ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

<< Ανάπτυξη εφαρμογής καταγραφής οπτικού υλικού με τη χρήση κάμερας σε Raspberry pi4, αναγνώριση κειμένου και αναπαραγωγή του με χρήση text to speech στο χρήστη. >>

Του φοιτητή
Σπυρίδων Τσερεντζούλιας
Α.Μ.: 512189

Επιβλέπων Λέκτορας:
Κ. Άγγελος Γιακουμής

Ημερομηνία παρουσίασης:00/00/0000

ΔΗΛΩΣΗ

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 πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της πτυχιακής 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.

ΕΥΧΑΡΙΣΤΙΕΣ

Σε αυτή μου την προσπάθεια υπάρχουν πολλοί που προσέφεραν τον πολύτιμο χρόνο τους για χρήσιμες υποδείξεις και για τη διεξαγωγή της έρευνας. Θα ήθελα να ευχαριστήσω τον υπεύθυνο καθηγητή Κ. Άγγελο Γιακουμή καθώς και όλους όσους βοήθησαν. Συγκεκριμένα:
Τους γονείς μου για την στήριξη και τη βοήθεια τους στη τροποποίηση της μάσκας και τους Νίκο Ραφαήλ Πέτρο, Εμμανουήλ Πλούμη και Δημήτριο Τζαβίδα για την σχεδίαση και υλοποίηση του πρωτότυπου της Μάσκας.

ΠΕΡΙΛΗΨΗ

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μιας εφαρμογής, η οποία θα μπορεί να μετατρέψει όποιες λέξεις μπορεί να αναγνωρίσει με την κάμερα που χρησιμοποιεί, σε ηχητικό μέσο το οποίο θα μεταδίδεται στον χρήστη. Η εφαρμογή είναι γραμμένη σε C++, λόγω της ικανότητάς της να εκτελεί έργο, σε καλύτερους χρόνους σε σχέση με άλλες γλώσσες. Χρησιμοποιώ Docker ώστε να έχω ένα περιβάλλον ξεχωριστό, για την αποφυγή προβλημάτων που έχουν να κάνουν με τις βιβλιοθήκες που εμπεριέχω στην εφαρμογή μου και την συμβατότητα αυτών με οποιοδήποτε λειτουργικό σύστημα. Δύο από τις βιβλιοθήκες που χρησιμοποιώ είναι η Open CV (Computer Vision) με την οποία γίνεται η λήψη εικόνων, και η Tesseract OCR (Optical Character Recognition) για τη λήψη των χαρακτήρων από τις εικόνες. Η εφαρμογή αποθηκεύει εικόνες και κείμενα σε αρχεία τα οποία διαβάζει, και έπειτα αναλύει και μετατρέπει σε ηχητικό υλικό όπου αναπαράγεται στο χρήστη. Το υλικό που απαιτείται για να επιτευχθεί η εφαρμογή, είναι ο μικροϋπολογιστής Raspberry Pi 4, με λειτουργικό σύστημα Raspberry Pi OS, και ένα Raspberry Pi Camera Module 2 ή Raspberry Pi Camera Module 2 NoIR.

Name: Spyridon Tserentzoulias

Subject Area: Development of an application for recording visual material using a camera on Raspberry pi4, with text recognition and playback using text to speech to the user

Abstract

The goal of this thesis is to develop an application, capable of utilizing sources such as photos or text, made available from the hardware provided, in order to produce an audio media which will be played to the user. The application was written in C++ since its capabilities to minimize time and resources needed in comparison with other programming languages. Docker was used in a separate environment, in order to not tangle the libraries implemented in my application, and the compatibility aspect with a different operating system. Two examples of libraries I use are Open CV (Computer Vision) which is needed to subtract images, and Tesseract OCR (Optical Character Recognition) for the character collection from any given image. The application saves pictures and documents in a preferably format, in order to process them and construct an audio media for the end user. The needed hardware is a Raspberry Pi 4, with Raspberry Pi OS installed, and a Raspberry Pi Camera Module 2 or Raspberry Pi Camera Module2NoIR.

Περιεχόμενα

Κατάλογος Σχημάτων

1. Εισαγωγή

Η ταχεία ανάπτυξη διαφόρων τομέων της τεχνολογίας, ιδίως του τομέα της πληροφορικής, βοήθησε στην ανάπτυξη συστημάτων και εργαλείων αυτοματισμού που σχετίζονται με τη φωνή. Αυτά τα συστήματα που χειρίζονται την άμεση ανθρώπινη επικοινωνία πρέπει να αναπαράγουν τις μεθόδους επικοινωνίας τους ώστε να είναι εύχρηστα.

Τις τελευταίες δεκαετίες, η ανάπτυξη συστημάτων που σχετίζονται με τη φωνητική αλληλεπίδραση ανθρώπου-μηχανής έχει αυξηθεί σημαντικά. Αυτή η αλληλεπίδραση περιλαμβάνει τη μετάδοση μηνυμάτων από άνθρωπο σε μηχανή, δηλαδή, τη μετατροπή της φυσικής ομιλίας σε κείμενο (αναγνώριση ομιλίας) και τη μετάδοση μηνυμάτων από μηχανή σε άνθρωπο, δηλαδή τη μετατροπή κειμένου σε σύνθετη ομιλία (ομιλία) σύνθεση).

Η τεχνολογία σύνθεσης ομιλίας έχει μόνο λίγα χρόνια ιστορίας, κατά τη διάρκεια της οποίας έχει αναπτυχθεί και ευημερήσει. Με την πρόοδο των υπολογιστών και της τεχνολογίας, χρησιμοποιείται όλο και περισσότερο σε νέους τομείς εφαρμογών. Επομένως, αποτελεί πρόκληση η εφαρμογή ενός αποτελεσματικού συστήματος σύνθεσης ομιλίας.

Γενικά, η "σύνθεση ομιλίας" αναφέρεται στην αυτόματη δημιουργία κυματομορφών σήματος ομιλίας, δηλαδή, αναφέρεται στη μετατροπή κειμένου εισόδου (που αποτελείται από λέξεις, φράσεις ή προτάσεις) σε κυματομορφές ομιλίας χρησιμοποιώντας έναν συγκεκριμένο αλγόριθμο και έναν συγκεκριμένο τύπο δεδομένων για κωδικοποίηση και αποθήκευση. Το κείμενο εισαγωγής μπορεί να προέρχεται από υπάρχουσα βάση δεδομένων ή σύστημα αναγνώρισης οπτικών χαρακτήρων.

1.1 Περιγραφή θέματος

Στην εργασία αυτή, ο σκοπός είναι η υλοποίηση μιας εφαρμογής, η οποία θα είναι εύκολη στη χρήση της και θα δίνει την δυνατότητα σε ανθρώπους με προβλήματα όρασης να «διαβάσουν» κείμενα και λέξεις που βρίσκονται σε επιφάνειες στο χώρο που βρίσκονται. Με τη βοήθεια της μηχανικής όρασης και της μηχανικής μάθησης ένας υπολογιστής μπορεί μετατρέψει μια φυσική εικόνα σε ψηφιακή πληροφορία, την οποία και μπορεί να χρησιμοποιήσει για να κάνει εξαγωγή των οπτικών χαρακτήρων, και στη συνέχεια να αναπαράξει την πληροφορία αυτή στο χρήστη με ακουστικό τρόπο. Με αυτό το τρόπο προσπερνάμε την ανάγκη εκμάθησης του κώδικα Braille σε άτομα με προβλήματα όρασης, και τους παρέχουμε μια εναλλακτική η οποία

δεν περιορίζεται σε κείμενα γραμμένα σε Braille, καθώς το πρόγραμμα δεν έχει τέτοιους περιορισμούς.

1.2 Περιγραφή τεχνολογιών και υλικού

Για την ολοκλήρωση της εφαρμογής χρησιμοποιήθηκε ο μικροϋπολογιστής Raspberry pi 4 με τοποθετημένη τη κάμερα Raspberry pi camera module. Για την λήψη και την επεξεργασία εικόνων σε πραγματικό χρόνο και την εξαγωγή των χαρακτήρων χρησιμοποιήθηκαν οι βιβλιοθήκες ανοιχτού κώδικα OpenCV και Tesseract OCR αντίστοιχα. Για την συμβατότητα και την εγγύηση καλής λειτουργίας σε διάφορα λειτουργικά συστήματα χρησιμοποιήθηκε το λογισμικό Docker και τέλος για την αναπαραγωγή του εξαγόμενου κειμένου σε ηχητικό μέσο χρησιμοποιήθηκε το λογισμικό Flite

2. Μηχανική Όραση

Μηχανική όραση είναι ο κλάδος της τεχνητής νοημοσύνης που επιχειρεί να μετατρέψει δεδομένα από έναν οπτικό αισθητήρα, όπως φωτογραφίες ή βίντεο, σε μια μορφή που μπορεί ο υπολογιστής να κατανοήσει και επεξεργαστεί.
Επειδή είμαστε οπτικά πλάσματα, είναι εύκολο να μας δημιουργηθεί η λανθασμένη εντύπωση ότι η μηχανική όραση είναι ένα πεδίο επιστήμης της τεχνητής νοημοσύνης, το οποίο δεν είναι σύνθετο αλλά σχετικά απλό και εύκολο στην χρήση του αλλά και στην τροποποίηση του. Έχουμε το προνόμιο να χειριζόμαστε ένα από τα καλυτέρα όργανα όρασης, μάλιστα δυο ταυτόχρονα.
Από τις αρχές του 19^{ου} αιώνα έως και την σήμερον ημέρα κατασκευάζουμε και χρησιμοποιούμε φωτογραφικές μηχανές που μπορούν να αποδώσουν οπτικά στιγμιότυπα είτε σε φυσικά μέσα, όπως γυαλί, φιλμ, χαρτί κτλ., είτε σε ηλεκτρονική μορφή. Για αυτό και η αρχική μας διαίσθηση για την τεχνητή όραση είναι παραπλανητική. Ο ανθρώπινος εγκέφαλος είναι από τους ισχυρότερους επεξεργαστές που υπάρχουν στον πλανήτη και είναι υπεύθυνος για την διαχείριση των ματιών αλλά και των δεδομένων που λαμβάνουμε από αυτά. Έχει ένα σύστημα αναγνώρισης το οποίο αναδεικνύει ποια είναι τα σημαντικά τμήματα της εικόνας για εξέταση και μειώνει την σημασία των άλλων τμημάτων. Αυτό που είναι δεδομένο είναι ότι η διαδικασία της όρασης αποτελείται από πολύ συνθέτες παραμέτρους κι η “μετάφραση” της σε κώδικα αποτελείται από τα δυσκολότερα εμπόδια της τεχνίτης νοημοσύνης.
Η ηλεκτρονική ερμηνεία του πραγματικού χώρου αποδίδεται σε διάφορες μορφές. Τέτοια δεδομένα μπορούμε να λάβουμε με διαφορά μέσα όπως, φωτογραφίες, βίντεο, προβολές από

μια η πολλαπλές κάμερες, πολυδιάστατα δεδομένα από ιατρικό σαρωτή η ακόμη και από ηχοεντοπιστικα συστήματα σόναρ, και με αυτά τα στοιχεία εφαρμόζουμε τις θεωρίες και τα μοντέλα της μηχανικής όρασης με σκοπό την κατασκευή συστημάτων υπολογιστικής όρασης. Στην παρών εφαρμογή θα χρησιμοποιήσω τη βιβλιοθήκη OpenCV για την μετάφραση των δεδομένων και μια απλή κάμερα για την αποθήκευση τους.

3. Ενοιολογήσεις

Η τεχνολογική πρόοδος κατέστησε δυνατή την ανάπτυξη μεθόδων επικοινωνίας μεταξύ ανθρώπων και αντικειμένων, διευκολύνοντας τη ροή πληροφοριών τόσο από άποψη ταχύτητας όσο και από άποψη ασφάλειας. Το Διαδίκτυο των πραγμάτων (IoT) χρησιμοποιεί τέτοιου είδους πρόοδο και ενσωμάτωση νέων στοιχείων. Ως μέρος των συστημάτων πληροφοριών, είναι δυνατή η απομακρυσμένη πρόσβαση και διάφοροι έλεγχοι συστήματος.

Σύμφωνα με την επικρατούσα φιλοσοφία της συνεχούς επικοινωνίας, συμπεριλαμβανομένης της επικοινωνίας μεταξύ μηχανών και μηχανών, το IoT μπορεί να οριστεί ως *ένα σύνολο τεχνολογιών σχεδιασμένων να επιτρέπουν τη σύνδεση ετερογενών αντικειμένων μέσω διαφόρων δικτύων και μεθόδων επικοινωνίας*. Ο κύριος στόχος είναι η τοποθέτηση έξυπνων συσκευών σε διαφορετικές τοποθεσίες για τη λήψη, αποθήκευση και διαχείριση πληροφοριών, καθιστώντας τις προσβάσιμες, σε οποιονδήποτε οπουδήποτε στον κόσμο

Ο επιστημονικός τομέας του υπολογιστή ο οποίος αναγνωρίσει πρόσωπα είναι σήμερα πιο δημοφιλής στην υπόθεση διαφόρων εφαρμογών στην καθημερινή ζωή. Είναι ήδη πρότυπο για την αναγνώριση ή τον εντοπισμό ενός ή περισσότερων προσώπων ή μεμονωμένων αντικειμένων.

Στη διαδικασία αναγνώρισης υπάρχουν 3 φάσεις: ανίχνευση, εξαγωγή και ταξινόμηση.

Η ανίχνευση χρησιμοποιείται για τον εντοπισμό αντικειμένων σε εικόνες. Η ανίχνευση προσώπου σε πραγματικό χρόνο με πολλά άλλα αντικείμενα στο παρασκήνιο δεν είναι ένα απλό πρόβλημα. Υπάρχουν καταστάσεις στις οποίες δεν είναι δυνατή η λήψη του προσώπου ενός αντικειμένου λόγω περιστροφής του προσώπου περισσότερο από περίπου 30 °, ή λόγω αλλαγής του φωτισμού, άλλοι παράγοντες που ενδέχεται να εμποδίσουν την εμφάνιση του προσώπου, όπως, γένια, γυαλιά.

Η δεύτερη φάση είναι η εξαγωγή, και αυτό καθορίζεται από ορισμένες προδιαγραφές και απαιτεί συγκεκριμένα χαρακτηριστικά για την ανίχνευση ενός αντικειμένου. Αυτές οι δυνατότητες χρησιμοποιούνται αργότερα για ταξινόμηση.

Η ταξινόμηση είναι η τελευταία φάση της διαδικασίας αναγνώρισης, στην οποία χρησιμοποιείται κυρίως στην ταξινόμηση των συναισθηματικών καταστάσεων όταν το αντικείμενο της ανίχνευσης είναι ένα ανθρώπινο πρόσωπο.

Τα τελευταία χρόνια, σημαντική βιβλιογραφία έχει εμφανιστεί να ασχολείται και να περιγράφει λεπτομερώς τις διάφορες φάσεις της διαδικασίας αναγνώρισης

Ο ερευνητικός τομέας της όρασης υπολογιστή χρησιμοποιείται ευρέως στην HSH. Το ηλεκτρονικό πλαίσιο όρασης εικόνωνη προσώπων με τη μορφή HSH παρέχει διάφορες λύσεις με τη μορφή υπηρεσιών υγείας και ασφάλειας. Πρόσφατα, έγινε επίσης συνώνυμο του μάρκετινγκ και της ψυχαγωγίας

Ωστόσο, τα δύο πεδία αναγνώρισης προσώπου και το IoT δεν είναι προς το παρόν συνδεδεμένα. Όσον αφορά τη βιβλιογραφία που ασχολείται με την ανίχνευση προσώπου στο IoT, υπάρχουν πολύ λίγες σχετικές πηγές

Το πρόβλημα έχει μεγάλες δυνατότητες, καθώς τα θέματα του IoT είναι επί του παρόντος ένα «*σημαντικό θέμα*» συζήτησης.[5] Επομένως, παρέχεται τουλάχιστον μια μερική επισκόπηση των πιο πρόσφατων ερευνητικών έργων.

Σύμφωνα με τους Kolias et al. (2010)[1] οι οποίοι ασχολήθηκαν με την αυτόματη παρακολούθηση μιας σκηνής στην HSH, πρέπει να εγκατασταθούν πολλές κάμερες σε μια δεδομένη περιοχή. Ωστόσο, πρέπει να είναι ανθεκτικές σε τρέχοντες περιορισμούς όπως θόρυβος, συνθήκες φωτισμού, ανάλυση εικόνας και υπολογιστικό κόστος. Για να ξεπεραστούν τέτοιοι περιορισμοί και να αυξηθεί η ακρίβεια αναγνώρισης, οι Kolias et al. χρησιμοποίησαν μια τεχνολογία αναγνώρισης ραδιοσυχνοτήτων, η οποία είναι ιδανική για τη μοναδική αναγνώριση αντικειμένων. Εξέτασαν τη δυνατότητα ενσωμάτωσης RFID με ημισφαιρικές βιντεοκάμερες απεικόνισης. Τα πλεονεκτήματα και οι περιορισμοί κάθε τεχνολογίας και η ενσωμάτωσή τους υποδηλώνει, ωστόσο, ότι ο συνδυασμός τους θα μπορούσε να οδηγήσει σε μια ισχυρή ανίχνευση αντικειμένων και τις αλληλεπιδράσεις τους μέσα σε ένα περιβάλλον. Η παρούσα εργασία ολοκληρώνεται με παρουσίαση ορισμένων πιθανών εφαρμογών αυτής της ολοκλήρωσης.

Η δομή αυτού του συστήματος αποτελείται από:

• Απλές συσκευές (π.χ. σε αυτήν την περίπτωση κάμερες HD) που λειτουργούν ως αισθητήρες και είναι σε θέση να συλλέγουν πληροφορίες σχετικά με την υγεία και το περιβάλλον του ασθενούς, εάν είναι απαραίτητο.

• Ένα στοιχείο απόφασης, δηλαδή ένας υπεύθυνος λήψης αποφάσεων με ισχυρότερους υπολογιστικούς πόρους, στην περίπτωση αυτή, το Raspberry Pi 4.

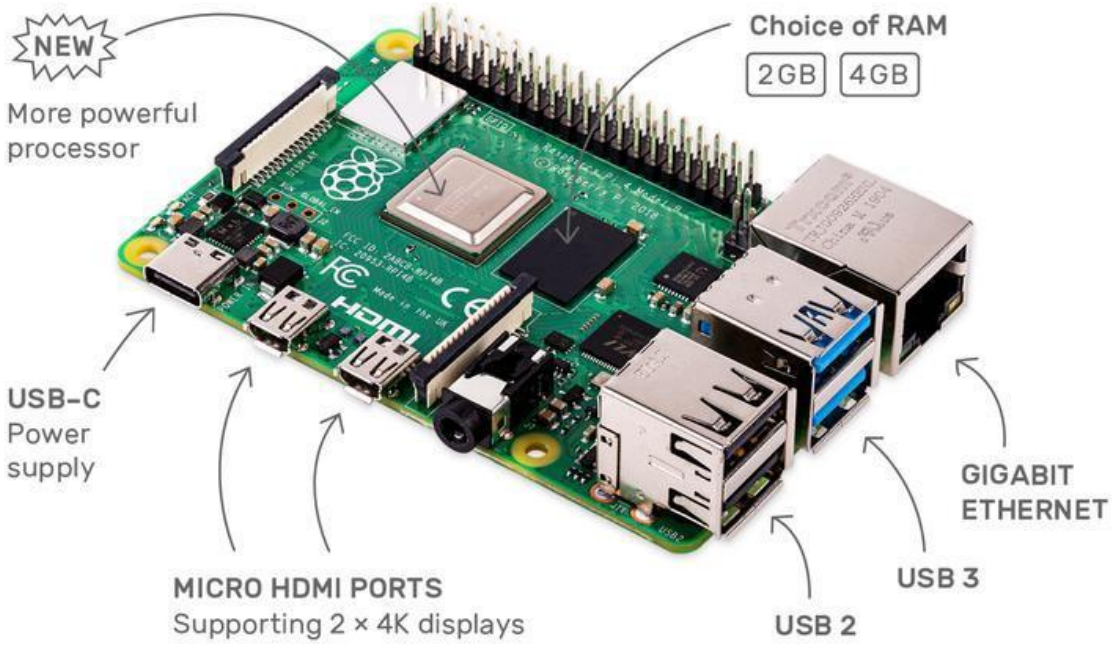
Οι αισθητήρες μπορούν να ανιχνεύσουν τη μη τυπική κίνηση ή πτώσεις και επίσης να συλλάβουν τα πρόσωπά τους. Όταν ένα άτομο μπαίνει στο δωμάτιο, οι αισθητήρες ανιχνεύουν

δραστηριότητα. Ανάλογα με τον τρόπο περιστροφής της κάμερας και τη λήψη του προσώπου, το άτομο μπορεί να αναγνωριστεί (οι συνθήκες εξαρτώνται από τον βαθμό περιστροφής του προσώπου, τις συνθήκες φωτός, την απόσταση της κάμερας κ.λπ.). Εάν το αναγνωρισμένο άτομο είναι ασθενής, η δραστηριότητα του ατόμου θα αρχίσει να παρακολουθείται

Η ανίχνευση προσώπου, η παρακολούθηση και η ανίχνευση κίνησης είναι ένα σημαντικό και δημοφιλές ερευνητικό θέμα στον τομέα της επεξεργασίας εικόνας. Το όραμα του υπολογιστή ως επιστημονική και τεχνολογική μελέτη ασχολείται με την ικανότητα των ηλεκτρονικών συσκευών να συλλέγουν πληροφορίες από μια ψηφιακή εικόνα, «*να κατανοούν μια κατάσταση*» και, συνεπώς, να αποφασίζουν εάν θα εκτελέσουν την εργασία ή όχι.

Αυτή η ποιότητα υπάρχει στην τεχνολογία που χρησιμοποιείται σε όλους τους τομείς της βιομηχανίας και της έρευνας. Εδώ, περιγράφεται ο μικρο-ελεγκτής Raspberry Pi 4 και το πακέτο OpenCV. Το OpenCV είχε ήδη χρησιμοποιηθεί σε προηγούμενα συστήματα. Από την άλλη πλευρά, ο σχεδιασμός του υλικού θα μπορούσε να περιγραφεί μαζί με τον κώδικα λογισμικού που είναι γραμμένος στο Python και η κύρια λειτουργία του είναι η αναγνώριση προσώπων και η ανίχνευση κίνησης. Έχουν δημιουργηθεί τρία προγράμματα για την ανίχνευση προσώπου και οι λειτουργίες τους μπορούν να συγκριθούν με βάση τους αλγόριθμους που χρησιμοποιήθηκαν. Η ταχύτητα και η ακρίβεια της αναγνώρισης προσώπου και κίνησης μπορούν επίσης να συγκριθούν.

Το Raspberry Pi είναι ένας υπολογιστής με ένα τσιπ συγκρίσιμο με έναν (ασθενέστερο) επιτραπέζιο υπολογιστή. Περιέχει μια θύρα για την οθόνη (HDMI) και μέσω μιας θύρας USB είναι δυνατή η σύνδεσή της σε ένα πληκτρολόγιο και ποντίκι. Έχουν ήδη αναπτυχθεί πολλές γενιές αυτού του υπολογιστή, με διαφορετική απόδοση και σκοπούμενη χρήση. Η CPU προέρχεται από την οικογένεια ARM, οπότε είναι συγκρίσιμη με ένα κοινό smartphone. Στο Raspberry Pi, μπορούμε να χρησιμοποιήσουμε διάφορες διανομές Linux, Windows 10 ή IoT Core, από τη Microsoft. Σε αντίθεση με το PC Arduino, είναι δυνατή η χρήση του Raspberry Pi όχι μόνο για διάφορους ελέγχους συσκευών (με επαφές GPIO), αλλά και για την ίδια τη σχετική ανάπτυξη εφαρμογών.



Σχήμα 3.1 – Raspberry pi 4 pcb

Χαρακτηριστηκά:

- **Quad core 64-bit ARM Cortex A53 with frequency 1.2 GHz, approx. 50% faster than Raspberry Pi 3**
- **802.11n Wireless LAN**
- **Bluetooth 4.1 (including Bluetooth Low Energy)**
- **400MHz Video Core IV multimedia**

Για τη δημιουργία ενός αυτόνομου συστήματος, χρησιμοποιείται το Raspberry Pi κάμερα module v2. Η κάμερα διαθέτει σαρωτή εικόνας Sony IMX219 υψηλής ποιότητας 8 megapixel. Από την άποψη των στατικών εικόνων, η κάμερα είναι σε θέση να δημιουργεί 3280 x 2464 σημεία εικόνας στατικών εικόνων, υποστηρίζει επίσης 1080p σε 30 fps, ανάλυση 720p σε 60 fps και 640x480p σε βίντεο 60 ή 90 fps.

Η οπτική ανίχνευση και η αναγνώριση προσώπου και αντικειμένων ήταν μία από τις μεγαλύτερες προκλήσεις στον τομέα της όρασης του υπολογιστή κατά την τελευταία δεκαετία. Η πιθανή χρήση συστημάτων ανίχνευσης και αναγνώρισης αντικειμένων είναι αρκετά μεγάλη,

από συστήματα ασφαλείας (αναγνώριση εξουσιοδοτημένων χρηστών), ιατρικές τεχνικές (ανίχνευση όγκων), βιομηχανικές εφαρμογές (οπτική επιθεώρηση προϊόντων), ρομποτική (πλοήγηση ρομπότ σε απρόσιτο έδαφος), στην επαυξημένη πραγματικότητα, και πολλά άλλα.

Κάθε φάση της διαδικασίας αναγνώρισης χρησιμοποιεί διάφορες μεθόδους και τεχνικές. Η ανίχνευση, η εξαγωγή και η ταξινόμηση έχουν επιλυθεί με αυτές τις μεθόδους. Με την πάροδο του χρόνου, όχι μόνο έχουν αναπτυχθεί νέες μέθοδοι, αλλά και αλγόριθμοι από τις αρχικές μεθόδους, μέχρι σήμερα, υπάρχουν πάνω από 200[19]. Σύμφωνα με αυτήν την προσέγγιση και τις διάφορες χρήσεις της, οι μέθοδοι ανίχνευσης μπορούν να χωριστούν σε τέσσερις κατηγορίες

- **Μέθοδοι βασισμένες στη γνώση**
- **Διαθέτουν αμετάβλητες προσεγγίσεις**
- **Αντιστοίχιση προτύπου**
- **Μέθοδοι που βασίζονται στην εμφάνιση**

Ο διαχωρισμός των μεθόδων δεν είναι ομοιόμορφος, επειδή θα μπορούσαν να χρησιμοποιηθούν οι μέθοδοι και για τις τρεις φάσεις της διαδικασίας αναγνώρισης. Από την άποψη της ταχύτητας και της απλότητας χρήσης, μπορεί να προταθεί μια βιβλιοθήκη OpenCV που περιλαμβάνει έναν ισχυρό ανιχνευτή Viola-Jones με βάση το Haar.

Το (Open-source Computer Vision, opencv.org)[20] διαθέτει μια σειρά από ενότητες με τις οποίες είναι δυνατόν να επιλυθούν ορισμένα προβλήματα στην όραση του υπολογιστή. Το πιο χρήσιμο μέρος του OpenCV μπορεί να είναι **η αρχιτεκτονική** και **η διαχείριση μνήμης**. Παρέχει ένα πλαίσιο εντός του οποίου είναι δυνατή η εργασία με εικόνες και βίντεο με πολλούς τρόπους. Οι αλγόριθμοι για την αναγνώριση προσώπων είναι προσβάσιμοι στη βιβλιοθήκη OpenCV και είναι οι ακόλουθοι:

- **FaceRecognizer.Eigenfaces: Eigenfaces,**
- **FaceRecognizer.Fisherfaces: Fisherfaces,**
- **FaceRecognizer.LBPH**

Η επιλογή του Fisherfaces γίνεται επειδή βασίστηκε στον αλγόριθμο LDA. Η βιβλιοθήκη OpenCV χρησιμοποιήθηκε για αναγνώριση προσώπου και ανίχνευση κίνησης. Είναι μια

βιβλιοθήκη πολλαπλών πλατφορμών λειτουργιών προγραμματισμού που σχετίζονται με την όραση του υπολογιστή.

Πρώτον, πρέπει να καθοριστεί μια σωστή επιλογή αλγορίθμου για την αναγνώριση προσώπου. Το OpenCV περιέχει δύο δημοφιλείς μεθόδους αναγνώρισης προσώπου, το Haar Cascade και τα τοπικά δυαδικά μοτίβα (LBP). Η μέθοδος Haar Cascade βασίζεται στην αρχή της μηχανικής μάθησης. Η συνάρτηση καταρράκτη εκπαιδεύεται χρησιμοποιώντας πολλές εικόνες και στη συνέχεια χρησιμοποιείται για τον εντοπισμό αντικειμένων σε άλλες εικόνες. Η μέθοδος LBP (τοπικό δυαδικό μοτίβο) χρησιμοποιείται για να περιγράψει τα χαρακτηριστικά των εικόνων με διάφορα χαρακτηριστικά που χαρακτηρίζουν την εικόνα. Κατά τη δημιουργία ενός χαρακτηριστικού, περνά μέσα από κάθε εικονοστοιχείο της εικόνας και χρησιμοποιώντας τα χαρακτηριστικά αξιολόγησης φτάνει σε μια τιμή. Σε αυτήν την περίπτωση, η μέθοδος LBP ήταν επειδή οι λειτουργίες της ήταν απλούστερες και γρηγορότερες από τη μέθοδο Haar Cascade.

Και οι δύο αλγόριθμοι χρησιμοποιούν αρχεία XML για την καταγραφή των ιδιοτήτων του αντικειμένου που πρέπει να εντοπιστεί. Η βιβλιοθήκη OpenCV περιέχει ήδη ορισμένα αρχεία XML για αναγνώριση προσώπου και ανίχνευση σώματος. Χρησιμοποιώντας αυτό, δημιουργείται ένα συγκεκριμένο αρχείο LBP ή Haar Cascade XML μέσω εκπαίδευσης.

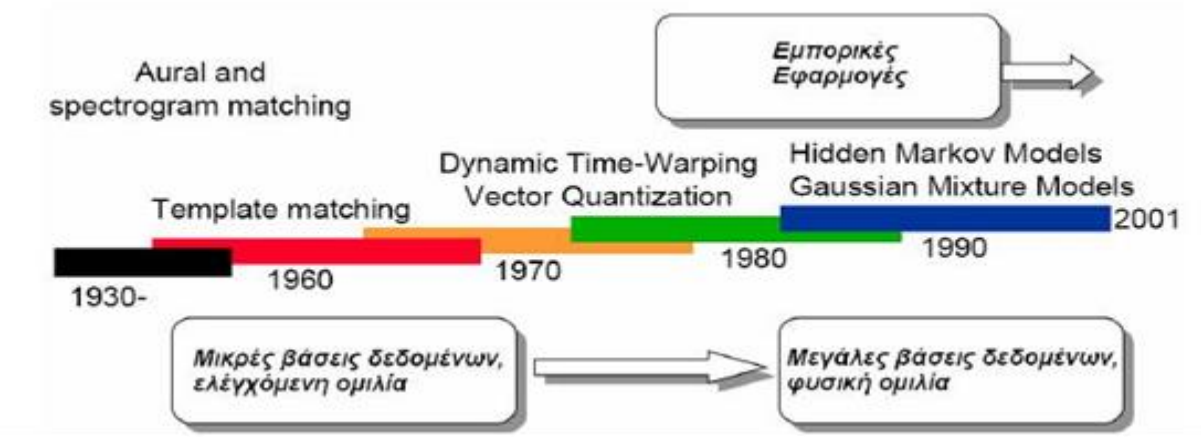
Σχήμα 3.2 – Image from OpenCV tracking a human face

Χρησιμοποιεί επεξεργαστή ενός πυρήνα για την επεξεργασία εικόνων που λαμβάνονται από τη μονάδα κάμερας Raspberry Pi. Ο αλγόριθμος προγραμματίστηκε χρησιμοποιώντας τη βιβλιοθήκη OpenCV και χρησιμοποίησε ένα επικαλυπτόμενο αρχείο "lbpcascades_frontalface.xml" για να αναγνωρίσει τα πρόσωπα στην εικόνα

Για την ανίχνευση κίνησης, δημιουργήθηκαν δύο προγράμματα. Το πρώτο πρόγραμμα (detektor pohybu.py) λειτουργεί χωρίς τη χρήση βιβλιοθήκης OpenCV και όταν αντιλαμβάνεται μια κίνηση δημιουργεί μια εικόνα και την αποθηκεύει σε ένα φάκελο. Το δεύτερο πρόγραμμα (monitorovaci_system.py) χρησιμοποιεί μια βιβλιοθήκη OpenCV και μετά την ανίχνευση κίνησης, σχεδιάζει ένα πράσινο πλαίσιο γύρω από την περιοχή που εντοπίστηκε και καταγράφει το αρχείο στο Dropbox

4. Text-to-speech

Αν και η τεχνολογία ομιλίας έχει ιστορία πάνω από 50 χρόνια, η έρευνα και οι επιχειρήσεις σε αυτόν τον τομέα έχουν αναπτυχθεί ραγδαία τις τελευταίες δύο δεκαετίες. Η συνεχής ανάπτυξη συστημάτων και λειτουργιών που περιλαμβάνουν φωνητική αλληλεπίδραση ανθρώπου-υπολογιστή οφείλεται κυρίως στη δημιουργία μεγάλου αριθμού βάσης δεδομένων φωνητικών αρχείων, δηλαδή αρκετών ωρών φωνής.



Σχήμα 4.1 – Aural and spectrogram matching

Η ανάπτυξη συστημάτων τεχνολογίας ομιλίας βασίζεται σε μεγάλες βάσεις δεδομένων ομιλίας λόγω των ειδικών χαρακτηριστικών της ομιλίας σε σχέση με άλλα βιολογικά χαρακτηριστικά. Για παράδειγμα, τα δακτυλικά αποτυπώματα ενός ατόμου παραμένουν σταθερά μετά την ηλικία των 10 ετών και κάθε άτομο είναι μοναδικό.

Σε αντίθεση με τα δακτυλικά αποτυπώματα, η ομιλία είναι πολύ ευαίσθητη στη μίμηση, τη συναισθηματική κατάσταση του ομιλητή και την υγεία του φωνητικού συστήματος. Επομένως, είναι προφανές ότι η ομιλία ποικίλλει σε μεγάλο βαθμό από ομιλητή σε ομιλητή ή ακόμα και υπό διαφορετικές συνθήκες του ίδιου ομιλητή. Αυτές οι διαφορές, σε συνδυασμό με το γεγονός ότι η τεχνολογία ομιλίας βασίζεται σε μεγάλο βαθμό σε στατιστικές μεθόδους, απαιτούν εκτεταμένα εκπαιδευτικά δεδομένα για να καλύψουν αυτές τις διαφορές όσο το δυνατόν περισσότερο..[8]

Η δημιουργία μιας μεγάλης βάσης δεδομένων που περιέχει φωνητικές εγγραφές διαφορετικών ηχείων σε διαφορετικές συνθήκες εργασίας και περιβάλλοντα, επιτρέπει την ανάπτυξη ανεξάρτητου ηχείου συστήματος αναγνώρισης ομιλίας και μεγάλου αριθμού λεξιλογίου, το οποίο είναι διαφορετικό από το παλιό σύστημα που συνήθως περιορίζεται σε ένα ηχείο και περιορισμένο λεξιλόγιο όπως ο προσδιορισμός αριθμών ή μεμονωμένων λέξεων). Ένα πρόβλημα με τις φωνητικές βάσεις δεδομένων είναι ότι για να μπορέσει το σύστημα να τις χρησιμοποιήσει σωστά, πρέπει να έχει προ-επεξεργαστεί.

Αυτή η προ-επεξεργασία συνήθως περιλαμβάνει μια ακολουθία λέξεων που αντιστοιχεί στο προφορικό φωνητικό μήνυμα. Επιπλέον, τα συστήματα τεχνολογίας ομιλίας χρησιμοποιούν πληροφορίες χαμηλότερου επιπέδου, όπως η φωνητική ακολουθία των λέξεων, οι συλλαβές των λέξεων, η προσωδία της ομιλίας και η θέση της μετατροπής της ομιλίας..

Η προ-επεξεργασία βάσεων δεδομένων διαφορετικών επιπέδων πληροφοριών μπορεί να γίνει χειροκίνητα, συνήθως από έναν φωνητικό. Ωστόσο, εάν η βάση αποτελείται από αρκετές ώρες εγγραφής φωνής, είναι σαφές ότι αυτή η λύση είναι απαγορευτική. Αν και οι περισσότερες από τις παραπάνω πληροφορίες μπορούν να εξαχθούν με ικανοποιητική ποιότητα και ακρίβεια με αυτοματοποιημένες μεθόδους, η εύρεση της θέσης της μετατροπής ομιλίας είναι πλέον ένας σημαντικός τομέας έρευνας, επειδή θεωρείται η πιο δύσκολη πληροφορία που πρέπει να εξαχθεί από τη βιβλιοθήκη ομιλίας.

Τα τελευταία χρόνια, νέα δεδομένα, στα οποία υπάρχουν αρκετές ώρες εγγραφής πολλαπλών ηχείων και φυσικής ομιλίας, οδήγησαν στην ανάπτυξη βελτιωμένων συστημάτων τεχνολογίας ομιλίας και νέων τεχνολογιών με βάση τη χρήση των διαθέσιμων εγγραφών. Για παράδειγμα, στον τομέα της αναγνώρισης ομιλίας, η μετάβαση από έναν μικρό αριθμό ηχογραφήσεων σε μια μεγάλη βάση αρκετών ωρών ομιλίας επέτρεψε τη μετάβαση από την τεχνολογία αντιστοίχισης προτύπων σε στατιστικά μοντέλα ομιλίας, καθώς και την εκπαίδευση ειδικών και ακόμη και γενικών -ανθεκτικά ακουστικά μοντέλα. Λεξικό εφαρμογής σκοπού (αναγνώριση ομιλίας μεγάλου λεξιλογίου).

Ένα τυπικό παράδειγμα του αποτελέσματος της ανάπτυξης βάσης δεδομένων ομιλίας είναι η σύνθεση ομιλίας, όπου η δημιουργία μιας μεγάλης βάσης δεδομένων επιτρέπει την ανάπτυξη νέων τεχνολογιών. Ειδικότερα, η πιο επιτυχημένη και συνήθης μέθοδος σύνθεσης ομιλίας σήμερα είναι η επιλογή μονάδας σύνδεσης μονάδας. Η επιτυχία αυτής της μεθόδου οφείλεται στην άμεση χρήση κατάλληλων τμημάτων ομιλίας χωρίς προ-επεξεργασία και η προκύπτουσα συνθετική ομιλία είναι πολύ κοντά στην αρχική προ-ηχογραφημένη ομιλία, σε σύγκριση με τις παλιές μεθόδους, όπως η σύνθεση formant (Allen et al., 1987; Dutoit, 1996; Huang et al. 2001)[3][11][13].

Η δημιουργημένη βάση δεδομένων φωνής αποτελείται από μια γενική τυπική δομή. Πιο συγκεκριμένα, για κάθε εγγραφή φωνής, υπάρχει ένα αντίστοιχο αρχείο ήχου. Οι ηχογραφήσεις ομαδοποιούνται και ταξινομούνται σύμφωνα με τον ομιλητή, τη διάλεκτο, τη γλώσσα, το περιεχόμενο κ.λπ. Κάθε αρχείο ήχου διαθέτει ένα σύνολο φωνητικών σημειώσεων, που καλύπτουν διαφορετικά επίπεδα[9] σε κάθε βάση, όπως περιεχόμενο φωνής σε επίπεδο λέξεων, σημειώσεις σχετικά με την εμφάνιση ηχητικών εφέ (όπως κάποιο θόρυβο από το

περιβάλλον, (από το ηχείο, εσφαλμένη προφορά κ.λπ. .)), οι νότες της ακολουθίας φωνητικών που αντιστοιχούν στην πρόταση ομιλίας κ.λπ.

Η ποιότητα των υποσημειώσεων που συνοδεύουν ένα σύνολο ηχογραφήσεων είναι πολύ σημαντική όσον αφορά την αξιοποίηση των θεμελίων της τεχνολογίας ή των συστημάτων για την επίτευξη ανταγωνιστικής απόδοσης.

Ένα από τα πιο σημαντικά επίπεδα βασικής σήμανσης ομιλίας είναι η καταγραφή ορίων ομιλίας. Στην πραγματικότητα, η υπόλοιπη κατασκευή της βάσης δεδομένων δεν απαιτεί πολλή προσπάθεια, επειδή το λεκτικό κείμενο συνήθως προετοιμάζεται, εξάγεται από το συγκεκριμένο κείμενο από την υπαγόρευση του ομιλητή, ή εάν δεν είναι διαθέσιμο, μπορεί να εξαχθεί αυτόματα από ένα έτοιμο σύστημα.[10]

Για την εξαγωγή των αντίστοιχων φωνημάτων, συνήθως εξάγεται από το λεξικό προφοράς, ή αν όχι, μπορεί να εξαχθεί μέσω των κανόνων μετατροπής γραμμάτων στο αντίστοιχο πρόγραμμα (grapheme-to-phone ή αλλιώς letter-to-sound κανόνες).

Μια βάση δεδομένων ομιλίας που περιέχει τοποθεσίες μετατροπής ομιλίας είναι απαραίτητη για την ανάπτυξη συστημάτων, όπως κείμενο-σε-ομιλία με συνδέσεις μονάδας. Ωστόσο, οι δυσκολίες που αναφέρθηκαν παραπάνω δείχνουν σαφώς ότι η χρήση φωνητικής για την επισήμανση μεγάλων βάσεων δεδομένων, όπως μερικές ώρες χρόνου ομιλίας, είναι απαγορευτική.

Αυτά τα δεδομένα δείχνουν σαφώς ότι η χρήση υπαρχουσών και μεγάλης κλίμακας βάσεων δεδομένων ομιλίας εξαρτάται άμεσα από την ανάπτυξη μεθόδων αυτόματης αποσύνθεσης σήματος ομιλίας και αυτές οι βάσεις δεδομένων έχουν αποδειχθεί υπεύθυνες για την ταχεία ανάπτυξη της τεχνολογίας ομιλίας. Επομένως, ένα από τα κύρια προβλήματα που μελετήθηκαν στη διεθνή βιβλιογραφία τεχνολογίας ομιλίας είναι η αυτόματη αποσύνθεση ψηφιακών σημάτων ομιλίας και η εφαρμογή της σε συστήματα ομιλίας.[7]

Το σύστημα κειμένου σε ομιλία διαθέτει ένα ευρύ φάσμα εφαρμογών. Η πρώτη πραγματική χρήση τους είναι σε ένα σύστημα ανάγνωσης για τους τυφλούς, όπου το σύστημα διαβάζει κείμενο από ένα βιβλίο και το μετατρέπει σε ομιλία. Σήμερα, υπάρχουν πιο περίπλοκα συστήματα που μπορούν να διευκολύνουν την αλληλεπίδραση ανθρώπου-υπολογιστή για τους τυφλούς, μεταξύ των οποίων το σύστημα TTS βοηθά τους χρήστες να πλοηγηθούν στο "σύστημα Windows". Τα αποτελέσματα των συστημάτων και της συνεχούς έρευνας στον τομέα της σύνθεσης TTS με την πάροδο των ετών και η ανάπτυξη τεχνολογιών νέας γενιάς έχουν ως αποτέλεσμα τη βελτίωση της ποιότητας της σύνθετης ομιλίας, ενώ παράλληλα επιτυγχάνεται χαμηλό κόστος.

Αυτό το γεγονός προάγει την ευρεία διάδοση των εφαρμογών TTS. Αυτός ο τύπος εφαρμογής χρησιμοποιείται για επικοινωνία, όπου τα μηνύματα που βασίζονται σε κείμενο (όπως email ή fax) ή πληροφορίες που συνδυάζουν κείμενο και εικόνες (όπως ιστοσελίδες) παρουσιάζονται φωνητικά. Υπάρχουν επίσης προγράμματα VoiceXML που παρέχουν διαδραστικές υπηρεσίες φωνής που βασίζονται στο Web. Γενικά, το σύστημα TTS πληροί τις απαιτήσεις φωνητικής

απόδοσης διαφόρων πληροφοριών που είναι αποθηκευμένες στη βάση δεδομένων (όπως αριθμοί τηλεφώνου, διευθύνσεις ή πληροφορίες πλοήγησης αυτοκινήτου).

Μπορούν επίσης να βρουν την εφαρμογή στην αυτοματοποιημένη υπηρεσία πληροφοριών, η οποία μπορεί να εκδίδει ανακοινώσεις καιρού και ειδήσεων μέσω τηλεφώνου ή αυτοματοποιημένων τηλεφωνικών κέντρων, όπου οι χρήστες μπορούν να καλέσουν για να κάνουν κρατήσεις και να καθοδηγήσουν ολόκληρη τη συναλλαγή μέσω ενός αυτοματοποιημένου συστήματος διαλόγου. Δίνουν επίσης λύσεις και σε πιο κλασικές ανάγκες του ανθρώπου, όπως είναι η αυτόματη ανάγνωση εντύπων από μια μηχανή (ομιλούντα βιβλία), που εξυπηρετεί ιδιαίτερα άτομα με προβλήματα όρασης.

Σήμερα, πολλές εφαρμογές που χρησιμοποιούν δεδομένα φωνής ως έξοδο δεν χρησιμοποιούν κείμενο-σε-ομιλία, αλλά ένα σύνολο εγγραφών φωνής που αναπαράγονται όταν χρειάζεται. Αυτά ονομάζονται "προκαθορισμένες φωνές" ή "προ-ηχογραφημένα μηνύματα", ανάλογα με την εφαρμογή. Ο τυπικός τρόπος για την ανάπτυξή τους είναι η δημιουργία μιας λίστας συστάσεων που απαιτούνται για την εφαρμογή. Αυτά είναι ειδικά για κάθε εφαρμογή και δημιουργούν ένα νέο σύνολο εγγραφών για διαφορετικές εφαρμογές.

Μόλις δημιουργηθούν οι προτάσεις, ο ομιλητής τις διαβάζει, τις καταγράφει και τις αποθηκεύει ως αρχεία κύματος. Η εφαρμογή αναπαράγει το απαιτούμενο αρχείο στον απαιτούμενο χρόνο. Ένα τυπικό σύστημα αυτής της φόρμας είναι μια ανακοίνωση σταθμού λεωφορείων ή ένα αυτόματο σύστημα κρατήσεων. Το πρακτικό μέρος της εργασίας περιλαμβάνει επίσης αυτήν τη σύνθεση ομιλίας.

Αυτά τα συστήματα συγκρίθηκαν αρχικά με το TTS και θεωρήθηκαν λιγότερο χρήσιμα επειδή έπρεπε να παρέχουν ένα νέο σύνολο συστάσεων για κάθε εφαρμογή και το TTS σχεδιάστηκε μία φορά και χρησιμοποιήθηκε για κάθε εφαρμογή. Ωστόσο, αυτή η δήλωση είναι απολύτως φυσική. Αντιμετωπίζοντας την επιλογή μεταξύ φυσικής και άκαμπτης ομιλίας και αφύσικου αλλά εντελώς άκαμπτου κειμένου σε ομιλία, ορισμένοι ερευνητές έχουν προτείνει ένα «σύστημα σύνθεσης πεπερασμένων πεδίων» που συνδυάζει τα δύο..

Το σύστημα TTS είναι ένα πλήρως αυτόματο σύστημα για τη μετατροπή κειμένου σε ομιλία. Πιο συγκεκριμένα, δέχεται κείμενο ως είσοδο και δημιουργεί πολύπλοκες ομιλίες, παρέχοντας έτσι στους χρήστες πληροφορίες κειμένου μέσω φωνητικών μηνυμάτων. Το κείμενο μπορεί να εισαχθεί απευθείας στον υπολογιστή ή να σαρωθεί από τον χρήστη και στη συνέχεια να περάσει μέσω ενός συστήματος οπτικής αναγνώρισης χαρακτήρων (OCR). Η AT&T Bell Labs έχει αναπτύξει το σύστημα TTS για τα αγγλικά και έχει αναπτύξει συστήματα σε πολλές άλλες γλώσσες..

Η δυσκολία στην ανάγνωση κειμένου γραμμένο από φωνητικούς συγγραφείς και τη μετατροπή του σε φωνή φυσικής ακρόασης είναι ότι το σύστημα γραφής δεν μπορεί να αναγνωρίσει πληροφορίες που σχετίζονται με τη φωνή. Η γραπτή ομιλία καθορίζει μόνο εν μέρει την έμφαση των φράσεων που χρησιμοποιούν σημεία στίξης, αλλά δεν καθορίζει ποιες λέξεις θα τονιστούν και ποιες όχι, το μήκος κάθε τμήματος και την ποιότητα της ομιλίας. Ο λόγος για τον οποίο ένα άτομο μπορεί να επιτύχει όλους αυτούς τους στόχους είναι επειδή δεν

καταλαβαίνει μόνο τη γραμματική της γλώσσας, αλλά επίσης κατανοεί το περιεχόμενο του κειμένου που διαβάζει.

Επομένως, το έργο του συστήματος TTS είναι περίπλοκο επειδή πρέπει να μιμηθεί έναν ανθρώπινο αναγνώστη. Αλλά οι μηχανές δεν κατανοούν πλήρως τους γραμματικούς κανόνες μιας γλώσσας, ούτε μπορούν να καταλάβουν οτιδήποτε διαβάζουν. Επομένως, ο αλγόριθμος TTS θα πρέπει να μπορεί να χρησιμοποιεί όσο το δυνατόν καθαρότερες γραμματικές πληροφορίες για να ορίζει στοιχεία όπως η προφορά και η προφορά, και να δίνει ένα μέσο επίπεδο επιρροής στην έξοδο με βάση την κατανόηση.

Τα θέματα TTS περιλαμβάνουν:

Α) τη μετατροπή του κειμένου σε γλωσσική αναπαράσταση, που περιλαμβάνει πληροφορίες για τα φωνήματα που θα παραχθούν, τη διάρκειά τους, την τοποθέτηση παύσης και την περιφέρεια της F0 που θα χρησιμοποιηθεί.

Β) τη σύνθεση ομιλίας, [19] δηλαδή μετατροπή της αναπαράστασης της πληροφορίας σε κυματομορφή ομιλίας.

α) Η μετατροπή του κειμένου σε γλωσσική αναπαράσταση αποτελείται από τα ακόλουθα:

- **προεπεξεργασία του κειμένου**
- **καθορισμός τονισμού**
- **προφορά λέξης**
- **τονικός διαχωρισμός σε φράσεις**
- **διάρκειες τμημάτων**

υπολογισμός της περιφέρειας της F0.

β) Η σύνθεση ομιλίας αποτελείται από τα εξής βήματα:

- **Επιλογή των μονάδων που θα συνενωθούν (concatenative units) με δεδομένη τη σειρά φωνημάτων που πρέπει να συντεθεί.**
- **Συνένωση (concatenation) των μονάδων αυτών.**
- **Σύνθεση μιας κυματομορφής ομιλίας με δεδομένα τις μονάδες και το μοντέλο «γλωττιδικής πηγής».**

Οι τεχνικές σύνθεσης ομιλίας χωρίζονται σε δύο μεγάλες κατηγορίες: Τα “system models”, που επιχειρούν να μοντελοποιήσουν το ανθρώπινο σύστημα παραγωγής ομιλίας και τα “signal models”, που επιχειρούν να μοντελοποιήσουν μόνο το παραγόμενο σήμα ομιλίας.

Στην πρώτη κατηγορία ανήκει η «αρθρωτική» σύνθεση (articulatory synthesis), η οποία μοντελοποιεί άμεσα το ανθρώπινο σύστημα παραγωγής ομιλίας. Στη δεύτερη κατηγορία ανήκουν:

- **α)** Η “formant” σύνθεση, που βασίζεται σε κανόνες (rule-based).
- **β)** Η σύνθεση με «συνένωση» (concatenative synthesis), που χρησιμοποιεί προηχογραφημένα μικρά τμήματα ομιλίας, τα οποία συνενώνει στο πεδίο του χρόνου.

Η σύνθεση με «βάση κανόνες» μαζί με τη σύνθεση με «συνένωση» αποτελούν τις δύο πιο συχνά χρησιμοποιούμενες μεθόδους στα παρόντα συστήματα σύνθεσης. Η πρώτη ήταν κυρίαρχη για πολύ καιρό, αλλά σήμερα η σύνθεση με «συνένωση» είναι πιο δημοφιλής. Η «*αρθρωτική*» μέθοδος είναι ακόμα πάρα πολύ περίπλοκη για υψηλής ποιότητας εφαρμογές, αλλά μπορεί να εξελιχθεί και να χρησιμοποιηθεί εκτενέστερα στο μέλλον.

α) Από τους δύο κύριους τύπους σύνθεσης, ο πρώτος χρησιμοποιεί ένα σύνολο κανόνων (βάσει κανόνων) που ελέγχουν το μοντέλο γραμμικού φίλτρου πηγής για τη δημιουργία ομιλίας, το "μοντέλο φίλτρου πηγής". Σύμφωνα με αυτό το μοντέλο, η πηγή διέγερσης είναι εντελώς ανεξάρτητη από τη μορφή του καναλιού ομιλίας. Το φίλτρο καναλιού ομιλίας καθορίζεται από παραμέτρους ελέγχου, όπως η συχνότητα και το εύρος ζώνης του σχηματιστή που υπολογίζεται για κάθε φωνή. Αυτό το συνδυασμένο σύστημα περιέχει κανόνες για τη μοντελοποίηση των φαινομένων προφοράς και υπολογίζει τις παραμέτρους ελέγχου για κάθε φωνή με βάση αυτούς τους κανόνες..

β) Στο δεύτερο κύριο τύπο σύνθεσης που χρησιμοποιείται από το σύστημα AT&T TTS, προστίθενται σύντομα τμήματα της διαμορφωμένης φυσικής ομιλίας, δηλαδή, συνδεδεμένα σε μια σύνθετη αναπαράσταση ομιλίας για τη δημιουργία προτάσεων. Ο όρος "τμήμα ομιλίας" αναφέρεται κυρίως σε φωνήματα ή ζεύγη φωνημάτων. Τα φασματικά χαρακτηριστικά ενός τμήματος ομιλίας διαφέρουν από το περιβάλλον ομιλίας του, το οποίο καθορίζεται από τη διαφορά σε γειτονικά φωνήματα, τόνους και θέσεις.

Η συγχώνευση είναι ευκολότερη και παρέχει καλύτερη ποιότητα ήχου. Ωστόσο, είναι αδύνατο να αλλάξετε τις φασματικές παραμέτρους και ο συνθέτης περιορίζεται συνήθως σε ένα ηχείο και έναν ήχο, οπότε απαιτείται περισσότερη χωρητικότητα αποθήκευσης.

Το σύστημα AT&T TTS αποτελείται από δεκατρείς μονάδες, καθεμία από τις οποίες είναι υπεύθυνη για ένα τμήμα του προβλήματος της μετατροπής text-to-speech, δηλαδή:

1)επεξεργασία κειμένου, 2)μετατροπή λέξεων σε λήμματα, 3)τονισμός, 4)προφορά λέξης, 5)καθορισμός ορίων τονικών φράσεων, 6)καθορισμός τονισμού φράσης, 7)διάρκεια τμήματος, 8)τονισμός φωνής, 9)πλάτος, 10)γλωττιδική πηγή, 11)επιλογή δυάδας φωνημάτων, 12)σύνδεση δυάδας, 13)σύνθεση.

Οι πληροφορίες μεταφέρονται από μονάδα σε μονάδα σε ενότητα προτάσεων και αποτελείται από ένα σύνολο πινάκων δομής γλώσσας. Ο πίνακας δομής περιέχει τον τύπο Τ της δομής (για παράδειγμα, λέξεις), τον αριθμό Ν του Τ στην πρόταση και το μέγεθος S κάθε Τ ως τίτλο. Επομένως, για Ν δομές τύπου Τ, έχουμε έναν πίνακα byte μεγέθους NxS. Κάθε μονάδα διαβάζει μια πρόταση κάθε φορά, επεξεργάζεται τα δεδομένα εισόδου και καταγράφει τη δομή της επόμενης μονάδας. Προσθέστε πληροφορίες σε κάθε μονάδα. Ένα από τα πλεονεκτήματα είναι η συνεχής παρακολούθηση.

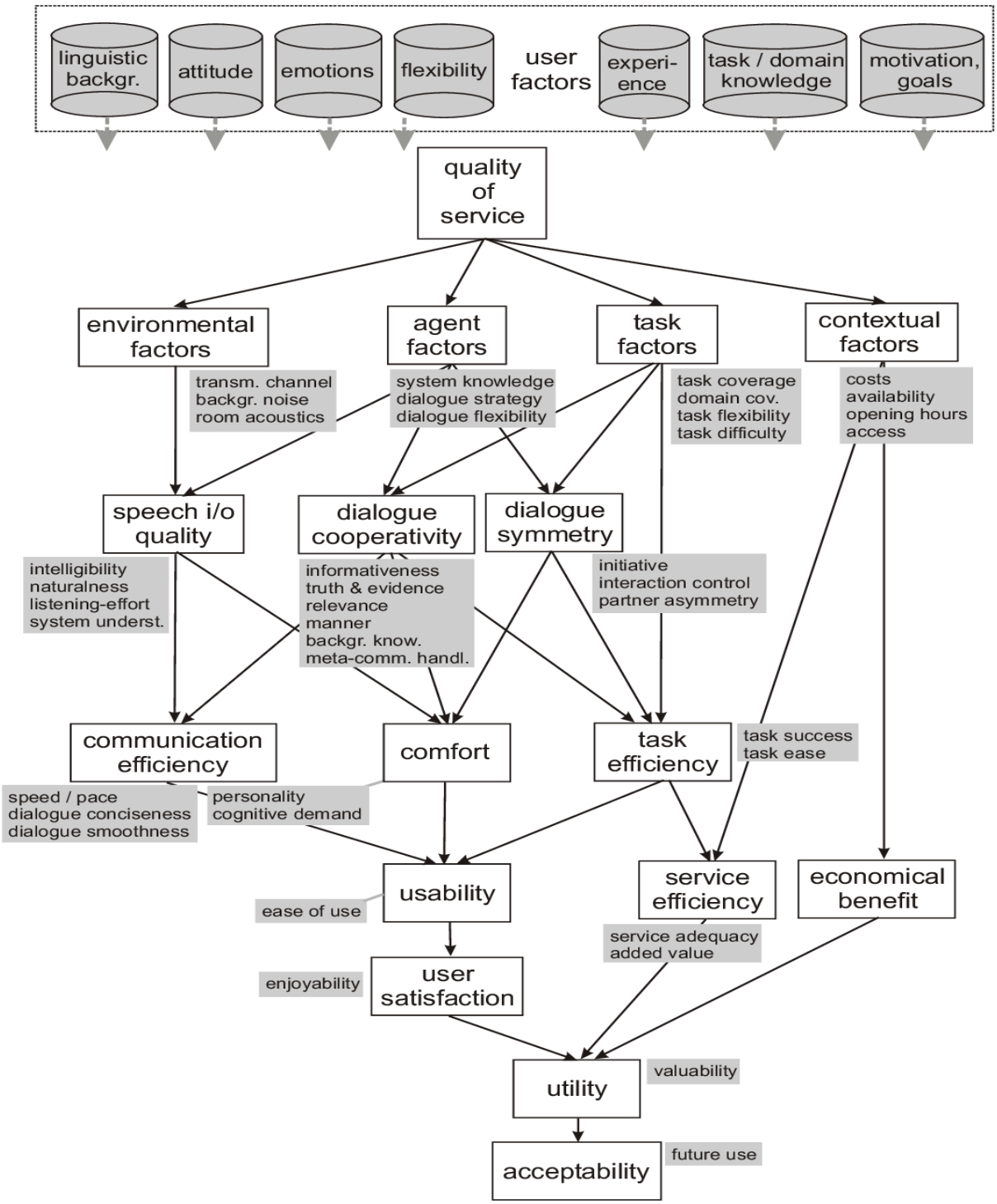
Επιπλέον, το σχήμα της δομής καθιστά εύκολη τη διακοπή της επεξεργασίας ανά πάσα στιγμή. Είναι επίσης εύκολο να εισαγάγετε ένα πρόγραμμα που αλλάζει τις παραμέτρους TTS οπουδήποτε στη δομή. Μπορεί να εισαχθεί μια εφαρμογή, όπως μια μονάδα μελωδίας, η οποία αντικαθιστά τη μονάδα φωνητικής επισήμανσης και αναγκάζει το σύστημα να ρυθμίσει το κείμενο στη μουσική με γραπτή ταχύτητα. Η σύνθεση "Επιλογή μονάδας" είναι η σημερινή κύρια μέθοδος σύνθεσης (σύνθεση τρίτης γενιάς) και θεωρείται επέκταση της συγχώνευσης.

Η μέθοδος επιλογής μονάδας χρησιμοποιεί περισσότερους τύπους δεδομένων φωνής για να επιτύχει μια πιο φυσική ποικιλομορφία που εξαρτάται λιγότερο από την επεξεργασία σήματος. Η κεντρική ιδέα είναι ότι για κάθε βασικό τύπο γλώσσας, έχουμε πολλές μονάδες, και ο ρυθμός και τα άλλα χαρακτηριστικά αυτών των μονάδων θα είναι διαφορετικά. Στη διαδικασία σύνθεσης, ο αλγόριθμος θα επιλέξει μια πιθανή μονάδα για να προσπαθήσει να βρει την καλύτερη συνολική ακολουθία μονάδας που πληροί τις προδιαγραφές. Σε σύγκριση με τη συνδυασμένη σύνθεση, μια εφαρμογή με μόνο μία μονάδα ή δύο φωνές ανά φωνή περιορίζει την ποιότητα, επομένως εξελίσσεται φυσικά για την αποθήκευση πολλαπλών μονάδων

Ο πιο λογικός τρόπος να γίνει αυτό είναι να ληφθούν υπόψη χαρακτηριστικά πέραν της συχνότητας και του συγχρονισμού, όπως τονισμός και χωρισμός σε φράσεις, και να θεωρηθεί μία μονάδα για καθένα από αυτά τα γνωρίσματα. Αντί να εγγράψουμε και να αναλύσουμε μία έκδοση κάθε φωνήματος, εγγράφουμε και αναλύουμε μία έκδοση για κάθε συνδυασμό των παραπάνω γνωρισμάτων.

Στο "Επιλογή μονάδας", έχουμε μια βάση δεδομένων και αναλύουμε για να χρησιμοποιήσουμε ολόκληρο το ΒΔ ως μονάδα στο χαρτοφυλάκιο. Αυτά τα συστήματα διαφέρουν ως προς το βαθμό σχεδιασμού του περιεχομένου ΒΔ. Η "επιλογή μονάδας" γίνεται με τη βοήθεια μεγαλύτερου ΒΔ από τη συγχώνευση. Χρησιμοποιώντας ένα μεγαλύτερο ΒΔ, συχνά διαπιστώνουμε ότι επιλέγεται ένα μεγαλύτερο τμήμα συνεχούς ομιλίας, το οποίο είναι ένας από τους βασικούς παράγοντες που οδηγούν σε υψηλή ποιότητα της πρότασης. Συνήθως δεν γίνονται τροποποιήσεις επεξεργασίας σήματος, που ονομάζεται "καθαρή επιλογή μονάδας" (pure unit selection).

Μια άλλη άποψη της επιλογής μονάδας είναι ότι είναι ένας αλγόριθμος που "τμηματοποιεί" την ομιλία και αναδιατάσσει την ομιλία. Αυτό οδηγεί σε μια βασική αρχή, δηλαδή, την επίτευξη των προδιαγραφών με την αναδιάταξη των αρχικών δεδομένων, με όσο το δυνατόν λιγότερα μέσα, προκειμένου να διατηρηθεί όσο το δυνατόν περισσότερο η ποιότητα της αρχικής φωνής. Πολλές παράμετροι καθορίζουν την ποιότητα της διεπαφής ήχου:



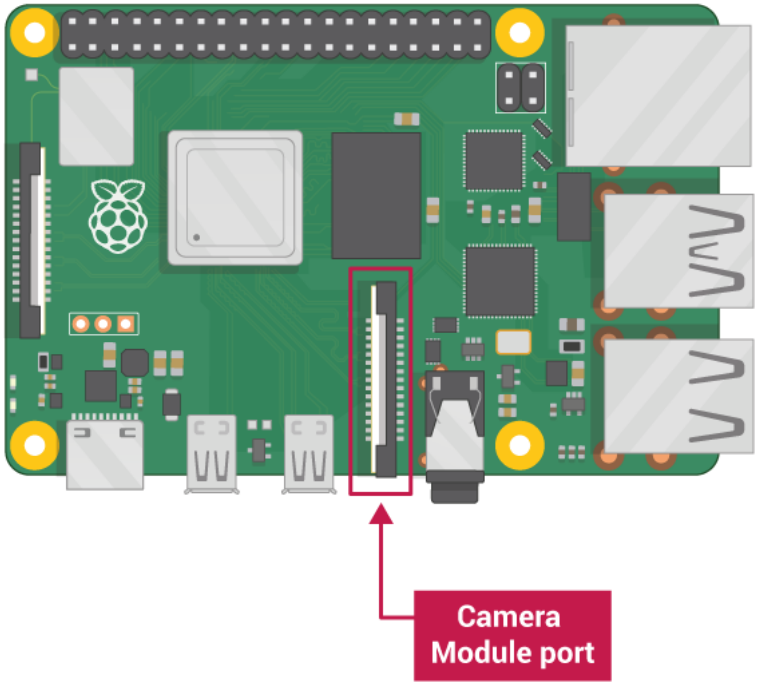
Σχήμα 4.2 - Taxonomy for the Quality of Telephone Services Based on Spoken Dialogue Systems

5. Ανάπτυξη εφαρμογής

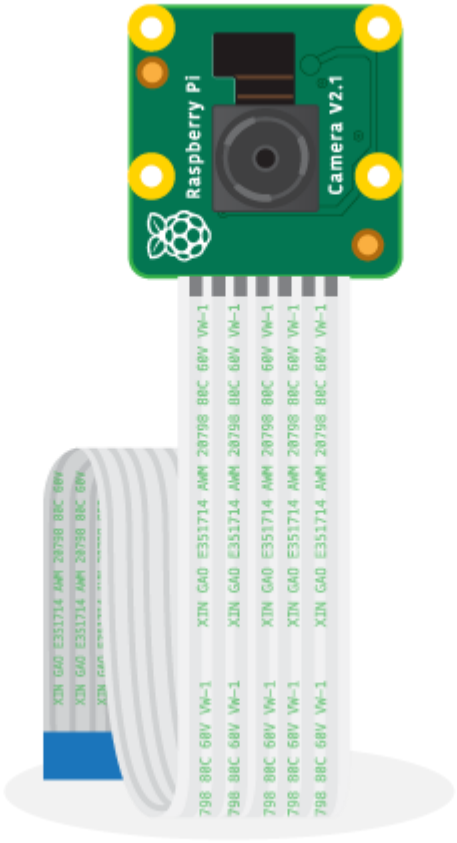
Θα χρειαστούμε τα κάτωθι:

- Raspberry Pi computer με ένα Camera Module port

Όλα τα τρέχοντα μοντέλα του Raspberry Pi διαθέτουν θύρα για τη σύνδεση της μονάδας κάμερας.



Σχήμα 5.1 - Raspberry Pi Camera Module port



Σχήμα 5.2 - Raspberry Pi Camera Module

Υπάρχουν δύο εκδόσεις του Camera Module:

- The standard version, που έχει σχεδιαστεί για τη λήψη φωτογραφιών σε κανονικό φως
- The NoIR version, το οποίο δεν διαθέτει φίλτρο υπερύθρων, οπότε μπορούμε να το χρησιμοποιήσουμε μαζί με μια πηγή φωτός υπερύθρων για να τραβήξουμε φωτογραφίες στο σκοτάδι

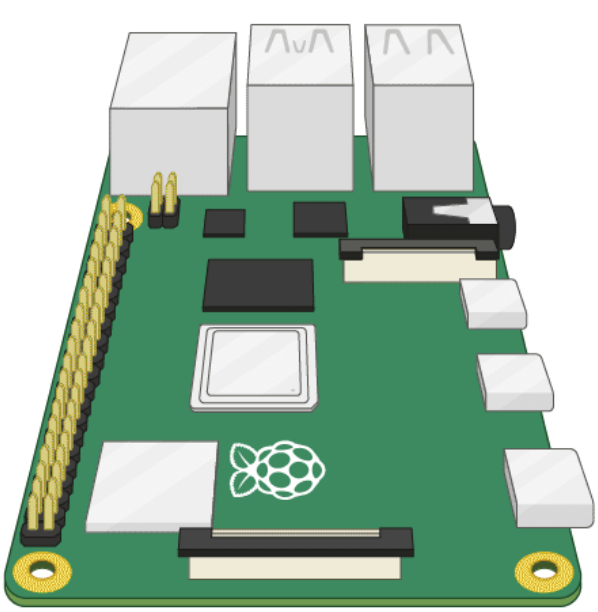
Βεβαιωνόμαστε ότι το Raspberry Pi είναι απενεργοποιημένο.

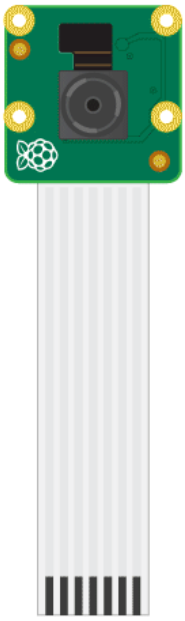
1. Εντοπίζουμε τη θύρα της μονάδας κάμερας

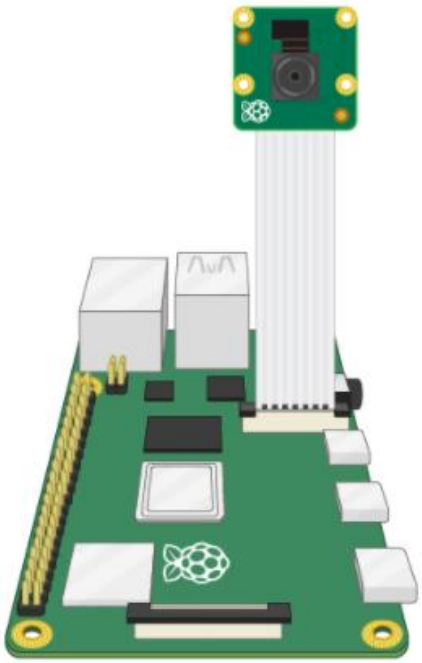
2. Τραβάμε απαλά τις άκρες του πλαστικού κλιπ της θύρας

3. Τοποθετούμε το καλώδιο κορδέλας της μονάδας κάμερας. Βεβαιωνόμαστε ότι το καλώδιο είναι σωστό

4. Ωθούμε το πλαστικό κλιπ πίσω στη θέση του



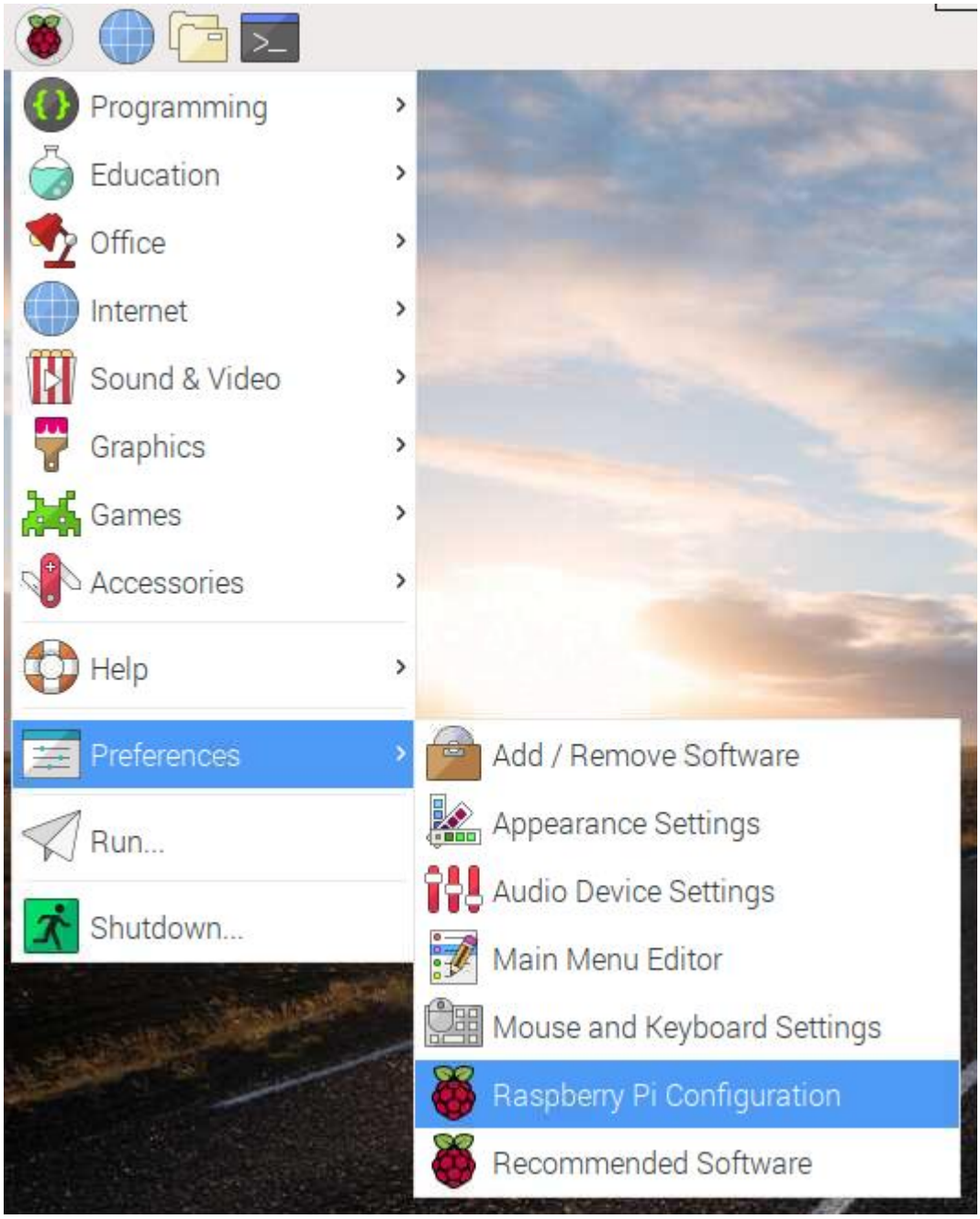




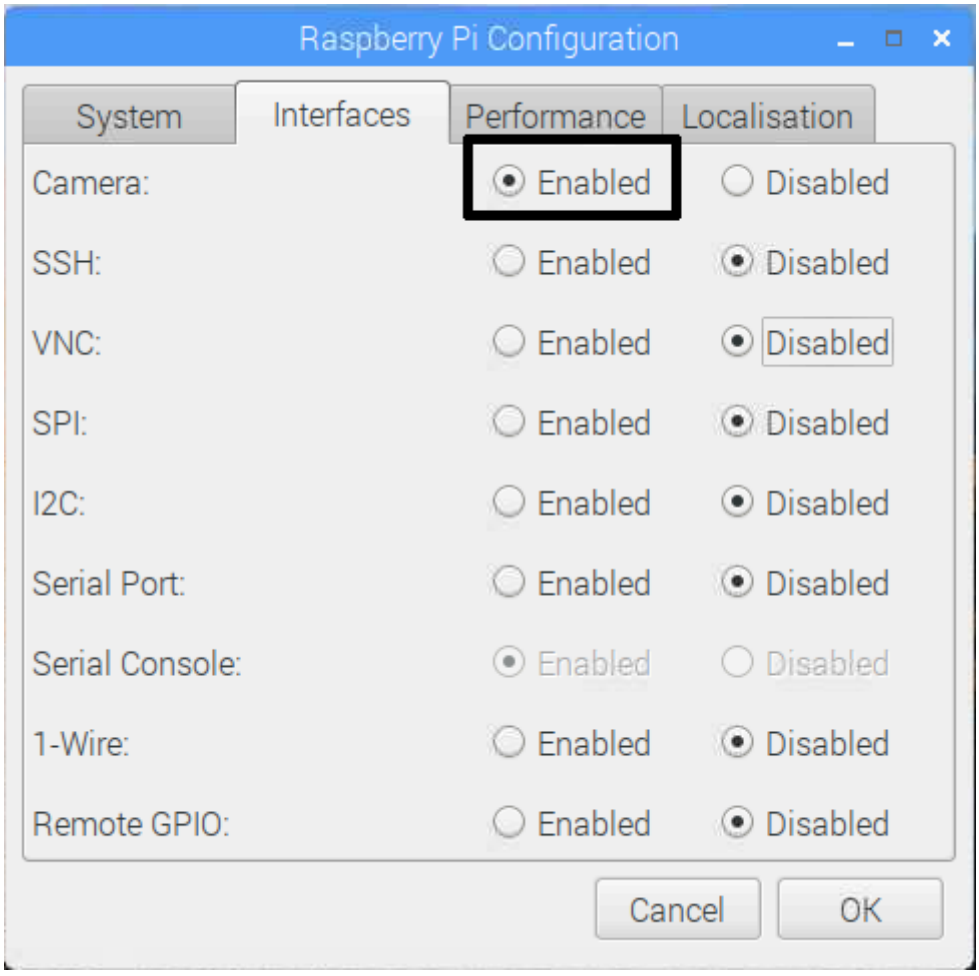
Σχήμα 5.3 & 5.4 - Raspberry Pi Camera installation

Ξεκινούμε το Raspberry Pi.

Μεταβαίνουμε στο κύριο μενού και ανοίγουμε το εργαλείο διαμόρφωσης Raspberry Pi.


Programming
Education
Office
Internet
Sound & Video
Graphics
Games
Accessories
Help
Preferences
Run...
Shutdown...
Add / Remove Software
Appearance Settings
Audio Device Settings
Main Menu Editor
Mouse and Keyboard Settings
Raspberry Pi Configuration
Recommended Software

Επιλέγουμε την καρτέλα Διεπαφές και βεβαιωνόμαστε ότι η κάμερα είναι ενεργοποιημένη:



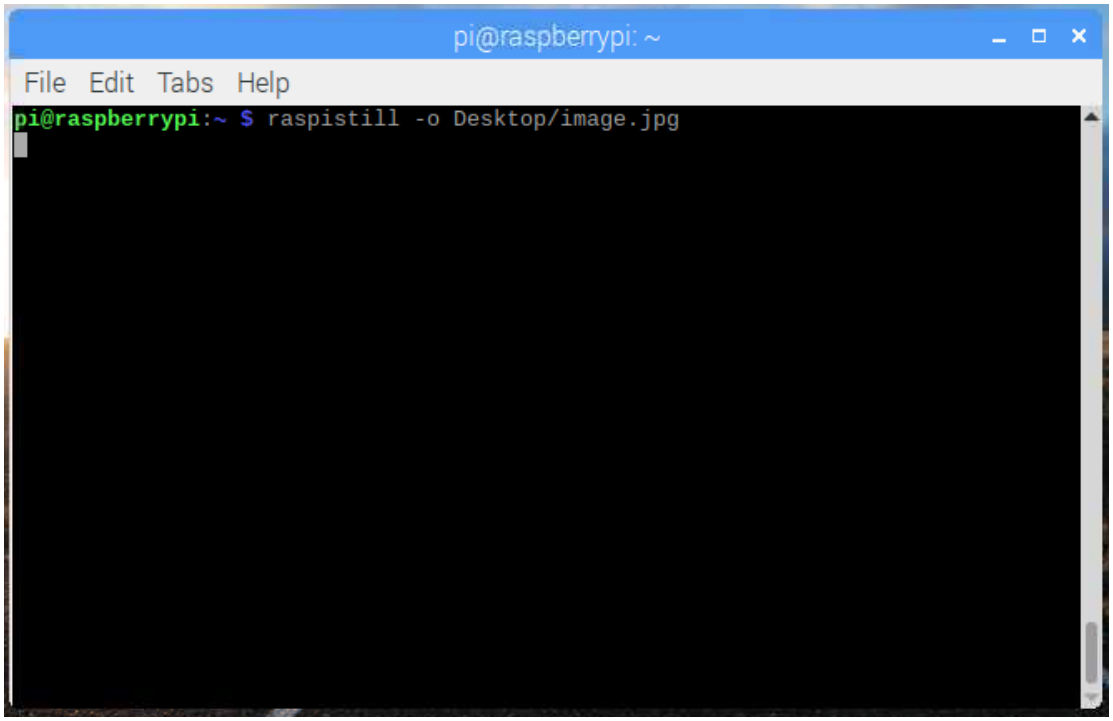
Κανουμε επανεκκίνηση το Raspberry Pi.

Ελέγχουμε το Camera Module μέσω του command line

Τώρα η μονάδα κάμερας είναι συνδεδεμένη και το λογισμικό είναι ενεργοποιημένο, δοκιμάζουμε τα εργαλεία της γραμμής εντολών raspistill και raspivid.
Ανοίγουμε ένα παράθυρο τερματικού κάνοντας κλικ στο εικονίδιο μαύρης οθόνης στη γραμμή εργασιών:

Πληκτρολογούμε την ακόλουθη εντολή για να τραβήξουμε μια φωτογραφία και να την αποθηκεύσουμε στην επιφάνεια εργασίας:

raspistill -o Desktop/image.jpg



1. Πληκτρολογούμε Enter για να τρέξει – εκτελεστεί η εντολή.

Όταν εκτελείται η εντολή, μπορούμε να δούμε την προεπισκόπηση της κάμερας ανοιχτή για πέντε δευτερόλεπτα πριν από τη λήψη μιας ακίνητης εικόνας.

2. Αναζητούμε το εικονίδιο αρχείου εικόνας στην επιφάνεια εργασίας και κάνουμε διπλό κλικ στο εικονίδιο αρχείου για να ανοίξουμε την εικόνα.

Προσθέτοντας διαφορετικές επιλογές, μπορούμε να ορίσουμε το μέγεθος και την εμφάνιση της εικόνας raspistill

.

Για παράδειγμα, προσθέτουμε -h και -w για να αλλάξουμε το ύψος και το πλάτος της εικόνας:

```
raspistill -o Desktop/image-small.jpg -w 640 -h 480
```

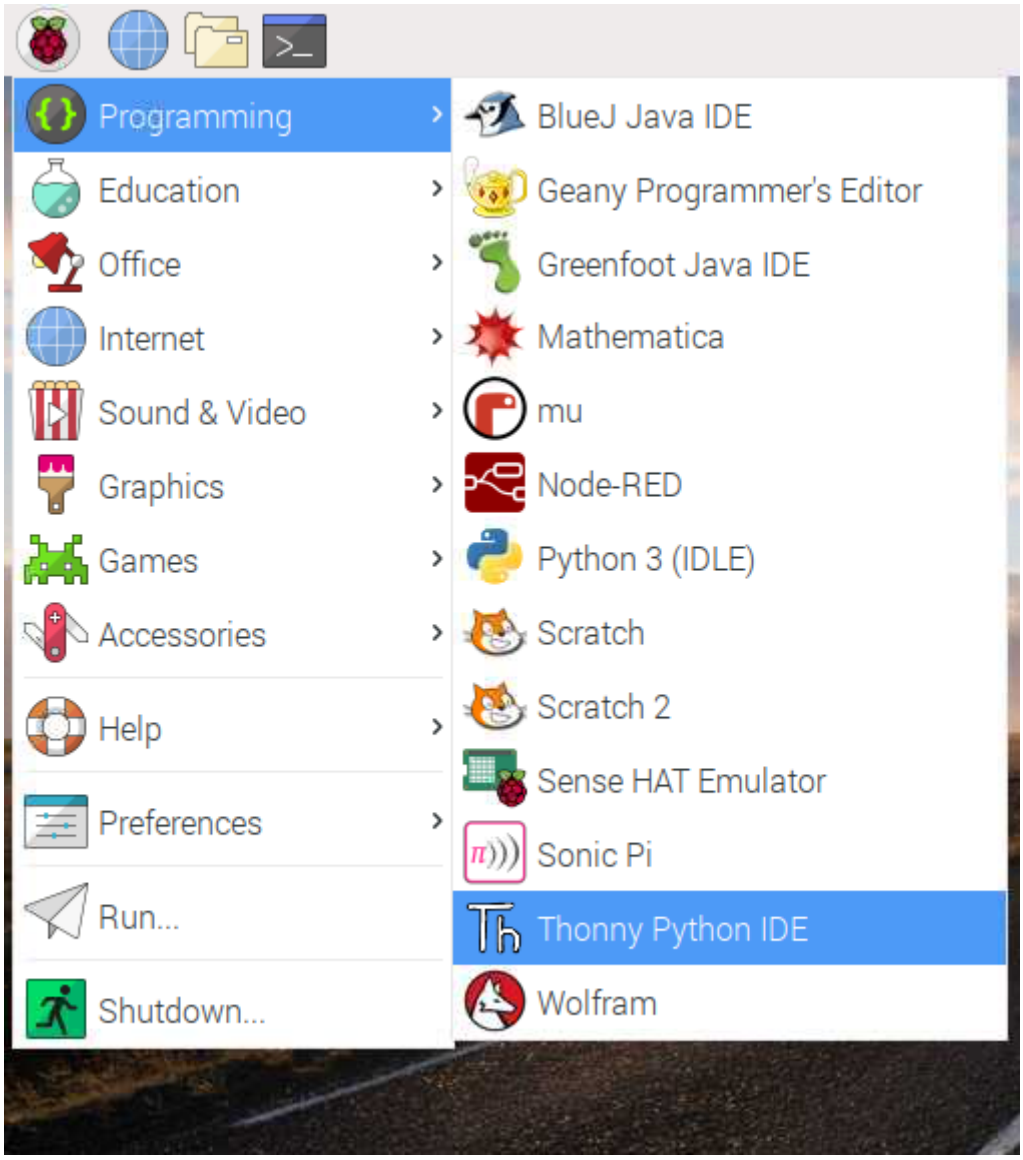
Τώρα εγγράφουμε ένα βίντεο με τη μονάδα κάμερας χρησιμοποιώντας την εντολή

raspivid: raspivid -o Desktop/video.h264

Παίζουμε το video **video.h264** file icon στο Desktop για να ανοίξουμε το αρχείο στο VLC Media Player.

Η βιβλιοθήκη Python picamera μάς επιτρέπει να ελέγχουμε το Camera Module.

Ανοίγουμε ένα πρόγραμμα επεξεργασίας Python 3, όπως το Thonny Python IDE:



- Ανοίγουμε ένα νέο αρχείο και το σώνουμε με το ονομα camera.py.
  picamera.py.
- Καταχωρούμε τον ακόλουθο κώδικα :

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
sleep(5)
camera.stop_preview()
```

Σώζουμε και Τρέχουμε το πρόγραμμα. Η προεπισκόπηση της κάμερας θα πρέπει να εμφανίζεται για πέντε δευτερόλεπτα και μετά να κλείνει ξανά.

Εάν η προεπισκόπηση μας είναι ανάποδα, μπορούμε να την περιστρέψουμε κατά 180 μοίρες με τον ακόλουθο κωδικό

```
camera = PiCamera()
camera.rotation = 180
```

Τώρα χρησιμοποιούμε το Camera Module και τη Python για να τραβήξουμε ακόμη μερικές φωτογραφίες.

Τροποποιούμε το κώδικα μας προσθέτοντας μια γραμμή `camera.capture()` :

```
camera.start_preview()
sleep(5)
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

Τρέχουμε το κώδικα .

Το Raspberry Pi θα πρέπει να ανοίξει μια προεπισκόπηση, να εγγράψει βίντεο 5 δευτερολέπτων και, στη συνέχεια, να κλείσει την προεπισκόπηση.

Το λογισμικό Python picamera παρέχει μια σειρά εφέ και διαμορφώσεων για να αλλάξετε τον τρόπο εμφάνισης των εικόνων μας.

Σημείωση: ορισμένες ρυθμίσεις επηρεάζουν μόνο την προεπισκόπηση και όχι την καταγεγραμμένη εικόνα, ορισμένες επηρεάζουν μόνο την καταγεγραμμένη εικόνα και πολλές άλλες επηρεάζουν και τα δύο.

5.1 Set the image resolution

Μπορούμε να χρησιμοποιήσουμε `camera.image_effect` για συγκεκριμένες παρουσιάσεις .

Οι επιλογές είναι :

- **none**
- **negative**
- **solarize**

- **sketch**
- **denoise**
- **emboss**
- **oilpaint**
- **hatch**
- **gpen**
- **pastel**
- **watercolor**
- **film**
- **blur**
- **saturation**
- **colorswap**
- **washedout**
- **posterise**
- **colorpoint**
- **colorbalance**
- **cartoon**
- **deinterlace1**
- **deinterlace2**

Επιλέγουμε μια εικόνα – παράμετρο:

```
camera.start_preview()
camera.image_effect = 'colorswap'
sleep(5)
camera.capture('/home/pi/Desktop/colorswap.jpg')
camera.stop_preview()
```

Τρέχουμε το κώδικα με τις παραπάνω παραμέτρους camera.IMAGE_EFFECTS:

```
camera.start_preview()
for effect in camera.IMAGE_EFFECTS:
    camera.image_effect = effect
    camera.annotate_text = "Effect: %s" % effect
    sleep(5)
camera.stop_preview()
```

5.2 DeepSpeech

Εγκατάσταση Mozilla DeepSpeech 0.9.3 στο Raspberry Pi 4

Απαιτούμενα

Raspberry Pi 4

- Raspbian Lite buster

Εγκατάσταση DeepSpeech 0.9.3

sudo apt install git python3-pip python3-scipy python3-numpy python3-pyaudio libatlas3-base
pip3 install deepspeech --upgrade
mkdir ~/dspeech
cd ~/dspeech
curl -LO https://github.com/mozilla/DeepSpeech/releases/download/v0.9.3/deepspeech-0.9.3-models.tflite
curl -LO https://github.com/mozilla/DeepSpeech/releases/download/v0.9.3/deepspeech-0.9.3-models.scorer
curl -LO https://github.com/mozilla/DeepSpeech/releases/download/v0.9.3/audio-0.9.3.tar.gz
tar xvf audio-0.9.3.tar.gz
source ~/.profile

Απομαγνητοφώνηση τριων αρχείων..

deepspeech --model deepspeech-0.9.3-models.tflite --scorer deepspeech-0.9.3-models.scorer --audio audio/2830-3980-0043.wav
deepspeech --model deepspeech-0.9.3-models.tflite --scorer deepspeech-0.9.3-models.scorer --audio audio/4507-16021-0012.wav
deepspeech --model deepspeech-0.9.3-models.tflite --scorer deepspeech-0.9.3-models.scorer --audio audio/8455-210777-0068.wav

Άμεση απομαγνητοφώνηση αρχείων από το μικροφωνο.

sudo nano /usr/share/alsa/alsa.conf
defaults.ctl.card 3
defaults.pcm.card 3

git clone https://github.com/mozilla/DeepSpeech-examples
pip3 install halo webrtcvad --upgrade
python3 DeepSpeech-examples/mic_vad_streaming/mic_vad_streaming.py -m deepspeech-0.9.3-models.tflite -s deepsp

5.3 RPi Text to Speech (Speech Synthesis)

Install supporting packages:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
sudo apt-get install alsa-utils
sudo nano /etc/modules
snd_bcm2835
sudo apt-get install mplayer
sudo nano /etc/mplayer/mplayer.conf
nolirc=yes
```

Cepstral Text to Speech

Το Cepstral είναι μια μηχανή Text to Speech που είναι εγκατεστημένη στο Pi και δεν απαιτεί σύνδεση στο Διαδίκτυο.

https://www.cepstral.com/raspberrypi

5.4 Docker

Η Docker είναι μια πλατφόρμα “εγκιβωτιοποίησης”.[17] Παρέχει μια ελαφριά και αποτελεσματική προσέγγιση για την συσκευασία και την εκτέλεση εφαρμογών σε απομονωμένα περιβάλλοντα που ονομάζονται “κοντέινερ (Containers)”. Συγκεκριμένα, η Docker επιτρέπει την δημιουργία ενός σταθερού και αναπαραγωγικού περιβάλλοντος για τα προγράμματα, εμπεριέχοντας όλες τις απαιτούμενες εξαρτήσεις και διαμορφώσεις, και την χρήση τους αδιάλειπτα σε διαφορετικά συστήματα.

Στον πυρήνα της, η Docker επιτρέπει την δημιουργία των εικόνων Docker, οι οποίες υπηρετούν ως πρότυπα ή προσχέδια για τα κοντέινερ. Οι εικόνες Docker εγκιβωτίζουν όλα τα απαραίτητα στοιχεία και εξαρτήσεις ενός προγράμματος, όπως το λογισμικό σύστημα, τις βιβλιοθήκες, δομές, και ακόμη τον κώδικα της εφαρμογής. Αυτές οι εικόνες κατασκευάζονται με την χρήση των Dockerfiles, τα οποία είναι κείμενο-βασισμένα σενάρια τα οποία προσδιορίζουν τις οδηγίες κατασκευής της εικόνας.

Τα κοντέινερ είναι περιπτώσεις (Instances) δημιουργημένες από τις εικόνες Docker. Είναι ελαφριά και απομονωμένα περιβάλλοντα τα οποία παρέχουν ένα σταθερό περιβάλλον (runtime environment). Κάθε κοντέινερ μοιράζεται τον πυρήνα του λογισμικού συστήματος του κεντρικού συστήματος, αλλά έχει το δικό του απομονωμένο σύστημα αρχείων, διεργασίες και στοίβα δικτύου. Αυτή η απομόνωση εξασφαλίζει ότι τα κοντέινερ δεν επεμβαίνουν μεταξύ τους και προσφέρει ασφάλεια καθώς και αποδοτικότητα των πόρων.

Τα βασικά στοιχεία της λειτουργίας της Docker είναι τα εξής:

1) Docker Engine: Στον πυρήνα της Docker υπάρχει το Docker Engine. Είναι το περιβάλλον runtime υπεύθυνο για την κατασκευή, τη λειτουργία και τη διαχείριση κοντέινερ. Το Docker Engine αποτελείται από το Docker daemon (διακομιστή), και τον πελάτη. Ο διακομιστής χειρίζεται τις λειτουργίες κοντέινερ, ενώ ο πελάτης αλληλοεπιδρά με τον διακομιστή μέσω της διεπαφής γραμμής εντολών Docker (CLI) ή των API.

2) Docker Images: Η εικόνα Docker υπηρετεί ως ένα προσχέδιο ή πρότυπο κατασκευής κοντέινερ. Είναι ένα στιγμιότυπο μόνο-ανάγνωσης που περιέχει τον κώδικα εφαρμογής, το περιβάλλον, τις βιβλιοθήκες, και άλλες εξαρτήσεις απαιτούμενες για την εκτέλεση της εφαρμογής. Αυτές οι εικόνες Docker κατασκευάζονται χρησιμοποιώντας Dockerfiles, τα οποία είναι σενάρια βασισμένα σε κείμενο που καθορίζουν τα βήματα για τη δημιουργία της εικόνας. Κάθε εντολή στο Dockerfile αντιπροσωπεύει ένα επίπεδο στην εικόνα, επιτρέποντας την αποτελεσματική αποθήκευση και κοινή χρήση κοινών επιπέδων σε πολλές εικόνες.

3) Container: Τα κοντέινερ είναι ελαφριά, απομονωμένα περιβάλλοντα runtime δημιουργημένα από τις εικόνες Docker. Κάθε κοντέινερ εκτελείται ως ξεχωριστή διαδικασία στο λειτουργικό σύστημα υποδοχής, αλλά έχει το δικό του απομονωμένο σύστημα αρχείων, διεργασιών, και στοίβα δικτύου. Τα κοντέινερ παρέχουν ένα σταθερό περιβάλλον, εξασφαλίζοντας ότι η εφαρμογή συμπεριφέρεται με τον ίδιο τρόπο ανεξαρτήτως το σύστημα υποδοχής.
4) Διαδικασία εγκιβωτιοποίησης: Η διαδικασία της χρήσης της Docker περιλαμβάνει τα ακόλουθα βήματα:
    a) Δόμηση εικόνας: Οι προγραμματιστές ορίζουν ένα αρχείο Docker που περιγράφει τις εξαρτήσεις, τις διαμορφώσεις, και τις οδηγίες κατασκευής της εφαρμογής. Τα Dockerfile χρησιμοποιούνται να κατασκευάσουν μια εικόνα Docker, η οποία εμπεριέχει την εφαρμογή και τις εξαρτήσεις της.
    b) Διανομή εικόνας: Οι εικόνες Docker μπορούν να αποθηκευτούν σε αποθετήρια, όπως το Docker Hub ή ιδιωτικά μητρώα. Αυτά τα αποθετήρια χρησιμεύουν ως κεντρικές τοποθεσίες για την αποθήκευση και την κοινή χρήση των εικόνων Docker.
    c) Εκτέλεση κοντέινερ: Οι εικόνες Docker χρησιμοποιούνται για τη δημιουργία κοντέινερ, τα οποία μπορούν να εκτελεστούν σε οποιοδήποτε σύστημα που έχει εγκαταστήσει το Docker. Το Docker Engine τραβά τα απαιτούμενα επίπεδα εικόνας από το αποθετήριο, τα συνδυάζει με ένα εγγράψιμο επίπεδο και εκτελεί το κοντέινερ.

Πλεονεκτήματα:

1) Επιτρέπει την ταχεία ανάπτυξη εφαρμογών επιτρέποντας στους προγραμματιστές να συσκευάζουν τις εφαρμογές και τις εξαρτήσεις τους σε φορητά κοντέινερ. Αυτά τα κοντέινερ μπορούν εύκολα να αναπτυχθούν σε οποιοδήποτε σύστημα που έχει εγκαταστήσει η Docker, χωρίς ανησυχία για ζητήματα συμβατότητας ή διενέξεις εξαρτήσεων.

2) Η Docker παρέχει απομόνωση και ασφάλεια. Τα κοντέινερ διασφαλίζουν ότι οι εφαρμογές είναι απομονωμένες η μία από την άλλη, αποτρέποντας τυχόν παρεμβολές ή συγκρούσεις.

Αυτή η απομόνωση ενισχύει επίσης την ασφάλεια περιορίζοντας την πρόσβαση και ελέγχοντας τις αλληλεπιδράσεις μεταξύ κοντέινερ και του συστήματος υποδοχής.

3) Η Docker προάγει την αποδοτικότητα των πόρων επιτρέποντας τη λειτουργία πολλών κοντέινερ σε ένα μοναδικό μηχάνημα υποδοχής. Βελτιστοποιεί τη χρήση των πόρων και επιτρέπει την αποτελεσματική κλιμάκωση των εφαρμογών. Η Docker υποστηρίζει επίσης εκδόσεις και επαναλήψεις, επιτρέποντας την εύκολη διαχείριση των ενημερώσεων εφαρμογών και τη δυνατότητα επαναφοράς σε προηγούμενες εκδόσεις, εάν χρειάζεται.

Η Docker παρέχει πολλά οφέλη σε διάφορους τομείς ανάπτυξης λογισμικού και διαχείρισης υποδομής. Μερικοί από τους βασικούς τομείς στους οποίους βοηθά η Docker περιλαμβάνουν:

1) Απομόνωση εφαρμογής: Η Docker επιτρέπει την εγκιβωτιοποίηση των εφαρμογών και των εξαρτήσεων τους σε κοντέινερ. Τα κοντέινερ είναι αυτόνομα, απομονωμένα περιβάλλοντα που περιλαμβάνουν όλα όσα χρειάζονται για την εκτέλεση μιας εφαρμογής. Αυτή η απομόνωση διασφαλίζει ότι οι εφαρμογές εκτελούνται με συνέπεια σε διαφορετικά περιβάλλοντα, χωρίς διενέξεις ή εξαρτήσεις από συγκεκριμένες διαμορφώσεις συστήματος.

2) Φορητότητα και συνέπεια: Η Docker προωθεί τη φορητότητα συσκευάζοντας εφαρμογές και τις εξαρτήσεις τους σε δοχεία. Αυτά τα κοντέινερ μπορούν να εκτελεστούν σε οποιοδήποτε σύστημα έχει εγκαταστήσει τη Docker, ανεξάρτητα από το υποκείμενο λειτουργικό σύστημα ή την υποδομή. Αυτή η φορητότητα διασφαλίζει τη συνεπή συμπεριφορά των εφαρμογών σε περιβάλλοντα ανάπτυξης, δοκιμών και παραγωγής.

3) Απλοποιημένη ανάπτυξη: Η Docker απλοποιεί τη διαδικασία ανάπτυξης παρέχοντας ένα συνεπές περιβάλλον χρόνου εκτέλεσης για εφαρμογές. Μόλις μια εφαρμογή συσκευαστεί σε ένα κοντέινερ, μπορεί εύκολα να αναπτυχθεί σε οποιοδήποτε σύστημα με δυνατότητα Docker με μία μόνο εντολή. Η Docker εξαλείφει την ανάγκη για χειροκίνητη διαμόρφωση και διαχείριση εξαρτήσεων, μειώνοντας τα σφάλματα ανάπτυξης και απλοποιώντας τη διαδικασία εγκατάστασης.

4) Επεκτασιμότητα και αποδοτικότητα πόρων: Η τεχνολογία κοντέινερ της Docker επιτρέπει την αποτελεσματική χρήση των πόρων και την επεκτασιμότητα. Τα κοντέινερ είναι ελαφριά και μοιράζονται τον πυρήνα του κεντρικού λειτουργικού συστήματος, γεγονός που μειώνει την επιβάρυνση των πόρων και παρέχει ταχύτερους χρόνους εκκίνησης σε σύγκριση με τις παραδοσιακές εικονικές μηχανές. Η Docker επιτρέπει επίσης την οριζόντια κλιμάκωση των εφαρμογών αναπαράγοντας κοντέινερ σε πολλούς κεντρικούς υπολογιστές ή χρησιμοποιώντας εργαλεία ενορχήστρωσης όπως τη Docker Swarm ή το Kubernetes.

5) Έλεγχος εκδόσεων και Rollback: Η Docker διευκολύνει τον έλεγχο εκδόσεων για εφαρμογές, επιτρέποντας στους προγραμματιστές να συσκευάζουν συγκεκριμένες εκδόσεις της εφαρμογής και τις εξαρτήσεις της σε κοντέινερ. Αυτό διασφαλίζει ότι η εφαρμογή μπορεί να αναπαραχθεί

αξιόπιστα ανά πάσα στιγμή. Επιπλέον, η Docker επιτρέπει την εύκολη επαναφορά σε προηγούμενες εκδόσεις με απλή αναδιάταξη του αντίστοιχου κοντέινερ.

6) Συνεργασία και αναπαραγωγιμότητα: Η Docker απλοποιεί τη συνεργασία μεταξύ προγραμματιστών και ομάδων παρέχοντας ένα τυποποιημένο περιβάλλον για την ανάπτυξη εφαρμογών. Οι προγραμματιστές μπορούν να μοιράζονται εικόνες Docker και αρχεία Docker, διασφαλίζοντας ότι όλοι λειτουργούν με τις ίδιες εξαρτήσεις και διαμορφώσεις. Η Docker διευκολύνει επίσης την αναπαραγωγιμότητα, επιτρέποντας σε ερευνητές και επιστήμονες να μοιράζονται και να αναπαράγουν πειραματικές ρυθμίσεις συσκευάζοντάς τις σε δοχεία.

7) Βελτιστοποίηση υποδομής: Η Docker απλοποιεί τη διαχείριση της υποδομής αφαιρώντας τις υποκείμενες λεπτομέρειες της. Οι προγραμματιστές και οι διαχειριστές συστημάτων μπορούν να επικεντρωθούν στην ανάπτυξη και διαχείριση εφαρμογών αντί να ασχολούνται με σύνθετες διαμορφώσεις υποδομής. Η Docker επιτρέπει επίσης τον διαχωρισμό των εφαρμογών σε μικροϋπηρεσίες, επιτρέποντας την πιο αποτελεσματική κατανομή και διαχείριση πόρων.

Το Docker έχει ένα ευρύ φάσμα περιπτώσεων χρήσης

1) .DevOps και CI/CD: Η Docker υιοθετείται ευρέως στις πρακτικές DevOps για τον εξορθολογισμό των διαδικασιών ανάπτυξης και ανάπτυξης λογισμικού. Επιτρέπει στις ομάδες να συσκευάζουν εφαρμογές, μαζί με τις εξαρτήσεις τους, σε κοντέινερ. Αυτά τα κοντέινερ μπορούν στη συνέχεια να αναπτυχθούν με συνέπεια σε διαφορετικά περιβάλλοντα, διασφαλίζοντας ότι η εφαρμογή συμπεριφέρεται με τον ίδιο τρόπο στα στάδια ανάπτυξης, δοκιμής και παραγωγής. Οι αγωγοί Συνεχούς Ολοκλήρωσης/Συνεχούς Ανάπτυξης (CI/CD) συχνά αξιοποιούν τη Docker για την αυτοματοποίηση της κατασκευής, της δοκιμής και της ανάπτυξης εφαρμογών.

2) Αρχιτεκτονική Microservices: Η Docker είναι μια δημοφιλής επιλογή για την κατασκευή και την ανάπτυξη αρχιτεκτονικών που βασίζονται σε microservices. Οι μικροϋπηρεσίες είναι μικρές, ανεξάρτητα αναπτυσσόμενες υπηρεσίες που συνεργάζονται για να σχηματίσουν μια μεγαλύτερη εφαρμογή. Τα ελαφριά και απομονωμένα δοχεία του Docker διευκολύνουν τη διαχείριση και την κλιμάκωση μεμονωμένων μικροϋπηρεσιών. Κάθε microservice μπορεί να συσκευαστεί σε ξεχωριστό δοχείο, επιτρέποντας ανεξάρτητη ανάπτυξη, δοκιμή και εφαρμογή.

3) Hybrid και Multi-Cloud Deployments: Η τεχνολογία εγκιβωτιοποίησης της Docker παρέχει φορητότητα σε διαφορετικούς παρόχους cloud και υποδομές εσωτερικής εγκατάστασης. Με τη Docker, οι εφαρμογές μπορούν να συσκευαστούν μία φορά και να αναπτυχθούν με συνέπεια σε διάφορες πλατφόρμες cloud, όπως οι Υπηρεσίες Ιστού Amazon (AWS), το Microsoft Azure ή η Google Cloud Platform (GCP). Αυτή η ευελιξία δίνει τη δυνατότητα στους οργανισμούς να υιοθετήσουν μια υβριδική ή Multi-Cloud στρατηγική, αξιοποιώντας τα οφέλη διαφορετικών περιβαλλόντων cloud μεν, διατηρώντας δε παράλληλα τη συνέπεια στην εφαρμογή της..

4) Αναπαραγωγική Έρευνα και Ανάπτυξη Εφαρμογών: Η Docker είναι πολύτιμο σε περιβάλλοντα έρευνας και ανάπτυξης όπου η αναπαραγωγιμότητα είναι ζωτικής σημασίας. Οι επιστήμονες και οι ερευνητές μπορούν να δημιουργήσουν εικόνες Docker που ενσωματώνουν τις πειραματικές ρυθμίσεις τους, συμπεριλαμβανομένων συγκεκριμένων εκδόσεων βιβλιοθηκών λογισμικού και εξαρτήσεων. Με την κοινή χρήση αυτών των εικόνων Docker, άλλοι ερευνητές μπορούν να αναπαράγουν το ίδιο ακριβώς περιβάλλον, διασφαλίζοντας την αναπαραγωγιμότητα των πειραμάτων και διευκολύνοντας τη συνεργασία.

5) Περιβάλλοντα δοκιμής και διασφάλισης ποιότητας: Η Docker απλοποιεί τη ρύθμιση και τη διαχείριση περιβαλλόντων δοκιμών και διασφάλισης ποιότητας (Quality Assurance). Οι υπεύθυνοι δοκιμών μπορούν να χρησιμοποιήσουν τη Docker για να δημιουργήσουν απομονωμένα κοντέινερ με συγκεκριμένες διαμορφώσεις για διαφορετικά σενάρια δοκιμών. Αυτό εξαλείφει την ανάγκη εγκατάστασης και διατήρησης χωριστών φυσικών ή εικονικών μηχανών για δοκιμές, καθιστώντας τη διαδικασία δοκιμής πιο αποτελεσματική και επεκτάσιμη.

Η ευελιξία και η φορητότητα του το καθιστούν εφαρμόσιμο σε διάφορους κλάδους και περιβάλλοντα, δίνοντας τη δυνατότητα στους οργανισμούς να βελτιώσουν τις πρακτικές ανάπτυξης και ανάπτυξης λογισμικού τους.

Εν ολίγοις, απλοποιεί τη συσκευασία εφαρμογών, και την ανάπτυξη και τη διαχείριση τους, παρέχοντας οφέλη όπως ταχεία ανάπτυξη, επεκτασιμότητα, φορητότητα, αναπαραγωγιμότητα καθώς και αποτελεσματική χρήση πόρων. Η Docker έχει γίνει ένα θεμελιώδες εργαλείο στη σύγχρονη ανάπτυξη λογισμικού, δίνοντας τη δυνατότητα σε προγραμματιστές και οργανισμούς να δημιουργούν και να αναπτύσσουν εφαρμογές με ευκολία και συνέπεια.

5.5 Tesseract OCR

Η βιβλιοθήκη Tesseract OCR (Optical Character Recognition) είναι ένα εργαλείο λογισμικού ανοιχτού κώδικα που επιτρέπει την αναγνώριση και εξαγωγή κειμένου από εικόνες ή σαρωμένα έγγραφα. Αναπτύχθηκε αρχικά από τα εργαστήρια Hewlett-Packard[18] και αργότερα συντηρήθηκε και χρηματοδοτήθηκε από την Google, το Tesseract χρησιμοποιείται ευρέως για διάφορες εφαρμογές που περιλαμβάνουν αναγνώριση κειμένου.

Το Tesseract έχει σχεδιαστεί για να επεξεργάζεται εικόνες και να αναγνωρίζει τους χαρακτήρες μέσα σε αυτές, μετατρέποντας την οπτική αναπαράσταση του κειμένου σε δεδομένα κειμένου αναγνώσιμα από μηχανή. Υποστηρίζει περισσότερες από 100 γλώσσες, συμπεριλαμβανομένων σύνθετων γραφικών χαρακτήρων.

Η βιβλιοθήκη ακολουθεί μια διαδικασία πολλαπλών βημάτων για την αναγνώριση και εξαγωγή κειμένου από εικόνες. Η γενική ροή εργασίας του Tesseract OCR περιλαμβάνει τα ακόλουθα βήματα:

1) Προεπεξεργασία εικόνας: Πριν από την εκτέλεση OCR, η εικόνα υποβάλλεται σε προεπεξεργασία για βελτίωση του κειμένου και βελτίωση της ακρίβειας αναγνώρισης. Αυτό το στάδιο προεπεξεργασίας μπορεί να περιλαμβάνει λειτουργίες όπως δυαδοποίηση εικόνας (μετατροπή της εικόνας σε ασπρόμαυρη), αφαίρεση θορύβου, αποσκλήρυνση (διόρθωση περιστροφής εικόνας) και ανάλυση διάταξης κειμένου (προσδιορισμός περιοχών, γραμμών και λέξεων).

2) Τμηματοποίηση χαρακτήρων: Σε αυτό το βήμα, το Tesseract αναλύει την προεπεξεργασμένη εικόνα για να αναγνωρίσει μεμονωμένους χαρακτήρες και να τους διαχωρίσει. Ανιχνεύει περιγράμματα, σχήματα και μοτίβα για να τμηματοποιήσει την εικόνα σε διακριτές περιοχές χαρακτήρων.

3) Εξαγωγή χαρακτηριστικών: Το Tesseract εξάγει ένα σύνολο χαρακτηριστικών από κάθε τμηματοποιημένο χαρακτήρα για να αναπαραστήσει τα οπτικά του χαρακτηριστικά. Αυτά τα χαρακτηριστικά μπορεί να περιλαμβάνουν πτυχές όπως το πλάτος διαδρομής, την καμπυλότητα, τις τομές γραμμών και άλλες σχετικές πληροφορίες που βοηθούν στη διάκριση ενός χαρακτήρα από τον άλλο.

4) Ταξινόμηση χαρακτήρων: Το Tesseract χρησιμοποιεί έναν συνδυασμό τεχνικών αναγνώρισης προτύπων και μηχανικής εκμάθησης για την ταξινόμηση των τμηματοποιημένων χαρακτήρων με βάση τα εξαγόμενα χαρακτηριστικά. Χρησιμοποιεί στατιστικά μοντέλα, όπως τεχνητά νευρωνικά δίκτυα, για να αναγνωρίζει και να εκχωρεί πιθανότητες σε διαφορετικές κατηγορίες χαρακτήρων.

5) Ανάλυση γλώσσας και περιβάλλοντος: Το Tesseract εκτελεί γλωσσική ανάλυση για να χειριστεί διάφορα κείμενα, γλώσσες και συστήματα γραφής. Λαμβάνει υπόψη το πλαίσιο, τη γραμματική και τα ειδικά χαρακτηριστικά της γλώσσας κατά τη διαδικασία αναγνώρισης. Το Tesseract εκπαιδεύεται σε μεγάλα σύνολα δεδομένων για διαφορετικές γλώσσες, επιτρέποντάς του να χειρίζεται πολύπλοκα σενάρια και διακριτικά σημάδια.

6) Μετα-επεξεργασία και έξοδος: Μόλις αναγνωριστούν οι χαρακτήρες, το Tesseract εφαρμόζει τεχνικές μετα-επεξεργασίας για να βελτιώσει τα αποτελέσματα. Αυτό περιλαμβάνει τη διόρθωση σφαλμάτων, την επίλυση ασαφειών και τη βελτίωση της συνολικής ακρίβειας του αναγνωρισμένου κειμένου. Η τελική έξοδος μπορεί να είναι σε διάφορες μορφές, όπως απλό κείμενο ή δομημένα δεδομένα, ανάλογα με τις απαιτήσεις της εφαρμογής.

Η βιβλιοθήκη Tesseract OCR (Optical Character Recognition) έχει ποικίλες εφαρμογές σε διαφορετικούς κλάδους. Μερικές από τις βασικές χρήσεις της βιβλιοθήκης Tesseract OCR περιλαμβάνουν:

1) Ψηφιοποίηση εγγράφων: Το Tesseract χρησιμοποιείται συνήθως για τη μετατροπή φυσικών ή σαρωμένων εγγράφων σε ψηφιακή μορφή. Επιτρέπει την εξαγωγή κειμένου από εικόνες ή

αρχεία PDF, επιτρέποντας τη μετατροπή στατικών εγγράφων σε επεξεργάσιμα και αναζητήσιμα δεδομένα κειμένου. Αυτό είναι ιδιαίτερα χρήσιμο σε κλάδους όπως της νομικής, της χρηματοοικονομικής, της υγειονομική περίθαλψης και της εκπαίδευσης, όπου μεγάλοι όγκοι εγγράφων πρέπει να υποβληθούν σε επεξεργασία και αρχειοθέτηση.

2) Εξαγωγή και ανάλυση δεδομένων: Το Tesseract OCR διευκολύνει την εξαγωγή σχετικών πληροφοριών από έγγραφα ή εικόνες για σκοπούς εξόρυξης δεδομένων, ανάλυσης και ανάκτησης πληροφοριών. Επιτρέπει την αυτοματοποιημένη εξαγωγή δεδομένων όπως ονόματα, διευθύνσεις, ημερομηνίες, αριθμούς τιμολογίων και άλλες δομημένες πληροφορίες. Αυτό είναι πολύτιμο σε κλάδους όπως της χρηματοοικονομικής, των ασφαλειών, του ηλεκτρονικού εμπορίου και της έρευνας, όπου τα δεδομένα πρέπει να εξαχθούν από διάφορες πηγές για περαιτέρω επεξεργασία και ανάλυση.

3) Αναγνώριση κειμένου σε εικόνες και στιγμιότυπα οθόνης: Το Tesseract χρησιμοποιείται ευρέως για την εξαγωγή κειμένου από εικόνες που τραβήχτηκαν από κάμερες ή στιγμιότυπα οθόνης. Επιτρέπει την αναγνώριση κειμένου σε εικόνες ή γραφικές διεπαφές χρήστη (GUI), καθιστώντας το πολύτιμο για εφαρμογές όπως η εξαγωγή κειμένου από εικόνες μέσων κοινωνικής δικτύωσης, η αναζήτηση βάσει εικόνων, η αυτοματοποίηση εισαγωγής δεδομένων και τα συστήματα επαυξημένης πραγματικότητας(AR).

4) Μετάφραση και πολύγλωσση υποστήριξη: Η γλωσσική υποστήριξη της Tesseract περιλαμβάνει περισσότερες από 100 γλώσσες, καθιστώντας την κατάλληλη για πολύγλωσσες εφαρμογές. Μπορεί να χειριστεί κείμενα από αριστερά προς τα δεξιά και από δεξιά προς τα αριστερά, καθώς και γλώσσες με περίπλοκες δομές και διακριτικά σημάδια. Το Tesseract OCR χρησιμοποιείται σε μεταφραστικές υπηρεσίες, διαχείριση πολυγλωσσικού περιεχομένου, τοπική προσαρμογή και εφαρμογές ανάλυσης γλώσσας.

5) Προσβασιμότητα και υποβοηθητική τεχνολογία: Το Tesseract OCR διαδραματίζει κρίσιμο ρόλο στη βελτίωση της προσβασιμότητας για άτομα με προβλήματα όρασης. Επιτρέπει τη μετατροπή έντυπου ή χειρόγραφου κειμένου σε αναγνώσιμες μορφές, διευκολύνοντας τη μετατροπή κειμένου σε ομιλία, προγράμματα ανάγνωσης οθόνης και βοηθητικές τεχνολογίες. Το Tesseract χρησιμοποιείται σε εφαρμογές που βοηθούν άτομα με προβλήματα όρασης στην πρόσβαση σε ψηφιακό περιεχόμενο και εγγράφων.

6) Αυτοματισμός και απόδοση ροής εργασίας: Το Tesseract OCR είναι ενσωματωμένο σε συστήματα αυτοματισμού και ροές εργασίας για τη βελτίωση της απόδοσης. Επιτρέπει την αυτοματοποιημένη εισαγωγή δεδομένων, την επεξεργασία εντύπων και την ταξινόμηση εγγράφων, μειώνοντας τη μη αυτόματη προσπάθεια και εξορθολογίζοντας τις επιχειρηματικές διαδικασίες. Το Tesseract βοηθά στην αυτοματοποίηση εργασιών σε κλάδους όπως τους προηγουμένως αναφερόμενους, όπου η εξαγωγή και η επεξεργασία κειμένου είναι βασικά στοιχεία.

Η βιβλιοθήκη Tesseract OCR προσφέρει API και την δυνατότητα στους προγραμματιστές να ενσωματώσουν τις λειτουργίες της στις εφαρμογές τους. Υποστηρίζει πολλές γλώσσες προγραμματισμού, συμπεριλαμβανομένων των C++, Python, Java και άλλων, επιτρέποντας στους προγραμματιστές να αξιοποιήσουν τις δυνατότητες OCR στα έργα τους.

Ακολουθώντας αυτήν την ολοκληρωμένη διαδικασία προεπεξεργασίας εικόνας, τμηματοποίησης χαρακτήρων, εξαγωγής χαρακτηριστικών, ταξινόμησης και ανάλυσης γλώσσας, το Tesseract OCR μπορεί να εξάγει με ακρίβεια κείμενο από εικόνες και σαρωμένα έγγραφα, επιτρέποντας μια πληθώρα εφαρμογών που απαιτούν αυτοματοποιημένη αναγνώριση κειμένου.

Η ευελιξία του και η φύση του ανοιχτού κώδικα το καθιστούν εφαρμόσιμο σε διάφορους τομείς, επιτρέποντας στους προγραμματιστές να ενσωματώνουν ισχυρές δυνατότητες αναγνώρισης κειμένου στις εφαρμογές και τα συστήματά τους.

6. Source Code – Πηγαίος Κώδικας

```
1  #include <fstream>
2  #include <iostream>
3  #include <string>
4  #include <filesystem>
5  #include <chrono>
6  #include <leptonica/allheaders.h>
7  #include <tesseract/baseapi.h>
8  #include <opencv2/opencv.hpp>
```

#include <fstream>: Αυτή η οδηγία περιλαμβάνει το αρχείο κεφαλίδας fstream, το οποίο παρέχει κλάσεις και συναρτήσεις για την ανάγνωση και εγγραφή σε αρχεία.

#include <iostream>: Αυτή η οδηγία περιλαμβάνει το αρχείο κεφαλίδας iostream, το οποίο περιέχει τα αντικείμενα της τυπικής εισόδου/εξόδου όπως το cin και το cout για τη διαχείριση εισόδου και εξόδου.

#include <string>: Αυτή η οδηγία περιλαμβάνει το αρχείο κεφαλίδας string, το οποίο παρέχει την κλάση string και διάφορες συναρτήσεις για την χειρισμό συμβολοσειρών.

#include <filesystem>: Αυτή η οδηγία περιλαμβάνει το αρχείο κεφαλίδας filesystem, το οποίο παρέχει κλάσεις και συναρτήσεις για την εργασία με αρχεία και φακέλους, εισήχθη στο πρότυπο C++17.

#include <chrono>: Αυτή η οδηγία περιλαμβάνει το αρχείο κεφαλίδας chrono, το οποίο παρέχει κλάσεις και συναρτήσεις για την εργασία με διαρκείες χρόνου και χρονικά σημεία.

#include <leptonica/allheaders.h>: Αυτή η οδηγία περιλαμβάνει το αρχείο κεφαλίδας allheaders.h από τη βιβλιοθήκη Leptonica, η οποία είναι μια ανοικτού κώδικα βιβλιοθήκη για επεξεργασία και χειρισμό εικόνων.

#include <tesseract/baseapi.h>: Αυτή η οδηγία περιλαμβάνει το αρχείο κεφαλίδας baseapi.h από τη βιβλιοθήκη Tesseract, η οποία είναι ένας μηχανισμός OCR (Optical Character Recognition) που χρησιμοποιείται για την εξαγωγή κειμένου από εικόνες.

#include <opencv2/opencv.hpp>: Αυτή η οδηγία περιλαμβάνει το αρχείο κεφαλίδας opencv.hpp από τη βιβλιοθήκη OpenCV, η οποία είναι μια δημοφιλής βιβλιοθήκη ανοικτού κώδικα για την επεξεργασία και ανάλυση εικόνων με διάφορες λειτουργίες επεξεργασίας εικόνας και ανάλυσης εικόνας.

Περιλαμβάνοντας αυτές τις κεφαλίδες, καθίσταται διαθέσιμη η αντίστοιχη λειτουργικότητα και οι κλάσεις στο πρόγραμμά σας, επιτρέποντάς σας να εκτελέσετε εργασίες που σχετίζονται με τη χειρισμό αρχείων, την επεξεργασία συμβολοσειρών, τη μέτρηση χρόνου, την επεξεργασία εικόνας και την OCR χρησιμοποιώντας τις βιβλιοθήκες Leptonica και Tesseract, καθώς και διάφορες λειτουργίες υπολογιστικής όρασης χρησιμοποιώντας το OpenCV.

```
10  #define MAX_THRESHOLD 255
11
12  cv::Mat image, src_gray;
13  int threshold = 100;
14  cv::RNG rng(12345);
```

#define MAX_THRESHOLD 255: Αυτή η δήλωση #define ορίζει μια σταθερά MAX_THRESHOLD με την τιμή 255. Η χρήση του #define μας επιτρέπει να ορίσουμε μια σταθερά που θα αντικαθίσταται αυτόματα με την τιμή της κατά τη διάρκεια της μεταγλώττισης του προγράμματος.

cv::Mat image, src_gray;: Δημιουργεί δύο αντικείμενα τύπου cv::Mat, το image και το src_gray. Το cv::Mat είναι η κλάση του OpenCV που χρησιμοποιείται για την αναπαράσταση και επεξεργασία εικόνων.

int threshold = 100;: Δημιουργεί μια μεταβλητή threshold τύπου ακέραιο και την αρχικοποιεί με την τιμή 100. Αυτή η μεταβλητή χρησιμοποιείται πιθανότατα για τον προσδιορισμό του ορίου ή του κατωφλίου για επεξεργασία εικόνας.cv::RNG rng(12345);:

Δημιουργεί ένα αντικείμενο rng τύπου cv::RNG (Random Number Generator) και το αρχικοποιεί με την τιμή 12345. Αυτό το αντικείμενο χρησιμοποιείται για την παραγωγή τυχαίων αριθμών στο πλαίσιο της επεξεργασίας εικόνας.

Ο παραπάνω κώδικας καθορίζει μια σταθερά, δημιουργεί δύο αντικείμενα cv::Mat για την αναπαράσταση εικόνων, αρχικοποιεί μια μεταβλητή threshold και δημιουργεί ένα αντικείμενο τύπου cv::RNG για την παραγωγή τυχαίων αριθμών.

```
77  void threshold_callback(int, void* )
78  {
79      cv::Mat canny_output;
80      cv::Canny( src_gray, canny_output, threshold, threshold*2 );
81      std::vector<std::vector<cv::Point> > contours;
82      findContours( canny_output, contours, cv::RETR_TREE, cv::CHAIN_APPROX_SIMPLE );
83      std::vector<std::vector<cv::Point> > contours_poly( contours.size() );
84      std::vector<cv::Rect> boundRect( contours.size() );
85      std::vector<cv::Point2f>centers( contours.size() );
86      std::vector<float>radius( contours.size() );
87      for( size_t i = 0; i < contours.size(); i++ )
88      {
89          cv::approxPolyDP( contours[i], contours_poly[i], 10, true );
90          boundRect[i] = cv::boundingRect( contours_poly[i] );
91          cv::minEnclosingCircle( contours_poly[i], centers[i], radius[i] );
92      }
93      cv::Mat drawing = cv::Mat::zeros( canny_output.size(), CV_8UC3 );
94      for( size_t i = 0; i< contours.size(); i++ )
95      {
96          cv::Scalar color = cv::Scalar( rng.uniform(0, 256), rng.uniform(0,256), rng.uniform(0,256) );
97          drawContours( drawing, contours_poly, (int)i, color );
98          rectangle( drawing, boundRect[i].tl(), boundRect[i].br(), color, 2 );
99      }
100     cv::imwrite("../contours/img_contours.jpg", drawing);
101 }
```

Ο παραπάνω κώδικας περιέχει τη συνάρτηση threshold_callback, η οποία είναι ένας callback handler για την ρύθμιση του ορίου. Ας δούμε τι συμβαίνει μέσα στη συνάρτηση:

Δημιουργείται ένα αντικείμενο canny_output τύπου cv::Mat. Αυτό το αντικείμενο θα χρησιμοποιηθεί για την αποθήκευση των αποτελεσμάτων της μεθόδου cv::Canny, η οποία εκτελεί τον αλγόριθμο Canny για τον εντοπισμό ακμών στην src_gray με το όριο threshold και το διπλάσιο του.

Δημιουργούνται αντικείμενα για την αποθήκευση των συνόρων, των πολυγώνων που προσεγγίζουν τα σύνορα, των ορθογώνιων πλαισίων που περικλείουν τα πολύγωνα, καθώς και των κέντρων και των ακτίνων για κάθε πολύγωνο που βρέθηκε.

Ένας βρόχος for περνάει από κάθε πολύγωνο που βρέθηκε και εκτελεί τις εξής ενέργειες:

Εφαρμόζει τη μέθοδο cv::approxPolyDP για να προσεγγίσει το πολύγωνο με καμπύλες που έχουν το μήκος 10 και τις αποθηκεύει στο αντίστοιχο contours_poly.

Υπολογίζει το ορθογώνιο πλαίσιο που περικλείει το προσεγγισμένο πολύγωνο και το αποθηκεύει στο boundRect.

Υπολογίζει το ελάχιστο εφαπτόμενο κύκλο για το προσεγγισμένο πολύγωνο και αποθηκεύει το κέντρο και την ακτίνα του στα αντίστοιχα centers και radius.

Δημιουργείται ένα αντικείμενο drawing τύπου cv::Mat με μέγεθος ίδιο με το canny_output και τύπο CV_8UC3. Αυτό το αντικείμενο θα χρησιμοποιηθεί για την απεικόνιση των αποτελεσμάτων.

Ένας άλλος βρόχος for περνά από κάθε πολύγωνο και εκτελεί τις εξής ενέργειες:
Δημιουργεί ένα τυχαίο χρώμα (color) με τη βοήθεια της RNG(random number generator) και το εφαρμόζει στα πολύγωνα και τα ορθογώνια πλαίσια με τη χρήση των συναρτήσεων drawContours και rectangle αντίστοιχα και τέλος, η εικόνα drawing αποθηκεύεται σε ένα αρχείο με το όνομα "../contours/img_contours.jpg".

Ο παραπάνω κώδικας εκτελεί τον αλγόριθμο Canny για την εντοπισμό ακμών στην src_gray, βρίσκει τα περίγραμματα των αντικειμένων στην εικόνα, προσεγγίζει αυτά τα περίγραμματα με πολύγωνα, και σχεδιάζει τα πολύγωνα και τα ορθογώνια πλαίσια πάνω στην εικόνα drawing. Το τελικό αποτέλεσμα αποθηκεύεται σε ένα αρχείο εικόνας.

```
18  int main(int argc, char **argv)
19  {
20    std::ofstream outfile;
21    std::string imagePath = "../images";
22
23    tesseract::TessBaseAPI *api = new tesseract::TessBaseAPI();
24    api->Init(NULL, "eng", tesseract::OEM_LSTM_ONLY);
25    api->SetPageSegMode(tesseract::PSM_AUTO);
26    api->SetVariable("debug_file", "tesseract.log");
27
28    cv::VideoCapture cap(0);
29
30    if (!cap.isOpened())
31    {
32      std::cerr << "Error opening camera" << std::endl;
33      return -1;
34    }
35
36    cv::Mat frame;
37    cap >> frame;
38
39    if (frame.empty())
40    {
41      std::cerr << "Error finding frame" << std::endl;
42    }
43
44    cv::imwrite("../images/img.jpg", frame);
45    cv::cvtColor(frame, src_gray, cv::COLOR_BGR2GRAY);
46    cv::blur(src_gray, src_gray, cv::Size(3,3));
47
48    cv::createTrackbar("Canny Threshold:", "SRC", &threshold, MAX_THRESHOLD, threshold_callback);
49    threshold_callback(0, 0);
50
```

Ο παραπάνω κώδικας περιέχει τη συνάρτηση main, η οποία είναι η κύρια συνάρτηση του προγράμματος. Ας δούμε τι συμβαίνει μέσα στη συνάρτηση:

Δημιουργούνται μεταβλητές για την αποθήκευση του αποτελέσματος σε ένα αρχείο (outfile) και του διαδρόμου της εικόνας (imagePath).

Δημιουργείται ένα αντικείμενο api τύπου tesseract::TessBaseAPI. Αυτό το αντικείμενο χρησιμοποιείται για την αναγνώριση κειμένου με τη βοήθεια του Tesseract OCR.

Αρχικοποιούνται οι παράμετροι του Tesseract OCR, όπως η γλώσσα ("eng"), η μέθοδος OCR (tesseract::OEM_LSTM_ONLY) και η λειτουργία ανάλυσης σελίδας (tesseract::PSM_AUTO). Επίσης, ορίζεται το αρχείο καταγραφής αποσφαλμάτωσης ("tesseract.log").

Δημιουργείται ένα αντικείμενο cap τύπου cv::VideoCapture για την ανοίγματα της κάμερας.

Ελέγχεται αν η κάμερα είναι ανοικτή. Αν δεν είναι, τυπώνεται ένα μήνυμα λάθους και το πρόγραμμα τερματίζεται.

Δημιουργείται ένα αντικείμενο frame τύπου cv::Mat και λαμβάνεται μια καρέ από την κάμερα.

Ελέγχεται αν το καρέ είναι κενό (δηλαδή δεν βρέθηκε κανένα πλαίσιο). Αν είναι, τυπώνεται ένα μήνυμα λάθους.

Το καρέ αποθηκεύεται σε ένα αρχείο εικόνας με το όνομα "../images/img.jpg".

Το καρέ μετατρέπεται σε κλίμακα του γκρι (src_gray) και εφαρμόζεται φίλτρο θολώματος με μέγεθος πυξίδας 3x3.

Δημιουργείται ένα παράθυρο χειριστηρίου με έναν κύλινδρο "Canny Threshold" στην οθόνη "SRC", με τιμή από 0 έως MAX_THRESHOLD και τη συνάρτηση threshold_callback ως αντίστοιχη επιστροφή κλήσης. Επίσης, καλείται η συνάρτηση threshold_callback με αρχικές παραμέτρους 0 και 0.

```
52  for (const auto &fn : std::filesystem::directory_iterator(imagePath))
53  {
54    auto start = std::chrono::steady_clock::now();
55    auto filepath = fn.path();
56    std::cout << "Detecting text in " << filepath << std::endl;
57
58    image = cv::imread(filepath, 1);
59
60    api->SetImage(image.data, image.cols, image.rows, 3, image.step);
61    std::string outText = api->GetUTF8Text();
62    std::cout << outText << std::endl;
63
64    auto end = std::chrono::steady_clock::now();
65    std::chrono::duration<double, std::milli> diff = end - start;
66    std::cout << "Computation time: " << diff.count() << "ms" << std::endl;
67
68    outfile.open("../output/text.txt");
69    outfile << outText;
70  }
```

Σε αυτό το σημείο, η εκτέλεση του προγράμματος εισέρχεται σε ένα βρόχο for που περνά από κάθε αρχείο στον φάκελο imagePath. Για κάθε αρχείο, καταγράφεται ο χρόνος έναρξης της επεξεργασίας, παίρνεται η τοποθεσία του αρχείου (filepath) και τυπώνεται ένα μήνυμα.

Η εικόνα αναγνωρίζεται από τον Tesseract OCR χρησιμοποιώντας τη μέθοδο SetImage και αποθηκεύεται το αναγνωρισμένο κείμενο στη μεταβλητή outText.

Το αναγνωρισμένο κείμενο τυπώνεται στην οθόνη.

Υπολογίζεται ο χρόνος επεξεργασίας και τυπώνεται στην οθόνη.

Ανοίγεται το αρχείο εξόδου "output/text.txt" για εγγραφή και το αναγνωρισμένο κείμενο γράφεται στο αρχείο εξόδου.

```
72      system("flite -f ../output/text.txt");
73      api->End();
74      return 0;
75  }
```

Τελικά, εκτελείται η εντολή συστήματος "flite -f ../output/text.txt" για την αναπαραγωγή του αναγνωρισμένου κειμένου με τη βοήθεια του Flite TTS (Text-to-Speech).

Ο Tesseract OCR καθαρίζεται (End()) και το πρόγραμμα τερματίζεται.

Αυτός είναι ο σημαντικός κώδικας που εκτελείται στη συνάρτηση main. Ο πυρήνας του προγράμματος είναι η αναγνώριση κειμένου στις εικόνες χρησιμοποιώντας τον Tesseract OCR και η αποθήκευση του αποτελέσματος σε ένα αρχείο κειμένου όπου θα αναπαραχθεί από την βιβλιοθήκη flite.

6.1 Dockerfile

```
FROM ubuntu:18.04

WORKDIR /home/pi

RUN apt-get update && apt-get install -y build-essential && \
    apt-get install -y cmake vim git libgtk2.0-dev pkg-config \
    libavcodec-dev libavformat-dev libswscale-dev && \
    apt-get install -y python-dev python-numpy libtbb2 \
    libtbb-dev libjpeg-dev libpng-dev libtiff-dev libdc1394-22-dev \
    && apt-get clean && rm -rf /var/lib/apt-lists/*

RUN git clone https://github.com/opencv/opencv.git
RUN git clone https://github.com/opencv/opencv_contrib.git

RUN cd opencv && mkdir -p build && cd build && \
    cmake -D CMAKE_BUILD_TYPE=Release -D \
          CMAKE_INSTALL_PREFIX=/usr/local -D \
          OPENCV_GENERATE_PKGCONFIG=ON -D \
          OPENCV_EXTRA_MODULES_PATH=/home/pi/opencv_contrib/modules .. \
    && make -j4 && make install

RUN apt-get install -y automake ca-certificates g++-8 \
    git libtool libleptonica-dev make pkg-config && \
    apt-get install -y --no-install-recommends asciidoc \
    docbook-xsl xsltproc && apt-get install -y libpango1.0-dev \
    && apt-get install -y libicu-dev libpango1.0-dev libcairo2-dev

RUN git clone https://github.com/tesseract-ocr/tesseract.git

RUN cd tesseract && ./autogen.sh && ./configure \
    && make && make install && make training && \
    make training-install && ldconfig

#RUN mkdir /usr/local/share/tessdata

RUN curl -o /usr/local/share/tessdata/eng.traineddata \
    https://raw.githubusercontent.com/tesseract-ocr/tessdata_best/master/eng.traineddata

RUN apt-get update && \
    apt-get install -y pulseaudio-utils alsa-utils && \
    rm -rf /var/lib/apt/lists/*

ENV PULSE_SERVER=unix:${XDG_RUNTIME_DIR}/pulse/native
ENV PULSE_COOKIE=/run/pulse/cookie

VOLUME /run/user/${UID}/pulse

ARG UID=1000
ENV USER_ID=${UID}
RUN usermod -u ${USER_ID} pulseaudio

USER pulseaudio
CMD pulseaudio --system --disallow-exit --verbose

RUN apt-get update && apt-get install -y flite
```

Το αρχείο Docker που δίνεται περιέχει μια σειρά από ενέργειες για την δημιουργία ενός εικονικού περιβάλλοντος με τις απαραίτητες εξαρτήσεις για την εκτέλεση εφαρμογών που χρησιμοποιούν την OpenCV και το Tesseract OCR. Ας δούμε τη λειτουργία του κάθε βήματος:

Ορίζεται η εικόνα βάσης ως "ubuntu:18.04".

Ορίζεται ο τρέχων φάκελος εργασίας στο /home/pi.

Εκτελείται η εντολή apt-get update για την ενημέρωση των διαθέσιμων πακέτων. Στη συνέχεια, εγκαθίστανται τα πακέτα build-essential, cmake, vim, git, libgtk2.0-dev, pkg-config, libavcodec-dev, libavformat-dev, libswscale-dev, python-dev, python-numpy, libtbb2, libtbb-dev, libjpeg-dev, libpng-dev, libtiff-dev, libdc1394-22-dev. Τέλος, γίνεται καθαρισμός των αρχείων από τα πακέτα που κατέβηκαν για να εξοικονομηθεί χώρος.

Εκτελούνται οι εντολές git clone για τη λήψη των απαραίτητων αποθετηρίων της OpenCV και των πρόσθετων modules.

Μεταβαίνει στον φάκελο opencv και δημιουργείται ο φάκελος build. Εκτελείται η εντολή cmake για τη δημιουργία των αρχείων Makefile και η εντολή make -j4 για τη μεταγλώττιση του κώδικα της OpenCV. Τέλος, εγκαθίσταται η OpenCV στο σύστημα.

Εγκαθίστανται διάφορα πακέτα για την υποστήριξη του Tesseract OCR, όπως automake, ca-certificates, g++-8, libtool, libleptonica-dev, make, pkg-config, asciidoc, docbook-xsl, xsltproc, libpango1.0-dev, libicu-dev, libcairo2-dev.

Εκτελείται η εντολή git clone για τη λήψη του αποθετηρίου του Tesseract OCR.

Μεταβαίνει στον φάκελο tesseract και εκτελούνται οι εντολές autogen.sh, configure, make, make install, make training, make training-install για την μεταγλώττιση, εγκατάσταση και εκπαίδευση του Tesseract OCR. Επίσης, γίνεται η ρύθμιση των απαραίτητων περιβαλλοντικών μεταβλητών με την εντολή ldconfig.

Η εντολή curl χρησιμοποιείται για τη λήψη του αρχείου eng.traineddata από το αποθετήριο tessdata_best του Tesseract OCR και αποθηκεύεται στον φάκελο /usr/local/share/tessdata.

Εκτελείται η εντολή apt-get update και γίνεται η εγκατάσταση των πακέτων pulseaudio-utils και alsa-utils για την υποστήριξη ήχου στο εικονικό περιβάλλον.

Ορίζονται περιβαλλοντικές μεταβλητές για τη ρύθμιση του PulseAudio server.

Ορίζεται ο φάκελος /run/user/${UID}/pulse ως τον φάκελο όπου θα αποθηκεύονται οι αρχεία ήχου για τον χρήστη.

Ορίζεται η μεταβλητή UID με την τιμή 1000 και η μεταβλητή περιβάλλοντος USER_ID με την ίδια τιμή. Επαναρυθμίζεται ο χρήστης pulseaudio με το UID που ορίστηκε.

Ορίζεται ο χρήστης pulseaudio ως ο χρήστης που θα εκτελεί την εντολή CMD για την εκκίνηση του PulseAudio server.

Εκτελείται η εντολή apt-get update και γίνεται η εγκατάσταση του πακέτου flite για την υποστήριξη της σύνθεσης φωνής.

Αυτός ο Dockerfile δημιουργεί ένα εικονικό περιβάλλον με τις απαραίτητες εξαρτήσεις για την εκτέλεση εφαρμογών που χρησιμοποιούν την OpenCV και το Tesseract OCR. Πραγματοποιείται η εγκατάσταση και η ρύθμιση των απαραίτητων βιβλιοθηκών και εργαλείων, καθώς και η λήψη και η μεταγλώττιση του πηγαίου κώδικα της OpenCV και του Tesseract OCR. Τέλος, εγκαθίσταται η υποστήριξη ήχου με το PulseAudio και η σύνθεση φωνής με το Flite. Αυτό το εικονικό περιβάλλον μπορεί να χρησιμοποιηθεί για την εκτέλεση εφαρμογών που απαιτούν την OpenCV και το Tesseract OCR μέσω ενός περιβάλλοντος Docker.

7. Συμπεράσματα & προτάσεις βελτίωσης

Με την εκπόνηση της εργασίας κατάφερα ώστε άνθρωποι με περιορισμένη όραση να δοκιμάσουν και να διαβάσουν σε ακουστική μορφή, οπτικό υλικο. Είναι μια εφαρμογή πολύ εύκολη στη χρήση της, φορητή και σχετικά φθηνή στην κατασκευή της από τη στιγμή που η χρήση των container του Docker μας δίνει τη δυνατότητα η εφαρμογή να είναι συμβατή με πολλών ειδών development boards.

Τα αποτελέσματα της εφαρμογής δεν είναι ικανοποιητικά λόγω του ότι η κάμερα του πειράματος έχει μικρή ανάλυση και δυσκολεύει τον αλγόριθμο να αναγνωρίσει τα γράμματα σε μια επιφάνεια ενός χαρτιού.

Ένας τρόπος για να βελτιώσουμε την κατασκευή του πειράματος είναι να χρησιμοποιήσουμε μια κάμερα με μεγαλύτερη ανάλυση και την προσθήκη μπαταρίας στη μάσκα, ώστε να γίνει ασύρματη. Τα επόμενα βήματα είναι η εφαρμογή, από offline, να συνδεθεί στο ίντερνετ, να εκπαιδευτεί ο αλγόριθμος της μάσκας ώστε να μπορεί η χρήση της μάσκας να επεκταθεί και σε άλλους τομείς, παραδείγματος χάριν, η μετακίνηση σε διάφορους χώρους με ακουστικό feedback του περίγυρου.

Τέλος, το μέγεθος της μάσκας θα μπορούσε να γίνει μικρότερο και ελαφρύτερο για την βέλτιστη εφαρμογή στο χρήστη.

8. Επίλογος

Το Διαδίκτυο των πραγμάτων είναι μια νέα τάση στην πληροφορική που συνδέει διαφορετικές συσκευές με το Διαδίκτυο, συνήθως ασύρματα, χρησιμοποιώντας Wi-Fi ή Bluetooth. Το σύστημα παρακολούθησης που σχεδιάστηκε εδώ, θα μπορούσε επίσης να χρησιμοποιηθεί αυτόνομα, πράγμα που σημαίνει ότι με τη βοήθεια μιας μπαταρίας ή ενός ηλιακού πλαισίου, θα μπορούσε να τοποθετηθεί

οπουδήποτε και θα λειτουργούσε χωρίς εξωτερικό έλεγχο. Θα μπορούσε να χρησιμοποιηθεί σε διάφορους τομείς, όπως η υγειονομική περίθαλψη, η παρακολούθηση ασθενών σε νοσοκομεία και ο έλεγχος της κατάστασής τους σε πραγματικό χρόνο. Η κίνηση μπορεί να παρακολουθείται, για παράδειγμα, σε χώρους στάθμευσης, επιτρέποντας την πρόσβαση μόνο εάν υπάρχουν κενές θέσεις

Ωστόσο, παρόλο που οι κάμερες μπορούν να παρέχουν σημαντικές ποσότητες πληροφοριών, η συντριπτική πλειονότητα είναι προσωπικά δεδομένα και υπάρχουν σημαντικές ανησυχίες ότι το ατομικό απόρρητο θα μπορούσε να τεθεί σε κίνδυνο. Επιπλέον, δεδομένου ότι οι οικιακές συσκευές συνδέονται όλο και περισσότερο με το Διαδίκτυο μέσω του IoT, κατέστη δυνατή η ακούσια διαρροή εικόνων χρηστών. Έχοντας υπόψη αυτές τις ανησυχίες, υπάρχει ανάγκη να προταθεί μια μέθοδος ανθρώπινης ανίχνευσης που προστατεύει το απόρρητο των χρηστών χρησιμοποιώντας σκόπιμα θολές εικόνες.

9. Βιβλιογραφία

[1] V. Kolias, I. Giannoukos, C. Anagnostopoulos, I. Anagnostopoulos, V. Loumos, & E. Kayafas, "Integrating RFID on event-based hemispheric imaging for internet of things assistive applications," Paper presented at the *ACM International Conference Proceeding Series,* 2010, doi:10.1145/1839294.1839367

[3] L. G. Guo, Y. R. Huang, J. Cai, L. G. Qu, "Investigation of architecture, key technology and application strategy for the internet of things,"

[4] Proceedings of 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference, CSQRWC 2011, 2011, pp. 1196-1199

[5] L., Atzori, A. Iera, G. Morabito, "The Internet of Things: A survey. Computer Networks 54(15), 2010, pp. 2787-2805, doi:10.1016/j.comnet.2010.05.010

[6] P. Dukan, A. Kovari: Cloud-based smart metering system, Proceeding of the 14th IEEE International Symposium on Computational Intelligence and Informatics, Budapest, Hungary, 2013, pp. 499-502

[7] Essa O., (1998), "Using prosody in automatic segmentation of speech", In Proc. 36th ACM Southeast Regional Conference.

[8] Farinas, J., Pellegrino, F., Rouas, J.L., Andre-Obrecht, R.: (2002) Merging Segmental and Rhythm Features for Automatic Language Identification. In: 2002 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 753--756

[9] Hazen, T., Zue, V.: Segment-based Automatic Language Identification. J. of the Acoustic Society of America, (101) 4, pp. 2323--2331 (1997) (available at: http://scitation.aip.org/getpdf/servlet/GetPDFServlet?filetype=pdf&id=JASMAN00010 10000040023230000001&idtype=cvips)

[10] Hirschberg, J., (1993) “Pitch accent in context: predicting intonational prominence from text”, Artificial Intelligence 63, pp. 429-432.

[11] Huang, R., Hansen, J.: (2005) Dialect/Accent Classification via Boosted Word

[12] Modeling. In: 2005 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 585--588 (2005). (available at: http://sail.usc.edu/~georgiou/pdfs/0100585.pdf)

[13] Huang X., Acero A., Hon H-W., (2001) “Spoken Language Processing: A Guide to Theory, Algorithm and System Development,” Pearson Publication. New Jersey.

[14] Klatt, D. H. (1982). “The Klattalk text-to-speech conversion system”, Proceedings on the International Conference on Acoustic, Speech and Signal Processing'82, Paris, pp. 1589--1592.

[17] https://www.docker.com/

[18] https://github.com/tesseract-ocr/tesseract

[19] http://www.leptonica.org/

[20] https://opencv.org/

10. Παράρτημα Α: Κώδικας Ανίχνευσης Κειμένου

```
#include <fstream>
#include <iostream>
#include <string>
#include <chrono>
#include <leptonica/allheaders.h>
#include <tesseract/baseapi.h>
#include <opencv2/opencv.hpp>

#define MAX_THRESHOLD 255

cv::Mat image, src_gray;
int threshold = 100;
cv::RNG rng(12345);

void threshold_callback(int, void* );
```

```
int main(int argc, char **argv)
{
  std::ofstream outfile;
  std::string imagePath = "../images";

  tesseract::TessBaseAPI *api = new tesseract::TessBaseAPI();
  api->Init(NULL, "eng", tesseract::OEM_LSTM_ONLY);
  api->SetPageSegMode(tesseract::PSM_AUTO);
  api->SetVariable("debug_file", "tesseract.log");

  cv::VideoCapture cap(0);

  if (!cap.isOpened())
  {
    std::cerr << "Error opening camera" << std::endl;
    return -1;
  }

  cv::Mat frame;
  cap >> frame;

  if (frame.empty())
  {
    std::cerr << "Error finding frame" << std::endl;
  }

  cv::imwrite("../images/img.jpg", frame);
  cv::cvtColor(frame, src_gray, cv::COLOR_BGR2GRAY);
  cv::blur(src_gray, src_gray, cv::Size(3,3));

  cv::createTrackbar("Canny Threshold:", "SRC", &threshold, MAX_THRESHOLD,
threshold_callback);
  threshold_callback(0, 0);

  auto start = std::chrono::steady_clock::now();
  auto filepath = "../images/img.jpg";
  std::cout << "Detecting text in ../images/img.jpg" << std::endl;
  image = cv::imread(filepath, IMREAD_COLOR);
  api->SetImage(image.data, image.cols, image.rows, 3, image.step);
  std::string outText = api->GetUTF8Text();
  std::cout << outText << std::endl;
  auto end = std::chrono::steady_clock::now();
  std::chrono::duration<double, std::milli> diff = end - start;
  std::cout << "Computation time: " << diff.count() << "ms" << std::endl;
  outfile.open("../output/text.txt");
  outfile << outText;
```

```
  system("flite -f ../output/text.txt");
  api->End();
  return 0;
}

void threshold_callback(int, void* )
{
   cv::Mat canny_output;
   cv::Canny( src_gray, canny_output, threshold, threshold*2 );
   std::vector<std::vector<cv::Point> > contours;
   findContours( canny_output, contours, cv::RETR_TREE, cv::CHAIN_APPROX_SIMPLE );
   std::vector<std::vector<cv::Point> > contours_poly( contours.size() );
   std::vector<cv::Rect> boundRect( contours.size() );
   std::vector<cv::Point2f>centers( contours.size() );
   std::vector<float>radius( contours.size() );
   for( size_t i = 0; i < contours.size(); i++ )
   {
      cv::approxPolyDP( contours[i], contours_poly[i], 10, true );
      boundRect[i] = cv::boundingRect( contours_poly[i] );
      cv::minEnclosingCircle( contours_poly[i], centers[i], radius[i] );
   }
   cv::Mat drawing = cv::Mat::zeros( canny_output.size(), CV_8UC3 );
   for( size_t i = 0; i< contours.size(); i++ )
   {
      cv::Scalar color = cv::Scalar( rng.uniform(0, 256), rng.uniform(0,256), rng.uniform(0,256) );
      drawContours( drawing, contours_poly, (int)i, color );
      rectangle( drawing, boundRect[i].tl(), boundRect[i].br(), color, 2 );
   }
   cv::imwrite("../contours/img_contours.jpg", drawing);
}
```

11. Παράρτημα Β: CMake files

```
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)

set(CMAKE_CXX_COMPILER /usr/bin/g++-8)

project(cav)

find_package(OpenCV REQUIRED)
find_package(PkgConfig REQUIRED)
pkg_search_module(TESSERACT REQUIRED tesseract)
pkg_search_module(LEPTONICA REQUIRED lept)

include_directories(${OpenCV_INCLUDE_DIRS})
```

```
include_directories(${TESSERACT_INCLUDE_DIRS})
include_directories(${LEPTONICA_INCLUDE_DIRS})

add_executable(main main.cpp)

target_link_libraries(main ${OpenCV_LIBS})
target_link_libraries(main ${TESSERACT_LIBRARIES})
target_link_libraries(main ${LEPTONICA_LIBRARIES})

target_link_libraries(main stdc++fs)

set_property(TARGET main PROPERTY CXX_STANDARD 17)
```

12. Παράρτημα Γ: Dockerfile

```
FROM ubuntu:18.04

WORKDIR /home/pi

RUN apt-get update && apt-get install -y build-essential && \
   apt-get install -y cmake vim git libgtk2.0-dev pkg-config \
   libavcodec-dev libavformat-dev libswscale-dev && \
   apt-get install -y python-dev python-numpy libtbb2 \
   libtbb-dev libjpeg-dev libpng-dev libtiff-dev libdc1394-22-dev \
   && apt-get clean && rm -rf /var/lib/apt-lists/*

RUN git clone https://github.com/opencv/opencv.git
RUN git clone https://github.com/opencv/opencv_contrib.git

RUN cd opencv && mkdir -p build && cd build && \
   cmake -D CMAKE_BUILD_TYPE=Release -D \
         CMAKE_INSTALL_PREFIX=/usr/local -D \
         OPENCV_GENERATE_PKGCONFIG=ON -D \
         OPENCV_EXTRA_MODULES_PATH=/home/pi/opencv_contrib/modules .. \
   && make -j4 && make install

RUN apt-get install -y automake ca-certificates g++-8 \
   git libtool liblеptonica-dev make pkg-config && \
   apt-get install -y --no-install-recommends asciidoc \
   docbook-xsl xsltproc && apt-get install -y libpango1.0-dev \
   && apt-get install -y libicu-dev libpango1.0-dev libcairo2-dev

RUN git clone https://github.com/tesseract-ocr/tesseract.git
```

```
RUN cd tesseract && ./autogen.sh && ./configure \
   && make && make install && make training && \
   make training-install && ldconfig

#RUN mkdir /usr/local/share/tessdata

RUN curl -o /usr/local/share/tessdata/eng.traineddata \
   https://raw.githubusercontent.com/tesseract-ocr/tessdata_best/master/eng.traineddata

RUN apt-get update && \
   apt-get install -y pulseaudio-utils alsa-utils && \
   rm -rf /var/lib/apt/lists/*

ENV PULSE_SERVER=unix:${XDG_RUNTIME_DIR}/pulse/native
ENV PULSE_COOKIE=/run/pulse/cookie

VOLUME /run/user/${UID}/pulse

ARG UID=1000
ENV USER_ID=${UID}
RUN usermod -u ${USER_ID} pulseaudio

USER pulseaudio
CMD pulseaudio --system --disallow-exit --verbose

RUN apt-get update && apt-get install -y flite
```

13. Παράρτημα Δ: 3D Design files