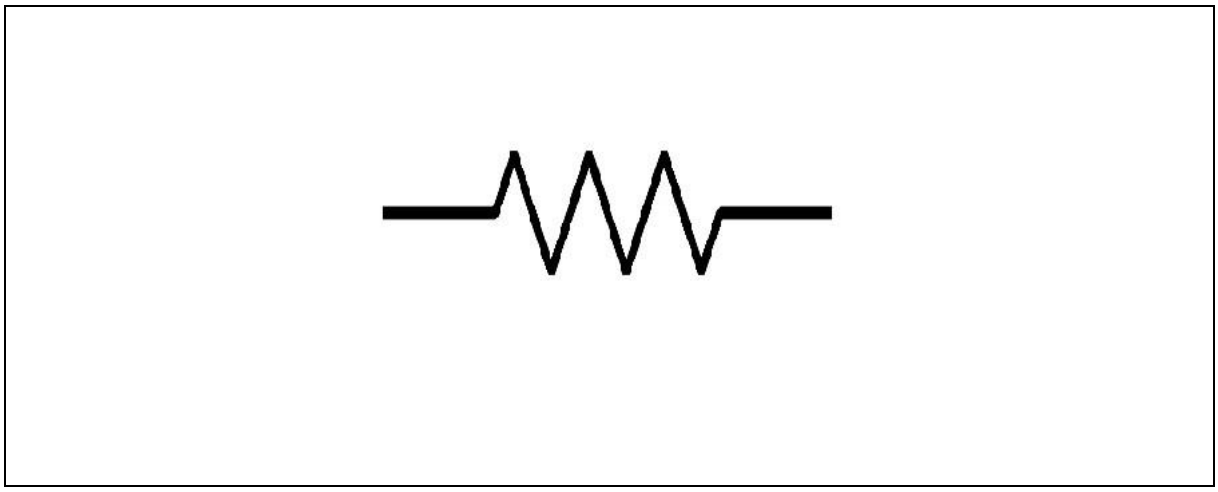
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Αναγνώριση τιμών αντιστάσεων μέσω κάμερας»



Του φοιτητή
Αλτή Παρασκευά
Αρ. Μητρώου: 512003

Επιβλέπων
Ονοματεπώνυμο Τζέκης
Παναγιώτης
Βαθμίδα Καθηγητής

Ημερομηνία 01/09/2025

«Αναγνώριση τιμών αντιστάσεων μέσω κάμερας»
Κωδικός Δ.Ε. 21205
Ονοματεπώνυμο φοιτητή/τών Αλτής Παρασκευάς
Ονοματεπώνυμο εισηγητή
Ημερομηνία ανάληψης Δ.Ε. 30/03/2021
Ημερομηνία περάτωσης Δ.Ε. 01/09/2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Πρόλογος

Η παρούσα εργασία εκπονήθηκε με στόχο την ενασχόλησή μου με τον τομέα της υπολογιστικής όρασης και την εφαρμογή του σε πραγματικά προβλήματα της ηλεκτρονικής. Η επιλογή του συγκεκριμένου θέματος έγινε λόγω του προσωπικού μου ενδιαφέροντος για τα ηλεκτρονικά κυκλώματα και την αυτοματοποίηση διαδικασιών που μέχρι σήμερα βασίζονται σε χειροκίνητες μεθόδους. Η αναγνώριση ηλεκτρονικών εξαρτημάτων, και ειδικότερα αντιστάσεων μέσω της χρωματικής τους κωδικοποίησης, αποτέλεσε μια πρόκληση που συνδύαζε τόσο το θεωρητικό υπόβαθρο των ηλεκτρονικών όσο και τις πρακτικές δεξιότητες στον προγραμματισμό και την ανάλυση εικόνας.

Μέσα από την εκπόνηση της εργασίας αυτής απέκτησα πολύτιμη εμπειρία στη χρήση της γλώσσας Python και των βιβλιοθηκών της (OpenCV, NumPy, Imutils), καθώς και βαθύτερη κατανόηση της διαδικασίας ανάπτυξης και δοκιμής ενός ολοκληρωμένου συστήματος. Επιπλέον, εξοικειώθηκα με ζητήματα συμβατότητας λογισμικού και μεθοδολογίες που διασφαλίζουν την αναπαραγωγιμότητα πειραμάτων. Το σημαντικότερο όφελος υπήρξε η καλλιέργεια δεξιοτήτων έρευνας, πειραματισμού και επίλυσης προβλημάτων. Η εργασία αυτή με βοήθησε να αναγνωρίσω τη σημασία της διεπιστημονικότητας και με ενίσχυσε να συνεχίσω την ενασχόλησή μου με την εφαρμογή της πληροφορικής σε τεχνολογικά αντικείμενα της ηλεκτρονικής.

Περίληψη

Η παρούσα εργασία ασχολείται με την ανάπτυξη ενός συστήματος αναγνώρισης και υπολογισμού της τιμής ηλεκτρονικών αντιστάσεων με βάση τον κώδικα χρωμάτων τους, αξιοποιώντας τεχνικές υπολογιστικής όρασης. Η υλοποίηση πραγματοποιήθηκε στη γλώσσα προγραμματισμού Python 3.6, με χρήση των βιβλιοθηκών OpenCV, NumPy και Imutils. Ως μέσο εισόδου χρησιμοποιήθηκε web camera, η οποία στράφηκε προς σύνολο εικόνων αντιστάσεων που είχαν αποθηκευτεί στον υπολογιστή, ώστε να διασφαλιστεί η επαναληψιμότητα και η συγκρισιμότητα των δοκιμών. Το σύστημα στηρίζεται σε ταξινομητή τύπου Haar Cascade για τον εντοπισμό του σώματος της αντίστασης και σε ανάλυση χρώματος στον χώρο HSV για τον εντοπισμό των χρωματικών λωρίδων. Στη συνέχεια, εφαρμόζονται τεχνικές προεπεξεργασίας, όπως φιλτράρισμα και ανίχνευση περιγραμμάτων, ώστε να απομονωθούν οι λωρίδες και να υπολογιστεί η τιμή της αντίστασης σύμφωνα με τον διεθνή κώδικα χρωμάτων. Τα πειραματικά αποτελέσματα κατέδειξαν ότι το σύστημα είναι σε θέση να αναγνωρίζει με ικανοποιητική ακρίβεια τις χρωματικές λωρίδες και να υπολογίζει σωστά τις τιμές των αντιστάσεων, ιδιαίτερα σε συνθήκες ελεγχόμενου φωτισμού και καθαρών εικόνων. Παράλληλα, αναδείχθηκαν περιορισμοί που σχετίζονται με θόρυβο εικόνας, χαμηλή αντίθεση ή αντανακλάσεις, καθώς και ζητήματα συμβατότητας λογισμικού λόγω εξάρτησης από συγκεκριμένες εκδόσεις Python και βιβλιοθηκών. Η εργασία απέδειξε ότι η χρήση τεχνικών υπολογιστικής όρασης μπορεί να οδηγήσει σε ένα λειτουργικό πρωτότυπο αναγνώρισης ηλεκτρονικών εξαρτημάτων. Το προτεινόμενο σύστημα μπορεί να αποτελέσει τη βάση για περαιτέρω βελτιώσεις και επεκτάσεις με σκοπό την ανάπτυξη πιο ανθεκτικών και γενικευμένων εφαρμογών στον τομέα της εκπαίδευσης και της ηλεκτρονικής.

"Recognition of resistor values through camera"

«Altis Paraskevas»

Abstract

This thesis focuses on the development of a system for the recognition and calculation of electronic resistor values based on their color code, by employing computer vision techniques. The implementation was carried out in Python 3.6, utilizing the OpenCV, NumPy, and Imutils libraries. As an input device, a web camera was employed, which was directed towards a dataset of resistor images stored on the computer. This approach ensured repeatability and comparability of the experimental tests. The system relies on a Haar Cascade classifier for detecting the resistor body and on color analysis in the HSV space for identifying the color bands. Subsequently, preprocessing techniques, such as bilateral filtering and contour detection, were applied to isolate the bands and compute the resistance value according to the international resistor color code. The experimental results demonstrated that the system is capable of recognizing the color bands with satisfactory accuracy and of calculating correct resistance values, especially under controlled lighting conditions and with clear input images. At the same time, certain limitations were identified, related to image noise, low contrast, and reflections, as well as software compatibility issues caused by the dependency on specific Python and library versions. Overall, the thesis showed that computer vision techniques can lead to a functional prototype for electronic component recognition. The proposed system may serve as a basis for further improvements and extensions, aiming at the development of more robust and generalized applications in the field of electronics and education.

Περιεχόμενα

Κατάλογος Εικόνων

Κεφάλαιο 1ο: ΕΙΣΑΓΩΓΗ

1.1 Τεχνητή Νοημοσύνη & Αναγνώριση Εικόνας

Η Τεχνητή Νοημοσύνη (ΤΝ) ορίζεται κλασικά ως ο κλάδος της πληροφορικής που ασχολείται με τον σχεδιασμό και τη δημιουργία ευφυών συστημάτων, ικανών να αντιλαμβάνονται το περιβάλλον τους και να λαμβάνουν αποφάσεις ή να εκτελούν ενέργειες που, αν γίνονταν από ανθρώπους, θα απαιτούσαν νοημοσύνη. Ο όρος *Artificial Intelligence* πρωτοχρησιμοποιήθηκε το 1956 στο συνέδριο του Dartmouth, και ο Marvin Minsky την περιέγραψε ως «την επιστήμη κατασκευής μηχανών που κάνουν πράγματα που θα απαιτούσαν ευφυΐα αν γίνονταν από ανθρώπους» [1] . Με άλλα λόγια, η ΤΝ επιχειρεί να προσομοιώσει γνωστικές λειτουργίες του ανθρώπου μέσω υπολογιστικών μοντέλων. Μέσα στο ευρύ αυτό πεδίο, η Μηχανική Μάθηση (Machine Learning, ML) αποτελεί ένα υποσύνολο της ΤΝ που επικεντρώνεται στην ανάπτυξη αλγορίθμων [2] οι οποίοι επιτρέπουν σε υπολογιστικά συστήματα να μαθαίνουν από δεδομένα και να βελτιώνονται αυτόνομα, αντί να προγραμματίζονται ρητά για κάθε εργασία. Για παράδειγμα, εργασίες όπως η αναγνώριση εικόνων, η αναγνώριση ομιλίας ή η κατανόηση φυσικής γλώσσας μπορούν να αντιμετωπιστούν με μεθόδους ML αντί με παραδοσιακό προγραμματισμό κανόνων. Ειδικότερα, η Βαθιά Μάθηση (Deep Learning, DL) [3] είναι μια κατηγορία μεθόδων μηχανικής μάθησης που βασίζονται σε τεχνητά νευρωνικά δίκτυα πολλαπλών επιπέδων – εξ ου και ο χαρακτηρισμός “βαθιά”. Τα βαθιά νευρωνικά δίκτυα αποτελούνται από πολλά διαδοχικά στρώματα επεξεργασίας/νευρώνων, επιτρέποντας στο σύστημα να ανακαλύπτει αυτόματα σύνθετες χαρακτηριστικές αναπαραστάσεις των δεδομένων χωρίς ανθρώπινη παρέμβαση. Αυτή η αυτόματη εξαγωγή χαρακτηριστικών καθιστά τη βαθιά μάθηση εξαιρετικά ισχυρή για απαιτητικές εργασίες όπως η αναγνώριση εικόνων και ομιλίας. Με άλλα λόγια, αντί να δίνουμε εμείς στο σύστημα κανόνες για να αναγνωρίσει, π.χ., αν μια εικόνα περιέχει ένα αντικείμενο, το ίδιο το νευρωνικό δίκτυο μαθαίνει από τα δεδομένα ποια χαρακτηριστικά της εικόνας είναι σημαντικά.

Στο πλαίσιο αυτό, η Αναγνώριση Εικόνας (Image Recognition) [4] αποτελεί έναν από τους σημαντικότερους τομείς εφαρμογής της ΤΝ και ειδικότερα της μηχανικής μάθησης. Η ικανότητα ενός υπολογιστή να “βλέπει” και να κατανοεί το οπτικό περιβάλλον είναι θεμελιώδης δεξιότητα της μηχανικής αντίληψης. Συγκεκριμένα, η Υπολογιστική Όραση (Computer Vision) [5] είναι ο κλάδος της ΤΝ που επιδιώκει να δώσει σε μηχανές την ικανότητα να ερμηνεύουν και να κατανοούν δεδομένα εικόνας και βίντεο – ουσιαστικά να αποκτήσουν “οπτική αντίληψη” παρόμοια με του ανθρώπου. Η μηχανική μάθηση αποτελεί τη βάση της μηχανικής αντίληψης και της υπολογιστικής όρασης, καθώς παρέχει τους αλγορίθμους για να μάθουν οι υπολογιστές από τα οπτικά δεδομένα. Ένα σύστημα υπολογιστικής όρασης μπορεί

να εκτελεί πληθώρα εργασιών, από την απλή ταξινόμηση μιας εικόνας (αναγνώριση του κύριου αντικειμένου που περιέχει) μέχρι τον εντοπισμό πολλαπλών αντικειμένων σε σκηνές, την ανάλυση ιατρικών σαρώσεων ή την πλοήγηση ενός ρομπότ στο χώρο [35]. Η αναγνώριση εικόνας –δηλαδή η ταυτοποίηση και ταξινόμηση αντικειμένων ή χαρακτηριστικών μέσα σε ψηφιακές εικόνες– εντάσσεται σε αυτό το πλαίσιο ως μια βασική λειτουργία: επιτρέπει σε έναν «ευφυή πράκτορα» να αντιληφθεί οπτικά το περιβάλλον του αναγνωρίζοντας πρόσωπα, αντικείμενα, σήματα, σκηνές κ.ά., κι έτσι να λάβει αποφάσεις ή δράσεις βάσει αυτής της αντίληψης.

Η εξέλιξη της αναγνώρισης εικόνας χαρακτηρίζεται από μια σαφή μετατόπιση μεθοδολογίας τις τελευταίες δεκαετίες. Αρχικά, οι προσεγγίσεις στην υπολογιστική όραση βασίζονταν σε παραδοσιακούς αλγορίθμους και χειροποίητα χαρακτηριστικά [6]. Προγραμματιστές και μηχανικοί όρασης σχεδίαζαν αλγορίθμους που ανίχνευαν απλές ιδιότητες στις εικόνες – όπως ακμές, γωνίες, υφές ή γεωμετρικά σχήματα – και στη συνέχεια χρησιμοποιούσαν κλασικούς ταξινομητές (π.χ. SVM, Random Forests) [36] για να αναγνωρίσουν πρότυπα βάσει αυτών των χαρακτηριστικών. Αυτή η *feature-based* προσέγγιση απαιτούσε σημαντική ανθρώπινη εξειδίκευση· οι αλγόριθμοι ήταν συχνά ευαίσθητοι σε μεταβολές στις συνθήκες (φωτισμός, οπτική γωνία κ.λπ.) και παρουσίαζαν περιορισμένη ικανότητα γενίκευσης πέρα από τα σενάρια για τα οποία είχαν ρυθμιστεί. Η μεγάλη επανάσταση ήρθε με την άνοδο των Συνελικτικών Νευρωνικών Δικτύων (Convolutional Neural Networks, CNNs). Τα CNNs επέτρεψαν την αυτόματη εκμάθηση ιεραρχικών χαρακτηριστικών απευθείας από τα δεδομένα εικόνας, εξαλείφοντας σε μεγάλο βαθμό την ανάγκη για χειροκίνητη εξαγωγή χαρακτηριστικών [7]. Στις αρχές της δεκαετίας του 2010, η σύζευξη βελτιωμένων αλγορίθμων νευρωνικών δικτύων με την αυξανόμενη υπολογιστική ισχύ (ιδίως μέσω GPUs) και την ύπαρξη μεγάλων συλλογών δεδομένων οδήγησε σε άλμα ακρίβειας στην αναγνώριση εικόνων. Χαρακτηριστικά, το 2012 ένα βαθύ συνελικτικό δίκτυο (CNN) [8] πέτυχε εντυπωσιακά καλύτερα αποτελέσματα από κάθε προηγούμενη μέθοδο στο διαγωνισμό ImageNet, σηματοδοτώντας την απαρχή της σύγχρονης εποχής του deep learning στην όραση. Έκτοτε, τα CNN εδραιώθηκαν ως η κυρίαρχη προσέγγιση στην πλειονότητα των προβλημάτων αναγνώρισης εικόνας, επιτυγχάνοντας συνεχώς βελτιούμενες επιδόσεις.

Τα τελευταία χρόνια, βρισκόμαστε αντιμέτωποι με μια νέα αλλαγή παραδείγματος: την εμφάνιση των Μετασχηματιστών Όρασης (Vision Transformers, ViTs) [9]. Οι μετασχηματιστές προέρχονται από τον χώρο της επεξεργασίας φυσικής γλώσσας, αλλά το 2020 προτάθηκε η προσαρμογή τους στην όραση υπολογιστή. Ένα μοντέλο Vision Transformer χωρίζει την εικόνα σε επιμέρους patches και τα επεξεργάζεται με μηχανισμούς αυτο-προσοχής (*self-attention*) αντί για συνελίξεις. Οι ViTs έχουν επιτύχει αντίστοιχη ή και ανώτερη απόδοση [37] σε σχέση με τα αντίστοιχα δίκτυα συνελίξεων σε διάφορα σύνολα δεδομένων ταξινόμησης

εικόνων, παρουσιάζοντας ένα εναλλακτικό μοντέλο μάθησης για προβλήματα όρασης. Η αυξανόμενη ερευνητική προσοχή στους μετασχηματιστές όρασης αντανακλά την υπόσχεσή τους ως μια νέα ανταγωνιστική αρχιτεκτονική, ενώ τα CNNs παραμένουν το καθιερωμένο θεμέλιο στις εφαρμογές όρασης. Συνοψίζοντας, η αναγνώριση εικόνας –ως πεδίο εντός της ΤΝ– έχει μεταβεί από *στατικούς, rule-based αλγορίθμους*, σε δυναμικά μοντέλα μάθησης (CNNs) που μαθαίνουν από τα δεδομένα [38], και πλέον διευρύνεται με υβριδικές ή νέες αρχιτεκτονικές όπως οι μετασχηματιστές, εντάσσοντας ακόμα πιο εξελιγμένες μεθόδους για την κατανόηση οπτικών πληροφοριών.

1.2 Ιστορική Αναδρομή Αναγνώρισης Εικόνας

Η εξέλιξη της αναγνώρισης εικόνας κατά τις τελευταίες δεκαετίες περιλαμβάνει μια σειρά από σταθμούς-κλειδιά, που συνδέονται τόσο με θεωρητικές προόδους όσο και με τεχνολογικές εξελίξεις [10]. Παρακάτω παρουσιάζεται μια χρονολογική αναδρομή των βασικών εξελίξεων, από τις απαρχές των νευρωνικών δικτύων μέχρι τις σύγχρονες deep learning αρχιτεκτονικές και τα σημαντικά σύνολα δεδομένων / διαγωνισμούς που ώθησαν την έρευνα προς τα εμπρός.

1950–1970: Θεμελίωση και πρώιμα νευρωνικά δίκτυα

Οι πρώτες απόπειρες για μηχανική αναγνώριση προτύπων ανάγονται στα μέσα του 20ού αιώνα. Το Perceptron του Frank Rosenblatt, που προτάθηκε το 1958, υπήρξε το πρώτο απλό νευρωνικό δίκτυο για ταξινόμηση προτύπων. Ο Rosenblatt ενέπνευσε από τη βιολογία το μοντέλο του τεχνητού νευρώνα και κατασκεύασε μια μηχανή (Mark I Perceptron) [11] ικανή να μαθαίνει να αναγνωρίζει απλά οπτικά σχήματα. Αν και το perceptron έδειξε τη δυναμική των μαθησιακών αλγορίθμων, είχε σημαντικούς περιορισμούς: ως μονοστρώματη αρχιτεκτονική δεν μπορούσε να μάθει προβλήματα που δεν είναι γραμμικά διαχωρίσιμα (διάσημο παράδειγμα το XOR). Το 1969, το βιβλίο “Perceptrons” των Marvin Minsky και Seymour Papert ανέδειξε θεωρητικά αυτούς τους περιορισμούς, οδηγώντας σε κάμψη του ενδιαφέροντος για τα νευρωνικά δίκτυα στα επόμενα χρόνια (γνωστή και ως πρώτη “AI winter”). Παράλληλα, τη δεκαετία του 1960 αναπτύχθηκαν κάποιες πρώτες πολυστρωματικές προσεγγίσεις (π.χ. το ADALINE του Widrow & Hoff) [12], αλλά η εκπαίδευσή τους παρέμενε δύσκολη χωρίς αποτελεσματική μέθοδο προσαρμογής βαρών.

1980–1990: Επανανακάλυψη της Μάθησης και πολυστρωματικά δίκτυα.

Η δεκαετία του 1980 σηματοδοτεί μια αναζωπύρωση του χώρου, κυρίως χάρη στην ανακάλυψη/διάδοση του αλγορίθμου της Οπισθοδιάδοσης σφάλματος (Backpropagation) [13]. Αν και οι βάσεις του backpropagation είχαν τεθεί θεωρητικά νωρίτερα [39], στα μέσα των ‘80s παρουσιάστηκε ως ένας πρακτικός τρόπος εκπαίδευσης πολυστρωματικών νευρωνικών δικτύων. Η οπισθοδιάδοση επέτρεψε την αποτελεσματική προσαρμογή των βαρών σε δίκτυα με κρυφά στρώματα, ξεπερνώντας τον περιορισμό του μονοστρώματου perceptron. Αυτή η εξέλιξη αναζωογόνησε την έρευνα στα νευρωνικά δίκτυα (Δεύτερη Άνοιξη της ΤΝ), ωστόσο τα βαθύτερα δίκτυα εξακολουθούσαν να αντιμετωπίζουν προβλήματα όπως η εξαφάνιση του βαθμίωτου (vanishing gradients) [40] και η υπερεκπαίδευση (overfitting) [41] λόγω ανεπαρκών δεδομένων και υπολογιστικών πόρων. Στα τέλη των ‘80s και αρχές ‘90s αναπτύχθηκαν επιτυχώς νευρωνικά δίκτυα για συγκεκριμένες εφαρμογές, όπως τα πρώτα συστήματα οπτικής αναγνώρισης χαρακτήρων (OCR) [14] και αναγνώρισης ομιλίας, όμως γενικά η υπολογιστική όραση στον ευρύτερο χώρο παρέμενε κυρίως προσανατολισμένη σε κλασικές μεθόδους επεξεργασίας εικόνας (φίλτρα, μορφολογικές πράξεις, ανίχνευση ακμών) και μη-νευρωνικούς ταξινομητές.

1990–2000: Συνελικτικά Δίκτυα και χειρόγραφοι χαρακτήρες (LeNet, MNIST).

Ένα ορόσημο στην αναγνώριση εικόνων υπήρξε η εργασία του Yann LeCun πάνω στα Συνελικτικά Νευρωνικά Δίκτυα (CNNs). Ο LeCun εισήγαγε την ιδέα ότι εφαρμόζοντας συνελίξεις (convolutions) πάνω σε εικόνες μπορούμε να αντλήσουμε τοπικά χαρακτηριστικά και να μειώσουμε τις διαστάσεις των δεδομένων, εκμεταλλευόμενοι την τοπολογική δομή της εικόνας. Το αποκορύφωμα αυτής της γραμμής έρευνας ήταν το LeNet-5 (1998), ένα CNN σχεδιασμένο για αναγνώριση χειρόγραφων ψηφίων. Το LeNet-5 [15] εκπαιδεύτηκε στο σύνολο δεδομένων MNIST [42] – μια συλλογή ~70.000 δειγμάτων ψηφίων 0–9 – και πέτυχε αξιοσημείωτα υψηλή ακρίβεια, καθιστώντας τον εαυτό του πρωτοπόρο για εφαρμογές αναγνώρισης χαρακτήρων (π.χ. σε ταχυδρομικούς κώδικες επιταγών). Η αρχιτεκτονική LeNet (με διαδοχικές συνελίξεις, pooling και πλήρως συνδεδεμένα στρώματα) θεμελίωσε πολλά από τα δομικά στοιχεία που χρησιμοποιούν τα σύγχρονα CNN. Μάλιστα, σε αυτό το περιορισμένο πεδίο του MNIST, οι βέλτιστες επιδόσεις πλησίασαν την ανθρώπινη ακρίβεια – σφάλμα κάτω από 0,3% στην ταξινόμηση ψηφίων. Παρά αυτές τις επιτυχίες, γενικότερα τα νευρωνικά δίκτυα στα ‘90s δεν κυριάρχησαν σε ευρύτερες εφαρμογές υπολογιστικής όρασης. Οι διαθέσιες βάσεις δεδομένων εικόνων ήταν ακόμη σχετικά μικρές (λίγες δεκάδες χιλιάδες δείγματα) και το υλικό (CPU) αργό για την εκπαίδευση μεγαλύτερων δικτύων, γεγονός που περιόριζε τις δυνατότητες των deep learning προσεγγίσεων εκείνη την περίοδο.

2010–2012: "Μεγάλα Δεδομένα" και το Άλμα του AlexNet (ImageNet).

Η πραγματική έκρηξη στην αναγνώριση εικόνων συνέβη τη δεκαετία του 2010, χάρη σε δύο συγκλίνοντες παράγοντες: (1) τη διαθεσιμότητα τεράστιων συνόλων δεδομένων εικόνων και (2) τη δραματική αύξηση της υπολογιστικής ισχύος (κυρίως μέσω μονάδων GPU). Έως το τέλος της δεκαετίας 2000, η έλλειψη μεγάλων βάσεων δεδομένων αποτελούσε τροχοπέδη – τα τυπικά datasets είχαν τάξη μεγέθους 10^4 εικόνων. Αυτό όμως άλλαξε με πρωτοβουλίες όπως το ImageNet. Το ImageNet [16], που καθοδηγήθηκε από τη Fei-Fei Li (Stanford) και παρουσιάστηκε το 2009, συγκέντρωσε άνω των 15 εκατομμυρίων εικόνων υψηλής ανάλυσης ταξινομημένες σε περίπου 22.000 κατηγορίες αντικειμένων. Ήταν ένα τεράστιο άλμα σε κλίμακα δεδομένων: για πρώτη φορά, οι ερευνητές είχαν στη διάθεσή τους ένα τόσο πλούσιο και ποικίλο corpus για να εκπαιδεύσουν βαθιά νευρωνικά δίκτυα. Παράλληλα, οι Graphics Processing Units (GPUs) – αρχικά σχεδιασμένες για γραφικά – άρχισαν να χρησιμοποιούνται για επιτάχυνση αλγορίθμων μηχανικής μάθησης [43]. Οι GPUs επέτρεψαν τον παραλληλισμό των μαθηματικών πράξεων των νευρωνικών δικτύων, μειώνοντας δραστικά τον χρόνο εκπαίδευσης: όπως αναφέρει σχετικό άρθρο, οι γρήγορες GPU της NVIDIA έκαναν δυνατή την εκπαίδευση δικτύων 10-20 φορές ταχύτερα απ' ό,τι πριν, μετατρέποντας δουλειά εβδομάδων σε ημέρες. Το 2012 όλα αυτά συνέτειναν στο περίφημο επίτευγμα του AlexNet. Ο Alex Krizhevsky, υπό την επίβλεψη του Geoffrey Hinton, παρουσίασε ένα βαθύ συνελικτικό δίκτυο (8 συνελικτικά/συνδεδεμένα στρώματα, ~60 εκατομμύρια παράμετροι) που συμμετείχε στον διαγωνισμό ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2012) [17]. Το AlexNet κατέκτησε την πρώτη θέση με ποσοστό σφάλματος Top-5 ~15.3%, έναντι ~26% του δεύτερου καλύτερου μοντέλου εκείνης της χρονιάς. Η διαφορά πάνω από 10 ποσοστιαίες μονάδες ήταν πρωτοφανής. Σημαντικό είναι ότι οι ανταγωνιστές του AlexNet ήταν παραδοσιακά συστήματα όρασης με χειροποίητα χαρακτηριστικά – το γεγονός ότι ένα πλήρως αυτόματο, *data-driven* μοντέλο τα συνέτριψε προκάλεσε τεράστια εντύπωση στην κοινότητα. Οι Krizhevsky et al. πέτυχαν αυτήν την απόδοση *χωρίς* να σχεδιάσουν ειδικούς ανιχνευτές χαρακτηριστικών για κάθε κατηγορία εικόνας· αντίθετα, το δίκτυό τους "έμαθε να βλέπει" από μόνο του, εκπαιδευόμενο σε ~1 εκατομμύριο εικόνες του ImageNet και πραγματοποιώντας *δισεκατομμύρια* πράξεις μέσω GPU. Το αποτέλεσμα αυτό συχνά χαρακτηρίζεται ως το σημείο καμπής που σηματοδότησε την "έκρηξη" του Deep Learning – απέδειξε εμπράκτως ότι οι μέθοδοι βαθιάς μάθησης μπορούν να ξεπεράσουν κατά πολύ τις συμβατικές προσεγγίσεις στην όραση υπολογιστή.

2013–2015: Βαθύτερα Δίκτυα και υπεροχή των Deep CNNs (VGG, GoogLeNet, ResNet).

Μετά τον θρίαμβο του AlexNet, ακολούθησε μια περίοδος ραγδαίας προόδου όπου κάθε χρόνο προτείνονταν ακόμα βαθύτερες και αποδοτικότερες αρχιτεκτονικές συνελικτικών δικτύων, ρίχνοντας ολοένα το σφάλμα στα benchmark σύνολα δεδομένων. Το 2014 εμφανίστηκε το VGGNet (Simonyan & Zisserman) [18], μια απλή αλλά βαθιά αρχιτεκτονική με 16–19 επίπεδα συνελίξεων, η οποία έδειξε ότι "το βαθύτερο είναι καλύτερο", αυξάνοντας σημαντικά το βάθος του δικτύου, πέτυχε top-5 σφάλμα ~7.3% στο ImageNet, σχεδόν στο μισό του AlexNet. Την ίδια χρονιά, η ομάδα της Google παρουσίασε το GoogLeNet (Inception v1) [19], εισάγοντας την έννοια των *Inception modules*: αντί για μια απλή ακολουθία στρωμάτων, το GoogLeNet χρησιμοποίησε παράλληλες διεργασίες συνελίξεων πολλαπλών μεγεθών στο ίδιο επίπεδο, εξασφαλίζοντας πλούσια χαρακτηριστικά με αποδοτικό υπολογιστικό κόστος. Το GoogLeNet είχε βάθος 22 στρωμάτων αλλά πολύ λιγότερες παραμέτρους από το VGG, κερδίζοντας το ILSVRC-2014 [44] με περαιτέρω μειωμένο σφάλμα. Σημαντική ήταν και η καινοτομία του να ενσωματώσει "global average pooling" αντί για πλήρως συνδεδεμένα στρώματα στο τέλος, μειώνοντας δραστικά τις παραμέτρους.

Το 2015 ήρθε ένα νέο άλμα με τα Residual Networks (ResNets) [20] των He et al. Οι ResNets εισήγαγαν τις *residual connections* (βραχυκυκλωμένες συνδέσεις), επιτρέποντας την παράκαμψη πληροφοριών ανάμεσα σε στρώματα. Με αυτήν την τεχνική μπόρεσαν να εκπαιδεύσουν εξαιρετικά βαθιά δίκτυα – έως και 152 επίπεδα – χωρίς το φαινόμενο υποβάθμισης της απόδοσης που παρατηρείται συνήθως σε πολύ μεγάλα βάθη. Το ResNet-152 σημείωσε νέο ρεκόρ στο ImageNet (top-5 error ~3.6%, που αντιστοιχεί σε άνω του 96% ακρίβεια) και ουσιαστικά υπερέβη την ανθρώπινη επίδοση στην ταξινόμηση ImageNet. Πράγματι, μέχρι το 2015 οι καλύτεροι αλγόριθμοι βαθιάς μάθησης είχαν φτάσει και ξεπεράσει την ακρίβεια ενός ανθρώπου στις 1000 κατηγορίες του ImageNet, ένα ορόσημο που σηματοδοτεί την ωριμότητα της τεχνολογίας. Εκτός από το ResNet, το 2015-2016 είδαν το φως και άλλα καινοτόμα μοντέλα: π.χ. το DenseNet (2017) [45], που εισήγαγε "πυκνές συνδέσεις" (*dense connections*) συνδέοντας κάθε επίπεδο με όλα τα επόμενα, ενισχύοντας την επαναχρησιμοποίηση χαρακτηριστικών. Τα διαρκώς βελτιωμένα αυτά δίκτυα οδήγησαν στην κυριαρχία των CNNs σε κάθε είδους πρόβλημα vision (ταξινόμηση, ανίχνευση αντικειμένων, σημειακό εντοπισμό χαρακτηριστικών κ.ά.).

2016–2020: Περαιτέρω εξέλιξη – Transfer Learning, EfficientNet, και Transformers.

Καθώς τα βαθιά δίκτυα πέτυχαν εξαιρετικές επιδόσεις, η έμφαση μετατοπίστηκε σε ζητήματα όπως η μεταφορά μάθησης και η αποδοτικότητα. Η ιδέα του Transfer Learning [21] άρχισε να γίνεται δημοφιλής ήδη από τα μέσα της δεκαετίας του 2010: αντί να εκπαιδεύεται κάθε νευρωνικό δίκτυο από μηδενική βάση, χρησιμοποιούνται προ-εκπαιδευμένα δίκτυα σε τεράστια σύνολα (όπως το ImageNet) ως αφετηρία, και κατόπιν γίνονται fine-tuning (επαναπροσαρμογή βαρών) σε μικρότερα σύνολα για συγκεκριμένες εφαρμογές [46]. Αυτό βρέθηκε εξαιρετικά αποδοτικό, με το fine tuning, ακόμη και μεσαίου μεγέθους datasets μπορούν να έχουν πολύ καλά αποτελέσματα, ενώ μειώνεται δραστικά ο απαιτούμενος χρόνος εκπαίδευσης. Παράλληλα, προτάθηκαν νέες αρχιτεκτονικές εστιασμένες στην *αποδοτικότητα*: το EfficientNet [22] (Tan & Le, 2019) χρησιμοποίησε ένα αυτοματοποιημένο μηχανισμό αναζήτησης αρχιτεκτονικών σε συνδυασμό με ομοιοθετική κλιμάκωση (compound scaling) για να βρει μια οικογένεια μοντέλων που πετυχαίνουν κορυφαία ακρίβεια με πολύ λιγότερες παραμέτρους. Η σειρά EfficientNet έδειξε ότι με προσεκτικό σχεδιασμό είναι εφικτό να φτιαχτούν "ελαφριά" αλλά ισχυρά δίκτυα, κάτι κρίσιμο για εφαρμογές σε κινητές συσκευές ή ενσωματωμένα συστήματα.

Γύρω στο 2020, μια νέα προσέγγιση –οι Μετασχηματιστές Όρασης (Vision Transformers)– έκανε την εμφάνισή της, σηματοδοτώντας ενδεχομένως μια τρίτη μεγάλη φάση στην ιστορία της αναγνώρισης εικόνων. Οι Dosovitskiy et al. (2020) [47] έδειξαν ότι προσαρμόζοντας την αρχιτεκτονική *Transformer* (που είχε κυριαρχήσει στη NLP) σε δεδομένα εικόνας, μπορούμε να πετύχουμε αποτελέσματα συγκρίσιμα με τα καλύτερα CNNs της εποχής. Οι Vision Transformers [23], χρησιμοποιώντας εντελώς διαφορετική φιλοσοφία (μηχανισμό αυτο-προσοχής αντί συνελίξεων), άνοιξαν νέο πεδίο έρευνας. Έκτοτε, έχουν προταθεί υβριδικές προσεγγίσεις (συνδυασμός CNN και Transformer), καθώς και βελτιώσεις των ViT (π.χ. Swin Transformer, DeiT κ.ά.), διευρύνοντας το ρεπερτόριο των μοντέλων για όραση.

Benchmark datasets & competitions

Παράλληλα με τις αρχιτεκτονικές, εξαιρετικά σημαντικό ρόλο στην ιστορική πρόοδο έπαιξαν οι βάσεις δεδομένων και οι διαγωνισμοί αξιολόγησης που καθιέρωσαν κοινά μέτρα σύγκρισης. Ήδη αναφέρθηκαν το MNIST [24] (χειρόγραφα ψηφία) που καθιερώθηκε το 1998 ως πρότυπο benchmark για ταξινόμηση ψηφίων, και το ImageNet (2009+) που έγινε ο πυρήνας των ετήσιων διαγωνισμών ILSVRC την περίοδο 2010-2017, οδηγώντας σε άμιλλα μεταξύ ερευνητικών ομάδων και σε αλματώδη βελτίωση αλγορίθμων. Εκτός αυτών, αξίζει να σημειωθούν: το

PASCAL VOC (2007–2012) [25] – ένα σημαντικό σετ δεδομένων και διαγωνισμός για εντοπισμό και αναγνώριση αντικειμένων σε εικόνες, που προηγήθηκε του ImageNet και συνέβαλε στις μετρικές για ανίχνευση αντικειμένων· το CIFAR-10/100 [48] – μικρότερα σύνολα γενικών εικόνων που χρησιμοποιήθηκαν ευρέως στη δεκαετία 2000 και στα early 2010s για έλεγχο αλγορίθμων· και το COCO (Common Objects in Context) [26] που εισήχθη από τη Microsoft το 2014. Το MS COCO [49] είναι ένα μεγάλης κλίμακας dataset για ανίχνευση αντικειμένων, τμηματοποίηση και περιγραφή εικόνων, με ~330.000 εικόνες (περισσότερες από 200.000 σχολιασμένες), ~1,5 εκατομμύρια παραδείγματα αντικειμένων και 80 κατηγορίες αντικειμένων κοινού ενδιαφέροντος. Ο διαγωνισμός COCO Challenge έδωσε κίνητρο για την ανάπτυξη προχωρημένων μεθόδων object detection (π.χ. τα δίκτυα τύπου R-CNN, Fast/Faster R-CNN, Mask R-CNN, YOLO, SSD κ.ά.) [50] και instance segmentation μετά το 2015. Συνολικά, αυτά τα ανοιχτά σύνολα δεδομένων και οι αντίστοιχοι διαγωνισμοί λειτούργησαν ως *καταλύτες* της προόδου: επέτρεψαν την αντικειμενική σύγκριση νέων αλγορίθμων, ενθάρρυναν τη συνεργασία και τον ανταγωνισμό στην παγκόσμια ερευνητική κοινότητα και επιτάχυναν την υιοθέτηση των βέλτιστων πρακτικών. Σήμερα, η αναγνώριση εικόνων έχει ωριμάσει σε σημείο που βαθιά νευρωνικά μοντέλα όχι μόνο πετυχαίνουν υψηλή ακρίβεια σε ορισμένα benchmarks, αλλά και μεταφέρονται επιτυχώς σε πλήθος εφαρμογών στον πραγματικό κόσμο. Στα επόμενα χρόνια (2020+), η έρευνα συνεχίζεται με έμφαση στη βελτίωση της αποδοτικότητας, της ερμηνευσιμότητας και της γενίκευσης των μοντέλων, αξιοποιώντας τεχνικές όπως η αυτοεπιβλεπόμενη μάθηση (self-supervised learning), τα γενετικά δίκτυα (GANs) [27] για δεδομένα που δεν επαρκούν, και η καλύτερη ολοκλήρωση των μοντέλων όρασης με άλλα είδη δεδομένων (π.χ. γλώσσα, απτικό ερέθισμα) ώστε να δημιουργηθούν πιο ολοκληρωμένοι *έξυπνοι πράκτορες*.

1.3 Εφαρμογές & Τεχνικές

Η σύγχρονη τεχνολογία αναγνώρισης εικόνας έχει πλέον διεισδύσει σε πληθώρα τομέων, από τη βιομηχανική παραγωγή μέχρι την υγεία και την καθημερινότητα, φέρνοντας επανάσταση σε διαδικασίες και υπηρεσίες. Στον βιομηχανικό τομέα [28], οι μέθοδοι υπολογιστικής όρασης χρησιμοποιούνται ευρέως για έλεγχο ποιότητας (Quality Control) και αυτοματοποιημένη επιθεώρηση προϊόντων. Σε γραμμές παραγωγής, "έξυπνες" κάμερες εξοπλισμένες με αλγόριθμους βαθιάς μάθησης μπορούν να ανιχνεύουν ελαττώματα σε προϊόντα με ταχύτητα και ακρίβεια που υπερβαίνει τις ανθρώπινες δυνατότητες, εξασφαλίζοντας υψηλότερα πρότυπα ποιότητας και μειωμένο κόστος. Για παράδειγμα, έχει αναπτυχθεί σύστημα οπτικού ελέγχου εκτυπωτικών κυλίνδρων όπου ένα βαθύ νευρωνικό δίκτυο συγκρίνει σε πραγματικό χρόνο την παραγόμενη εικόνα με το επιθυμητό πρότυπο και εντοπίζει αυτόματα ελαττώματα (π.χ. μικροσκοπικές οπές) με ακρίβεια άνω του 98%. Τέτοιες εφαρμογές *"machine vision"* στην

βιομηχανία 4.0 βελτιώνουν δραστικά την αποδοτικότητα, επιτρέποντας στις μηχανές να «αντιλαμβάνονται» μόνες τους την κατάσταση της παραγωγικής διαδικασίας και να απορρίπτουν ελαττωματικά τεμάχια έγκαιρα. Επιπλέον, στην ρομποτική και στα αυτοματοποιημένα συστήματα, η όραση υπολογιστή επιτρέπει σε ρομπότ να εντοπίζουν αντικείμενα, να καθοδηγούν την κίνησή τους στο χώρο και να αλληλεπιδρούν με δυναμικά περιβάλλοντα – από βιομηχανικούς βραχίονες που αναγνωρίζουν και ταξινομούν κομμάτια πάνω σε έναν ιμάντα, μέχρι drones [51] που χαρτογραφούν τον χώρο γύρω τους αποφεύγοντας εμπόδια. Η υποβοηθούμενη πλοήγηση σε ρομπότ ή οχηματαγωγά συστήματα βασίζεται σε κάμερες και αλγορίθμους όρασης για την κατανόηση της σκηνής.

Στον χώρο των αυτόνομων οχημάτων [29] και των προηγμένων συστημάτων οδηγικής υποβοήθησης (ADAS), η αναγνώριση εικόνας είναι απόλυτα κρίσιμη. Τα αυτοκίνητα χωρίς οδηγό, όπως και τα σύγχρονα συμβατικά οχήματα με συστήματα αποφυγής σύγκρουσης, χρησιμοποιούν πληθώρα αισθητήρων – με κυριότερες τις κάμερες – για να "δει" το όχημα το περιβάλλον του. Οι αλγόριθμοι βαθιάς μάθησης επεξεργάζονται σε πραγματικό χρόνο τα βίντεο από αυτές τις κάμερες και αναλαμβάνουν εργασίες όπως: ανίχνευση πεζών και ποδηλατών στον δρόμο, αναγνώριση και κατηγοριοποίηση άλλων οχημάτων, ανάγνωση σημάτων κυκλοφορίας και φώτων, καθώς και αναγνώριση της διαγράμμισης και του οδοστρώματος για διατήρηση λωρίδας. Η πρόκληση εδώ είναι να λαμβάνονται αποφάσεις σε κλάσματα δευτερολέπτου με υψηλή αξιοπιστία, ώστε το όχημα να είναι ασφαλές. Η πρόοδος της όρασης υπολογιστή έχει βελτιώσει θεαματικά αυτές τις δυνατότητες. Για παράδειγμα, ερευνητές του UC San Diego [52] ανέπτυξαν ένα νέο σύστημα εντοπισμού πεζών που συνδυάζει έναν γρήγορο παραδοσιακό αλγόριθμο ανίχνευσης (cascade detection) για τα πρώτα στάδια με ένα βαθύ νευρωνικό δίκτυο στα τελικά στάδια – δημιουργώντας μια υβριδική προσέγγιση που πετυχαίνει πολύ καλύτερη ισορροπία μεταξύ ταχύτητας και ακρίβειας. Το σύστημα αυτό κατόρθωσε να μειώσει στο μισό το ποσοστό σφάλματος στην ανίχνευση πεζών σε σχέση με προηγούμενες τεχνικές, φέρνοντας την απόδοση πιο κοντά στις απαιτήσεις πραγματικού χρόνου. Γενικά, εταιρείες όπως η Tesla, η Google (Waymo) και άλλες έχουν αναπτύξει εξελιγμένα μοντέλα όρασης όπου συνδυάζονται συνελικτικά δίκτυα και μετασχηματιστές για να αναλύσουν πολυκάναλα οπτικά δεδομένα (ορατό, υπέρυθρο κ.ά.) και να παράγουν μια πλούσια κατανόηση του οδικού περιβάλλοντος. Χάρη σε αυτά, σύγχρονα οχήματα μπορούν αυτόματα να φρενάρουν όταν πεταχτεί ένας πεζός, να διατηρούν ασφαλή απόσταση από προπορευόμενα αυτοκίνητα και να παραμένουν στη λωρίδα – όλα μέσω της "αντίληψης" που τους προσδίδει η υπολογιστική όραση.

Στον κρίσιμο τομέα της Υγείας [30], και ειδικότερα στη ιατρική διάγνωση μέσω εικόνων, η συνεισφορά της βαθιάς μάθησης είναι επαναστατική. Η ιατρική πρακτική παράγει τεράστιο όγκο οπτικών δεδομένων – ακτινογραφίες, αξονικές (CT), μαγνητικές τομογραφίες (MRI),

υπερηχογραφήματα, μικροσκοπικές εικόνες ιστών κ.ά. Παραδοσιακά, η ερμηνεία αυτών των εικόνων γίνεται από ειδικούς ιατρούς (ακτινολόγους, παθολογοανατόμους), όμως οι αλγόριθμοι CNN έχουν δείξει ότι μπορούν να αναλάβουν πολλές από αυτές τις διαγνώσεις με ταχύτητα και αξιοθαύμαστη ακρίβεια. Ήδη από το 2016–2017 δημοσιεύθηκαν μελέτες-ορόσημα που έδειξαν ότι ένα CNN μπορεί να διαγνώσει κακοήθειες δέρματος σε δερματοσκοπικές εικόνες [53] με ευαισθησία και ειδικότητα αντίστοιχη ενός έμπειρου δερματολόγου, ή ότι μοντέλα βαθιάς μάθησης σε οφθαλμολογικές εικόνες αμφιβληστροειδούς μπορούν να ανιχνεύσουν διαβητική αμφιβληστροειδοπάθεια στο επίπεδο ενός ειδικού. Μεταγενέστερα, ένας μεγάλος αριθμός εργασιών έχει τεκμηριώσει την υψηλή επίδοση της TN σε πλήθος διαγνωστικών καθηκόντων. Σύμφωνα με μια πρόσφατη μετα-ανάλυση (Aggarwal et al., 2021) [54], σε ειδικότητες όπως η οφθαλμολογία, η πνευμονολογία και η μαστογραφία, τα μοντέλα DL πέτυχαν περιοχές υπό την καμπύλη ROC (AUC) εξαιρετικά υψηλές: π.χ. 0.93–1.00 στην ανίχνευση διαβητικής αμφιβληστροειδοπάθειας, [55] εκφύλισης ωχράς κηλίδας και γλαυκώματος από απεικονίσεις αμφιβληστροειδούς, 0.86–0.94 στην ανίχνευση οζιδίων ή καρκίνου του πνεύμονα σε ακτινογραφίες θώρακος και CT, και 0.87–0.91 στην ανίχνευση καρκίνου μαστού σε μαστογραφίες, υπερήχους και MRI. Αυτές οι επιδόσεις είναι συγκρίσιμες με των ανθρώπων ειδικών – σε ορισμένες περιπτώσεις αγγίζουν ή και υπερβαίνουν το μέσο επίπεδο ακρίβειας των ιατρών – γεγονός που υπογραμμίζει τις δυνατότητες της βαθιάς μάθησης να μεταμορφώσει τη διαγνωστική διαδικασία. Ήδη, ο FDA στις ΗΠΑ έχει εγκρίνει εμπορικά εργαλεία λογισμικού βασισμένα σε TN για υποβοήθηση διάγνωσης, όπως για αυτόματη ανάλυση OCT αμφιβληστροειδούς ή ανίχνευση μικροαιμορραγιών διαβητικής αμφιβληστροειδοπάθειας. Πέρα από τη διάγνωση, τα οράματα υπολογιστή συνδράμουν σε χειρουργικές εφαρμογές (π.χ. εντοπισμός όγκων κατά τη διάρκεια επεμβάσεων), στην οργάνωση και ανάκτηση ιατρικών εικόνων, καθώς και στην εξατομίκευση θεραπειών μέσω καλύτερης ποσοτικοποίησης βιοδεικτών από εικόνες. Είναι σημαντικό ότι τα συστήματα αυτά δεν κουράζονται, μπορούν να αξιολογούν τεράστιο αριθμό εξετάσεων αδιάλειπτα και να ειδοποιούν για ύποπτα ευρήματα, αυξάνοντας την αποδοτικότητα και μειώνοντας την πιθανότητα ανθρώπινων λαθών.

Επιπλέον, στην εποχή του Διαδικτύου των Πραγμάτων (IoT) [31], η αναγνώριση εικόνων βρίσκει θέση σε πληθώρα καθημερινών εφαρμογών και συσκευών. Πλέον, αισθητήρες εικόνας και κάμερες συνδεδεμένες στο διαδίκτυο υπάρχουν στα κινητά τηλέφωνα, στα έξυπνα οικιακά συστήματα, στις κάμερες ασφαλείας, στους αισθητήρες κυκλοφορίας των “έξυπνων πόλεων” και αλλού. Ένας συνδυασμός της υπολογιστικής όρασης με τη συνδεσιμότητα IP και το cloud computing δημιουργεί ένα νέο οικοσύστημα εφαρμογών. Συστήματα επιτήρησης και ασφάλειας, που παραδοσιακά βασίζονταν σε ανθρώπους για την παρακολούθηση καμερών, τώρα ενσωματώνουν AI που μπορεί αυτόματα να ανιχνεύει εισβολείς, να αναγνωρίζει

πρόσωπα ή να ανιχνεύει ύποπτες συμπεριφορές σε πραγματικό χρόνο (video analytics). Στον καταναλωτικό τομέα, κάμερες σε κονσόλες παιχνιδιών αναγνωρίζουν τις χειρονομίες του παίκτη (π.χ. Microsoft Kinect), smartphones χρησιμοποιούν αναγνώριση προσώπου για το ξεκλείδωμα της συσκευής ή για φίλτρα επαυξημένης πραγματικότητας στα μέσα κοινωνικής δικτύωσης, και έξυπνες εφαρμογές μπορούν να κατανοούν το περιεχόμενο μιας φωτογραφίας (π.χ. το Google Lens αναγνωρίζει αντικείμενα ή κείμενο όταν στρέφουμε την κάμερα προς αυτά).

Επιπλέον, στην αλυσίδα λιανικής και logistics [32], η όραση υπολογιστή αξιοποιείται για καταμέτρηση αποθεμάτων μέσω καμερών στα ράφια, για αυτόματες πληρωμές χωρίς ταμείο (Amazon Go), ή για την επιτήρηση συνθηκών μεταφοράς ευαίσθητων προϊόντων. Ακόμη και στα οχήματα IoT (όπως drones διανομής) ή σε έξυπνες συσκευές σπιτιού (π.χ. ρομποτικές σκούπες με κάμερες που "βλέπουν" εμπόδια), η αναγνώριση εικόνων παίζει καθοριστικό ρόλο. Συνολικά, η υπολογιστική όραση στο περιβάλλον IoT επεκτείνει την "αντιληπτική ικανότητα" σε αντικείμενα καθημερινής χρήσης, δημιουργώντας πιο έξυπνα και αυτόνομα οικοσυστήματα. Όπως χαρακτηριστικά αναφέρεται, η τεχνολογία αυτή συναντάται παντού: από κονσόλες παιχνιδιών που αναγνωρίζουν χειρονομίες, έως κάμερες κινητών που εστιάζουν αυτόματα στα πρόσωπα – ήδη επηρεάζει πολλές πτυχές της ζωής μας. Καθώς οι αισθητήρες γίνονται φτηνότεροι και ενεργειακά αποδοτικότεροι, είναι εφικτό να αναπτυχθούν δίκτυα έξυπνων καμερών που θα παρακολουθούν το περιβάλλον για μεγάλο χρονικό διάστημα (π.χ. περιβαλλοντική παρακολούθηση, "έξυπνες" πόλεις) χωρίς ανθρώπινη παρέμβαση, δίνοντας τη δυνατότητα σε πραγματικό χρόνο συλλογής και ανάλυσης οπτικών δεδομένων προς όφελος της κοινωνίας.

1.4 Παραδοσιακές vs. Deep Learning προσεγγίσεις

Αξίζει να συνοψίσουμε πώς διαφέρουν οι κλασικές μέθοδοι αναγνώρισης εικόνων από τις σύγχρονες προσεγγίσεις βαθιάς μάθησης [33] [34], καθώς και ποιες τεχνικές συμβάλλουν στην επιτυχία των δεύτερων. Όπως αναφέρθηκε, οι παραδοσιακές προσεγγίσεις βασίζονταν στην εξαγωγή χαρακτηριστικών με ανθρώπινη επιμέλεια: σχεδιασμένοι αλγόριθμοι ανίχνευαν ακμές, γωνίες (π.χ. Harris detector), τοπικές υφές ή σύνθετα descriptors όπως οι SIFT και HOG, και κατόπιν ένας κλασικός ταξινομητής προσπαθούσε να αναγνωρίσει το αντικείμενο βάσει αυτών των χαρακτηριστικών. Παρά την επιτυχία αυτών των μεθόδων σε ελεγχόμενες συνθήκες, είχαν περιορισμένη ευρωστία: η απόδοσή τους συχνά υποβαθμιζόταν σε πραγματικά περιβάλλοντα με θόρυβο, μεταβολές φωτισμού ή μη αναμενόμενες οπτικές περιστάσεις. Αντίθετα, τα βαθιά νευρωνικά δίκτυα δεν απαιτούν ρητή σχεδίαση χαρακτηριστικών – μαθαίνουν αυτόνομα ποιοι οπτικοί δείκτες είναι σημαντικοί. Στα πρώτα επίπεδα ενός CNN, για

παράδειγμα, το δίκτυο μαθαίνει να ανιχνεύει απλές ακμές και γωνίες, σε ενδιάμεσα επίπεδα ανιχνεύει πιο σύνθετα μοτίβα (π.χ. υφές, σχήματα), και στα τελικά επίπεδα αναγνωρίζει ολόκληρα αντικείμενα ή μέρη τους. Αυτή η πολυεπίπεδη μάθηση χαρακτηριστικών έχει αποδειχθεί απείρως πιο αποτελεσματική από την χειροτεχνία: όπως είδαμε με το AlexNet, ένα deep learning σύστημα κατάφερε μέσα σε λίγες ώρες εκπαίδευσης να ξεπεράσει ό,τι είχαν επιτύχει δεκαετίες χειροκίνητου σχεδιασμού αλγορίθμων από ειδικούς. Επιπλέον, τα βαθιά μοντέλα είναι σε θέση να γενικεύουν καλύτερα, καθώς μαθαίνουν αναπαραστάσεις που μπορούν να προσαρμοστούν και σε νέα δεδομένα ή κατηγορίες με ελάχιστη προσαρμογή, κάτι που δεν ίσχυε για τα άκαμπτα κλασικά συστήματα.

Ένα κομβικό στοιχείο που αξιοποιείται στις σύγχρονες πρακτικές είναι η Μεταφορά Μάθησης (Transfer Learning). Στον πραγματικό κόσμο, συχνά δεν υπάρχουν διαθέσιμα τεράστια σύνολα δεδομένων για κάθε επιμέρους εφαρμογή. Η λύση που έχει επικρατήσει είναι να επαναχρησιμοποιούνται γνώση και πρότυπα που έμαθε ένα δίκτυο σε ένα μεγάλο dataset (π.χ. ImageNet) για να λυθεί μια διαφορετική αλλά συναφής εργασία. Συγκεκριμένα, τα βαθιά CNN που εκπαιδεύτηκαν στο ImageNet (το οποίο περιλαμβάνει 1000 ποικίλες κατηγορίες αντικειμένων) έχουν μάθει πολύ γενικές και χρήσιμες χαρακτηριστικές αναπαραστάσεις στα πρώτα τους επίπεδα – όπως άκρα, γωνίες, υφές – που είναι χρήσιμες σχεδόν σε οποιαδήποτε οπτική αναγνώριση. Έτσι, μια κοινή πρακτική είναι να λαμβάνουμε ένα προ-εκπαιδευμένο δίκτυο (π.χ. ResNet50) και να το χρησιμοποιούμε ως βάση για μια νέα εργασία (π.χ. αναγνώριση τύπων καρκινικών κυττάρων από ιστολογικές εικόνες). Διατηρώντας τα αρχικά στρώματα και προσαρμόζοντας μόνο τα τελικά (ή ακόμη και όλα τα στρώματα με μικρό ρυθμό μάθησης) – διαδικασία γνωστή ως fine-tuning – μπορούμε με σχετικά λίγα δεδομένα να πετύχουμε εξαιρετική απόδοση. Το transfer learning έχει καθιερωθεί ως στάνταρ πρακτική, διότι μειώνει δραστικά τις απαιτήσεις σε δεδομένα και χρόνο: αντί για εβδομάδες εκπαίδευσης από το μηδέν, συχνά λίγες ώρες ή λεπτά αρκούν για να προσαρμόσουμε ένα δίκτυο σε νέα καθήκοντα, εκμεταλλευόμενοι τη "γνώση" που έχει ήδη αποκτήσει.

Έτερο κρίσιμο στοιχείο είναι η Διεύρυνση Δεδομένων (Data Augmentation). Ακόμα και με μεγάλα datasets, τα deep models έχουν τάση να υπερ-προσαρμόζονται (overfit) στα δεδομένα εκπαίδευσης αν δεν δουν αρκετή ποικιλία εισόδων. Οι ερευνητές αντιμετώπισαν αυτό το ζήτημα εφαρμόζοντας τεχνικές augmentation – δηλαδή τεχνητές μεταμορφώσεις των εικόνων που παράγουν νέες, διαφορετικές αλλά ισοδύναμες εκπαιδευτικές περιπτώσεις. Το AlexNet (2012) ήταν από τα πρώτα που αξιοποίησαν εκτενώς αυτή την πρακτική: κατά την εκπαίδευση, κάθε εικόνα υποβαλλόταν σε τυχαίες μεταφράσεις (μετακινήσεις), περιστροφές, συμμετρίες οριζόντιες, τυχαίες περικοπές (cropping) κ.ά., πριν δοθεί στο δίκτυο. Έτσι, το δίκτυο μάθαινε να εστιάζει στα ουσιώδη χαρακτηριστικά του αντικειμένου ανεξάρτητα από θέση/κλίμακα, αντί να "αποστηθίζει" συγκεκριμένες εικόνες. Το αποτέλεσμα ήταν ένα πιο *γενικευμένο* μοντέλο με

σημαντικά μειωμένο σφάλμα. Σήμερα, το data augmentation θεωρείται απαραίτητο βήμα σε κάθε pipeline εκπαίδευσης μοντέλου όρασης: από απλές λειτουργίες (μεταστροφές, θόρυβος, αλλαγές φωτεινότητας/αντίθεσης) μέχρι προχωρημένες τεχνικές όπως Cutout, Mixup ή Generative augmentation (χρήση GANs για παραγωγή συνθετικών εικόνων), όλες αποσκοπούν στο να αυξηθεί η ποικιλία των δεδομένων και να καταστεί το μοντέλο πιο ανθεκτικό.

Εν κατακλείδι, η σύγχρονη αναγνώριση εικόνων συνδυάζει: ισχυρά μοντέλα βαθιάς μάθησης που αυτόματα μαθαίνουν χαρακτηριστικά, έξυπνες στρατηγικές εκπαίδευσης όπως η μεταφορά μάθησης και η αύξηση δεδομένων, καθώς και την επεξεργαστική ισχύ του σημερινού υλικού, προκειμένου να επιτύχει επιδόσεις που ήταν αδιανόητες μόλις μια δεκαετία πριν. Αυτές οι τεχνικές, συνεπικουρούμενες από τη διαθεσιμότητα ανοιχτών βιβλιοθηκών (TensorFlow, PyTorch κ.ά.) και την πρόσβαση σε μεγάλα σύνολα δεδομένων, έχουν εκδημοκρατίσει το πεδίο – επιτρέποντας τόσο στην ακαδημαϊκή έρευνα όσο και στη βιομηχανία να αναπτύσσουν συνεχώς νέες εφαρμογές. Η αναγνώριση εικόνας σήμερα αποτελεί αναπόσπαστο κομμάτι της τεχνητής νοημοσύνης που βλέπουμε γύρω μας, και οι προοπτικές της, με αλγόριθμους ολοένα πιο έξυπνους και με δυνατότητα κατανόησης του οπτικού κόσμου σε βάθος, διαγράφονται συναρπαστικές για το μέλλον.

Κεφάλαιο 2ο: ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ & ΥΛΟΠΟΙΗΣΗ

2.1 Περιγραφή Λειτουργίας

Στην παρούσα εργασία αναπτύχθηκε ένα ολοκληρωμένο σύστημα αυτοματοποιημένης ανίχνευσης, αναγνώρισης και υπολογισμού της τιμής ηλεκτρονικών αντιστάσεων, με χρήση τεχνικών επεξεργασίας εικόνας και αλγορίθμων μηχανικής όρασης. Ο πυρήνας του συστήματος βασίζεται στην ανάλυση ζωντανής εικόνας (real-time video stream) που λαμβάνεται από εξωτερική κάμερα, με τελικό σκοπό την ακριβή ταυτοποίηση των χρωματικών δακτυλίων πάνω στις αντιστάσεις και τον αυτόματο υπολογισμό της τιμής τους σύμφωνα με τον διεθνή κώδικα χρωμάτων.

Η βασική ροή λειτουργίας του συστήματος μπορεί να περιγραφεί μέσα από τρία κύρια διαδοχικά στάδια: (α) λήψη και προεπεξεργασία της εικόνας, (β) εντοπισμός των ίδιων των αντιστάσεων εντός της σκηνής, και (γ) ανάλυση και ερμηνεία των χρωματικών δακτυλίων για την εξαγωγή της τιμής της αντίστασης. Αναλυτικότερα, κάθε στάδιο περιλαμβάνει επιμέρους βήματα και τεχνικές, που σχεδιάστηκαν με γνώμονα τόσο την ακρίβεια όσο και την αποδοτικότητα του συστήματος, επιτρέποντας τη λειτουργία σε πραγματικό χρόνο και σε περιβάλλον με θόρυβο ή ποικίλες συνθήκες φωτισμού.

Στο πρώτο στάδιο, το πρόγραμμα χρησιμοποιεί τη βιβλιοθήκη OpenCV [56] για την απόκτηση ακολουθίας καρέ από συνδεδεμένη κάμερα (webcam), η οποία λειτουργεί ως ο βασικός αισθητήρας εισόδου. Για κάθε καρέ που λαμβάνεται, πραγματοποιείται προεπεξεργασία που περιλαμβάνει, μεταξύ άλλων, εφαρμογή διμερούς φιλτραρίσματος (bilateral filtering) για την απομάκρυνση του θορύβου και διατήρηση των άκρων, καθώς και μετασχηματισμό χρωματικού χώρου από RGB/BGR σε HSV. Η χρήση του χρωματικού χώρου HSV είναι ιδιαίτερα σημαντική, καθώς προσφέρει σταθερότητα στην ανίχνευση χρωμάτων, μειώνοντας την ευαισθησία του αλγορίθμου σε αλλαγές της φωτεινότητας και των συνθηκών φωτισμού του περιβάλλοντος.

Ακολουθεί το στάδιο εντοπισμού των αντιστάσεων μέσα στη σκηνή, το οποίο βασίζεται στη χρήση προκαθορισμένου αλγορίθμου Haar Cascade Classifier [57]. Ο συγκεκριμένος ανιχνευτής εκπαιδεύεται ώστε να αναγνωρίζει τις τυπικές γεωμετρικές ιδιότητες των ηλεκτρονικών αντιστάσεων (δηλαδή το ορθογώνιο σχήμα, τη χαρακτηριστική αναλογία διαστάσεων κ.λπ.) μέσα στο καρέ. Σε κάθε λήψη, ο ανιχνευτής εφαρμόζεται στο γκρίζο επίπεδο της εικόνας, ανιχνεύοντας περιοχές ενδιαφέροντος που πιθανώς να περιέχουν αντίσταση. Για την περαιτέρω μείωση των ψευδών θετικών (false positives), πραγματοποιείται και δεύτερος έλεγχος (“second pass”) πάνω στις ίδιες περιοχές, απομονώνοντας τελικά μόνο όσες περιοχές περνούν με επιτυχία και τα δύο στάδια φιλτραρίσματος.

Οι περιοχές που έχουν εντοπιστεί ως αντιστάσεις υποβάλλονται στη συνέχεια σε διαδικασία περαιτέρω ανάλυσης, με στόχο την εξαγωγή των χρωματικών δακτυλίων που καθορίζουν τη μοναδική τιμή της κάθε αντίστασης. Σε αυτό το τρίτο στάδιο, η περιοχή της εντοπισμένης αντίστασης αποκόπτεται και μεγεθύνεται για να αυξηθεί η ανάλυση, ενώ εφαρμόζεται αλγόριθμος προσαρμοστικού κατωφλίου (adaptive thresholding) και άλλες μορφολογικές πράξεις, ώστε να αναδειχθούν τα χρωματικά χαρακτηριστικά σε σχέση με το υπόβαθρο. Το επόμενο καθοριστικό βήμα είναι η δημιουργία μασκών (masking) για κάθε πιθανό χρώμα, με χρήση ορίων (thresholds) που έχουν καθοριστεί χειροκίνητα και αντιστοιχούν στις τυπικές αποχρώσεις που εμφανίζονται στους δακτυλίους των αντιστάσεων. Η διαδικασία αυτή περιλαμβάνει επιπλέον ειδικούς χειρισμούς, όπως για το κόκκινο χρώμα (RED), το οποίο λόγω της κατανομής στο HSV απαιτεί συνένωση δύο διαστημάτων τιμών (lower/upper bounds).

Μετά την εφαρμογή των μασκών, το σύστημα εντοπίζει τα περιγράμματα (contours) των χρωματικών περιοχών, τα οποία φιλτράρονται βάσει εμβαδού και λόγου διαστάσεων για να απορρίπτονται τυχόν θόρυβοι ή μη σχετικές περιοχές. Για κάθε έγκυρο χρωματικό δακτύλιο, καταγράφεται η θέση του και το χρώμα του, ώστε να δημιουργηθεί μια σειρά από rules (συντεταγμένες, όνομα χρώματος, αριθμητική τιμή βάσει κώδικα χρωμάτων). Τα δεδομένα αυτά ταξινομούνται με βάση τη θέση τους, αναδομώντας την πραγματική σειρά των δακτυλίων πάνω στη φυσική αντίσταση.

Τελικά, ο υπολογισμός της τιμής της αντίστασης γίνεται μέσω της σύνθεσης των διακριτών ψηφίων που αντιπροσωπεύουν οι δακτύλιοι. Ανάλογα με τον αριθμό των εντοπισμένων ζωνών (συνήθως 4 ή 5), ο αλγόριθμος συνενώνει τα ψηφία και εφαρμόζει τον κατάλληλο πολλαπλασιαστή, δίνοντας ως αποτέλεσμα την πραγματική τιμή της αντίστασης σε ohm. Το αποτέλεσμα προβάλλεται απευθείας πάνω στο αρχικό καρέ της εικόνας, με χρήση επικαλυπτόμενου γραφικού στοιχείου (overlay), επιτρέποντας στον χρήστη να δει σε πραγματικό χρόνο τόσο τη θέση και το περίγραμμα της ανιχνευθείσας αντίστασης, όσο και την τιμή που αυτή έχει υπολογιστεί.

Επιπροσθέτως, το σύστημα περιλαμβάνει ειδικό “debug mode”, το οποίο επιτρέπει στον χρήστη να προσαρμόζει δυναμικά τα όρια των χρωμάτων μέσω κατάλληλων sliders (trackbars), προσφέροντας έτσι τη δυνατότητα προσαρμογής του αλγορίθμου σε διαφορετικές κάμερες, συνθήκες φωτισμού ή χρωματικές αποκλίσεις, καθιστώντας τη διαδικασία ιδιαίτερα ευέλικτη και αξιόπιστη σε ποικίλα περιβάλλοντα εργασίας.

2.2 Περιβάλλον & Βιβλιοθήκες

Η ανάπτυξη της εφαρμογής πραγματοποιήθηκε σε περιβάλλον Python έκδοσης 3.6, αξιοποιώντας το οικοσύστημα της διανομής **Anaconda** και το ολοκληρωμένο περιβάλλον ανάπτυξης **Spyder** [58] (Scientific Python Development Environment). Η επιλογή της συγκεκριμένης διανομής και IDE (Integrated Development Environment) δεν είναι τυχαία, καθώς το Anaconda προσφέρει εξαιρετική διαχείριση εξαρτήσεων και εικονικών περιβαλλόντων, καθώς και προεγκατεστημένες πληθώρα επιστημονικών βιβλιοθηκών, διευκολύνοντας την εγκατάσταση, την εκτέλεση και την αναπαραγωγή πειραμάτων σε διαφορετικά συστήματα. Το Spyder αποτελεί ένα από τα πιο φιλικά περιβάλλοντα για τον επιστήμονα δεδομένων και τον μηχανικό, προσφέροντας λειτουργίες debugging, inline visualization, διαχείριση μεταβλητών και real-time feedback, χαρακτηριστικά που συνέβαλαν καθοριστικά στη γρήγορη ανάπτυξη και έλεγχο του κώδικα.

Η Python 3.6 [59] επελέγη διότι διατηρεί μεγάλη συμβατότητα με βιβλιοθήκες που χρησιμοποιούνται ευρέως στον χώρο της επιστημονικής υπολογιστικής και της μηχανικής όρασης, ενώ υποστηρίζεται πλήρως από το Anaconda Distribution, εξασφαλίζοντας σταθερότητα και αξιοπιστία του περιβάλλοντος εκτέλεσης. Επιπλέον, η ύπαρξη virtual environments στο Anaconda επιτρέπει την απομόνωση της εργασίας αυτής από άλλες εγκαταστάσεις Python, ελαχιστοποιώντας το ενδεχόμενο συγκρούσεων ή προβλημάτων εξαρτήσεων.

OpenCV (cv2)

Η βασική βιβλιοθήκη που χρησιμοποιείται για την επεξεργασία και ανάλυση εικόνας είναι η OpenCV [56] (έκδοση 3.4.x για συμβατότητα με Python 3.6). Η OpenCV αποτελεί παγκόσμιο σημείο αναφοράς για εφαρμογές υπολογιστικής όρασης και προσφέρει εκτεταμένες λειτουργίες για τη λήψη, επεξεργασία και ανάλυση εικόνων και βίντεο, ανίχνευση αντικειμένων, μορφολογικούς μετασχηματισμούς, και εφαρμογή διαφόρων φίλτρων. Στην παρούσα εργασία, το OpenCV αξιοποιείται για την πρόσβαση σε βίντεο μέσω κάμερας (cv2.VideoCapture), τον μετασχηματισμό χρωματικού χώρου (cv2.cvtColor), την εξομάλυνση της εικόνας (cv2.bilateralFilter), την κατωφλίωση (cv2.adaptiveThreshold), την ανίχνευση και σχεδίαση περιγραμμάτων (cv2.findContours, cv2.drawContours), αλλά και την οπτικοποίηση των ενδιάμεσων και τελικών αποτελεσμάτων μέσω παραθύρων (cv2.imshow, cv2.putText). Σημαντικό είναι επίσης το γεγονός ότι το OpenCV προσφέρει native υποστήριξη για Haar Cascade Classifiers, διευκολύνοντας την ενσωμάτωση προεκπαιδευμένων αλγορίθμων ανίχνευσης αντικειμένων.

NumPy

Αναπόσπαστο κομμάτι της υλοποίησης αποτελεί η βιβλιοθήκη NumPy [60], η οποία προσφέρει αποτελεσματική διαχείριση και επεξεργασία πολυδιάστατων πινάκων (arrays), καθώς και ταχύτατους υπολογισμούς σε μεγάλους όγκους δεδομένων, χαρακτηριστικά απαραίτητα για την επεξεργασία εικόνας σε πραγματικό χρόνο. Με την NumPy επιτυγχάνεται η εκτέλεση λογικών πράξεων, η δημιουργία μασκών και η ταχύτατη μετατροπή και ανάκτηση pixel δεδομένων σε όλα τα ενδιάμεσα στάδια του αλγορίθμου. Η χρήση NumPy arrays επιτρέπει τόσο την αποδοτική διασύνδεση με το OpenCV, όσο και την περαιτέρω ανάπτυξη ή επέκταση των λειτουργιών, π.χ. μέσω vectorized πράξεων.

imutils

Συμπληρωματικά, χρησιμοποιείται και η βιβλιοθήκη imutils [61], ένα ελαφρύ και πρακτικό εργαλείο που απλουστεύει βασικές λειτουργίες επεξεργασίας εικόνας, όπως το resizing, το cropping και διάφορες βοηθητικές μετατροπές. Το imutils ενισχύει την αναγνωσιμότητα και συντηρησιμότητα του κώδικα, επιτρέποντας στον προγραμματιστή να εστιάσει στη λογική της εφαρμογής αντί για την υλοποίηση σύνθετων αλλά κοινότυπων λειτουργιών.

Haar Cascade Classifier

Για την ανίχνευση των ίδιων των αντιστάσεων στην εικόνα αξιοποιήθηκε η μέθοδος Haar Cascade, η οποία βασίζεται σε προκαθορισμένα πρότυπα (features) που έχουν εξαχθεί από κατάλληλη εκπαίδευση σε σύνολο εικόνων [62]. Το απαραίτητο αρχείο εκπαίδευσης (XML) τοποθετείται τοπικά, στη διαδρομή που υποδεικνύει ο κώδικας, και χρησιμοποιείται από το OpenCV για την ταχεία αναγνώριση περιοχών που είναι πιθανό να περιέχουν αντιστάσεις.

Οδηγίες Εγκατάστασης & Λειτουργίας

Η εγκατάσταση των απαραίτητων βιβλιοθηκών εντός του Anaconda πραγματοποιείται είτε κατά τη δημιουργία του περιβάλλοντος είτε εκ των υστέρων μέσω του Anaconda Prompt ή του ενσωματωμένου διαχειριστή του Spyder. Για παράδειγμα, η δημιουργία νέου περιβάλλοντος με Python 3.6 και εγκατάσταση των βασικών πακέτων πραγματοποιείται με τις ακόλουθες εντολές:

```
conda create -n ohmvision_env python=3.6
```

```
conda activate ohmvision_env

conda install opencv numpy

pip install imutils
```

Επιπλέον, ο χρήστης θα πρέπει να διασφαλίσει ότι το αρχείο Haar Cascade (haarcascade_resistors_0.xml) βρίσκεται στη σωστή διαδρομή του project, όπως ορίζεται στον κώδικα. Το περιβάλλον Spyder, μέσω του Variable Explorer, επιτρέπει τον άμεσο έλεγχο των μεταβλητών, διευκολύνοντας τον εντοπισμό σφαλμάτων και τη βήμα-βήμα ανάλυση της ροής του αλγορίθμου, σημαντικό πλεονέκτημα στη διαδικασία ανάπτυξης και βελτιστοποίησης.

Υποστηριζόμενο Υλικό & Διαλειτουργικότητα

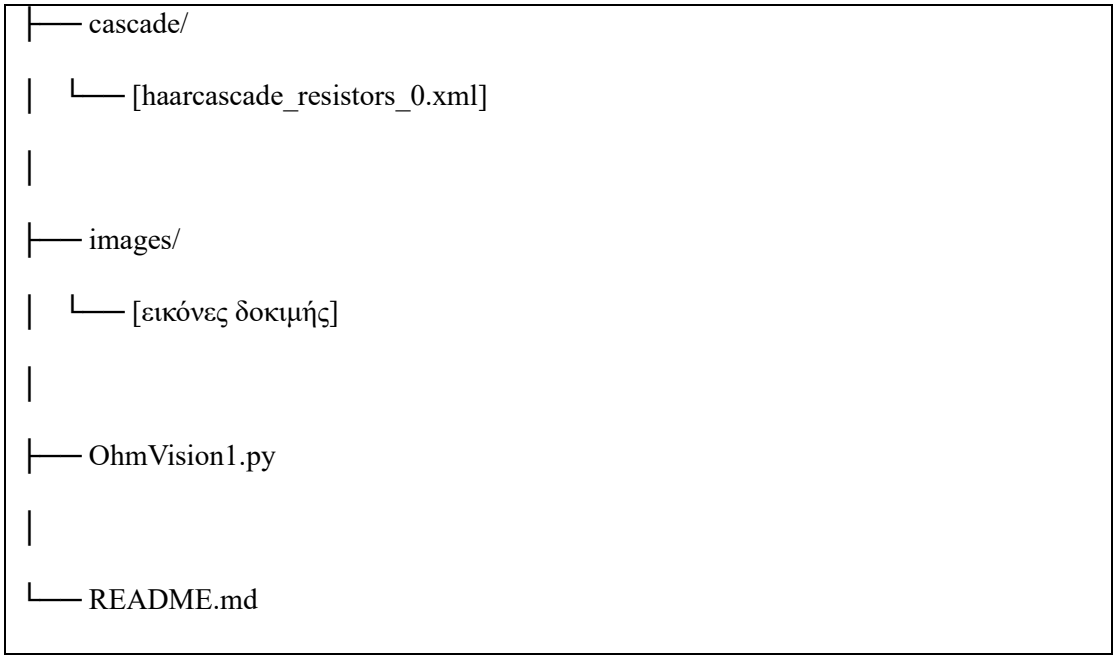
Η εφαρμογή είναι σχεδιασμένη ώστε να λειτουργεί σε σύγχρονους υπολογιστές με λειτουργικό σύστημα Windows 10 ή μεταγενέστερο, Linux (διανομές με υποστήριξη Anaconda) ή macOS, απαιτώντας τουλάχιστον 2GB μνήμης RAM, επεξεργαστή με δύο πυρήνες και πρόσβαση σε κάμερα (ενσωματωμένη ή εξωτερική USB). Χάρη στη χρήση των βιβλιοθηκών ανοιχτού κώδικα, το έργο παραμένει εύκολα μεταφέρσιμο και επαναλήψιμο σε διαφορετικές πλατφόρμες, ενώ η δυνατότητα δημιουργίας εικονικών περιβαλλόντων με το Anaconda εξασφαλίζει την απομόνωση και ασφάλεια κάθε project.

2.3 Δομή κώδικα

2.3.1 Διάγραμμα αρχείων/πακέτων

Η υλοποίηση του έργου ακολουθεί μια λιτή και πρακτική δομή φακέλων και αρχείων, κατάλληλη για ατομικά ερευνητικά project και πτυχιακές εργασίες που εστιάζουν σε μία βασική εφαρμογή με περιορισμένες εξαρτήσεις. Η συνολική οργάνωση του project όπως φαίνεται στον φάκελο εργασίας, αποτυπώνεται ως εξής:

```
OhmVisionProject/
|
```



Περιγραφή φακέλων και αρχείων:

- **cascade/**
  Φάκελος που περιέχει το προεκπαιδευμένο μοντέλο Haar Cascade σε μορφή XML, απαραίτητο για την ανίχνευση αντιστάσεων στην εικόνα μέσω OpenCV.
- **images/**
  Φάκελος που προορίζεται για τη φύλαξη εικόνων που χρησιμοποιούνται ως είσοδος ή.
- **OhmVision1.py**
  Το βασικό αρχείο κώδικα Python, που περιέχει όλη τη λογική της εφαρμογής: λήψη εικόνας, εντοπισμό αντιστάσεων, ανάλυση χρωματικών δακτυλίων και υπολογισμό τιμής. Πρόκειται για το script που εκτελείται απευθείας από το χρήστη, περιλαμβάνοντας τόσο τις κύριες συναρτήσεις όσο και τον βρόχο εκτέλεσης (main loop).
- **README.md**
  Συνοδευτικό αρχείο τεκμηρίωσης, με οδηγίες εκτέλεσης, περιγραφή του project, απαιτήσεις συστήματος και βασικά σημεία λειτουργίας του κώδικα.

Η δομή αυτή προάγει την καθαρότητα, καθώς ο διαχωρισμός ανάμεσα σε αρχεία μοντέλων (cascade), δεδομένα (images) και τον κύριο κώδικα (OhmVision1.py) είναι σαφής. Ο φάκελος images μπορεί να χρησιμοποιηθεί τόσο για δοκιμαστική ανάλυση στατικών εικόνων όσο και για καταγραφή αποτελεσμάτων. Το README.md λειτουργεί ως σημείο αναφοράς για

οποιονδήποτε τρίτο (ή για τον μελλοντικό εαυτό του δημιουργού) θελήσει να τρέξει, να κατανοήσει ή να επεκτείνει το έργο.

2.3.2. Ροή Δεδομένων

Η εκτέλεση του προγράμματος αρχίζει με τη συνάρτηση **init(DEBUG)**, η οποία αποτελεί το σημείο εισόδου και διαμορφώνει το περιβάλλον εργασίας της εφαρμογής. Σε αυτή τη φάση, το σύστημα προβαίνει σε βασικές ενέργειες προετοιμασίας: αρχικά, πραγματοποιείται η εκκίνηση του υποσυστήματος λήψης εικόνας, με τη δημιουργία αντικειμένου Video Capture μέσω της βιβλιοθήκης OpenCV, επιτρέποντας τη ζωντανή απόκτηση καρέ (frames) από τη συνδεδεμένη κάμερα του υπολογιστή. Παράλληλα, γίνεται φόρτωση του προεκπαιδευμένου Haar Cascade Classifier από το σχετικό αρχείο XML, το οποίο βρίσκεται εντός του φακέλου cascade/. Σημαντικό στοιχείο είναι και η ύπαρξη επιλογής "debug mode", όπου το πρόγραμμα εμφανίζει διαδραστικά παράθυρα με sliders (trackbars), δίνοντας στον χρήστη τη δυνατότητα να ρυθμίσει παραμέτρους ανίχνευσης χρωμάτων σε πραγματικό χρόνο, διευκολύνοντας έτσι τη βελτιστοποίηση του αλγορίθμου σε διαφορετικές συνθήκες φωτισμού ή κάμερας.

Σε κάθε καρέ που λαμβάνεται από την κάμερα, ενεργοποιείται η συνάρτηση **findResistors(liveimg, rectCascade)**, η οποία αποτελεί το κεντρικό μηχανισμό εντοπισμού των αντιστάσεων στην εικόνα. Αρχικά, το καρέ μετατρέπεται σε γκρίζα κλίμακα, ώστε να ενισχυθεί η αντίθεση και να καταστεί η διαδικασία ανίχνευσης πιο ανθεκτική σε χρωματικό θόρυβο. Η επεξεργασμένη εικόνα στη συνέχεια τροφοδοτείται στον Haar Cascade Classifier, ο οποίος εντοπίζει γεωμετρικές περιοχές που είναι πιθανό να αντιστοιχούν σε αντιστάσεις, βασισμένος σε χαρακτηριστικά που έχουν εξαχθεί από πλήθος προτύπων κατά τη φάση εκπαίδευσης του μοντέλου. Προκειμένου να ελαχιστοποιηθούν τα ψευδώς θετικά αποτελέσματα (false positives), το σύστημα εφαρμόζει έναν επιπλέον έλεγχο ("δεύτερο πέρασμα") εντός της κάθε εντοπισμένης περιοχής, αυξάνοντας την αξιοπιστία του εντοπισμού. Οι περιοχές που περνούν και τα δύο στάδια θεωρούνται έγκυρες και καταγράφονται, μαζί με τη θέση και το αντίστοιχο τμήμα της εικόνας, ώστε να αποτελέσουν είσοδο για τα επόμενα στάδια ανάλυσης.

Αφού απομονωθούν οι περιοχές που περιέχουν αντιστάσεις, ακολουθεί η εφαρμογή της συνάρτησης **findBands(resistorInfo, DEBUG)**, η οποία αναλαμβάνει την ανίχνευση και ταξινόμηση των χρωματικών δακτυλίων πάνω στην κάθε αντίσταση. Σε αυτό το βήμα, η εικόνα της αντίστασης αρχικά μεγεθύνεται για να αυξηθεί η ακρίβεια εντοπισμού, ενώ εφαρμόζονται φίλτρα εξομάλυνσης και μορφολογικές πράξεις για τη βελτίωση της ευκρίνειας των ορίων των δακτυλίων. Η εικόνα μετατρέπεται στον χρωματικό χώρο HSV, ο οποίος προσφέρει πλεονεκτήματα σε ζητήματα διαχωρισμού χρωμάτων, καθώς διαχωρίζει πληροφορία

φωτεινότητας από την πληροφορία απόχρωσης και κορεσμού. Η διαδικασία ανίχνευσης χρωματικών ζωνών γίνεται με τη δημιουργία μασκών για κάθε πιθανό χρώμα, βάσει προκαθορισμένων ή δυναμικά ρυθμιζόμενων ορίων (thresholds), ενώ ακολουθεί εξαγωγή περιγραμμάτων (contours) και φιλτράρισμα με βάση τη γεωμετρία τους για να διασφαλιστεί ότι τα ανιχνευόμενα στοιχεία πράγματι αποτελούν δακτυλίους αντίστασης και όχι θόρυβο. Τελικώς, για κάθε έγκυρο δακτύλιο, καταγράφεται η θέση του και το αναγνωρισμένο χρώμα, ενώ η σειρά τους ταξινομείται με βάση τη χωρική τους διάταξη στην εικόνα.

Η διαδικασία ολοκληρώνεται με τη συνάρτηση **printResult(sortedBands, liveimg, resPos)**, η οποία είναι υπεύθυνη για την ερμηνεία των αποτελεσμάτων της προηγούμενης ανάλυσης και τον υπολογισμό της τιμής της αντίστασης. Εφόσον έχουν εντοπιστεί και ταξινομηθεί επιτυχώς τα χρώματα των δακτυλίων, η συνάρτηση ανασύρει τους αντίστοιχους αριθμητικούς κωδικούς σύμφωνα με τον διεθνή κώδικα χρωμάτων και συνθέτει το τελικό αποτέλεσμα, εφαρμόζοντας τον κατάλληλο πολλαπλασιαστή. Η υπολογισθείσα τιμή εμφανίζεται απευθείας πάνω στο καρέ της εικόνας, μαζί με σχετικό περίγραμμα γύρω από την αντίσταση, επιτρέποντας στον χρήστη να δει σε πραγματικό χρόνο το αποτέλεσμα της διαδικασίας. Σε περιπτώσεις όπου η ανίχνευση ή η αναγνώριση αποτύχει (π.χ. λόγω θορύβου, λάθους αριθμού δακτυλίων ή αποτυχίας ταξινόμησης), η περιοχή σημειώνεται με κόκκινο περίγραμμα, υποδηλώνοντας την ανάγκη επανάληψης ή αναπροσαρμογής των παραμέτρων ανίχνευσης.

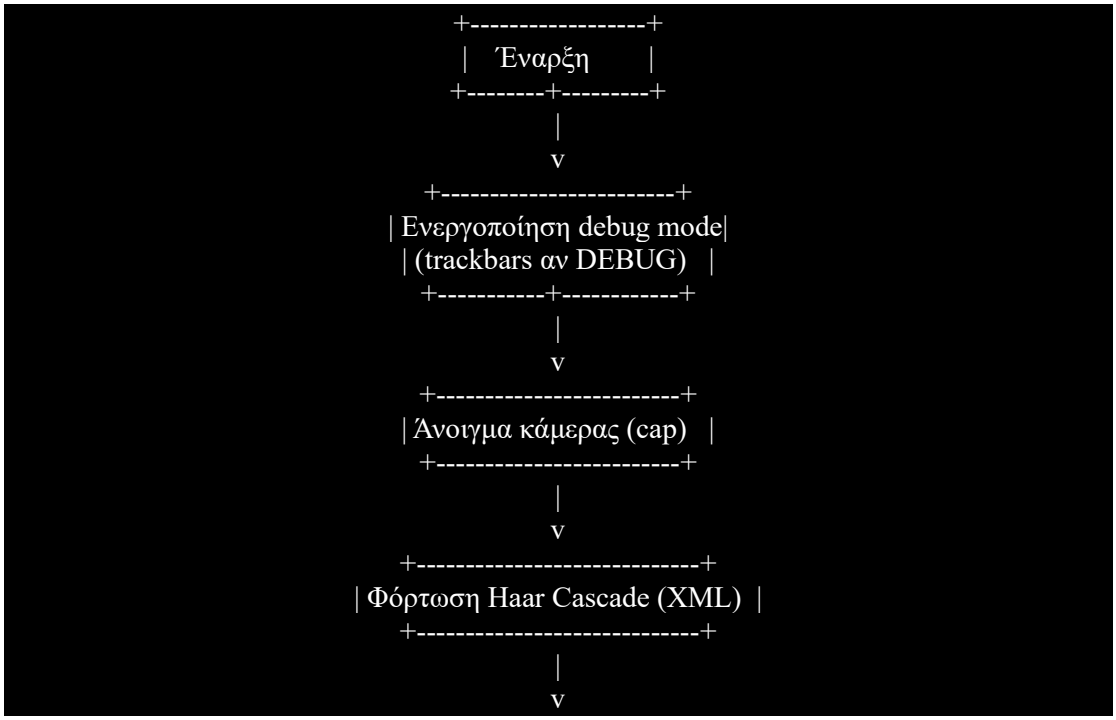
Η συνολική ροή των δεδομένων είναι αυστηρά ιεραρχημένη και ξεκινά από τη λήψη της εικόνας μέσω της κάμερας, συνεχίζεται με την επεξεργασία και ανίχνευση των περιοχών ενδιαφέροντος, ακολουθεί η ανάλυση και ταυτοποίηση των χρωματικών δακτυλίων και καταλήγει στην υπολογιστική εξαγωγή και προβολή της τελικής τιμής της αντίστασης. Ο διαχωρισμός των λειτουργικών σταδίων του αλγορίθμου σε αυτοτελείς συναρτήσεις επιτρέπει την ανεξάρτητη βελτιστοποίηση και συντήρηση κάθε επιμέρους τμήματος, ενώ διευκολύνει σημαντικά τον εντοπισμό σφαλμάτων και την εισαγωγή νέων μεθόδων, όπως τεχνητή νοημοσύνη, σε μελλοντικές εκδόσεις της εφαρμογής. Έτσι, η αρχιτεκτονική του αλγορίθμου υπηρετεί πλήρως τις αρχές της ευελιξίας, της επεκτασιμότητας και της διαφάνειας στη ροή των δεδομένων.

2.3.3 Αναλυτική επεξήγηση

Αρχικοποίηση περιβάλλοντος – Συνάρτηση init

Η συνάρτηση init αποτελεί το πρώτο λειτουργικό βήμα της εφαρμογής. Όταν ορίζεται η μεταβλητή DEBUG σε True, δημιουργεί ένα παράθυρο με όνομα "frame" και επτά sliders (trackbars), μέσω των οποίων ο χρήστης μπορεί να προσαρμόσει δυναμικά τα όρια (thresholds) που αφορούν τα χαρακτηριστικά του χρώματος HSV. Τα όρια αυτά αντιστοιχούν στα κατώτερα (lh, ls, lv) και ανώτερα (uh, us, uv) επιτρεπτά όρια των συνιστωσών Hue, Saturation και Value, αντίστοιχα. Αυτή η δυνατότητα είναι πολύτιμη σε περιβάλλοντα με διαφορετικές συνθήκες φωτισμού ή με κάμερες διαφορετικών χαρακτηριστικών.

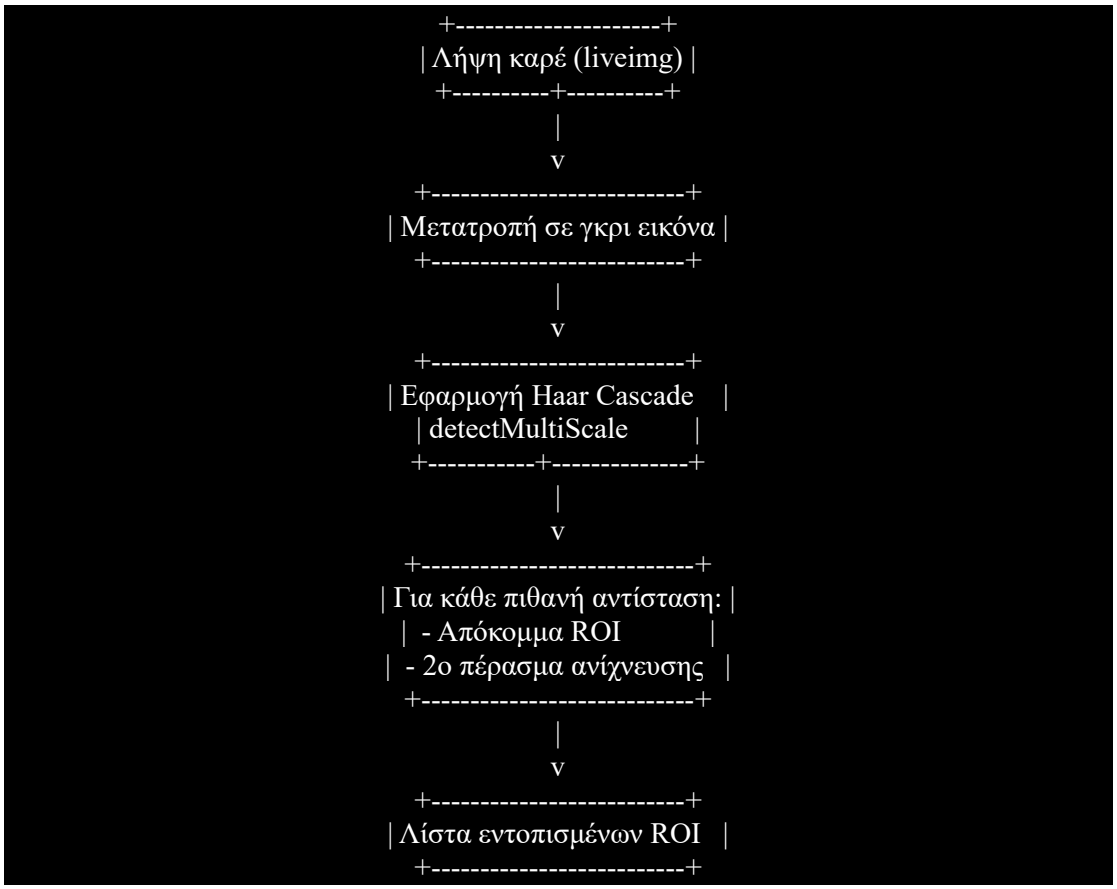
Αφού οριστούν τα παραπάνω, γίνεται λήψη του τρέχοντος path (όχι άμεσα απαραίτητο σε αυτή τη συνάρτηση, αλλά μπορεί να χρησιμοποιηθεί για ευελιξία στη διαχείριση αρχείων). Έπειτα, δημιουργείται το αντικείμενο cap που συνδέει το πρόγραμμα με την προεπιλεγμένη κάμερα του συστήματος (συνήθως η ενσωματωμένη webcam), και εισάγεται τεχνητή αναμονή 3 δευτερολέπτων (με τη χρήση της time.sleep), έτσι ώστε να δοθεί χρόνος στο hardware να ενεργοποιηθεί πλήρως πριν ξεκινήσει η λήψη εικόνας. Τέλος, το αρχείο Haar Cascade φορτώνεται στη μεταβλητή rectCascade, η οποία είναι υπεύθυνη για τον εντοπισμό αντιστάσεων στις εικόνες. Η συνάρτηση επιστρέφει δύο τιμές: το αντικείμενο κάμερας και το αντικείμενο ανιχνευτή.

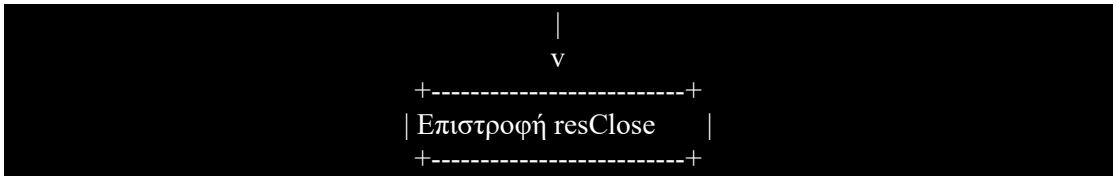


Εντοπισμός αντιστάσεων – Συνάρτηση findResistors

Η συγκεκριμένη συνάρτηση δέχεται ως είσοδο το τρέχον καρέ από την κάμερα και το μοντέλο Haar Cascade. Η εικόνα μετατρέπεται πρώτα σε γκρίζα κλίμακα (BGR2GRAY), καθώς η ανίχνευση αντικειμένων βασίζεται κυρίως στη γεωμετρία και όχι στα χρώματα, διευκολύνοντας τη διαδικασία και μειώνοντας τον υπολογιστικό φόρτο. Η μετατροπή του τύπου δεδομένων σε uint8 διασφαλίζει συμβατότητα με τις απαιτήσεις των συναρτήσεων του OpenCV.

Στη συνέχεια, εφαρμόζεται ο ανιχνευτής Haar Cascade (detectMultiScale), ο οποίος επιστρέφει λίστα με τα ορθογώνια που περιέχουν πιθανές αντιστάσεις. Για κάθε ορθογώνια περιοχή (που ορίζεται από τις συντεταγμένες x, y, w, h), αποκόπτεται η αντίστοιχη περιοχή τόσο από τη γκρίζα όσο και από την έγχρωμη εικόνα. Για περαιτέρω μείωση των ψευδώς θετικών αποτελεσμάτων, εφαρμόζεται δεύτερος έλεγχος εντός του ROI (region of interest) με πιο αυστηρές παραμέτρους. Εάν σε αυτή τη δεύτερη ανίχνευση επιστραφεί τουλάχιστον ένα αποτέλεσμα, το ROI προστίθεται στη λίστα των πιθανών αντιστάσεων. Η συνάρτηση τελικά επιστρέφει μια λίστα με τα κομμάτια της εικόνας που περιέχουν εντοπισμένες αντιστάσεις και τις αντίστοιχες θέσεις τους.



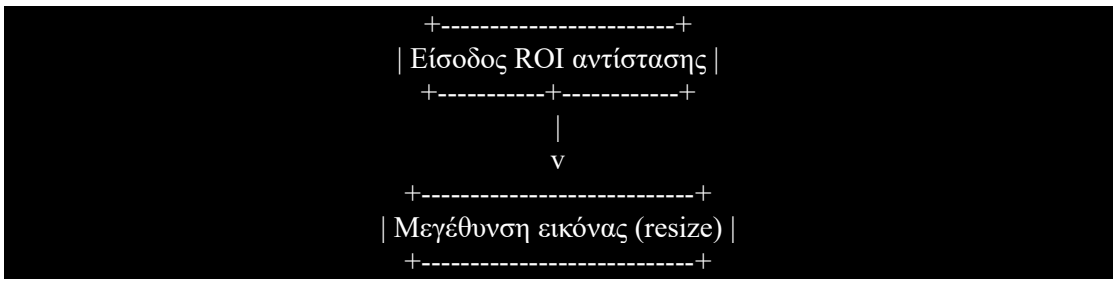


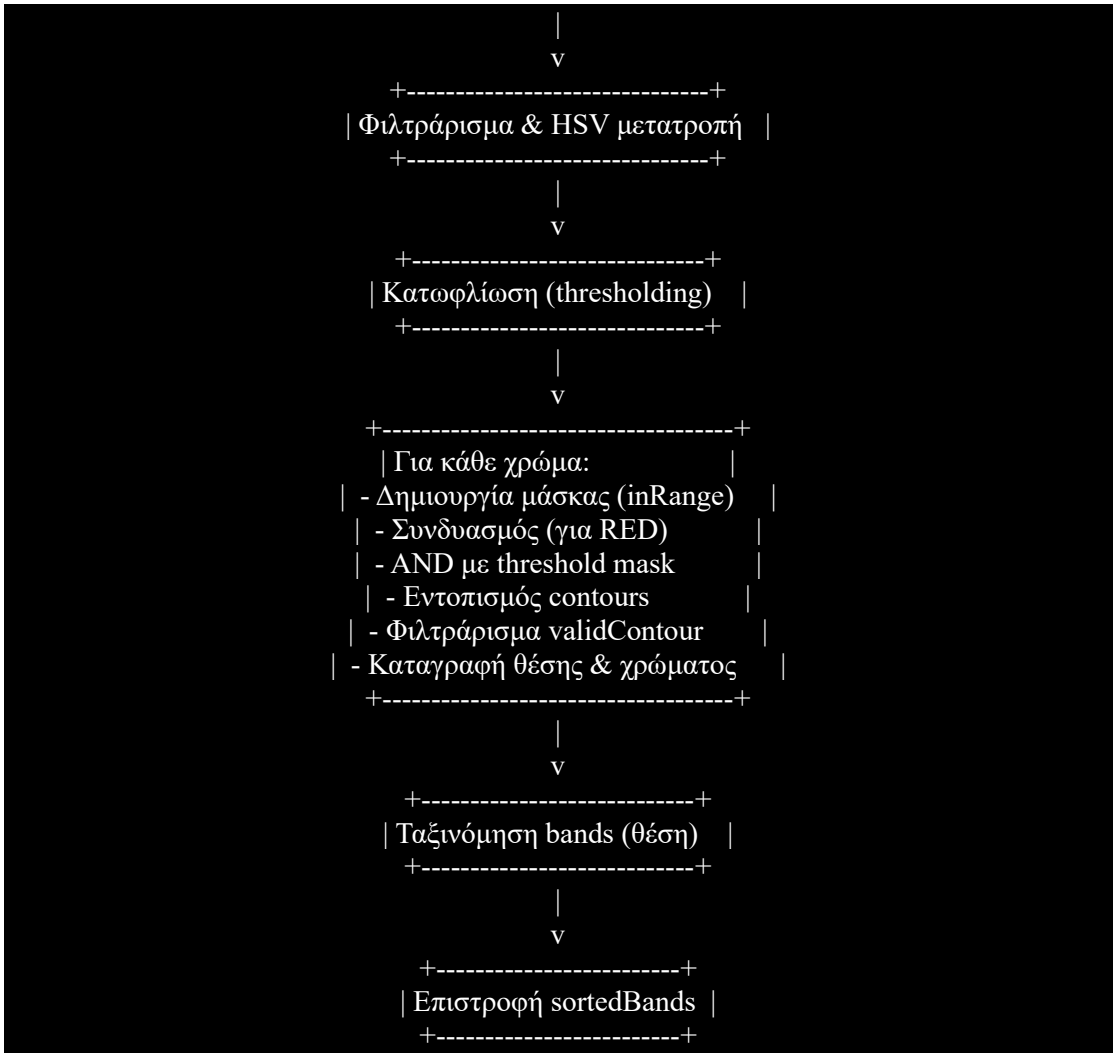
Ανίχνευση χρωματικών δακτυλίων – Συνάρτηση findBands

Η συνάρτηση αυτή δέχεται ως είσοδο την εικόνα της αντίστασης και πληροφορία για το αν βρίσκεται σε debug mode. Σε περίπτωση debug, ανακτά τις τρέχουσες τιμές των sliders που ρυθμίζουν τα όρια HSV για το χρώμα που επιθυμεί ο χρήστης να ανιχνεύσει. Η είσοδος εικόνας μεγεθύνεται για μεγαλύτερη ακρίβεια, και στη συνέχεια εφαρμόζεται bilateral filter, που βοηθά στη διατήρηση των άκρων παράλληλα με την εξομάλυνση θορύβου. Η εικόνα μετατρέπεται στον χρωματικό χώρο HSV, όπου τα χρώματα διαχωρίζονται πιο αποδοτικά σε σχέση με το RGB/BGR.

Ακολουθεί προσαρμοστική κατωφλίωση της εικόνας σε γκρίζα κλίμακα (adaptiveThreshold), ώστε να δημιουργηθεί μάσκα που ενισχύει τις περιοχές ενδιαφέροντος, και το αποτέλεσμα αντιστρέφεται (bitwise_not), καθώς οι ζώνες στις αντιστάσεις συνήθως προβάλλονται ως σκοτεινές περιοχές. Για κάθε ορισμένο χρώμα (ή για το ένα επιλεγμένο χρώμα σε debug mode), δημιουργείται μάσκα μέσω της cv2.inRange, η οποία επιστρέφει δυαδική εικόνα όπου μόνο τα pixels που ανήκουν στο συγκεκριμένο χρωματικό διάστημα εμφανίζονται ως λευκά.

Ειδική μεταχείριση γίνεται για το κόκκινο (RED), καθώς το HSV χρωματικό διάστημα για κόκκινο εκτείνεται και στις δύο άκρες του κύκλου Hue· για τον λόγο αυτό, γίνεται bitwise OR δύο μασκών ώστε να περιλαμβάνονται και τα δύο εύρη. Η μάσκα συνδυάζεται με το threshold mask για περαιτέρω απομόνωση των έγκυρων περιοχών. Στη συνέχεια, αναζητούνται τα περιγράμματα μέσω της findContours και φιλτράρονται μέσω της βοηθητικής validContour ώστε να απορριφθούν μικρές ή μη κανονικές περιοχές. Για κάθε έγκυρο δακτύλιο, καταγράφεται η θέση του αριστερότερα σημείου, η ονομασία του χρώματος και ο αντίστοιχος κωδικός, και οπτικοποιείται με κύκλο επάνω στην επεξεργασμένη εικόνα. Τέλος, τα αποτελέσματα επιστρέφονται ταξινομημένα ως προς τη θέση τους.



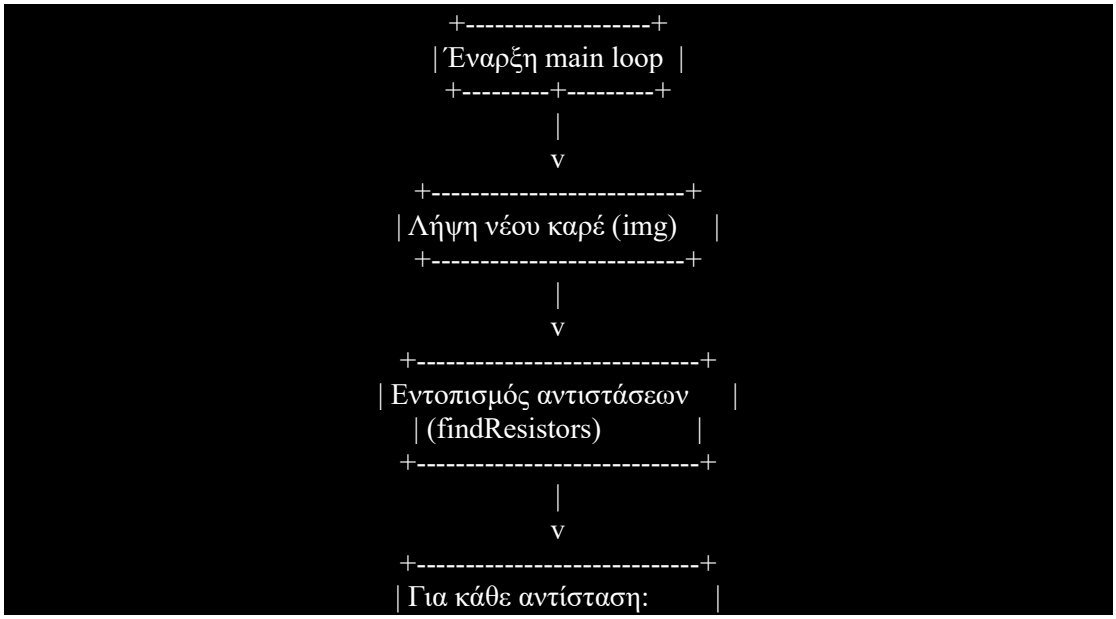


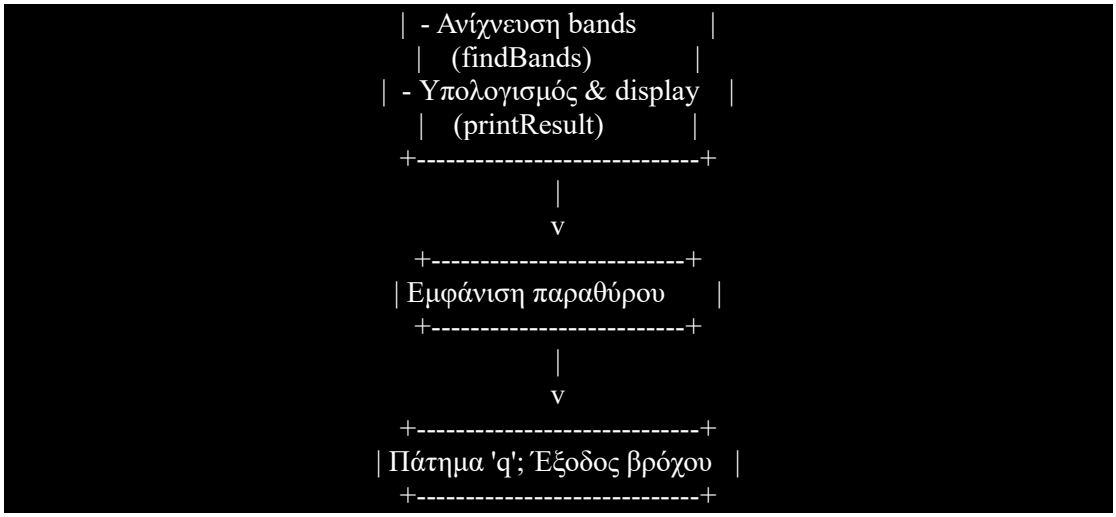
Υπολογισμός τιμής αντίστασης – Συνάρτηση printResult

Η συνάρτηση αυτή αναλαμβάνει να υπολογίσει το τελικό αποτέλεσμα – την τιμή της αντίστασης – και να το οπτικοποιήσει στο καρέ της κάμερας. Πρώτα, ανακτά τη θέση της αντίστασης στην εικόνα. Αν ο αριθμός των ανιχνευμένων δακτυλίων βρίσκεται εντός των αποδεκτών ορίων (3, 4 ή 5), τότε συνενώνει τους αριθμητικούς κωδικούς των ζωνών (πλην του τελευταίου, που αποτελεί πολλαπλασιαστή), σχηματίζει τον αριθμητικό χαρακτήρα της τιμής και τον πολλαπλασιάζει με την αντίστοιχη δύναμη του δέκα. Η περιοχή της αντίστασης περιβάλλεται με πράσινο περίγραμμα και εμφανίζεται η τιμή (σε OHMS) στην εικόνα, σε ευδιάκριτη θέση. Σε περίπτωση αποτυχίας αναγνώρισης, η περιοχή σημειώνεται με κόκκινο περίγραμμα, δίνοντας οπτική ένδειξη σφάλματος στον χρήστη.

Κύριος βρόχος εκτέλεσης (Main Loop)

Ο βασικός βρόχος εκτέλεσης της εφαρμογής ξεκινά με την αρχικοποίηση της κάμερας και του ανιχνευτή. Σε κάθε επανάληψη του βρόχου, λαμβάνεται νέο καρέ από την κάμερα. Στο καρέ αυτό εφαρμόζεται η ανίχνευση αντιστάσεων. Για κάθε εντοπισμένη αντίσταση, ανιχνεύονται οι χρωματικοί δακτύλιοι και υπολογίζεται η τιμή. Όλα τα αποτελέσματα προβάλλονται σε παράθυρο, επιτρέποντας στον χρήστη να παρακολουθεί ζωντανά το αποτέλεσμα. Ο βρόχος συνεχίζει να εκτελείται έως ότου ο χρήστης πατήσει το πλήκτρο 'q', οπότε η κάμερα απελευθερώνεται και τα παράθυρα κλείνουν.




| - Ανίχνευση bands |
| (findBands) |
| - Υπολογισμός & display |
| (printResult) |
+----------+
|
v
+----------+
| Εμφάνιση παραθύρου |
+----------+
|
v
+----------+
| Πάτημα 'q'; Έξοδος βρόχου |
+----------+

Κεφάλαιο 3ο: ΠΕΙΡΑΜΑΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ & ΑΠΟΤΕΛΕΣΜΑΤΑ

3.1 Ρύθμιση Πειράματος

Η πειραματική διαδικασία υλοποιήθηκε στο προγραμματιστικό περιβάλλον Python 3.6, με αξιοποίηση των βιβλιοθηκών OpenCV, NumPy και Imutils. Η συγκεκριμένη έκδοση της Python επιλέχθηκε λόγω πλήρους συμβατότητας με τις χρησιμοποιούμενες βιβλιοθήκες και τον υπάρχοντα κώδικα ανίχνευσης χρωματικών λωρίδων.

Ως μέσο συλλογής δεδομένων χρησιμοποιήθηκε μία συμβατική web camera. Σε αντίθεση με κλασικά πειραματικά σενάρια, όπου η κάμερα εστιάζει σε πραγματικά φυσικά αντικείμενα, στο πλαίσιο της παρούσας εργασίας η κάμερα τοποθετήθηκε σε σταθερή θέση και στράφηκε προς το σύνολο εικόνων αντιστάσεων που είχαν αποθηκευτεί τοπικά στον υπολογιστή. Με τον τρόπο αυτό δημιουργήθηκε ένα ελεγχόμενο σενάριο λήψης, όπου η κάμερα κατέγραφε τις προβεβλημένες εικόνες αντιστάσεων όπως θα κατέγραφε ένα πραγματικό φυσικό δείγμα.

Εικόνα 1: Web Camera που χρησιμοποιήθηκε για τη πραγματοποίηση των πειραμάτων

Η συγκεκριμένη μεθοδολογία επέτρεψε τη διατήρηση της τυπικής ροής του αλγορίθμου αναγνώρισης (καθώς ο αλγόριθμος «έβλεπε» δεδομένα μέσω της κάμερας), ενώ παράλληλα παρείχε τη δυνατότητα χρήσης ενός σταθερού dataset εικόνων. Με αυτόν τον τρόπο επιτεύχθηκε επαναληψιμότητα, αφού οι ίδιες εικόνες μπορούσαν να καταγραφούν ξανά και ξανά, αλλά και συγκρισιμότητα, δεδομένου ότι όλες οι δοκιμές πραγματοποιήθηκαν με ακριβώς τις ίδιες συνθήκες εισόδου.

Η απόσταση της κάμερας από την οθόνη και η γωνία λήψης διατηρήθηκαν σταθερές σε όλα τα πειράματα, ώστε να αποφεύγονται παραμορφώσεις που θα επηρέαζαν την ανάλυση. Παράλληλα, οι συνθήκες φωτισμού του χώρου ελέγχθηκαν προσεκτικά, ώστε να ελαχιστοποιηθούν τυχόν αντανακλάσεις στην οθόνη που θα αλλοίωναν τα χρώματα των αντιστάσεων.

3.2 Εκτέλεση & Στιγμιότυπα

Μετά την ολοκλήρωση της ρύθμισης του πειράματος, πραγματοποιήθηκε η εκτέλεση του προγράμματος σε πραγματικό χρόνο. Η εκτέλεση έγινε μέσω του περιβάλλοντος Python 3.6, όπου ο αλγόριθμος φόρτωνε αρχικά τον ταξινομητή Haar Cascade για τον εντοπισμό των αντιστάσεων και, στη συνέχεια, εφάρμοζε τις διαδικασίες εντοπισμού και ανάλυσης των χρωματικών λωρίδων.

Η κάμερα, στραμμένη προς την οθόνη στην οποία προβάλλονταν οι εικόνες των αντιστάσεων, παρείχε στο σύστημα καρέ συνεχούς ροής (frames), τα οποία επεξεργάζονταν σε πραγματικό χρόνο. Για κάθε καρέ, το πρόγραμμα πραγματοποιούσε τα ακόλουθα βήματα:

1. **Ανίχνευση του σώματος της αντίστασης** μέσω του ταξινομητή Haar Cascade.
2. **Απομόνωση της περιοχής ενδιαφέροντος (ROI)**, ώστε να αφαιρεθεί το υπόλοιπο υπόβαθρο.
3. **Προεπεξεργασία της εικόνας**, με εφαρμογή φίλτρων και μετατροπή σε χρωματικό χώρο HSV.
4. **Εντοπισμός και καταγραφή των χρωματικών λωρίδων** μέσω κατάλληλων κατωφλίων (thresholds) και ανάλυσης των περιγραμμάτων (contours).
5. **Υπολογισμός της τελικής τιμής αντίστασης** σύμφωνα με τον κώδικα χρωμάτων και απεικόνιση της τιμής επάνω στην εικόνα με χρήση του OpenCV.

Κατά τη διάρκεια της εκτέλεσης, στο παράθυρο εξόδου εμφανίζονταν σε πραγματικό χρόνο τόσο το πλαίσιο εντοπισμού γύρω από τον αντιστάτη όσο και η εκτιμώμενη τιμή της αντίστασης σε Ωhms. Σε περιπτώσεις που η ανάλυση των λωρίδων δεν ήταν επιτυχής, το σύστημα παρείχε οπτική ένδειξη σφάλματος, επισημαίνοντας το αντικείμενο με κόκκινο περίγραμμα.

Στιγμιότυπα από την εκτέλεση της διαδικασίας καταγράφηκαν με σκοπό την τεκμηρίωση και την περαιτέρω ανάλυση. Τα στιγμιότυπα αυτά απεικονίζουν χαρακτηριστικά παραδείγματα σωστής αναγνώρισης, αλλά και περιπτώσεις

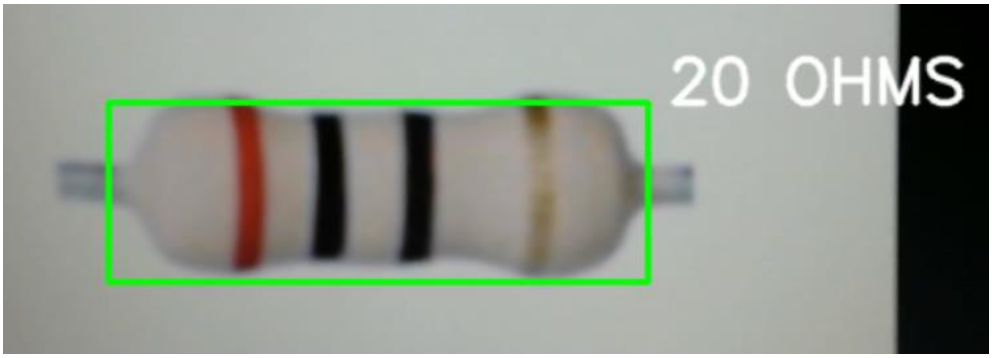
σφαλμάτων, αναδεικνύοντας τόσο τα πλεονεκτήματα όσο και τους περιορισμούς της μεθόδου. Μέσα από αυτήν την καταγραφή καθίσταται δυνατή η σύγκριση των αποτελεσμάτων υπό διαφορετικές συνθήκες φωτισμού, απόστασης και ποιότητας εικόνας.

Ακολουθούν ενδεικτικά παραδείγματα από την εκτέλεση της πειραματικής διαδικασίας.

3.2.1 Πρώτο Παράδειγμα

Στην επόμενη εικόνα παρουσιάζεται η ανίχνευση μιας αντίστασης με εκτιμώμενη τιμή 20 Ωhms. Ο αλγόριθμος εντόπισε με επιτυχία το σώμα της αντίστασης, γεγονός που φαίνεται από το πράσινο ορθογώνιο πλαίσιο γύρω από το εξάρτημα. Οι χρωματικές λωρίδες απομονώθηκαν και αναλύθηκαν, με αποτέλεσμα να εξαχθεί αριθμητική τιμή που εμφανίζεται στην έξοδο του προγράμματος.

Η περίπτωση αυτή δείχνει την ικανότητα του συστήματος να αναγνωρίζει σωστά αντιστάσεις μικρών τιμών, όπου οι λωρίδες είναι σχετικά ευδιάκριτες. Ωστόσο, παρατηρείται ότι η εικόνα παρουσιάζει ελαφρά θόλωση και διαφορά στον φωτισμό κατά μήκος του σώματος της αντίστασης, κάτι που ενδεχομένως δυσχεραίνει τη σαφή διάκριση όλων των χρωμάτων. Παρά τις μικρές αυτές αλλοιώσεις, το αποτέλεσμα που παρήχθη είναι ορθό, γεγονός που αποδεικνύει τη σταθερότητα του συστήματος σε συνθήκες μη ιδανικής απεικόνισης.



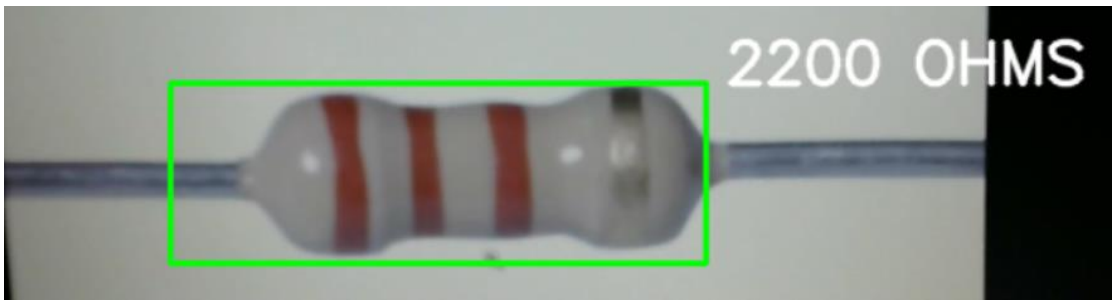
Εικόνα 2: Παράδειγμα εκτέλεσης 1

3.2.2 Δεύτερο Παράδειγμα

Στην εικόνα εμφανίζεται η αναγνώριση αντίστασης με εκτιμώμενη τιμή 2200 Ωhms (2.2 kΩ). Η παρουσία του πράσινου ορθογωνίου πλαισίου καταδεικνύει ότι το σύστημα εντόπισε και στη συγκεκριμένη περίπτωση με επιτυχία το σώμα της αντίστασης. Οι χρωματικές λωρίδες είναι

αρκετά διακριτές και απομονώθηκαν σωστά, οδηγώντας στον υπολογισμό της σωστής αριθμητικής τιμής.

Αξιοσημείωτο είναι ότι, παρά τη σχετικά απλή χρωματική αντίθεση (λευκό υπόβαθρο με κόκκινες λωρίδες), το σύστημα κατάφερε να αναγνωρίσει με σαφήνεια τις λωρίδες και να αποφύγει την πιθανότητα σύγχυσης με θόρυβο ή αντανακλάσεις. Το αποτέλεσμα αυτό αποδεικνύει ότι η μέθοδος μπορεί να αποδώσει αξιόπιστα σε περιπτώσεις όπου οι λωρίδες παρουσιάζουν υψηλή χρωματική καθαρότητα.

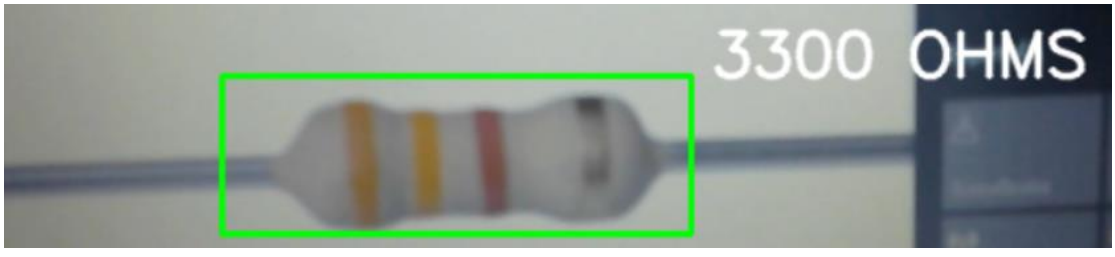


Εικόνα 3: Παράδειγμα εκτέλεσης 2

3.2.3 Τρίτο Παράδειγμα

Στην εικόνα φαίνεται η ανίχνευση μιας αντίστασης με υπολογισμένη τιμή 3300 Ωhms (3.3 kΩ). Η αντίσταση έχει εντοπιστεί επιτυχώς, όπως καταδεικνύεται από το πράσινο ορθογώνιο πλαίσιο που περικλείει το σώμα της. Η διαδικασία αναγνώρισης των χρωματικών λωρίδων κατέληξε σε τιμή που απεικονίζεται στην οθόνη, επιβεβαιώνοντας τη σωστή λειτουργία του αλγορίθμου.

Η εικόνα παρουσιάζει ορισμένες προκλήσεις: το σώμα της αντίστασης εμφανίζεται ελαφρώς θολό, ενώ η ένταση φωτισμού δεν είναι απόλυτα ομοιόμορφη, με αποτέλεσμα οι λωρίδες να μην είναι απόλυτα ευδιάκριτες. Παρά τις δυσκολίες αυτές, το σύστημα κατάφερε να αναγνωρίσει σωστά τις λωρίδες και να υπολογίσει τη σωστή τιμή.



Εικόνα 4: Παράδειγμα εκτέλεσης 3

3.3 Ζητήματα Συμβατότητας Εκδόσεων

Κατά την ανάπτυξη και εκτέλεση του συστήματος αναγνώρισης αντιστάσεων, παρουσιάστηκαν ζητήματα που σχετίζονται με τη συμβατότητα των εκδόσεων τόσο της γλώσσας προγραμματισμού Python όσο και των βιβλιοθηκών που χρησιμοποιήθηκαν.

Αρχικά, η υλοποίηση βασίστηκε στην Python 3.6, έκδοση η οποία θεωρείται πλέον παρωχημένη αλλά παραμένει απαραίτητη για τη συγκεκριμένη εφαρμογή λόγω συμβατότητας με παλαιότερες εκδόσεις του OpenCV. Ορισμένες νεότερες εκδόσεις του OpenCV (π.χ. από την 4.6 και άνω) δεν υποστηρίζουν πλέον την Python 3.6, γεγονός που καθιστά δυσχερή τη συντήρηση του έργου σε σύγχρονα περιβάλλοντα ανάπτυξης.

Ένα χαρακτηριστικό παράδειγμα ασυμβατότητας αφορά τη συνάρτηση cv2.findContours. Στις εκδόσεις OpenCV 3 η συνάρτηση αυτή επιστρέφει τρεις τιμές (εικόνα, περίγραμμα, ιεραρχία), ενώ στις εκδόσεις OpenCV 4 επιστρέφει δύο τιμές (περίγραμμα, ιεραρχία). Το γεγονός αυτό οδήγησε σε σφάλματα κατά την εκτέλεση του κώδικα, τα οποία έπρεπε να διορθωθούν με την ενσωμάτωση ελέγχου που να διαχειρίζεται και τα δύο πιθανά σενάρια.

Αντίστοιχα, ζήτημα ασυμβατότητας εντοπίστηκε και στη φόρτωση των Haar Cascades. Το αρχικό path που χρησιμοποιούσε ο κώδικας δεν ήταν σωστά ορισμένο για περιβάλλον Windows, με αποτέλεσμα ο ταξινομητής να μην φορτώνεται και το πρόγραμμα να καταρρέει. Η λύση δόθηκε με την αναθεώρηση του path σε συμβατή μορφή και με την προσθήκη ελέγχου εγκυρότητας (rectCascade.empty()), ώστε να εντοπίζεται άμεσα το πρόβλημα.

Τέλος, απαιτήθηκε προσαρμογή και στις εκδόσεις των βασικών βιβλιοθηκών. Για παράδειγμα, η βιβλιοθήκη NumPy σε εκδόσεις άνω της 1.20 δεν υποστηρίζει Python 3.6, γεγονός που κατέστησε αναγκαία την εγκατάσταση της έκδοσης 1.19.5. Παρόμοιοι περιορισμοί ίσχυσαν και για την Imutils, όπου απαιτήθηκε συγκεκριμένη έκδοση για να λειτουργεί απρόσκοπτα με τον υπόλοιπο κώδικα.

Κεφάλαιο 4ο: ΣΥΜΠΕΡΑΣΜΑΤΑ & ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

Η παρούσα εργασία επικεντρώθηκε στην ανάπτυξη ενός πρωτοτύπου συστήματος αναγνώρισης ηλεκτρονικών αντιστάσεων με βάση τον κώδικα χρωμάτων τους, μέσω της χρήσης υπολογιστικής όρασης. Ο στόχος ήταν διττός: αφενός, να εξεταστεί αν ένα σύστημα που βασίζεται αποκλειστικά σε παραδοσιακές τεχνικές επεξεργασίας εικόνας (OpenCV, HSV thresholds, Haar cascades) μπορεί να αναγνωρίσει αξιόπιστα τις χρωματικές λωρίδες, και αφετέρου, να αξιολογηθεί η ευχρηστία και η αποτελεσματικότητα της μεθόδου σε πραγματικές συνθήκες.

Τα αποτελέσματα που παρουσιάστηκαν στο Κεφάλαιο 3 επιβεβαίωσαν ότι το σύστημα κατάφερε να εντοπίσει και να αναγνωρίσει με επιτυχία ένα σημαντικό ποσοστό των δοκιμασμένων περιπτώσεων. Συγκεκριμένα, σε εικόνες με ευδιάκριτες λωρίδες και επαρκή φωτισμό, η αναγνώριση υπήρξε απολύτως επιτυχής, με την υπολογισθείσα τιμή της αντίστασης να συμπίπτει με την πραγματική. Οι δοκιμές αυτές καταδεικνύουν ότι ακόμη και με απλές τεχνικές, όπως η ανάλυση χρώματος στο χώρο HSV και η ανίχνευση περιγραμμάτων, είναι δυνατό να επιτευχθεί ένα αξιόπιστο αποτέλεσμα.

Παράλληλα, η χρήση web camera στραμμένης προς το σύνολο εικόνων αντιστάσεων που είχαν αποθηκευτεί στον υπολογιστή, απέδειξε ότι το σύστημα μπορεί να λειτουργήσει σε σενάρια ελεγχόμενων συνθηκών. Η μεθοδολογία αυτή προσέφερε επαναληψιμότητα και σταθερότητα στις δοκιμές, αφού τα ίδια δεδομένα μπορούσαν να επαναπροβληθούν και να αξιολογηθούν εκ νέου.

Ωστόσο, αναδείχθηκαν και ορισμένοι περιορισμοί. Σε περιπτώσεις όπου υπήρχαν αντανακλάσεις στην οθόνη, θόρυβος στην εικόνα ή λωρίδες με χαμηλή χρωματική αντίθεση, η ακρίβεια της αναγνώρισης μειωνόταν αισθητά. Η διαδικασία contour detection απέτυχε σε μερικές περιπτώσεις να απομονώσει σωστά όλες τις λωρίδες, οδηγώντας σε σφάλματα στον υπολογισμό της τελικής τιμής. Επίσης, παρατηρήθηκε ότι όταν η αντίσταση κατελάμβανε μικρό τμήμα του κάδρου, το εμβαδόν των λωρίδων δεν επαρκούσε ώστε να ξεχωρίσουν από το σώμα, με αποτέλεσμα αναγνώριση. Αυτά τα προβλήματα καταδεικνύουν την ισχυρή εξάρτηση του συστήματος από τις συνθήκες λήψης (φωτισμός, απόσταση, ανάλυση εικόνας).

Ένα ακόμη κρίσιμο ζήτημα αποτέλεσαν οι ασυμβατότητες εκδόσεων. Το πρόγραμμα αναπτύχθηκε στην Python 3.6, καθώς οι βιβλιοθήκες OpenCV και NumPy στις πιο πρόσφατες εκδόσεις τους δεν υποστηρίζουν πλέον αυτή την έκδοση. Προβλήματα, όπως η διαφορετική συμπεριφορά της συνάρτησης cv2.findContours σε OpenCV 3 και OpenCV 4, κατέστησαν

αναγκαία την τροποποίηση του κώδικα ώστε να υποστηρίζει και τα δύο σενάρια. Παράλληλα, η φόρτωση του Haar cascade απαίτησε διόρθωση των διαδρομών αρχείων, καθώς η αρχική μορφή του path δεν ήταν συμβατή με Windows. Όλα αυτά επιβεβαιώνουν ότι η διαχείριση εκδόσεων και η δημιουργία απομονωμένων περιβαλλόντων εκτέλεσης (π.χ. μέσω Conda) είναι απαραίτητη προϋπόθεση για την αναπαραγωγιμότητα και τη μακροπρόθεσμη βιωσιμότητα τέτοιων έργων.

Συνοψίζοντας, το σύστημα που υλοποιήθηκε μπορεί να χαρακτηριστεί ως ένα λειτουργικό πρωτότυπο με σαφείς ενδείξεις αξιοπιστίας, αλλά και με εμφανείς περιορισμούς. Απέδειξε ότι μπορεί να αναγνωρίσει σωστά τις τιμές αντιστάσεων σε ελεγχόμενες συνθήκες, αλλά χρειάζεται περαιτέρω βελτιώσεις ώστε να είναι ανθεκτικό σε πιο απαιτητικά σενάρια πραγματικής χρήσης.

Με βάση τις παραπάνω παρατηρήσεις, μπορούν να προταθούν οι ακόλουθες κατευθύνσεις για μελλοντική εργασία και βελτίωση του συστήματος:

1. **Ενσωμάτωση μεθόδων βαθιάς μάθησης**: Η χρήση συνελικτικών νευρωνικών δικτύων (CNNs) για την αναγνώριση χρωματικών λωρίδων θα μπορούσε να βελτιώσει σημαντικά την ακρίβεια, επιτρέποντας στο σύστημα να διαχειρίζεται με επιτυχία ποικίλες συνθήκες φωτισμού, γωνίες λήψης και αλλοιώσεις χρώματος.

2. **Αυτόματη ρύθμιση παραμέτρων**: Ανάπτυξη μηχανισμού που θα προσαρμόζει δυναμικά τα όρια HSV ανάλογα με τις συνθήκες φωτισμού, μειώνοντας την ανάγκη για χειροκίνητη βαθμονόμηση.

3. **Εμπλουτισμός του dataset**: Δημιουργία μιας εκτεταμένης βάσης εικόνων αντιστάσεων υπό διαφορετικές συνθήκες, που θα επιτρέψει τόσο την εκπαίδευση πιο εξελιγμένων μοντέλων όσο και την πιο αξιόπιστη αξιολόγηση της μεθόδου.

4. **Εφαρμογές σε φορητές συσκευές**: Προσαρμογή της μεθόδου σε εφαρμογή για smartphones ή tablets, ώστε να καταστεί δυνατή η αναγνώριση αντιστάσεων σε εκπαιδευτικά ή επαγγελματικά περιβάλλοντα χωρίς την ανάγκη εξωτερικού εξοπλισμού.

5. **Διαχείριση περισσότερων εξαρτημάτων**: Επέκταση της μεθόδου για την αναγνώριση και άλλων ηλεκτρονικών εξαρτημάτων, όπως πυκνωτών, διόδων ή τρανζίστορ, δημιουργώντας μια ολοκληρωμένη πλατφόρμα αυτόματης ταυτοποίησης.

6. **Βελτίωση διεπαφής χρήστη**: Ενσωμάτωση γραφικής διεπαφής που θα επιτρέπει στον χρήστη να αποθηκεύει αποτελέσματα, να συγκρίνει διαφορετικές μετρήσεις και να διαχειρίζεται τα δεδομένα με πιο εύχρηστο τρόπο.

7. **Αξιοποίηση τεχνικών οπτικής μέτρησης**: Διερεύνηση της δυνατότητας συνδυασμού με φυσικά όργανα μέτρησης (π.χ. πολύμετρα), ώστε να δημιουργηθεί ένα υβριδικό σύστημα που θα συγκρίνει τις οπτικές εκτιμήσεις με πραγματικές μετρήσεις.

Η μελλοντική έρευνα σε αυτούς τους άξονες θα μπορούσε να καταστήσει το προτεινόμενο σύστημα πιο ανθεκτικό, πιο ακριβές και πιο χρήσιμο για εφαρμογές στην εκπαίδευση, στη βιομηχανία αλλά και στην καθημερινή πρακτική των ηλεκτρονικών.

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΠΗΓΕΣ

[1]. Lee, R. S. (2020). AI fundamentals. In Artificial intelligence in daily life (pp. 19-37). Singapore: Springer Singapore.

[2]. Badillo, S., Banfai, B., Birzele, F., Davydov, I. I., Hutchinson, L., Kam-Thong, T., ... & Zhang, J. D. (2020). An introduction to machine learning. Clinical pharmacology & therapeutics, 107(4), 871-885.

[3]. Choi, R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F., & Campbell, J. P. (2020). Introduction to machine learning, neural networks, and deep learning. Translational vision science & technology, 9(2), 14-14.

[4]. Zhang, Z., Zhao, L., & Yang, T. (2021, August). Research on the application of artificial intelligence in image recognition technology. In Journal of Physics: Conference Series (Vol. 1992, No. 3, p. 032118). IOP Publishing.

[5]. Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. Computational intelligence and neuroscience, 2018(1), 7068349.

[6]. Soujanya, A., Goud, O. S. C., Prasad, K. S., & Reddy, G. P. (2017). Featured based pattern analysis using machine learning and artificial intelligence techniques for multiple featured dataset. Materials Today: Proceedings, 4(8), 9039-9048.

[7]. Chauhan, R., Ghanshala, K. K., & Joshi, R. C. (2018, December). Convolutional neural network (CNN) for image detection and recognition. In 2018 first international conference on secure cyber computing and communication (ICSCCC) (pp. 278-282). IEEE.

[8]. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

[9]. Park, N., & Kim, S. (2022). How do vision transformers work?. arXiv preprint arXiv:2202.06709.

[10]. Wang, X. (2018, March). A Review of Image Recognition Technology. In 2018 2nd International Conference on Artificial Intelligence: Technologies and Applications (ICAITA 2018) (pp. 24-28). Atlantis Press.

[11]. Kanal, L. N. (2003). Perceptron. In Encyclopedia of Computer Science (pp. 1383-1385).

[12]. Widrow, B., & Katz, E. P. (2025). Adaline. In Cognitive Memory: Human Memory| Machine Memory (pp. 25-28). Cham: Springer Nature Switzerland.

[13]. Cilimkovic, M. (2015). Neural networks and back propagation algorithm. Institute of Technology Blanchardstown, Blanchardstown Road North Dublin, 15(1), 18.

[14]. Ganis, M. D., Wilson, C. L., & Blue, J. L. (1998). Neural network-based systems for handprint OCR applications. IEEE Transactions on Image Processing, 7(8), 1097-1112.

[15]. Zhang, J., Yu, X., Lei, X., & Wu, C. (2022). A novel deep LeNet-5 convolutional neural network model for image recognition. Computer Science and Information Systems, 19(3), 1463-1480.

[16]. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[17]. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.

[18]. Verdhan, V. (2021). VGGNet and AlexNet networks. In Computer Vision Using Deep Learning: Neural Network Architectures with Python and Keras (pp. 103-139). Berkeley, CA: Apress.

[19]. Mohapatra, R. K., Shaswat, K., & Kedia, S. (2019, November). Offline handwritten signature verification using CNN inspired by inception V1 architecture. In 2019 Fifth International Conference on Image Information Processing (ICIIP) (pp. 263-267). IEEE.

[20]. Xu, W., Fu, Y. L., & Zhu, D. (2023). ResNet and its application to medical image processing: Research progress and challenges. Computer Methods and Programs in Biomedicine, 240, 107660.

[21]. Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. Journal of Big data, 3(1), 9.

[22]. Koonce, B. (2021). EfficientNet. In Convolutional neural networks with swift for Tensorflow: image recognition and dataset categorization (pp. 109-123). Berkeley, CA: Apress.

[23]. Fu, Z. (2022). Vision transformer: Vit and its derivatives. arXiv preprint arXiv:2205.11239.

[24]. Baldominos, A., Saez, Y., & Isasi, P. (2019). A survey of handwritten character recognition with mnist and emnist. Applied sciences, 9(15), 3169.

[25]. Hoiem, D., Divvala, S. K., & Hays, J. H. (2009). Pascal VOC 2008 challenge. World Literature Today, 24(1), 1-4.

[26]. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Cham: Springer International Publishing.

[27]. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139-144.

[28]. Song, J., Lee, S. B., & Park, A. (2020). A study on the industrial application of image recognition technology. The Journal of the Korea Contents Association, 20(7), 86-96.

[29]. Lim, H. (2021). Overview of image-based object recognition AI technology for autonomous vehicles. Journal of the Korea Institute of Information and Communication Engineering, 25(8), 1117-1123.

[30]. Santosh, K. C., Antani, S., Guru, D. S., & Dey, N. (Eds.). (2019). Medical imaging: artificial intelligence, image recognition, and machine learning techniques. CRC Press.

[31]. Rodriguez, M., Sridharlakshmi, N. R. B., Boinapalli, N. R., Allam, A. R., & Devarapu, K. (2020). Applying Convolutional Neural Networks for IoT Image Recognition. International Journal of Reciprocal Symmetry and Theoretical Physics, 7, 32-43.

[32]. Ferreira, B., & Reis, J. (2023). A systematic literature review on the application of automation in logistics. Logistics, 7(4), 80.

[33]. Wang, P., Fan, E., & Wang, P. (2021). Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. Pattern recognition letters, 141, 61-67.

[34]. Lai, Y. (2019, October). A comparison of traditional machine learning and deep learning in image recognition. In Journal of Physics: Conference Series (Vol. 1314, No. 1, p. 012148). IOP Publishing.

[35]. Okarma, K. (2020). Applications of computer vision in automation and robotics. *Applied sciences*, *10*(19), 6783.

[36]. Sheykhmousa, M., Mahdianpari, M., Ghanbari, H., Mohammadimanesh, F., Ghamisi, P., & Homayouni, S. (2020). Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 13, 6308-6325.

[37]. Naseer, M. M., Ranasinghe, K., Khan, S. H., Hayat, M., Shahbaz Khan, F., & Yang, M. H. (2021). Intriguing properties of vision transformers. Advances in Neural Information Processing Systems, 34, 23296-23308.

[38]. Pak, M., & Kim, S. (2017, August). A review of deep learning in image recognition. In 2017 4th international conference on computer applications and information processing technology (CAIPT) (pp. 1-3). IEEE.

[39]. Cilimkovic, M. (2015). Neural networks and back propagation algorithm. Institute of Technology Blanchardstown, Blanchardstown Road North Dublin, 15(1), 18.

[40]. Hanin, B. (2018). Which neural net architectures give rise to exploding and vanishing gradients?. Advances in neural information processing systems, 31.

[41]. Ying, X. (2019, February). An overview of overfitting and its solutions. In Journal of physics: Conference series (Vol. 1168, p. 022022). IOP Publishing.

[42]. Lejeune, E. (2020). Mechanical MNIST: A benchmark dataset for mechanical metamodels. Extreme Mechanics Letters, 36, 100659.

[43]. Li, X., Zhang, G., Huang, H. H., Wang, Z., & Zheng, W. (2016, August). Performance analysis of GPU-based convolutional neural networks. In 2016 45th International conference on parallel processing (ICPP) (pp. 67-76). IEEE.

[44]. Szegedy, C., Reed, S., Erhan, D., Anguelov, D., & Ioffe, S. (2014). Scalable, high-quality object detection. arXiv preprint arXiv:1412.1441.

[45]. Wang, G., Guo, Z., Wan, X., & Zheng, X. (2021, June). Study on image classification algorithm based on improved DENSENET. In Journal of Physics: Conference Series (Vol. 1952, No. 2, p. 022011). IOP Publishing.

[46]. Krishna, S. T., & Kalluri, H. K. (2019). Deep learning and transfer learning approaches for image classification. International Journal of Recent Technology and Engineering (IJRTE), 7(5S4), 427-432.

[47]. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[48]. González-Santoyo, C., Renza, D., & Moya-Albor, E. (2025). Identifying and Mitigating Label Noise in Deep Learning for Image Classification. Technologies, 13(4), 132.

[49]. Chen, X., Fang, H., Lin, T. Y., Vedantam, R., Gupta, S., Dollár, P., & Zitnick, C. L. (2015). Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325.

[50]. Sumit, S. S., Watada, J., Roy, A., & Rambli, D. R. A. (2020, April). In object detection deep learning methods, YOLO shows supremum to Mask R-CNN. In Journal of Physics: Conference Series (Vol. 1529, No. 4, p. 042086). IOP Publishing.

[51]. Zientara, P. A., Choi, J., Sampson, J., & Narayanan, V. (2018, January). Drones as collaborative sensors for image recognition. In 2018 IEEE International Conference on Consumer Electronics (ICCE) (pp. 1-4). IEEE.

[52]. Antonio, J. A., & Romero, M. (2018, December). Pedestrians' detection methods in video images: A literature review. In 2018 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 354-360). IEEE.

[53]. Fu'adah, Y. N., Pratiwi, N. C., Pramudito, M. A., & Ibrahim, N. (2020, December). Convolutional neural network (CNN) for automatic skin cancer classification system. In IOP conference series: materials science and engineering (Vol. 982, No. 1, p. 012005). IOP Publishing.

[54]. Agarwal, A., Aggarwal, K., Pichi, F., Meng, T., Munk, M. R., Bazgain, K., ... & Gupta, V. (2021). Clinical and multimodal imaging clues in differentiating between tuberculomas and sarcoid choroidal granulomas. American Journal of Ophthalmology, 226, 42-55.

[55]. Bali, A., & Mansotra, V. (2024). Analysis of deep learning techniques for prediction of eye diseases: A systematic review. Archives of Computational Methods in Engineering, 31(1), 487-520.

[56]. Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. " O'Reilly Media, Inc.".

[57]. Mordvintseva, Y. A., & Belitsky, A. A. (2021, January). Training the haar cascade classifier in sorting radio-electronic components. automating the collection of images for teaching. In 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus) (pp. 1019-1022). IEEE.

[58]. Kadiyala, A., & Kumar, A. (2017). Applications of Python to evaluate environmental data science problems. Environmental Progress & Sustainable Energy, 36(6), 1580-1586.

[59]. van Rossum, G., & Team, P. D. (2016). Python 3.6 Language Reference.

[60]. Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. nature, 585(7825), 357-362.

[61]. MAR-HERNÁNDEZ, P. G., IBARRA-ANGULO, P. L., GRIJALVA-ACUÑA, J. C., & ABRIL-GARCÍA, J. H. (2023). Image recognition software geometry with Python and OpenCV. Journal Computer Technology, 7(19).

[62]. Sushitha, S., & Sachin, C. D. Image Recognition using Python Opencv-Haar Cascade Classifiers.

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ

```
import cv2 #epexergasia eikonas kai video
import numpy as np # mathimatikes praxeis kai pinakes
import os #diaxerhsh paths
import imutils #eykolh epexergasia eikonas
import time Xrisimo gia kathysteriseis ston xrono
#eisagwgh bibliothikon

#metavlhth debugging
DEBUG = False

# Oria HSV gia diafora xromata (kato kai ano orio), mazi me onoma kai index
# I teleytaia timi einai to xroma se BGR gia emfanisi
COLOUR_BOUNDS = [
            [(0, 0, 0)      , (179, 255, 93)  , "BLACK"  , 0 , (0,0,0)       ],
            [(0, 90, 10)    , (15, 250, 100)  , "BROWN"  , 1 , (0,51,102)    ],
            [(0, 30, 80)    , (10, 255, 200)  , "RED"    , 2 , (0,0,255)     ],
            [(10, 70, 70)   , (25, 255, 200)  , "ORANGE" , 3 , (0,128,255)   ],
            [(30, 170, 100) , (40, 250, 255)  , "YELLOW" , 4 , (0,255,255)   ],
            [(35, 20, 110)  , (60, 45, 120)   , "GREEN"  , 5 , (0,255,0)     ],
            [(65, 0, 85)    , (115, 30, 147)  , "BLUE"   , 6 , (255,0,0)     ],
            [(120, 40, 100) , (140, 250, 220) , "PURPLE" , 7 , (255,0,127)   ],
            [(0, 0, 50)     , (179, 50, 80)   , "GRAY"   , 8 , (128,128,128) ],
            [(0, 0, 90)     , (179, 15, 250)  , "WHITE"  , 9 , (255,255,255) ],
            ];

Sympliromatiko orio HSV gia to kokkino (pou spaei se dyo perioxes sto HSV)
RED_TOP_LOWER = (160, 30, 80)
RED_TOP_UPPER = (179, 255, 200)

# Elaxisti epitrepti epifaneia perigrammatos
MIN_AREA = 700
# Grammatoseira pou tha xrisimopoiithei
FONT = cv2.FONT_HERSHEY_SIMPLEX
```

```
#required for trackbars
def empty(x):
    pass

# Arxikopoiisi kameras kai fortomatos cascade
def init(DEBUG):
# An eimaste se debug mode, ftiaxnoume trackbars gia na rythmizoume HSV oria xeirokinita
    if (DEBUG):
        cv2.namedWindow("frame")
        cv2.createTrackbar("lh", "frame",0,179, empty)
        cv2.createTrackbar("uh", "frame",0,179, empty)
        cv2.createTrackbar("ls", "frame",0,255, empty)
        cv2.createTrackbar("us", "frame",0,255, empty)
        cv2.createTrackbar("lv", "frame",0,255, empty)
        cv2.createTrackbar("uv", "frame",0,255, empty)
    tPath = os.getcwd() # Trexon fakelos
    cap = cv2.VideoCapture(0) # Anoigma kameras (0=proti, 1=deyteri kamera)
    time.sleep(3) # Mikri kathysterisi gia na xekinisei sosta i kamera
# Fortonoume Haar Cascade pou anixneyei antistaseis
    rectCascade=cv2.CascadeClassifier("C:Desktop\OhmVision-
master\cascade\haarcascade_resistors_0.xml")
    return (cap,rectCascade)

# Elegxos egkyrotitas perigrammatos (contour)
def validContour(cnt):
    # An to embadon einai mikrotero apo to elaxisto, aporriptetai
    if(cv2.contourArea(cnt) < MIN_AREA):
        return False
    else:
# Ypologismos aspect ratio (platos / ypsos)
        x,y,w,h = cv2.boundingRect(cnt)
        aspectRatio = float(w)/h
        if (aspectRatio > 0.4): # An einai poly "tetragono", den moiazei me daktylio
            return False
    return True
```

```
# Ypologismos kai emfanisi tis timis antistasis
def printResult(sortedBands, liveimg, resPos):
    x,y,w,h = resPos
    strVal = ""
# Dexomaste antistaseis me 3, 4 i 5 daktylious
if (len(sortedBands) in [3,4,5]):
# Pairnoume ola ektos tou teleytaiou (o teleytaios einai o pollaplasiastis)
        for band in sortedBands[:-1]:
            strVal += str(band[3]) # Prosthetoume ta psifia
        intVal = int(strVal)
        intVal *= 10**sortedBands[-1][3] # Pollaplasiazoume me basi to teleytaio band

# Sxediazoume prasino plaisio gyro apo tin antistasi
        cv2.rectangle(liveimg,(x,y),(x+w,y+h),(0,255,0),2)
# Emfanizoume tin timi se Om
    cv2.putText(liveimg,str(intVal)+"OHMS",(x+w+10,y),FONT,
1,(255,255,255),2,cv2.LINE_AA)
        return



    # An apotyxei i anagnorisi, sxediazoume kokkino plaisio
    cv2.rectangle(liveimg,(x,y),(x+w,y+h),(0,0,255),2)

# Xrisi Haar Cascade gia entopismo antistaseon sto kare
def findResistors(liveimg, rectCascade):
    gliveimg = cv2.cvtColor(cliveimg, cv2.COLOR_BGR2GRAY)
    gliveimg = gliveimg.astype('uint8')

    resClose = []

    # Anixneysi perioxon pou moiazoun me antistaseis
    ressFind = rectCascade.detectMultiScale(gliveimg,1.1,25)
    for (x,y,w,h) in ressFind: #SWITCH TO H,W FOR <CV3

        roi_gray = gliveimg[y:y+h, x:x+w]
        roi_color = cliveimg[y:y+h, x:x+w]
```

```
        # Deytero perasma anixneysis gia meiosi false positives
        secondPass = rectCascade.detectMultiScale(roi_gray,1.01,5)

        if (len(secondPass) != 0):
            resClose.append((np.copy(roi_color),(x,y,w,h)))
    return resClose

# Epexergasia kontinis eikonas antistasis gia anixneysi xromatikon daktylion
def findBands(resistorInfo, DEBUG):
    if (DEBUG):
# An eimaste se debug mode, pairnoume ta HSV oria apo ta trackbars
        uh = cv2.getTrackbarPos("uh","frame")
        us = cv2.getTrackbarPos("us","frame")
        uv = cv2.getTrackbarPos("uv","frame")
        lh = cv2.getTrackbarPos("lh","frame")
        ls = cv2.getTrackbarPos("ls","frame")
        lv = cv2.getTrackbarPos("lv","frame")
    # Allagi megethous eikonas antistasis
    resImg = cv2.resize(resistorInfo[0], (400, 200))
    resPos = resistorInfo[1]
    # Filtrarisma thorybou me bilateral filter + metatropi se HSV
    pre_bil = cv2.bilateralFilter(resImg,5,80,80)
    hsv = cv2.cvtColor(pre_bil, cv2.COLOR_BGR2HSV)
    # Dimiourgia maskas gia fonto/antistasi
    thresh=cv2.adaptiveThreshold(cv2.cvtColor(pre_bil,
cv2.COLOR_BGR2GRAY),255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BI
NARY,59,5)
    thresh = cv2.bitwise_not(thresh)

    bandsPos = []

    # An eimaste se debug mode, elegxoume mono to proto xroma
    if (DEBUG): checkColours = COLOUR_BOUNDS[0:1]
    else:       checkColours = COLOUR_BOUNDS
```

```
    for clr in checkColours:
        if (DEBUG):
# Dimiourgia maskas me ta trackbars
            mask = cv2.inRange(hsv, (lh,ls,lv),(uh,us,uv)) #use trackbar values
        else:
# Dimiourgia maskas me ta prokathorismena oria
            mask = cv2.inRange(hsv, clr[0], clr[1])
            if (clr[2] == "RED"): #combining the 2 RED ranges in hsv
# To kokkino xreiazetai 2 perioxes HSV
                redMask2 = cv2.inRange(hsv, RED_TOP_LOWER, RED_TOP_UPPER)
                mask = cv2.bitwise_or(redMask2,mask,mask)
             # Maska se syndyasmo me to threshold
        mask = cv2.bitwise_and(mask,thresh,mask=mask)
# Entopismos perigrammaton
        im2, contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

        # Filtrarisma egkyron perigrammaton
        for k in range(len(contours) -1,-1,-1):
            if (validContour(contours[k])):
# Pairnoume to pio aristero simeio tou daktyliou
                leftmostPoint = tuple(contours[k][contours[k][:,:,0].argmin()][0])
                bandsPos += [leftmostPoint + tuple(clr[2:])]
                cv2.circle(pre_bil, leftmostPoint, 5, (255,0,255),-1)
            else:
                contours.pop(k)
     # Sxediazoume ta perigrammata gia elegxo
        cv2.drawContours(pre_bil, contours, -1, clr[-1], 3)
        if(DEBUG):
            cv2.imshow("mask", mask)
            cv2.imshow('thresh', thresh)
# Emfanisi telikis eikonas me ta bands
    cv2.imshow('Contour Display', pre_bil)

    # Taxinomisi daktylion apo aristera pros dexia
    return sorted(bandsPos, key=lambda tup: tup[0])
```

```
# MAIN LOOP
cap,rectCascade = init(DEBUG)

while(not (cv2.waitKey(1) == ord('q'))): # Trexei mexri na patithei to "q"
    ret, cliveimg = cap.read()# Pairnoume frame apo tin kamera
    resClose = findResistors(cliveimg, rectCascade) # Entopizoume antistaseis

    for i in range(len(resClose)): # Gia kathe antistasi
        sortedBands = findBands(resClose[i],DEBUG) # Entopizoume daktylious
        printResult(sortedBands, cliveimg, resClose[i][1]) # Ypologizoume timi antistasis
    cv2.imshow("Frame",cliveimg) # Emfanizoume to kare
cap.release()
cv2.destroyAllWindows()
```