή φωτογραφιών στον αλγόριθμο

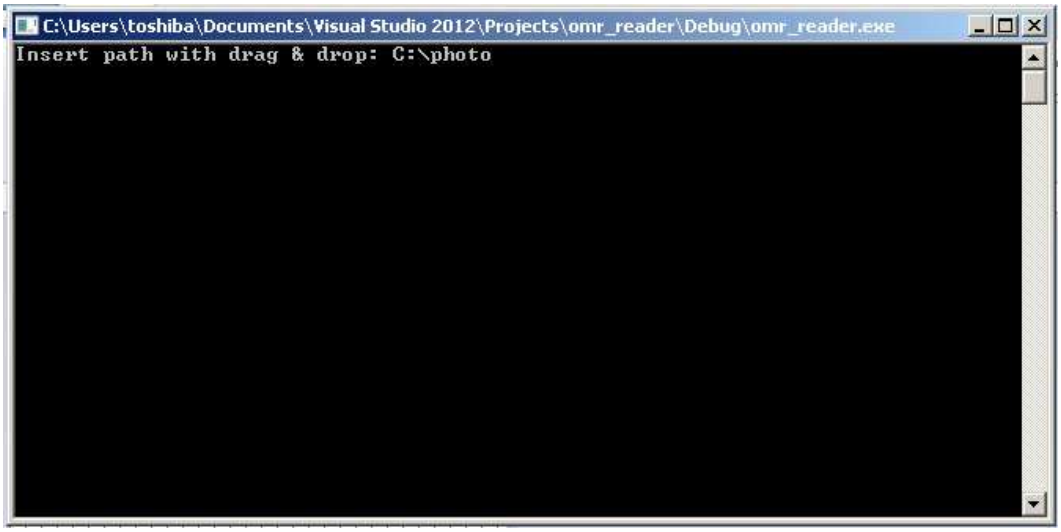
Αυτή η διαδικασία (Εικόνα 19) υλοποιείτε από την συνάρτηση `getImageis(string path,vector<string>& paths)` η οποία παίρνει δυο ορίσματα. Το πρώτο όρισμα με όνομα path και τύπου string δέχεται το path όπου βρίσκονται οι εικόνες. Το δεύτερο όρισμα είναι ένα vector τύπου string και αποθηκεύει τα απόλυτα path των φωτογραφιών που βρήκε η συνάρτηση. Η συνάρτηση επιστέφει ένα vector που περιέχει τις εικόνες που της δώσαμε.

Η συνάρτηση αρχικά ελέγχει αν το path είναι σωστό, και σε περίπτωση λάθους (π.χ. ανύπαρκτο path) τυπώνει μήνυμα λάθους στην κονσόλα. Έπειτα, με τις συναρτήσεις της dirent opendir() και readdir() έχουμε πρόσβαση στα αρχεία του φακέλου. Για κάθε αρχείο που υπάρχει στον φάκελο γίνετε έλεγχος αν είναι jpeg και αν δεν είναι κατεστραμμένο και τέλος τα σωστά αρχεία αποθηκεύονται σε ένα vector τύπου <Mat>. Όταν ολοκληρώσει η συνάρτηση την δουλεία της επιστρέφει στο πρόγραμμα ένα vector με εικόνες.



Εικόνα 19 Διάγραμμα ροής για την εισαγωγή εικόνων στον αλγόριθμο

Αρχικά, για την εισαγωγή των φωτογραφιών στον αλγόριθμο θα πρέπει να κάνουμε (Drag n' Drop) τον φάκελο που περιέχει τις εικόνες των τεστ ή μια εικόνα μόνη της στην κονσόλα (Εικόνα 20)



Εικόνα 20 Εισαγωγή path

Αφού αποθηκευτεί το path σε μια μεταβλητή τύπου string αφαιρείτε το σύμβολο " από την αρχή και το τέλος του path που προστίθεται αυτόματα από το λειτουργικό σύστημα.

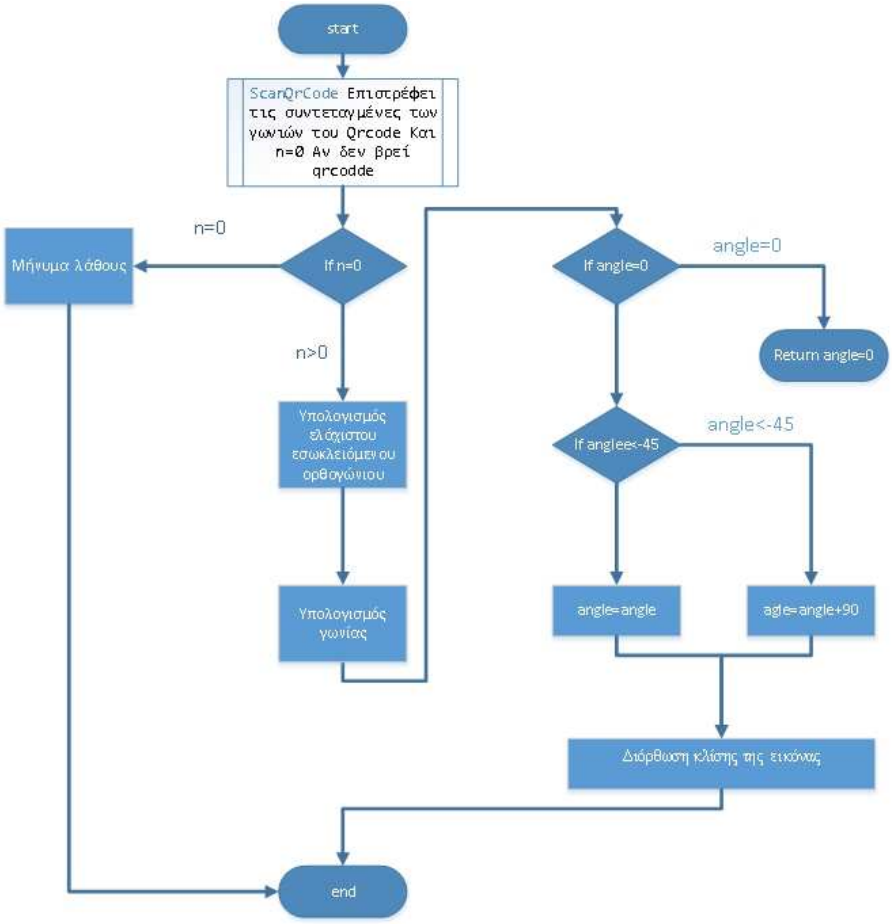
```
path.erase(remove( path.begin(), path.end(), '\"' ),path.end());
```

Έπειτα, κάνουμε έλεγχο για το αν είναι σωστό το path (1) και μετά με την χρήση της while(2) διαβάζουμε όλα τα αρχεία του φακέλου. Για κάθε αρχείο που διαβάζουμε γίνεται έλεγχος αν είναι σωστό(3) και σε περίπτωση λάθους τυπώνετε μήνυμα λάθους (4). Τέλος, τα σωστά αρχεία αποθηκεύονται σε ένα vector τύπου <Mat> με όνομα OmrImages(5) το οποίο επιστέφει η συνάρτηση μόλις ολοκληρωθεί η διαδικασία.

```
DIR *dir;
struct dirent *ent;
   (1)      if ((dir = opendir (path.c_str())) != NULL) {

                  int i=0;
                  (2) while ((ent = readdir (dir)) != NULL) {
                        string str_n(ent->d_name);

                        if(str_n.compare(".") &&  str_n.compare(".."))
                        {
                              {
                              (3)Mat temp=imread(path+"\\"+ent->d_name,1);

                                    (3)if(!temp.data)
                                    {
                  (4)cout<<"Fail image :"+path+"\\"+ent->d_name<<endl;
                                          continue;
                                    }
                  (5)OmrImages.push_back(imread(path+"\\"+ent->d_name,1));
                                    paths.push_back(path+"\\"+ent->d_name);
                                    i++;
                              }
                        }
                  }
                  closedir (dir);
            }
```

5.2.2 Διόρθωση σφάλματος στην κλίση της εικόνας

Για την διόρθωση κλίσης της εικόνας δημιουργήσαμε την συνάρτηση int deskew(Mat& img_in,Mat& img_out) (Εικόνα 21),που έχει ως είσοδο μια εικόνα και έξοδο την διορθωμένη εικόνα. Αρχικά εισάγουμε την εικόνα στην συνάρτησης int ScanQrCode(Mat img,vector<Point>& vp) που έχει ως είσοδο μια εικόνα και έξοδο ένα vector τεσσάρων σημείων τύπου <Point> που αντιπροσωπεύουν τις τέσσερεις κορυφές του qrcode.



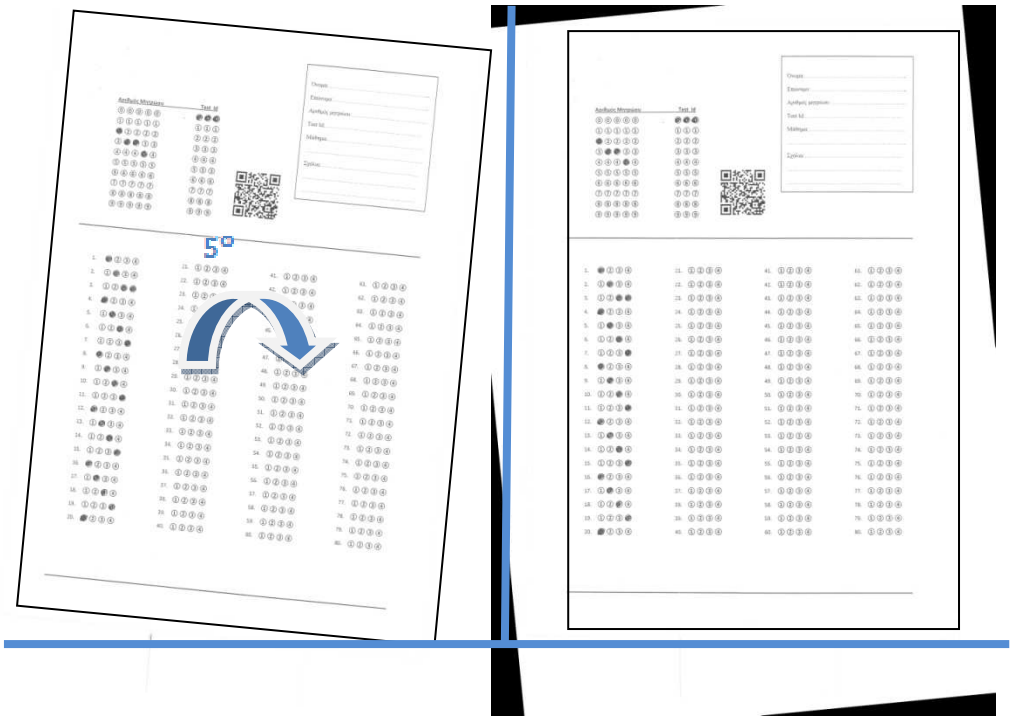
Εικόνα 21 Διάγραμμα ροής συνάρτησης διόρθωσης κλίσης της εικόνας

Έχοντας τέσσερα σημεία χρησιμοποιούμε την συνάρτηση `minAreaRect` που έχει είσοδο ένα vector τεσσάρων σημείων και επιστρέφει ένα αντικείμενο τύπου `RotatedRect` που έχει σαν ιδιότητα με όνομα angle, την γωνιά κλίσης του ορθογωνίου που σχηματίζουν οι τέσσερεις κορυφές του qrcode.

Η συνάρτηση minAreaRect παίρνει τιμές από -90 έως μηδέν μοίρες, μη συμπεριλαμβανομένου του μηδενός, δηλαδή, στο διάστημα [-90, 0). Όταν η γωνιά είναι -90 μοίρες η κλίση της εικόνας είναι μηδενική.

Για την διόρθωση της κλίσης εικόνας, χρησιμοποιούμε την συνάρτηση getRotationMatrix2D η οποία υπολογίζει μια μήτρα affine δισδιάστατης περιστροφής, μεγέθους, 3X2. Η getRotationMatrix2D έχει ως ορίσματα το κέντρο της εικόνας τύπου <Point> και την γωνιά περιστροφής σε μοίρες. Η συνάρτηση επιστρέφει μια εικόνα τύπου <Mat> μεγέθους 3X2, που χρησιμεύει ως είσοδος στην συνάρτηση `warpAffine`.

Η warpAffine έχει ως είσοδο την εικόνα που χρειάζεται διόρθωση και την μήτρα που δημιούργησε η getRotationMatrix2D και έξοδο την διορθωμένη εικόνα (Εικόνα 22).



Εικόνα 22 Διόρθωση κλίσης μετά από Affine μετασχηματισμό 5 μοίρες αριστερά

Στο παράδειγμα στην παραπάνω εικόνα φαίνετε η διόρθωση κλίσης κατά 5 μοίρες αριστερά. Το αποτέλεσμα της angle ήταν -85 μοίρες που προσθέτουμε 90 μοίρες 90+(-85)=5. Αν η κλίση ήταν αριστερά η angle θα είχε τιμή -5 μοίρες.

Σχολιασμός κώδικα

Η ScanQrCode(1) έχει ως είσοδο εικόνα και έξοδο ένα διάνυσμα τεσσάρων σημείων τα σημεία αυτά είναι συντεταμένες της μορφής x,y στο καρτεσιανό σύστημα, που αντιπροσωπεύουν τις κορυφές του qrcode. Η ScanQrCode επιστρέφει 0 αν δεν βρει qrcode.

Για να μπορέσουμε να βρούμε την γωνιά αρχικά δημιουργούμε ένα αντικείμενο RotatedRect με την συνάρτηση minAreaRect(2).

Στην if που ακολουθεί (3) γίνετε πρώτα έλεγχος αν η εικόνα δεν έχει κλίση. Όταν η τιμή της angle είναι -90 έχουμε μηδενική κλίση. Το διάστημα τιμών της angle είναι από [-90, 0). Γι΄αυτό η angle όταν παίρνει τιμές από (-45,-90] έχουμε δεξιά κλίση και απο [-45,0) αριστερή κλίση. Ενώ η getRotationMatrix2D όταν θέλουμε να περιστρέψουμε αριστερόστροφα έχουμε θετικό(+) πρόσημο και δεξιόστροφα αρνητικό πρόσημο(-).

Η getRotationMatrix2D(5) παίρνει ως είσοδο, ένα σημείο τύπου <Point> που αντιπροσωπεύει το κέντρο που θα περιστραφεί η εικόνα(4). Την γωνιά περιστροφής της εικόνας σε μοίρες, και ένα παράγοντα κλίμακας που εξ ορισμού είναι 1.

Η συνάρτηση υπολογίζει την παρακάτω μήτρα:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot \texttt{center.x} - \beta \cdot \texttt{center.y} \\ -\beta & \alpha & \beta \cdot \texttt{center.x} + (1-\alpha)\cdot \texttt{center.y} \end{bmatrix}$$

Όπου

$$\alpha = \texttt{scale} \cdot \cos \texttt{angle},$$
$$\beta = \texttt{scale} \cdot \sin \texttt{angle}$$

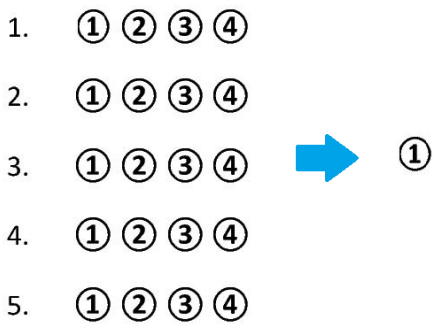
Το αποτέλεσμα της συνάρτησης ένας πινάκας 3X2 που χρησιμοποιεί η warpAffine(6) για να περιστρέψει την εικόνα.

Κώδικας σε C++ της συνάρτησης deskew

```
vector<Point> vp;
	(1)int n=ScanQrCode(img_in,vp);
	if(n==0){return 0;}
	(2)RotatedRect r1 = minAreaRect(vp);

	float angle=0;
	(3)if(r1.angle==-90)
	{
		angle=0;
	}
	else { if(r1.angle<-45)
	{
		angle=90+r1.angle;
	}
	else
	{
		angle= r1.angle;
	}
	}
	(4)cv::Point2d pt(img_in.size().width/2., img_in.size().height/2.);
	(5)cv::Mat r1r = cv::getRotationMatrix2D(pt, angle, 1.0);
	(6)cv::warpAffine (img_in, img_out, r1r, cv::Size(img_in.size().width,
	img_in.size().height));
}
```

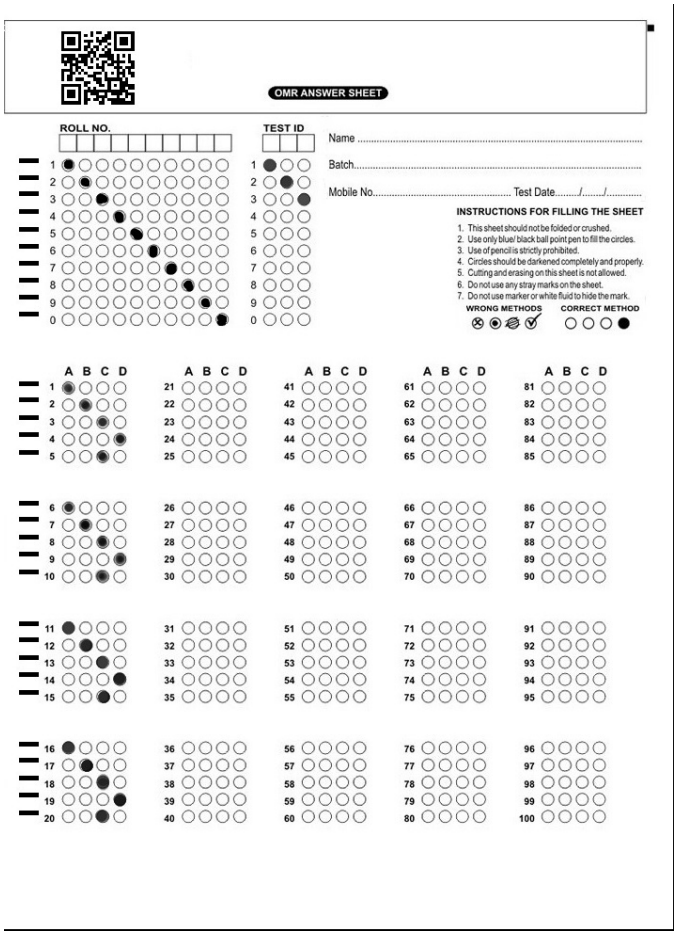
5.2.3 Εξαγωγή των αποτελεσμάτων

Στο υποκεφάλαιο αυτό θα αναλύσουμε την μέθοδο εξαγωγής των αποτελεσμάτων του συστήματος που αναπτύξαμε. Σκοπός του αλγόριθμου είναι να δέχεται μια εικόνα και να επιστρέφει ένα διάνυσμα vector<BubbleAnswers> που θα έχει αποθηκευμένες τις συντεταγμένες από όλα τα κέντρα των κύκλων(bubbles) (Εικόνα 23).



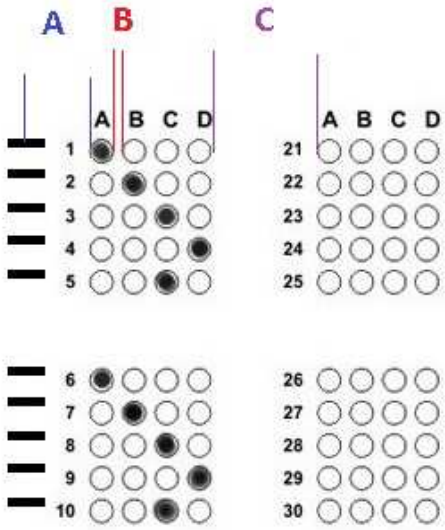
Εικόνα 23 Παράδειγμα κύκλου(bubble) απάντησης

Σημαντικότερο πρόβλημα που αντιμετωπίσαμε ήταν ο εντοπισμός των bubble στην εικόνα. Η πρώτη λύση που εξετάστηκε ήταν ο σχεδιασμός παυλών σε κάθε οριζόντια γραμμή από bubbles Εικόνα 24.



Εικόνα 24 Πρώτη υλοποίηση του τεστ

Αρχικά, ο αλγόριθμος που αναπτύξαμε εντόπιζε την κάθε παύλα και υπολόγιζε το κέντρο αυτής. Αφού πλέον έχουμε ένα σταθερό σημείο αναφοράς για κάθε γραμμή, μπορούμε να υπολογίσουμε την θέση κάθε bubble αυτόματα. Τα μόνα επιπλέον στοιχειά που χρειάζονται για τον υπολογισμό της θέσης κάθε bubble είναι τρεις αποστάσεις σε pixels. Θέλουμε λοιπόν να γνωρίζουμε την απόσταση μεταξύ του κέντρου της παύλας από το πρώτο bubble [Εικόνα 25(A)],την απόσταση μεταξύ των bubbles (B), και την απόσταση μεταξύ των στηλών (C). Οι αποστάσεις αυτές μένουν σταθερές σε για κάθε γραμμή. Με αυτόν τον τρόπο υπολογίζετε το κάθε κέντρο από όλα τα bubbles.



Εικόνα 25 Αποστάσεις μεταξύ των bubble

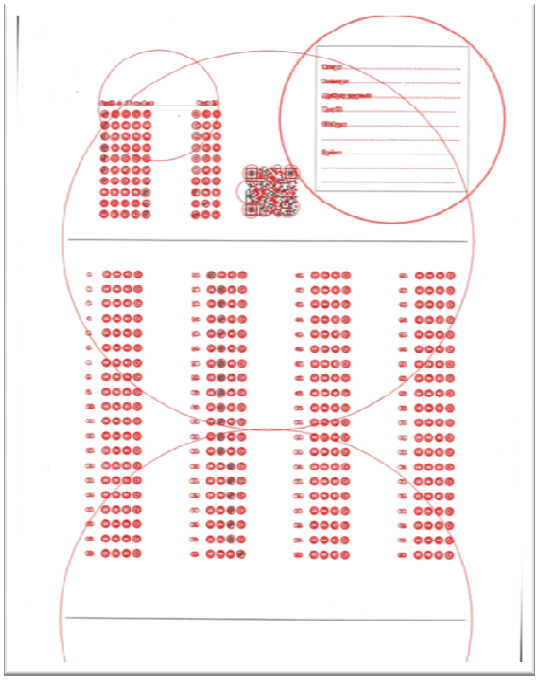
Αν και αυτή η τεχνική μας έδωσε ικανοποιητικά αποτελέσματα εγκαταλείφτηκε λόγο της πολυπλοκότητας της και του υπολογιστικού κόστους που απαιτούνταν.

Η δεύτερη λύση που δοκιμαστικέ, και υιοθετήθηκε τελικά, δεν χρησιμοποίει κάποιο σημάδι ελέγχου όπως η παύλα αλλά διαφορετική προσέγγιση. Ποιο αναλυτικά, για τον εντοπισμό των bubbles θα πρέπει πρώτα να μετατρέψουμε την εικόνα σε μια μορφή που να ευνοεί την αναγνώριση περιγραμμάτων(contours) . Αυτή η προεπεξεργασία γίνετε με την κατωφλίωση(thresholding) της εικόνας με τον αλγόριθμο του Otsu. Ο αλγόριθμος υποθέτει ότι η εικόνα που πρόκειται να κατωφλιωθεί περιέχει δύο κατηγορίες pixels ή διτροπικό ιστόγραμμα (π.χ. προσκήνιο και στο παρασκήνιο) και στη συνέχεια υπολογίζει το βέλτιστο όριο που χωρίζει αυτές τις δύο κατηγορίες, έτσι ώστε η διασπορά (intra-class διακύμανση) να είναι ελάχιστη.

Για την εξαγωγή των περιγραμμάτων χρησιμοποιείτε η συνάρτηση της OpenCV findContours η οποία έχει ως είσοδο την κατωφλιωμένη εικόνα και έξοδο ένα vector σε vector τύπου <Point> που έχει αποθηκευμένα τα σημεία που ορίζουν το περίγραμμα (Εικόνα 26).

Εικόνα 26 Σχεδιασμένα περιγράμματα

Επόμενο στάδιο της επεξεργασίας, είναι η εύρεση του ελάχιστου εσωκλειόμενου κύκλου για κάθε περίγραμμα. Το στάδιο αυτό υλοποιείτε από την minEnclosingCircle που έχει ως είσοδο το vector με τα περιγράμματα και ως έξοδο δυο vector που περιέχει το κέντρο κάθε κύκλου που δημιουργήθηκε και την ακτίνα του (Εικόνα 27).



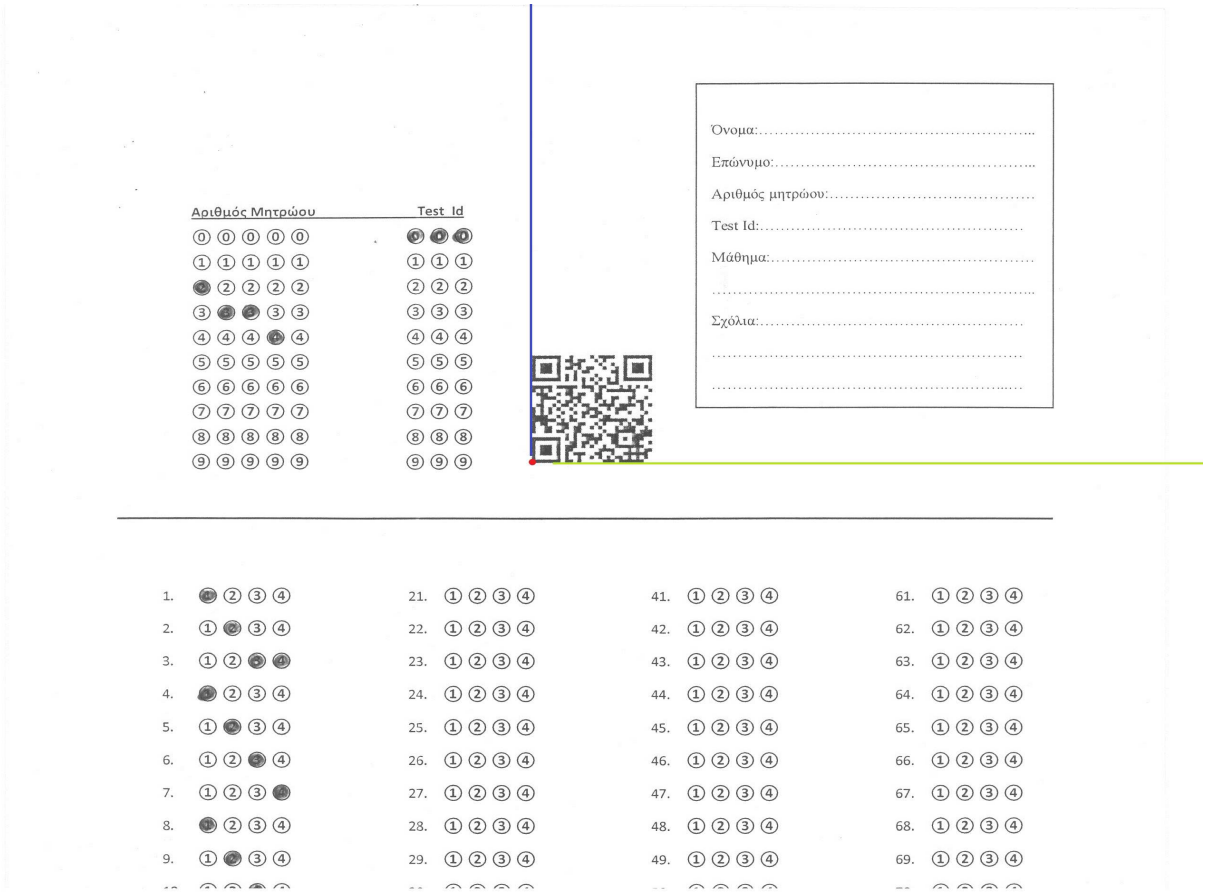
Εικόνα 27 Αποτέλεσμα της minEnclosingCircle

Όπως φαίνετε και στην προηγούμενη εικόνα όλα τα περιγράμματα που βρήκε η findContours και υπολογίστηκε ο ελάχιστος εσωκλειόμενος κύκλος σχεδιάστηκαν με κόκκινο χρώμα. Πολλοί κύκλοι όμως δεν μας αφορούν τους οποίους πρέπει να φιλτράρουμε.

Η φόρμα απαντήσεων σχεδιάστηκε με τέτοιο τρόπο ώστε το μέγεθος των bubbles να είναι συγκεκριμένο και να μην υπάρχει κάποιο άλλο στοιχειό πάνω στην εικόνα με αυτό το μέγεθος., το πρόβλημα όμως που πρόεκυψε ήταν ότι οι διαφορετικές αναλύσεις των εικόνων άλλαζαν και την κλίμακα οπότε το μέγεθος σε πίξελ του κάθε bubble δεν ήταν σταθερό.

Αυτό το πρόβλημα λύθηκε με την κλιμάκωση της εικόνας. Για να κλιμακώσουμε την εικόνα μετρήσαμε την ακμή του qrcode. Κάθε ακμή του qrcode είναι σταθερού μεγέθους το οποίο είναι 2,23 cm και διαιρέσαμε με το πλάτος σε πίξελ του qrcode το αποτέλεσμα είναι το μέγεθος κάθε πίξελ σε εκατοστά. Με αυτόν τον τρόπο μπορούμε να φιλτράρουμε τα περιγράμματα ώστε να πάρουμε το επιθυμητό αποτέλεσμα, γνωρίζοντας και την διάμετρο της bubble που είναι `0.3441 cm`

Ένας ακόμη περιορισμός που χρησιμοποιήσαμε είναι η απομόνωση της περιοχής που ο φοιτητής γράφει τα προσωπικά στοιχεία του. Αυτή η περιοχή ορίστηκε να είναι με αρχικό σημείο την κάτω αριστερή κορυφή του qrcode κόκκινο χρώμα (Εικόνα 28). Όσα σημεία είναι στον άξονα X δεξιά από την μπλε γραμμή από και όσα στον άξονα Y είναι πάνω από την πράσινη γραμμή δεν λαμβάνονται υπόψη από το πρόγραμμα (Εικόνα 28).



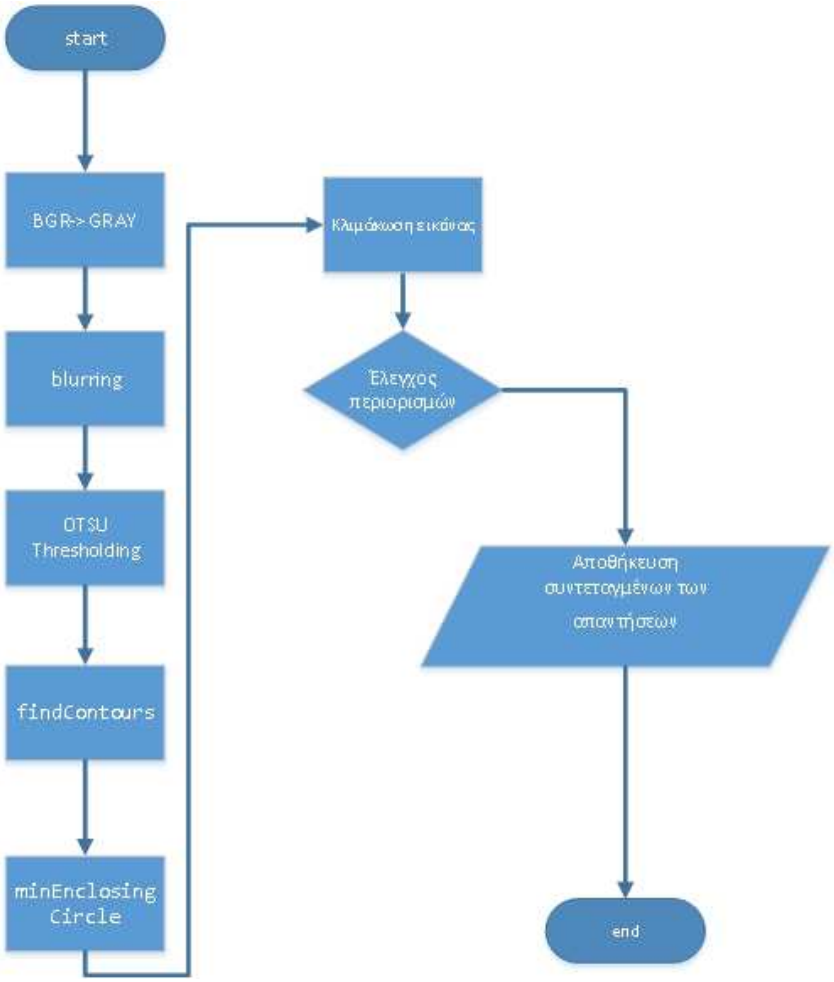
Εικόνα 28 Περιορισμοί για την αναγνώριση των bubble

Τέλος, αφού τα περιγράμματα φιλτραριστούν θα πρέπει να μείνουν μόνο τα bubbles τα οποία είναι 400 στο αριθμό, 320 είναι οι απαντήσεις 80X4, 50 είναι για τον αριθμό μητρώου και 30 για τον κωδικό του τεστ. Αν το αποτέλεσμα είναι διάφορο του 400 το τεστ απορρίπτετε και εμφανίζετε μήνυμα λάθους.

Τελικό στάδιο της επεξεργασίας των απαντήσεων είναι η αποθήκευση κάθε απάντησης σε μια δομή τύπου BubbleAnswers που δημιουργήσαμε για τον σκοπό αυτό. Η δομή κατασκευαστικέ με σκοπό την αποθήκευση της περικομένης εικόνας της bubble,το κέντρο της bubble και την ακτίνα της. Επίσης περιλαμβάνει και μια μεταβλητή που αποθηκεύει την κατάσταση της bubble ,-1 αν είναι επιλεγμένη και 1 αν δεν είναι. Μόλις ολοκληρωθεί η διαδικασία αυτή η συνάρτηση επιστρέφει ένα vector τύπου BubbleAnswers που περιέχει τις 400 bubbles.

Εξήγηση κώδικα

Η find_bubble_contours Εικόνα είναι η βασική συνάρτηση της εφαρμογής που υλοποιήσαμε, σαν είσοδο δέχεται μια εικόνα που προηγούμενος έχει υποστεί προεπεξεργασία ώστε να γίνουν τα περιγράμματα ποιο ευδιάκριτα, και αφού βρει τα περιγράμματα των bubbles επιστρέφει ένα vector τύπου BubbleAnswers.



Εικόνα 29 Διάγραμμα ροής της find_bubble_contours

Η δομή BubbleAnswers αποτελείτε από τέσσερεις μεταβλητές, την Mat_1D που αποθηκεύει την περικομμένη bubble, την center που συγκρατεί τις συντεταγμένες κάθε bubble,την radius στην οποία αποθηκεύεται η ακτίνα του.

```
struct BubbleAnswers {
	Mat Mat_1D;
	Point2f center;
	float radius;
	float response;
};
```

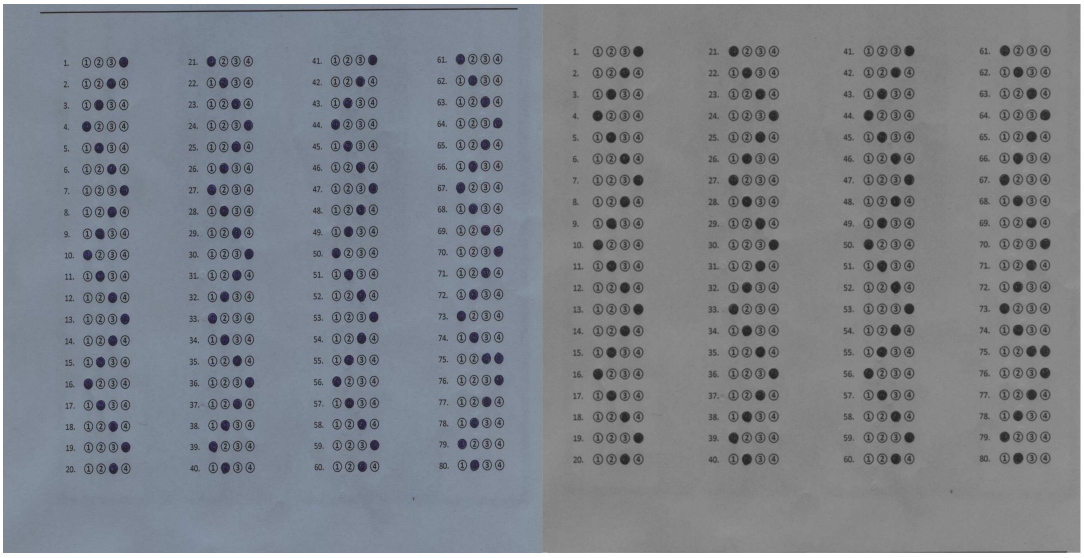
Πρώτο βήμα της διαδικασίας εξαγωγής των αποτελεσμάτων είναι η προεπεξεργασία των εικόνων. Αρχικά η εικόνα μετατρέπετε από έγχρωμη σε κλίμακα του γκρι.

```
cvtColor( img, img, CV_BGR2GRAY );
```

Δεύτερο βήμα είναι το θόλωμα της εικόνας έτσι ώστε να μειωθεί ο θόρυβος με την βοήθεια της συνάρτησης blur(Εικόνα 30). Η συνάρτηση εξομαλύνει μία εικόνα χρησιμοποιώντας τον πυρήνα που στην περίπτωσης μας είναι μεγέθους 3:

$$K = \frac{1}{\text{ksize.width*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ & & \cdots\cdots\cdots & & & \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

```
blur( img, img,Size(3,3));
```



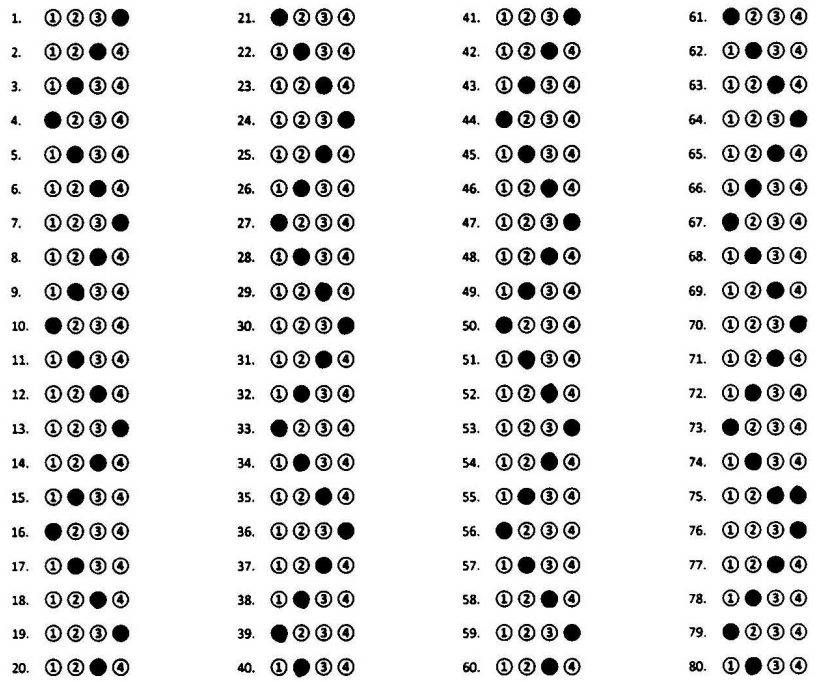
Εικόνα 30 Αριστερά αρχική εικόνα, δεξιά θολωμένη εικόνα

Τρίτο βήμα είναι η κατωφλίωση της εικόνας με την μέθοδο του Otsu (Nobuyuki, 1979). Ο αλγόριθμος του Otsu που τα βήματα του φαίνονται παρακάτω:

1. Υπολογισμός ιστογράμματος και πιθανοτήτων κάθε επιπέδου έντασης
2. Αρχικοποίηση $\omega_i(0)$ και $\mu_i(0)$
3. Υπολογισμός όλον των πιθανών κατωφλιών $t = 1 \ldots$ μέγιστη ένταση
    1. Ενημέρωση ω_i και μ_i
    2. Υπολογισμός $\sigma_b^2(t)$
4. Επιθυμητό κατώφλι αντιστοιχεί στο ανώτατο $\sigma_b^2(t)$
5. Μπορούμε να υπολογίσουμε τα δύο μέγιστα (και τα δύο αντίστοιχα κατώφλια) $\sigma_{b1}^2(t)$ είναι το μεγαλύτερο μέγιστο και $\sigma_{b2}^2(t)$ είναι το μεγαλύτερο ή ίσο με το μέγιστο.
6. Επιθυμητό κατώφλι= $\dfrac{\text{threshold}_1 + \text{threshold}_2}{2}$

Στο πρόγραμμα μας η παραπάνω διαδικασία γίνετε από την παρακάτω συνάρτηση
η τιμή CV_THRESH_OTSU καθορίζει το είδος της κατωφλίωσης το αποτέλεσμα της συνάρτησης φαίνετε στην (Εικόνα 31).

```
threshold( img, img,0 ,255, CV_THRESH_OTSU );
```



Εικόνα 31 Otsu threshold

Τελικό στάδιο για την εξαγωγή των αποτελεσμάτων, είναι η εύρεση των περιγραμμάτων της εικόνας ,και το φιλτράρισμα αυτών αφού πρώτα κλιμακώσουμε την εικόνα μας.
Η συνάρτηση findContours()(1) παίρνει τρεις παραμέτρους, η πρώτη είναι η πηγαία εικόνα δεύτερη είναι η λειτουργία ανάκτησης περιγράμματος ,τρίτη είναι η μέθοδος προσέγγισης του περιγράμματος. Και εξάγει τα περιγράμματα και την ιεραρχία. Η contours(2) είναι ένα vector που αποθηκεύει όλα τα περιγράμματα που εντόπισε στην εικόνα. Κάθε επιμέρους περίγραμμα είναι ένα vector τύπου <Point> (x, y) που περιέχει τις συντεταγμένες των σημείων του ορίου του αντικειμένου.

```
(2)vector<vector<Point> > contours;
   vector<Vec4i> hierarchy;

(1)findContours( img,contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point(0, 0) );
```

Αφού η συνάρτηση findContours μας επιστρέψει τα περιγράμματα θα πρέπει να φιλτραριστούν αφού πρώτα υπολογιστεί το μέγεθος του πίξελ σε cm. Για την διαδικασία αυτή μετρήσαμε το πλάτος του qrcode και την διάμετρο τις bubble. Αυτές οι μετρισεις οριζοντε στο πρόγραμμα μας με δυο σταθερές:

```
QRCODE_SIZE_IN_CM 2.23 cm
BUBBLE_DIAMETER_IN_CM 0.3441 cm
```

Πρώτα υπολογίζετε το μέγεθος του πίξελ σε εκατοστά, ο υπολογισμός γίνετε διαιρώντας το πλάτος του qrcode σε πίξελ με με το πλάτος του σε εκατοστά(1). Έπειτα υπολογίζετε το ελάχιστο(2) και μέγιστο(3) πλάτος κάθε bubble σε πίξελ και προστίθεται επίσης ένα offset για να υπάρχει ευελιξία στην αναγνώριση της bubble σε περίπτωση που ο φοιτητής βγει εκτός ορίων όταν συμπληρώνει την απάντηση. Το offset είναι -1 για το ελάχιστο μέγεθος και +19 για το μέγιστο μέγεθος.

```
(1)float pixel_size_in_cm = QRCODE_SIZE_IN_CM/qrcode.size.width;
(2)float min_bubble_size_in_pixels=((BUBBLE_DIAMETER_IN_CM*r.size.width)/QRCODE_SIZE_IN_CM-1);
(3)float max_bubble_size_in_pixels=((BUBBLE_DIAMETER_IN_CM*r.size.width)/QRCODE_SIZE_IN_CM)+19;
```

Για κάθε περίγραμμα που βρήκε η findContours γίνετε έλεγχος αν ικανοποίει τους περιορισμούς διαμέτρου(1).
.

```
(1)if(2*radius[i]>min_bubble_size_in_pixels && 2*radius[i]<max_bubble_size_in_pixels)
```

Και τους περιορισμούς θέσης της κάθε Bubble που βρήκε η συνάρτηση μας(2). Οι περιορισμοί θέσης αποκλείουν το ορθογώνιο που σχηματίζετε από την κάτω αριστερή κορυφή του qrcode.

```
(2)if(!(center[i].x>pts_bl.x && center[i].y<pts_bl.y))
```

Αφού ικανοποίει τις προηγούμενες σύνηθες περιορισμών κάθε bubble που μας αφορά περικόπτετε και η περικομμένη εικόνα της bubble με όνομα imgroi αποθηκεύεται σε μια μεταβλητή τύπου Mat(1).Έπειτα, γίνετε αλλαγή μεγέθους στην εικόνα σε συγκεκριμένο μέγεθος 30X30 πίξελ(2). Ο σκοπός αυτής της μετατροπής είναι ότι η μηχανή υποστήριξης διανυσμάτων(SVM), που η λειτουργιά της εξετάζετε στο επόμενο υποκεφάλαιο, ως είσοδο θέλει πρότυπα σταθερού μεγέθους. Αλλά, το μέγεθος κάθε περικομένης εικόνας εξαρτάτε από την ανάλυση της πηγαίας εικόνας και από το αν ο φοιτητής έχει βγει εκτός της bubble.

```
(1)Mat imgroi = img_src(Rect( center[i].x-radius[i],
       center[i].y-radius[i],2*radius[i],2*radius[i]));

(2)resize(imgroi,imgroi_resize,Size(30,30), 0, 0, INTER_CUBIC);
```

Τέλος, αφού έχουμε περικόψει την εικόνα και της αλλάξαμε μέγεθος, την μετατρέπουμε σε μονοδιάστατη(1) σε μορφή δηλαδή, που είναι συμβατή με την μηχανή υποστήριξης διανυσμάτων(SVM) και αποθηκεύουμε σε μια δομή τύπου BubbleAnswers(5) την περικομμένη εικόνα(2), το κέντρο της bubble(3), και την ακτίνα της(4).

```
(1)convert2Dto1D(imgroi_resize,tmp_1D_2D);
BubbleAnswers tmp;
(2)tmp.Mat_1D=tmp_1D_2D;
(3)tmp.center=center[i];
(4)tmp.radius= radius[i];
(5)bubbles.push_back(tmp);
```

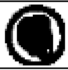
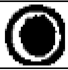
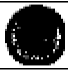
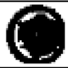
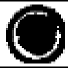
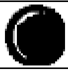
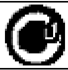
Μόλις γίνει αυτή η διαδικασία και για όλες τις bubbles η συνάρτηση επιστρέφει ένα vector τύπου `BubbleAnswers.`

5.2.4 Εκπαίδευση μηχανής υποστήριξης διανυσμάτων(SVM)

Η μηχανές υποστήριξης διανυσμάτων χρησιμοποιούνται για να ταξινομήσουμε πρότυπα σε δυο ή περισσότερες κατηγορίες, όπως αναφέραμε και στο υποκεφάλαιο 3.3. Η OpenCV υποστηρίζει αυτή την τεχνική μέσο μια βιβλιοθήκης.Η βιβλιοθήκη αυτή ονομάζεται Machine Learning Library (MLL), η οποία μας παρέχει την δυνατότητα, μέσω συναρτήσεων που υλοποιούνται σε C++, να κάνουμε στατιστικές αναλύσεις, ταξινόμηση προτύπων κ.α.

Αρχικά, χωρίσαμε τα πρότυπα μας σε δυο κατηγορίες, η πρώτη κατηγορία θα περιέχει όλα εκείνα τα πρότυπα(bubbles) που θα αναγνωρίσει η μηχανή υποστήριξης διανυσμάτων ως επιλεγμένα από τον χρήστη, στην δεύτερη κατηγορία θα περιέχει όλα τα πρότυπα(bubble) που ο χρήστης δεν έχει επιλέξει.

Η μηχανή υποστήριξης διανυσμάτων(SVM) αρχικά θα πρέπει να εκπαιδευτεί. Η εκπαίδευση έγινε με τη χρήση γνωστών προτύπων, τα οποία δημιουργήσαμε συμπληρώνοντας τεστ. Έπειτα, αποθηκεύσαμε τα δείγματα (Πίνακας 3) και από τις δυο κλάσεις με σκοπό την εισαγωγή τους στην μηχανή υποστήριξης διανυσμάτων, για να την εκπαιδεύσουμε ώστε να αναγνωρίζει και τις δυο κλάσεις.

Τα Επιλεγμένα από τον χρηστη πρότυπα		Πρότυπα που δεν έχουν επιλεγεί από τον χρήστη
		1
		2
		3
		4
		5
		6
		7
		8
		9
		0

Πίνακας 3 Πρότυπα με τα οποία εκπαιδεύτηκε η μηχανή διανυσμάτων υποστήριξης

Κάθε πρότυπο αποτελείται από μια εικόνα από 30X30 πίξελ. Για να εκπαιδεύσουμε το SVM θα πρέπει τα πρότυπα των εικόνων να τα μετατρέψουμε σε μονοδιάστατες εικόνες 1X900 πίξελ. Οπότε έχουμε μια εικόνα τύπου <Mat> με όνομα training_mat μεγέθους 19X900, αφού έχουμε 19 πρότυπα.

Επίσης, δημιουργήσαμε ακόμα μια εικόνα τύπου <Mat> με όνομα labelsMat, διαστάσεων 19X1 που θα περιέχει σε κάθε θέση της που αντιστοιχεί σε κάθε ένα από τα πρότυπα -1 αν είναι επιλεγμένο πρότυπο και 1 αν δεν είναι όπως φαίνεται και από παρακάτω.

```
Mat labelsMat(19, 1, CV_32FC1, {-1.0,-1.0,-1.0,-1.0,-1.0,-1.0, -1.0,-1.0,-1.0,1.0,1.0,1.0,1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0});
```

Συνοψίζοντας, για την εκπαίδευση του SVM, λοιπόν, θέλουμε δυο πίνακες τύπου <Mat>. Ο πρώτος θα περιέχει τα 19 πρότυπα εικόνων με όνομα training_mat, με τα πρώτα 9 να αντιπροσωπεύουν τα επιλεγμένα πρότυπα και τα υπόλοιπα 10 να αντιπροσωπεύουν αυτά που δεν έχουν επιλεγεί, και έναν δεύτερο πίνακα με όνομα labelsMat ,που για κάθε πρότυπο, θα έχουμε σε κάθε μια από τις 19 θέσεις, αν είναι επιλεγμένο -1, και αν δεν με 1.

Κώδικας C++ για την εκπαίδευση του SVM

```
const int num_files = 19;
       int img_area = 30*30;
       int file_num=0;
       float labels[num_files] = {-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,
-1.0,-1.0,1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0};

      Mat training_mat(num_files,img_area,CV_32FC1);
      Mat labelsMat(num_files, 1, CV_32FC1, labels);

      stringstream str;
      for(int i=1;i<=num_files;i++)
      {
```

```
            str<<"C:\\test_img\\"<<i<<".jpg";
            string fullPath = str.str();
            str.str("");
            Mat img_mat = imread(fullPath,0);

            int ii = 0;

            for (int i = 0; i<img_mat.rows; i++)
            {
                  for (int j = 0; j < img_mat.cols; j++)
                  {
                        training_mat.at<float>(file_num,ii++) =
img_mat.at<uchar>(i,j);
                  }
            }

            file_num++;
      }
      // Set up SVM's parameters
    CvSVMParams params;
    params.svm_type    = CvSVM::C_SVC;
    params.kernel_type = CvSVM::LINEAR;
    params.term_crit   = cvTermCriteria(CV_TERMCRIT_ITER, 100, 1e-6);

    // Train the SVM
    CvSVM SVM;
    SVM.train(training_mat, labelsMat, Mat(), Mat(), params);
      SVM.save("trainingData.dat");
```

Εξήγηση κώδικα εκπαίδευσης SVM:

1. Δημιουργία δεδομένων εκπαίδευσης

   - Τα δεδομένα εκπαίδευσης αποτελούνται από ένα σύνολο 19 εικόνων, οι όποιες είναι επισημασμένες και ανήκουν σε μια από τις δύο διαφορετικές κατηγορίες. Τα πρώτα 9 από τα πρότυπα, είναι επισημασμένα ως επιλεγμένα και τα υπόλοιπα 10 ως μη επιλεγμένα. Η συνάρτηση CvSVM::train απαιτεί τα δεδομένα που θα χρησιμοποιηθούν για την εκπαίδευση, να αποθηκευτούν ως <Mat> αντικείμενα float.:

```
    float labels[num_files] = {-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,
     -1.0,-1.0,1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0};

   Mat training_mat(num_files,img_area,CV_32FC1);
   Mat labelsMat(num_files, 1, CV_32FC1, labels);
```

   - Μετατροπή των εικόνων από δισδιάστατες εικόνες σε μονοδιάστατες. Οι εικόνες που προορίζονται για την εκπαίδευση του SVM, αρχικά διαβάζονται από τον φάκελο που είναι αποθηκευμένες (1) και έπειτα κάθε εικόνα, αφού έχει διαβαστεί, μετατρέπεται σε μονοδιάστατη εικόνα και αποθηκεύεται στον `training_mat(2)`.

```
      stringstream str;
      for(int i=1;i<=num_files;i++)
      {
      (1)   str<<"C:\\test_img\\"<<i<<".jpg";
            string fullPath = str.str();
            str.str("");
            Mat img_mat = imread(fullPath,0);
            int ii = 0;

      (2)   for (int i = 0; i<img_mat.rows; i++)
            {
                  for (int j = 0; j < img_mat.cols; j++)
                  {
                        training_mat.at<float>(file_num,ii++) =
img_mat.at<uchar>(i,j);
                  }
            }

            file_num++;
      }
```

2. Ρύθμιση των παραμέτρων του SVM

Για την εκπαίδευση του SVM, θα πρέπει πρώτα να ρυθμίσουμε κάποιες παραμέτρους. Αυτές οι παράμεροι αποθηκεύονται σε ένα αντικείμενο της κλάσης CvSVMParams.

```
CvSVMParams params;
params.svm_type    = CvSVM::C_SVC;
params.kernel_type = CvSVM::LINEAR;
params.term_crit   = cvTermCriteria(CV_TERMCRIT_ITER, 100, 1e-6);
```

- Τύπος SVM. Επιλέξαμε τον τύπο CvSVM::C_SVC, ο οποίος χρησιμοποιείται για n-κλάσεις ταξινόμηση ($n \geq 2$). Αυτή η παράμετρος ορίζεται στην ιδιότητα CvSVMParams.svm_type.
- Τύπος πυρήνα SVM. Επιλέξαμε CvSVM::LINEAR, αυτό σημαίνει ότι δεν γίνεται χαρτογράφηση. Αυτή η παράμετρος καθορίζεται από την ιδιότητα *CvSVMParams.kernel_type*.
- Κριτήρια τερματισμού του αλγορίθμου. Η διαδικασία εκπαίδευσης του SVM, υλοποιείται επιλύοντας ένα πρόβλημα τετραγωνικής βελτιστοποίησης, κατά επαναλαμβανόμενο τρόπο. Εδώ καθορίσαμε το μέγιστο αριθμό των επαναλήψεων και μια ανοχή σφάλματος, έτσι ώστε να επιτρέπουν στον αλγόριθμο να τελειώσει σε λιγότερο αριθμό των βημάτων, ακόμη και αν το βέλτιστο υπερεπίπεδο δεν έχει υπολογιστεί ακόμη. Αυτή η παράμετρος ορίζεται σε μια δομή cvTermCriteria

3. Εκπαίδευση SVM

Καλούμε τη μέθοδο CvSVM :: train για την κατασκευή του μοντέλου SVM. Ως δεδομένα εισόδου, έχουμε τον training_mat, που περιέχει τις εικόνες που θέλουμε να διαχωρίσουμε και τον labelsMat, που περιέχει τις ετικέτες για κάθε πρότυπο.

```
      CvSVM SVM;
       SVM.train(training_mat, labelsMat, Mat(), Mat(), params);
```

Αφού γίνει η εκπαίδευση του SVM, στο τέλος τα δεδομένα εκπαίδευσης αποθηκεύονται σε ένα αρχείο με όνομα trainingData.dat. Πλέον, το αρχείο αυτό περιέχει όλα τα δεδομένα που χρειάζεται το σύστημα μας για να κάνει τον διαχωρισμό των απαντήσεων, σε επιλεγμένες και μη επιλεγμένες.

```
SVM.save("trainingData.dat");
```

5.2.5 Αποθήκευση των αποτελεσμάτων

Για την αποθήκευση των αποτελεσμάτων χρησιμοποιήθηκε η μορφή του csv, τα comma separated values(csv) είναι αρχεία κειμένου των οποίον οι τιμές σε κάθε γραμμή του αρχείου είναι χωρισμένες με κόμμα. Τα πλεονεκτήματα των csv, είναι η συμβατότητα που έχουν με όλα τα υπολογιστικά φύλλα (OpenOffice,LibreOffice,MSOffice) και η ευκολία εισόδου προς τις γλώσσες προγραμματισμού, αφού όλες υποστηρίζουν είσοδο αρχείων χωρίς να χρειάζεται ειδικές βιβλιοθήκες.

Μόλις η συνάρτηση find_bubble_answers μας επιστρέψει τα αποτελέσματα ,δηλαδή, ένα πινάκα τύπου vector που περιέχει δομές τύπου BubbleAnswers γίνετε έλεγχος αν είναι 400 σε αριθμό αλλιώς τυπώνετε μήνυμα λάθους.

Επόμενο βήμα είναι η ταξινόμηση των bubbles,πρώτα ως προς την συντεταγμένη Y και έπειτα ως προς X. Αυτή η διαδικασία γίνετε γιατί η find_bubble_answers επιστέφει ανακατεμένες τις bubbles με αποτέλεσμα να μην μπορούμε να τις τμηματοποιήσουμε.

Έπειτα, το vector με τις bubble χωρίζετε σε 3 μέρη τα οποία αποθηκεύονται σε ισάριθμα vectors. Το πρώτο μέρος με όνομα Answers αποθηκεύει τις 320 πρώτες bubbles, το δεύτερο με όνομα UserId αποθηκεύει τις 50 bubbles που αφορούν τον αριθμό μητρώου, και το τελευταίο με όνομαTestId αποθηκεύει τις 30 bubbles του κωδικού του τεστ.

Τελευταίο στάδιο είναι η αναγνώριση των απαντήσεων από την SVM. Για την αναγνώριση των απαντήσεων πλέον θέλουμε μόνο το αρχείο trainingData.dat ο κώδικας που χρησιμοποιήθηκε για την εκπαίδευση του SVM δεν περιλαμβάνεται στον κώδικα της εφαρμογής.

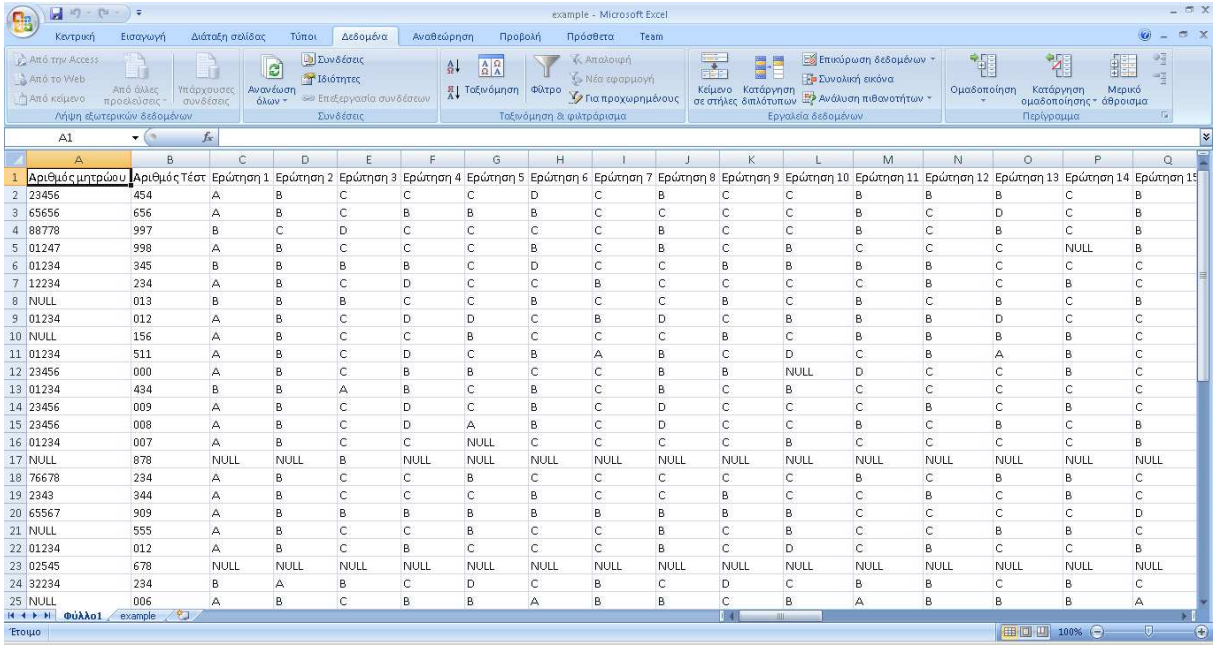
Αρχικά, φορτώνεται το αρχείο trainingData.dat (1), που περιέχει όλες τις απαραίτητες πληροφορίες σχετικά με τον διαχωρισμό των προτύπων. Έπειτα, ο πινάκας img_1D που περιεχει ολετε τις απαντήσεις που έχει δώσει ο φοιτητής μέσω μιας for (2), όπου για κάθε εικόνα μέσω της svm.predict γίνεται πρόβλεψη για το αν η απάντηση που αντιπροσωπεύει κάθε εικόνα είναι επιλεγμένη ή όχι. Αν είναι επιλεγμένη, η μεταβλητή response παίρνει την τιμή -1, αλλιώς 1 και σχεδιάζεται ο αντίστοιχος κύκλος πάνω στην απάντηση, πράσινος για τις επιλεγμένες απαντήσεις και κόκκινος για τις μη επιλεγμένες (3)(4) (Εικόνα 32).

```
CvSVM svm;
(1) svm.load("trainingData.dat");

(2)for(int i=0;i<img_1D.size();i++){
     float response=svm.predict(img_1D[i].Mat_1D);

          if(responce==-1)
      {

(3)circle(preview,img_1D[i].center,img_1D[i].radius,Scalar(0,255,0),2);

     }
else
      {
```

```
(4)circle(preview,img_1D[i].center,img_1D[i].radius,Scalar(0,0,255),2);


      }
}
```

Εικόνα 32 Απόσπασμα απαντήσεων

Τέλος, οι σωστές απαντήσεις, αυτές που έχουν -1(πράσινο χρώμα) στην μεταβλητή response,μετατρέπονται σε A,B,C,D ανάλογα με την απάντηση που έδωσε ο φοιτητής και αποθηκεύονται μαζί με των αριθμό μητρώου και τον αριθμό τεστ στο αρχείο csv. Στον ίδιο φάκελο αποθηκεύονται και όλες οι εικόνες των τεστ με σημειωμένες τις σωστές απαντήσεις(Εικόνα 33).



Αριθμός μητρώου	Αριθμός Τέστ	Ερώτηση 1	Ερώτηση 2	Ερώτηση 3	Ερώτηση 4	Ερώτηση 5	Ερώτηση 6	Ερώτηση 7	Ερώτηση 8	Ερώτηση 9	Ερώτηση 10	Ερώτηση 11	Ερώτηση 12	Ερώτηση 13	Ερώτηση 14	Ερώτηση 15
23456	454	A	B	C	C	C	D	C	B	C	C	B	B	B	C	B
65656	656	A	B	C	B	B	B	C	C	C	C	B	C	D	C	B
88778	997	B	C	D	C	C	C	C	B	C	C	B	C	B	C	B
01247	998	A	B	C	C	C	B	C	B	C	B	C	C	C	NULL	B
01234	345	B	B	B	B	C	D	C	C	B	B	B	B	C	C	C
12234	234	A	B	C	D	C	C	B	C	C	C	C	B	C	B	C
NULL	013	B	B	B	C	C	B	C	C	B	C	B	C	B	C	B
01234	012	A	B	C	D	D	C	B	D	C	B	B	B	D	C	C
NULL	156	A	B	C	C	B	C	C	C	B	C	B	B	B	B	C
01234	511	A	B	C	D	C	B	A	B	C	D	C	B	A	B	C
23456	000	A	B	C	B	B	C	C	B	B	NULL	D	C	C	B	C
01234	434	B	B	A	B	C	B	C	B	C	B	C	C	C	C	C
23456	009	A	B	C	D	C	B	C	D	C	C	C	B	C	B	C
23456	008	A	B	C	D	A	B	C	D	C	C	B	C	B	C	B
01234	007	A	B	C	C	NULL	C	C	C	C	B	C	C	C	C	B
NULL	878	NULL	NULL	B	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
76678	234	A	B	C	C	B	C	C	C	C	C	B	C	B	B	C
2343	344	A	B	C	C	C	B	C	C	B	C	C	B	C	B	C
65567	909	A	B	B	B	B	B	B	B	B	B	B	C	C	C	D
NULL	555	A	B	C	C	B	C	C	B	C	B	C	C	B	B	C
01234	012	A	B	C	B	C	C	C	B	C	D	C	B	C	C	B
02545	678	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
32234	234	B	A	B	C	D	C	B	C	D	C	B	B	C	B	C
NULL	006	A	B	C	B	B	A	B	B	C	B	A	B	B	B	A

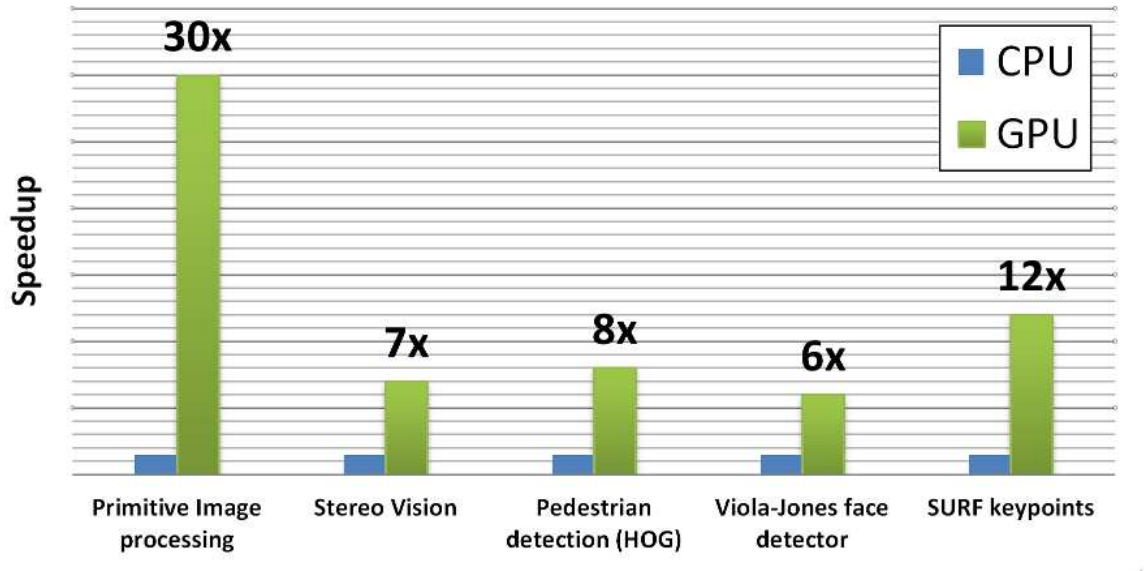
Εικόνα 33 Παράδειγμα μορφής απαντήσεων

6. Αποτελέσματα-Μετρήσεις

Για την επαλήθευση των αποτελεσμάτων δημιουργήθηκε ένα δείγμα διακοσίων τεστ που το καθένα έχει 20 απαντημένες ερωτήσεις, τα οποία σαρωθήκαν από το πολυμηχάνημα HP Officejet 5600 series. Τα τεστ σαρωθήκαν σε 3 διαφορετικές αναλύσεις σε 100dpi,200dpi,300dpi

7. Μελλοντικές επεκτάσεις

Η εργασία έχει πολλές δυνατότητες εξέλιξης και επέκτασης, όπως η μετατροπή του κώδικα της εφαρμογής μας, ώστε να επιταχύνεται από την GPU με τη χρήση της CUDA. Σημαντικό μέρος της υπολογιστικής όρασης, είναι η επεξεργασία εικόνας περιοχή, για την οποία έχουν εφαρμογή οι επιταχυντές γραφικών. Έτσι, είναι πολύ ενδιαφέρον να εφαρμοστούν όλα τα πλεονεκτήματα μιας κάρτας γραφικών για να επιταχύνει την OpenCV.
Η αύξηση στην απόδοση των αλγορίθμων, όπως φαίνεται και στην (Εικόνα 16), είναι πολύ μεγάλη έως και 30X φορές.



Εικόνα 34 Απόδοση της GPU σε σχέση με την CPU για επεξεργασία εικόνας.

Τα τελευταία χρόνια οι online διαδικτυακές υπηρεσίες εξαπλώνονται ραγδαία. Όλοι μας λίγο ή πολύ καθημερινά χρησιμοποιούμε αρκετές από αυτές. Με αυτό το σκεπτικό θα ήταν πολύ ενδιαφέρουσα η μετατροπή της εφαρμογής μας σε δικτυακή υπηρεσία, όπου οι χρήστες θα έχουν την δυνατότητα από οποιανδήποτε υπολογιστή που διαθέτει πρόσβαση στο διαδίκτυο, αφού σαρώσουν τα τεστ να τα ανεβάσουν στον εξυπηρετητή και να διορθωθούν εκεί τοπικά και μετά να σταλούν τα αποτέλεσμα στον χρήστη.

Τέλος, το QRcode, αφενός, χρησιμοποιείται από την εφαρμογή μας για αναγνώριση της κλίσης του τεστ και διόρθωσης αυτής, όπως αναφέραμε και στο κεφάλαιο 5, αφετέρου, η δυνατότητα αποθήκευσης δεδομένων του QRcode μας δίνει πολλά πλεονεκτήματα, όπως η αποθήκευση κρυπτογραφημένων απαντήσεων για αυτόματη διόρθωση, χωρίς δηλαδή ο καθηγητής να χρειάζεται να έχει πρόσβαση στη βάση δεδομένων με τις σωστές απαντήσεις.

Βιβλιογραφία

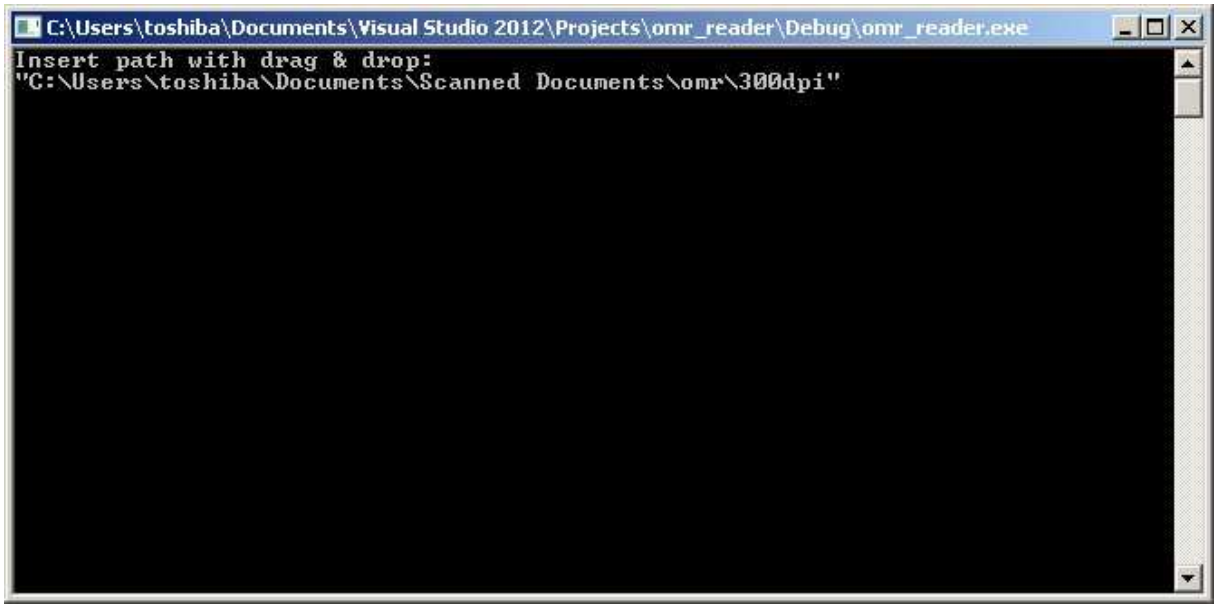
- Accusoft,FormSuite. (n.d.). *accusoft ICR/OCR/OMR Component for .NET 32/64-bit and ActiveX 32-bit*. Ανάκτηση από http://www.accusoft.com/formsuite.htm.
- Akash, E., Tewari, A., & Jain, T. (2012). Computer vision based omr sheet evaluation using OpenCV.
- Bergeron, & Bryan, P. (1998). Optical mark recognition. Postgraduate Medicine online.
- Chidrewar, V., Yang, J., & Moon, D. (2013). Mobile Based Auto Grading Of Answersheets.
- Corinna, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning* , σ. 273.
- Das, S. (2010). *"Optical Mark Recognition Technology for Rural Health Data Collection"*.
- Dillman. (2000). *"Optical Mark Recognition", Presents the Impact of OMR Forms on Response Rates that it is a Relevant Issue"*.
- Furht, B. (2011). QRcode based augmented reality applications. *Handbook of Augmented Reality* , σ. 341.
- Green, & Phil. (2000). Optical Scanning Systems.
- Hussmann, S., & Weiping Deng, P. (2005). *"A High Speed Optical Mark Reader Hardware Implementation at Low Cost using Programmable Logic",*.
- LoPresti, Zvia Segal, F., & Zvia Segal, N. (1996). Using Scanners and OMR Software for Affordable Data Input. Statistics and the Social Sciences. .
- Symons. (2001). *"States MCQ is Generally Recognized as One Question with Some Alternative Answers where One is Correct and the Others are Distracters"*.
- Nobuyuki, O. (1979). A threshold selection method from gray-level histograms. *IEEE Trans. Sys., Man., Cyber. 9 (1)* , σσ. 62–66.
- Wikipedia QR code. (n.d.). QR code
- http://docs.opencv.org/doc/user_guide/user_guide.html
- http://opencv.org/downloads.html

Παράρτημα

Οδηγός χρήσης

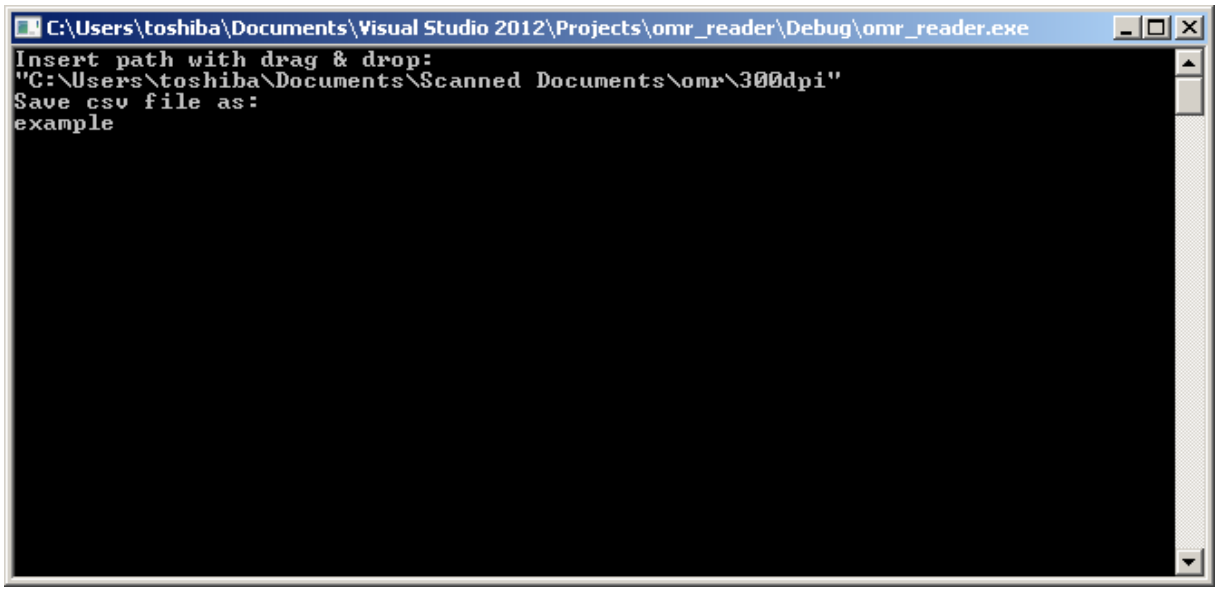
- **Βήμα 1**

Αφού εκτελέσουμε το αρχείο omr_reader.exe σέρνουμε τον φάκελο με τις εικόνες των τεστ πάνω στην κονσόλα και πατάμε enter.



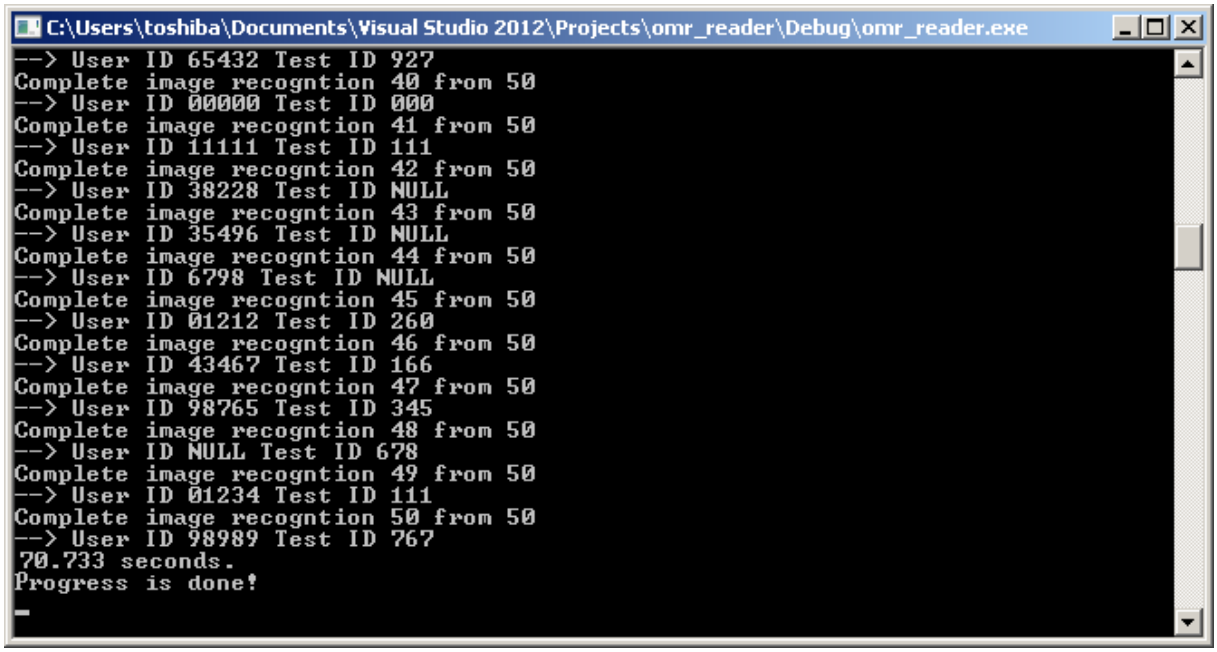
- **Βήμα 2**

Το πρόγραμμα μας ζητάει όνομα για τον φάκελο που θα αποθηκευτούν τα αποτελέσματα δινουμε ένα όνομα και πατάμε enter.



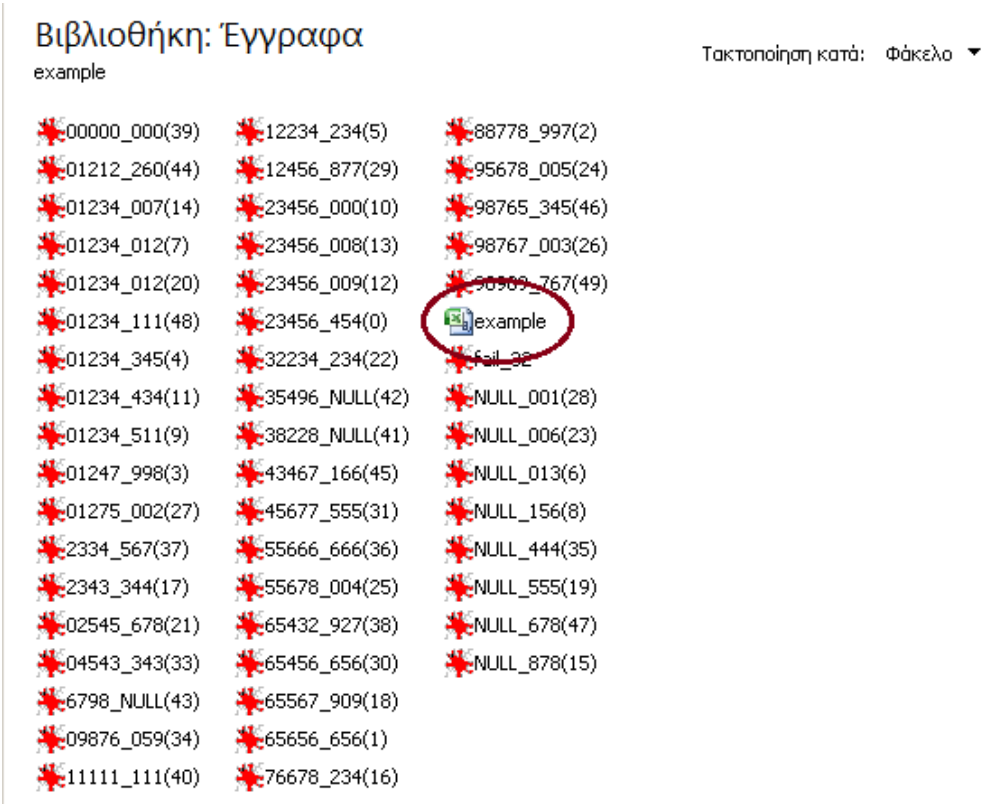
- **Βήμα 3**

Η διαδικασία της διόρθωσης ξεκινά αυτόματα. Μόλις ολοκληρωθεί η διαδικασία τα αποτελέσματα αποθηκεύονται στον ίδιο φάκελο που βρίσκετε το εκτελέσιμο αρχείο της εφαρμογής .



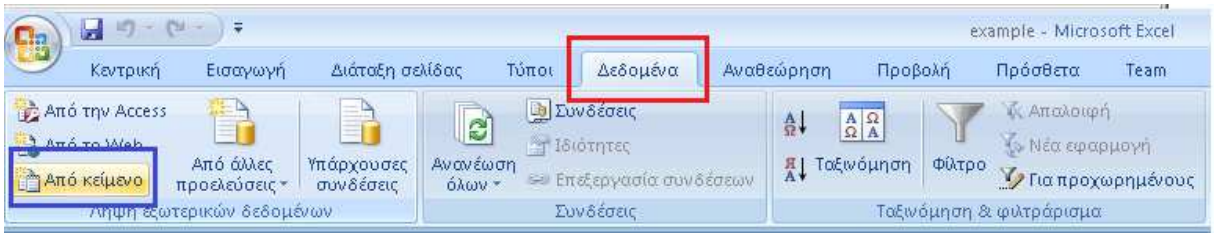
- Βήμα 4

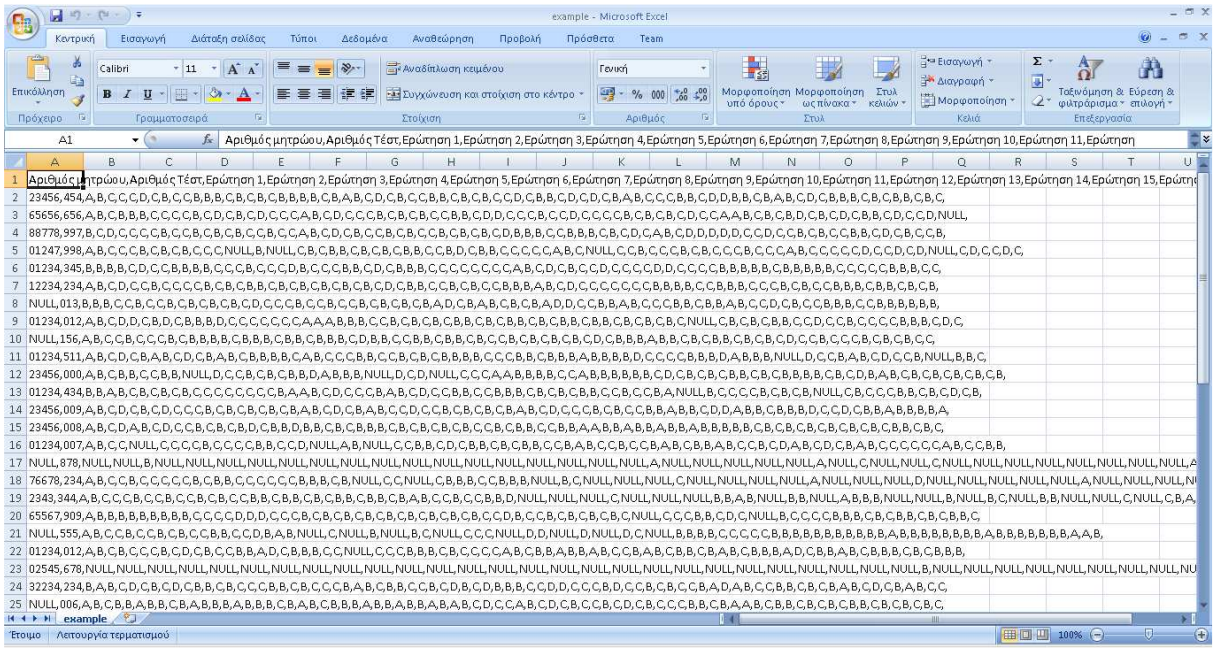
Στον ίδιο φάκελο με τα αποτελέσματα βρίσκονται και οι εικόνες των τεστ, μόλις εντοπίσουμε το αρχείο αποτελεσμάτων το ανοίγουμε.



- Βήμα 5

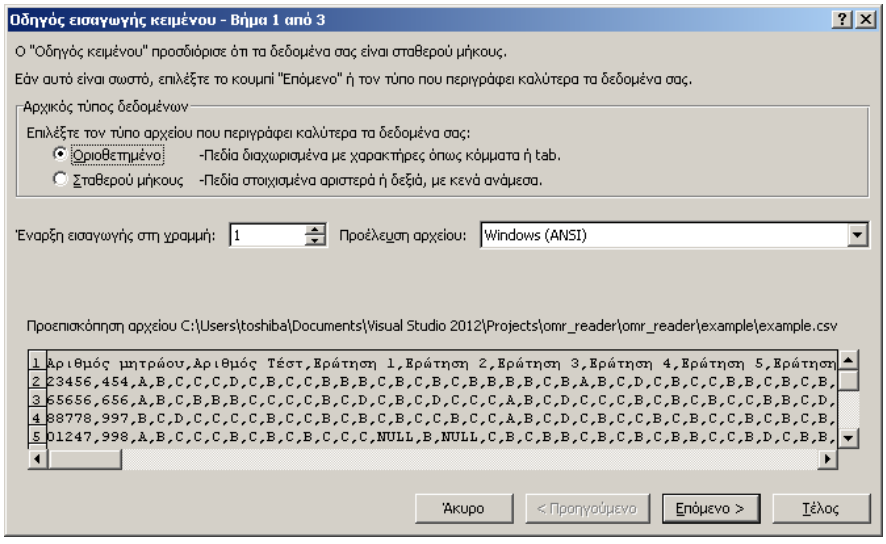
Τα αποτελέσματα δεν είναι ακόμα στοιχισμένα στα κελία για να γίνει αυτό πατάμε την καρτέλα “δεδομένα(κόκκινο χρώμα)” και μετά την επιλογή “από κείμενο(μπλε χρώμα)”





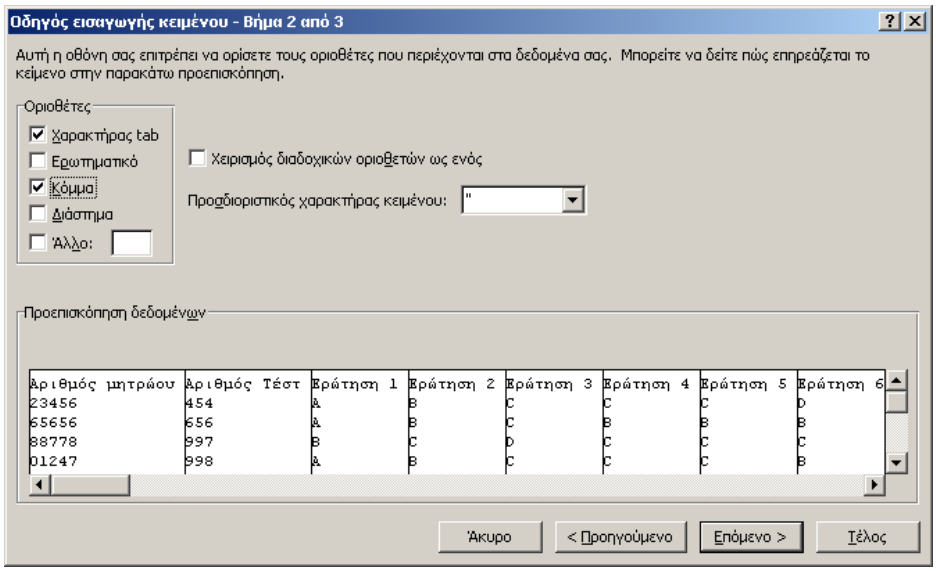
- Βήμα 6

Στο παράθυρο που θα ανοίξει πηγαίνουμε στον φάκελο που είναι το αρχείο csv με τα αποτελέσματα και το επιλεγούμε. Στον οδηγό που θα ανοίξει άμεσος μετά κάνουμε κλικ στην επιλογή “οροθετημένο” και έπειτα “Επόμενο”.



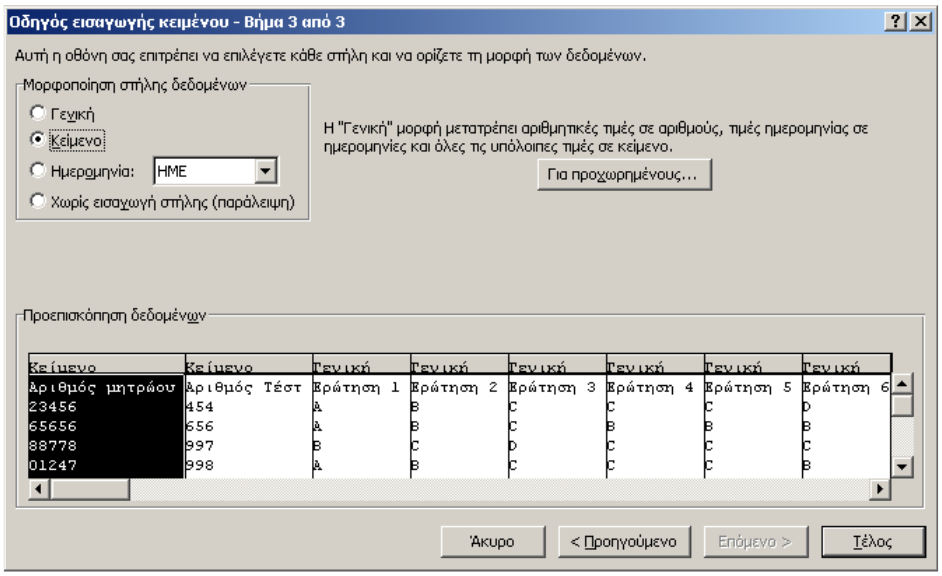
- Βήμα 7

Στο επόμενο βήμα μαρκάρουμε την επιλογή “κόμμα” και πατάμε “Επόμενο”



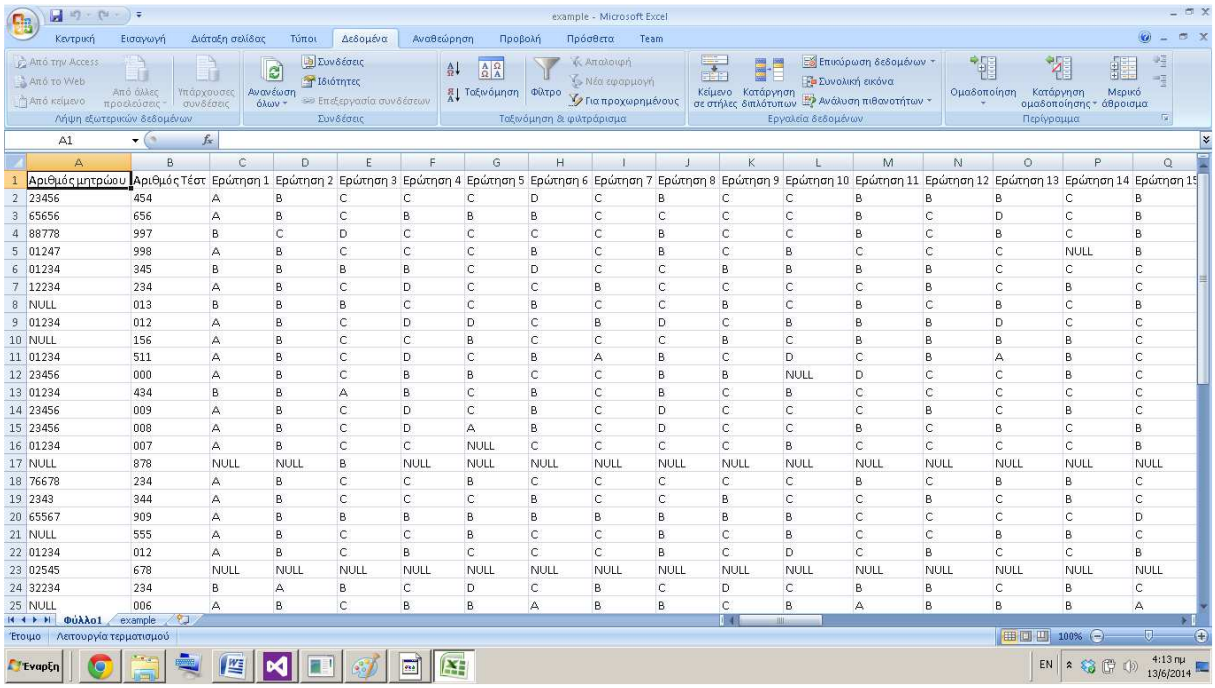
- Βήμα 8

Επιλεγούμε την πρώτη στήλη με όνομα αριθμός μητρώου και κάνουμε κλικ στην επιλογή “κείμενο” την ίδια διαδικασία επαναλαμβάνουμε και για την δεύτερη στήλη με όνομα αριθμός τεστ και πατάμε Τέλος.



- Βήμα 9

Τα δεδομένα είναι έτοιμα για περεταίρω επεξεργασία.



Αριθμός μητρώου	Αριθμός Τέστ	Ερώτηση 1	Ερώτηση 2	Ερώτηση 3	Ερώτηση 4	Ερώτηση 5	Ερώτηση 6	Ερώτηση 7	Ερώτηση 8	Ερώτηση 9	Ερώτηση 10	Ερώτηση 11	Ερώτηση 12	Ερώτηση 13	Ερώτηση 14	Ερώτηση 15
23456	454	A	B	C	C	C	D	C	B	C	C	B	B	B	C	B
65656	656	A	B	C	B	B	B	C	C	C	C	B	C	D	C	B
88778	997	B	C	D	C	C	C	C	B	C	C	B	C	B	C	B
01247	998	A	B	C	C	C	B	C	B	C	B	C	C	C	NULL	B
01234	345	B	B	B	B	C	D	C	C	B	B	B	B	C	C	C
12234	234	A	B	C	D	C	C	B	C	C	C	C	B	C	B	C
NULL	013	B	B	B	C	C	B	C	C	B	C	B	C	B	C	B
01234	012	A	B	C	D	D	C	B	D	C	B	B	B	D	C	C
NULL	156	A	B	C	C	B	C	C	C	B	C	B	B	B	B	C
01234	511	A	B	C	D	C	B	A	B	C	D	C	B	A	B	C
23456	000	A	B	C	B	B	C	C	B	B	NULL	D	C	C	B	C
01234	434	B	B	A	B	C	B	C	B	C	B	C	C	C	C	C
23456	009	A	B	C	D	C	B	C	D	C	C	C	B	C	B	C
23456	008	A	B	C	D	A	B	C	D	C	C	B	C	B	C	B
01234	007	A	B	C	C	NULL	C	C	C	C	B	C	C	C	C	B
NULL	878	NULL	NULL	B	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
76678	234	A	B	C	C	B	C	C	C	C	C	B	C	B	B	C
2343	344	A	B	C	C	C	B	C	C	B	C	C	B	C	B	C
65567	909	A	B	B	B	B	B	B	B	B	B	C	C	C	C	D
NULL	555	A	B	C	C	B	C	C	B	C	B	C	C	B	B	C
01234	012	A	B	C	B	C	C	C	B	C	D	C	B	C	C	B
02545	678	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
32234	234	B	A	B	C	D	C	B	C	D	C	B	B	C	B	C
NULL	006	A	B	C	B	B	A	B	B	C	B	A	B	B	B	A