Πανεπιστήμιο Πελοποννήσου

Σχολή Οικονομίας και Τεχνολογίας

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Π.Μ.Σ. στην Επιστήμη Υπολογιστών

Διπλωματική Εργασία

«3D Hand Animation using Python and Blender»

Μακρής Βασίλειος

Α.Μ.: 2022202102008 Email: dit2108cst@go.uop.gr

Υπεύθυνος Καθηγητής: κ. Νικόλαος Τσελίκας

Τρίπολη, Μάρτιος 2023

1. Περιεχόμενα

Ευχαριστίες

Με την ολοκλήρωση της παρούσας εργασίας θα ήθελα να ευχαριστήσω θερμά τον καθηγητή κ. **Νικόλαο Τσελίκα**, για την υπομονή, τη δεκτικότητα και την αμέριστη αρωγή του. Επίσης, θα ήθελα να ευχαριστήσω όλους τους συναδέλφους αλλά και καθηγητές μου στο μεταπτυχιακό πρόγραμμα, που μου έδωσαν αμέριστα τη βοήθειά τους για να υλοποιηθεί η παραπάνω διπλωματική εργασία. Επιπλέον, δεν μπορώ να παραλείψω να ευχαριστήσω την οικογένειά μου, καθώς χωρίς τη δική της στήριξη και βοήθεια δεν θα μπορούσα να επιτύχω τον σκοπό μου .

Αφιερώνεται, στην μητέρα και στον πατέρα μου…Ευχαριστώ

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με τη σχεδίαση και ανάπτυξη ενός συστήματος 3D απεικόνισης του ανθρώπινου χεριού. Με βάση αυτό θα εφαρμόσουμε αλγόριθμο που να επιτρέπει τη ζωντανή αναπαράσταση τον κινήσεων του χεριού και τη μεταφορά των δεδομένων του σε 3D αναπαραγωγή με τη χρήση προγράμματος (Blender) σε πραγματικό χρόνο. Η εφαρμογή δημιουργήθηκε σε γλώσσα προγραμματισμού Python σε συνδυασμό με το πρόγραμμα Blender με τίτλο **" 3D Hand Animation using Python and Blender** ". Η τεχνολογία αυτή περιλαμβάνει τη δημιουργία ενός ψηφιακού μοντέλου του χεριού ενός ατόμου με τη χρήση του λογισμικού και απλού υλικού (κάμερες). Στην συνέχεια ακολουθεί η επιλογή 21 σημείων από το χέρι που θα τα ονομάσουμε ορόσημα (landmarks) με ταυτόχρονη εξαγωγή των συντεταγμένων τους (x, y,z) από την τοποθέτηση, στην εικόνα, του χεριού. Στη συνέχεια οι τιμές αυτές σε byte αποστέλλονται στο πρόγραμμα Blender που μετατρέπονται σε νέες συντεταγμένες (float numbers), εκπροσωπώντας τις θέσεις των σημείων, με τη βοήθεια πάλι της γλώσσας Python γίνεται 3D απεικόνιση του χεριού. Τα αποτελέσματα που παίρνουμε από τη ζωντανή αναπαραγωγή εικόνων άνω ακρών έχουν ως στόχο, τη γρήγορη και εύκολη πρόσβαση σε γιατρό για διάγνωση, συντονισμό της φροντίδας, απομακρυσμένη παρακολούθηση και επαγγελματική υγεία. Η προσπάθειά μας είναι η αποτύπωση της χρήσης του αλγορίθμου σε περαιτέρω τομείς της σύγχρονης εποχής μας. Η τρισδιάστατη (3D) αναπαράσταση χεριών είναι μια χρήσιμη τεχνική που μπορεί να βοηθήσει σε ένα ευρύ φάσμα εφαρμογών, όπως ιατρικές διαδικασίες, παιχνίδια, ρομποτική και εικονική πραγματικότητα. Στο 1ο κεφάλαιο θα αναλύσουμε τη φυσιολογία του χεριού. Στο 2ο κεφάλαιο γίνεται μια αναφορά στις τεχνολογίες ανάπτυξης. Το 3ο κεφάλαιο περιλαμβάνει τον σχεδιασμό και τον τρόπο μοντελοποίησης. Στο 4ο κεφάλαιο αναφέρονται όλα τα μέρη της υλοποίησης με περιγραφές κώδικα ενώ στο 5ο κεφάλαιο αναφέρονται πιθανές μελλοντικές επεκτάσεις της εφαρμογής. Τέλος, στο 6ο κεφάλαιο γίνεται αναφορά στα συμπεράσματα που προκύπτουν από τη χρήσης της και στο 7ο κεφάλαιο παρουσιάζεται η συνολική βιβλιογραφία, που περιλαμβάνει και τους διαδικτυακούς ιστότοπους, που βοήθησαν στην ολοκλήρωση της παρούσας εργασίας.

Λέξεις κλειδιά

Ανάπτυξη εφαρμογών σε Python, Python , Blender, UDP server , Τεχνητή νοημοσύνη, Νευρωνικά Δίκτυα, Οπτική αναγνώριση χεριών, Αναγνώριση χειρονομιών, Επεξεργασία εικόνας , Ανίχνευση αντικειμένων,landmarks

Abstract

This thesis deals with the design and development of a 3D imaging system of the human hand. Based on this, we will implement an algorithm that allows the live representation of the hand movements and the transfer of the data to 3D reproduction using a program (Blender) in real time. The application was created in Python programming language in combination with the Blender program entitled "3D Hand Animation using Python and Blender ". This technology involves creating a digital model of a person's hand using software and simple hardware (cameras). Then follows the selection of 21 points from the hand that we will call landmarks with simultaneous extraction of their coordinates (x, y, z) from the positioning of the hand on the image. These byte values are then sent to the Blender program where they are converted to new coordinates (float numbers), representing the positions of the points, again with the help of the Python language, a 3D representation of the hand is made. The results we get from the live playback of upper limb images are aimed at providing quick and easy access to a doctor for diagnosis, care coordination, remote monitoring and professional health care. Our effort is to illustrate the use of the algorithm in further areas of our modern era. Three-dimensional (3D) hand representation is a useful technique that can help in a wide range of applications, including medical procedures, games, robotics and virtual reality. In chapter 1 we will analyze the physiology of the hand. In chapter 2, a reference is made to development technologies. Chapter 3 includes the design and modelling. In chapter 4 all parts of the implementation are listed with code descriptions while in chapter 5 possible future extensions of the application are mentioned. Finally, in chapter 6, reference is made to the conclusions that arise from its use and in chapter 7, the overall bibliography is presented, including the online websites that helped in the completion of this thesis.

Key words

Application development in Python, Python , Blender, UDP server , Artificial Intelligence, Neural Networks, Visual Hand Recognition, Gesture Recognition, Image Processing, Object Detection, Landmarks.

1. Εισαγωγή

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η δημιουργία μιας 3D απεικόνισης του χεριού με στόχο τη συγκομιδή δεδομένων κυρίως για ιατρικούς σκοπούς (διάγνωση από απόσταση, φυσιοθεραπεία) αλλά και γενικότερα τη βελτίωση και αξιολόγηση των διεπαφών ανθρώπου-υπολογιστή, ώστε να δημιουργηθεί μια πιο φυσική, διαισθητική επικοινωνία μεταξύ των ανθρώπων και όλων των ειδών των συσκευών που βασίζονται σε αισθητήρες, μοιάζοντας έτσι περισσότερο με την επικοινωνία ανθρώπου-ανθρώπου. Με βάση αυτό, προσπαθήσαμε να εφαρμόσουμε αλγόριθμο που να επιτρέπει την απομακρυσμένη διάγνωση, μελετώντας αποτελέσματα που παίρνουμε από τη ζωντανή αναπαραγωγή εικόνων άνω ακρών με στόχο τη γρήγορη και εύκολη πρόσβαση σε γιατρό για διάγνωση, συντονισμό της φροντίδας, απομακρυσμένη παρακολούθηση και επαγγελματική υγεία, αλλά και μία προσπάθεια να αποτυπωθεί η χρήση του αλγορίθμου σε περαιτέρω τομείς της σύγχρονης εποχής μας.

1.1 **«Μάθημα» Ανατομίας**

Το ανθρώπινο χέρι, το πιο απομακρυσμένο τμήμα του άνω άκρου, είναι ένα αξιοσημείωτο επίτευγμα μηχανικής και εξέλιξης. Είναι αρκετά ισχυρό ώστε να επιτρέπει στους ορειβάτες να αντιμετωπίσουν οποιοδήποτε βουνό, αλλά και αρκετά ακριβές για τον χειρισμό μερικών από τα μικρότερα αντικείμενα στον κόσμο και την εκτέλεση πολύπλοκων ενεργειών. Το ίδιο το χέρι αποτελείται από συγκεκριμένα οστά στα οποία προσκολλώνται διάφοροι μύες, καθώς και από μια συλλογή νεύρο-αγγειακών δομών που είναι υπεύθυνες για την αποχέτευση της πληροφορίας και τη νεύρωση. Ωστόσο, οι εγγενείς μύες του χεριού είναι μόνο εν μέρει υπεύθυνοι για όλο το εύρος της κίνησής του. Οι άλλοι σημαντικοί συντελεστές είναι στην πραγματικότητα οι μύες του αντιβραχίου, οι οποίοι προβάλλουν τένοντες προς το χέρι μέσω μιας εξίσου πολύπλοκης και ευέλικτης ανατομικής δομής, που ονομάζεται καρπός.

Η απόκτηση των δεξιοτήτων χειρισμού ενός εργαλείου πιστεύεται ότι είναι η κύρια δύναμη πίσω από την εξέλιξη του χεριού. Η αρχέγονη «λαβή ρίψης» και η «λαβή ροπάλου ήταν αναπαραγωγικά πλεονεκτήματα (όσο ισχυρότερη η λαβή, τόσο καλύτερος ο κυνηγός και καλύτερες οι πιθανότητες επιβίωσης). Τα ανατομικά χαρακτηριστικά, που κάνουν δυνατό το βελτιωμένο

δραγμό[1] αναπτύχθηκαν σε εκατομμύρια χρόνια και κατέληξαν στην εκπληκτική λειτουργική ικανότητα του ανθρώπινου χεριού. Μια δυσανάλογη περιοχή του εγκεφαλικού φλοιού είναι αφοσιωμένη στον έλεγχο των 40 περίπου μυών που είναι απαραίτητοι για να θέσουν σε λειτουργία το χέρι. Πιστεύεται σήμερα ότι η πολυπλοκότητα του ελέγχου στο επίπεδο του πρωτογενούς κινητικού φλοιού, ελαττώνεται σε ένα σύνολο νευρώνων με επιθυμητές κατευθύνσεις, οι οποίες απλώς καθορίζουν την κατεύθυνση της κίνησης του χεριού. Ο έλεγχος του χεριού και άλλες στερεοτυπικές κινήσεις βασίζονται σε διακλαδιζόμενες ,κατερχόμενες προβολές και σπονδυλικά κυκλώματα που ορίζουν μια υποκείμενη βασική ομάδα μυϊκών συνεργειών. Αυτές οι συνέργειες μπορεί να συνδυαστούν και δυνητικά να απλοποιήσουν τον έλεγχο ενός εύρους πολύπλοκων κινήσεων.

Βασικοί χειρισμοί που εκτελούνται από το χέρι: **Εικόνα 1-2**:

- Σύλληψη (τελική) ακριβείας (π.χ. σηκώνοντας μια βελόνη)
- Αντιθετική σύλληψη (π.χ. κρατώντας ένα χαρτί)
- Σύλληψη κλειδιού
- Κατευθυνόμενη λαβή (π.χ. χρησιμοποιώντας ένα κατσαβίδι)
- Λαβή δίκην γάντζου (π.χ. σηκώνοντας μια βαλίτσα)
- Ισχυρός δραγμός[2] (π.χ. κρατώντας ένα ρόπαλο)
- Δραγμός[3] δίκην σπιθαμής (π.χ. κρατώντας μια μπάλα)

Ο αντίχειρας συνεισφέρει στο 40% περίπου της λειτουργικότητας του χεριού και είναι απαραίτητος για τον ακριβή και ισχυρό δραγμό. Είναι επίσης προικισμένος με τη μοναδική δυνατότητα για περιαγωγή και αντίθεση, μέσω της κίνησής του έξω από το επίπεδο του χεριού. Ο συνδυασμός των αυτοχθόνων και των ετεροχθόνων κινητικών μονάδων διευκολύνει τη λεπτή

[1] Δραγμός =Δράττομαι, Δράση
[2] Δραγμός =Δράττομαι,Δράση

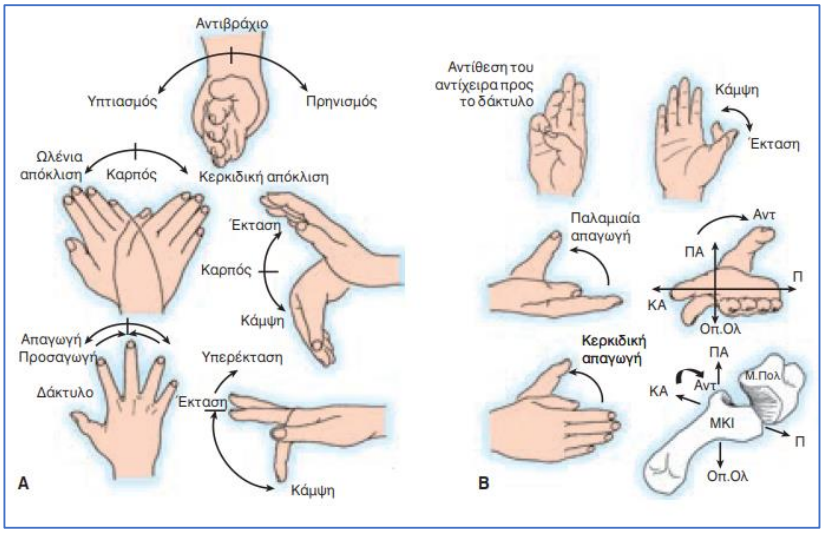
κινητική δεξιότητα του αντίχειρα, του χεριού και του καρπού. Οι τένοντες με τους μύες που εκφύονται κεντρικά του καρπού (π.χ. ο μακρύς απαγωγός του αντίχειρα) αναφέρονται ως ετερόχθονες και οι κινητικές μονάδες με τους μύες που εκφύονται μέσα από το χέρι (π.χ. βραχύς απαγωγός του αντίχειρα) αναφέρονται ως αυτόχθονες μύες. Ο καρπιαίος σωλήνας και το κανάλι του Guyon είναι δυο οριοθετημένα περάσματα στην παλαμιαία πλευρά του καρπού. Ο καρπιαίος σωλήνας είναι το μεγαλύτερο από τα δυο και περιέχει εννέα ετερόχθονες καμπτήρες τένοντες του αντίχειρα και των υπόλοιπων δακτύλων, συνοδευόμενους από το μέσο νεύρο· το κανάλι του Guyon βρίσκεται δίπλα στον καρπιαίο σωλήνα και περιέχει το ωλένιο νεύρο και την ωλένια αρτηρία. Τα ραχιαία διαμερίσματα των εκτεινόντων τενόντων σχηματίζονται κάτω από τον εγκάρσιο καθεκτικό σύνδεσμο των εκτεινόντων. Οι τένοντες διασχίζουν τον καρπό μέσα από έξι διακριτά διαμερίσματα, τα οποία εμποδίζουν τη δημιουργία του φαινομένου της χορδής τόξου (bowstringing) κατά την έκταση του καρπού. Σχηματίζουν, έτσι αξιόπιστα ανατομικά τοπογραφικά σημεία. Υπάρχουν τρεις φάλαγγες στα δάκτυλα και δύο στον αντίχειρα. Ο αντίχειρας παράγει το μεγαλύτερο τμήμα του εύρους της κίνησης του στην πολύγωνο-μετακάρπια (ΠΜΚ) άρθρωση. Το κερκιδικό, το ωλένιο και το μέσο νεύρο, μαζί με μια συνεισφορά του μύο-δερματικού νεύρου, παρέχουν αισθητική και κινητική νεύρωση περιφερικά του αγκώνα. Η αισθητική νεύρωση στην παλαμιαία επιφάνεια του χεριού γίνεται με το μέσο και το ωλένιο νεύρο· το κερκιδικό και το ωλένιο νεύρο παρέχουν αισθητικότητα στο μεγαλύτερο μέρος της ραχιαίας επιφάνειας του χεριού. Το ωλένιο νεύρο νευρώνει τους περισσότερους αυτόχθονες μύες, παρόλο που το μέσο νεύρο ελέγχει την αντίθεση του αντίχειρα νευρώνοντας τους περισσότερους μύες του θέναρος. Το κερκιδικό νεύρο και ο οπίσθιος μεσόστεος κλάδος του (ΟΜΝ) νευρώνουν τους ετερόχθονες εκτείνοντες τένοντες του αντίχειρα, των δακτύλων και του καρπού. Το δέρμα της παλάμης είναι άτριχο και σταθερά καθηλωμένο στην παλαμιαία απονεύρωση για τη διευκόλυνση του δραγμού. Τα δύο επάρματα στην παλάμη είναι το θέναρ και το υποθέναρ. **Εικόνα 1-3**

Ξεκινώντας, θα παρουσιάσουμε την ανατομία των άνω άκρων και συγκεκριμένα του χεριού. Το χέρι αποτελείται από πολλά διαφορετικά οστά, τους μύες και τους συνδέσμους που επιτρέπουν για ένα μεγάλο ποσό της κίνησης και αλλά και επιδεξιότητα. **Εικόνα 1-4**

Εικόνα 1-1: Επιφανειακή ανατομία της παλάμης

Εικόνα 1-2:Η οστική ανατομία του χεριού

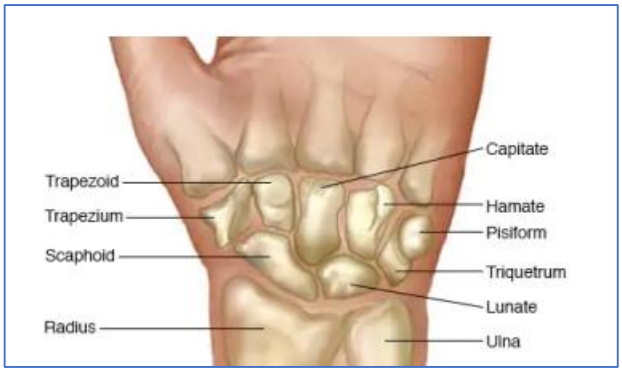
Εικόνα 1-3: Ανατομία του χεριού



Εικόνα 1-4: Α. Κοινή ορολογία για το εύρος κίνησης του χεριού. Β. Οι αμφίκοιλες επιφάνειες της πολύγωνο μετακάρπια άρθρωσης του αντίχειρα: περιστροφή, κάμψη/έκταση , απαγωγή/προσαγωγή

Πιο αναλυτικά: Το **χέρι** αποτελείται από πολλά μικρά οστά:

• Τα οστάρια του καρπού



Εικόνα 1-5: Τα οστάρια του καρπού

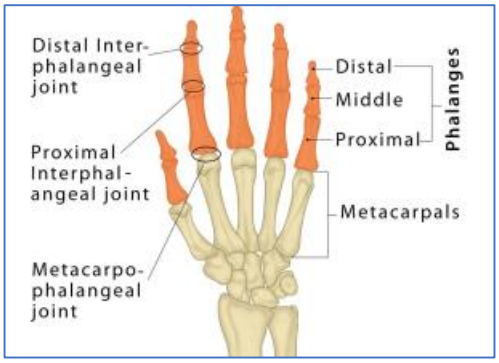
Τα καρπιαία οστά (δηλ. καρπός) είναι **οκτώ οστά** ακανόνιστου σχήματος που βρίσκονται στην περιοχή του καρπού. Τα οστά αυτά συνδέουν τις άπω πλευρές των μακρών οστών του αντιβραχίου (κερκίδα και ωλένη) με τις εγγύς πλευρές των μετά-καρπικών οστών και προσφέρουν επιδεξιότητα και μεγάλο εύρος κίνησης στο χέρι. Τα καρπιαία οστά είναι οργανωμένα σε δύο σειρές: εγγύς και άπω. Η εγγύς σειρά των καρπιαίων οστών (από το κερκιδικό προς το ωλένιο) περιλαμβάνει το σκαφοειδές, το σεληνιακό, το τρίκετρο και το ψιλοειδές οστό. Η άπω σειρά περιλαμβάνει το τραπέζιο, το τραπεζοειδές, το κιονόκρανο και το αγκιστροειδές οστό. Κάθε καρπιαίο οστό έχει το δικό του μοναδικό σχήμα και είναι πολύπλευρο, πράγμα που σημαίνει ότι έχει την ικανότητα να αρθρώνεται με διάφορα γύρω οστά, μύες και συνδέσμους του αντιβραχίου και του χεριού. Με αυτόν τον τρόπο, τα καρπιαία οστά παρέχουν ευελιξία και διάφορους τύπους κινήσεων στους μαλακούς ιστούς του χεριού. Παρέχουν επίσης το μεγαλύτερο μέρος του σκελετικού πλαισίου του καρπού που επιτρέπει τη διέλευση των διαφόρων νεύρο-αγγειακών δομών του χεριού. **Εικόνα 1-5**

• Τα μετακάρπια

Εικόνα 1-6: Τα μετακάρπια

Το μετακάρπιο είναι μια ομάδα πέντε οστών του χεριού μεταξύ των φαλάγγων και του καρπού. Παρόλο που τα μετακάρπια οστά είναι μικρά, ταξινομούνται ως μακρά οστά, καθώς έχουν δομικά χαρακτηριστικά μακρών οστών- κάθε μετακάρπιο οστό αποτελείται από έναν άξονα, μια άπω κεφαλή και μια ευρεία εγγύς βάση Τα μετακάρπια οστά αρθρώνονται με τα καρπιαία οστά μέσω των εγγύς άκρων τους (βάσεις) και με τις εγγύς φάλαγγες μέσω των άπω άκρων τους (κεφαλές). **Εικόνα 1-6**

• Τις φάλαγγες των δακτύλων.



Εικόνα 1-7: Οι φάλαγγες των δακτύλων

Οι φάλαγγες του χεριού είναι η ομάδα μικρών οστών που αποτελούν τον οστέινο πυρήνα των δακτύλων του χεριού. Παρόλο που οι φάλαγγες είναι μικρές σε μέγεθος, ταξινομούνται ως μακρά οστά λόγω των δομικών τους χαρακτηριστικών- κάθε φάλαγγα αποτελείται από έναν άξονα, μια άπω κεφαλή και μια εγγύς βάση. Υπάρχουν δεκατέσσερις φάλαγγες σε κάθε χέρι- κάθε ένα από τα μεσαία τέσσερα δάκτυλα έχει τρεις φάλαγγες (εγγύς, μέση και άπω), ενώ ο αντίχειρας έχει μόνο δύο (εγγύς και άπω). Οι φάλαγγες συνδέονται μεταξύ τους με μεσοφαλαγγικές αρθρώσεις και αγγειοποιούνται μέσω των θρεπτικών ραγών προς τις φάλαγγες, οι οποίες προέρχονται από τις παλαμιαίες ψηφιακές αρτηρίες.

Τα δάκτυλα έχουν ένα καθολικό σύστημα επισήμανσης που χρησιμοποιεί την ανατομική θέση του χεριού (παλάμη προς τα εμπρός) ως αναφορά. Πηγαίνοντας από πλάγια προς τα μέσα, ονομάζονται αντίχειρας (ψηφίο 1), δείκτης (ψηφίο 2), μέσος δάκτυλος (ψηφίο 3), παράμεσος (ψηφίο 4) και μικρό δάκτυλο (ψηφίο 5). **Εικόνα 1-7**

1.2 **Οπτική αναγνώριση χεριού.**

Εμείς οι άνθρωποι χρησιμοποιούμε χειρονομίες για να αλληλοεπιδρούμε με το περιβάλλον μας κατά τα πρώτα στάδια της ανάπτυξής μας. Επικοινωνούμε επίσης με χειρονομίες όπως η κίνηση του σώματος, η έκφραση του προσώπου και η υπόδειξη με τα δάχτυλα. Το ανθρώπινο χέρι διαδραματίζει πολύ σημαντικό ρόλο στην επικοινωνία όταν οι άνθρωποι αλληλοεπιδρούν ο ένας με τον άλλον και με το περιβάλλον τους στην καθημερινή ζωή. Η αναγνώριση χειρονομιών και ανθρώπινων δραστηριοτήτων συνδέεται στενά με τις τοποθεσίες και τις κατευθύνσεις των ανθρώπινων χεριών.

Η αναγνώριση εικόνας γίνεται ένα κρίσιμο βήμα στα περισσότερα από τα σύγχρονα συστήματα επίλυσης προβλημάτων στον κόσμο. Η δυναμική αναγνώριση του σχήματος του χεριού, παραμένει μια πρόκληση. Η αναγνώριση χειρονομιών και ανθρώπινων δραστηριοτήτων συνδέεται στενά με τις τοποθεσίες και τις κατευθύνσεις των ανθρώπινων χεριών. Ως εκ τούτου, η ανίχνευση του ανθρώπινου χεριού, αξιόπιστα, από έγχρωμες εικόνες ή βίντεο που καταγράφονται από κοινούς αισθητήρες εικόνας διαδραματίζει σημαντικό ρόλο σε πολλές εφαρμογές που σχετίζονται με την όραση του υπολογιστή, όπως η αλληλεπίδραση ανθρώπου-υπολογιστή η εκτίμηση της ανθρώπινης στάσης , η αναγνώριση ανθρώπινων χειρονομιών και η ανάλυση ανθρώπινης δραστηριότητας .

Χτυπάμε τα δάχτυλά μας και η καφετιέρα και θα ετοιμάσει ένα φρέσκο φλιτζάνι καφέ. Κουνάμε το χέρι μας κοντά στην έξυπνη τηλεόρασή (Smart TV) και ενεργοποιείται η πρόγνωση του καιρού. Πατάμε ένα δάχτυλο κοντά στο έξυπνο ρολόι μας και ρυθμίζουμε το ξυπνητήρι στο υπνοδωμάτιο. Πόσο υπέροχο θα ήταν να μπορούμε να κάνουμε πράγματα μόνο με χειρονομίες ;.Δεν είναι πια τόσο εξωπραγματικό, αφού με τις νέες τεχνολογίες εντοπισμού χεριών και αναγνώρισης χειρονομιών μπορούμε να διεισδύσουμε σε πολλούς κλάδους της σύγχρονης ζωής μας. Η χειρονομία τελικά είναι ένας φυσικός και διαισθητικός τρόπος αλληλεπίδρασης με τους ανθρώπους και το περιβάλλον. Είναι λοιπόν απολύτως λογικό να χρησιμοποιούνται οι χειρονομίες ως μέθοδος αλληλεπίδρασης ανθρώπου-μηχανής. Υπάρχουν όμως αρκετές προκλήσεις, ξεκινώντας από την ανάγκη να κουνάμε τα χέρια μας μπροστά από τη μικρή οθόνη του smartphone και καταλήγοντας στους πολύπλοκους αλγορίθμους μηχανικής μάθησης.

Όμως πρέπει να λάβουμε υπόψη ότι η παρακολούθηση χεριών και η αναγνώριση χειρονομιών δεν είναι το ίδιο πράγμα. Και οι δύο τεχνολογίες υποτίθεται ότι χρησιμοποιούν τα χέρια για την αλληλεπίδραση ανθρώπου-μηχανής χωρίς να αγγίζουν, να αλλάζουν ή να χρησιμοποιούν χειριστήρια. Μερικές φορές, τα συστήματα για την παρακολούθηση χεριών και την αναγνώριση χειρονομιών απαιτούν τη χρήση μαρκαδόρων, γαντιών ή αισθητήρων, αλλά ένα ιδανικό σύστημα δεν απαιτεί τίποτε άλλο παρά ένα ανθρώπινο χέρι. Τα συστήματα που χρησιμοποιούν τεχνολογία αναγνώρισης χειρονομιών είναι ικανά να διακρίνουν μόνο συγκεκριμένες χειρονομίες: αντίχειρες προς τα πάνω, κύμα, σήμα ειρήνης, σύμβολο βράχου κ.λπ. Η παρακολούθηση χεριών είναι πιο σύνθετη: παρέχει μεγαλύτερη μεταβλητότητα, καθώς παρακολουθεί το μέγεθος του χεριού, τη

θέση των δακτύλων και άλλα χαρακτηριστικά. Ο αριθμός των πιθανών αλληλεπιδράσεων με τα ψηφιακά αντικείμενα είναι απεριόριστος, αλλά εμφανίζονται προβλήματα επικάλυψης.

Στο πεδίο όρασης του υπολογιστή, ο αγωγός εφαρμογών που σχετίζονται με το χέρι περιέχει συνήθως τρία βήματα:

- Ανίχνευση χεριών,
- Εκτίμηση πόζας χεριών
- Στατική αναγνώριση χειρονομίας ή δυναμική αναγνώριση δράσης.

Το δεύτερο βήμα είναι προαιρετικό, επειδή η αναγνώριση χειρονομίας/δράσης μπορεί να πραγματοποιηθεί με ή χωρίς εκτίμηση πόζας χεριού. Η εκτίμηση του χεριών και η αναγνώριση της δράσης είναι τα πιο δύσκολα βήματα σε περιορισμένο περιβάλλον (συνήθως μόνο ένα χέρι και απλό φόντο σε μια εικόνα) από τα οποία το χέρι μπορεί εύκολα να ανιχνευθεί.

Οι μέθοδοι ανίχνευσης χεριών χρησιμοποιούν κυρίως χαρακτηριστικά εικόνας χαμηλού επιπέδου, όπως το χρώμα του δέρματος και το σχήμα για την ανίχνευση περιοχής χεριών. Σήμερα, οι περίπλοκες προσεγγίσεις ανίχνευσης που βασίζονται στα νευρωνικά δίκτυα αποδεικνύονται πιο ισχυρές και ακριβείς λόγω των διακριτών και βαθιών χαρακτηριστικών που έχουν μάθει. Ωστόσο, σε σύγκριση με τα κοινά αντικείμενα, τα ανθρώπινα χέρια είναι ιδιαίτερα αρθρωμένα, εμφανίζονται σε διάφορους προσανατολισμούς, κλίμακες, σχήματα, και χρώματα δέρματος.

Η προσέγγιση που χρησιμοποιήθηκε στην προσπάθεια αυτής της εργασίας στη ανίχνευση αλλά και απόδοση των δεδομένων και των αναλύσεων προέρχονται από βίντεο πραγματικού χρόνου (Real Time Video). Ως αποτέλεσμα τα δύο βασικά επίπεδα της διαδικασίας :

• Παρακολούθηση. Μια συσκευή (κάμερα) παρακολουθεί το πλαίσιο των κινήσεων για να συλλάβει κάθε κίνηση και να παρέχει ακριβή είσοδο για την ανάλυση δεδομένων.

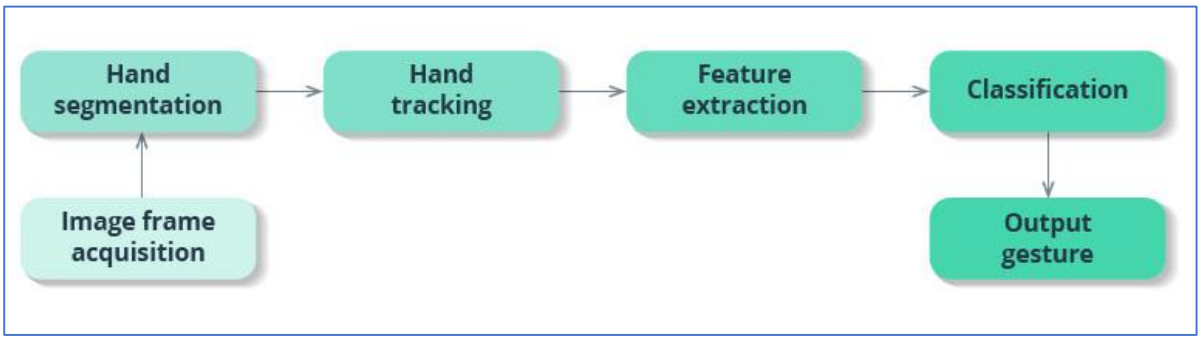
• Ανίχνευση. Με τη βοήθεια της κάμερας ο αλγόριθμος χρησιμοποιώντας τμήματα της εικόνας για βρίσκει τις προεπιλεγμένες θέσεις , ορόσημα (Landmarks) .

• Αποθήκευση . Ο αλγόριθμος με βάση τα δεδομένα που συγκεντρώθηκαν εξάγει και ταξινομεί όλα τα χαρακτηριστικά.

Το σύστημα διακρίνει το χέρι από το φόντο χρησιμοποιώντας δεδομένα χρώματος και βάθους. Εστιάζοντας περαιτέρω τον καρπό, την παλάμη και τα δάχτυλα. Στη συνέχεια, το σύστημα λαμβάνει πληροφορίες σχετικά με την απόσταση από τα δάχτυλα στο κέντρο της παλάμης, την ανύψωση των δακτύλων, το σχήμα της παλάμης, τη θέση των δακτύλων, με σημείο αναφοράς των μετρήσεων και απεικόνισης των δεδομένων το σημείο 0 (Wrist) () .

Η προσέγγιση για να αναγνωρίσουμε κάθε δάχτυλο στηρίζεται στην εκμετάλλευση των γεωμετρικών ιδιοτήτων των δαχτύλων όπως αυτά εμφανίζονται στις εικόνες του χεριού (Landmarks). Κοιτάζοντας από το κέντρο του χεριού (Landmark 0), τα δάχτυλα θα είναι σχετικά μακριά και λεπτά σε σύγκριση με το αντιβράχιο και τις πλευρές του χεριού. Αυτή η ιδιότητα μπορεί να αξιοποιηθεί για την εύρεση των δακτύλων εξετάζοντας τις αποστάσεις από το κέντρο της παλάμης έως το όριο του αντικειμένου. Σε ένα χέρι με έως και πέντε διαφορετικά δάχτυλα, το όριο αυτό κατά μήκος της πλευράς ενός δακτύλου πρέπει να αυξάνεται σταθερά μέχρι να επιτευχθεί ένα μέγιστο γύρω από το άκρο του δακτύλου. Κάθε άκρο δακτύλου θα είναι ένα τοπικό μέγιστο σε ένα γράφημα που περιέχει αποστάσεις από κάθε ένα από τα οριακά εικονοστοιχεία (pixels) και έτσι τα δάκτυλα μπορούν να ανιχνευθούν επιλέγοντας κάθε τοπικό μέγιστο σε μια γωνιακή απόστασης με αφετηρία το κέντρο της παλάμης. Η τυπική θέση του χεριού (ανίχνευση αρθρώσεων) και η θέση των εκτεταμένων δακτύλων σε σχέση με το χέρι, που είναι η πιο χαρακτηριστική ιδιότητα μεταξύ των διαφορετικών στάσεων του χεριού, θα πρέπει να αποτελεί την βάση του αλγορίθμου. Ωστόσο, ο προσανατολισμός των δακτύλων επιτρέπει στη θέση διαφορετικών δακτύλων να καταλαμβάνουν τον ίδιο χώρο και έτσι προκαλείται επικάλυψη μεταξύ των διαφορετικών στάσεων του χεριού. Για να διασφαλιστεί ότι ο αλγόριθμος ταξινόμησης είναι αναλλοίωτος στην περιστροφή, δεν πρέπει να αντιμετωπίσει μόνο τον προσανατολισμό του χεριού, αλλά και τον προσανατολισμό κάθε δακτύλου. Δεδομένου ότι το άκρο του δακτύλου είναι το πιο έντονο μέρος του δακτύλου, το οποίο μπορεί να βρεθεί αξιόπιστα αλλά και ταυτόχρονα να είναι ανθεκτικό στις διακυμάνσεις της τμηματοποίησης, χρησιμοποιείται ως το κύριο σημείο αναφοράς για την εξαγωγή χαρακτηριστικών.

Η διαδικασία σχηματικά παρουσιάζεται στην **Εικόνα 1-8**



Εικόνα 1-8: Σχηματική αναπαράσταση της διαδικασίας

Η παραπάνω διαδικασία με χρήση κάμερας (ανίχνευση) σε πραγματικό χρόνο, ο εντοπισμός και η αναγνώριση γίνεται με την βοήθεια του αλγορίθμου που χρησιμοποιώντας τα παραπάνω σημεία δημιουργεί τις προϋποθέσεις για την αξιοποίηση των δεδομένων που εξάγονται και αποθηκεύονται σε αρχείο μορφής TEXT , Csv, ή και Excel.

2. Τεχνολογικά Περιβάλλοντα

Στο κεφάλαιο αυτό θα γίνει μια περιγραφή των εργαλείων που χρησιμοποιήθηκαν για την υλοποίηση του αλγορίθμου, καθώς και οι λόγοι οδήγησαν στην επιλογή αυτών των συγκεκριμένων τεχνολογικών εργαλείων.

2.1 Python –Open CV

2.1.1 Γιατί Python;

Εικόνα 2-1:Logo Python

Η Python είναι μια υψηλού επιπέδου, διερμηνευόμενη γλώσσα προγραμματισμού που δημιουργήθηκε από τον Guido van Rossum και κυκλοφόρησε το 1991. Είναι επίσης μια γλώσσα προγραμματισμού γενικής χρήσης, απλή και εύκολη στην εκμάθηση, ισχυρή, δυναμική, αποδοτική, παραγωγική και επεκτάσιμη. Μπορεί να χρησιμοποιηθεί για την ανάπτυξη ολοκληρωμένων εφαρμογών, συμπεριλαμβανομένης της ανάπτυξης ιστοσελίδων, της ανάλυσης δεδομένων, επιστημονικών υπολογισμών αλλά και της τεχνητής νοημοσύνης. Μερικά από τα πλεονεκτήματα της χρήσης της Python περιλαμβάνουν:

- Εύκολη εκμάθηση και χρήση: Η Python έχει μια απλή και διαισθητική σύνταξη που καθιστά εύκολη την ανάγνωση και τη συγγραφή κώδικα. Αυτό την καθιστά ιδανική γλώσσα για αρχάριους που μόλις ξεκινούν τον προγραμματισμό.

- Συμβατότητα σε διάφορες πλατφόρμες: Η Python μπορεί να τρέξει σε μια ποικιλία λειτουργικών συστημάτων, συμπεριλαμβανομένων των Windows, macOS και Linux, καθιστώντας την εξαιρετικά φορητή γλώσσα.

- Μεγάλη τυπική βιβλιοθήκη: Η Python διαθέτει μια μεγάλη τυπική βιβλιοθήκη που παρέχει πολλές προ-δημιουργημένες ενότητες και συναρτήσεις για εργασίες όπως η εργασία με αρχεία, οι

κανονικές εκφράσεις και η δικτύωση. Διαθέτει πληθώρα έτοιμων βιβλιοθηκών που μπορούν να χρησιμοποιηθούν εύκολα και άμεσα.

- Ισχυρή και ευέλικτη: Η Python είναι μια ισχυρή γλώσσα που μπορεί να χρησιμοποιηθεί για ένα ευρύ φάσμα εργασιών. Είναι επίσης εξαιρετικά ευέλικτη, πράγμα που σημαίνει ότι μπορεί να χρησιμοποιηθεί σε διάφορα πλαίσια και να ενσωματωθεί εύκολα με άλλες γλώσσες και εργαλεία.

- Ανοιχτού κώδικα και δωρεάν: Η Python είναι λογισμικό ανοιχτού κώδικα, που σημαίνει ότι είναι ελεύθερο να χρησιμοποιηθεί, να διανεμηθεί και να τροποποιηθεί. Αυτό την καθιστά ιδανική γλώσσα τόσο για προσωπική όσο και για εμπορική χρήση.

Η απάντηση λοιπόν, στο ερώτημα "Γιατί Python;" είναι προφανής. Εκμεταλλευόμενοι την πολύ-χρηστικότητα της Python αλλά και την δυναμική της ,κυρίως αναφορικά με την ύπαρξη μιας μεγάλης τυπικής βιβλιοθήκης (Three part Libraries) που παρέχει πολλές προ-δημιουργημένες ενότητες και συναρτήσεις για απλές και πολύπλοκες εργασίες , για την εργασία μας επιλέχθηκε η **cv** module ως έτοιμη βιβλιοθήκη της Python.

2.2 OpenCV (Open Source Computer Vision Library)

Η ενότητα **cv** στην Python, συντομογραφία του **OpenCV (Open Source Computer Vision Library),** είναι μια δημοφιλής βιβλιοθήκη για εργασίες υπολογιστικής όρασης. Παρέχει ένα σύνολο ισχυρών και εύχρηστων λειτουργιών για την επεξεργασία εικόνων και βίντεο, την ανίχνευση και αναγνώριση αντικειμένων, την εξαγωγή χαρακτηριστικών και πολλά άλλα. Μια από τις κύριες χρήσεις της ενότητας cv είναι η επεξεργασία εικόνων. Παρέχει ένα ευρύ φάσμα λειτουργιών για φιλτράρισμα, βελτίωση και μετασχηματισμό εικόνων. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση **cv.cvtColor()** για να μετατρέψουμε μια εικόνα από έναν χρωματικό χώρο σε έναν άλλο, ή να χρησιμοποιήσουμε τη συνάρτηση **cv.GaussianBlur()** για να εφαρμόσουμε ένα φίλτρο Gaussian blur σε μια εικόνα. Επιπλέον, η συνάρτηση **cv.threshold**() μπορεί να χρησιμοποιηθεί για τη δυαδική απόδοση των εικόνων, η οποία είναι μια κοινή τεχνική που χρησιμοποιείται σε πολλές εφαρμογές υπολογιστικής όρασης.

Η ενότητα **cv** παρέχει επίσης λειτουργίες για εξαγωγή και αντιστοίχηση χαρακτηριστικών. Οι δυνατότητες είναι ξεχωριστές περιοχές ή σημεία σε μια εικόνα που μπορούν να χρησιμοποιηθούν για ανίχνευση, αναγνώριση και παρακολούθηση αντικειμένων. Οι συναρτήσεις **cv.SIFT()** και **cv.SURF()** μπορούν να χρησιμοποιηθούν για την εξαγωγή χαρακτηριστικών. Επίσης με την χρήση της ενότητας **cv** μπορεί να γίνει ανίχνευση και αναγνώριση αντικειμένων. Παρέχει μια ποικιλία αλγορίθμων και τεχνικών για την ανίχνευση και την αναγνώριση αντικειμένων σε εικόνες και βίντεο. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε τη λειτουργία **cv.CascadeClassifier()** για να ανιχνεύσουμε πρόσωπα σε μια εικόνα ή να χρησιμοποιήσουμε τη λειτουργία cv.**HOGDescriptor()** για να ανιχνεύσουμε πεζούς σε ένα βίντεο. Η συνάρτηση **cv.matchTemplate()** μπορεί να χρησιμοποιηθεί για αντιστοίχηση προτύπων, η οποία είναι μια τεχνική που χρησιμοποιείται για την εύρεση παρουσιών μιας εικόνας προτύπου σε μια μεγαλύτερη εικόνα.

Η ενότητα **cv** περιλαμβάνει επίσης λειτουργίες για τη βαθμονόμηση της κάμερας και την τρισδιάστατη ανακατασκευή. Η βαθμονόμηση κάμερας είναι η διαδικασία εκτίμησης των εσωτερικών και εξωτερικών παραμέτρων μιας κάμερας, ενώ η τρισδιάστατη ανακατασκευή περιλαμβάνει την ανακατασκευή της τρισδιάστατης δομής ενός αντικειμένου ή μιας σκηνής από πολλαπλές εικόνες 2D. Οι συναρτήσεις **cv.calibrateCamera()** και **cv.stereoCalibrate()** μπορούν να χρησιμοποιηθούν για τη βαθμονόμηση της κάμερας, ενώ η συνάρτηση **cv.triangulatePoints()** μπορεί να χρησιμοποιηθεί για την τρισδιάστατη ανακατασκευή. Συμπερασματικά, η ενότητα **cv** της Python είναι μια ισχυρή και ευέλικτη βιβλιοθήκη για εργασίες υπολογιστικής όρασης. Παρέχει ένα ευρύ φάσμα λειτουργιών και αλγορίθμων για επεξεργασία εικόνας και βίντεο, ανίχνευση και αναγνώριση αντικειμένων, εξαγωγή χαρακτηριστικών, βαθμονόμηση κάμερας και τρισδιάστατη ανακατασκευή αντικειμένων.

2.2.1 Pycharm IDE

Εικόνα 2-2:Logo Pycharm Community.

Ο κώδικας δημιουργήθηκε σε PyCharm (Community edition 2021) που αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) σχεδιασμένο ειδικά για τη γλώσσα προγραμματισμού Python. Πρόκειται για ένα ισχυρό εργαλείο που προσφέρει μια σειρά

χαρακτηριστικών που βοηθούν τους προγραμματιστές να γράφουν, να δοκιμάζουν και να αποσφαλματώνουν τον κώδικα Python. Ένα από τα βασικά χαρακτηριστικά της PyCharm είναι ο έξυπνος επεξεργαστής κώδικα, ο οποίος παρέχει προηγμένη συμπλήρωση κώδικα, επισήμανση σφαλμάτων και πλοήγηση στον κώδικα. Περιλαμβάνει επίσης έναν αποσφαλματωτή (debugger), ο οποίος επιτρέπει στους προγραμματιστές να εξετάζουν τον κώδικά τους και να εντοπίζουν και να διορθώνουν σφάλματα που προκύπτουν. Το PyCharm υποστηρίζει επίσης μια σειρά από πλαίσια και βιβλιοθήκες Python, συμπεριλαμβανομένων των Django, Flask και NumPy, και περιλαμβάνει ενσωματωμένη υποστήριξη για συστήματα ελέγχου εκδόσεων όπως το Git και το Mercurial. Άλλα αξιοσημείωτα χαρακτηριστικά του PyCharm περιλαμβάνουν τις ενσωματωμένες δοκιμές μονάδας, τη σκιαγράφηση κώδικα και την ενσωμάτωση του με διάφορα εργαλεία.

2.3 **MediaPipe**

Εικόνα 2-3: Mediapipe

Η χρήση API Python οδήγησε στη εφαρμογή της Mediapipe. Το Mediapipe είναι ένα πλαίσιο ανοικτού κώδικα για την κατασκευή σωληνώσεων υπολογιστικής όρασης και μηχανικής μάθησης σε πραγματικό χρόνο. Δημιουργήθηκε από την Google και κυκλοφόρησε το 2019 και παρέχει ένα ολοκληρωμένο σύνολο εργαλείων και προ-δημιουργημένων μοντέλων για την επεξεργασία και ανάλυση δεδομένων εικόνας και βίντεο.

Το Mediapipe μπορεί να χρησιμοποιηθεί για ένα ευρύ φάσμα εφαρμογών, συμπεριλαμβανομένης της αναγνώρισης προσώπου, της ανίχνευσης και παρακολούθησης αντικειμένων, της παρακολούθησης χεριών, της εκτίμησης πόζας και της επαυξημένης πραγματικότητας.

Το Mediapipe διαθέτει ένα API Python, το οποίο καθιστά εύκολη τη χρήση του με τη γλώσσα προγραμματισμού Python. Οι προγραμματιστές της Python μπορούν να χρησιμοποιήσουν το Mediapipe Python API για να δημιουργήσουν εφαρμογές υπολογιστικής όρασης σε πραγματικό χρόνο χρησιμοποιώντας προκατασκευασμένα μοντέλα ή δημιουργώντας τα δικά τους μοντέλα.

Τελικά ,η χρήση του Mediapipe με την Python μπορεί να είναι ένας πολύ καλός τρόπος για τη γρήγορη δημιουργία πρωτοτύπων και εφαρμογών υπολογιστικής όρασης. Η ευκολία χρήσης της Python και το ευρύ φάσμα βιβλιοθηκών την καθιστούν ιδανική γλώσσα για τη δημιουργία τέτοιων εφαρμογών, ενώ το Mediapipe παρέχει ένα ισχυρό σύνολο εργαλείων και μοντέλων για την επεξεργασία δεδομένων εικόνας και βίντεο. Στο πλαίσιο του Mediapipe, το cv χρησιμοποιείται συχνά σε συνδυασμό με το Mediapipe για την επεξεργασία και ανάλυση δεδομένων εικόνας και βίντεο. Για παράδειγμα, η ενότητα Mediapipe Hand Tracking χρησιμοποιεί το **cv** για τη λήψη εικόνων από μια ροή βίντεο και την προ-επεξεργασία τους πριν την αποστολή τους στο μοντέλο εντοπισμού χεριών, για ανάλυση.

Είναι φανερό ότι ο συνδυασμός **cv** and Mediapipe επιτρέπει αρκετές εργασίες που συμπεριλαμβάνουν την

- Λήψη και επεξεργασία ροών βίντεο
- Προ-επεξεργασία εικόνων για χρήση με μοντέλα μηχανικής μάθησης
- Επικάλυψη γραφικών στοιχείων πάνω σε εικόνες και ροές βίντεο
- Μετατροπή εικόνων και ροών βίντεο σε διαφορετικές μορφές και αναλύσεις
- Εκτέλεση φιλτραρίσματος και επεξεργασίας εικόνας, όπως θόλωση, όξυνση και ανίχνευση ακμών.

Συνολικά, το cv είναι ένα βασικό εργαλείο για την εργασία με δεδομένα εικόνας και βίντεο στην Python και παίζει σημαντικό ρόλο σε πολλές εφαρμογές υπολογιστικής όρασης, συμπεριλαμβανομένων εκείνων που έχουν κατασκευαστεί και με τη χρήση του Mediapipe

2.4 **Networking- UDP Server - Μεταφορά δεδομένων**

Η μεταφορά δεδομένων μέσω δικτύου αναφέρεται στη διαδικασία μετάδοσης δεδομένων μεταξύ δύο ή περισσότερων συσκευών μέσω δικτύου. Η διαδικασία αυτή περιλαμβάνει διάφορα στάδια, συμπεριλαμβανομένης της κωδικοποίησης, της μετάδοσης και της αποκωδικοποίησης δεδομένων, και μπορεί να εκτελεστεί με τη χρήση διαφόρων πρωτοκόλλων και τεχνολογιών, όπως TCP/IP, HTTP, FTP και άλλα. Η σημασία της μεταφοράς δεδομένων μέσω δικτύων δεν

μπορεί να υπερτιμηθεί, καθώς αποτελεί τη βάση για πολλές σύγχρονες τεχνολογίες και εφαρμογές, όπως το Διαδίκτυο, το cloud computing και η ανάλυση μεγάλων δεδομένων. Η αξιόπιστη και αποτελεσματική μεταφορά δεδομένων είναι ζωτικής σημασίας για να διασφαλιστεί ότι οι εφαρμογές και οι υπηρεσίες μπορούν να λειτουργούν σωστά και να παρέχουν αξία στους χρήστες. Συνολικά, η μεταφορά δεδομένων μέσω δικτύωσης είναι ένα κρίσιμο στοιχείο της σύγχρονης τεχνολογίας και διαδραματίζει καθοριστικό ρόλο στην επικοινωνία και τη συνεργασία μεταξύ διαφορετικών συσκευών και δικτύων. Η μεταφορά των δεδομένων στην παρούσα εργασία πραγματοποιήθηκε με την βοήθεια του UDP πρωτοκόλλου.

2.4.1 User Datagram (UDP)

Το πρωτόκολλο User Datagram (UDP) είναι ένα πρωτόκολλο χωρίς σύνδεση που λειτουργεί στο επίπεδο μεταφοράς του μοντέλου OSI. Σε αντίθεση με το Πρωτόκολλο Ελέγχου Μετάδοσης (TCP), το οποίο παρέχει αξιόπιστη, προσανατολισμένη στη σύνδεση επικοινωνία, το UDP παρέχει γρήγορη, αναξιόπιστη, χωρίς σύνδεση επικοινωνία.

Το UDP χρησιμοποιείται συνήθως για εφαρμογές πραγματικού χρόνου, όπως διαδικτυακά παιχνίδια, ροή βίντεο και φωνή μέσω IP (VoIP), όπου μικρές ποσότητες δεδομένων πρέπει να αποστέλλονται γρήγορα και αποτελεσματικά. Το πρωτόκολλο χρησιμοποιείται επίσης σε εφαρμογές του Διαδικτύου των Πραγμάτων (IoT) για την αποστολή μικρών ποσοτήτων δεδομένων μεταξύ συσκευών.

Τα πακέτα UDP, γνωστά και ως datagrams, αποστέλλονται χωρίς να δημιουργείται σύνδεση μεταξύ του αποστολέα και του παραλήπτη και δεν υπάρχει καμία εγγύηση ότι τα δεδομένα θα παραδοθούν ή θα ληφθούν με τη σωστή σειρά. Ωστόσο, το UDP είναι ένα ελαφρύ και γρήγορο πρωτόκολλο που είναι ιδανικό για εφαρμογές που δίνουν προτεραιότητα στην ταχύτητα έναντι της αξιοπιστίας.

Όμως παρά τους περιορισμούς του, το UDP έχει αρκετά πλεονεκτήματα έναντι του TCP σε ορισμένες περιπτώσεις. Για παράδειγμα, το UDP χρησιμοποιείται συχνά για εφαρμογές

πραγματικού χρόνου, όπως διαδικτυακά παιχνίδια και ροή βίντεο, όπου μικρές ποσότητες δεδομένων πρέπει να αποστέλλονται γρήγορα και αποτελεσματικά.

Ένας διακομιστής UDP μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα εφαρμογών, όπως:

- Διαδικτυακά παιχνίδια: Το UDP χρησιμοποιείται συχνά σε διαδικτυακά παιχνίδια για την παροχή γρήγορης επικοινωνίας σε πραγματικό χρόνο μεταξύ των παικτών.
- Ροή βίντεο: Το UDP χρησιμοποιείται σε εφαρμογές ροής βίντεο για τη γρήγορη και αποτελεσματική παράδοση περιεχομένου βίντεο.
- Φωνή μέσω IP (VoIP): Το UDP χρησιμοποιείται συχνά σε εφαρμογές VoIP για την παροχή γρήγορης φωνητικής επικοινωνίας σε πραγματικό χρόνο.
- Διαδίκτυο των πραγμάτων (IoT): Το UDP μπορεί να χρησιμοποιηθεί σε εφαρμογές IoT για την αποστολή μικρών ποσοτήτων δεδομένων μεταξύ συσκευών.

Συνολικά, ένας διακομιστής UDP μπορεί να είναι ένα χρήσιμο εργαλείο για τη δημιουργία εφαρμογών πραγματικού χρόνου που απαιτούν γρήγορη και αποτελεσματική επικοινωνία μεταξύ πελατών και διακομιστών μέσω δικτύου και παίζει καθοριστικό ρόλο σε πολλές σύγχρονες τεχνολογίες και εφαρμογές, ιδίως σε εκείνες που απαιτούν επικοινωνία σε πραγματικό χρόνο με χαμηλή καθυστέρηση. Από την άλλη πλευρά η απλότητα του UDP το καθιστά εύκολο στη χρήση και την υλοποίηση και απαιτεί λιγότερα γενικά έξοδα από το TCP, καθιστώντας το ελκυστική επιλογή για ορισμένους τύπους εφαρμογών κάτι που εμφανώς βοήθησε την επιλογή του συγκεκριμένου πρωτοκόλλου στην παρούσα εργασία.

2.3 Το πρόγραμμα Blender

Το πρόβλημα της μοντελοποίησης τον δεδομένων εμφανίστηκε στο προσκήνιο. Υπάρχουν πολλά λογισμικά δημιουργίας 3D με αρκετά δημοφιλή τα Blender και Unity .

2.4.2 Blender

Το Blender είναι ένα ισχυρό και ευέλικτο λογισμικό δημιουργίας 3D ανοιχτού κώδικα που χρησιμοποιείται ευρέως στις βιομηχανίες Animation, ανάπτυξης βιντεοπαιχνιδιών και οπτικών

εφέ. Προσφέρει ένα ευρύ φάσμα χαρακτηριστικών και δυνατοτήτων που επιτρέπουν στους καλλιτέχνες και τους σχεδιαστές να δημιουργούν εντυπωσιακά τρισδιάστατα μοντέλα, κινούμενα σχέδια και διαδραστικές εμπειρίες. Ένα από τα βασικά πλεονεκτήματα του Blender είναι η ευελιξία και η πολύ-χρηστικότητά του. Το λογισμικό μπορεί να χρησιμοποιηθεί για ένα ευρύ φάσμα εργασιών, από τη δημιουργία απλών τρισδιάστατων μοντέλων έως την ανάπτυξη σύνθετων κινούμενων σχεδίων και οπτικών εφέ. Αυτή η ευελιξία καθιστά το Blender ένα πολύτιμο εργαλείο για καλλιτέχνες και σχεδιαστές που εργάζονται σε διάφορους κλάδους. Ένα άλλο πλεονέκτημα του Blender, είναι το μοντέλο ανάπτυξης του με γνώμονα την κοινότητα. Ως λογισμικό ανοικτού κώδικα, το Blender ενημερώνεται και βελτιώνεται συνεχώς από μια κοινότητα προγραμματιστών και χρηστών από όλο τον κόσμο. Αυτό σημαίνει ότι προστίθενται τακτικά νέα χαρακτηριστικά και δυνατότητες και ότι τα σφάλματα και τα προβλήματα αντιμετωπίζονται γρήγορα.

2.4.3 Unity

Από την άλλη πλευρά, το Unity έχει σχεδιαστεί κυρίως για την ανάπτυξη παιχνιδιών και προσφέρει ένα ευρύ φάσμα εργαλείων και χαρακτηριστικών ειδικά προσαρμοσμένων για αυτόν τον σκοπό. Περιλαμβάνει μια ισχυρή μηχανή παιχνιδιών που μπορεί να χρησιμοποιηθεί για τη δημιουργία παιχνιδιών για διάφορες πλατφόρμες, όπως επιτραπέζια, κινητά και VR. Το Unity προσφέρει επίσης μια σειρά από δυνατότητες για τη δημιουργία διαδραστικών τρισδιάστατων περιβαλλόντων, όπως εφέ φωτισμού και σωματιδίων.

2.4.4 Γιατί Blender

Όταν πρόκειται για τρισδιάστατη μοντελοποίηση , το Blender προσφέρει ορισμένα σαφή πλεονεκτήματα έναντι του Unity. Το Blender περιλαμβάνει μια σειρά εργαλείων για τη δημιουργία και τον χειρισμό τρισδιάστατων μοντέλων, καθώς και έναν ενσωματωμένο επεξεργαστή βίντεο που μπορεί να χρησιμοποιηθεί για τη δημιουργία και την επεξεργασία περιεχομένου βίντεο απευθείας μέσα στο λογισμικό.

Η επιλογή του συγκεκριμένου προγράμματος ,έγινε φανερά με κριτήριο την λειτουργικότητα του Blender για την συγκεκριμένη εργασία. Σε σύγκριση με άλλα λογισμικά δημιουργίας 3D, το

Blender προσφέρει αρκετά μοναδικά χαρακτηριστικά και δυνατότητες που το κάνουν να ξεχωρίζει. Περιλαμβάνει επίσης έναν ενσωματωμένο επεξεργαστή βίντεο που μπορεί να χρησιμοποιηθεί για τη δημιουργία και επεξεργασία περιεχομένου βίντεο απευθείας μέσα στο λογισμικό. Το API Python, που λειτουργεί μέσα από το Blender του επιτρέπει τη δημιουργία προσαρμοσμένων σεναρίων και plugins που μπορούν να επεκτείνουν τη λειτουργικότητα του λογισμικού ακόμη περισσότερο. Αυτό το χαρακτηριστικό καθιστά το Blender μια εξαιρετική επιλογή αφού το ζητούμενο είναι η προσαρμογή των ροών εργασίας που οδηγούν σε αυτοματοποίηση. Παρόλο τους μερικούς περιορισμούς όπως η εκμάθηση και η χρήση του ,να είναι πιο δύσκολη από άλλα λογισμικά δημιουργίας 3D (Unity) , καθώς και τα προβλήματα σταθερότητας και απόδοσης, ιδίως όταν εργάζονται με πολύ μεγάλες ή πολύπλοκες σκηνές η επιλογή του αξιολογήθηκε ως απόλυτα ικανοποιητική.

3. Σχεδιασμός / Μοντελοποίηση

Αρχικά προσδιορίζεται η σκοπιμότητα ανάπτυξης της εφαρμογής ,η πλατφόρμα λειτουργίας της καθώς και οι δυνατότητες που καθορίζονται από τις λειτουργικές της απαιτήσεις. Η διαδικασία που ακολουθήθηκε έχει δυο σκέλη. Το κομμάτι της εφαρμογής το οποίο τρέχει σε κώδικα Python ,χρησιμοποιώντας το **cv** module της γλώσσας μαζί με το Mediapipe και το κομμάτι που θα τρέχει στον εξυπηρετητή και μετατρέπει τα δεδομένα που προσλαμβάνονται από το πρώτο κομμάτι σε μια 3D απεικόνιση πραγματικού χρόνου(Blender-Python). Για να δημιουργήσουμε το πρόγραμμα που θα εκτελεί την παρακολούθηση των χεριών, θα χρειαστούμε δύο βιβλιοθήκες Python. Αυτά είναι το openCV ,κυρίως τη βιβλιοθήκη Cv , και το MediaPipe. Θα χρησιμοποιήσουμε το openCV για την εκτέλεση λειτουργιών που σχετίζονται με την επεξεργασία των δεδομένων εικόνας από την κάμερα ,ενώ η χρήση του MediaPipe θα χρειαστεί στην εκτέλεση για την πραγματική ανίχνευση και παρακολούθηση του χεριού στην εικόνα εισόδου μας. Επιπροσθέτως η χρήση της Python θα γίνει μέσα από το Pycharm IDE.

Αρχικά γίνεται εγκατάσταση των βιβλιοθηκών με τις ακόλουθες εντολές στο τερματικό μέσω IDE της Pycharm για να εγκαταστήσουμε το MediaPipe καθώς και το OpenCv:

- **pip install mediapipe**
- **pip install opencv-python**

Τώρα με έτοιμο το περιβάλλον μας ,Θα ξεκινήσουμε με τη δημιουργία ενός προγράμματος που κάνει παρακολούθησης χεριών. Πριν ξεκινήσουμε την κωδικοποίηση, ας αναφερθούμε πώς το MediaPipe εκτελεί την παρακολούθηση των χεριών. Η ανίχνευση χεριών με τη χρήση του MediaPipe περιλαμβάνει δύο στάδια:

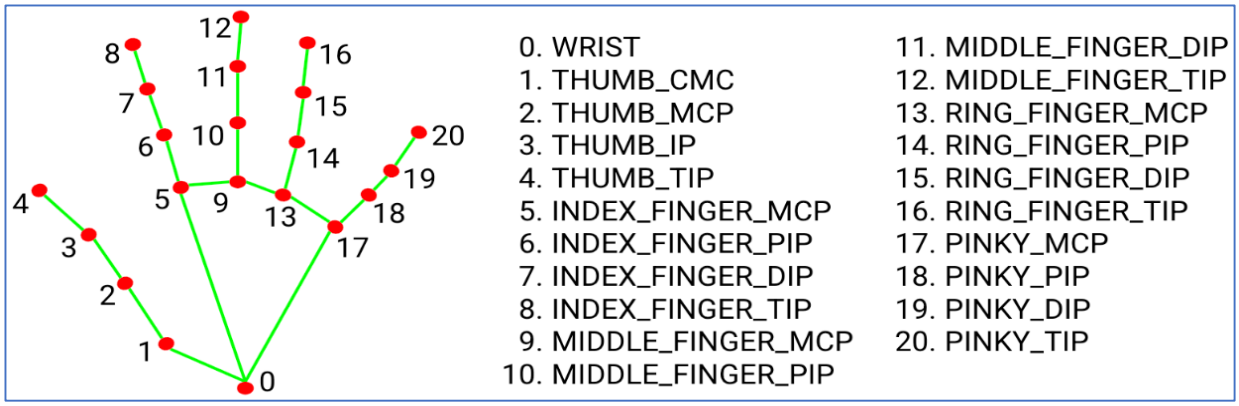
Ανίχνευση παλάμης : Το MediaPipe εργάζεται στην πλήρη εικόνα εισόδου και παρέχει μια περικομμένη εικόνα του χεριού.

Αναγνώριση ορόσημων (landmarks) του χεριού : Το MediaPipe βρίσκει τα 21 ορόσημα του χεριού στην περικομμένη εικόνα του χεριού. Εικόνα 3-1

Αρχικά γίνεται η ανίχνευση χεριών. Ο κώδικας ξεκινά με έναν βρόχο που καταγράφει συνεχώς καρέ βίντεο από την προεπιλεγμένη κάμερα χρησιμοποιώντας τη συνάρτηση "cap.read()". Στη συνέχεια, η εικόνα μετατρέπεται από τη μορφή BGR σε RGB χρησιμοποιώντας την "cv2.cvtColor()". Η συνάρτηση " detector = HandDetector ()" χρησιμοποιείται για τον εντοπισμό των χεριών στην εικόνα και την επιστροφή των σημείων αναφοράς τους οι παράμετροι της συνάρτησης **detectionCon**, δίνουν ικανότητα στην ανίχνευση του χεριού σε ποσοστό 80%, **maxHands** με τον αριθμό των χεριών που μπορεί να ανιχνεύσει σε 1.

Στην συνέχεια το χέρι αφού ανιχνευθεί σχηματίζεται ένα περίγραμμα με το χέρι μέσα όπου φαίνονται τα σημεία (Landmarks) αυτή είναι μια πρώτη απόδοση των αποτελεσμάτων εντοπισμού των χεριών. Εάν η μεταβλητή "results.multi_hand_landmarks" περιέχει ορόσημα χεριών, ο κώδικας επαναλαμβάνει όλα τα ανιχνευμένα χέρια, αναθέτει ένα μοναδικό αναγνωριστικό σε κάθε χέρι και σχεδιάζει τα ορόσημα στην εικόνα χρησιμοποιώντας τη συνάρτηση "mp_drawing.draw_landmarks()". Η συνάρτηση λαμβάνει διάφορα ορίσματα: την

εικόνα εισόδου, τα ορόσημα του χεριού, τις συνδέσεις μεταξύ των ορόσημων και δύο αντικείμενα "DrawingSpec" που καθορίζουν το χρώμα, το πάχος και την ακτίνα των κύκλων που θα σχεδιαστούν γύρω από τα ορόσημα. Το πρώτο αντικείμενο "DrawingSpec" χρησιμοποιείται για τη σχεδίαση των ορόσημων και το δεύτερο για τη σχεδίαση των συνδέσεων μεταξύ των ορόσημων.



Εικόνα 3-1: Τα 21 ορόσημα (Landmarks) μαζί με τις 21 συνδέσεις τους.

Στην περίπτωση μας , όπως παρουσιάζεται στην **Εικόνα 3-1** ,έχουν αποτυπωθεί τα 21 σημεία καθώς και η δημιουργία των 21 landmarks (σημείων), ξεκινώντας από το κεντρικό σημείο της παλάμης (0 Wrist) μέχρι το 20 (Pinky_tip), μαζί με τις συνδέσεις τους. Η σύνδεση των οροσήμων γίνεται με την αποθήκευση των σημείων id (αναγνωριστικά σημείων 0--20) των landmark points (0-20) σε μία άδεια αρχικά λίστα **lmList** , και την δημιουργία ενώσεων με την ήδη έτοιμη λίστα **join_list** [(0, 1), (1, 2), (2, 3), (3, 4), (0, 5), (5, 6), (6, 7), (7, 8), (5, 9), (9, 10), (10, 11), (11, 12), (9, 13), (13, 14), (14, 15), (15, 16), (0, 17), (13, 17), (17, 18), (18, 19), (19, 20)]

Από την στιγμή που το χέρι έχει αποτυπωθεί στην οθόνη σε πραγματικό χρόνο και έχει γίνει η αποτύπωση των οροσήμων (landmarks) με την ταυτόχρονη σύνδεση τους με γραμμές , μπορεί να βρεθεί γωνιακή απόσταση των δακτύλων καθώς και η γωνία που σχηματίζεται από την κλίση των οροσήμων. Τα αποτελέσματα καταγράφονται και αποθηκεύονται σε αρχείο κειμένου TEXT ή και σε αρχείο Excel, ώστε να χρησιμοποιήσουν αργότερα.

Πρέπει να τονισθεί στο σημείο αυτό ότι οι συναρτήσεις που χρησιμοποιήθηκαν στον κώδικα και δημιουργήθηκαν ανεξάρτητα για να υλοποιήσουν την λειτουργικότητα της εργασίας αυτής

ενσωματώθηκαν σε μια βιβλιοθήκη της Python (module) με το όνομα HandsModule για λόγους απλότητας και συνοχής του κώδικα αλλά και για την επέκταση της εφαρμογής στην λειτουργία της 3D απεικόνισης μέσω του προγράμματος Blender.

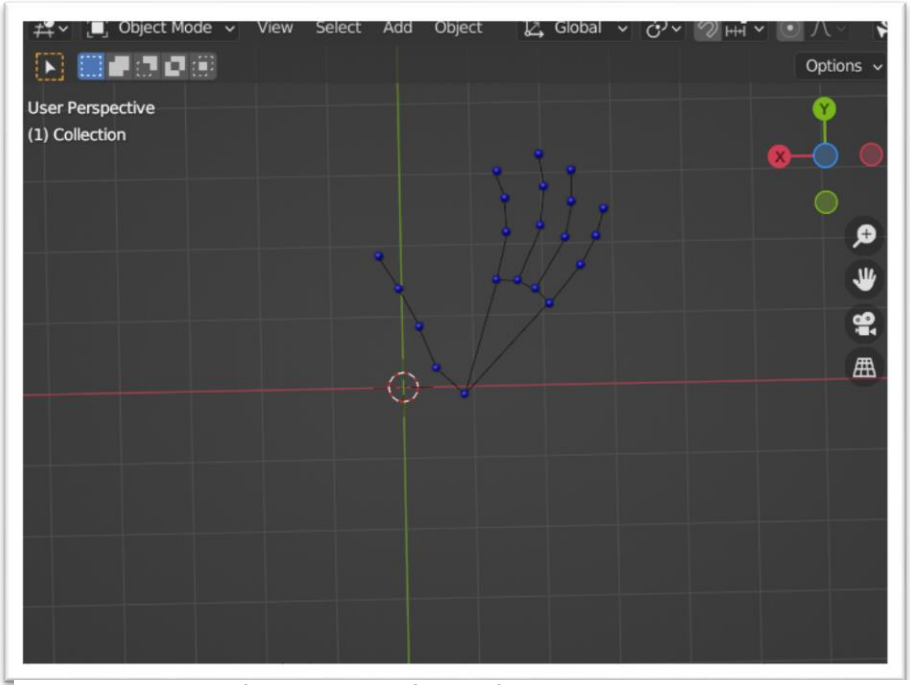
Με το τέλος αυτής της διαδικασίας που γίνεται σε πραγματικό χρόνο ακολουθεί το δεύτερο σκέλος της, όπου με την χρήση του προγράμματος Blender σε συνδυασμό με κώδικα σε Python API που ενσωματώνεται εύκολα στο Blender , μπορούμε να αναλύσουμε το δεύτερο σκέλος της διαδικασίας.

Αρχικά, πρέπει να τονισθεί ότι τα δεδομένα που αποκτήθηκαν σε πραγματικό χρόνο στο πρώτο σκέλος της διαδικασίας αλλάζουν συνεχώς. Το χέρι στην οθόνη της κάμερας σε κάθε καρέ που λαμβάνεται δεν είναι στην ίδια θέση, επομένως οι συντεταγμένες των οροσήμων (Landmark points) αλλάζουν. Οι τιμές αυτές πρέπει να συγκεντρωθούν και να σταλούν στο πρόγραμμα Blender , ώστε να πραγματοποιήσει το κόμματι της 3D απεικόνισης.

Στο σημείο αυτό πρέπει να αναφερθεί ότι στην αρχική εφαρμογή αναγράφεται στην οθόνη ο αριθμός των frames της εικόνας της κάμερας , ανά δευτερόλεπτο (FPS frame per second) ώστε να υπάρχει μια κατάλληλη ρύθμιση και στο Blender, για την αποφυγή της ασύγχρονης αποδοχής των δεδομένων.

Η μεταφορά των δεδομένων ,συντεταγμένες των οροσήμων (Landmark points), γίνεται με την χρήση του πρωτοκόλλου UDP στο πρόγραμμα Blender που με την χρήση κώδικα Python λειτουργεί ώς αποδέκτης των δεδομένων (Server). Τα δεδομένα κωδικοποιούνται ως bytes χρησιμοποιώντας την “str.encode()” συνάρτηση , η οποία επιστρέφει ένα αντικείμενο bytes. Αυτό το αντικείμενο bytes αποστέλλεται στη συνέχεια στον προορισμό που καθορίζεται από την διεύθυνση του αποδέκτη, συνεχίζοντας και μετά την παραλαβή των δεδομένων, γίνεται η αποκωδικοποίηση τους από bytes σε αριθμητικές τιμές (float numbers) που εκφράζουν τις συντεταγμένες των οροσήμων (landmarks) και τις οποίες χρησιμοποιούμε στο πρόγραμμα Blender ως τις νέες συντεταγμένες των σημείων που θέλουμε να προβάλουμε σε 3D .

Το πρόγραμμα Blender μαζί με τον κώδικα σε python αναλαμβάνουν να οδηγήσουν τη διαδικασία στο τελικό αποτέλεσμα ,την 3D απεικόνιση των οροσήμων ,σε πραγματικό χρόνο δημιουργώντας και της συνδέσεις μεταξύ των οροσήμων. Το τελικό αποτέλεσμα αποτυπώνεται στην **Εικόνα 3-2.**



Εικόνα 3-2:Τελικό αποτέλεσμα στο Blender

4. Υλοποίηση του προγράμματος -κωδικοποίηση

Σε αυτό το κεφάλαιο θα παρουσιαστούν τα κομμάτια κώδικα που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Αρχικά έγινε χρήση ενός module που δημιουργήθηκε για να κάνει το κώδικα μας ποιο ευέλικτο και λειτουργικό. Εδώ περιέχονται όλες οι συναρτήσεις που θα χρειαστούν κατά τη διάρκεια της αναγνώρισης του χεριού.

Πίνακας 4-1:Module Hand Tracking σε Python 3.10

```
"""
Hand Tracking Module
"""

import cv2
import mediapipe as mp
import math


class HandDetector:
    """
    Finds Hands using the mediapipe library. Exports the landmarks
```

```
    in pixel format. Adds extra functionalities like finding how
    many fingers are up or the distance between two fingers. Also
    provides bounding box info of the hand found.
    """

    def __init__(self, mode=False, maxHands=2, detectionCon=0.5,
minTrackCon=0.5):
        """
        :param mode: In static mode, detection is done on each image: slower
        :param maxHands: Maximum number of hands to detect
        :param detectionCon: Minimum Detection Confidence Threshold
        :param minTrackCon: Minimum Tracking Confidence Threshold
        """
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.minTrackCon = minTrackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(static_image_mode=self.mode,
max_num_hands=self.maxHands,

min_detection_confidence=self.detectionCon,

min_tracking_confidence=self.minTrackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]
        self.fingers = []
        self.lmList = []

    def findHands(self, img, draw=True, flipType=True):
        """
        Finds hands in a BGR image.
        :param img: Image to find the hands in.
        :param draw: Flag to draw the output on the image.
        :return: Image with or without drawings
        """
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        allHands = []
        h, w, c = img.shape
        if self.results.multi_hand_landmarks:
            for handType, handLms in zip(self.results.multi_handedness,
self.results.multi_hand_landmarks):
                myHand = {}
                ## lmList
                mylmList = []
                xList = []
                yList = []
                for id, lm in enumerate(handLms.landmark):
                    px, py, pz = int(lm.x * w), int(lm.y * h), int(lm.z * w)
                    mylmList.append([px, py, pz])
                    xList.append(px)
```

```
                    yList.append(py)

                ## bbox
                xmin, xmax = min(xList), max(xList)
                ymin, ymax = min(yList), max(yList)
                boxW, boxH = xmax - xmin, ymax - ymin
                bbox = xmin, ymin, boxW, boxH
                cx, cy = bbox[0] + (bbox[2] // 2), \
                         bbox[1] + (bbox[3] // 2)

                myHand["lmList"] = mylmList
                myHand["bbox"] = bbox
                myHand["center"] = (cx, cy)

                if flipType:
                    if handType.classification[0].label == "Right":
                        myHand["type"] = "Left"
                    else:
                        myHand["type"] = "Right"
                else:
                    myHand["type"] = handType.classification[0].label
                allHands.append(myHand)

                ## draw
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                               self.mpHands.HAND_CONNECTIONS)
                    cv2.rectangle(img, (bbox[0] - 20, bbox[1] - 20),
                                  (bbox[0] + bbox[2] + 20, bbox[1] + bbox[3] +
20),
                                  (255, 0, 255), 2)
                    cv2.putText(img, myHand["type"], (bbox[0] - 30, bbox[1] -
30), cv2.FONT_HERSHEY_PLAIN,
                                2, (255, 0, 255), 2)
        if draw:
            return allHands, img
        else:
            return allHands
```

Παρουσιάζεται η συνάρτηση που αναγνωρίζει το χέρι κάνει διάκριση σε Αριστερό ή Δεξί χέρι ενώ ,ταυτόχρονα δημιουργεί μια λίστα (lmList) από τα landmarks του χεριού (ορόσημα) , υπολογίζει τις συντεταγμένες px, py, pz των οροσήμων ,τις οποίες πολλαπλασιάζει με το Height, Width της εικόνας από την κάμερα. Αξίζει να αναφερθεί ότι η προηγούμενη διαδικασία γίνεται για την κανονικοποίηση των σημείων, ως προς σημείο αναφοράς (landmark 0) . Στην συνέχεια

δημιουργεί μια νέα λίστα με τις παραπάνω τιμές , (mylist), από την οποία επιλέγουμε τις νέες κανονικοποιμένες τιμές **x, y** των ακραίων σημείων της εικόνα του χεριού και με την χρήση max ,min τιμών δημιουργεί το περίγραμμα (Box) γύρω από το χέρι ενώ πάνω σε αυτό αναγράφεται αριστερό ή δεξί ανάλογα με πιο χέρι εμφανίζουμε μπροστά στην εικόνα.

Μια άλλη συνάρτηση από το module Hand Tracking είναι η συνάρτηση fingersUp που βρίσκει πόσα δάχτυλα είναι ανοιχτά και τα επιστρέφει σε μια λίστα. Λαμβάνει υπόψη το αριστερό και το δεξί χέρι χωριστά.

Πίνακας 4-2:Συνάρτηση fingersUp

```
def fingersUp(self, myHand):
        """
        Finds how many fingers are open and returns in a list.
        Considers left and right hands separately
        :return: List of which fingers are up
        """
        myHandType = myHand["type"]
        myLmList = myHand["lmList"]
        if self.results.multi_hand_landmarks:
            fingers = []
            # Thumb
            if myHandType == "Right":
                if myLmList[self.tipIds[0]][0] > myLmList[self.tipIds[0] -
1][0]:
                    fingers.append(1)
                else:
                    fingers.append(0)
            else:
                if myLmList[self.tipIds[0]][0] < myLmList[self.tipIds[0] -
1][0]:
                    fingers.append(1)
                else:
                    fingers.append(0)

            # 4 Fingers
            for id in range(1, 5):
                if myLmList[self.tipIds[id]][1] < myLmList[self.tipIds[id] -
2][1]:
                    fingers.append(1)
                else:
                    fingers.append(0)
        return fingers
```

Η συνάρτηση findDistance, βρίσκει την απόσταση μεταξύ δύο ορόσημων (landmarks) με βάση τον μοναδικό αριθμό που έχει αποδοθεί (0-20) σε κάθε ορόσημο (id)

Πίνακας 4-3:Συνάρτηση findDistance

```
def findDistance(self, p1, p2, img=None):
        """
        Find the distance between two landmarks based on their
        index numbers.
        :param p1: Point1
        :param p2: Point2
        :param img: Image to draw on.
        :param draw: Flag to draw the output on the image.
        :return: Distance between the points
                 Image with output drawn
                 Line information
        """

        x1, y1 = p1
        x2, y2 = p2
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2
        length = math.hypot(x2 - x1, y2 - y1)
        info = (x1, y1, x2, y2, cx, cy)
        if img is not None:
            cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), 15, (255, 0, 255), cv2.FILLED)
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), 3)
            cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)
            return length, info, img
        else:
            return length, info
```

Στο τέλος εμφανίζεται η συνάρτηση που αναζητά και υπολογίζει την γωνιά μεταξύ συγκεκριμένων οροσήμων από μία λίστα (joint_list). Οι τιμές των γωνιών υπολογίζονται σε πραγματικό χρόνο ανάλογα με την κάμψη των δακτύλων, ενώ τα αποτελέσματα αποθηκεύονται σε TEXT κείμενο, για περαιτέρω μελέτη από κάθε χρήστη της εφαρμογής. Σημειώνεται ότι οι αποθήκευση των τιμών μπορεί να γινεί σε Excel ή Csv αρχείο. Τελικά γίνεται η ταυτοποίηση του χεριού και αναγράφεται στα επιλεγμένα ορόσημα η γωνίας τους στην εικόνα του βίντεο ,πάντα σε πραγματικό χρόνο.

Πίνακας 4-4:Συνάρτηση draw_finger_angle

```
joint_list = [[8,7,6], [12,11,10], [16,15,14], [20,19,18]]
```

```
def draw_finger_angles(image, results, joint_list):

    # Loop through hands
    for hand in results.multi_hand_landmarks:
        #Loop through joint sets
        for joint in joint_list:
            a = np.array([hand.landmark[joint[0]].x, hand.landmark[joint[0]].y])
# First coord
            b = np.array([hand.landmark[joint[1]].x, hand.landmark[joint[1]].y])
# Second coord
            c = np.array([hand.landmark[joint[2]].x, hand.landmark[joint[2]].y])
# Third coord

            radians = np.arctan2(c[1] - b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1],
a[0]-b[0])
            angle = np.abs(radians*180.0/np.pi)

            if angle > 180.0:
                angle = 360-angle

            cv2.putText(image, str(round(angle, 2)), tuple(np.multiply(b, [640,
480]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA)
    return image
```

Το υπόλοιπο module που ακολουθεί είναι η κύρια συνάρτηση όπως αυτή θα χρησιμοποιηθεί ανάλογα με τις ανάγκες της εφαρμογής.

Πίνακας 4-5: Συνάρτηση Main από το Hand Tracking module

```
def main():
    cap = cv2.VideoCapture(0)
    detector = HandDetector(detectionCon=0.8, maxHands=2)
    while True:
        # Get image frame
        success, img = cap.read()
        # Find the hand and its landmarks
        hands, img = detector.findHands(img)  # with draw
        # hands = detector.findHands(img, draw=False)  # without draw

        if hands:
            # Hand 1
            hand1 = hands[0]
            lmList1 = hand1["lmList"]  # List of 21 Landmark points
            bbox1 = hand1["bbox"]  # Bounding box info x,y,w,h
            centerPoint1 = hand1['center']  # center of the hand cx,cy
            handType1 = hand1["type"]  # Handtype Left or Right
```

```
            fingers1 = detector.fingersUp(hand1)

            if len(hands) == 2:
                # Hand 2
                hand2 = hands[1]
                lmList2 = hand2["lmList"]  # List of 21 Landmark points
                bbox2 = hand2["bbox"]  # Bounding box info x,y,w,h
                centerPoint2 = hand2['center']  # center of the hand cx,cy
                handType2 = hand2["type"]  # Hand Type "Left" or "Right"

                fingers2 = detector.fingersUp(hand2)

                # Find Distance between two Landmarks. Could be same hand or
different hands
                length, info, img = detector.findDistance(lmList1[8][0:2],
lmList2[8][0:2], img)  # with draw
                # length, info = detector.findDistance(lmList1[8], lmList2[8])
# with draw
        # Display
        cv2.imshow("Image", img)
        cv2.waitKey(1)


if __name__ == "__main__":
    main()
```

Ακολουθεί η διαδικασία της χρήσης του Hand Tracking module ως βιβλιοθήκη στο κυρίως πρόγραμμα. Αρχικά γίνεται η παραμετροποίηση και η αρχικοποίηση των μεταβλητών του θα χρειαστούν στην συνέχεια καθώς και η δημιουργία της θύρας του UDP Server όπου και θα σταλούν τα δεδομένα για την 3D αναπαραγωγή του χεριού ,που θα πραγματοποιηθεί με το πρόγραμμα Blender.

Πίνακας 4-6: Main program ,Hand3D_Sender

```
1  #usage of python 3.10
 2 import cv2
 3 import mediapipe as mp
 4 import math
 5 import time
 6 import socket
 7 from cvzone import HandTrackingModule
 8
 9 # define the time for frames  time fps
10 prevtime = 0
11 curtime = 0
12
13 #define as variables the video parameters
```

```
width, height =1280, 720
#640-->1280
#Webcamera
cap = cv2.VideoCapture(0)
cap.set(3,  width)
cap.set(4, height)

#detector
detector = HandTrackingModule.HandDetector(maxHands=1, detectionCon=0.8)

#communication
port = 5053
serverAddressPorts =("127.0.0.1", port)
sock=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

prevtime = time.time()
while True:
    # Get the frame of the webcam
    success, img = cap.read()
    #Hands
    hands, img = detector.findHands(img)

    data = [] # these data is going to send

    # is putting in the while loop because whe want each time fresh data and
not the last one

    handpoints_txt = open("C:\\Users\\vasil\\Downloads\\textfiles\\Hand.txt",
'w')

    # Landmarks  values (x,y,z)*21=63 values and will send in a single list
    if hands:
        #Get the first hand detected
        hand = hands[0]
        # Get the landmark list
        lmList = hand['lmList']  # this is dictionary
       # print(lmList)
        for lm in lmList:
             data.extend([lm[0],height-lm[1],lm[2]])
        print(data)
        # #the data is not in string format so convert them in string

        sock.sendto(str.encode(str(data)), serverAddressPorts)
        # text creation with the data
        handpoints_txt.write(f"[{lm[0]}, {height - lm[1]}, {lm[2]}]")

        curtime = time.time()
        fps = 1 / (curtime - prevtime)
        prevtime=curtime

        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 0, 255), 2, cv2.LINE_AA)
```

```
66
67      cv2.imshow("Hand image", img)
68
69      k = cv2.waitKey(1)
70      # with the "escape" close the window
71      if k % 256 == 27:  # escape
72          # print("close the Window")
73          break
74      if k % 256 == 32:  # space key
75          # print("close the Window")
76          break
77      handpoints_txt.close()
78  cap.release()
    cv2.destroyAllWindows()
    sock.close()
    print('End,the file is saved')
```

Το κύριο μέρος της διαδικασίας ακολουθεί. Εδώ γίνεται η διαδικασία αποκομιδής των οροσήμων, η αποθήκευση τους σε TEXT αρχείο αλλά και η ταυτόχρονη αποστολή τους μέσω UDP πρωτοκόλλου στο πρόγραμμα Blender. Εδώ αξίζει να αναφερθεί ότι η αρχικά έγινε χρήση της Python μέσα από το IDE της Pycharm στην έκδοση 3.7 για λόγους απόδοσης και καλύτερης λειτουργικότητας της βιβλιοθήκης **cv**, ωστόσο τελικά επιλέχθηκε η έκδοση 3.10 που συνεργάζεται με την έκδοση της συγκεκριμένης έκδοσης του προγράμματος Blender (3.2). Η διαδικασία συλλογής των συντεταγμένων των οροσήμων ακολουθεί .

Τα δεδομένα αρχικά των οροσήμων (x, y ,z) πολλαπλασιάζονται με 21, όσα και τα ορόσημα και στην συνέχεια στέλνονται ως μονοδιάστατη λίστα (type dictionary) στην διεύθυνση αποδοχής, η οποία είναι το πρόγραμμα Blender, δηλαδή λειτουργεί ως ο server της εφαρμογής μας. Πρέπει να επισημανθεί εδώ ότι γίνεται μία μετατροπή data.extend([lm[0],**height**-lm[1],lm[2]]) (γραμμή 52 κώδικας) στις συντεταγμένες των οροσήμων (Landmarks) πριν σταλούν. Στην Opencv βιβλιοθήκη η αρχικοποίηση των εικόνων γίνεται θεωρώντας το 0 στο height της εικόνας από την πάνω δεξιά πλευρά (σημείο 0) ενώ το μέγιστο height ορίζεται στην κάτω δεξιά πλευρά της εικόνας, όμως στο Blender ως σημείο αναφοράς 0 στην εικόνα θεωρείται η κάτω αριστερή πλευρά με μέγιστη τιμή height στην εικόνα να είναι η πάνω αριστερή πλευρά. Η μετατροπή αυτή γίνεται ώστε να αντιστραφεί ο y άξονας (height εικόνας) των δεδομένων που αποκτήθηκαν από την αρχική εικόνα και να αποσταλούν στον server σωστά.

Οι συντεταγμένες των σημείων μετά την μετατροπή τους για να σταλούν, αποθηκεύονται σε ένα αρχείο TEXT, και το οποίο ανανεώνεται συνεχώς μέχρι το κλείσιμο της εφαρμογής. (γραμμή 58 κώδικας)

Τα δεδομένα ακολούθως αποστέλλονται στο Blender μέσο του UDP πρωτοκόλλου και επεξεργάζονται ώστε να εξαχθεί το τελικό αποτέλεσμα. (γραμμή 56 κώδικας)

Το πρόγραμμα Blender όπως αναφέρθηκε σε προηγούμενο κεφάλαιο υποστηρίζει την Python. Έχει scripting area στην οποία γράφτηκε ο κώδικας. Αρχικά για λόγους υπολογιστικής μνήμης και απλότητας φτιαχτήκαν 21 **icospheres** που ουσιαστικά προσομοιώνουν τα 21 ορόσημα (landmarks) της εφαρμογής μας. Έγιναν κάποιες ρυθμίσεις όπως smoothing and surface calibration στις **icospheres** για καλύτερη απεικονιστική απόδοση στην περίπτωση zooming της εικόνας.

Πίνακας 4-7:Python στο Blender

```
import bpy
import bpy.types
from mathutils import Vector
import socket
import threading
import atexit


BUFFER_SIZE = 7046
UDP_PORT = 5053
UDP_IP = "127.0.0.1"
printdata = False

# declare global variable
global street_lines
street_lines = []

connections = [(0, 1), (1, 2), (2, 3), (3, 4), (0, 5), (5, 6), (6, 7), (7,
8),
                    (5, 9), (9, 10), (10, 11), (11, 12), (9, 13), (13, 14),
(14, 15), (15, 16),
                    (0, 17), (13, 17), (17, 18), (18, 19), (19, 20)]

icospheres = bpy.data.collections[0].objects

def ico_scale_material():

    material = bpy.data.materials.new(name="IcoMat")
    material.diffuse_color = (0.0, 0.0, 1.0, 1.0)  # blue

    for i in range(21):
        ico = icospheres[i]
        ico.scale = (0.2, 0.2, 0.2)
        ico.active_material = material

def create_line(pos1, pos2):

    # Create a new mesh
    mesh = bpy.data.meshes.new('Line')

    # Define the vertices of the  line
    vertices = [pos1, pos2]

    # Define the edges of the  line
    edges = [(0, 1)]

    # Create the  line object and add it to the scene
    street_line = bpy.data.objects.new('Line', mesh)
    bpy.context.collection.objects.link(street_line)

```

```
    # Set the mesh data of the  line
    mesh.from_pydata(vertices, edges, [])

    # Update the mesh and object
    #mesh.update()

    # Add the line object to the list
    #street_lines.append(street_line)

    return street_line


def create_connections():
     global street_lines

     for line in street_lines:
        bpy.data.objects.remove(line, do_unlink=True)

     street_lines.clear()

     for i in range(21):
         index1, index2 = connections[i]
         line = create_line(icospheres[index1].location,
icospheres[index2].location)
         street_lines.append(line)



def icosheres_3dmove(new_positions):
     global street_lines

     for i in range(21):
         index1, index2 = connections[i]
         ico = icospheres[i]
         ico.location = new_positions[i]

         line1=street_lines[i]
         line1.data.vertices[0].co = icospheres[index1].location
         line1.data.vertices[1].co = icospheres[index2].location

def handleData(data, clientAddress):
    hand_points = []
    points = data
    # handle incoming data here
    for i in range(21):
        x = 7 - float(points[i * 3]) / 100
        y = float(points[i * 3 + 1]) / 100
        z = float(points[i * 3 + 2]) / 100
        hand_points.append((x, y, z))
    return hand_points


```

```
103 def ReceiveData():
104     # Define the server address and port
105     serverAddress = (UDP_IP, UDP_PORT)
106     # Create a UDP socket
107     serverSock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
108     # Bind the socket to the server address and port
109     serverSock.bind(serverAddress)
110     while True:
111         try:
112             handpoints = []
113             # Continuously listen for incoming data
114             data, clientAddress = serverSock.recvfrom(BUFFER_SIZE)
115             data1 = data.decode("utf-8")
116             dataStr = str(data1)
117             points1 = dataStr.replace('[', '')
118             points2 = points1.replace(']', '')
119             points = points2.split(",")
120 
121             handpoints = handleData(points, clientAddress)
122 
123             icosheres_3dmove(handpoints)
124             #bpy.app.handlers.frame_change_post.append(icosheres_3dmove)
125 
126 #bpy.app.handlers.frame_change_post.append(update_lines_positions)
127             if printdata:
128                 print("handle=", handpoints)
129 
130         except Exception as e:
131             print(" The error is:", e)
132 
133     serverSock.close()
134 
135 
136 # Function delete the lines in the scene
137 def cleanup():
138     for obj in bpy.context.scene.objects:
139         if obj.name.startswith("Line"):
140             bpy.data.objects.remove(obj, do_unlink=True)
141 
142 # Function to create a new thread for the  incoming data in a separate
143 thread
144 def Register():
145     # start a new thread to process the data
146     t = threading.Thread(target=ReceiveData)
147     t.start()
148     # print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")
149     bpy.app.handlers.frame_change_post.append(icosheres_3dmove)
150 
151 
152 
153 if __name__ == '__main__':
154 
```

```
155     create_connections()

        # Start the data reception thread
        Register()
```

Αξίζει να αναφερθεί ότι δημιουργήθηκαν σε μια λίστα (connections), όλες οι πιθανές συνδέσεις μεταξύ των οροσήμων, που να επιτρέπουν την δημιουργία των οστών (bones) ώστε να απεικονισθεί το χέρι ως σκελετός. (γραμμή 18 κώδικα) .Σημειώνεται ότι ως οστά στον σκελετό του χεριού θεωρούμε τις γραμμές (lines) που συνδέουν τα ορόσημα.

Τα δεδομένα επεξεργάζονται από την παρακάτω συνάρτηση . Η διαδικασία εφαρμόζεται σε κάθε ορόσημο (landmark) ώστε οι τιμές που χρησιμοποιούνται να έχουν αριθμητική παράσταση ως δεκαδικοί (float numbers), και να εμφανίζονται σωστά ως συντεταγμένες στο Blender. (γραμμή 91 κώδικα)

Δημιουργήθηκε επίσης η **ico_scale_material,** συνάρτηση που δίνει το χρώμα στις icospheres. Η επιλογή έγινε σε Μπλε χρώμα (γραμμή 26 κώδικα). Συνεχίζοντας δημιουργήσαμε την συνάρτηση , που ουσιαστικά δημιουργεί , δυναμικά ,την γραμμή (line) ανάμεσα στα ορόσημα (landmarks), ώστε το χέρι να αποτυπώνεται ως σκελετός. Αξίζει να αναφερθεί ότι η επιλογή της συγκεκριμένης μορφής ένωσης έγινε για καθαρά λόγους απλότητας και μειωμένης υπολογιστικής μνήμης. Εύκολα θα μπορούσε να αναβαθμιστεί σε μορφή (Bones) με την εφαρμογή Rigging του Blender ,ή σε οποιαδήποτε άλλη μορφή ένωσης μεταξύ των οροσήμων (Landmarks) που επιθυμεί ο χρήστης.

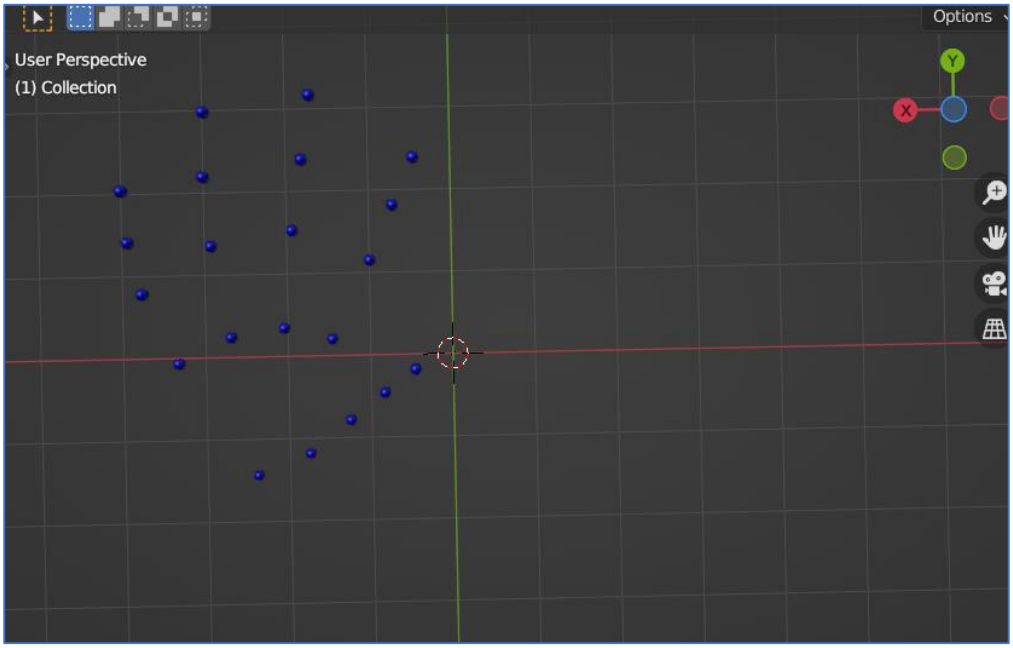
Η συνάρτηση αυτή δημιουργεί τις ενώσεις των icospheres (γραμμή 36 κώδικα) που αντιστοιχούν στα ορόσημα (landmarks). Αφού, αρχικά εξετάζει την ύπαρξη παλιών συνδέσεων ώστε να αποφευκτεί η συνεχόμενη διαδικασία παραγωγής ενώσεων μεταξύ των οροσήμων (Landmarks) αποσυνδέει τις παλιές ενώσεις, εάν υπάρχουν, και δημιουργεί καινούργιες σε πραγματικό χρόνο. (γραμμή 63 κώδικα)

Τα σημεία με τις νέες συντεταγμένες αποδίδονται στις icospheres σε πραγματικό χρόνο. Ταυτόχρονα οι συνδέσεις (lines) ανανεώνουν τις συνεχώς τις θέσεις τους μεταξύ των icospheres, ώστε να δημιουργηθεί το τελικό αποτύπωμα του χεριού. (γραμμή 79 κώδικα)

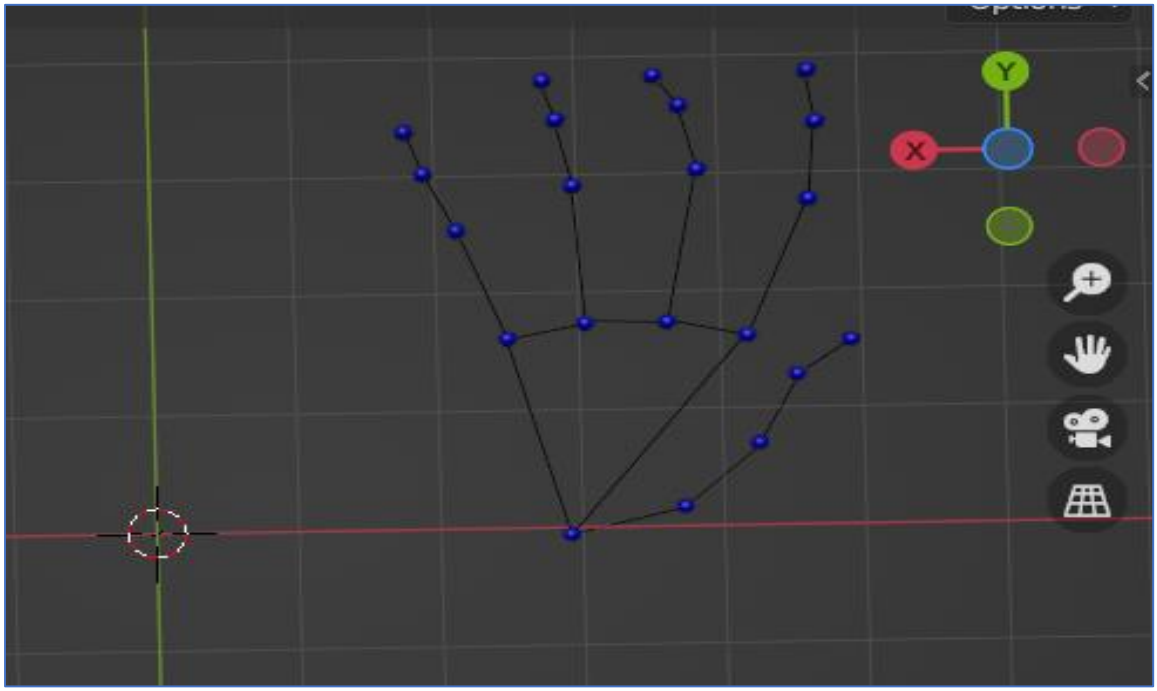
Όλα τα παραπάνω βρίσκονται μέσα στο **while True**: της συνάρτησης **ReceiveData()** που ουσιαστικά δέχεται τα δεδομένα μέσο του UDP πρωτοκόλλου σε πραγματικό χρόνο τα επεξεργάζεται ,αφού η μεταφορά τους γίνεται σε μορφή byte ,τα μετατρέπει σε μορφή string μονοδιάστατης λίστας , που την σειρά της την μετατρέπει σε νέα λίστα (float numbers) και αποδίδει τελικά τις τιμές στα ορόσημα (landmarks) που εμφανίζονται ως icospheres (γραμμή 103 κώδικα)

Η συνάρτηση Register() με την βοήθεια του t = threading.Thread(target=ReceiveData), εκτελείται στο κυρίως πρόγραμμα που επίσης ανανεώνει τα δεδομένα για κάθε καρέ (frame) της εικόνας ώστε να εξαχθεί το τελικό αποτέλεσμα του Animation του χεριού. Αναφορά επίσης, πρέπει να γίνει για την χρήση της Socket βιβλιοθήκης της Python που ουσιαστικά είναι υπεύθυνη για την δημιουργία της σύνδεσης μέσω UDP πρωτοκόλλου αλλά και την μεταφορά των δεδομένων (γραμμή 4, 114 κώδικα)

Η συνάρτηση **threading.Thread(target=)** (γραμμή 146 κώδικα) στην Python χρησιμοποιείται για τη δημιουργία ενός νέου νήματος εκτέλεσης, το οποίο μπορεί να εκτελείται ταυτόχρονα με το κύριο νήμα του προγράμματος. Η διαχείριση των εισερχόμενων συνδέσεων συνεχίζεται ώστε το κύριο νήμα του προγράμματος συνεχίζει να ακούει για νέες συνδέσεις. Αυτό συμβάλει στη βελτίωση της απόδοσης του προγράμματος, καθώς επιτρέπει την ταυτόχρονη διαχείριση πολλαπλών συνδέσεων. Το κυρίως πρόγραμμα τρέχει με τελικό αποτέλεσμα, το animation των οροσήμων (landmarks) σε κάθε καρέ της εικόνας του video.Εικόνα 4-1



Εικόνα 4-1:Το χέρι όπως εμφανίζεται στην οθόνη του Blender χωρίς τις συνδέσεις.



Εικόνα 4-2: Το χέρι με τις συνδέσεις μεταξύ των Landmarks

5. Χρήσεις και μελλοντικές επεκτάσεις

Σε αυτό το κεφάλαιο θα γίνει μια εκτενής ανάπτυξη των λόγων που οδήγησαν στη δημιουργία της εργασίας με την συμβολή των τεχνολογιών που παρουσιάστηκαν σε προηγούμενο κεφάλαιο, καθώς και οι μελλοντικές επεκτάσεις της σε διάφορους τομείς της επιστήμης καθώς επίσης και της καθημερινής ζωής.

5.1 **Γενικά**

Ως σύνολο δεδομένων λαμβάνονται οι τυπικές εικόνες του χεριού ενός ατόμου που έχουν ληφθεί υπό διάφορες συνθήκες. Οι εικόνες αυτές προέρχονται ως δεδομένα σε πραγματικό χρόνο από μια κάμερα είτε από εξαγωγή τους σε μεταγενέστερο χρόνο. Ο κύριος στόχος είναι η αναγνώριση και η ταξινόμηση τους στη σωστή τους σημασία με τη μέγιστη δυνατή ακρίβεια. Οι άνθρωποι χρησιμοποιούμε χειρονομίες για να αλληλοεπιδρούμε με το περιβάλλον μας. Επικοινωνούμε με χειρονομίες όπως η κίνηση του σώματος (Body Language), έκφραση του προσώπου (Face Gesture) και αλλά και υπόδειξη με τα δάχτυλα κ.α .

Αμέσως λοιπόν, τίθεται το ερώτημα : Ποιος είναι ο λόγος , η αιτία, για αυτή την εργασία; Ποια άμεση ή μελλοντική χρήση είναι ικανή στην μορφή αυτή ή σε μια εξέλιξη της να οδηγήσει αυτή η εργασία;

5.2 Υγειονομική περίθαλψη- Ιατρικά συστήματα και υποστηρικτικές τεχνολογίες

Οι αίθουσες επειγόντων περιστατικών και οι χειρουργικές αίθουσες μπορεί να είναι χαοτικές, με πολύ θόρυβο από το προσωπικό και τα μηχανήματα. Σε τέτοια περιβάλλοντα, οι φωνητικές εντολές είναι λιγότερο αποτελεσματικές από τις χειρονομίες. Οι οθόνες αφής δεν αποτελούν επίσης επιλογή, καθώς υπάρχουν αυστηρά όρια μεταξύ του τι είναι και τι δεν είναι αποστειρωμένο. Αλλά η πρόσβαση σε πληροφορίες και εικόνες κατά τη διάρκεια χειρουργικής επέμβασης ή άλλου χειρισμού είναι δυνατή. Παρέχετε στους επιστήμονες υγείας, η δυνατότητα να ελέγχουν μαζί με την χρήση υπολογιστικών συστημάτων εξετάσεις /δεδομένα από μαγνητική αξονική τομογραφία και άλλες εικόνες με απλές χειρονομίες χωρίς να χρειάζεται η επαφή. Ακόμα και άνθρωποι με τα σωματικά μειονεκτήματα, μπορούν να αξιοποιήσουν τα αποτελέσματα της εφαρμογής

Κατά τη διάρκεια κλινικών επεμβάσεων, ένας χειρουργός μπορεί να χρειαστεί λεπτομέρειες σχετικά με ολόκληρη τη δομή του σώματος του ασθενούς ή ένα λεπτομερές μοντέλο οργάνου προκειμένου να συντομεύσει τον χρόνο χειρουργικής επέμβασης ή να αυξήσει την ακρίβεια του αποτελέσματος. Αυτό επιτυγχάνεται με τη χρήση ενός συστήματος ιατρικής απεικόνισης, όπως η μαγνητική τομογραφία, η αξονική τομογραφία ή το σύστημα ακτίνων Χ το οποίο συλλέγει δεδομένα από το σώμα του ασθενούς και τα εμφανίζει στην οθόνη ως λεπτομερή εικόνα. Ο χειρουργός μπορεί να διευκολύνει την αλληλεπίδραση με τις προβαλλόμενες εικόνες εκτελώντας χειρονομίες μπροστά από την κάμερα χρησιμοποιώντας μια τεχνική όρασης υπολογιστή. Αυτές οι χειρονομίες μπορούν να επιτρέψουν ορισμένες λειτουργίες, όπως ζουμ, περιστροφή, περικοπή εικόνας και μετάβαση στην επόμενη ή προηγούμενη διαφάνεια χωρίς τη χρήση περιφερειακής συσκευής, όπως ποντίκι, πληκτρολόγιο ή οθόνη αφής.

Η κλινική διάγνωση σε ασθενής με προβλήματα στον καρπό ή στα δάκτυλα, είναι μια ακόμα εφαρμογή που μπορεί και να πραγματοποιηθεί από απόσταση αρκεί ο ασθενής να διαθέτει μια κάμερα υπολογιστή, πράγμα τόσο συνηθισμένο στην εποχή μας. Επιπλέον, οι χειρονομίες μπορούν να χρησιμοποιηθούν για βοηθητικούς σκοπούς, όπως ο έλεγχος αναπηρικού αμαξιδίου.

5.3 Έλεγχος Robot

Η τεχνολογία των ρομπότ χρησιμοποιείται σε πολλούς τομείς εφαρμογών, όπως η βιομηχανία, οι υποστηρικτικές υπηρεσίες ,ο αθλητισμός και η ψυχαγωγία. Τα ρομποτικά συστήματα ελέγχου χρησιμοποιούν τεχνικές μηχανικής μάθησης (NN), τεχνητή νοημοσύνη (AI) και πολύπλοκους αλγορίθμους για την εκτέλεση μιας συγκεκριμένης εργασίας, η οποία επιτρέπει στο ρομποτικό σύστημα να αλληλοεπιδράσει φυσικά με το περιβάλλον και να λάβει μια ανεξάρτητη απόφαση. Η τεχνολογία της όρασης υπολογιστών με ένα ρομπότ μπορεί να χρησιμοποιηθεί για την κατασκευή βοηθητικών συστημάτων για ηλικιωμένους.

5.4 **Εικονική πραγματικότητα/Gaming**

Από τον έλεγχο ενός υπολογιστή μέχρι τον έλεγχο οικιακών συσκευών και συσκευών για άτομα με σωματικές αναπηρίες ή/και ηλικιωμένους χρήστες με μειωμένη κινητικότητα μέχρι χειρονομίες στην εικονική πραγματικότητα. Η αναγνώριση χειρονομιών σε 3D διαστάσεις χρησιμοποιώντας μια κάμερα smartphone ή ενός υπολογιστή μπορεί να εφαρμοστεί σε περιβάλλοντα που οι περιπτώσεις χρήσης αυτής περιλαμβάνουν παιχνίδια, ηλεκτρονικά είδη ευρείας κατανάλωσης και ρομπότ. Από την άλλη, τα μεγάλα δεδομένα μέσα από τον χειρισμό εικόνων υψηλής ποιότητας μέσω διαισθητικών ενεργειών επωφελούνται από την τρισδιάστατη αλληλεπίδραση όπως αυτή ανθρώπου-ρομπότ είναι μια άλλη εφαρμογή όπου το κύριο κίνητρο για τα συστήματα που βασίζονται σε χειρονομίες είναι η επικοινωνία αυτή να μοιάζει όσο το δυνατόν περισσότερο με τον φυσικό ανθρώπινο διάλογο.

Τέλος, οι χειρονομίες παρέχουν μια πηγή εκφραστικότητας όταν βυθίζονται σε ρεαλιστικά βιντεοπαιχνίδια. Ορισμένες αξιοσημείωτες τεχνολογίες (όπως το Microsoft Kinect, το Sony PSP και τα Nintendo DS και Wii) περιλαμβάνουν αναγνώριση χειρονομιών στις κονσόλες τους. Δυστυχώς, μέχρι στιγμής αναγνωρίζονται μόνο δυναμικές χειρονομίες (όπως το κούνημα και το χτύπημα της γροθιάς)

Τα ηλεκτρονικά παιχνίδια αποτελούν ένα ιδιαίτερα τεχνολογικά υποσχόμενο και εμπορικά αποδοτικό πεδίο για καινοτόμες διεπαφές, λόγω του ψυχαγωγικού χαρακτήρα της αλληλεπίδρασης. Οι χρήστες είναι πρόθυμοι να δοκιμάσουν νέα παραδείγματα διασύνδεσης, δεδομένου ότι είναι πιθανό να βυθίζονται σε ένα δύσκολο περιβάλλον που μοιάζει με παιχνίδι. Σε μια συσκευή πολλαπλής αφής, ο έλεγχος παρέχεται μέσω των δακτύλων του χρήστη. Το ποιο

δάχτυλο αγγίζει την οθόνη δεν έχει σημασία- το πιο σημαντικό είναι το σημείο που γίνεται η αφή και ο αριθμός των δαχτύλων που χρησιμοποιούνται.

5.5 Οικιακός αυτοματισμός

Ο οικιακός αυτοματισμός είναι ένας άλλος ευρύς τομέας στον τομέα των ηλεκτρονικών ειδών ευρείας κατανάλωσης στον οποίο χρησιμοποιείται η αναγνώριση χειρονομιών. Για παράδειγμα, φανταστείτε πόσο διαισθητικό θα ήταν να χρησιμοποιείτε χειρονομίες για να λέτε σε ένα ρομπότ τι να κάνει ή πού να πάει. Δείχνοντας ένα σημείο σκόνης, οι χρήστες θα μπορούν να οδηγούν μια ρομποτική σκούπα στον να καθαρίσει καθαρισμό. Το λογισμικό θα μπορούσε να χρησιμοποιηθεί στο να κάνει τις έξυπνες τηλεοράσεις να αντιλαμβάνονται τις κινήσεις των δακτύλων και τις χειρονομίες, ή να προσφέρει έλεγχο χωρίς επαφή σε συστήματα φωτισμού και ήχου. Το κούνημα του χεριού ή η εκτέλεση κάποιας χειρονομίας μπορεί εύκολα να επιτρέψει τον έλεγχο του φωτισμού, των ανεμιστήρων, της τηλεόρασης, του ραδιοφώνου κ.λπ.

5.6 Αναγνώριση νοηματικής γλώσσας

Η νοηματική γλώσσα είναι μια εναλλακτική μέθοδος που χρησιμοποιείται από άτομα που δεν μπορούν να επικοινωνήσουν με άλλους με την ομιλία. Αποτελείται από ένα σύνολο χειρονομιών, όπου κάθε χειρονομία αντιπροσωπεύει ένα γράμμα, έναν αριθμό ή μια έκφραση. Πολλές ερευνητικές εργασίες έχουν προτείνει την αναγνώριση της νοηματικής γλώσσας για κωφάλαλους, χρησιμοποιώντας έναν αισθητήρα που φοριέται στο χέρι και δίνει απαντήσεις ανάλογα με την κίνηση του χεριού. Εναλλακτικά, μπορεί να περιλαμβάνει ακάλυπτη αλληλεπίδραση του χεριού με την κάμερα, χρησιμοποιώντας τεχνικές όρασης υπολογιστή για τον εντοπισμό της χειρονομίας. Και για τις δύο προαναφερθείσες προσεγγίσεις, το σύνολο δεδομένων που χρησιμοποιείται για την ταξινόμηση των χειρονομιών ταιριάζει με μια χειρονομία που κάνει ο χρήστης σε πραγματικό χρόνο.

5.7 Προσωπικός υπολογιστής και Tablet

Οι χειρονομίες μπορούν να χρησιμοποιηθούν ως εναλλακτική συσκευή εισόδου που επιτρέπει την αλληλεπίδραση με τον υπολογιστή χωρίς ποντίκι ή πληκτρολόγιο, όπως το «σύρσιμο» scrolling,

η απόθεση και η μετακίνηση αρχείων στο περιβάλλον της επιφάνειας εργασίας, καθώς και οι λειτουργίες αποκοπής και επικόλλησης. Επιπλέον, μπορούν να χρησιμοποιηθούν για τον έλεγχο παρουσιάσεων διαφανειών Επιπλέον, χρησιμοποιούνται με ένα tablet για να επιτρέπουν σε κωφάλαλους ανθρώπους να αλληλοεπιδρούν με άλλους ανθρώπους μετακινώντας το χέρι τους μπροστά από την κάμερα του tablet..

5.8 **Virtual Music**

Στην σύγχρονη εποχή μας μια μέθοδος εντοπισμού και αναγνώρισης τρισδιάστατων χειρονομιών δακτύλων θα μπορούσε να αξιοποιηθεί πάντα με χρήση αισθητήρων βάθους και για τα δύο χέρια για την αναπαραγωγή μουσικής σε πραγματικό χρόνο. Οι τελικοί χρήστες δεν θα πρέπει πλέον να κουβαλούν τίποτα στα χέρια τους. Υλοποιώντας τρισδιάστατα όργανα που να βασίζονται σε χειρονομίες, χρειάζεται ακριβή παρακολούθηση των δακτύλων σε 3D περιβάλλον . Ο εντοπισμός των χεριών (αριστερό ,δεξί), όπου η ανίχνευση των χειρονομιών των δακτύλων, γίνεται με χρήση νευρωνικού δικτύου (NN) για την ανίχνευση των ειδικών χειρονομιών, έτσι ώστε να μπορούν να αναγνωριστούν με ακρίβεια οι λεπτομερείς χειρονομίες των δακτύλων.

6. Συμπεράσματα

Παρουσιάσαμε μια μέθοδο ανίχνευσης του χεριού αλλά και αναγνώρισης της χειρονομίας του, που βασίζεται σε τεχνικές υπολογιστικής όρασης, σε μια εφαρμογή που λειτουργεί σε πραγματικό χρόνο με μια συνηθισμένη κάμερα. Η μέθοδος συνδυάζει την ανίχνευση οροσήμων (landmarks), τον υπολογισμό των συντεταγμένων των σημείων μέσα από την εικόνα του video Capture και την μεταφορά των δεδομένων σε πρόγραμμα δημιουργίας 3D animation ,του χεριού, με συλλογιστική βάσει κανόνων της αποτύπωσης των κινήσεων σε πραγματικό χρόνο. Είναι σαφές ότι πρέπει να καταβληθεί μεγάλη προσπάθεια για την ανάπτυξη αξιόπιστων και εύρωστων αλγορίθμων με τη βοήθεια της χρήσης ενός αισθητήρα κάμερας που έχει ένα συγκεκριμένο χαρακτηριστικό για την αντιμετώπιση κοινών προβλημάτων και την επίτευξη ενός αξιόπιστου αποτελέσματος.

Όσον αφορά το ερευνητικό ερώτημα, το αντίκτυπο στη δημόσια υγεία, μπορούμε να συμπεράνουμε ότι υπάρχει ενδιαφέρον για την ανάπτυξη και τη βελτίωσή της, στην υποστήριξη της υγείας των πληθυσμών και αλλά και του περιβάλλοντος. Υπάρχει ποικιλία στην χρήση των ερευνητικών προσπαθειών ,όπως παρουσιαστήκαν παραπάνω. Παρόλο που ο χειροκίνητος έλεγχος των πραγμάτων είναι πιο φυσικός, ευκολότερος, πιο ευέλικτος και φθηνότερος, ενώ δεν υπάρχει ανάγκη να διορθώνονται προβλήματα που προκαλούνται από συσκευές υλικού, ωστόσο η αναγνώριση και ανάλυση χειρονομιών αλλά και η κίνηση του χεριού στα συστήματα αλληλεπίδρασης και διεπαφής ανθρώπου-μηχανής σίγουρα θα βοηθήσει. Η αναγνώριση χειρονομιών σε πραγματικό χρόνο για την αλληλεπίδραση με τον υπολογιστή είναι απλώς το επόμενο βήμα στην τεχνολογική εξέλιξη και είναι ιδανική για το σημερινό καταναλωτικό τοπίο.

Η εφαρμογή μπορεί και πρέπει να αναπτυχθεί περεταίρω. Βελτιώσεις μπορούν να γίνουν σε επίπεδο κώδικα και User-Interface. Η χρήση καλύτερων και ποιο γρήγορων λειτουργικών συστημάτων με μεγαλύτερη υπολογιστική ισχύ θα ήταν μια αρχή.

Εκτός από τη χρήση χειρονομιών όταν δεν μπορούμε να αγγίξουμε τον εξοπλισμό, η παρακολούθηση χεριών μπορεί να εφαρμοστεί σε περιβάλλοντα επαυξημένης και εικονικής πραγματικότητας, στην αναγνώριση νοηματικής γλώσσας, σε παιχνίδια και σε άλλες περιπτώσεις χρήσης. Το υψηλό κόστος των προϊόντων ανίχνευσης χωρίς επαφή είναι μία από τις

σημαντικότερες προκλήσεις αυτής της τεχνολογίας, μαζί με την πολυπλοκότητα της ανάπτυξης λογισμικού. Για τη δημιουργία ενός αξιόπιστου συστήματος που ανιχνεύει τις θέσεις των χεριών, μια λύση εντοπισμού χεριών απαιτεί, μεταξύ άλλων, την εφαρμογή προηγμένων αλγορίθμων μηχανικής μάθησης και βαθιάς μάθησης.

7. Βιβλιογραφία

1. Real-Time Hand Gesture Replication System using 3D Modelling Software [Akshita Gupta, Abhinav Shukla, Rahul Yadav, Shubham Mishra, Mamoon Rashid, Sachin Kumar Gupta]

2. Artificial Reality II by Myron K. Krueger (1991-06-04)
Published June 14th 1991 by Addison-Wesley Professional (first published May 1991) ISBN 0201522608 English

3. Shai Luria, MD, Jay T. Bridgeman, MD, DDS, και Thomas E. Trumble, MD

4. Moore, K. L., Dalley, A. F., & Agur, A. M. R. (2014). Clinically Oriented Anatomy (7th ed.). Philadelphia, PA: Lippincott Williams & Wilkins.

5. Netter, F. (2019). Atlas of Human Anatomy (7th ed.). Philadelphia, PA: Saunders. Standring, S. (2016). Gray's Anatomy (41st ed.). Edinburgh: Elsevier Churchill Livingstone.

6. Tubbs, R. S., Shoja, M. M., Loukas, M., & Bergman, R. A. (2016). Bergman's comprehensive encyclopedia of human anatomic variation. Hoboken: Wiley Blackwell

7. Prediction of fingers posture using artificial neural networks: Visual Hand Recognition in Hand Laterality and Self-Other Discrimination Tasks: Relationships to Autistic Traits

8. Wachs, J.P.; Kölsch, M.; Stern, H.; Edan, Y. Vision-based hand-gesture applications. Commun. ACM 2011, 54, 60–71.

9. Zeng, J.; Sun, Y.; Wang, F. A natural hand gesture system for intelligent human-computer interaction and medical assistance. In Proceedings of the 2012 Third Global Congress on Intelligent Systems, Wuhan, China, 6–8 November 2012; pp. 382–385.

10. Gallo, L.; Placitelli, A.P.; Ciampi, M. Controller-free exploration of medical image data: Experiencing the Kinect. In Proceedings of the 2011 24th international symposium on computer-based medical systems (CBMS), Bristol, UK, 27–30 June 2011; pp. 1–6

11. Pansare, J.R.; Gawande, S.H.; Ingle, M. Real-time static hand gesture recognition for American Sign Language (ASL) in complex background. JSIP 2012, 3, 22132.

12. Kulkarni, V.S.; Lokhande, S.D. Appearance based recognition of American sign language using gesture segmentation. Int. J. Comput. Sci. Eng. 2010, 2, 560–565

13. Lee, D.-H.; Hong, K.-S. Game interface using hand gesture recognition. In Proceedings of the 5th International Conference on Computer Sciences and Convergence Information Technology, Seoul, South Korea, 30 November–2 December 2010; pp. 1092–1097

14. Kaur, H.; Rani, J. A review: Study of various techniques of Hand gesture recognition. In Proceedings of the 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), Delhi, India, 4–6 July 2016; pp. 1–5.

15. Zhao, X.; Naguib, A.M.; Lee, S. Kinect based calling gesture recognition for taking order service of elderly care robot. In Proceedings of the The 23rd IEEE international symposium on robot and human interactive communication, Edinburgh, UK, 25–29 August 2014; pp. 525–530.

16. Van den Bergh, M.; Carton, D.; De Nijs, R.; Mitsou, N.; Landsiedel, C.; Kuehnlenz, K.; Wollherr, D.; Van Gool, L.; Buss, M. Real-time 3D hand gesture interaction with a robot for understanding directions from humans. In Proceedings of the 2011 Ro-Man, Atlanta, GA, USA, 31 July–3 August 2011; pp. 357–362

17. Thomas, J.J. and Cook, K.A., Eds. Illuminating the Path: The Research and Development Agenda for Visual Analytics. IEEE CS Press, 2005.

18. Krahnstoever, N., Kettebekov, S., Yeasin, M., and Sharma, R. A real-time framework for natural multimodal interaction with large-screen displays. In *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces* (Pittsburgh, PA Oct. 1416). IEEE Computer Society, Washington, D.C., 2002, 349.

19. Lee U., Tanaka J. Finger identification and hand gesture recognition techniques for natural user interface; Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction; Bangalore, India. 24–27 September 2013; pp. 274–279.

20. Taylor J., Bordeaux L., Cashman T., Corish B., Keskin C., Sharp T., Soto E., Sweeney D., Valentin J., Luff B. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. ACM Trans. Graph. 2016;35:1–12. doi: 10.1145/2897824.2925965.

21. Thomas, J.J. and Cook, K.A., Eds. Illuminating the Path: The Research and Development Agenda for Visual Analytics. IEEE CS Press, 2005.

22. Krahnstoever, N., Kettebekov, S., Yeasin, M., and Sharma, R. A real-time framework for natural multimodal interaction with large-screen displays. In Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces (Pittsburgh, PA Oct. 1416). IEEE Computer Society, Washington, D.C., 2002, 349.

23. Bolt, R.A. 'Put-That-There': Voice and gesture at the graphics interface. In Proceedings of the Seventh International Conference on Computer Graphics and Interactive Techniques. ACM Press, New York, 1980, 262270.

24. 3D finger tracking and recognition image processing for real-time music playing with depth sensors Enkhtogtokh Togootogtokh, Timothy K. Shih, W. G. C. W. Kumara, Shih-Jung Wu, Shih-Wei Sun & Hon-Hang Chang Multimedia Tools and Applications volume 77, pages9233–9248 (2018)

25. Kim, M.-S.; Lee, C.H. Hand Gesture Recognition for Kinect v2 Sensor in the Near Distance Where Depth Data Are Not Provided. Int. J. Softw. Eng. Its Appl. 2016, 10, 407–418

26. Bamwenda, J.; Özerdem, M.S. Recognition of Static Hand Gesture with Using ANN and SVM. Dicle Univ. J. Eng. 2019, 10, 561–568.

27. Hand Gesture Recognition using Image Processing and Feature Extraction Techniques January 2020 Procedia Computer Science 173:181-190 DI:10.1016/j.procs.2020.06.022

28. Accurate Hand Detection from Single-Color Images by Reconstructing Hand Appearances Chi Xu Wendi Cai Yongbo Li Jun Zhou Longsheng Wei Affiliations expand PMID: **31905746** PMCID: PMC6982909 DOI: 10.3390/s20010192

29. Finger Recognition for Hand Pose Determination November 2009 DOI:10.1109/ICSMC.2009.5346342 Source IEEE Xplore Conference: Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference

30. Desai, S.; Desai, A. Human Computer Interaction through hand gestures for home automation using Microsoft Kinect. In Proceedings of the International Conference on Communication and Networks, Xi'an, China, 10–12 October 2017; pp. 19–29

31. Mubashar KA, Umar W, Choudhary T, Hussain F, Haroon YM (2013) A new algorithmic approach for fingers detection and identification. In Proc of SPIE 8768:1–6

32. Li Y, (2012) Hand gesture recognition using Kinect. In: Software Engineering and Service Science (ICSESS), 2012 I.E. 3rd International Conference on IEEE, pp. 196–199

33. Bergh MV, Koller-Meier E, Bosché F, Gool LV (2009) Haarlet-based hand gesture recognition for 3D interaction, In: 2009 Workshop on Applications of Computer Vision (WACV), pp. 1–8

34. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques Munir Oudah Ali Al-Naji and Javaan Chahl J Imaging. 2020 Aug; 6(8): 73. Published online 2020 Jul 23. doi: 10.3390/jimaging6080073

35. Real-Time Hand Gesture Recognition Using Finger Segmentation Zhi-hua Chen 1, Jung-Tae Kim 1, Jianning Liang 1, Jing Zhang 2, Yu-Bo Yuan 1 Affiliations expand PMID: **25054171** PMCID: PMC4099175 DOI: 10.1155/2014/267872

36. https://www.jetbrains.com/pycharm/

37. https://www.computervision.zone/

38. https://www.python.org/downloads/

39. https://docs.python.org/3/

40. https://www.blender.org/download/

41. https://raw.githubusercontent.com/madhav727

42. https://www.geeksforgeeks.org/python-programming-language/?ref=shm

43. https://stackoverflow.com/questions/

44. https://google.github.io/mediapipe/solutions/hands.html

45. https://www.w3schools.com/

46. https://www.grepper.com/