ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εντοπισμός εμποδίων με αξιοποίηση δυνατοτήτων υπολογισμού βάθους Επαυξημένης Πραγματικότητας για κινητές συσκευές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ορέστης Ζάρας

Επιβλέπων : Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2024

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εντοπισμός εμποδίων με αξιοποίηση δυνατοτήτων υπολογισμού βάθους Επαυξημένης Πραγματικότητας για κινητές συσκευές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ορέστης Ζάρας

Επιβλέπων : Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 17/7/2024

.....	.....	.....
Τσανάκας Παναγιώτης	Γεώργιος Ματσόπουλος	Ηλίας Μαγκλογιάννης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής ΠΑ.ΠΕΙ.

Αθήνα, 2024

.....

Ορέστης Ζάρας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στη σύγχρονη εποχή, η Επαυξημένη Πραγματικότητα (AR) αποτελεί μια ευρέως διαδεδομένη τεχνολογία που χρησιμοποιείται από πληθώρα φορητών και μη συσκευών. Οι συνεχείς ερευνητικές εξελίξεις στον τομέα αυτό ξεκλειδώνουν συνεχώς νέες δυνατότητες χρήσης της, ενώ παράλληλα βελτιώνουν την αποδοτικότητα των ήδη υφιστάμενων εφαρμογών. Η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη μιας εφαρμογής που εκμεταλλεύεται τις δυνατότητες υπολογισμού βάθους της Επαυξημένης Πραγματικότητας για τον εντοπισμό εμποδίων και γκρεμών στο περιβάλλον του χρήστη. Στόχος της εφαρμογής είναι να παρέχει ένα εργαλείο το οποίο μπορεί να χρησιμοποιηθεί σε πραγματικό χρόνο, αυξάνοντας την ασφάλεια και διευκολύνοντας την κίνηση σε διάφορα περιβάλλοντα.

Αρχικά, θα παρουσιαστεί το τεχνολογικό υπόβαθρο που απαιτείται για τη δημιουργία μιας εφαρμογής Επαυξημένης Πραγματικότητας. Σε αυτό το πλαίσιο, θα αναλυθούν οι βασικές αρχές που διέπουν τη λειτουργία μιας τέτοιας εφαρμογής και θα αναφερθούν σχετικά έργα για να προσφέρουν ένα συνολικό υπόβαθρο. Στη συνέχεια, θα εξηγηθεί λεπτομερώς ο Αλγόριθμος που χρησιμοποιεί η εφαρμογή, υπογραμμίζοντας την επαναληπτική του φύση και τον τρόπο εκτέλεσής του. Επιπροσθέτως, θα παρουσιαστεί η Αρχιτεκτονική του συστήματος που αναπτύχθηκε, περιγράφοντας τη λειτουργία κάθε επιμέρους τμήματος. Ακολούθως, θα αναφερθούν τα τεχνολογικά εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του κώδικα, και θα αναλυθεί η δομή του. Τέλος, θα εξεταστούν οι παράμετροι που ορίστηκαν κατά την ανάπτυξη της εφαρμογής και θα παρουσιαστεί η έρευνα που διεξήχθη για την εύρεση των βέλτιστων τιμών για κάθε παράμετρο.

Λέξεις κλειδιά

Εντοπισμός εμποδίων, Επαυξημένη Πραγματικότητα, βάθος, αισθητήρες, κάμερα, Android, Java, ARCore, Sceneform, μετρικές σύγκρισης

Abstract

Nowadays, augmented reality (AR) is a widely used technology employed by a plethora of portable and non-portable devices. Ongoing research developments in this field continually unlock new possibilities for its use while also improving the efficiency of existing applications. This thesis focuses on the development of an application that exploits the depth computing capabilities of augmented reality to detect obstacles and cliffs in the user's environment. The goal of the application is to provide a tool that can be used in real-time, enhancing safety and facilitating movement in various environments.

Initially, the technological background required for creating an augmented reality application will be presented. In this context, the fundamental principles governing the operation of such an application will be analyzed, and related work will be mentioned to provide an overall background. Next, the algorithm used by the application will be explained in detail, highlighting its iterative nature and the way it is executed. Following this, the architecture of the developed system will be presented, describing the function of each component. Then, the technological tools used for code development will be mentioned, and its structure will be analyzed. Finally, the parameters set during the application's development will be examined, and the research conducted to find the optimal values for each parameter will be presented.

Keywords

Obstacle detection, augmented reality, depth, sensors, camera, Android, Java, ARCore, Sceneform, comparison metrics

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Παναγιώτη Τσανάκα για την ανάθεση της διπλωματικής αυτής. Επίσης θα ήθελα να ευχαριστήσω τον συμμετέχοντα στην επίβλεψη, κ. Διονύσιο Κουλούρη για την βοήθεια και τις συμβουλές που μου πρόσφερε κατά την ανάπτυξη της εργασίας.

Στην συνέχεια θα ήθελα να ευχαριστήσω τον πατέρα μου Βασίλειο και τις αδερφές μου Μιρέλλα, Κατερίνα και Δανάη για τη στήριξή τους.

Τέλος, αφιερώνω αυτή την διπλωματική εργασία στη μητέρα μου, Αθανασία που απεβίωσε νωρίς.

Πίνακας Περιεχομένων

Λίστα Εικόνων

Λίστα Πινάκων

Κεφάλαιο 1
Εισαγωγή

Τα τελευταία χρόνια, η Επαυξημένη Πραγματικότητα έχει εισέλθει στο χώρο της τεχνολογίας, συμβάλλοντας έτσι στην επικοινωνία ανθρώπου και μηχανής. Αυτή η τεχνολογία συνδυάζει τον φυσικό κόσμο με τον ψηφιακό, προσφέροντας μια άμεση και βαθιά εμπειρία στους χρήστες, από εκπαιδευτικές εφαρμογές έως καλλιτεχνικές δημιουργίες και επαγγελματικές εφαρμογές. Τι είναι όμως η Επαυξημένη Πραγματικότητα;

1.1 Ορισμός Επαυξημένης Πραγματικότητας

Επαυξημένη Πραγματικότητα[1] είναι η σε πραγματικό χρόνο άμεση ή έμμεση θέαση ενός φυσικού, πραγματικού περιβάλλοντος, του οποίου τα στοιχεία επαυξάνονται από στοιχεία αναπαραγόμενα από συσκευές υπολογιστών, όπως ήχος, βίντεο, γραφικά ή δεδομένα τοποθεσίας.

Η Επαυξημένη Πραγματικότητα (Augmented Reality - AR)[2] στοχεύει στο να απλοποιήσει τη ζωή του χρήστη φέρνοντας εικονικές πληροφορίες όχι μόνο στον άμεσο περίγυρό του, αλλά και σε οποιαδήποτε έμμεση προβολή του πραγματικού περιβάλλοντος, όπως το live-video stream. Ενισχύει την αντίληψη και την αλληλεπίδραση του χρήστη με τον πραγματικό κόσμο. Σε αντίθεση με την Εικονική Πραγματικότητα (Virtual Reality - VR), η οποία προσομοιώνει ένα συνθετικό κόσμο, η Επαυξημένη Πραγματικότητα δείχνει στον χρήστη τον πραγματικό κόσμο και προβάλλει σε αυτόν εικονικά αντικείμενα σε πραγματικό χρόνο.

1.2 Άτομα με έλλειψη όρασης

Σύμφωνα με την Παγκόσμια Ένωση Τυφλών (World Blind Union) υπάρχουν περίπου 253 εκατομμύρια άτομα που είναι τυφλά ή με περιορισμένη όραση[3]. Η έλλειψη όρασης συνοδεύεται συνήθως από έλλειψη ανεξαρτησίας, καθώς τέτοια άτομα δε μπορούν να χρησιμοποιήσουν διάφορες υπηρεσίες και έχουν περιορισμένη κινητικότητα. Οι περισσότεροι τυφλοί και μερικώς τυφλοί άνθρωποι μαθαίνουν να χρησιμοποιούν την ακοή τους για να αντισταθμίσουν την έλλειψη όρασης. Τα περιβαλλοντικά σήματα επιτρέπουν στους ανθρώπους να μαθαίνουν για πηγές και ήχους από το περιβάλλον, όπως φανάρια, κίνηση, θορύβους από μηχανήματα, ζώα ή άνθρωποι, κλπ. Οι τυφλοί άνθρωποι κάνουν μέγιστη χρήση του ήχου ώστε να διατηρήσουν την ασφάλειά τους.

1.2 Στόχος

Ο στόχος της διπλωματικής εργασίας αυτής, είναι η δημιουργία μιας εφαρμογής που χρησιμοποιεί τεχνολογίες Επαυξημένης Πραγματικότητας για τον εντοπισμό εμποδίων ή γκρεμών στο μονοπάτι του χρήστη και την έγκαιρη ενημέρωσή του μέσω ήχου. Παρουσιάζονται ο Αλγόριθμος, η Αρχιτεκτονική του συστήματος και οι παράμετροι που απαιτούνται για τη σωστή λειτουργία του. Γίνεται επίσης ανάλυση των παραμέτρων μέσω δοκιμών για την εύρεση της κατάλληλης τιμής για την κάθε μια.

Το υπόλοιπο της παρούσας εργασίας είναι δομημένο ως ακολούθως:

Στο Κεφάλαιο δύο (2) γίνεται εισαγωγή στον τρόπο λειτουργίας των εφαρμογών Επαυξημένης Πραγματικότητας και αναφέρονται τα συστήματα που χρησιμοποιούνται από αυτή. Δίνονται επίσης παραδείγματα σχετικών εργασιών.

Στο Κεφάλαιο τρία (3) αναλύεται περιγραφικά και με ψευδοκώδικα ο Αλγόριθμος που ακολουθεί η εφαρμογή κατά τη λειτουργία της και δίνεται το διάγραμμα ροής του.

Στο Κεφάλαιο τέσσερα (4) παρατίθεται η Αρχιτεκτονική του συστήματος. Εκεί εξηγείται η λειτουργία κάθε κομματιού σε αυτή καθώς και ο τρόπος με τον οποίο συνδέονται μεταξύ τους.

Στο Κεφάλαιο (5) αναφέρονται οι τεχνολογίες που χρησιμοποιήθηκαν για τη δημιουργία της εφαρμογής, δηλαδή η συσκευή και το λειτουργικό της σύστημα, η γλώσσα προγραμματισμού, το περιβάλλον στο οποίο αναπτύχθηκε η εφαρμογή και η βιβλιοθήκη που εισήχθη για τις λειτουργίες Επαυξημένης Πραγματικότητας.

Στο Κεφάλαιο έξι (6) παρατίθεται ο σκελετός του κώδικα της εφαρμογής και εξηγούνται οι συναρτήσεις και η λειτουργία τους.

Στο Κεφάλαιο επτά (7) εξηγείται η λειτουργία της εφαρμογής στην πράξη. Δίνονται οδηγίες για την εγκατάστασή της καθώς και τα σενάρια στα οποία υποτίθεται ότι αυτή θα τρέχει και εξηγούνται οι οθόνες της. Έπειτα αναφέρονται οι μετρικές που χρησιμοποιήθηκαν για τη σύγκριση των τιμών της κάθε παραμέτρου. Τέλος εξηγείται η λειτουργία της κάθε παραμέτρου και γίνεται ανάλυση των τιμών που χρησιμοποιήθηκαν στις δοκιμές για την κάθε μια.

Στο Κεφάλαιο οκτώ (8) γίνεται η αξιολόγηση της εφαρμογής και δίνονται προτάσεις για μελλοντικές επεκτάσεις της.

2 Τεχνολογικό Υπόβραθρο

2.1 Σχετική βιβλιογραφία

Η Επαυξημένη Πραγματικότητα αποτελεί μια διαρκώς εξελισσόμενη τεχνολογία καλύπτοντας έτσι πολλά πεδία εφαρμογής. Ένα από τα βασικότερα είναι η κατηγορία των παιχνιδιών. Με την ενσωμάτωση της Επαυξημένης Πραγματικότητας, μπορούν να δημιουργηθούν παιχνίδια που επιτρέπουν την άμεση αλληλεπίδραση με τον ψηφιακό και τον πραγματικό κόσμο, επιτρέποντας έτσι στον παίκτη να βυθιστεί στην εμπειρία. Τέτοιο παράδειγμα είναι το ARZombie[12], μια εφαρμογή για φορητές συσκευές που εξασκεί τεχνικές αναγνώρισης προσώπων, στην οποία ο παίκτης πρέπει να εξοντώσει τέρατα (Zombie) που εντοπίζονται και εμφανίζονται στην οθόνη. Παράλληλα με την ψυχαγωγία, η ενσωμάτωση της τεχνολογίας αυτής στα παιχνίδια μπορεί να χρησιμοποιηθεί για να προβάλλει ένα πιο υγιή τρόπο ζωής. Για παράδειγμα, στο άρθρο [13] δημιουργήθηκε ένα παιχνίδι που παροτρύνει το χρήστη να ασκηθεί σωματικά, ενώ ταυτόχρονα επιβλέπει τις λειτουργίες του σώματός του σε πραγματικό χρόνο. Παρόμοια είναι και η ιδέα πίσω από το γνωστό παιχνίδι Pokémon Go! [14]. Τέλος, στο [15] αναπτύχθηκε ένα παιχνίδι με σκοπό την προώθηση των μνημείων στο πανεπιστήμιο Mil. Nueva Granada της Κολομβίας.

Η Επαυξημένη Πραγματικότητα βρίσκει εφαρμογή και στο πεδίο της σχεδίασης και παραγωγής[16], δηλαδή στη διαδικασία μετατροπής πρώτων υλών σε αγαθά και προϊόντα που έχουν κάποια αξία για τον άνθρωπο. Αυτή μπορεί να διασφαλίσει ότι δραστηριότητες όπως ο σχεδιασμός, ο προγραμματισμός και η επεξεργασία θα γίνονται σωστά από την πρώτη φορά, χωρίς την ανάγκη για επαναλαμβανόμενες εργασίες και τροποποιήσεις μειώνοντας έτσι το κόστος της συνολικής διαδικασίας. Στο άρθρο [49] παρουσιάζεται η ανάπτυξη ενός απομακρυσμένου συστήματος για τη διευκόλυνση της μετακίνησης πετρών και άλλων υλικών σε κατασκευές που βρίσκονται κάτω από το νερό. Στο [50] ερευνάται η ενσωμάτωση της Επαυξημένης Πραγματικότητας σε ένα σύστημα με σκοπό την επιθεώρηση και τη συντήρηση υπόγειων σωλήνων. Το [51] μιλά για μια πλατφόρμα παρακολούθησης μηχανών παραγωγής για τον υπολογισμό του χρόνου μεταξύ βλαβών ώστε να γίνει απομακρυσμένη συντήρηση αυτών. Το [52] αναφέρει μια διεπαφή αρωγής των χρηστών στο σχεδιασμό της κίνησης ενός μηχανήματος εφαρμογής κόλλας στον τρισδιάστατο χώρο. Παρόμοια στο [53] αναφέρεται μια μέθοδος σχεδιασμού του μονοπατιού κίνησης ενός ρομπότ για την αποφυγή συγκρούσεων. Στο [54] παρουσιάζεται ένα πείραμα χρήσης της Επαυξημένης Πραγματικότητας για την επιθεώρηση του πάχους των κατασκευασμένων εξαρτημάτων. Τέλος, στο [55] περιγράφεται ένα σύστημα διασκέψεων που χρησιμοποιεί την επικάλυψη εικόνων στον πραγματικό κόσμο.

Παρατηρούνται επίσης εξελίξεις στην Επαυξημένη Πραγματικότητα όταν αυτή αφορά τον ιατρικό κλάδο. Στα άρθρα [17] και [11] αναλύεται ο συνδυασμός της ρομποτικής και της Επαυξημένης Πραγματικότητας για τη δημιουργία ενός περιβάλλοντος που διευκολύνει τόσο τους χειρούργους να εκτελέσουν την επέμβαση, όσο και τους ασθενείς να αναρρώσουν πιο γρήγορα. Τέλος, στο [18] αναπτύσσεται ένα σύστημα που βοηθά στο σχεδιασμό επεμβάσεων που αφορούν την εκτομή εγκεφαλικών όγκων.

2.2 Εφαρμογές Επαυξημένης Πραγματικότητας

Η Επαυξημένη Πραγματικότητα περιλαμβάνει μια σειρά τεχνικών στοιχείων που εμφανίζονται σε ηλεκτρονικές συσκευές για να παρατηρούν, άμεσα ή έμμεσα, φυσικά περιβάλλοντα στον πραγματικό κόσμο και να τα συνδυάζουν με εικονικά στοιχεία[5].

Μια εφαρμογή Επαυξημένης Πραγματικότητας κάνει δύο βασικά βήματα σε κάθε χρονικό βήμα της εκτέλεσής της[4]:

1. Εξετάζει και αξιολογεί την κατάσταση του φυσικού κόσμου και παράγει την ανάλογη κατάσταση του εικονικού κόσμου.
2. Εμφανίζει τον εικονικό κόσμο και τον φυσικό κόσμο με συγκεκριμένο τρόπο ώστε ο χρήστης να αισθάνεται ότι ο εικονικός κόσμος είναι μέρος του πραγματικού και γυρνά στο βήμα 1 για την επόμενη επανάληψη.

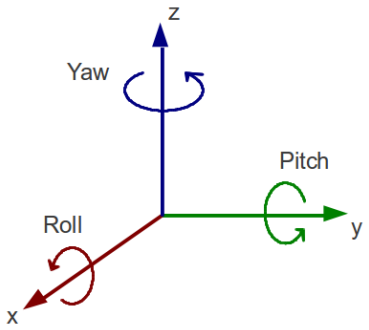
Τα στοιχεία που αποτελούν μέρος ενός συστήματος AR περιλαμβάνουν:

1. Αισθητήρες για την αξιολόγηση της κατάστασης του φυσικού κόσμου στον οποίο βρίσκεται το σύστημα. Τέτοιοι είναι η κάμερα, το GPS, το επιταχυνσιόμετρο, η πυξίδα και το γυροσκόπιο.
2. Μια Κεντρική Μονάδα Επεξεργασίας (CPU) για την αξιοποίηση των δεδομένων των αισθητήρων.
3. Μια οθόνη που χρησιμοποιείται για την προβολή των εικονικών πληροφοριών στις εικόνες που αποκτήθηκαν από τη συσκευή καταγραφής. Υπάρχουν 2 είδη τεχνολογίας οθόνης:[5]
    - Οθόνες με μίξη βίντεο (Video-mixed displays). Αυτές συγχωνεύουν τον πραγματικό κόσμο με τα εικονικά αντικείμενα και τα εμφανίζουν ψηφιακά. Στην εν λόγω εφαρμογή χρησιμοποιείται αυτή η μέθοδος στην οθόνη ενός έξυπνου κινητού τηλεφώνου.
    - Οπτικές διαφανείς οθόνες. Σε αυτές, ένα οπτικό σύστημα προβολής υπερθέτει εικονικές πληροφορίες στο πεδίο θέασης του ελεγκτή. Τέτοιες είναι τα γυαλιά Επαυξημένης Πραγματικότητας.

Εικόνα 1: Γυαλιά Επαυξημένης Πραγματικότητας

Η Επαυξημένη Πραγματικότητα βασίζεται σε μεγάλο βαθμό στη θέση του χρήστη, ή της συσκευής που αυτός χρησιμοποιεί, στο χώρο. Έτσι κρίνεται αναγκαίος ο

καθορισμός τόσο της τοποθεσίας όσο και τον προσανατολισμού. Για τον πλήρη καθορισμό αυτών, χρειάζονται δεδομένα για 6 βαθμούς ελευθερίας. Αυτοί είναι οι συντεταγμένες x, y και z της συσκευής καθώς και οι γωνίες εκτροπής(yaw), κλίσης(pitch) και περιστροφής(roll).[4]



Εικόνα 2: Απεικόνιση των 6 βαθμών ελευθερίας

Για τον καθορισμό αυτών των δεδομένων χρησιμοποιούνται διάφορα είδη αισθητήρων.

2.2.1 Αισθητήρες συσκευής

Στις εφαρμογές Επαυξημένης Πραγματικότητας, ιδιαίτερα σε αυτές που εκτελούνται σε κινητές συσκευές, χρησιμοποιείται ένα ευρύ φάσμα αισθητήρων. Ένας τέτοιος είναι το GPS. Πρόκειται για ένα σύστημα πλοήγησης που χρησιμοποιεί ένα δίκτυο δορυφόρων και μπορεί να καθορίσει τις συντεταγμένες x,y,z της συσκευής πάνω στη Γη. Για να το πετύχει αυτό επικοινωνεί με τουλάχιστον 3 δορυφόρους και μετρά το χρόνο που χρειάζεται ο κάθε ένας για να απαντήσει. Έτσι καθορίζει τις αποστάσεις από κάθε δορυφόρο και τις συγκρίνει για να καθορίσει την τοποθεσία της συσκευής. Ωστόσο το GPS δεν έχει τη δυνατότητα να υπολογίσει τον προσανατολισμό της συσκευής. Για το λόγο αυτό χρησιμοποιούνται αισθητήρες όπως η πυξίδα, το γυροσκόπιο και το επιταχυνσιόμετρο[6].

Το γυροσκόπιο[7] είναι μια συσκευή που μετρά γωνιακή ταχύτητα, δηλαδή το ρυθμό αλλαγής της γωνίας γύρω από τους 3 άξονες. Δε δίνει δεδομένα για την τοποθεσία της συσκευής, αλλά μπορεί να χρησιμοποιηθεί για τον καθορισμό του προσανατολισμού της συσκευής.

Το επιταχυνσιόμετρο[8] μετρά την επιτάχυνση της συσκευής στους 3 άξονες x,y,z. Μπορεί να χρησιμοποιηθεί για να παρέχει την κατεύθυνση στην οποία κινείται η συσκευή.

Η πυξίδα χρησιμοποιείται για τη μέτρηση της γωνίας της συσκευής γύρω από τους 3 άξονες, δηλαδή μετρά τις γωνίες εκτροπής(yaw), κλίσης(pitch) και περιστροφής(roll).

2.2.2 Κάμερα

Ο καθορισμός της θέσης της συσκευής μπορεί να γίνει και με τη χρήση της κάμερας. Μέσω της εικόνας που δίνεται από αυτή χρησιμοποιούνται τεχνικές όρασης υπολογιστών ώστε να εκτιμηθεί η θέση καθώς και η κίνηση της συσκευής στο χώρο. Αυτό μπορεί να επιτευχθεί με τον εντοπισμό ιδιαίτερων χαρακτηριστικών ή μνημείων

στην εικόνα της κάμερας και μελετώντας τον τρόπο με τον οποίο αυτά μεταβάλλονται όσο ο χρήστης κινείται στον τρισδιάστατο κόσμο[9].

2.2.3 Βάθος

Η όραση υπολογιστών δε χρησιμοποιείται μόνο για τον υπολογισμό της θέσης της συσκευής αλλά και για την εκτίμηση της απόστασης κάθε σημείου της εικόνας από την κάμερα υπολογίζοντας έτσι το βάθος της εικόνας. Άλλοι τρόποι υπολογισμού του βάθους συμπεριλαμβάνουν κάποιου είδους αισθητήρα. Τέτοιοι είναι οι αισθητήρες LiDAR, που χρησιμοποιούν οπτικά ηλεκτρομαγνητικά κύματα ώστε να πετύχουν με μεγάλη ακρίβεια τη δημιουργία μιας τρισδιάστατης εικόνας βάθους[10]. Έτσι μπορεί κανείς να εκμεταλλευτεί τις λειτουργίες της Επαυξημένης Πραγματικότητας για να παράξει ένα χάρτη βάθους και να τον χρησιμοποιήσει για τη σωστή τοποθέτηση αντικειμένων καθώς και για τον εντοπισμό εμποδίων[11].

3 Μέθοδοι

3.1 Αλγόριθμος

Για τον επιτυχή εντοπισμό εμποδίων και γκρεμών, προτείνεται η εφαρμογή ενός επαναληπτικού Αλγορίθμου. Ο Αλγόριθμος αυτός αξιοποιεί τη λειτουργία εκτίμησης βάθους στην εικόνα της κάμερας, όπως παρέχεται από τη βιβλιοθήκη Επαυξημένης Πραγματικότητας που χρησιμοποιείται. Μέσω αυτής, η εικόνα διαχωρίζεται σε επιμέρους τμήματα, στα οποία εντοπίζονται εμπόδια ή γκρεμοί, με στόχο την ενημέρωση του χρήστη μέσω ηχητικών μηνυμάτων. Προσφέρεται επίσης η δυνατότητα απεικόνισης του χάρτη βάθους που έχει υπολογιστεί. Επιπλέον, έχουν τεθεί συγκεκριμένα όρια στον προσανατολισμό της συσκευής, προκειμένου να διασφαλιστεί ότι η εικόνα της κάμερας περιλαμβάνει το μονοπάτι του χρήστη και όχι περιττό περιεχόμενο. Ο Αλγόριθμος υπολογίζει συνεχώς τον προσανατολισμό της συσκευής και ενημερώνει τον χρήστη σε περίπτωση που τα όρια αυτά παραβιαστούν. Παρατίθεται ο ψευδοκώδικας του Αλγόριθμου καθώς και η αναλυτικότερη εξήγησή του:

Algorithm 1 Detect Obstacles and Steep Paths

```
1: while true do
2:   Update orientation angles
3:   if angleY > maxYangle or angleY < minYangle or |angleZ| > maxZangle then
4:     Warn user
5:     continue
6:   end if
7:   Acquire depth image
8:   if Depth switch then
9:     Visualize depth map
10:  end if
11:  Split image into a 4x3 grid
12:  for i = 0 to 4 do
13:    for j = 0 to 3 do
14:      Calculate and show average depth in cell i, j
15:    end for
16:  end for
17:  for i = 0 to 4 do
18:    for j = 0 to 3 do
19:      if cell i, j contains an obstacle then
20:        Paint the text in the cell i, j red
21:      else if i = 3 and cell i, j contains a steep path then
22:        Paint the text in the cell i, j blue
23:      end if
24:    end for
25:  end for
26:  if any steep path was found then
27:    Warn user through sound
28:  else if any obstacle was found then
29:    Warn user through sound
30:  end if
31: end while
```

Εικόνα 3: Ψευδοκώδικας Αλγόριθμου

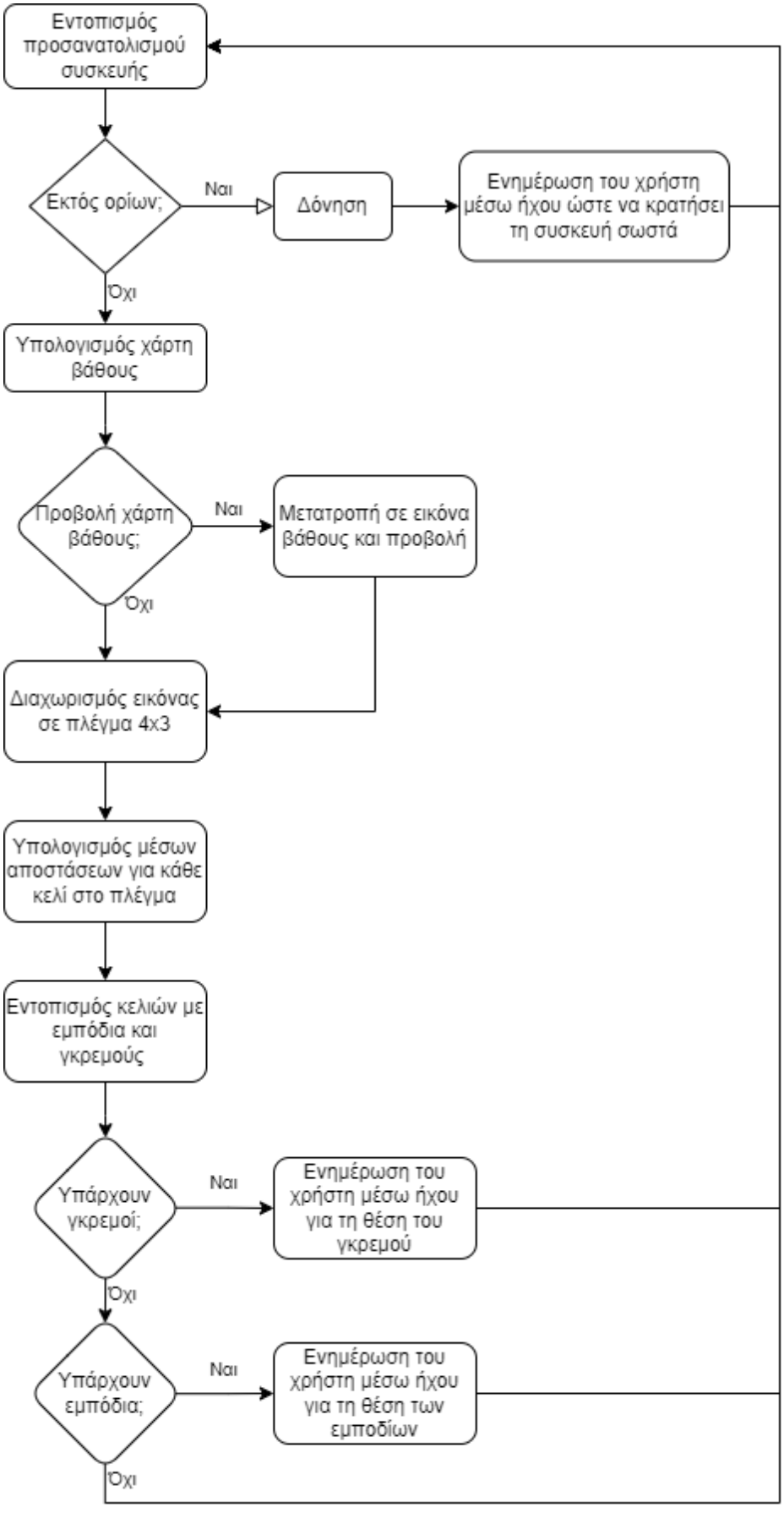
Όπως φαίνεται από την έκφραση while true, ο Αλγόριθμος είναι επαναληπτικός. Το πρώτο βήμα της κάθε επανάληψης είναι ο έλεγχος του προσανατολισμού της συσκευής. Αυτός γίνεται για να αποθαρρύνει το χρήστη από το να κρατά τη συσκευή σε παράλογες γωνίες θέασης, όπως για παράδειγμα την οριζόντια θέση όπου η κάμερα κοιτά το έδαφος ή τον ουρανό. Ο έλεγχος πραγματοποιείται με μετρήσεις που λαμβάνονται από τους αισθητήρες της συσκευής που αναλύθηκαν νωρίτερα. Εάν η συσκευή είναι κεκλιμένη εκτός των επιτρεπόμενων ορίων τότε ειδοποιείται ο χρήστης μέσω δόνησης, ήχου και κειμένου και ο Αλγόριθμος προχωρά στην επόμενη επανάληψη.

Μετά τον επιτυχημένο έλεγχο προσανατολισμού, χρησιμοποιούνται οι δυνατότητες βάθους της Επαυξημένης Πραγματικότητας ώστε να δημιουργηθεί μια εικόνα βάθους βασισμένη στην εικόνα της κάμερας. Η εικόνα αυτή μπορεί να προβληθεί σαν χάρτης βάθους στην οθόνη εάν ο χρήστης έχει επιλέξει την ανάλογη λειτουργία. Στη συνέχεια η εικόνα βάθους χωρίζεται σε ένα πλέγμα 4 γραμμών και 3 στηλών, υπολογίζεται ο μέσος όρος βάθους σε κάθε κελί και προβάλλεται στην οθόνη.

Το επόμενο βήμα αφορά τον εντοπισμό εμποδίων και γκρεμών. Για κάθε κελί του πλέγματος συγκρίνεται ο μέσος όρος που υπολογίστηκε στο προηγούμενο βήμα με ένα προκαθορισμένο όριο. Εάν ο μέσος όρος είναι μικρότερος από το όριο, τότε θεωρείται ότι σε αυτό το κελί υπάρχει εμπόδιο. Αντίστοιχα, για τον εντοπισμό γκρεμών ελέγχεται μόνο η τελευταία γραμμή του πλέγματος. Εάν σε κάποιο από τα κελιά της γραμμής εντοπιστεί μέσος όρος μεγαλύτερος από το προκαθορισμένο όριο τότε θεωρείται ότι εκεί υπάρχει γκρεμός.

Το τελευταίο βήμα είναι η προειδοποίηση του χρήστη. Κατά την ανάπτυξη της εφαρμογής κρίθηκε πως η ύπαρξη γκρεμού αποτελεί μεγαλύτερο κίνδυνο για τον χρήστη από κάποιο εμπόδιο. Έτσι εάν έχει εντοπιστεί γκρεμός, ο χρήστης ειδοποιείται για αυτόν και όχι για τα πιθανά εμπόδια στο δρόμο του. Η ειδοποίηση αυτή γίνεται μέσω ήχου και ενημερώνει το χρήστη για τη θέση του γκρεμού, ευθεία, δεξιά ή αριστερά. Ωστόσο εάν δεν εντοπιστεί γκρεμός αλλά εμπόδια, ο Αλγόριθμος ειδοποιεί τον χρήστη για τη θέση των εμποδίων μπροστά του. Τέλος, στην περίπτωση μη ύπαρξης τόσο γκρεμών, όσο και εμποδίων ο Αλγόριθμος προχωρά στην επόμενη επανάληψη.

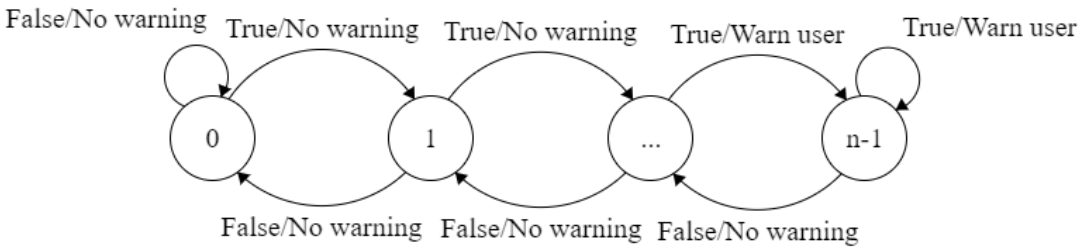
Ακολουθεί το διάγραμμα ροής του Αλγόριθμου:



Εικόνα 4: Διάγραμμα ροής Αλγόριθμου

3.2 Μηχανές Καταστάσεων

Ο προτεινόμενος Αλγόριθμος δύναται να αποτελέσει και σύνολο από βασικές Πεπερασμένες Μηχανές Καταστάσεων[19]. Μια Πεπερασμένη Μηχανή Καταστάσεων περιέχει συγκεκριμένο αριθμό καταστάσεων και παράγει εξόδους σε κάθε αλλαγή κατάστασης αφού λάβει μια είσοδο. Στο προτεινόμενο σύστημα χρησιμοποιούνται για τη μείωση της ευαισθησίας του Αλγόριθμου σε εσφαλμένες μετρήσεις. Η μορφή τους είναι η ακόλουθη:



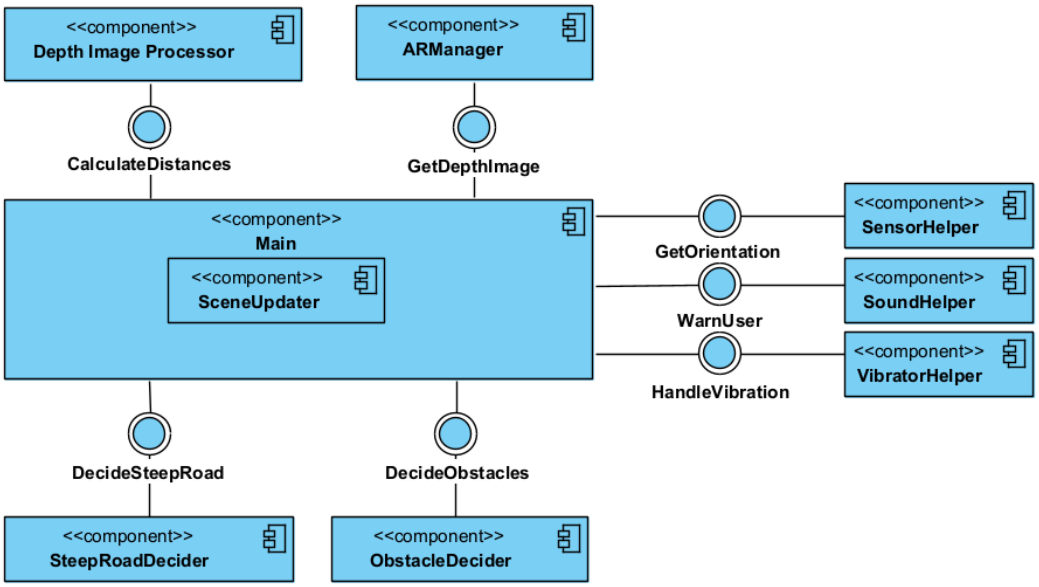
Εικόνα 5: Μορφή Μηχανών Καταστάσεων

4 Αρχιτεκτονική Συστήματος

Το προτεινόμενο Σύστημα προκειμένου να επιτύχει τον στόχο του κρίθηκε εύλογο να χωριστεί στα εξής Τμήματα - Components:

- Κύρια Components: Main Component - Scene Updater, ARManager, Depth Image Processor
- Components απόφασης: SteepRoadDecider, ObstacleDecider
- Components βοηθητικών λειτουργιών: SensorHelper, SoundHelper, VibratorHelper

Η βασική Αρχιτεκτονική της εφαρμογής περιγράφεται στο παρακάτω διάγραμμα:



Εικόνα 6: Αρχιτεκτονική συστήματος

4.2 Components

4.2.1 AR Manager

Ο AR Manager είναι το component που διαχειρίζεται τις λειτουργίες Επαυξημένης Πραγματικότητας. Είναι υπεύθυνος για την καταγραφή της εικόνας της κάμερας και την προβολή της εικονικής κάμερας στην οθόνη. Παράλληλα γνωρίζει ανά πάσα στιγμή τη θέση της κάμερας στον τρισδιάστατο χώρο καθώς και τον προσανατολισμό της συσκευής. Χρησιμοποιώντας αυτά τα στοιχεία έχει τη δυνατότητα εντοπίζει επιφάνειες στο χώρο. Παρέχει επίσης τη λειτουργία εντοπισμού βάθους της εικόνας, δίνοντας στο χρήστη τη δυνατότητα να εντοπίσει κοντινά αντικείμενα η εμπόδια στο μονοπάτι του χρήστη.

4.2.2 Sensor Helper

Ο Sensor Helper λαμβάνει τα δεδομένα του επιταχυνσιόμετρου και του αισθητήρα μαγνητικού πεδίου της συσκευής και τα χρησιμοποιεί για τον υπολογισμό του προσανατολισμού της συσκευής στους 3 άξονες. Οι μετρήσεις του χρησιμοποιούνται από τα επόμενα components της Αρχιτεκτονικής, ώστε να αποφασιστεί εάν η κλίση

της συσκευής είναι εκτός των ορίων που έχουν τεθεί για τη σωστή λειτουργία του Αλγόριθμου.

4.2.3 Sound Helper

Ο Sound Helper παρέχει τη λειτουργία ηχητικής ενημέρωσης του χρήστη. Στην περίπτωση που βρεθεί κάποιο εμπόδιο, κάποιος γκρεμός ή στην περίπτωση που η συσκευή είναι κεκλιμένη εκτός ορίων, το component αυτό μπορεί να χρησιμοποιηθεί για την αναπαραγωγή του κατάλληλου ήχου, είτε για τη θέση του εμποδίου ή του γκρεμού στο χώρο είτε για να παροτρύνει το χρήστη να κρατήσει όρθια τη συσκευή. Τα ονόματα των αρχείων ήχου και το περιεχόμενό τους, καθώς και τα σενάρια στα οποία αυτά αναπαράγονται αναλύονται στην παράγραφο 6.5.6 στο κεφάλαιο της ανάλυσης του κώδικα.

4.2.4 Vibrator Helper

Ο Vibrator Helper είναι ένα ακόμα component που χρησιμοποιείται για την προειδοποίηση του χρήστη. Όταν υπάρχει παραβίαση των ορίων της κλίσης χρησιμοποιεί το δονητή της συσκευής για να παράξει περιοδικές δονήσεις. Ο Vibrator Helper διαβάζει τις μετρήσεις του SensorHelper για να ελέγχει την παραβίαση ορίων. Ωστόσο οι μετρήσεις αυτές βασίζονται στο επιταχυνσιόμετρο, και είναι λανθασμένες για ένα μικρό χρονικό διάστημα σε κάθε βήμα του χρήστη, λόγω της επιτάχυνσης που προκαλεί αυτός στη συσκευή. Έτσι, για την παραγωγή δόνησης, πρέπει να εντοπιστεί παραβίαση ορίων περισσότερες από μια φορές, ανάλογα με το ρυθμό εκτέλεσης του Αλγόριθμου. Αυτό μειώνει την ευαισθησία του component σε σύντομες λανθασμένες ή μη ακριβείς μετρήσεις. Για να επιτευχθεί αυτό, το συγκεκριμένο component αποτελεί μια Μηχανή Καταστάσεων με τη μορφή της εικόνας Εικόνα 5. Ο αριθμός των καταστάσεων της μηχανής είσαι ίσος με το ρυθμό εκτέλεσης του Αλγορίθμου ανά δευτερόλεπτο. Αυτό οδηγεί στην προειδοποίηση του χρήστη ακριβώς ένα δευτερόλεπτο μετά την παραβίαση των ορίων κλίσης.

4.2.5 Depth Image Processor

Ο Depth Image Processor επεξεργάζεται την εικόνα βάθους που υπολογίζει ο ARManager. Κατά τη διάρκεια των δοκιμών αποφασίστηκε ότι για να επιτευχθεί βέλτιστη απόδοση εντοπισμού εμποδίων χρειάζεται μόνο ένα ποσοστό της εικόνας αυτής. Συγκεκριμένα τα σημεία που βρίσκονται στα δεξιά και στα αριστερά της εικόνας δεν αποτελούν το μονοπάτι του χρήστη με αποτέλεσμα να μην απαιτείται εκεί ο εντοπισμός εμποδίων. Έτσι, το κομμάτι αυτό λαμβάνει την εικόνα βάθους και αγνοεί ένα ποσοστό της στα αριστερά και τα δεξιά. Το ποσοστό αυτό ορίζεται από το χρήστη. Στη συνέχεια, χωρίζει το ωφέλιμο μέρος της εικόνας σε ένα πλέγμα 4x3, δηλαδή 4 γραμμών και 3 στηλών και για κάθε κομμάτι του πλέγματος υπολογίζει το μέσο όρο των αποστάσεων των σημείων που βρίσκονται μέσα σε αυτό. Οι τιμές που προκύπτουν χρησιμοποιούνται αργότερα για τον εντοπισμό εμποδίων ή γκρεμών. Παράλληλα μπορεί να μετατρέψει τον πίνακα αυτό σε μορφή έτοιμη προς απεικόνιση στην οθόνη.

4.2.6 Steep Road Decider

Ο Steep Road Decider είναι υπεύθυνος για τον εντοπισμό απότομων μονοπατιών ή γκρεμών στην εικόνα της κάμερας. Για να το πετύχει αυτό χρησιμοποιεί την κατώτερη γραμμή του πλέγματος που υπολογίζει ο Depth Image Processor.

Χρησιμοποιώντας το μέσο όρο του βάθους της κάθε στήλης, αποφασίζει εάν υπάρχει απότομο μονοπάτι η γκρεμός ευθεία, αριστερά ή δεξιά. Εάν η τιμή του μέσου όρου είναι μεγαλύτερη από το προκαθορισμένο όριο το κομμάτι αυτό αποφασίζει ότι σε αυτό το κελί υπάρχει γκρεμός. Για να ληφθεί μια τέτοια απόφαση το ίδιο απότομο μονοπάτι πρέπει να εντοπιστεί περισσότερες από μια φορές. Αυτό μειώνει την ευαισθησία του component σε σύντομες λανθασμένες ή μη ακριβείς μετρήσεις. Έτσι, ο Steep Road Decider είναι ένας πίνακας με 3 Μηχανές Καταστάσεων, μια για κάθε κελί της τελευταίας στήλης. Κάθε μηχανή έχει τη μορφή της εικόνας Εικόνα 5. Ο αριθμός των καταστάσεων της κάθε μηχανής είσαι ίσος με το ρυθμό εκτέλεσης του Αλγορίθμου ανά δευτερόλεπτο. Αυτό οδηγεί στην προειδοποίηση του χρήστη ακριβώς ένα δευτερόλεπτο μετά τον εντοπισμό γκρεμού.

4.2.7 Obstacle Decider

Ο Obstacle Decider χρησιμοποιείται για τον εντοπισμό εμποδίων στην εικόνα της κάμερας. Εδώ χρησιμοποιείται όλο το πλέγμα που υπολογίζει ο Depth Image Processor. Εάν ο μέσος όρος βάθους κάποιου κελιού είναι μικρότερος από το προκαθορισμένο όριο, το component αποφασίζει ότι σε αυτό το τμήμα της εικόνας υπάρχει εμπόδιο. Το όριο αυτό είναι συνάρτηση της γραμμής στην οποία βρίσκεται το κελί. Παρόμοια με τον Steep Road Decider, το ίδιο εμπόδιο πρέπει να εντοπιστεί περισσότερες από μια φορές, ανάλογα με το ρυθμό λήψης μετρήσεων βάθους, για την αποφυγή προειδοποίησης του χρήστη λόγω μεμονωμένων λανθασμένων μετρήσεων. Έτσι ο Obstacle Decider είναι ένας πίνακας με 12 Μηχανές Καταστάσεων, μια για κάθε κελί του πλέγματος. Κάθε μηχανή έχει τη μορφή της εικόνας Εικόνα 5. Ο αριθμός των καταστάσεων της μηχανής είσαι ίσος με το ρυθμό εκτέλεσης του Αλγορίθμου ανά δευτερόλεπτο. Αυτό οδηγεί στην προειδοποίηση του χρήστη ακριβώς ένα δευτερόλεπτο μετά τον εντοπισμό εμποδίου.

4.2.8 Main Component

Το Main Component αποτελεί την καρδία της εφαρμογής. Εδώ επιτελούνται όλες οι λειτουργίες της γραφικής διεπαφής της εφαρμογής. Παράλληλα συνδυάζονται όλα τα υπόλοιπα στοιχεία της Αρχιτεκτονικής ώστε να παραχθεί το επιθυμητό αποτέλεσμα. Όπως φαίνεται στο διάγραμμα, περιλαμβάνει τον SceneUpdater, ο οποίος ρυθμίζει το σταθερό ρυθμό εκτέλεσης του Αλγόριθμου.

Σε κάθε επανάληψη παίρνει τη γωνία κλίσης που υπολογίζει ο SensorHelper. Αν αυτή είναι εκτός ορίων, ενημερώνει το χρήστη ηχητικά μέσω του Sound Helper, με κείμενο στην οθόνη και με δόνηση μέσω του VibratorHelper και σταματά την εκτέλεση.

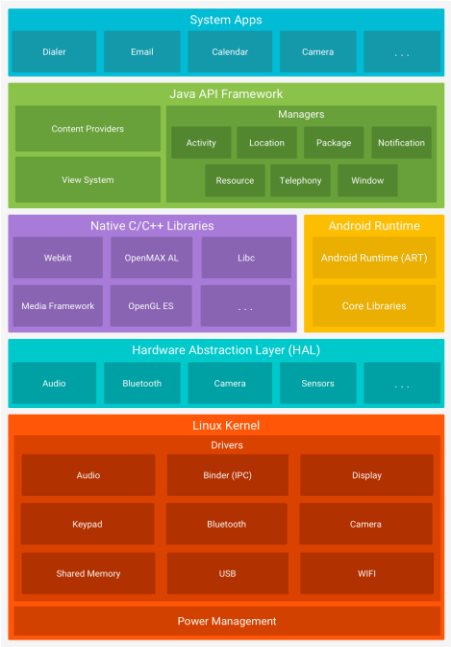
Αν η γωνία κλίσης είναι εντός ορίων, παίρνει τον πίνακα βάθους που υπολογίζει ο ARManager και χρησιμοποιεί τον DepthImageProcessor για να αποκτήσει το πλέγμα που περιέχει τους μέσους όρους βάθους. Προαιρετικά εμφανίζει και το χάρτη βάθους εάν έχει ζητηθεί από το χρήστη μέσω του κατάλληλου διακόπτη στη γραφική διεπαφή.

Στη συνέχεια παρέχει το πλέγμα στον Obstacle Decider και στον Steep Road Decider. Εάν αποφασιστεί ότι υπάρχει εμπόδιο ή γκρεμός, το Main Component χρησιμοποιεί τον SoundHelper για να προειδοποιήσει τον χρήστη.

Κεφάλαιο 5
Τεχνολογίες Προγραμματισμού
5.1 Java
Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε η γλώσσα Java[20][21]. Η Java είναι μια γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems το 1995. Είναι μια αντικειμενοστραφής γλώσσα, βασίζεται δηλαδή σε κλάσεις και αντικείμενα. Μια κλάση αποτελεί ένα προσχέδιο που μπορεί να χρησιμοποιηθεί για τη δημιουργία αντικειμένων. Περιέχει πεδία, τα οποία προσδιορίζουν ιδιότητες των αντικειμένων που θα δημιουργηθούν, καθώς και μεθόδους που επιτελούν λειτουργίες στα αντίστοιχα αντικείμενα.

Η Java δεν εξαρτάται από το λειτουργικό σύστημα του μηχανήματος στο οποίο τρέχει. Οι εφαρμογές εκτελούνται τόσο σε λειτουργικά συστήματα σταθερών υπολογιστών (Windows, MacOS, Linux κτλ.), όσο και σε λειτουργικά συστήματα φορητών συσκευών (στην προκειμένη περίπτωση Android) χωρίς να απαιτείται καινούρια μεταγλώττιση ή αλλαγή του κώδικα. Αυτό επιτυγχάνεται με τη χρήση της Εικονικής Μηχανής της Java (Java Virtual Machine). Ο κώδικας μεταγλωττίζεται σε κώδικα byte (bytecode) με τη χρήση του μεταγλωττιστή javac. Έπειτα η Εικονική Μηχανή της Java μετατρέπει τον κώδικα byte σε γλώσσα μηχανής που υποστηρίζεται από το λειτουργικό σύστημα και το μηχάνημα στο οποίο πρόκειται να εκτελεστεί. Υπάρχουν διαφορετικές εκδόσεις της Εικονικής Μηχανής για κάθε λειτουργικό σύστημα που υποστηρίζεται.

5.2 Λειτουργικό Σύστημα Android
Το Android είναι μια στοίβα λογισμικού ανοιχτού κώδικα που βασίζεται σε Linux[22].



Εικόνα 7: Στοίβα λογισμικού Android

Η βάση του Android είναι ο πυρήνας Linux ο οποίος διαχειρίζεται λειτουργίες όπως η νημάτωση και η διαχείριση μνήμης, ενώ παρέχει πολλά χαρακτηριστικά ασφαλείας. Το επίπεδο αφαίρεσης υλικού (Hardware Abstraction Layer) παρέχει διεπαφές οι οποίες επιτρέπουν στο Java API Framework να εκμεταλλευτεί τις δυνατότητες του υλικού της συσκευής. Κάθε εφαρμογή τρέχει σε δική της διεργασία με το δικό της στιγμιότυπο (instance) του Android Runtime (ART). Τέλος το Java API Framework παρέχει όλες τις δυνατότητες του Android ως διεπαφές και συναρτήσεις γραμμένες σε Java.

5.2.1 Εκτέλεση εφαρμογών

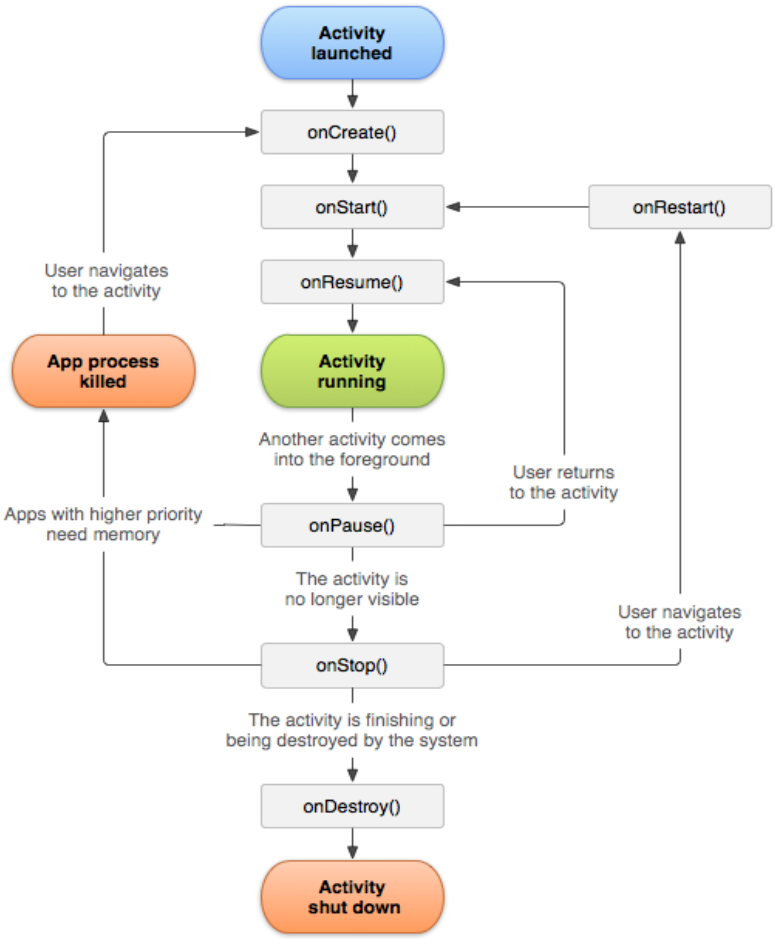
5.2.1.1 Δραστηριότητες (Activities)

Το Android χρησιμοποιεί την κλάση Activity[23] για να διαχειριστεί την εκτέλεση εφαρμογών. Η εκτέλεση του κώδικα ξεκινά σε ένα στιγμιότυπο της κλάσης Activity. Οι εφαρμογές κινητών συσκευών μπορούν να περιέχουν αρκετές οθόνες και αποτελούνται από πολλές δραστηριότητες (activities), μια για κάθε οθόνη που περιέχουν. Τυπικά υπάρχει μια κύρια δραστηριότητα η οποία υλοποιεί την αρχική οθόνη, δηλαδή αυτή που εμφανίζεται όταν ξεκινά η εφαρμογή. Τα στιγμιότυπα της κλάσης Activity παρέχουν το παράθυρο στο οποίο η εφαρμογή εμφανίζει τη γραφική της διεπαφή(GUI). Οι εφαρμογές δεν ξεκινούν απαραίτητα από την ίδια οθόνη. Συχνά μια εφαρμογή χρειάζεται να καλέσει μια άλλη και να ανοίξει μια οθόνη διαφορετική από την αρχική της. Για να το πετύχει αυτό, καλεί την κατάλληλη δραστηριότητα στην άλλη εφαρμογή.

Όσο ο χρήστης περιηγείται μέσα ή έξω από την εφαρμογή, οι δραστηριότητες περνούν από τα στάδια του κύκλου ζωής τους. Η κλάση Activity παρέχει callback συναρτήσεις ώστε η κάθε μια να γνωρίζει πότε αλλάζει μια κατάσταση του κύκλου ζωής. Μέσα στις συναρτήσεις αυτές δηλώνεται η συμπεριφορά της εφαρμογής όταν εισέρχεται σε μια κατάσταση. Οι συναρτήσεις αυτές είναι οι εξής[24]:

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroy()

Στο επόμενο διάγραμμα φαίνεται ο κύκλος ζωής μιας δραστηριότητας:

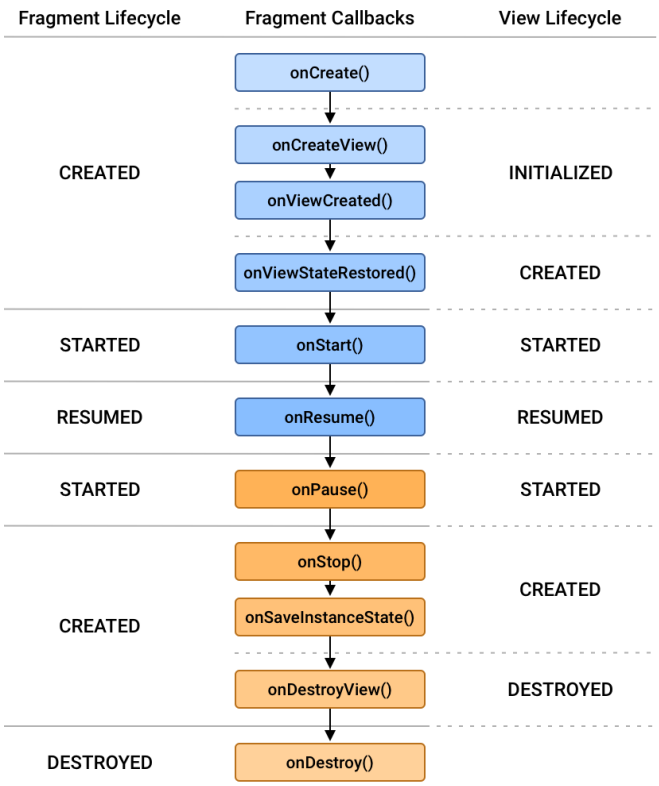


Εικόνα 8: Κύκλος ζωής δραστηριότητας Android

5.3.1.2 Fragments (Κομμάτια)

Η κλάση Fragment[25] του Android χρησιμοποιείται για να αναπαραστήσει κομμάτια της γραφικής διεπαφής μιας εφαρμογής. Η χρήση τους επιτρέπει το διαχωρισμό της γραφικής διεπαφής σε τμήματα καθώς και την προσαρμογή της σε διάφορα μεγέθη οθόνης. Τα κομμάτια αυτά δε μπορούν να υπάρχουν μόνα τους καθώς πρέπει να βρίσκονται μέσα σε μια εφαρμογή.

Κάθε κομμάτι έχει το δικό του κύκλο ζωής[26], ανεξάρτητο από αυτόν της δραστηριότητας στην οποία βρίσκεται. Η διαχείριση της συμπεριφοράς του κομματιού όταν αυτό αλλάζει κατάσταση γίνεται μέσω της διεπαφής LifecycleOwner που υλοποιεί η κλάση Fragment. Εναλλακτικά, η κλάση Fragment παρέχει τις ίδιες callback μεθόδους που παρέχει η κλάση Activity.



Εικόνα 9: Κύκλος ζωής Android Fragment

Σε αυτή την εφαρμογή, το Sceneform χρησιμοποιεί ένα αντικείμενο Fragment για να εμφανίζει μέσα σε αυτό την εικόνα της κάμερας μαζί με τις πιθανές επιφάνειες που εντοπίζει.

5.2.2 Αισθητήρες (Sensors)

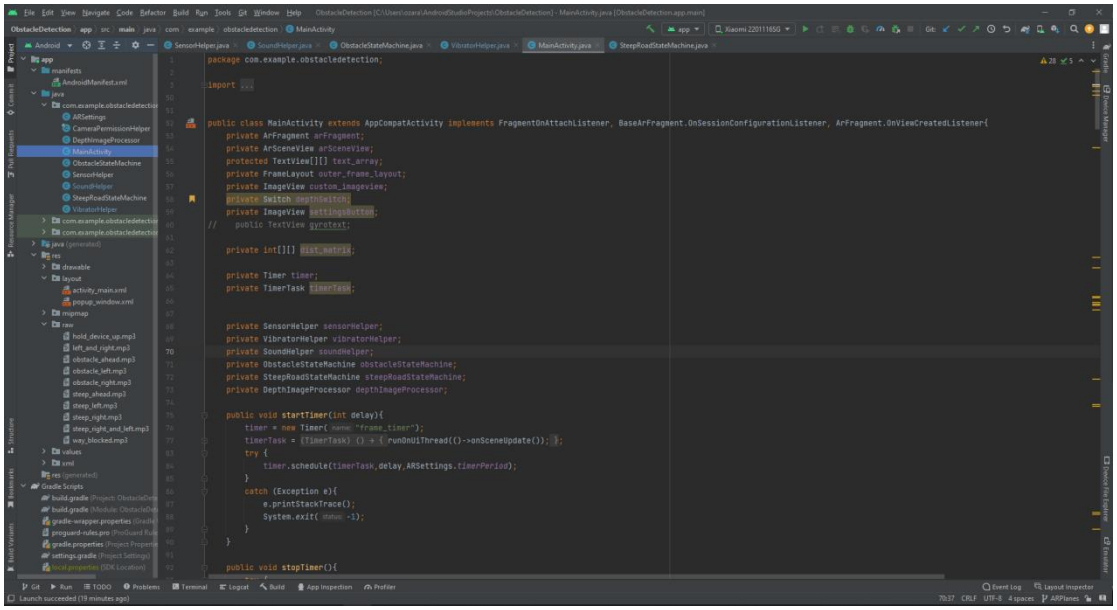
Υπάρχουν δύο είδη αισθητήρων, αυτοί που είναι βασισμένοι στο υλικό της συσκευής και αυτοί που παράγονται μέσω λογισμικού. Οι πρώτοι λαμβάνουν τα δεδομένα τους απευθείας από φυσικά εξαρτήματα της συσκευής, ενώ οι δεύτεροι παράγουν τα δεδομένα τους συνδυάζοντας μετρήσεις από 1 ή περισσότερους φυσικούς αισθητήρες. Το Android παρέχει πρόσβαση σε αυτούς μέσω του Android Sensor Framework[27] που αποτελεί κομμάτι του πακέτου android.hardware. Περιέχει τις κλάσεις SensorManager, Sensor και SensorEvent και τη διεπαφή SensorEventListener για την αρχικοποίηση και τη χρήση των αισθητήρων. Η συγκεκριμένη εφαρμογή χρησιμοποιεί μόνο αισθητήρες βασισμένους σε υλικό και συγκεκριμένα το επιταχυνσιόμετρο (accelerometer) και τον αισθητήρα μαγνητικού πεδίου (magnetic field).

5.2.3 Εξαρτήματα

Το Android παρέχει επίσης διεπαφές για τη χρήση διαφόρων εξαρτημάτων της συσκευής όπως η Κάμερα, οι Αισθητήρες, το Bluetooth, ο Δονητής[28] και τα Ηχεία.

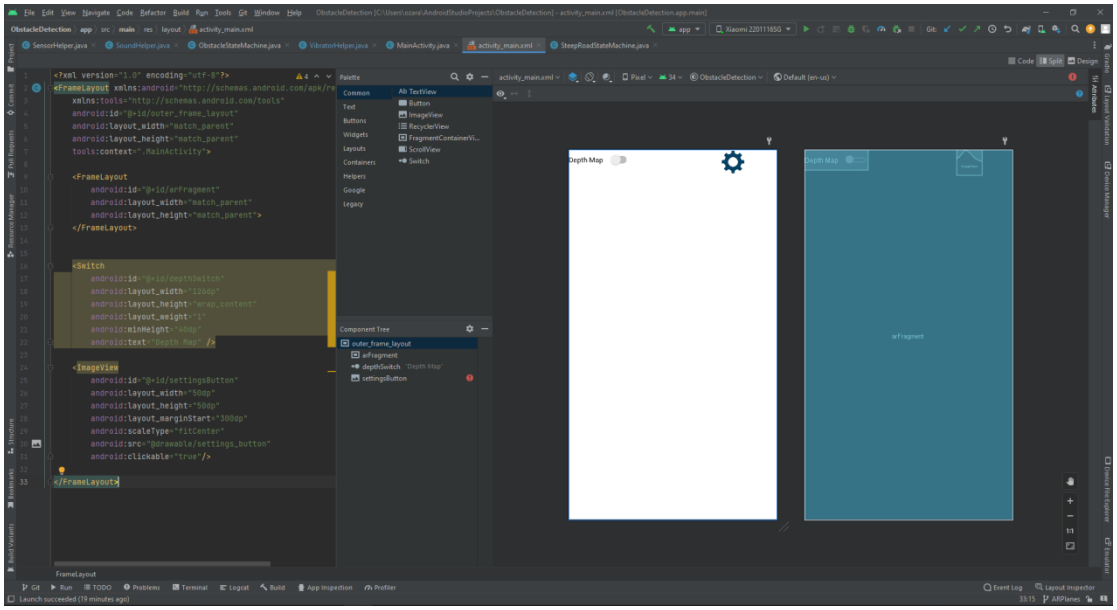
5.3 Android Studio

Το Android Studio[29] είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment, IDE) για τη δημιουργία εφαρμογών Android.



Εικόνα 10: Το περιβάλλον ανάπτυξης Android Studio

Εστιάζει στη διευκόλυνση του χρήστη προσφέροντας λειτουργίες όπως η δομή του project (στα αριστερά) και ο έξυπνος code editor που επιτρέπει την αυτόματη συμπλήρωση εντολών και την αυτόματη εισαγωγή των κατάλληλων βιβλιοθηκών. Περιέχει επίσης ένα ενσωματωμένο Layout Editor για τη δημιουργία και την επεξεργασία της γραφικής διεπαφής χωρίς να είναι απαραίτητη η χρήση κώδικα ή η γραφή XML αρχείων.



Εικόνα 11: Περιβάλλον επεξεργασίας xml αρχείων

Τέλος, παρέχει ένα αρκετά γρήγορο προσομοιωτή με πολλές λειτουργίες για να δώσει στο χρήστη τη δυνατότητα να δοκιμάσει και να διορθώσει τη συμπεριφορά της εφαρμογής του σε διάφορες καταστάσεις. Ωστόσο, η χρήση του προσομοιωτή δεν είναι απαραίτητη, αφού υπάρχει επίσης η επιλογή να δοκιμαστεί η εφαρμογή σε φυσική συσκευή.

5.4 Google ARCore

Το ARCore[30] είναι η πλατφόρμα της Google για την ανάπτυξη εφαρμογών Επαυξημένης Πραγματικότητας. Παρέχει διάφορα API που δίνουν τη δυνατότητα σε μια κινητή συσκευή να κατανοήσει το περιβάλλον γύρω της και να αλληλεπιδράσει με αυτό. Για να το πετύχει αυτό χρησιμοποιεί τεχνικές όπως εντοπισμό κίνησης, κατανόηση περιβάλλοντος και κατανόηση βάθους.

5.4.1 Εντοπισμός κίνησης (Motion Tracking)

Το ARCore χρησιμοποιεί την τεχνική SLAM[31] (Simultaneous Localization and Mapping) για να καταλάβει τη θέση της συσκευής στο χώρο. SLAM ονομάζεται το υπολογιστικό πρόβλημα της κατασκευής ή ενημέρωσης ενός χάρτη ενός άγνωστου περιβάλλοντος, διατηρώντας ταυτόχρονα τη θέση ενός παράγοντα μέσα σε αυτό.

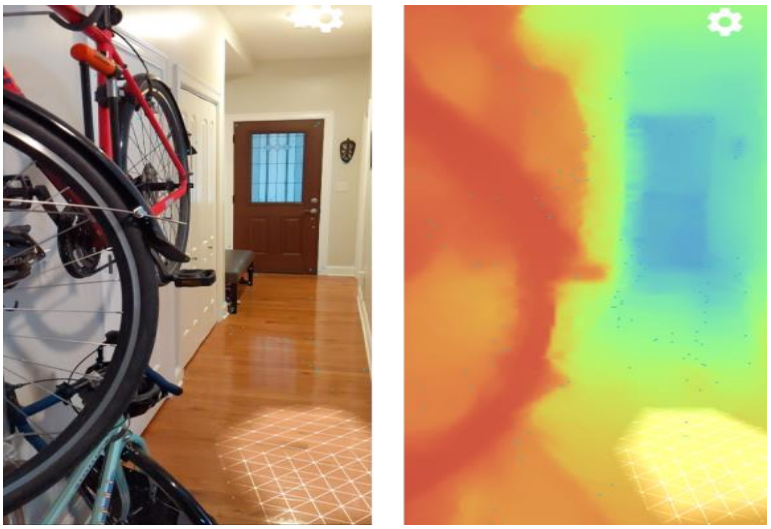
Ο Αλγόριθμος της Google για το SLAM εντοπίζει ιδιαίτερα οπτικά χαρακτηριστικά(feature points) στην εικόνα της κάμερας και τα χρησιμοποιεί για να υπολογίσει την αλλαγή της θέσης της συσκευής στο χώρο. Παράλληλα, εκτιμά τη θέση και τον προσανατολισμό της συσκευής χρησιμοποιώντας τις μετρήσεις της μονάδας μέτρησης αδράνειας της συσκευής (Inertial Measurement Unit - IMU). Χρησιμοποιώντας αυτά τα στοιχεία το ARCore μπορεί να εμφανίσει εικονικό περιεχόμενο στην οθόνη με τρόπο που το κάνει να φαίνεται σαν κομμάτι της πραγματικότητας[32] .

5.4.2 Κατανόηση Περιβάλλοντος (Environmental understanding)

Η πλατφόρμα εντοπίζει συστάδες από feature points που φαίνεται να σχηματίζουν οριζόντιες ή κάθετες επιφάνειες και δημιουργεί γεωμετρικές επιφάνειες. Αυτές και τα όριά τους δίνονται στο χρήστη για την τοποθέτηση αντικειμένων σε επιφάνειες. Ωστόσο επιφάνειες χωρίς υφή είναι δύσκολο να εντοπιστούν σωστά[32].

5.4.3 Υπολογισμός βάθους (Depth understanding)

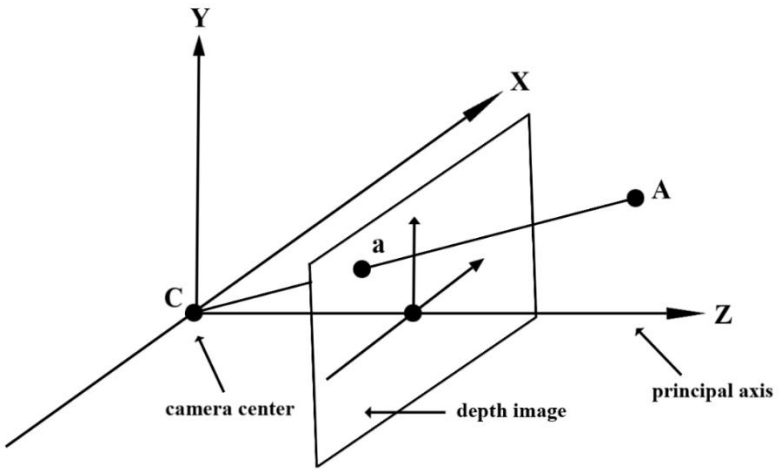
Ο υπολογισμός βάθους της εικόνας είναι η κύρια λειτουργία του ARCore που χρησιμοποιείται από την εφαρμογή για την αποφυγή εμποδίων. Το ARCore μπορεί να δημιουργήσει χάρτες βάθους, δηλαδή πίνακες ή εικόνες που περιέχουν δεδομένα για την απόσταση των επιφανειών χρησιμοποιώντας την κάμερα της συσκευής[33].



Εικόνα 12: Χάρτης βάθους της βιβλιοθήκης ARCore

Το Depth API χρησιμοποιεί ένα Αλγόριθμο Depth-From-Motion[32] για να δημιουργήσει εικόνες βάθους. Κάθε pixel στην εικόνα βάθους αναπαριστά την απόσταση της κάθε επιφάνειας από την κάμερα. Ο Αλγόριθμος παίρνει πολλές εικόνες από πολλές οπτικές γωνίες και τις συγκρίνει μεταξύ τους για τον υπολογισμό του βάθους. Παράλληλα χρησιμοποιεί μηχανική μάθηση επιλεκτικά για αυξημένη ακρίβεια, ακόμα και με ελάχιστη κίνηση της συσκευής. Τέλος, χρησιμοποιεί οποιοδήποτε διαθέσιμο αισθητήρα που μπορεί να υπολογίσει βάθος, όπως οι αισθητήρες ToF(Time-of-flight). Ωστόσο, στην προκειμένη περίπτωση η συσκευή που χρησιμοποιείται δε διαθέτει τέτοιο αισθητήρα.

Η τιμές αποστάσεων στην εικόνα βάθους υπολογίζονται χρησιμοποιώντας το παρακάτω διάγραμμα[34]:



Εικόνα 13: Επεξήγηση τιμών βάθους στην εικόνα

Το σημείο C αναπαριστά τη θέση της κάμερας στο χώρο ενώ το σημείο A αναπαριστά τη θέση του αντικειμένου στον πραγματικό κόσμο. Το σημείο a είναι το pixel στο οποίο το αντικείμενο εμφανίζεται στην οθόνη της συσκευής. Η τιμή της απόστασης που θα εμφανιστεί στο σημείο a, δεν είναι το μέγεθος του διανύσματος CA, αλλά το μέγεθος της προβολής του CA στον άξονα Z.

5.4.4 Sceneform

Η χρήση του ARCore απαιτεί κάποια βιβλιοθήκη για την απόδοση 2D και 3D γραφικών. Η ανάπτυξη της εφαρμογής ξεκίνησε με τη βιβλιοθήκη OpenGL (Open Graphics Library)[35]. Η OpenGL προσφέρει καλύτερη απόδοση στην εφαρμογή. Ωστόσο η χρήση της αποδείχθηκε δύσκολη καθώς ο χαμηλού επιπέδου χαρακτήρας της εισήγαγε μια σημαντική καμπύλη μάθησης, ενώ παράλληλα απαιτούσε μεγάλο όγκο πολύπλοκου κώδικα. Έτσι χρησιμοποιήθηκε η βιβλιοθήκη Sceneform.

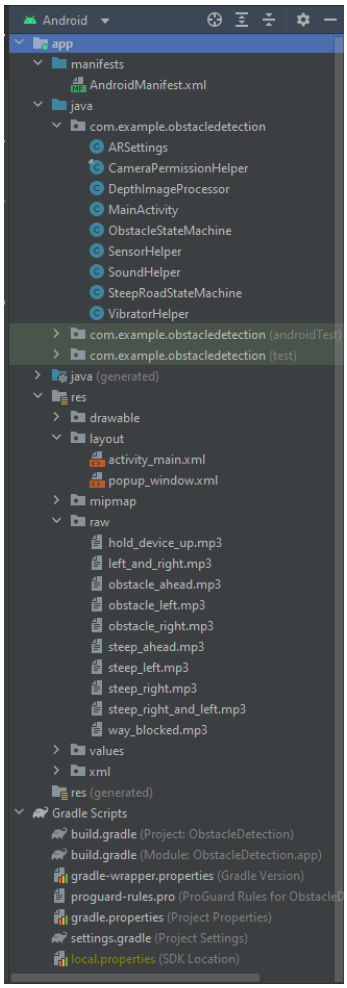
Η Sceneform[36][37] είναι μια πλατφόρμα απόδοσης τρισδιάστατων γραφικών η οποία δημιουργήθηκε από τη Google. Πρόκειται για μια βιβλιοθήκη βελτιστοποιημένη για κινητές συσκευές που χρησιμοποιείται για τη δημιουργία εφαρμογών Επαυξημένης Πραγματικότητας χωρίς να είναι απαραίτητη η χρήση της OpenGL και παρέχει τις λειτουργικότητες της βιβλιοθήκης ARCore μέσω ενός Android API. Η Google σταμάτησε την υποστήριξη του Sceneform και η ανάπτυξη του λογισμικού γίνεται πλέον από τον Thomas Gorisse και άλλους contributors.

Κεφάλαιο 6
Κώδικας

6.1 Δημιουργία Project

Για τη δημιουργία του project στο Android Studio επιλέχθηκε από το μενού File → New → New Project → Basic Activity → Finish. Τέλος, ορίστηκε το όνομα του project και επιλέχθηκε σαν Minimum SDK το API 26. Η δομή του project τροποποιήθηκε με τον ακόλουθο τρόπο:

Στους φακέλους java και res χρειάστηκαν μόνο τα αρχεία MainActivity και activity_main.xml, ενώ τα υπόλοιπα διαγράφηκαν. Διαγράφηκαν επίσης τα αρχεία στους φακέλους menu και navigation. Τέλος δημιουργήθηκαν τα απαραίτητα αρχεία κλάσεων. Η τελική δομή είναι η ακόλουθη:



Εικόνα 14: Τελική δομή αρχείων του πρότζεκτ

6.2 Gradle Files

Υπάρχουν 2 αρχεία που ονομάζονται build.gradle. Το πρώτο (Project build.gradle) αφορά ολόκληρο το project και είναι διαθέσιμο σε όλα τα modules που υπάρχουν σε αυτό.

Το δεύτερο gradle αρχείο (Module build.gradle) αφορά μόνο το συγκεκριμένο module. Σε αυτό προστέθηκε στα dependencies η εξής γραμμή:

```
dependencies {
    implementation "com.gorisse.thomas.sceneform:sceneform:1.23.0"
}
```

Με αυτή τη γραμμή δηλώνεται σαν απαιτούμενο πακέτο η βιβλιοθήκη Sceneform.

Παράλληλα αλλάχθηκαν οι παράμετροι compileSdk και targetSdk από 32 σε 34, καθώς αυτή η έκδοση του API απαιτούνταν από τα dependencies.

```
compileSdk 34
targetSdk 34
```

6.3 Manifest File

Στο αρχείο AndroidManifest.xml δηλώνονται οι άδειες και οι λειτουργίες που απαιτούνται για τη χρήση της εφαρμογής. Σε αυτό προστέθηκαν οι παρακάτω γραμμές.

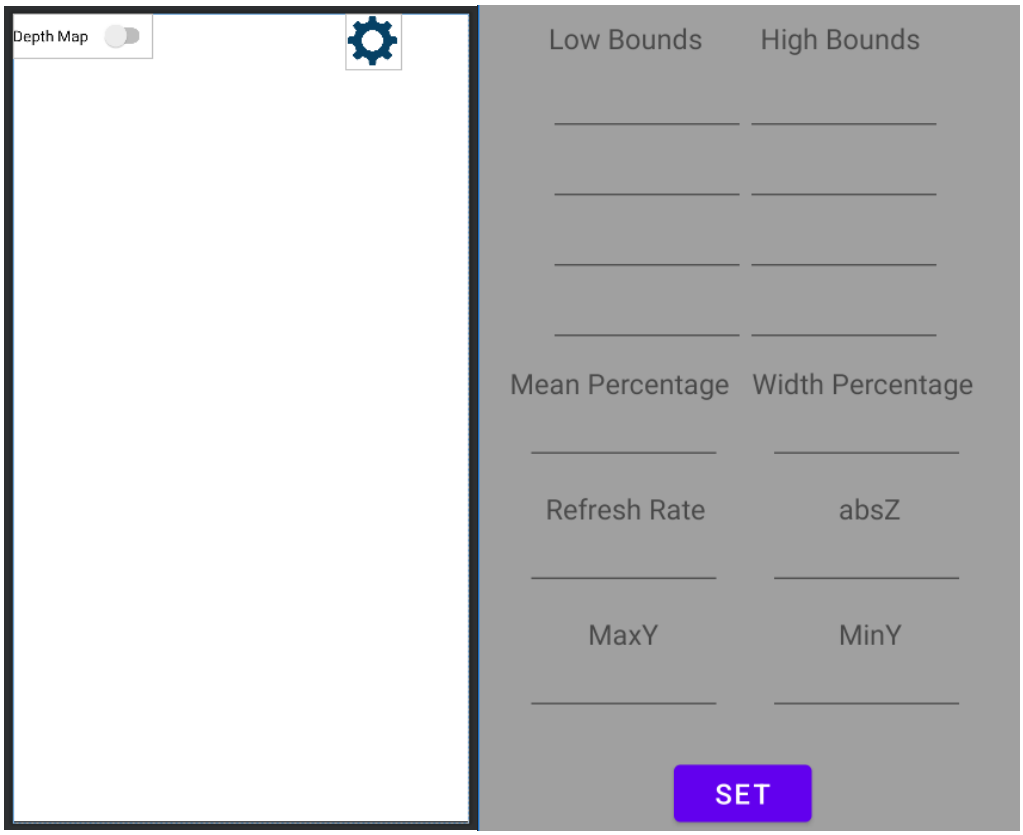
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-feature android:name="com.google.ar.core.depth" android:required="true" />
    <uses-feature android:glEsVersion="0x00030000" android:required="true" />
    <uses-feature android:name="android.hardware.camera.ar" android:required="true"/>
    <uses-feature
        android:name="android.hardware.sensor.accelerometer"
        android:required="true" />
    <uses-feature
        android:name="android.hardware.sensor.compass"
        android:required="true" />
    <application>
        <meta-data android:name="com.google.ar.core" android:value="required" />
        <activity
            android:screenOrientation="portrait">
        </activity>
    </application>
</manifest>
```

Με αυτό τον τρόπο δηλώνεται ότι για τη χρήση της εφαρμογής απαιτούνται οι άδειες κάμερας και δονητή καθώς και η ύπαρξη επιταχυνσιόμετρου και αισθητήρα μαγνητικού πεδίου. Επίσης η συσκευή πρέπει να υποστηρίζει λειτουργίες βάθους, τη βιβλιοθήκη ARCore καθώς και να έχει OpenGL.

6.4 Αρχεία Γραφικών

Τα αρχεία activity_main.xml και popup_window.xml ορίζουν τα στοιχεία που εμφανίζονται στη γραφική διεπαφή της εφαρμογής. Το πρώτο είναι η γενική διεπαφή

της εφαρμογής, ενώ το δεύτερο είναι το μενού επιλογών που εμφανίζεται εάν ο χρήστης πατήσει το κουμπί των ρυθμίσεων.



Εικόνα 15: Απεικόνιση αρχείων γραφικών της εφαρμογής

6.5 Αρχεία κλάσεων

6.5.1 ARSettings

Η συγκεκριμένη κλάση δε περιέχει μεθόδους. Χρησιμοποιείται για την ομαδοποίηση των παραμέτρων που χρησιμοποιεί η εφαρμογή:

```
public class ARSettings {
    protected static boolean depthMap=false;
    protected static final int numLabelRows=4;
    protected static final int numLabelCols=3;
    protected static int [] lowBoundArr = new int[] {2500,5000,4500,2000};
    protected static int [] highBoundArr = new int[] {5000,10000,9000,5500};
    protected static int width_percentage = 60;
    protected static int timerPeriod = 333;
    protected static int maxYAngle = 20;
    protected static int minYAngle = -10;
    protected static int absZAngle = 20;
}
```

- Η μεταβλητή depthMap ορίζει εάν θα εμφανίζεται στην οθόνη ο χάρτης βάθους.
- Η numLabelRows ορίζει τον αριθμό των γραμμών στο πλέγμα.

- Η numLabelCols ορίζει τον αριθμό των στηλών στο πλέγμα.
- Ο πίνακας lowBoundArr ορίζει το κάτω όριο των αποστάσεων σε κάθε γραμμή. Αν κάποιο σημείο βρίσκεται σε απόσταση μικρότερο από αυτό το όριο θεωρείται εμπόδιο και χρωματίζεται κόκκινο στο χάρτη βάθους.
- Ο πίνακας highBoundArr ορίζει το πάνω όριο των αποστάσεων σε κάθε γραμμή. Αν κάποιο σημείο βρίσκεται σε απόσταση μεγαλύτερη από αυτό το όριο και χρωματίζεται μπλε στο χάρτη βάθους. Η τελευταία τιμή του πίνακα αυτού αντιστοιχεί στην 4η γραμμή του πλέγματος και χρησιμοποιείται για τον εντοπισμό γκρεμών.
- Η width_percentage ορίζει το ποσοστό της εικόνας που λαμβάνεται υπ' όψη. Μέσω αυτής αγνοείται ένα μέρος της εικόνας στα αριστερά και στα δεξιά.
- Η timerPeriod ορίζει τη συχνότητα εκτέλεσης του Αλγορίθμου.
- Η maxYAngle ορίζει το όριο κλίσης της συσκευής προς τα κάτω.
- Η minYAngle ορίζει το όριο κλίσης της συσκευής προς τα πάνω.
- Η absZAngle ορίζει το όριο κλίσης της συσκευής γύρω από τον άξονα που είναι κάθετος στην οθόνη.
- Η mean_percent καθορίζει το πλάτος του πλέγματος στο οποίο γίνεται εντοπισμός εμποδίων.

6.5.2 Depth Image Processor

Η κλάση DepthImageProcessor αφορά το component Image Processor και περιέχει συναρτήσεις που αφορούν την επεξεργασία της εικόνας βάθους. Όπως εξηγήθηκε παραπάνω η κλάση αυτή χρησιμοποιείται για να χωρίζει το χρήσιμο μέρος μιας εικόνας βάθους σε ένα πλέγμα και να υπολογίζει τους μέσους όρους των αποστάσεων στο πλέγμα. Η δομή της είναι η εξής:

```
public class DepthImageProcessor {

    public int[][] getAverageDistances(Image depthImage, int rows, int cols, int
width_percentage){}

     public int getAverageSubImageDist(Image depthImage, int heightstart, int
heightend, int widthstart, int widthend, float percetage){}

    public Bitmap ImageToBitmap(Image depthImage,int numRows, int[] lowBoundArr,
int[] highBoundArr) {}
}
```

Η συνάρτηση getAverageSubImageDist() δέχεται μια εικόνα βάθους, καθώς και 4 συντεταγμένες μέσα σε αυτή που σχηματίζουν ένα ορθογώνιο κελί. Υπολογίζει και επιστρέφει το μέσο όρο των αποστάσεων των σημείων στο κελί αυτό. Υπάρχει επίσης η μεταβλητή percentage, που καθορίζει το ποσοστό των τιμών βάθους του κελιού που θα ληφθούν υπόψη στον υπολογισμό του μέσου όρου.

Η συνάρτηση getAverageDistances() παίρνει σαν ορίσματα μια εικόνα βάθους(depthImage), τον αριθμό των γραμμών(rows), τον αριθμό των στηλών(cols) και το ποσοστό της εικόνας που είναι χρήσιμο(width_percentage). Έπειτα χωρίζει την

εικόνα αυτή σε ένα πλέγμα [rows x cols] και υπολογίζει το μέσο όρο των αποστάσεων σε κάθε κελί χρησιμοποιώντας τη συνάρτηση getAverageSubImageDist(). Τέλος επιστρέφει ένα πίνακα [rows x cols] με τους μέσους όρους που υπολογίστηκαν.

Η συνάρτηση ImageToBitmap() παίρνει μια εικόνα βάθους, των αριθμό γραμμών του πλέγματος, τον πίνακα χαμηλών ορίων και τον πίνακα υψηλών ορίων και δημιουργεί ένα χάρτη βάθους. Τα σημεία με απόσταση μικρότερη από το χαμηλό τους όριο χρωματίζονται κόκκινα. Τα σημεία με απόσταση μεγαλύτερη από το υψηλό τους όριο χρωματίζονται μπλε. Τα σημεία ανάμεσα στα δύο όρια χρωματίζονται πράσινα. Τα όρια για κάθε σημείο καθορίζονται από τη γραμμή στην οποία βρίσκονται.

6.5.3 Obstacle State Machine

Η κλάση αυτή χρησιμοποιείται για τον εντοπισμό εμποδίων. Περιλαμβάνει μια πίνακα καταστάσεων που ενημερώνεται σε κάθε επανάληψη του Αλγορίθμου. Αυτό συμβαίνει για την αποφυγή ενημέρωσης του χρήστη σε περίπτωση μεμονωμένου λάθους. Η αριθμός καταστάσεων είναι πάντα ίσος με το ρυθμό εντοπισμού εμποδίων ανά δευτερόλεπτο. Έτσι εάν υπάρχει εντοπισμός εμποδίου με διάρκεια τουλάχιστον ένα δευτερόλεπτο, θα υπάρξει προειδοποίηση.

Η δομή της κλάσης είναι η εξής:

```
public class ObstacleStateMachine {
    private int [][] stateArr;
    private int states;

    public ObstacleStateMachine(int states) {}

    public void setStates(int states) {}

    public boolean[][] decideObstacles(int[][] dist_matrix, float current_angle){}

    public boolean[] updateObstacleStateMachine(boolean [][] obstacleArr){}
}
```

Η μέθοδος ObstacleStateMachine() αποτελεί τον constructor της κλάσης και αρχικοποιεί τον αριθμό καταστάσεων και τον πίνακα καταστάσεων.

Η μέθοδος setStates() χρησιμοποιείται για την αλλαγή του αριθμού καταστάσεων σε περίπτωση που ο χρήστης θέλει να αλλάξει το ρυθμό της εκτέλεσης του Αλγορίθμου. Έτσι εξασφαλίζεται ότι οι καταστάσεις θα είναι πάντα ίσες με το ρυθμό ανανέωσης και ότι ο εντοπισμός εμποδίων θα γίνεται μετά από 1 δευτερόλεπτο.

Η μέθοδος decideObstacles() λαμβάνει τον πίνακα μέσων όρων που παράγει ο ImageProcessor και τον χρησιμοποιεί για να εντοπίσει εμπόδια. Εάν σε κάποιο κελί η απόσταση είναι μικρότερη από το κάτω όριο του, αποφασίζει ότι σε αυτό υπάρχουν εμπόδια. Το όριο αυτό είναι συνάρτηση της γραμμής του κελιού και της γωνίας κλίσης της συσκευής. Έτσι η μέθοδος αυτή δέχεται και τη γωνία κλίσης της συσκευής. Το σταθερό όριο της γραμμής λαμβάνεται από την μεταβλητή

lowBoundArr της κλάσης ARSettings και μπορεί να παραμετροποιηθεί από το χρήστη. Η μέθοδος επιστρέφει ένα πίνακα boolean, ο οποίος περιέχει true στα κελιά που εντοπίστηκαν εμπόδια και false στα υπόλοιπα.

Η μέθοδος updateObstacleStateMachine() χρησιμοποιεί τον πίνακα που επιστρέφει η decideObstacles() για να ενημερώσει τον πίνακα καταστάσεων. Εάν σε κάποιο κελί υπάρχει true, υπάρχει δηλαδή εμπόδιο, αυξάνει το αντίστοιχο κελί του πίνακα καταστάσεων κατά 1. Εάν κάποιο κελί του πίνακα καταστάσεων είναι ίσο με states-1, όπου states οι ορισμένες καταστάσεις, τότε πρέπει για αυτό το κελί να ειδοποιηθεί ο χρήστης. Η μέθοδος επιστρέφει ένα πίνακα boolean με 3 κελιά. Κάθε κελί αναπαριστά μια στήλη. Αυτό συμβαίνει καθώς ο χρήστης ενημερώνεται για εμπόδια αριστερά, δεξιά ή ευθεία, δηλαδή κατά στήλες.

6.5.4 Steep Road State Machine

Η κλάση αυτή χρησιμοποιείται για τον εντοπισμό γκρεμών ή απότομων μονοπατιών. Η λειτουργία της και η δομή της μοιάζει πολύ με την Obstacle State Machine:

```
public class SteepRoadStateMachine {
    private int [] steepArr;
    private int states;

    public SteepRoadStateMachine(int states) {}

    public void setStates(int states) {}

    public boolean[] decideSteepAhead(int[][] dist_matrix, float current_angle){}

    public boolean[] updateSteepAheadStateMachine(boolean [] steepRoadArr){}
}
```

Η μέθοδος SteepRoadStateMachine() αποτελεί τον constructor της κλάσης και αρχικοποιεί τον αριθμό καταστάσεων και τον πίνακα καταστάσεων.

Η μέθοδος setStates() χρησιμοποιείται για την χρησιμοποιείται για την αλλαγή του αριθμού καταστάσεων σε περίπτωση που ο χρήστης θέλει να αλλάξει το ρυθμό της εκτέλεσης του Αλγορίθμου.

Η μέθοδος decideSteepAhead() λαμβάνει τον πίνακα μέσων όρων που παράγει ο ImageProcessor και τον χρησιμοποιεί για να εντοπίσει γρεμούς. Εάν σε κάποιο κελί η απόσταση είναι μεγαλύτερη από το πάνω όριο του, αποφασίζει ότι σε αυτό υπάρχει γκρεμός. Το όριο αυτό είναι συνάρτηση της γραμμής του κελιού και της γωνίας κλίσης της συσκευής. Έτσι η μέθοδος αυτή δέχεται και τη γωνία κλίσης της συσκευής. Το σταθερό όριο της γραμμής λαμβάνεται από την μεταβλητή highBoundArr της κλάσης ARSettings και μπορεί να παραμετροποιηθεί από το χρήστη. Η μέθοδος επιστρέφει ένα πίνακα boolean, ο οποίος περιέχει true στα κελιά που εντοπίστηκαν εμπόδια και false στα υπόλοιπα.

Η μέθοδος updateSteepAheadStateMachine() χρησιμοποιεί τον πίνακα που επιστρέφει η decideSteepAhead() για να ενημερώσει τον πίνακα καταστάσεων. Εάν

σε κάποιο κελί υπάρχει true, υπάρχει δηλαδή γκρεμός, αυξάνει το αντίστοιχο κελί του πίνακα καταστάσεων κατά 1. Εάν κάποιο κελί του πίνακα καταστάσεων είναι ίσο με states-1, όπου states οι ορισμένες καταστάσεις, τότε πρέπει για αυτό το κελί να ειδοποιηθεί ο χρήστης. Η μέθοδος επιστρέφει ένα πίνακα boolean με 3 κελιά, όπου κάθε κελί αναπαριστά μια στήλη.

6.5.5 Sensor Helper

Η κλάση αυτή χρησιμοποιείται για τη διαχείριση και τη χρήση των κατάλληλων αισθητήρων. Συγκεκριμένα ελέγχει την ύπαρξη του επιταχυνσιόμετρου και του αισθητήρα μαγνητικού πεδίου και τους χρησιμοποιεί για να υπολογίσει τον προσανατολισμό και την κλίση της συσκευής.

```
public class SensorHelper implements SensorEventListener {

    public SensorHelper(MainActivity mainActivity) {}

    @Override
    public void onSensorChanged(SensorEvent event) {}

    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {}

    public void updateOrientationAngles() {}

    public void resumeSensors(){}

    public void pauseSensors(){}
}
```

Η μέθοδος SensorHelper() είναι ο constructor της κλάσης. Ελέγχει την ύπαρξη των αισθητήρων στο κινητό και αρχικοποιεί τα κατάλληλα αντικείμενα για τη χρήση τους.

Για τη λήψη των δεδομένων που παράγουν οι αισθητήρες το Android παρέχει τη διεπαφή SensorEventListener. Αυτή η διεπαφή περιέχει τις callback μεθόδους onSensorChanged() και onAccuracyChanged(). Η πρώτη καλείται όταν η ακρίβεια ενός αισθητήρα αλλάζει ενώ η δεύτερη καλείται όταν ο αισθητήρας παρέχει μια νέα τιμή.

Η onAccuracyChanged() είναι κενή και συμπεριλαμβάνεται υποχρεωτικά αφού υλοποιείται η διεπαφή SensorEventListener.

Η onSensorChanged() ελέγχει τον τύπο του αισθητήρα που ενημερώθηκε. Εάν πρόκειται για το επιταχυνσιόμετρο ή για τον αισθητήρα μαγνητικού πεδίου, τότε αποθηκεύει τη νέα μέτρηση στη θέση της προηγούμενης.

Η updateOrientationAngles() χρησιμοποιεί τις πιο πρόσφατες μετρήσεις των 2 αισθητήρων για να παράξει την κλίση της συσκευής σε κάθε άξονα. Για να το πετύχει αυτό αλλάζει το σύστημα συντεταγμένων έτσι ώστε ο άξονας y να είναι κάθετος προς το έδαφος. Έτσι υπολογίζεται η κλίση της συσκευής όσο τη χειρίζεται ο χρήστης.

Οι μέθοδοι pauseSensors() και resumeSensors()[27] σταματάνε ή ξεκινάνε να ελέγχουν τους αισθητήρες για νέες τιμές. Αυτές χρησιμοποιούνται σε περίπτωση που ο χρήστης βγει από την εφαρμογή χωρίς όμως να την τερματίσει.

6.5.6 Sound Helper

Η κλάση αυτή είναι απαραίτητη για τη διαχείριση και την αναπαραγωγή των κατάλληλων ήχων για την προειδοποίηση του χρήστη. Για να το πετύχει αυτό χρησιμοποιεί την κλάση MediaPlayer του Android. Η δομή της είναι η εξής:

```
public class SoundHelper {
    private MainActivity mainActivity;
    private boolean isPlaying=false;
    private Map<String,MediaPlayer> soundMap;

    public SoundHelper(MainActivity mainActivity) {}

    public void playSound(String sound){}

    public boolean announceSteepRoad(boolean [] steepRoadArr){}

    public void announceObstacles(boolean [] obstacleArr){}
}
```

Ο constructor της κλάσης αρχικοποιεί τον χάρτη soundMap. Αυτός χρησιμοποιείται για την αντιστοίχιση των ονομάτων των ήχων της εφαρμογής με τα κατάλληλα αντικείμενα της κλάσης MediaPlayer. Οι ήχοι που χρησιμοποιούνται στην εφαρμογή παράχθηκαν από ένα μοντέλο τεχνητής νοημοσύνης[38] και είναι οι ακόλουθοι:

Όνομα	Ηχητικό μήνυμα
hold_device_up.mp3	Παρακαλώ κρατήστε όρθια τη συσκευή.
left_and_right.mp3	Εμπόδια αριστερά και δεξιά.
obstacle_ahead.mp3	Εμπόδιο ευθεία.
obstacle_left.mp3	Εμπόδιο αριστερά.
obstacle_right.mp3	Εμπόδιο δεξιά.
steep_ahead.mp3	Απότομα ευθεία.
steep_left.mp3	Απότομα αριστερά.
steep_right.mp3	Απότομα δεξιά.
steep_right_and_left.mp3	Απότομα αριστερά και δεξιά.
way_blocked.mp3	Ο δρόμος έχει εμπόδια.

Πίνακας 1: Ονόματα και περιεχόμενα αρχείων ήχου

Η μέθοδος playSound χρησιμοποιείται από τις επόμενες μεθόδους για την απαραγωγή του κατάλληλου ήχου. Δέχεται το όνομα του ήχου σαν όρισμα και αναζητά στο soundMap το αντικείμενο MediaPlayer που αντιστοιχεί σε αυτό. Τέλος ελέγχει αν αναπαράγεται τη συγκεκριμένη στιγμή κάποιος άλλος ήχος πιθανώς από την προηγούμενη επανάληψη του Αλγόριθμου. Εάν όχι, τότε αναπαράγει τον ήχο.

Η μέθοδος announceSteepRoad() δέχεται τον πίνακα steepRoadArr που παράγει η updateSteepAheadStateMachine() της κλάσης SteepRoadStateMachine. Πρόκειται

για ένα boolean πίνακα 1x3, όπου κάθε κελί αναπαριστά μια στήλη. Εάν σε κάποιο κελί υπάρχει true, αυτό σημαίνει ότι ο χρήστης πρέπει να ενημερωθεί για την ύπαρξη γκρεμού σε αυτό το σημείο. Η συνάρτηση ελέγχει τα εξής σενάρια με τη σειρά:

steepRoadArr[0]	steepRoadArr[0]	steepRoadArr[0]	Ήχος που αναπαράγεται
true	true	true	steep_ahead.mp3
true	*	true	steep_right_and_left.mp3
true	*	*	steep_left.mp3
*	*	true	steep_right.mp3
*	true	*	steep_ahead.mp3

Πίνακας 2: Ήχοι που αναπαράγονται με βάση τις περιπτώσεις εντοπισμού εμποδίου

Η μέθοδος announceObstacles() λειτουργεί με παρόμοιο τρόπο. Δέχεται τον πίνακα obstacleArr που παράγει η updateObstacleStateMachine() της κλάσης ObstacleStateMachine. Όπως και πριν, αυτός είναι ένας πίνακας 1x3 όπου κάθε κελί αναπαριστά μια στήλη. Εάν σε κάποιο κελί υπάρχει true, αυτό σημαίνει ότι ο χρήστης πρέπει να ενημερωθεί για την ύπαρξη εμποδίου σε αυτή τη στήλη. Η συνάρτηση ελέγχει τα εξής σενάρια με τη σειρά:

obstacleArr [0]	obstacleArr [0]	obstacleArr [0]	Ήχος που αναπαράγεται
true	true	true	way_blocked.mp3
true	*	true	left_and_right.mp3
true	*	*	obstacle_left.mp3
*	*	true	obstacle_right.mp3
*	true	*	obstacle_ahead.mp3

Πίνακας 3: Ήχοι που αναπαράγονται με βάση τις περιπτώσεις εντοπισμού γκρεμού

Ο ήχος hold_device_up.mp3 δε χρησιμοποιείται σε κάποια από της μεθόδους αυτής της κλάσης. Αυτό συμβαίνει καθώς η MainActivity είναι η κλάση που ελέγχει εάν η κλίση της συσκευής είναι εκτός ορίων. Σε αυτή την περίπτωση καλεί τη συνάρτηση playSoynd με το όνομα του ήχου hold_device_up.

6.5.7 Vibrator State Machine

Η κλάση VibratorStateMachine αποτελεί μια μηχανή καταστάσεων που διαχειρίζεται το δονητή της συσκευής. Η δομή της είναι η ακόλουθη:

```
public class VibratorStateMachine {
    private MainActivity mainActivity;
    private Vibrator vibrator;
    private boolean isVibrating;
    private int states;
    private int currState=0;

    public VibratorStateMachine(MainActivity mainActivity, int states){}

    public void setStates(int states) {}

    public boolean updateVibratorStateMachine(float yAngle, float zAngle){}

    public void vibrate(){}
```

```
    public void stopVibrating(){}
}
```

Ο constructor ελέγχει την ύπαρξη δονητή στη συσκευή και αρχικοποιεί το αντικείμενο Vibrator που παρέχει το Android.

Η μέθοδος setStates() χρησιμοποιείται για την χρησιμοποιείται για την αλλαγή του αριθμού καταστάσεων σε περίπτωση που ο χρήστης θέλει να αλλάξει το ρυθμό της εκτέλεσης του Αλγορίθμου.

Η μέθοδος updateVibratorStateMachine() αυξάνει τη μεταβλητή currState κατά 1 αν ανιχνεύσει ότι υπάρχει παραβίαση των ορίων στους άξονες. Αν ισχύει η συνθήκη currState==states-1, τότε η συσκευή πρέπει να δονηθεί και επιστρέφεται true. Διαφορετικά επιστρέφεται false.

Η vibrate() ξεκινά τη δόνηση. Αυτή έχει περίοδο 2 δευτερολέπτων, δηλαδή η συσκευή δονείται για 1 δευτερόλεπτο και σταματά για 1 δευτερόλεπτο.

Η stopVibrating() σταματά τη δόνηση.

6.5.8 Main Activity

Η κλάση MainActivity αποτελεί την καρδιά της εφαρμογής. Δημιουργεί την activity στην οποία εκτελείται η εφαρμογή και επιτελεί όλα τα στάδια του κύκλου ζωής της. Εδώ χρησιμοποιούνται οι μέθοδοι όλων των υπόλοιπων κλάσεων για τη διασφάλιση της ομαλής εκτέλεσης του Αλγόριθμου. Παράλληλα γίνεται η διαχείριση της γραφικής διεπαφής της εφαρμογής. Η δομή της είναι η ακόλουθη:

```
public class MainActivity extends AppCompatActivity implements
FragmentOnAttachListener, BaseArFragment.OnSessionConfigurationListener,
ArFragment.OnViewCreatedListener{

    @Override
    protected void onCreate(Bundle savedInstanceState) {}

    @Override
    protected void onResume() {}

    @Override
    protected void onPause() {}

    @Override
    public void onAttachFragment(@NonNull FragmentManager fragmentManager, @NonNull
Fragment fragment) {}

    @Override
    public void onSessionConfiguration(Session session, Config config) {}

    @Override
    public void onViewCreated(ArSceneView arSceneView) {}
```

```
    public void inflatePopupMenu(View view){}

    public void createLabelGrid(){}

    public void startTimer(int delay){}

    public void stopTimer(){}

    public void onSceneUpdate() {}
}
```

Οι πρώτες 3 μέθοδοι είναι οι callback μέθοδοι του κύκλου ζωής της εφαρμογής.

Η onCreate() αρχικοποιεί τα αντικείμενα όλων των κλάσεων που περιγράφηκαν παραπάνω. Θέτει επίσης τις λειτουργίες των στοιχείων της γραφικής διεπαφής και τα εμφανίζει στην οθόνη.

Η onResume() χρησιμοποιεί τη μέθοδο resumeSensors() της κλάσης SensorHelper για να ξεκινήσει να λαμβάνει μετρήσεις από τους αισθητήρες. Καλεί επίσης τη μέθοδο startTimer() που αναλύεται αργότερα.

Η onPause() χρησιμοποιεί τη μέθοδο pauseSensors() της κλάσης SensorHelper για να σταματήσει να λαμβάνει μετρήσεις από τους αισθητήρες. Καλεί επίσης τη μέθοδο stopTimer() που αναλύεται αργότερα.

Οι επόμενες 3 μέθοδοι σχετίζονται με τη διαχείριση της συνεδρίας Εικονικής Πραγματικότητας και του fragment στο οποίο αυτή εμφανίζει την εικονική κάμερα.

Η onAttachFragment() θέτει ως listeners τις επόμενες δύο μεθόδους.

Η onSessionConfiguration() ελέγχει εάν η συσκευή υποστηρίζει τις λειτουργίες βάθους του Sceneform. Εάν όχι, ενημερώνει το χρήστη και τερματίζει την εφαρμογή. Διαφορετικά παραμετροποιεί αυτόματα τις λειτουργίες βάθους.

Η onViewCreated() θέτει το ρυθμό ανανέωσης της κάμερας στο μέγιστο και απενεργοποιεί τη λειτουργία εμφάνισης επιφανειών του ARCore.

Η inflatePopupMenu() χρησιμοποιείται για την εμφάνιση του παραθύρου των ρυθμίσεων σε περίπτωση που πατηθεί το ανάλογο κουμπί. Η συγκεγκριμένη μέθοδος σταματά την εκτέλεση του κύριου Αλγόριθμου καλώντας την stopTimer() όσο ο χρήστης αλλάζει τις παραμέτρους, ώστε να μη χρησιμοποιούνται άσκοπα οι πόροι της συσκευής. Όταν ο χρήστης βγει από το μενού των ρυθμίσεων, καλείται ξανά η createLabelGrid() με τις νέες παραμέτρους και η startTimer(), ώστε να συνεχιστεί η εκτέλεση του Αλγόριθμου.

Η createLabelGrid() δημιουργεί στην οθόνη το πλέγμα των αποστάσεων που παράγει ο Depth Image Processor. Αυτό θα μπορούσε να γίνει και χωρίς τη χρήση κάποιας συνάρτησης, με την τροποποίηση του κύριου αρχείου γραφικών activity_main.xml. Ωστόσο η χρήση της συνάρτησης επιτρέπει τη δυναμική προσαρμογή παραμέτρων όπως το πλάτος του πλέγματος κατά την εκτέλεση της εφαρμογής.

Οι μέθοδοι startTimer() και stopTimer() χρησιμοποιούνται για την εκκίνηση και την παύση της εκτέλεσης του επαναληπτικού Αλγόριθμου. Η κλάση MainActivity περιέχει ένα αντικείμενο της κλάσης Timer που παρέχει το API του Android. Η κλάση αυτή επιτρέπει την εκτέλεση κώδικα επαναληπτικά σε σταθερά διαστήματα. Ο κώδικας που εκτελείται εδώ είναι η onSceneUpdate() ενώ το χρονικό διάστημα ανάμεσα σε κάθε εκτέλεση ορίζεται από τη μετραβλητή timerPeriod της κλάσης ARSettings, την οποία μπορεί να ορίσει ο χρήστης.

Η startTimer() δημιουργεί ένα τέτοιο αντικείμενο Timer ώστε να αρχίσει η εκτέλεση του Αλγόριθμου, ενώ η stopTimer() το διαγράφει.

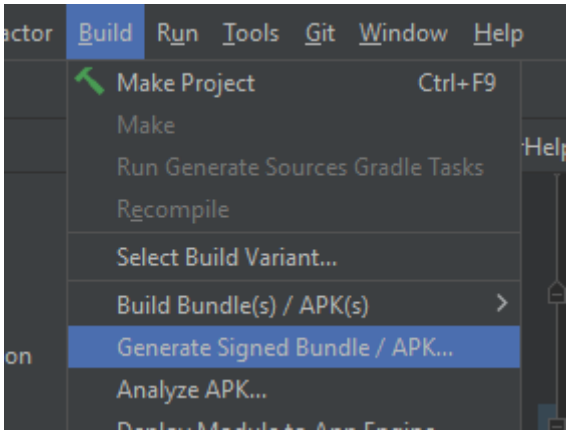
Η onSceneUpdate() περιέχει όλη τη λογική του Αλγόριθμου της εφαρμογής. Ξεκινά καλώντας τη μέθοδο sensorHelper.updateOrientationAngles() για να ενημερώσει τις μετρήσεις των αισθητήρων και έπειτα ελέγχει εάν η κλίση της συσκευής είναι εκτός των ορίων που έχουν οριστεί στην κλάση ARSettings. Σε περίπτωση όμως που η κλίση είναι εκτός αυτών των ορίων, καλεί τη μέθοδο vibratorHelper.vibrate() για να αρχίσει να δονείται η συσκευή και τη μέθοδο soundHelper.playSound("hold_device_up") και εμφανίζει στην οθόνη το μήνυμα «Παρακαλώ κρατήστε όρθια τη συσκευή» ώστε να παραπέμψει το χρήστη να κρατήσει τη συσκευή σωστά.

Εάν η συσκευή έχει αποδεκτές τιμές κλίσης ακολουθεί τα ακόλουθα βήματα:

- Χρησιμοποιεί τη μέθοδο acquireDepthImage16Bits() του Sceneform για να αποκτήσει την εικόνα βάθους.
- Καλεί την depthImageProcessor.getAverageDistances() και της δίνει σαν ένα από τα ορίσματα την εικόνα βάθους. Έτσι παίρνει τον πίνακα των μέσων όρων αποστάσεων.
- Μέσω των μεθόδων obstacleStateMachine.decideObstacles() και steepRoadStateMachine.decideSteepAhead() ενημερώνεται για την ύπαρξη εμποδίων ή γκρεμών σε κάποιο από τα κελιά του πίνακα μέσων όρων.
- Εμφανίζει τον πίνακα μέσων όρων στην οθόνη και χρωματίζει με κόκκινη γραμματοσειρά τις μετρήσεις των κελιών με εμπόδια και με μπλε γραμματοσειρά τις μετρήσεις των κελιών με γκρεμούς. Όλες οι υπόλοιπες μετρήσεις έχουν άσπρη γραμματοσειρά.
- Καλεί την steepRoadStateMachine.updateSteepAheadStateMachine() για να ενημερώσει τη Μηχανή Καταστάσεων που ελέγχει για γκρεμούς. Εάν αυτή αποφασίσει ότι ο χρήστης πρέπει να ενημερωθεί, τότε καλείται η soundHelper.announceSteepRoad() και ο Αλγόριθμος τερματίζει. Η ενημέρωση για εμπόδια δε γίνεται σε αυτή την περίπτωση, καθώς δόθηκε προτεραιότητα στην προειδοποίηση για γκρεμούς.
- Εαν δεν υπάρχουν γκρεμοί, τότε καλείται η obstacleStateMachine.updateObstacleStateMachine() για να ενημερωθεί η Μηχανή Καταστάσεων που ελέγχει για εμπόδια. Εάν αυτή αποφασίσει ότι ο χρήστης πρέπει να ενημερωθεί, τότε καλείται η soundHelper.announceObstacles() και ο Αλγόριθμος τερματίζει.

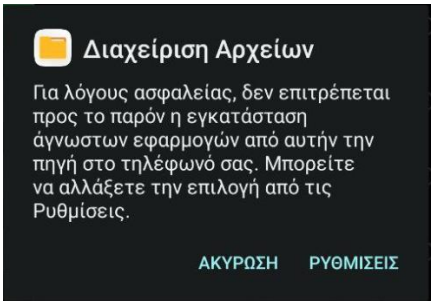
Κεφάλαιο 7
Το σύστημα στην πράξη
7.1 Συσκευή
Τόσο η λειτουργία της εφαρμογής όσο και οι τιμές των διαφόρων παραμέτρων δοκιμάστηκαν στην κινητή συσκευή Xiaomi Redmi Note 11 Pro 5G. Πρόκειται για μια συσκευή που παράχθηκε το 2022. Έχει 6 GB μνήμης RAM και επεξεργαστή Qualcomm Snapdragon 695 γεγονός που την κάνει αρκετά ισχυρή για την εκτέλεση της εφαρμογής. Έχει επίσης τους απαραίτητους αισθητήρες, όπως η κάμερα, το επιταχυνσιόμετρο και τον αισθητήρα μαγνητικού πεδίου, καθώς και εξαρτήματα όπως ηχεία και δονητής για την ενημέρωση του χρήστη. Τέλος υποστηρίζει το Depth API της βιβλιοθήκης Google ARCore, που χρησιμοποιείται για την αναγνώριση του βάθους της εικόνας.

7.2 Εγκατάσταση Συστήματος
Για την εγκατάσταση μιας εφαρμογής Android συνήθως προτείνεται η δημοσίευσή της σε κάποια αγορά εφαρμογών. Μια τέτοια είναι το Google Play[39] και είναι ιδιαίτερα χρήσιμη για τη διανομή εφαρμογών σε μεγάλα κοινά. Ωστόσο, κρίθηκε ότι δεν είναι απαραίτητη η χρήση αγοράς εφαρμογών, καθώς η εγκατάσταση μπορεί να γίνει και τοπικά. Για να γίνει αυτό απαιτείται ένα αρχείο APK (Android Package). Αυτό παράχθηκε μέσω του Android Studio μέσω της επιλογής Build à Generate Signed Bundle/APK…



Εικόνα 16: Δημιουργία αρχείου APK

Έπειτα το αρχείο apk μεταφέρθηκε στο σύστημα αρχείων της κινητής συσκευής και ανοίχτηκε. Το λειτουργικό σύστημα Android προειδοποιεί ότι πρόκειται για εφαρμογή άγνωστης πηγής και απαιτεί την παραχώρηση ειδικής άδειας από το μενού των ρυθμίσεων για να συνεχίσει.



Εικόνα 17: Προειδοποίηση του συστήματος Android για άγνωστες πηγές

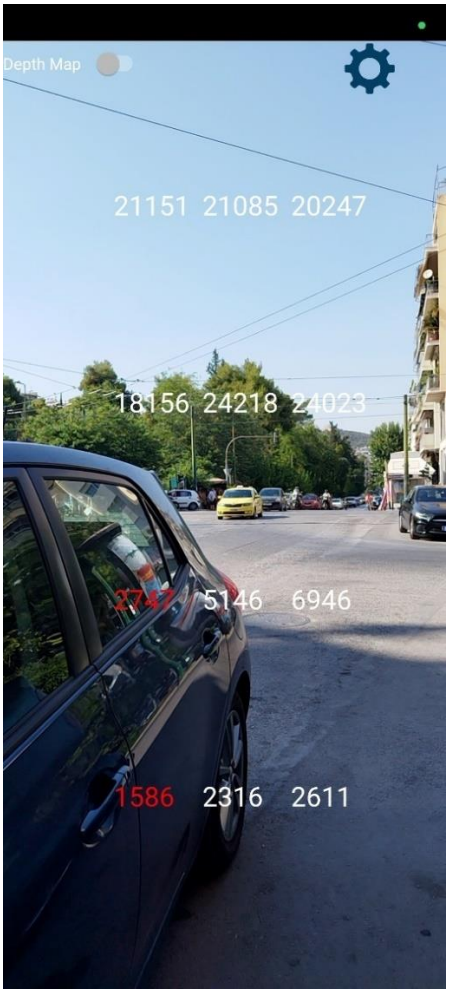
Αφού παραχωρηθεί η άδεια αυτή, η εγκατάσταση της εφαρμογής είναι επιτυχής.

7.3 Σενάρια και υποθέσεις

Για τη σωστή λειτουργία του συστήματος θεωρείται ότι ο χρήστης θα βρίσκεται σε εξωτερικό χώρο και κρατά όρθια τη συσκευή μπροστά του στο ύψος του στήθους. Θεωρείται επίσης ότι η εφαρμογή θα χρησιμοποιηθεί σε συνθήκες επαρκούς φωτισμού, καθώς οι κάμερες των κινητών συσκευών τείνουν να παρουσιάζουν πολύ κακή ποιότητα εικόνας όταν ο φωτισμός είναι χαμηλός.

7.4 Οθόνες εφαρμογής

Κατά την εκκίνηση της εφαρμογής εμφανίζεται στο χρήστη η κύρια οθόνη:



Εικόνα 18: Κύρια οθόνη εφαρμογής

Σε αυτήν εμφανίζεται το πλέγμα με τους μέσους όρους αποστάσεων και χρωματίζονται με κόκκινο χρώμα τα εμπόδια και με μπλε χρώμα οι γκρεμοί.

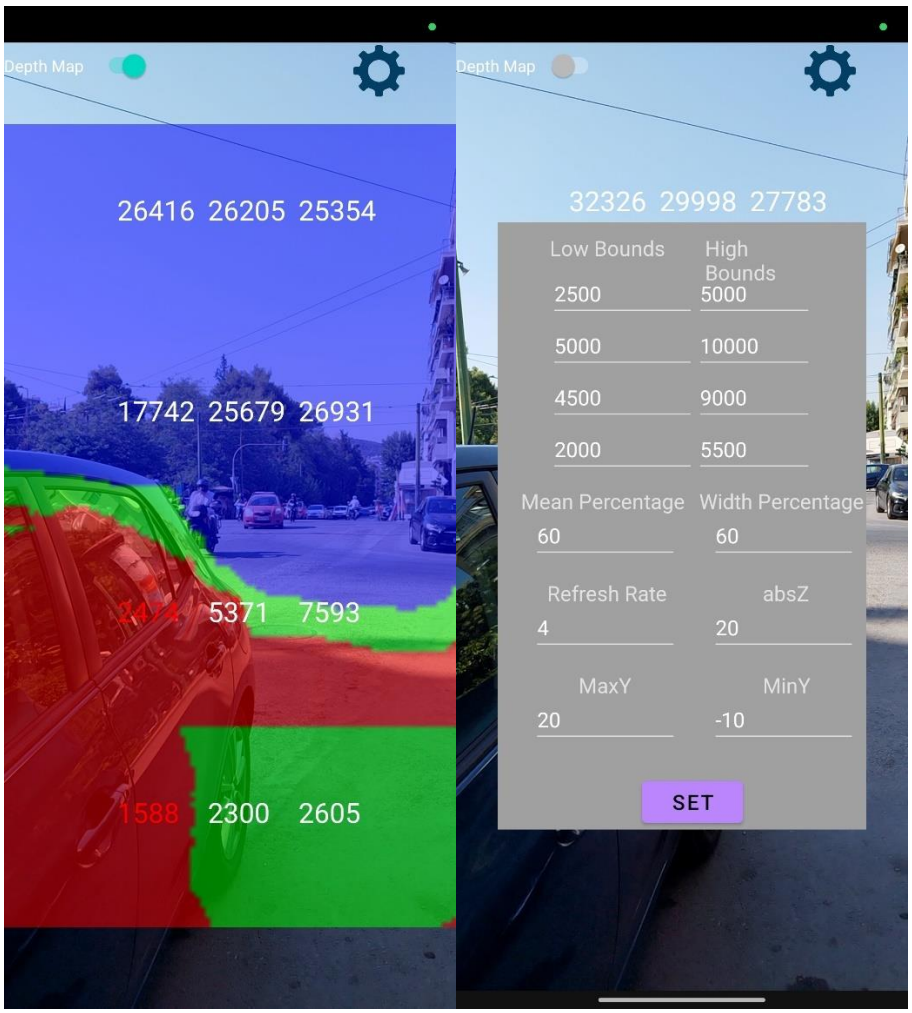
Με τη χρήση του διακόπτη πάνω αριστερά ο χρήστης μπορεί να ενεργοποιήσει την εμφάνιση του χάρτη βάθους. Σε αυτόν χρωματίζεται το κάθε pixel ανάλογα με το βάθος του. Με βάση τον παρακάτω πίνακα:

Τιμή Βάθους	Χρώμα
Μικρότερη από το όριο εμποδίων	Κόκκινο
Μεγαλύτερη από το όριο γκρεμών	Μπλε
Ανάμεσα στα δύο όρια	Πράσινο

Πίνακας 4: Χρώματα σημείων του χάστη βάθους με βάση την υπολογισμένη τιμή βάθους

Τα όρια που αναφέρονται θα εξηγηθούν παρακάτω στο κεφάλαιο των παραμέτρων.

Με το κουμπί του μενού πάνω δεξιά ο χρήστης μπορεί να ανοίξει το παράθυρο των ρυθμίσεων. Παρακάτω απεικονίζεται ο χάρτης βάθους μέσα από την εφαρμογή καθώς και το μενού:



Εικόνα 19: Χάρτης βάθους και μενού ρυθμίσεων μέσα από την εφαρμογή

7.4 Παραμετροποίηση εφαρμογής

Η εφαρμογή περιλαμβάνει παραμέτρους που είτε είναι στατικές, είτε μπορούν να μεταβληθούν από το χρήστη, όπως φαίνεται και από το παράθυρο των ρυθμίσεων. Για την ομαλή λειτουργία της εφαρμογής, δοκιμάστηκαν πολλές τιμές για κάθε παράμετρο και συγκρίθηκαν μεταξύ τους βάσει ορισμένων μετρικών.

7.4.1 Μετρικές σύγκρισης

Για τη σύγκριση μεταξύ παραμέτρων έγιναν δοκιμές σε ένα μονοπάτι με τα ίδια εμπόδια για κάθε παράμετρο. Έτσι συλλέχθηκαν τα εξής δεδομένα για κάθε τιμή:

- True Positives (TP): Ο αριθμός των εμποδίων ή γκρεμών που εντόπισε η εφαρμογή.
- False Positives (FP): Ο αριθμός των προειδοποιήσεων που έδωσε η εφαρμογή χωρίς να υπάρχει εμπόδιο ή γκρεμός.
- False Negatives (FN): Ο αριθμός των εμποδίων που δε κατάφερε να εντοπίσει η εφαρμογή.

Τα True Negatives (TN) είναι οι φορές που δεν υπήρχε εμπόδιο ή γκρεμός και η εφαρμογή δεν προειδοποίησε το χρήστη. Τα δεδομένα αυτά δεν καταγράφηκαν, καθώς η εφαρμογή περνά τον περισσότερο χρόνο σε αυτή την κατάσταση και ο αριθμός αυτός θα ήταν τάξεις μεγέθους μεγαλύτερος από τα υπόλοιπα δεδομένα.

Τα δεδομένα αυτά χρησιμοποιήθηκαν στις μετρικές Precision, Recall και F score[40].

Η μετρική Precision δίνεται από τον παρακάτω τύπο:

$$Precision = \frac{TP}{TP + FP}$$

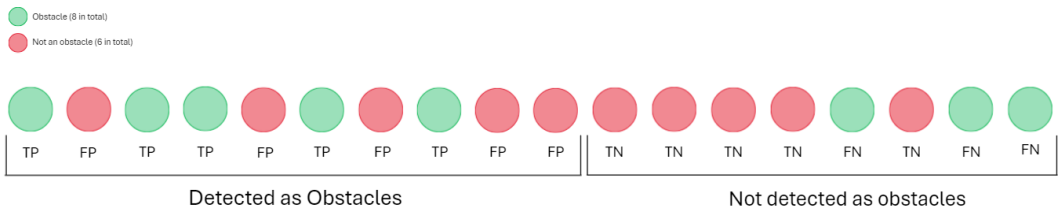
Η Precision δείχνει ποιο μέρος από τα εμπόδια που εντοπίστηκαν υπήρχαν πραγματικά.

Η μετρική Recall ορίζεται ως εξής:

$$Recall = \frac{TP}{TP + FN}$$

Η Recall δείχνει το μέρος των πραγματικών εμποδίων που εντοπίστηκαν από τον Αλγόριθμο.

Στο επόμενο παράδειγμα υπολογίζονται οι τιμές precision και recall:



Εικόνα 20: Απεικόνιση μετρικών precision και recall

$TP = 5, FP = 5, FN = 3$

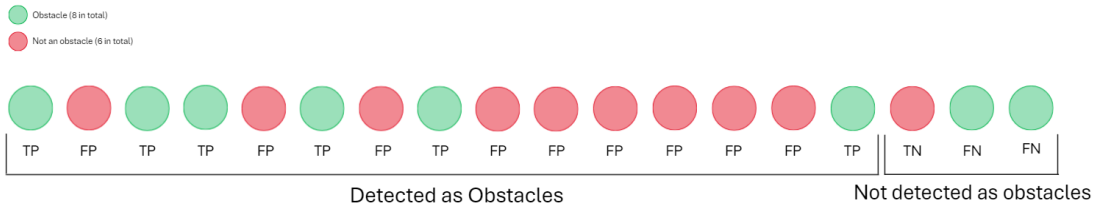
$$Precision = \frac{TP}{TP + FP} = \frac{5}{5 + 5} = 0{,}5 = 50\%$$

Δηλαδή το 50% των εμποδίων που εντοπίστηκαν ήταν πραγματικά.

$$Recall = \frac{TP}{TP + FN} = \frac{5}{5 + 3} = 0{,}625 = 62{,}5\%$$

Δηλαδή εντοπίστηκε το 62,5% των συνολικών πραγματικών εμποδίων.

Για τη σωστή αξιολόγηση της εφαρμογής πρέπει να μελετηθούν και οι 2 μετρικές. Ωστόσο, τις περισσότερες φορές η αύξηση της μια μετρικής οδηγεί στη μείωση της άλλης. Αυτό συμβαίνει καθώς για τον εντοπισμό περισσότερων πραγματικών εμποδίων πρέπει να αυξηθεί η ευαισθησία της εφαρμογής, με αποτέλεσμα να αυξάνονται και τα False Positives. Στο προηγούμενο παράδειγμα:



Εικόνα 21: Ισορροπία μεταξύ των μετρικών precision και recall

$$TP = 6, FP = 9, FN = 2$$

$$Precision = \frac{TP}{TP + FP} = \frac{6}{6 + 9} = 0{,}4 = 40\%$$

Δηλαδή το 40% των εμποδίων που εντοπίστηκαν ήταν πραγματικά.

$$Recall = \frac{TP}{TP + FN} = \frac{6}{6 + 2} = 0{,}75 = 75\%$$

Δηλαδή εντοπίστηκε το 75% των συνολικών πραγματικών εμποδίων.

Αυτή η αντίθετη φύση των 2 μετρικών οδηγεί στη χρήση μιας νέας μετρικής ώστε να βρεθεί η καλύτερη ισορροπία μεταξύ τους. Μια από τις πιο διαδεδομένες είναι η F score. Η F score είναι ο αρμονικός μέσος των precision και recall και δίνεται από τον παρακάτω τύπο[41]:

$$F_\beta = (1 + \beta^2)\frac{Precision \cdot Recall}{\beta^2 Precision + Recall}$$

Τυπικά η F score χρησιμοποιείται για β=1 και αποκαλείται F1 Score. Επομένως:

$$F1\ score = (1 + 1^2)\frac{Precision \cdot Recall}{1^2 Precision + Recall} = 2\frac{Precision \cdot Recall}{Precision + Recall}$$

Δηλαδή:

$$F1\ score = \frac{2TP}{2TP + FP + FN}$$

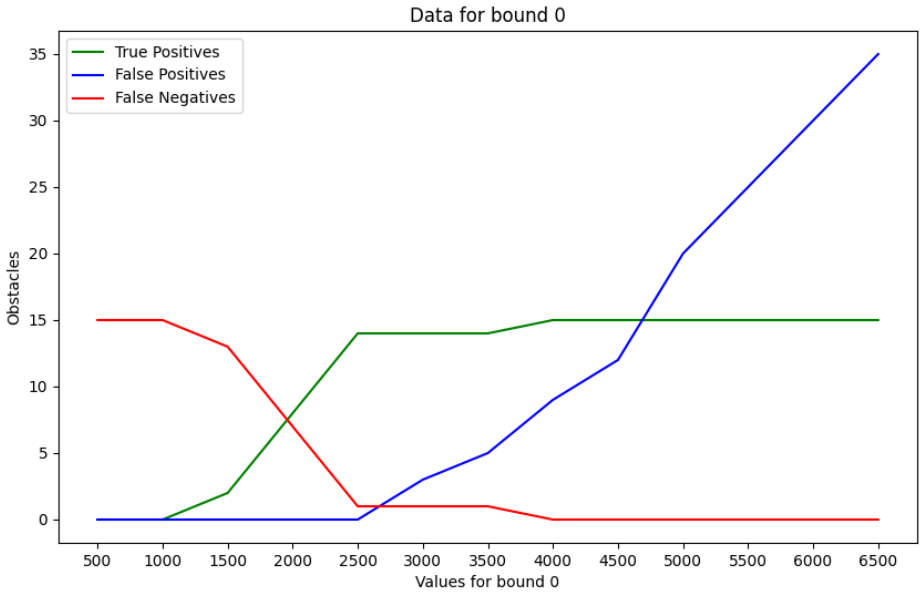
Αυτή είναι η μετρική που θα χρησιμοποιηθεί.

7.4.2 Παράμετροι εφαρμογής

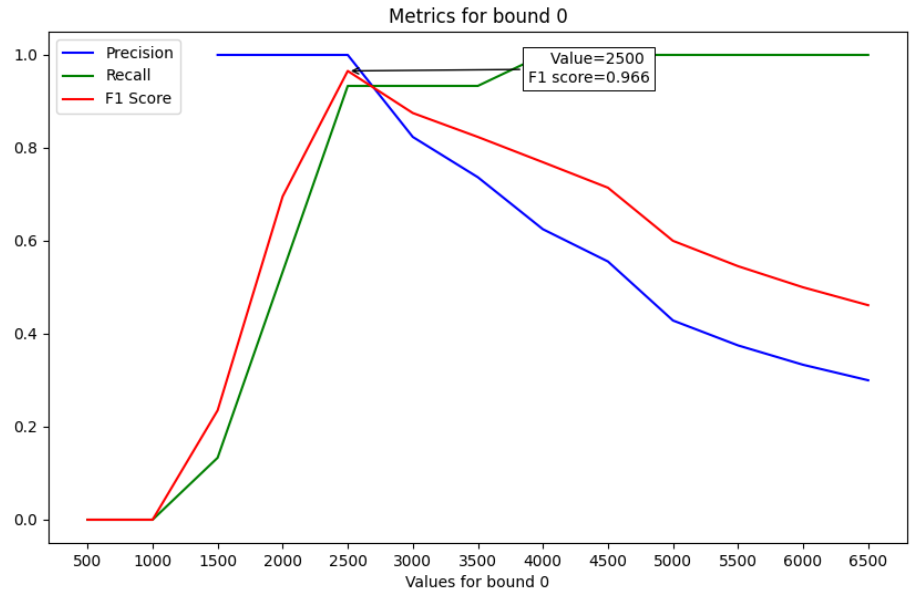
7.4.2.1 Low Bounds

Η παράμετρος Low Bounds καθορίζει τα όρια στα οποία αποφασίζεται ότι εντοπίστηκε εμπόδιο. Αποτελείται από 4 τιμές, μια για κάθε γραμμή του πλέγματος. Για παράδειγμα, αν η τιμή LowBounds[0] ισούται με 3000 και υπολογιστεί σε κάποιο κελί της πρώτης γραμμής μέσος όρος αποστάσεων μικρότερος από 3000 ο Αλγόριθμος θα αποφασίσει ότι σε αυτό το κελί υπάρχει εμπόδιο. Οι τιμές αυτές χρησιμοποιούνται και για το χρωματισμό των σημείων στο χάρτη βάθους.

Για τα χαμηλά όρια έγιναν συνολικά 52 δοκιμές, 13 σε κάθε γραμμή στο εύρος [500,1000,…,6500]. Οι δοκιμές είχαν μέση διάρκεια περίπου 10 λεπτά και έγιναν σε ένα μονοπάτι με 15 εμπόδια για κάθε όριο. Παρακάτω φαίνονται τα διαγράμματα των δεδομένων και των μετρικών για το κάθε όριο:

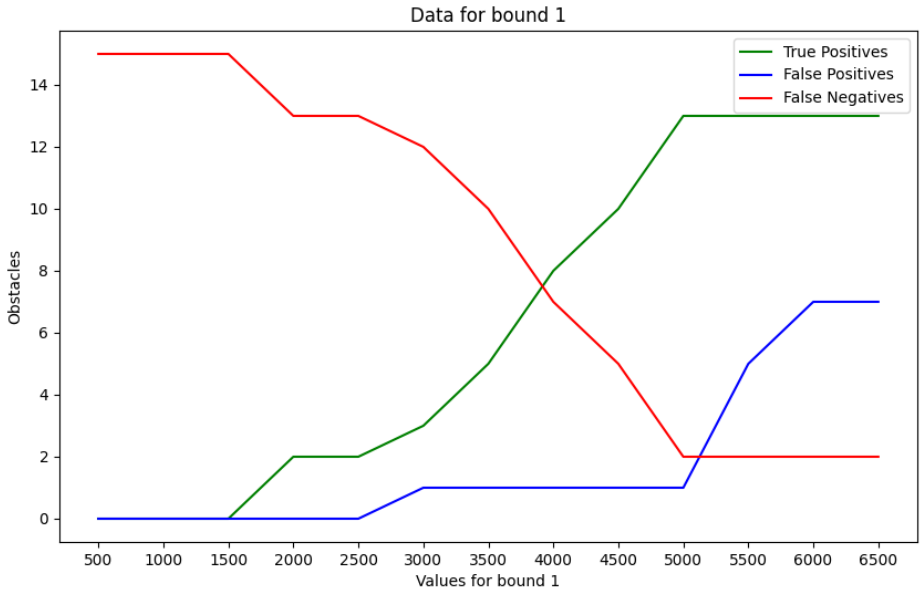


Εικόνα 22: Διάγραμμα δεδομένων για την παράμετρο LowBounds[0]

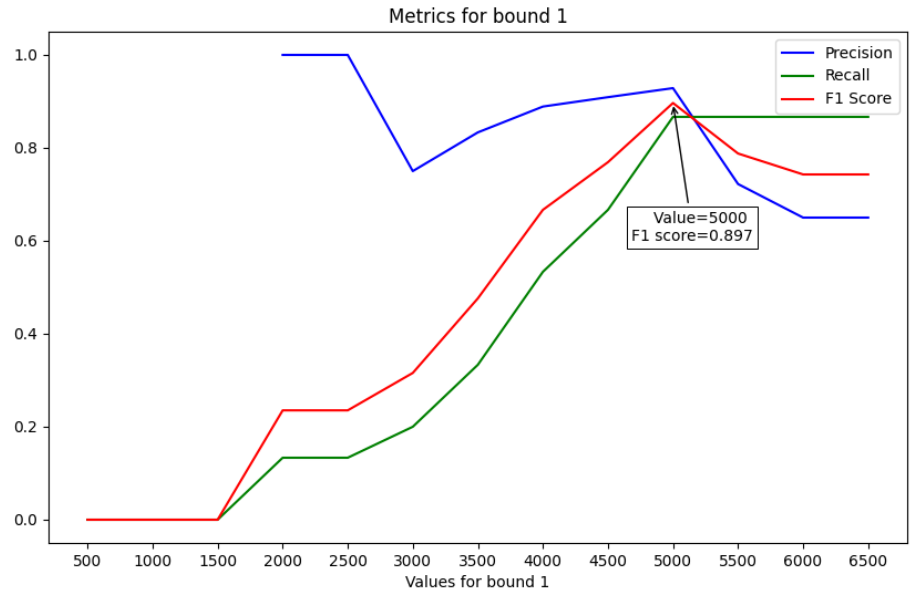


Εικόνα 23: Διάγραμμα μετρικών για την παράμετρο LowBounds[0]

Το όριο 0 είναι το όριο της πρώτης γραμμής και παρουσιάζει βέλτιστο εντοπισμό εμποδίων όταν έχει τιμή 2500 χιλιοστά. Ο εντοπισμός όλων των εμποδίων γίνεται στα 4000 χιλιοστά. Ωστόσο, από τα 3000 χιλιοστά η εφαρμογή αρχίζει να προειδοποιεί το χρήστη για εμπόδια που δεν υπάρχουν (false positives) με αποτέλεσμα να πέφτει η τιμή του Precision και του F1 Score.

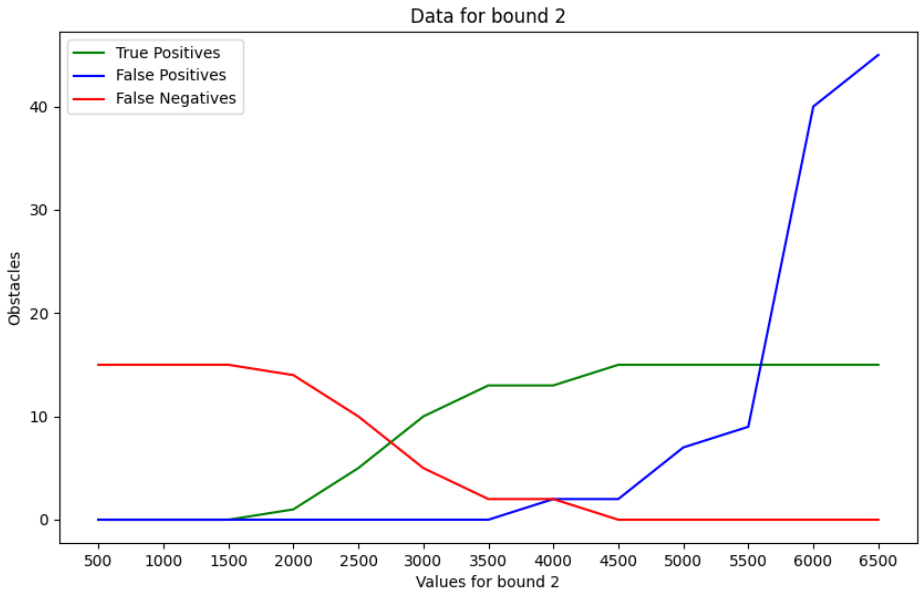


Εικόνα 24:Διάγραμμα δεδομένων για την παράμετρο LowBounds[1]

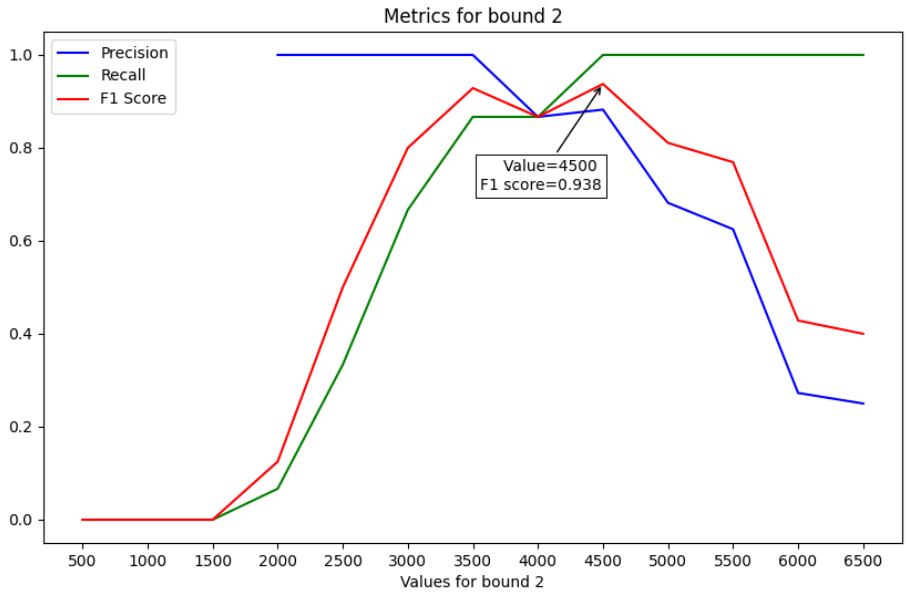


Εικόνα 25: Διάγραμμα μετρικών για την παράμετρο LowBounds[1]

Η βέλτιστη τιμή για τη δεύτερη γραμμή του πλέγματος είναι τα 5000 χιλιοστά. Σε αυτή την τιμή εντοπίζονται επιτυχώς 13 από τα 15 εμπόδια ενώ υπάρχει μόλις 1 false positive. Αξίζει να σημειωθεί πως 2 από τα προκαθορισμένα εμπόδια δεν εντοπίστηκαν ποτέ.

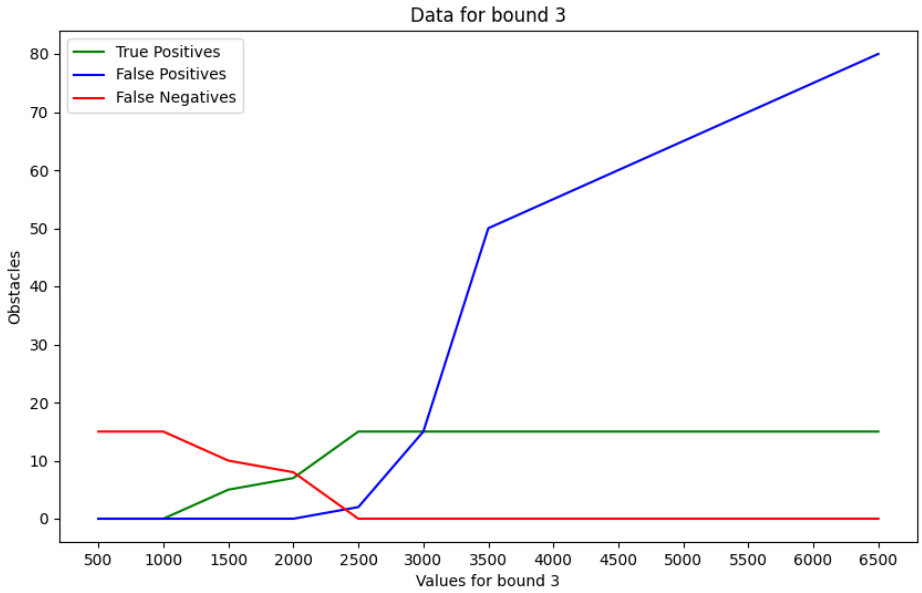


Εικόνα 26: Διάγραμμα δεδομένων για την παράμετρο LowBounds[2]

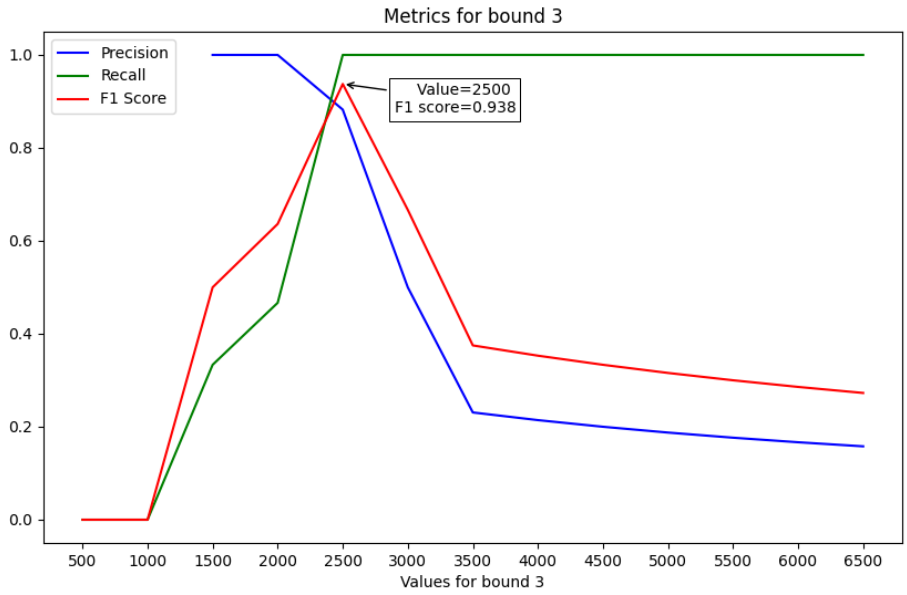


Εικόνα 27:Διάγραμμα μετρικών για την παράμετρο LowBounds[2]

Για το όριο της τρίτης γραμμής επιλέχθηκε η τιμή των 4500 χιλιοστών. Σε αυτή την τιμή εντοπίζονται όλα τα εμπόδια, ενώ υπάρχουν 2 false positives. Αξίζει να σημειωθεί πως η τιμή f1 score στα 3500 χιλιοστά είναι αρκετά υψηλή, δηλαδή 0,297. Αυτό συμβαίνει καθώς εντοπίζονται τα 13 από τα 15 εμπόδια ενώ δεν υπάρχει κανένα false positive.



Εικόνα 28: Διάγραμμα δεδομένων για την παράμετρο LowBounds[3]

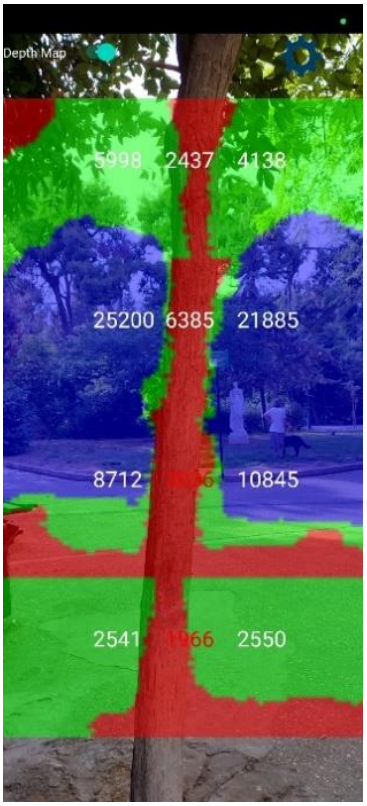


Εικόνα 29: Διάγραμμα μετρικών για την παράμετρο LowBounds[3]

Για το όριο της τελευταίας γραμμής επιλέχθηκαν τα 2500 χιλιοστά. Σε αυτό το σημείο εντοπίζονται όλα τα εμπόδια ενώ υπάρχουν 2 false positives. Από αυτό το σημείο και μετά η τιμή του Precision πέφτει αρκετά απότομα λόγω του μεγάλου αριθμού false positives. Αυτά εμφανίζονται καθώς η εφαρμογή εντοπίζει συνέχεια το έδαφος σαν εμπόδιο πολλές φορές στη διάρκεια της κάθε δοκιμής.

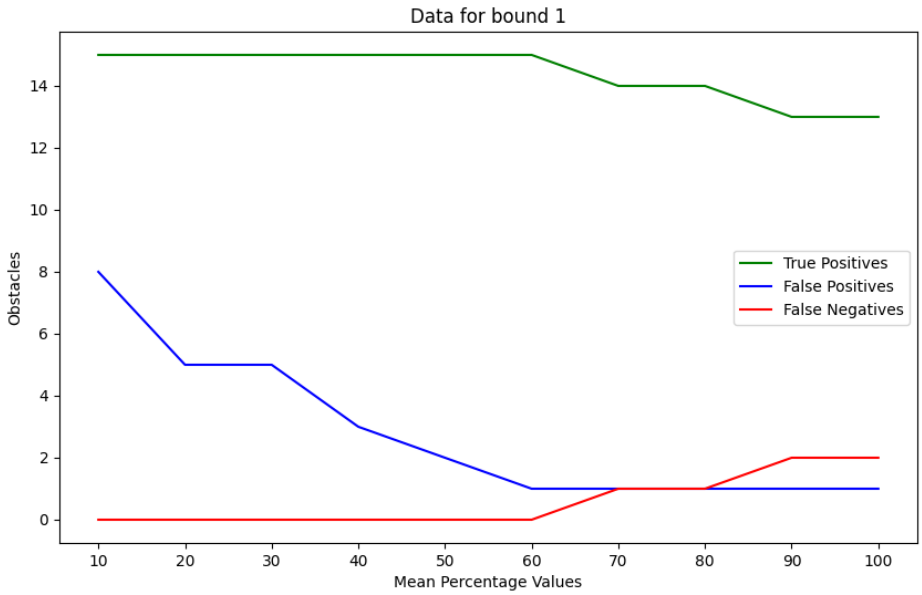
7.4.2.2 Mean percentage

Όπως σημειώθηκε νωρίτερα εμφανίζεται δυσκολία στον εντοπισμό μερικών εμποδίων της δεύτερης γραμμής. Αυτή οφείλεται στο μικρό σχετικά πλάτος αυτών των εμποδίων σε συνδυασμό με μεγάλες αποστάσεις βάθους πίσω του. Η χρήση του μέσου όρου των αποστάσεων στο κελί λαμβάνει υπόψη τις μικρές τιμές βάθους του εμποδίου σε συνδυασμό με τις μεγάλες τιμές βάθους γύρω από αυτό και επιστρέφει μια μέτρηση που παραπέμπει σε έλλειψη εμποδίου παρά την ύπαρξή του.

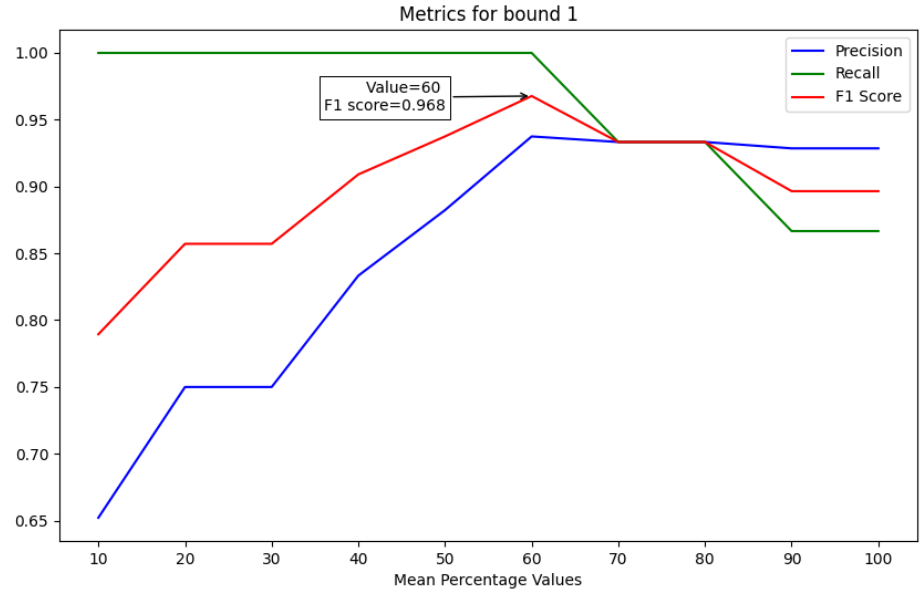


Εικόνα 30: Παράδειγμα αδυναμίας εντοπισμού εμποδίου μικρού πλάτους στη δεύτερη γραμμή

Για την αντιμετώπιση του προβλήματος αυτού τροποποιήθηκε ο Αλγόριθμος υπολογισμού βάθους μόνο στα κελιά της δεύτερης γραμμής. Συγκεκριμένα, οι τιμές βάθους του κάθε pixel εισάγωνται σε ένα πίνακα και ταξινομούνται σε αύξουσα σειρά. Έπειτα επιλέγεται ένα μέρος του πίνακα για τον υπολογισμό του μέσου όρου. Το μέρος αυτό καθορίζεται σαν ποσοστό από τη μεταβλητή mean percentage. Εάν αυτή έχει τιμή 50, τότε λαμβάνεται υπόψη μόνο το 50% των pixel μέσα στο κελί, δηλαδή τα μισά. Η ταξινόμηση του πίνακα εξασφαλίζει ότι τα σημεία που συμμετέχουν στον υπολογισμό του μέσου όρου είναι αυτά με τη μικρότερη τιμή βάθους. Για την εν λόγω παράμετρο έγιναν 10 δοκιμές στο εύρος [10,20,…,90,100] με το όριο της δεύτερης γραμμής να παραμένει στα 5000 χιλιοστά που υπολογίστηκαν παραπάνω. Τα αποτελέσματα φαίνονται παρακάτω:

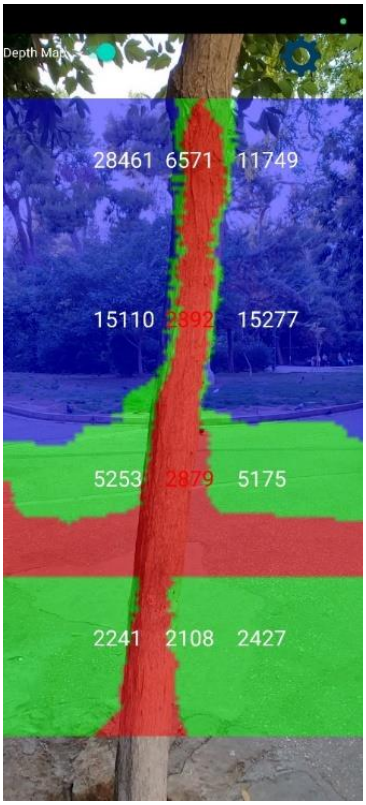


Εικόνα 31: Διάγραμμα δεδομένων για την παράμετρο mean percentage



Εικόνα 32: Διάγραμμα μετρικών για την παράμετρο mean percentage

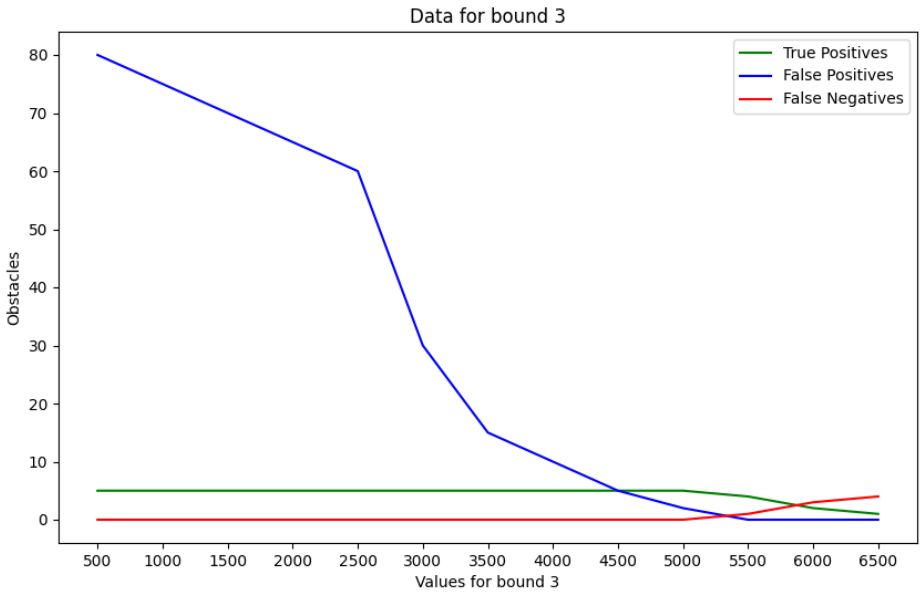
Όπως φαίνεται από το διάγραμμα δεδομένων, ο εντοπισμός τέτοιου είδους εμποδίων γίνεται για τις τιμές [0…60]. Ωστόσο στην τιμή 60 ελαχιστοποιούνται τα false positives με αποτέλεσμα αυτή να παρουσιάζει το μεγαλύτερο f1 score με τιμή 0.968. Εδώ παρατηρείται αύξηση της τιμής σε σχέση με αυτή που είχε υπολογιστεί παραπάνω όπου ήταν ίση με 0.897.



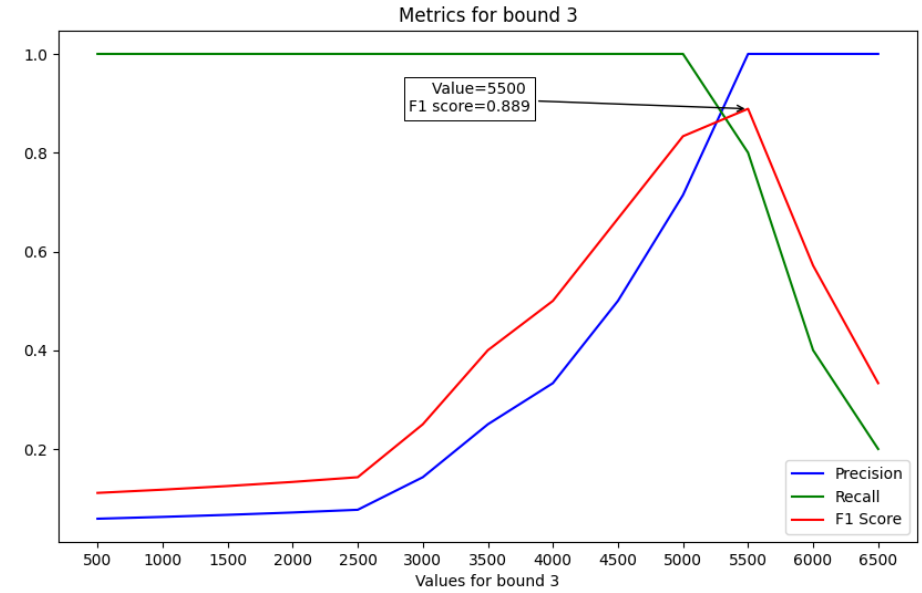
Εικόνα 33: Παράδειγμα επιτυχούς εντοπισμού εμποδίου μικρού πλάτους στη δεύτερη γραμμή

7.4.2.3 High Bounds

Η παράμετρος High Bounds καθορίζει το όριο στο οποίο αποφασίζεται ότι εντοπίστηκε γκρεμός. Αποτελείται από 4 τιμές, μια για κάθε γραμμή του πλέγματος. Ωστόσο για τον εντοπισμό γκρεμών χρησιμοποιείται μόνο η τελευταία γραμμή, δηλαδή η τιμή HighBounds[3]. Για παράδειγμα, αν η τιμή HighBounds[3] ισούται με 3000 και υπολογιστεί σε κάποιο κελί της τελευταίας γραμμής μέσος όρος αποστάσεων μεγαλύτερος από 3000, ο Αλγόριθμος θα αποφασίσει ότι σε αυτό το κελί υπάρχει γκρεμός. Οι υπόλοιπες τιμές χρησιμοποιούνται μόνο για το χρωματισμό των σημείων στο χάρτη βάθους. Για την παράμετρο αυτή έγιναν 13 δοκιμές στο εύρος [500,1000,…,6000,6500]. Τα αποτελέσματα φαίνονται παρακάτω:



Εικόνα 34: Διάγραμμα δεδομένων για την παράμετρο HighBounds[3]



Εικόνα 35: Διάγραμμα δεδομένων για την παράμετρο HighBounds[3]

Ο εντοπισμός όλων των γκρεμών είναι επιτυχής μέχρι την τιμή 5000. Ωστόσο για τις σχετικά μικρές τιμές εμφανίζεται ένας τεράστιος αριθμός false positives. Η καλύτερη τιμή του f1 score είναι τα 5500 χιλιοστά.

Στις τιμές HighBounds[0..2] δόθηκε τυπικά η διπλάσια τιμή των αντίστοιχων LowBounds[0..2].

7.4.2.4 Width Percentage

Κατά τη διάρκεια των δοκιμών παρατηρήθηκε ότι ο Αλγόριθμος εντόπιζε σαν εμπόδια τοίχους στα δεξιά και τα αριστερά του χρήστη. Αυτοί ωστόσο δε παρουσίαζαν κίνδυνο σύγκρουσης. Έτσι έπρεπε να αγνοηθεί ένα μέρος των σημείων στα αριστερά και τα δεξιά της οθόνης. Για το σκοπό αυτό δημιουργήθηκε η παράμετρος Width Percentage που καθορίζει το πλάτος του πλέγματος σε μορφή ποσοστού. Η τιμή 100 σημαίνει ότι χρησιμοποιείται το 100% του πλάτους της εικόνας, ενώ η τιμή 50 σημαίνει ότι το μισό πλάτος αγνοείται.

Εικόνα 36:Παράδειγμα των τιμών 100 και 60 της παραμέτρου width percentage

Για αυτή την παράμετρο δοκιμάστηκαν οι τιμές [10..100] και κρίθηκε ότι η τιμή 60 είναι η καλύτερη, καθώς εκεί σταματούν να εντοπίζονται εμπόδια έξω από το μονοπάτι του χρήστη.

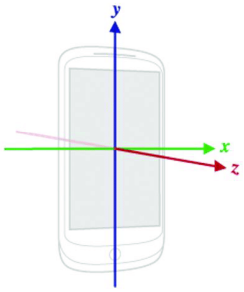
7.4.2.5 Μέγιστες γωνίες κλίσης

Αρχικά χρησιμοποιήθηκε μια διαφορετική προσέγγιση η οποία μετέβαλλε σε πραγματικό χρόνο τα όρια των γραμμών του πλέγματος, δηλαδή τις τιμές lowBounds και highBounds, σε συνάρτηση με τη γωνία κλίσης της συσκευής στους 3 άξονες. Η ιδέα πίσω από αυτό ήταν ότι τα εμπόδια που βρίσκονται στην ίδια απόσταση από την κάμερα εμφανίζονται με διαφορετική τιμή βάθους για διαφορετικές γωνίες κλίσης της συσκευής, λόγω της απόστασης που περιγράφεται από τη μέτρηση βάθους της βιβλιοθήκης ARCore(βλ. διάγραμμα στο κεφάλαιο 5.5.3). Ωστόσο αυτό επέτρεπε στο χρήστη να κρατά τη συσκευή σε παράλογους προσανατολισμούς, όπως για παράδειγμα με την κάμερα να κοιτά το έδαφος, με αποτέλεσμα ο εντοπισμός εμποδίων να είναι ανεπιτυχής. Αυτό οδήγησε στη δημιουργία των ακόλουθων 3 μεταβλητών με τις αντίστοιχες τιμές τους:

Μεταβλητή	Περιγραφή	Τιμή
maxY	Ορίζει το όριο κλίσης της συσκευής προς τα εμπρός σε μοίρες (γωνία μεταξύ των αξόνων y και z)	20
minY	Ορίζει το όριο κλίσης της συσκευής προς τα πίσω σε μοίρες (γωνία μεταξύ των αξόνων y και z)	-10
absZ	Ορίζει το όριο κλίσης της συσκευής προς τα δεξιά και τα αριστερά σε μοίρες (γωνία μεταξύ των αξόνων y και x)	20

Πίνακας 5: Περιγραφή των παραμέτρων ορίων κλίσης

Παρακάτω απεικονίζεται το σύστημα συντεταγμένων με Y=0, Z=0:



Εικόνα 37: Το σύστημα αξόνων της συσκευής

Οι τιμές που επιλέχθηκαν εξασφαλίζουν ότι η κάμερα της συσκευής καταγράφει τη διαδρομή μπροστά από το χρήστη. Υπενθυμίζεται ότι εάν η συσκευή βρεθεί εκτός των επιτρεπόμενων ορίων, ο Αλγόριθμος σταματά τον εντοπισμό εμποδίων και ενημερώνει το χρήστη με ήχο και δόνηση.

Η προσέγγιση της δυναμικής μεταβολής των ορίων απορρίφθηκε, καθώς κατά τη διάρκεια των δοκιμών για τον καθορισμό όλων των μεταβλητών αποδείχθηκε πως δεν είναι αναγκαία. Η απλή οριοθέτηση των γωνιών κλίσης ήταν αρκετή για τον επιτυχή εντοπισμό όλων των εμποδίων.

7.4.2.6 Refresh Rate

Η παράμετρος Refresh Rate καθορίζει το ρυθμό εκτέλεσης του Αλγόριθμου σε δευτερόλεπτα. Ο ρυθμός αυτός καθορίζει τόσο την απόδοση όσο και την ενεργειακή κατανάλωση της εφαρμογής. Η χρήση μικρής τιμής μπορεί να οδηγήσει στον μη έγκαιρο εντοπισμό εμποδίων, καθώς ο έλεγχος δε γίνεται αρκετά συχνά. Αντίθετα, η χρήση πολύ μεγάλης τιμής μπορεί να οδηγήσει στον κορεσμό του επεξεργαστή, καθώς απαιτείται πολύ μεγάλη υπολογιστική ισχύς για την πολλαπλή επιτυχή εκτέλεση του Αλγόριθμου. Η μεγάλη αυτή υπολογιστική ισχύς συνεπάγεται και τη χρήση περισσότερης ενέργειας από τη μπαταρία της συσκευής.

Για την παρακολούθηση της επίδοσης της συσκευής χρησιμοποιήθηκε ως μέτρο η λειτουργία της κάμερας. Αξίζει να σημειωθεί ότι η εικόνα της κάμερας δεν ανανεώνεται με τον ίδιο ρυθμό που εκτελείται ο Αλγόριθμος, αλλά ανανεώνεται ανεξάρτητα με σταθερό ρυθμό. Ο Αλγόριθμος χρησιμοποιεί την πιο πρόσφατη εικόνα που κατέγραψε η κάμερα για τον υπολογισμό της εικόνας βάθους. Στην εν λόγω συσκευή, επιλέχθηκε ο μέγιστος ρυθμός ανανέωσης της κάμερας που επιτρέπει η βιβλιοθήκη Sceneform, δηλαδή 30 καρέ το δευτερόλεπτο. Ωστόσο, η αυξημένη

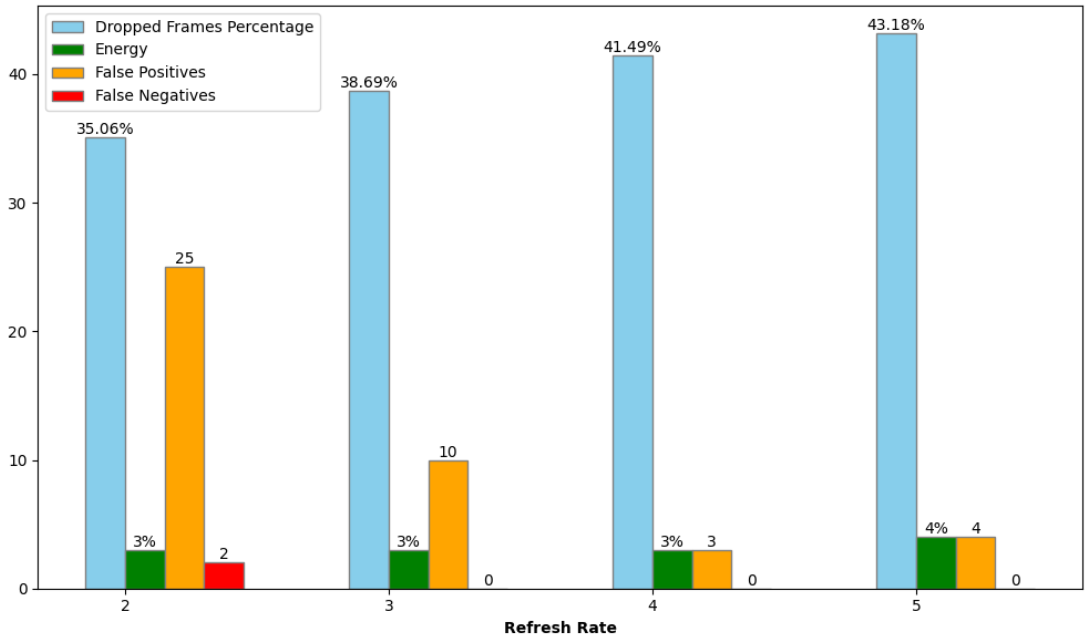
ανάγκη για υπολογιστική ισχύ που παρουσιάζεται από τον Αλγόριθμο οδηγεί στην αδυναμία της συσκευής να διατηρήσει το σταθερό ρυθμό λειτουργίας της κάμερας. Έτσι, για κάθε τιμή της παραμέτρου Refresh Rate μετρήθηκαν τα καρέ που δεν καταγράφηκαν (Dropped Frames) και υπολογίστηκε ο συνολικός αριθμός καρέ που έπρεπε να εμφανιστούν (Theoretical Frames). Η χρήση αυτών των τιμών μπορεί να χρησιμοποιηθεί για τον υπολογισμό του εξής κλάσματος:

$$Dropped\% = \frac{Dropped\ Frames}{Theoretical\ Frames} * 100\%$$

Όσο μεγαλύτερη είναι η τιμή του κλάσματος, τόσο μεγαλύτερη είναι η πτώση της επίδοσης της συσκευής. Επομένως, εδώ αναζητείται η κατάλληλη τιμή της παραμέτρου, ώστε να επιτευχθεί ισορροπία μεταξύ επίδοσης και ενεργειακής κατανάλωσης διατηρώντας τα αποτελέσματα εντοπισμού εμποδίων που έχουν ήδη υπολογιστεί.

Υπενθυμίζεται ότι ο ρυθμός ανανέωσης συσχετίζεται με τον αριθμό καταστάσεων των Μηχανών Καταστάσεων. Αυτό σημαίνει ότι η τιμή 1 δεν είναι επιτρεπτή, καθώς οι Μηχανές Καταστάσεων χρειάζονται τουλάχιστον 2 καταστάσεις, μια για έλλειψη και μια για ύπαρξη εμποδίου, γκρεμού η παραβίασης των ορίων κλίσης.

Έγιναν δοκιμές για τις τιμές 2,3,4,5 και τα αποτελέσματα φαίνονται παρακάτω:



Εικόνα 38: Διάγραμμα μετρικών για την παράμετρο refresh rate

Στο παραπάνω διάγραμμα απεικονίζονται τα εξής δεδομένα για τις τιμές της μεταβλητής Refresh Rate:

- Dropped Frames Percentage: Το ποσοστό των χαμένων καρέ.
- Energy: Το ποσοστό της μπαταρίας που χρησιμοποιήθηκε κατά τη δοκιμή της εκάστοτε τιμής.

- False Positives: Ο συνολικός αριθμός προειδοποιήσεων που έδωσε ο Αλγόριθμος χωρίς την ύπαρξη εμποδίων η γκρεμών.
- False Negatives: Ο συνολικός αριθμός εμποδίων και γκρεμών που δε κατάφερε να εντοπίσει ο Αλγόριθμος.

Όπως φαίνεται, για την τιμή 2 ο Αλγόριθμος δε καταφέρνει να εντοπίσει 2 εμπόδια καθώς ο έλεγχος δεν ήταν αρκετά γρήγορος. Όλα τα εμπόδια και οι γκρεμοί εντοπίζονται επιτυχώς για ρυθμούς ανανέωσης από 3 φορές το δευτερόλεπτο και πάνω. Ωστόσο, για την τιμή 3 εμφανίζονται 10 false positives. Αυτό συμβαίνει λόγω της συσχέτισης του ρυθμού ανανέωσης με τις Μηχανές Καταστάσεων. Κάθε μηχανή έχει σε αυτή την περίπτωση 3 καταστάσεις. Αυτές δεν είναι αρκετές, καθώς η ευαισθησία του Αλγόριθμου σε λανθασμένες τιμές βάθους είναι πολύ μεγάλη. Το ίδιο παρατηρήθηκε και για τη Μηχανή κατάστασης της δόνησης, η οποία έδινε οδηγία να κρατηθεί όρθια η συσκευή λόγω των μετρήσεων του επιταχυνσιόμετρου χωρίς αυτό να είναι απαραίτητο.

Η τιμή 4 φαίνεται να είναι η πιο ωφέλιμη. Παρά το σχετικά υψηλό ποσοστό χαμένων καρέ (41.49%), εντοπίζονται όλα τα εμπόδια και οι γκρεμοί και ελαχιστοποιούνται τα false positives. Η τιμή 5 έχει παρόμοια αποτελέσματα. Ωστόσο αρχίζει να ανεβαίνει το ποσό της ενέργειας που χρειάζεται η εφαρμογή.

7.5 Σύνοψη

Οι παράμετροι που επιλέχθηκαν αναφέρονται συγκεντρωμένα παρακάτω:

- LowBounds = [2500,5000,4500,2000]
- Mean Percentage = 60
- HighBounds = [5000,10000,9000,4000]
- Width Percentage = 60
- maxY=20, minY=-10, absZ=20
- Refresh Rate = 4

Με αυτή την ομάδα παραμέτρων έγινε δοκιμή για την αξιολόγηση της εφαρμογής. Η δοκιμή διήρκησε 7 λεπτά και εντοπίστηκαν επιτυχώς όλα τα εμπόδια και οι γκρεμοί, ενώ υπήρξαν μόνο 2 λανθασμένες προειδοποιήσεις για γκρεμούς λόγω σφαλμάτων στον υπολογισμό βάθους της βιβλιοθήκης ARCore.

Κεφάλαιο 8
Επίλογος

Σε αυτό το κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας καθώς και πιθανές μελλοντικές προεκτάσεις για την επέκταση λειτουργίας της εφαρμογής.

8.1 Αξιολόγηση

Συνολικά, προτάθηκε και αναπτύχθηκε μια εφαρμογή που κάνει χρήση των λειτουργιών εκτίμησης βάθους της Επαυξημένης Πραγματικότητας με σκοπό την αποφυγή εμποδίων και γκρεμών στο μονοπάτι του χρήστη. Ο Αλγόριθμος της εφαρμογής χρησιμοποιεί τον μέσο όρο ή, σε ορισμένες περιπτώσεις, έναν σταθμισμένο μέσο όρο των αποστάσεων βάθους που προκύπτουν. Αυτές χρησιμοποιούνται για την ανίχνευση εμποδίων και γκρεμών, με στόχο την προειδοποίηση του χρήστη μέσω κατάλληλων ηχητικών μηνυμάτων και δονήσεων.

Στο παρελθόν, έχουν υλοποιηθεί παρόμοια παραδείγματα έρευνας, με κοινό παρονομαστή την ανάπτυξη σύνθετων συστημάτων και τεχνολογιών από την αρχή, όπως τα βαθιά νευρωνικά δίκτυα, ειδικά προσαρμοσμένα για τη συγκεκριμένη περίπτωση[56][57]. Παράλληλα, χρησιμοποιείται μεγάλος αριθμός αισθητήρων, όπως οι αισθητήρες LiDAR, οι υπερηχητικοί αισθητήρες[58] και μικρόφωνα[56]. Επιπροσθέτως, χρησιμοποιείται εξοπλισμός όπως ηλεκτρονικοί υπολογιστές και εξωτερικές κάμερες για την απόκτηση των δεδομένων και τη διαχείριση του υπολογιστικού φόρτου εργασίας που απαιτεί η εκτέλεση του εκάστοτε Αλγόριθμου[58].

Σε αντίθεση με τις προηγούμενες ερευνητικές προσεγγίσεις, ένα σημαντικό πλεονέκτημα της εφαρμογής είναι η ανεξαρτησία της από περίπλοκα ή κοστοβόρα συστήματα για τον εντοπισμό εμποδίων. Συγκεκριμένα, χρησιμοποιεί την έτοιμη βιβλιοθήκη ARCore, γεγονός που καθιστά περιττή τη δημιουργία και εκπαίδευση ενός εξειδικευμένου και σύνθετου νευρωνικού δικτύου. Επιπλέον, αποφεύγεται η χρήση εξειδικευμένων αισθητήρων, καμερών και υπολογιστών για τον υπολογισμό βάθους. Αντιθέτως, η εφαρμογή μπορεί να λειτουργήσει σε καθημερινές φορητές συσκευές με λειτουργικό σύστημα Android, οι οποίες βρίσκονται ήδη στα χέρια πολλών χρηστών. Αποδείχθηκε επίσης ότι η υπολογιστική ισχύς που απαιτείται δεν είναι υψηλή, επιτρέποντας την εκτέλεση της εφαρμογής ακόμα και σε παλαιότερα μοντέλα συσκευών. Παράλληλα, η εφαρμογή προσφέρει στον χρήστη τη δυνατότητα να παραμετροποιήσει όλες τις διαθέσιμες μεταβλητές ώστε να επιτύχει το καλύτερο δυνατό αποτέλεσμα στη δική του συσκευή. Τέλος, ως επιπλέον λειτουργία, παροτρύνει τον χρήστη να κρατήσει τη συσκευή σωστά για την αποτελεσματικότερη εκτέλεση του αλγορίθμου, ενισχύοντας έτσι την αλληλεπίδραση μεταξύ του χρήστη και της συσκευής.

8.2 Προτάσεις για μελλοντικές επεκτάσεις

Μπορούν να γίνουν αρκετές προσθήκες και αλλαγές για την επέκταση των λειτουργιών της εφαρμογής που αξίζουν να σημειωθούν.

8.2.1 Δοκιμές σε μεγαλύτερο σύνολο συσκευών

Μέχρι στιγμής, η συνολική έρευνα και παραμετροποίηση πραγματοποιήθηκε αποκλειστικά στη συγκεκριμένη συσκευή που αναλύθηκε στην παράγραφο 7.1. Ωστόσο, κάθε συσκευή παρουσιάζει μοναδικές διαφορές στην υπολογιστική ισχύ και στα εξαρτήματά της, καθώς και στις λειτουργίες τους. Για παράδειγμα, η κάμερα ενός μοντέλου ενδέχεται να διαθέτει ευρύτερο πεδίο όρασης, με αποτέλεσμα να απαιτούνται διαφορετικές τιμές για τις παραμέτρους LowBounds, HighBounds, Mean Percentage και Width Percentage, ενώ μια άλλη μονάδα μπορεί να προσφέρει υψηλότερη υπολογιστική ισχύ που επιτρέπει τη χρήση υψηλότερου ρυθμού ανανέωσης του Αλγόριθμου. Έτσι θα ήταν συνετή η πραγματοποίηση δοκιμών σε ένα ευρύτερο φάσμα συσκευών που διατίθενται στην αγορά. Αυτό θα συνέβαλε στην καλύτερη κατανόηση των διαφορών στην υπολογιστική ισχύ και στις λειτουργίες τους, βοηθώντας έτσι στην πιο εκτεταμένη εφαρμογή των παραμέτρων. Επίσης, η συμπερίληψη μιας λειτουργίας που επιτρέπει την εισαγωγή ομάδων παραμέτρων θα διευκόλυνε σημαντικά τη διαχείριση αυτού του έργου, επιτρέποντας την οργάνωση και την εφαρμογή προσαρμοσμένων ρυθμίσεων σε διαφορετικές ομάδες συσκευών.

8.2.2 Ενσωμάτωση τεχνητής νοημοσύνης για πρόσθετες λειτουργίες

Η ενσωμάτωση τεχνητής νοημοσύνης σε εφαρμογές μπορεί να προσφέρει σημαντικές επιπλέον λειτουργίες και δυνατότητες. Για παράδειγμα, είναι δυνατή η χρήση βιβλιοθηκών ανοιχτού κώδικα όπως η OpenCV[42], η οποία είναι μια προηγμένη βιβλιοθήκη όρασης υπολογιστών. Η OpenCV επιτρέπει την ανάλυση και την επεξεργασία εικόνας σε πραγματικό χρόνο και μπορεί να ενσωματωθεί σε λειτουργικό σύστημα Android[43]. Άλλες τέτοιες βιβλιοθήκες είναι η Tensorflow Lite[44][45] και η ML Kit της Google[46][47]. Με αυτές τις τεχνολογίες, είναι δυνατή η ανάπτυξη λειτουργιών όπως ο εντοπισμός σκαλοπατιών, φαναριών ή διαβάσεων πεζών, βελτιώνοντας έτσι την ασφάλεια του χρήστη κατά τη χρήση της εφαρμογής.

8.2.3 Επέκταση σε άλλα λειτουργικά συστήματα

Το λειτουργικό σύστημα Android είναι ένα ευρέως διαδεδομένο λειτουργικό σύστημα που χρησιμοποιείται από ένα μεγάλο αριθμό χρηστών. Ωστόσο, η επέκταση της εφαρμογής σε άλλα λειτουργικά συστήματα όπως το iOS της Apple[48] θα επιτρέψει στους χρήστες των συσκευών Apple να χρησιμοποιούν την εφαρμογή προσφέροντας πρόσβαση σε μεγαλύτερο κοινό και διευρύνοντας έτσι τη βάση χρηστών.

8.2.4 Συμπερίληψη λειτουργίας για εσωτερικούς χώρους

Όπως αναφέρθηκε στο κεφάλαιο 7.3, η τρέχουσα έκδοση της εφαρμογής προϋποθέτει ότι ο χρήστης τη χρησιμοποιεί σε εξωτερικούς χώρους. Επομένως, είναι συνετό να διεξαχθεί έρευνα για την προσαρμογή των παραμέτρων της εφαρμογής ώστε να επιτρέπεται η ομαλή λειτουργία της και σε εσωτερικούς χώρους, όπως σπίτια ή μουσεία.

8.2.5 Χρήση περισσότερων αισθητήρων

Μια ακόμα σημαντική πτυχή έρευνας είναι η χρήση συσκευών με εξειδικευμένους αισθητήρες υπολογισμού βάθους, όπως οι κάμερες ToF (time-of-flight). Κατά τη διάρκεια των δοκιμών παρατηρήθηκε ότι ο Αλγόριθμος της βιβλιοθήκης ARCore

είναι επιρρεπής σε σφάλματα. Για το λόγο αυτό χρησιμοποιήθηκαν Μηχανές κατάστασης ώστε να διασφαλιστεί η ύπαρξη γκρεμού ή εμποδίου καθώς και η παραβίαση των ορίων κλίσης. Ένας αισθητήρας ToF θα μπορούσε να εξαλείψει τα σφάλματα αυτά αλλά και να δώσει πιο ακριβείς τιμές μετρήσεων. Επιπλέον, θα διευκόλυνε τον εντοπισμό εμποδίων μικρότερου μεγέθους και την ακριβή ανίχνευση σκαλοπατιών, βελτιώνοντας σημαντικά την ακρίβεια και την απόδοση της εφαρμογής σε διαφορετικά περιβάλλοντα χρήσης.

8.3 Κατακλείδα

Κατά τη διάρκεια της διπλωματικής εργασίας, εξετάστηκε εκτενώς η τεχνολογία της Επαυξημένης Πραγματικότητας. Αναπτύχθηκε μια πρότυπη εφαρμογή για συσκευές Android για τον εντοπισμό εμποδίων και γκρεμών, η οποία δύναται να αποδειχθεί ιδιαίτερα χρήσιμη σε άτομα που πάσχουν από προβλήματα όρασης. Αξιοποιήθηκε η δυνατότητα εκτίμησης του βάθους που προσφέρεται από την Επαυξημένη Πραγματικότητα για την δημιουργία Αλγόριθμου εντοπισμού εμποδίων ή γκρεμών και την υλοποίησή του. Αναλύθηκαν τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του κώδικα της πρότυπης αυτής εφαρμογής και παραστάθηκαν επεξηγήσεις για τη λειτουργία του και τον τρόπο εκτέλεσής του. Δημιουργήθηκαν κατάλληλες παράμετροι για τον προσδιορισμό των εμποδίων και γκρεμών, βασισμένες σε γνωστές μετρικές σύγκρισης και πραγματοποιήθηκε αναλυτική έρευνα για τις βέλτιστες τιμές αυτών των παραμέτρων σε πραγματικό σενάριο χρήσης. Τέλος, παρουσιάστηκαν τα αποτελέσματα αυτής της έρευνας τα οποία αναδεικνύουν τη υψηλή απόδοση της εφαρμογής στον πραγματικό κόσμο, προσδίδοντας έτσι μια κατεύθυνση προς την υποβοήθηση ατόμων με προβλήματα όρασης κάνοντας χρήση προηγμένων μεθόδων επικοινωνίας Ανθρώπου - Μηχανής.

Βιβλιογραφία

[1] Carmigniani, Julie, and Borko Furht. "Augmented reality: an overview." *Handbook of augmented reality* (2011): 3-46.

[2] Carmigniani, Julie, et al. "Augmented reality technologies, systems and applications." *Multimedia tools and applications* 51 (2011): 341-377.

[3] https://worldblindunion.org/

[4] Craig, Alan B. "Understanding augmented reality: Concepts and applications." (2013).

[5] Aguero, Marlon, et al. "Design and implementation of a connection between augmented reality and sensors." *Robotics* 9.1 (2020): 3.

[6] Xu, Guochang, and Yan Xu. *GPS*. Springer-Verlag Berlin Heidelberg, 2007.

[7] Passaro, Vittorio MN, et al. "Gyroscope technology and applications: A review in the industrial perspective." *Sensors* 17.10 (2017): 2284.

[8] Faisal, I. Arun, T. Waluyo Purboyo, and A. Siswo Raharjo Ansori. "A review of accelerometer sensor and gyroscope sensor in IMU sensors on motion capture." *J. Eng. Appl. Sci* 15.3 (2019): 826-829.

[9] Koller, Dieter, et al. "Real-time vision-based camera tracking for augmented reality applications." *Proceedings of the ACM symposium on Virtual reality software and technology*. 1997.

[10] Li, Nanxi, et al. "A progress review on solid-state LiDAR and nanophotonics-based LiDAR sensors." *Laser & Photonics Reviews* 16.11 (2022): 2100511.

[11] Kalia, Megha, Nassir Navab, and Tim Salcudean. "A real-time interactive augmented reality depth estimation technique for surgical robotics." *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[12] Cordeiro, Diogo, Nuno Correia, and Rui Jesus. "ARZombie: A mobile augmented reality game with multimodal interaction." *2015 7th International Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN)*. IEEE, 2015.

[13] Koulouris, Dionysios, Andreas Menychtas, and Ilias Maglogiannis. "An IoT-enabled platform for the assessment of physical and mental activities utilizing augmented reality exergaming." *Sensors* 22.9 (2022): 3181.

[14] Althoff, Tim, Ryen W. White, and Eric Horvitz. "Influence of Pokémon Go on physical activity: study and implications." *Journal of medical Internet research* 18.12 (2016): e315.

[15] Garay-Cortes, Juan, and Alvaro Uribe-Quevedo. "Location-based augmented reality game to engage students in discovering institutional landmarks." *2016 7th*

International Conference on Information, Intelligence, Systems & Applications (IISA). IEEE, 2016.

[16] Nee, Andrew YC, et al. "Augmented reality applications in design and manufacturing." *CIRP annals* 61.2 (2012): 657-679.

[17] Pratt, Philip, and Asit Arora. "Transoral robotic surgery: image guidance and augmented reality." *ORL* 80.3-4 (2018): 204-212.

[18] Abhari, Kamyar, et al. "Training for planning tumour resection: augmented reality and human factors." *IEEE Transactions on Biomedical Engineering* 62.6 (2014): 1466-1477.

[19] Lee, David, and Mihalis Yannakakis. "Principles and methods of testing finite state machines-a survey." *Proceedings of the IEEE* 84.8 (1996): 1090-1123.

[20] Flanagan, David. *Java in a Nutshell*. " O'Reilly Media, Inc.", 2005.

[21] https://www.java.com/en/download/help/whatis_java.html

[22] https://developer.android.com/guide/platform

[23] https://developer.android.com/guide/components/activities/intro-activities

[24] https://developer.android.com/guide/components/activities/activity-lifecycle

[25] https://developer.android.com/guide/fragments

[26] https://developer.android.com/guide/fragments/lifecycle

[27] https://developer.android.com/develop/sensors-and-location/sensors/sensors_overview

[28] https://developer.android.com/reference/android/os/Vibrator

[29] https://developer.android.com/studio

[30] https://developers.google.com/ar

[31] Taheri, Hamid, and Zhao Chun Xia. "SLAM; definition and evolution." *Engineering Applications of Artificial Intelligence* 97 (2021): 104032.

[32] https://developers.google.com/ar/develop/fundamentals

[33] https://developers.google.com/ar/develop/depth

[34] https://developers.google.com/ar/develop/java/depth/developer-guide

[35] Shreiner, Dave. *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*. Pearson Education, 2009.

[36] https://github.com/SceneView/sceneform-android

[37] https://sceneview.github.io/sceneform-android/

[38] https://ttsfree.com/text-to-speech/greek-greece

[39] https://developer.android.com/studio/publish

[40] https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative

[41] Derczynski, Leon. "Complementarity, F-score, and NLP Evaluation." *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. 2016.

[42] Bradski, Gary, and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[43] Kapur, Salil, and Nisarg Thakkar. *Mastering opencv android application programming*. Packt Publishing Ltd, 2015.

[44] Alsing, Oscar. "Mobile object detection using tensorflow lite and transfer learning." (2018).

[45] https://www.tensorflow.org/lite

[46] Nguyen, Chi. "Utilizing Google's Machine Learning Kit in Developing Android Application." (2024).

[47] https://developers.google.com/ml-kit

[48] https://www.apple.com/ios/ios-17/

[49] Utsumi, Makoto, Taketsugu Hirabayashi, and Muneo Yoshie. "Development for teleoperation underwater grasping system in unclear environment." *Proceedings of the 2002 Interntional Symposium on Underwater Technology (Cat. No. 02EX556)*. IEEE, 2002.

[50] Lawson, Shaun W., and John RG Pretlove. "Augmented reality for underground pipe inspection and maintenance." *Telemanipulator and Telepresence Technologies V*. Vol. 3524. SPIE, 1998.

[51] Mourtzis, Dimitris, Aikaterini Vlachou, and Vasilios Zogopoulos. "Cloud-based augmented reality remote maintenance through shop-floor monitoring: a product-service system approach." *Journal of Manufacturing Science and Engineering* 139.6 (2017): 061011.

[52] Chu, Chih-Hsing, et al. "Programming by demonstration in augmented reality for the motion planning of a three-axis CNC dispenser." *International Journal of Precision Engineering and Manufacturing-Green Technology* 7 (2020): 987-995.

[53] Chong, Jonathan Wun Shiung, et al. "Robot programming using augmented reality: An interactive method for planning collision-free paths." *Robotics and Computer-Integrated Manufacturing* 25.3 (2009): 689-701.

[54] Chung, Kyung H., John P. Shewchuk, and Robert C. Williges. "An application of augmented reality to thickness inspection." *Human Factors and Ergonomics in Manufacturing & Service Industries* 9.4 (1999): 331-342.

[55] Kato, Hirokazu, and Mark Billinghurst. "Marker tracking and hmd calibration for a video-based augmented reality conferencing system." *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*. IEEE, 1999.

[56] Połap, Dawid, et al. "Obstacle detection as a safety alert in augmented reality models by the use of deep learning techniques." *Sensors* 17.12 (2017): 2803.

[57] Carrio, Adrian, et al. "Drone detection using depth maps." *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018.

[58] Shahira, K. C., Sagar Tripathy, and A. Lijiya. "Obstacle detection, depth estimation and warning system for visually impaired people." *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*. IEEE, 2019.