ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κατασκευή διαδραστικών βιβλίων με τεχνολογία επαυξημένης πραγματικότητας

Γιακουμίδης Στυλιανός

Εισηγητής: Δρ Νικόλαος Ζάχαρης, Καθηγητής

Κατασκευή διαδραστικών βιβλίων με τεχνολογία επαυξημένης πραγματικότητας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κατασκευή διαδραστικών βιβλίων με τεχνολογία επαυξημένης πραγματικότητας

Γιακουμίδης Στυλιανός
Α.Μ. 71343995

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Digitally signed by Nikolaos Zacharis Date: 2021.09.06 15:34:10 +03'00'	Panagiotis Giannakopoulos Digitally signed by Panagiotis Giannakopoulos Date: 2021.09.08 19:30:11 +03'00'	Georgios Prezerakos Digitally signed by Georgios Prezerakos Date: 2021.09.07 12:03:14 +03'00'
.....	.....	.....
Καθηγητής Ζάχαρης Νικόλαος	Καθηγητής Γιαννακόπουλος Παναγιώτης	Καθηγητής Πρεζεράκος Γεώργιος

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Γιακουμίδης Στυλιανός. του Αλεξάνδρου, με αριθμό μητρώου 71343995 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον υπεύθυνο καθηγητή της διπλωματικής μου, κ. Νικόλαο Ζάχαρη, για τις συμβουλές του τόσο σε ακαδημαϊκό όσο και σε επαγγελματικό επίπεδο, καθώς και την οικογένεια και τους φίλους μου για την ανιδιοτελή στήριξή τους. Επιπλέον, θα ήθελα να ευχαριστήσω τον φίλο και συνάδελφο μου Θεόφιλο Καξηρή με τον οποίο ξεκίνησε ομαδικά η παρούσα διπλωματική, αλλά, λόγω ανωτέρας βίας, κατέληξε ατομική.

ΠΕΡΙΛΗΨΗ

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η παροχή μιας εμβυθιστικής εμπειρίας ανάγνωσης βιβλίων χρησιμοποιώντας τις εξελίξεις στην τεχνολογία της επαυξημένης πραγματικότητας. Η Android εφαρμογή που κατασκευάστηκε προσθέτει ψηφιακά στοιχεία οπτικού ή ακουστικού περιεχομένου κατά τη διάρκεια της ανάγνωσης του βιβλίου ανάλογα με την τρέχουσα σελίδα. Η επιλογή των ψηφιακών στοιχείων καθώς και του βιβλίου πραγματοποιείται κατά την διάρκεια εκτέλεσης της εφαρμογής, μέσω αρχείων εισόδου, εξαλείφοντας την ανάγκη δημιουργίας διαφορετικών εκδόσεων της εφαρμογής για κάθε βιβλίο.

ABSTRACT

The present thesis concerns the development of an immersive book-reading experience by taking advantage of the advances in augmented reality technologies. The Android application that will be developed will provide visual or auditory digital elements during the reading process depending on the current page. The choice of these elements, as well as the book itself, is made during runtime of the application using input files, therefore there will not be a need for a different version of the application for each book.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: επαυξημένη πραγματικότητα
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: κάμερα, ανίχνευση προτύπων, ομογραφία, επεξεργασία εικόνας
KEY WORDS: camera, feature detection, homography, image processing

Περιεχόμενα

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1 – ΕΙΣΑΓΩΓΗ

1.1 – Τι είναι επαυξημένη πραγματικότητα

Ο όρος “επαυξημένη πραγματικότητα” (Augmented Reality – A.R.), όπως φαίνεται και από την ετοιμολογία του, αναφέρεται στον τομέα της επιστήμης υπολογιστών ο οποίος πραγματεύεται την “ενίσχυση” του πραγματικού κόσμου με ψηφιακά αντικείμενα ή στοιχεία, με σκοπό να εμπλουτίσει την εμπειρία του χρήστη. Τα στοιχεία αυτά μπορεί να είναι από απλά ηχητικά μηνύματα ή μουσικά κομμάτια, μέχρι τρισδιάστατες αναπαραστάσεις συγκεκριμένων αντικειμένων. Ο συνήθης εξοπλισμός για εφαρμογές αυτής της μορφής είναι ένα headset, το οποίο προβάλει στα μάτια του χρήστη τα ψηφιακά στοιχεία, και κάμερες ώστε το σύστημα να αντιλαμβάνεται τον κόσμο. Σε πιο προηγμένες εφαρμογές, ο εξοπλισμός μπορεί να καλύψει ολόκληρο το σώμα του χρήστη ώστε να αξιοποιήσει περισσότερες αισθήσεις του πέρα από την όραση και την ακοή. Ωστόσο, όταν το κόστος είναι σημαντικός παράγοντας, μια απλή συσκευή smartphone μπορεί να καλύψει τις πιο βασικές ανάγκες μιας εφαρμογής. Η εμφάνιση του όποιου ψηφιακού περιεχομένου πυροδοτείται από την ανίχνευση ενός στοιχείου του πραγματικού κόσμου από τον εξοπλισμό, όπως για παράδειγμα ένα αντικείμενο ή μια εικόνα.

Η A.R. είναι αρκετά σύγχρονος τομέας ο οποίος απέκτησε μεγάλο επιστημονικό και ερευνητικό ενδιαφέρον τα τελευταία χρόνια, με μεγάλες εταιρίες όπως η Google, η Microsoft και το Facebook να επενδύουν πολλά εκατομμύρια σε αυτόν. Αυτό είναι αναμενόμενο δεδομένου ότι οι εφαρμογές της A.R. είναι πολυπληθείς στους τομείς της ιατρικής, της εκπαίδευσης, της βιομηχανικής παραγωγής, της ψυχαγωγίας και σε άλλους.

1.2 – Θέμα διπλωματικής

Η παρούσα διπλωματική στοχεύει στην αξιοποίηση της επαυξημένης πραγματικότητας στον τομέα της ψυχαγωγίας, και συγκεκριμένα στην ανάγνωση

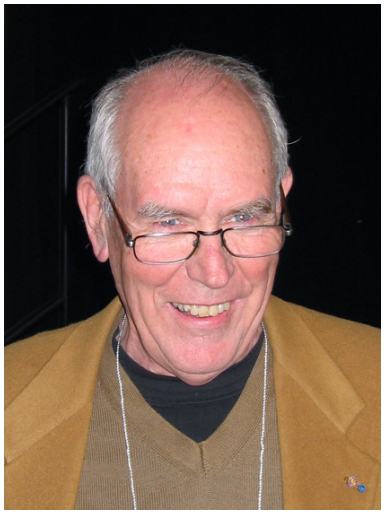
βιβλίων. Ο χρήστης θα βλέπει το βιβλίο μέσω της κάμερας της συσκευής του και πάνω σε αυτό θα εμφανίζονται εικόνες, ή θα αναπαράγονται ήχοι μέσω των ηχείων της συσκευής. Η λειτουργικότητα αυτού του σεναρίου θα υλοποιηθεί σε μια εφαρμογή που θα εκτελείται στην συσκευή του χρήστη και θα της δίνει το ρόλο του headset. Αυτό έχει ως αποτέλεσμα να μην απαιτείται εξειδικευμένος εξοπλισμός για την χρήση του συστήματος γεγονός που το καθιστά προσβάσιμο. Ένα σημαντικό στοιχείο της εφαρμογής που θα κατασκευαστεί είναι η προσαρμοστικότητά της σε πολλά βιβλία. Αυτό θα επιτευχθεί με χρήση αρχείων εισόδου συγκεκριμένης δομής, τα οποία η εφαρμογή θα διαβάζει και θα προσαρμόζεται ανάλογα. Συνεπώς ο χρήστης θα μπορεί να αλλάξει το βιβλίο που διαβάζει γρήγορα και χωρίς να απαιτείται νέα μεταγλώττιση. Επιπλέον, πέρα από το πρακτικό κομμάτι της υλοποίησης θα παρουσιαστεί η έρευνα που πραγματοποιήθηκε για να κατασκευαστεί το συγκεκριμένο λογισμικό καθώς θα γίνει και ανάλυση προαπαιτούμενων θεωρητικών γνώσεων.

1.3 – Ιστορική αναδρομή

Αν και σύγχρονος, ο τομέας της επαυξημένης πραγματικότητας είχε ξεκινήσει να εμφανίζεται σε έρευνες αρκετές δεκαετίες πριν. Τα κύρια εμπόδια με τα οποία ήρθαν αντιμέτωποι οι επιστήμονες ήταν η ανάγκη ακριβού εξοπλισμού για την αξιοποίησή της, καθώς και η ελλιπής υπολογιστική ισχύς. Παρακάτω θα αναλυθεί η πορεία εξέλιξης της A.R. και οι σημαντικότερες εφαρμογές της που αναπτύχθηκαν.

Το πρώτο headset επαυξημένης πραγματικότητας κατασκευάστηκε το 1968 από τον καθηγητή του Πανεπιστημίου του Χάρβαρντ Ivan Edward Sutherland και μια ομάδα φοιτητών του. Το σύστημα που κατασκεύασε το ονόμασε “The Sword of Damocles” και το παρουσίασε μαζί με την έρευνά του σε επιστημονική δημοσίευση του με τίτλο “A head-mounted three dimensional display”. Ο χρήστης του μπορούσε να βλέπει διαφανή τρισδιάστατα αντικείμενα σε μορφή απλών γραμμών που ένωναν τις κορυφές τους (wireframe), τα οποία φαίνονταν να περιστρέφονται ανάλογα με το που κοιτούσε. Το ίδιο το headset δεν ήταν φορητό σαν τα σημερινά, καθώς ήταν

ογκώδες και κρεμασμένο από το ταβάνι του εργαστηρίου, ωστόσο θεμελίωσε την δομή και λειτουργία αυτών που ακολούθησαν.



Εικόνα 1.1: Ivan Edward Sutherland Dick Lyon / CC BY-SA (https://creativecommons.org/licenses/by-sa/3.0)

Το 1974, ο Myron Krueger κατασκεύασε το Videoplace. Πρόκειται για σύστημα τεχνητής πραγματικότητας το οποίο αποτελούνταν από ένα χώρο με πολλούς αισθητήρες γύρω από τον χρήστη, οι οποίοι παρακολουθούσαν την κάθε του κίνηση. Το περίγραμμα του χρήστη εμφανιζόταν σε μια οθόνη μαζί με ψηφιακά στοιχεία με τα οποία μπορούσε να αλληλεπιδράσει με διαφόρους τρόπους χρησιμοποιώντας τα χέρια του. Το γεγονός ότι ο εξοπλισμός του συστήματος δεν ήταν τοποθετημένος πάνω στον χρήστη έκανε την χρήση του συστήματος λιγότερο κουραστική για αυτόν.

Το 1990 ο Tom Caudell, ερευνητής της αεροδιαστημικής εταιρίας Boeing, επινόησε τον όρο “επαυξημένη πραγματικότητα” όταν ανέπτυξε μαζί με τον συνεργάτη του David Mizell ένα σύστημα για να βοηθήσει την βιομηχανική παραγωγή των αεροσκαφών της εταιρίας. Αυτό το σύστημα χρησιμοποιούσε ένα headset για να δείχνει στους μηχανικούς πληροφορίες για την κατασκευή και συνδεσμολογία

διαφόρων μοντέλων αεροσκαφών, αυξάνοντας έτσι την παραγωγικότητα της εταιρίας. Πρόκειται ίσως για την πρώτη εφαρμογή της A.R. στη διαδικασία της βιομηχανικής παραγωγής.

Το 1992 αναπτύχθηκε το “Virtual Fixtures” από τον Luis Rosenberg στο εργαστήριο έρευνας της αεροπορίας των Ηνωμένων Πολιτειών Αμερικής. Πρόκειται για ένα σύστημα A.R. στο οποίο ο χρήστης φορούσε έναν ολόσωμο εξοπλισμό μέσω του οποίου μπορούσε να χειριστεί απομακρυσμένα δύο ρομποτικούς βραχίονες χρησιμοποιώντας τα ίδια του τα χέρια. Η κίνηση του κάθε βραχίονα ακολουθούσε την κίνηση του χεριού του χρήστη, δίνοντας του τη δυνατότητα μέσω του headset να βλέπει σε πραγματικό χρόνο τον χώρο με τους βραχίονες. Για να επιτευχθεί μια εμβυθιστική εμπειρία για τον χρήστη, οι βραχίονες είχαν τοποθετηθεί με τέτοιο τρόπο ώστε ο χειριστής να τους βλέπει στη κάμερα στην ίδια απόσταση και θέση με τα δικά του χέρια. Δηλαδή η απόσταση των χεριών από τα μάτια του ήταν ίδια με την απόσταση ανάμεσα στους βραχίονες και την κάμερα. Έλεγχοι απόδοσης που πραγματοποιήθηκαν έδειξαν για πρώτη φορά ότι η σωστή χρήση της A.R. μπορεί να προκαλέσει σημαντική βελτίωση στην απόδοση των ανθρώπων σε έργα που περιλαμβάνουν απομακρυσμένο έλεγχο.

Εικόνα 1.2: Ο Luis Rosenberg σε ομιλία του το 2017
GardenM / CC BY-SA
(https://creativecommons.org/licenses/by-sa/4.0)

Εικόνα 1.3: Ο Luis Rosenberg κατά την διάρκεια δοκιμής του Virtual Fixtures
WCS100 / CC BY-SA
(https://creativecommons.org/licenses/by-sa/3.0)

Το 1994 δημιουργήθηκε η πρώτη θεατρική παραγωγή που αξιοποίησε την A.R. Η παράσταση “Dancing in Cyberspace” της Julie Martin περιλάμβανε ακροβάτες που χόρευαν ανάμεσα σε ψηφιακά αντικείμενα.

Το 1998 η A.R. χρησιμοποιήθηκε για πρώτη φορά σε αγώνα ράγκμπι στην Αμερική. Το σύστημα “1st & Ten”, όπως ονομάστηκε, τοποθετούσε κατά την διάρκεια του αγώνα στο δάπεδο του γηπέδου μια επιπλέον γραμμή με ψηφιακό τρόπο, η οποία βοηθούσε τους τηλεθεατές να γνωρίζουν την θέση ενός συγκεκριμένου ορίου που ήταν σημαντικό για τους κανόνες του παιχνιδιού. Η γραμμή αυτή ήταν κίτρινη σε χρώμα, ώστε να ξεχωρίζει από τις πραγματικές, λευκές, γραμμές, και δεν εμφανιζόταν πάνω από αντικείμενα που ήταν μπροστά της όπως τους παίχτες ή την μπάλα. Αυτό καθιερώθηκε και άρχισε να χρησιμοποιείται σε περισσότερους αγώνες και να εμφανίζει περισσότερα στοιχεία. Για παράδειγμα, σε καταστάσεις ομίχλης όπου το γήπεδο δεν ήταν ευδιάκριτο, το σύστημα μπορούσε να προβάλει επάνω του όλες τις γραμμές και τα όρια ώστε να μπορούν οι τηλεθεατές και οι σχολιαστές να παρακολουθήσουν τον αγώνα χωρίς προβλήματα.

Το 1999 κυκλοφόρησε το ARToolKit. Πρόκειται για βιβλιοθήκη ανάπτυξης λογισμικού επαυξημένης πραγματικότητας που ανέπτυξε ο Dr. Hirokazu Kato. Το ARToolKit ήταν από τις πρώτες βιβλιοθήκες A.R. που υποστήριξαν και φορητές συσκευές, με την πρώτη αξιοποίησή της να σημειώνεται αρχικά το 2005 στο λειτουργικό σύστημα Symbian και αργότερα για τα iOS και Android. Επιπλέον, το 2009 η βιβλιοθήκη έφερε την επαυξημένη πραγματικότητα στα προγράμματα πλοήγησης του διαδικτύου. Σήμερα βρίσκεται υπό συνεχή ανάπτυξη στο GitHub με 21 contributors την στιγμή συγγραφής του κειμένου.

Εικόνα 1.4: Dr. Hirokazu Kato

Το 2009 σημειώθηκε χρήση της A.R. από το περιοδικό “Esquire”. Το περιοδικό σε ένα τεύχος του παρότρυνε τους αναγνώστες να τρέξουν ένα συγκεκριμένο πρόγραμμα στον προσωπικό τους υπολογιστή και στη συνέχεια να τοποθετήσουν το τεύχος κοντά στην κάμερα. Το πρόγραμμα αναγνώριζε ένα συγκεκριμένο μοτίβο που ήταν εκτυπωμένο στο εξώφυλλο του τεύχους και αυτό που ακολούθησε ήταν ο ηθοποιός James Downey Jr. να “ζωντανεύει” και να αρχίζει να μιλάει στον αναγνώστη.

Το 2013 η εταιρία αυτοκινητοβιομηχανίας Volkswagen ανέπτυξε την εφαρμογή MARTA (Mobile Augmented Reality Technical Assistance) για τους τεχνικούς της. Η εφαρμογή εμφάνιζε μπροστά στα μάτια τους πληροφορίες για το αυτοκίνητο και τα επιμέρους εξαρτήματά του όταν το εστίαζαν στην κάμερα του tablet που χρησιμοποιούσαν. Παρόλο που η εφαρμογή ήταν δοκιμαστική και πολύ περιορισμένη καθώς υποστήριζε μόνο ένα μοντέλο αυτοκινήτου, αποτέλεσε μια ακόμη χρήσιμη αξιοποίηση των τεχνολογιών A.R. η οποία θα μπορούσε να εφαρμοστεί σε μεγαλύτερο σκέλος στο μέλλον.

Το 2016 ήταν η χρονιά που η A.R. προστέθηκε στις συσκευές πολλών χρηστών με το Pokémon Go. Πρόκειται για βιντεοπαιχνίδι που αναπτύχθηκε από την Niantic σε συνεργασία με την Nintendo και την The Pokémon Company για τα λειτουργικά συστήματα iOS και Android. Το παιχνίδι αξιοποιούσε σε μεγάλο βαθμό το G.P.S. της συσκευής μαζί την A.R. για να προσομοιώσει τον κόσμο της γνωστής σειράς κινουμένων σχεδίων, ώστε ο παίκτης να αποκτήσει μια καθηλωτική εμπειρία. Σε αυτό εμφανιζόταν περιστασιακά στην γειτονιά του παίχτη pokémon τα οποία μπορούσε να τα κυνηγήσει και να τα προσθέσει στη συλλογή του. Η υψηλή καινοτομία του παιχνιδιού σε συνδυασμό με το πολύ γνωστό όνομα των pokémon οδήγησε σε εξαιρετική απήχηση, με εκατομμύρια παίχτες να κυκλοφορούν στους δρόμους παίζοντας το παιχνίδι, και άνοιξε τις πόρτες για μια νέα αξιοποίηση της A.R. στην βιομηχανία των βιντεοπαιχνιδιών.

Εικόνα 1.5: Ένα pokémon βρέθηκε στον πραγματικό κόσμο από έναν παίχτη του Pokémon Go.

Σήμερα, το μεγάλο πλήθος εφαρμογών της A.R., τόσο στον τομέα της έρευνας και ανάπτυξης όσο και στη ψυχαγωγία, έχει οδηγήσει στην επένδυση πολλών

εκατομμυρίων από τους γίγαντες τεχνολογιών. Πρόσφατα επιτεύγματα όπως το Google Glass, που συμπίεσε όλο το υλικό του εξοπλισμού στο μέγεθος ενός ζευγαριού γυαλιών, ή το Microsoft HoloLens, προχωρά ένα βήμα παραπέρα επιτρέποντας στον χρήστη να αλληλεπιδρά με τα ψηφιακά στοιχεία, δείχνουν ότι η A.R. θα αποτελέσει σημαντικό ερευνητικό πεδίο για τα επόμενα χρόνια.

Εικόνα 1.6: Google glass
[Εικόνα από: Mikepanhu – Απρίλιος 2014 (https://upload.wikimedia.org/wikipedia/commons/b/be/Google_Glass_with_frame.jpg)]

Εικόνα 1.7: Microsoft HoloLens
[Εικόνα από: Jean-Pierre Dalbéra – Ιούνιος 2016 (https://live.staticflickr.com/7140/27628074995_e28f32372f_o_d.jpg)]

ΚΕΦΑΛΑΙΟ 2 - ΕΡΕΥΝΑ

2.1 – Εισαγωγή

Όπως για κάθε έργο λογισμικού, έτσι και για το συγκεκριμένο υπάρχει πλήθος διαφορετικών τρόπων υλοποίησης, ο καθένας με τα πλεονεκτήματα και τα μειονεκτήματά του. Ως αποτέλεσμα, χρειάστηκε να πραγματοποιηθεί εκτενής έρευνα ώστε να βρεθεί η καταλληλότερη μεθοδολογία. Στο παρόν κεφάλαιο, θα παρουσιαστεί η πορεία της έρευνας που εκτελέστηκε, οι μεθοδολογίες που λήφθηκαν υπ' όψιν και οι λόγοι για τους οποίους αυτές απορρίφθηκαν ή εγκρίθηκαν.

2.2 – Επιλογή πλατφόρμας ανάπτυξης

Η πρώτη κίνηση μετά την σύλληψη της ιδέας ήταν η επιλογή της πλατφόρμας πάνω στην οποία θα κατασκευαστεί το λογισμικό. Αυτή η επιλογή είναι κρίσιμη καθώς καθορίζει το πλεόνασμα των δυνατοτήτων και αδυναμιών του λογισμικού, όπως επίσης και τα λειτουργικά συστήματα πάνω στα οποία θα μπορεί να εκτελεστεί. Δεδομένου ότι στοχεύεται η ανάπτυξη λογισμικού για το λειτουργικό σύστημα Android, οι επιλογές πλατφόρμας μειώνονται σημαντικά.

Η πλατφόρμα που επιλέχθηκε τελικά είναι η μηχανή παιχνιδιών Unity. Ο κύριος λόγος αυτής της επιλογής είναι ότι η Unity παρέχει ένα προσιτό περιβάλλον προγραμματισμού, γεγονός που οδηγεί σε ευκολότερη και γρηγορότερη δημιουργία ενός πρωτοτύπου για μια ιδέα, χωρίς να απαιτούνται εξειδικευμένες γνώσεις της δομής κάποιου συγκεκριμένου λειτουργικού συστήματος. Επιπλέον, η Unity έχει την δυνατότητα να παράγει εκτελέσιμα αρχεία για πολλά λειτουργικά συστήματα πέρα του Android, γεγονός το οποίο καθιστά μια πιθανή αναβάθμιση του λογισμικού με στόχο την υποστήριξη και άλλων λειτουργικών συστημάτων ευκολότερη.

2.3 – Επιλογή τρόπου υλοποίησης

Από τις δυνατότητες του λογισμικού, οι πιο σύνθετες είναι η ανίχνευση της σελίδας στην οποία βρίσκεται ο χρήστης και η εύρεση της ακριβής τοποθεσίας του βιβλίου εντός του καρέ της κάμερας. Και οι δύο αυτές πληροφορίες είναι απαραίτητες για την εμφάνιση των ψηφιακών στοιχείων. Δεδομένου ότι τα υπόλοιπα τμήματα του λογισμικού είναι αρκετά πιο προφανή ως προς την ανάπτυξη, η έρευνα που θα παρουσιαστεί παρακάτω αφορά μόνο τον υπολογισμό των δύο προαναφερθέντων χαρακτηριστικών.

2.3.1 – Vuforia

Μια πιθανή επιλογή για την υλοποίηση της ανίχνευσης της σελίδας και της εύρεσης των συντεταγμένων του βιβλίου είναι η βιβλιοθήκη Vuforia. Πρόκειται για μία από τις πιο σύγχρονες και ισχυρές βιβλιοθήκες στον τομέα της ανάπτυξης A.R. εφαρμογών. Προσφέρει υποστήριξη για πολλές γλώσσες, όπως Java, C++ καθώς και C#. Από τις διαθέσιμες γλώσσες είναι φανερό ότι μπορεί να χρησιμοποιηθεί σε μεγάλο εύρος συσκευών, από σταθερούς υπολογιστές μέχρι και κινητά, καθώς και σε πολλά λειτουργικά συστήματα.

Για να χρησιμοποιηθεί, πρέπει να δημιουργηθεί ένας λογαριασμός στον διαδικτυακό χώρο της βιβλιοθήκης, απ’ όπου γίνεται η διαχείριση των A.R. εφαρμογών του κατόχου. Επόμενο βήμα είναι το ανέβασμα των εικόνων που θα πρέπει να αναγνωρίζονται από την βιβλιοθήκη. Οι εικόνες αυτές περνούν από επεξεργασία και κρίνονται ως προς το πόσο εύκολα αναγνωρίσιμες είναι στον χώρο. Αυτή η πληροφορία εμφανίζεται στον προγραμματιστή μέσω μιας βαθμολόγησης της εικόνας και επιδεικνύοντας τα “σημεία κλειδιά” (key points) της, από τα οποία καθορίζεται η ποιότητα και η ευκολία ανίχνευσής της (εικόνα 2.1). Τέλος, παράγεται από τη διαδικτυακή πλατφόρμα ένα αρχείο που χαρακτηρίζει τις προς-ανίχνευση εικόνες και μπορεί να φορτωθεί από την βιβλιοθήκη για να ξεκινήσει η ανίχνευση

τους. Στο κεφάλαιο 3 θα αναλυθούν με πιο μεγάλη λεπτομέρεια τα key points καθώς και τα χαρακτηριστικά που καθορίζουν μια εικόνα εύκολα ανιχνεύσιμη.

Εικόνα 2.1: Δείγμα από τη πλατφόρμα της Vuforia. Φαίνονται τα keypoints ως κίτρινοι σταυροί και η ευκολία ανίχνευσης (ένδειξη “Augmentable”) μιας εικόνας.

Για την δοκιμή της Vuforia δημιουργήθηκε ένα project στη Unity το οποίο ανιχνεύει μια συγκεκριμένη εικόνα στη σκηνή, μέσω της κάμερας της συσκευής, και στη συνέχεια εμφανίζει ένα απλό μοντέλο πάνω της. Ο αλγόριθμος ανίχνευσης είναι πολύ γρήγορος με αποτέλεσμα να μην υπάρχει σχεδόν καθόλου καθυστέρηση κατά τη διαδικασία της ανίχνευσης, ακόμα και όταν εκτελείται σε πραγματικό χρόνο με πολλά καρέ ανά δευτερόλεπτο από την κάμερα. Επιπλέον, η ανίχνευση έχει μεγάλη ακρίβεια και απόδοση με τον αλγόριθμο να είναι σε θέση να αντιληφθεί την τοποθεσία του αντικειμένου και στην περίπτωση που αυτό είναι υπό γωνία και σε απόσταση από την κάμερα ή όταν παρεμβάλλονται μικρά εμπόδια μπροστά του. Μετά την δοκιμή της Vuforia έγινε φανερό γιατί θεωρείται ένα από τα πιο δυνατά πακέτα επαυξημένης πραγματικότητας που υπάρχουν στην αγορά.

Εικόνα 2.2: Το Vuforia ανίχνευσε την εικόνα και εμφάνισε πάνω της το μοντέλο που είχε προκαθοριστεί.

Ωστόσο, παρά την καλή της απόδοση, δεν μπορεί να χρησιμοποιηθεί για την υλοποίηση του συγκεκριμένου λογισμικού. Όπως αναφέρθηκε και στο κεφάλαιο 1.2, απαραίτητο στοιχείο της τελικής εφαρμογής είναι η προσαρμοστικότητά της σε διαφορετικά βιβλία. Αυτό δεν μπορεί να επιτευχθεί με τη Vuforia αφού για να ανιχνεύσει μια εικόνα πρέπει πρώτα αυτή να ανέβει στη διαδικτυακή πλατφόρμα και, στη συνέχεια, να φορτωθεί στο πρόγραμμα το αρχείο της ανίχνευσης. Αυτό το γεγονός το καθιστά ακατάλληλο αφού απαιτείται αλλαγή και μεταγλώττιση του προγράμματος για να προσαρμοστεί σε νέο βιβλίο.

2.3.2 – Σύστημα ανίχνευσης διαφορών στα καρέ

Η δεύτερη επιλογή για την υλοποίηση είναι ένα σύστημα του οποίου σκοπός είναι η ανίχνευση της κίνησης της αλλαγής σελίδας από την κάμερα χρησιμοποιώντας μεθόδους επεξεργασίας εικόνας. Με βάση αυτό, ο χρήστης θα όριζε την αρχική σελίδα κατά την εκκίνηση της ανάγνωσης και, μετέπειτα, το λογισμικό θα αντιλαμβανόταν πότε γύρισε η σελίδα παρατηρώντας για κινήσεις με αριστερή ή δεξιά κατεύθυνση στην κάμερα.

Η ανίχνευση επιτυγχάνεται υπολογίζοντας και αποθηκεύοντας το κέντρο βάρους των διαφορών ανάμεσα σε δύο διαδοχικά καρέ της κάμερας. Δεδομένου ότι όταν πραγματοποιείται αλλαγή σελίδας το κέντρο βάρους θα έχει μια τάση να μεταφέρεται είτε αριστερά είτε δεξιά για ένα μικρό χρονικό διάστημα, θα μπορούσε να φτιαχτεί αλγόριθμος, ο οποίος θα παρατηρούσε σε πραγματικό χρόνο την κίνηση του κέντρου βάρους και θα έκρινε εάν όντως πρόκειται για γύρισμα σελίδας.

Αν και υποσχόμενο, αυτό το σύστημα απορρίφθηκε πριν ολοκληρωθεί πλήρως η ανάπτυξή του. Οι λόγοι απόρριψης ήταν κυρίως δύο. Ο πρώτος σχετίζεται με ανησυχίες για την ακρίβεια ανίχνευσης της αλλαγής σελίδας, καθώς είναι πιθανό το κέντρο βάρους της διαφοράς να ακολουθήσει αριστερή ή δεξιά πορεία χωρίς την παρέμβαση του χρήστη λόγω αλλαγών στο παρασκήνιο ή αστάθειας της κάμερας. Ο δεύτερος λόγος απόρριψης είναι το γεγονός ότι αυτό το σύστημα δεν παρέχει κάποιο τρόπο ακριβούς υπολογισμού της τοποθεσίας του βιβλίου, ώστε να εμφανιστούν επάνω σε αυτό τα ψηφιακά στοιχεία.

2.3.3 – Ανίχνευση εικόνων μέσω OpenCV

Η τρίτη και τελική επιλογή για την υλοποίηση είναι η κατασκευή ενός συστήματος αναγνώρισης εικόνων παρόμοιο με τη Vuforia χρησιμοποιώντας την ανοιχτού κώδικα βιβλιοθήκη επεξεργασίας εικόνας και μηχανικής όρασης, OpenCV (Open Computer Vision).

Η OpenCV είναι μια από τις επικρατέστερες και πιο γνωστές βιβλιοθήκες στον τομέα της, με πάνω από 1000 contributors στην σελίδα της στο GitHub. Είναι γραμμένη κυρίως σε C++ με αποτέλεσμα να είναι γρήγορη, καθώς δεν τρέχει σε κάποιο ενδιάμεσο περιβάλλον εκτέλεσης. Ωστόσο, παρέχει wrappers για άλλες γλώσσες όπως Java και Python, και υπάρχουν εκδόσεις της για χρήση σε φορητές συσκευές Android και iOS.

Ένας λόγος που καθιστά την OpenCV κατάλληλη για την ανάπτυξης της εφαρμογής είναι το γεγονός ότι περιλαμβάνει υλοποιήσεις γνωστών αλγόριθμων

αναγνώρισης εικόνων, οι οποίοι είναι απαραίτητοι για να μπορεί η εφαρμογή να βρίσκει τις σελίδες του βιβλίου. Η χρήση της OpenCV είναι παρόμοια με αυτή του Vuforia στο ότι πρέπει αρχικά να φορτωθεί η προς-ανίχνευση εικόνα και, στη συνέχεια, να περάσει από επεξεργασία, ώστε τελικά να είναι έτοιμη η ανίχνευση. Ωστόσο, δεν υπάρχει ο περιορισμός της διαδικτυακής πλατφόρμας του Vuforia, καθώς ο προγραμματισμός της επεξεργασίας και της ανίχνευσης γίνεται χειροκίνητα, γεγονός που καθιστά την διαδικασία ανίχνευσης πιο ευέλικτη.

Μετά την ανίχνευση της σελίδας, λαμβάνονται οι συντεταγμένες της μέσα στην σκηνή καθώς και η γωνία περιστροφής της σε σχέση με την κάμερα. Τα παραπάνω στοιχεία χρησιμοποιούνται για την σωστή τοποθέτηση του ψηφιακού στοιχείου στη σκηνή.

ΚΕΦΑΛΑΙΟ 3 – ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

3.1 – Εισαγωγή

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, για την ανίχνευση αντικειμένων πρέπει να υπάρχει μια εικόνα του αντικειμένου τραβηγμένη σε ιδανικές συνθήκες φωτισμού χωρίς άλλα στοιχεία πέρα από αυτό. Αυτή η εικόνα στη συνέχεια συγκρίνεται με μία εικόνα σκηνής από την κάμερα της συσκευής με σκοπό να ελεγχθεί αν το αντικείμενο υπάρχει σε αυτή τη σκηνή. Στον τομέα της μηχανικής όρασης η αρχική εικόνα ονομάζεται “train” και η εικόνα της κάμερας “query”. Συνοπτικά, η εύρεση αντικειμένου περιλαμβάνει τα εξής στάδια:

1. Εύρεση των χαρακτηριστικών περιοχών στις δύο εικόνες.
2. Υπολογισμός ενός περιγραφέα (descriptor) για καθεμιά από αυτές.
3. Σύγκριση των descriptors που υπολογίστηκαν για τις δύο εικόνες ώστε να βρεθεί αν υπάρχει το αντικείμενο στη σκηνή.
4. Αν υπάρχει, τότε υπολογισμός της θέσης του αντικειμένου με χρήση της ομογραφίας.

Σε αυτό το κεφάλαιο θα αναλυθεί τι περιλαμβάνει κάθε ένα από αυτά τα στάδια και για ποιο λόγο χρειάζεται.

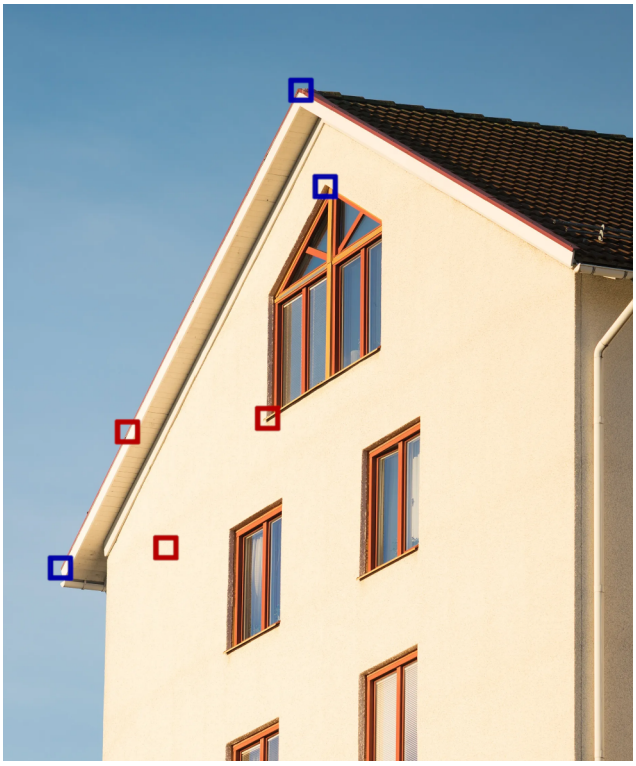
3.2 – Χαρακτηριστικές περιοχές εικόνας (keypoints)

Απαραίτητο στοιχείο για την εύρεση αντικειμένου σε μια εικόνα είναι οι χαρακτηριστικές περιοχές, ή αλλιώς features ή keypoints της εικόνας. Τα keypoints είναι μικρά τμήματα της εικόνας τα οποία περιλαμβάνουν πληροφορία που είναι αρκετά ξεχωριστή και μοναδική ώστε να μπορούν να χρησιμοποιηθούν για την σύγκρισή της με άλλη εικόνα. Η εύρεση τους γίνεται με χρήση ειδικών αλγορίθμων που τα εξάγουν με μεθόδους επεξεργασίας εικόνας. Το πλήθος και η ποιότητα των

keypoints καθορίζουν σε σημαντικό βαθμό την αξιοπιστία της ανίχνευσης. Η διαδικασία εύρεσης των keypoints σε μια εικόνα ονομάζεται “feature detection”.

Όπως αναφέρθηκε η ανίχνευση μιας εικόνας εξαρτάται από τα keypoints που περιέχει, συνεπώς μπορεί να εξαχθεί το συμπέρασμα ότι δεν είναι όλες οι εικόνες κατάλληλες ή εύκολα ανιχνεύσιμες. Ένα “ισχυρό” keypoint πρέπει να βρίσκεται σε μια περιοχή που να είναι μοναδική εντός της εικόνας και να υπάρχει χρωματική αντίθεση. Κοινές περιοχές για keypoints είναι οι γωνίες και οι άκρες των απεικονιζόμενων αντικειμένων καθώς και συγκεκριμένα μοτίβα, όσο αυτά δεν επαναλαμβάνονται. Μια επαναλαμβανόμενη περιοχή δεν αποτελεί ιδανικό χαρακτηριστικό, διότι όταν γίνεται η σύγκριση δεν είναι ξεκάθαρο που ταιριάζει το συγκεκριμένο σημείο αφού επαναλαμβάνεται. Αποτέλεσμα του παραπάνω φαινομένου είναι η πρόκληση απρόβλεπτης συμπεριφοράς. Συνεπώς προκύπτει το συμπέρασμα ότι η σελίδα ενός βιβλίου που περιλαμβάνει μόνο κείμενο δεν μπορεί να βρεθεί εύκολα στη σκηνή, καθώς το κείμενο είναι γεμάτο από μικρές επαναλαμβανόμενες περιοχές που δεν μπορούν να ανιχνευθούν αποτελεσματικά με αυτές τις μεθόδους. Ένα παράδειγμα αυτών των κανόνων φαίνεται στην εικόνα 3.1.

Για την εύρεση των keypoints μιας εικόνας έχουν δημιουργηθεί και παρουσιαστεί σε επιστημονικές δημοσιεύσεις πολλοί αλγόριθμοι. Οι περισσότεροι χρησιμοποιούν, μεταξύ άλλων, ένα συνδυασμό των ήδη υπαρχόντων αλγορίθμων για την εύρεση άκρων (edge detection), γωνιών (corner detection) και ομοιόμορφων περιοχών (blob detection) σε μια εικόνα. Μερικοί από τους πιο γνωστούς αλγόριθμους για feature detection είναι οι SIFT, SURF, AKAZE, STAR, FAST και BRISK.



Εικόνα 3.1: Παράδειγμα ποιότητας των keypoints. Οι κόκκινες περιοχές έχουν χαμηλή ανιχνευσιμότητα ενώ οι μπλε υψηλή.

3.3 – Περιγραφή χαρακτηριστικών περιοχών (descriptors)

Αφού βρεθούν τα χαρακτηριστικά σημεία μιας εικόνας, επόμενο βήμα είναι ο υπολογισμός ενός περιγραφέα (descriptor) για κάθε ένα από αυτά. Ο περιγραφέας είναι μια ακολουθία αριθμών που κωδικοποιούν την χρωματική πληροφορία στην γειτονική περιοχή του keypoint. Σκοπός είναι αυτή η κωδικοποίηση να είναι αφενός αντιπροσωπευτική της περιοχής που αναφέρεται και αφετέρου να μην εξαρτάται σε μεγάλο βαθμό από συνήθεις παραμορφώσεις του αντικειμένου. Οι συνήθεις παραμορφώσεις περιλαμβάνουν αλλαγές που έχει ένα απεικονιζόμενο αντικείμενο όταν αλλάξει η γωνία θέασης ή η απόστασή του από την κάμερα. Με βάση αυτές τις ιδιότητες, ένας αλγόριθμος μπορεί να χαρακτηριστεί ως scale-invariant (ανθεκτικός

στις αλλαγές κλίμακας ή μεγέθους) ή/και rotation-invariant (ανθεκτικός στις αλλαγές περιστροφής ή της οπτικής γωνίας).

Εικόνα 3.2: Το αντικείμενο προς ανίχνευση σε ουδέτερη κατάσταση, όπως παραδίδεται στον αλγόριθμο.

Εικόνα 3.3: Το αντικείμενο σε μετατροπή κλίμακας. Το εμβαδόν που καταλαμβάνει σε pixels είναι διαφορετικό από την αρχική εικόνα.

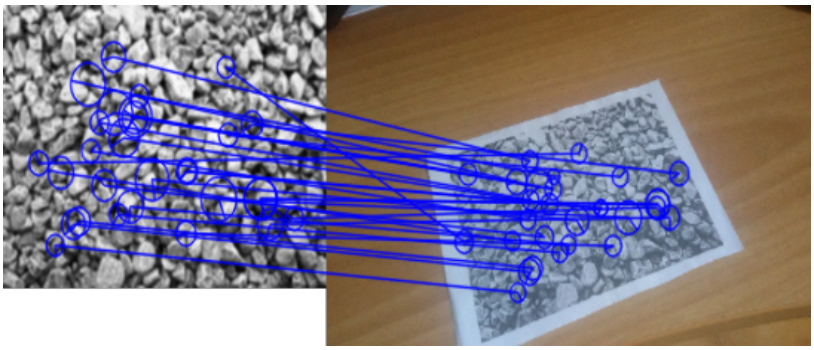
Εικόνα 3.4: Το αντικείμενο σε μετατροπή περιστροφής και κλίμακας.

Είναι συχνό φαινόμενο ένας αλγόριθμος εύρεσης των keypoints να έχει και έναν αντίστοιχο για την περιγραφή των τελευταίων, ενώ υπάρχουν και αλγόριθμοι που δεν έχουν κάποιο ζευγάρι οι οποίοι μπορούν να χρησιμοποιηθούν αυθαίρετα με διαφορετικούς αλγόριθμους περιγραφής. Στην πρώτη περίπτωση, συνίσταται η χρήση του ταιριαστού αλγόριθμου καθώς είναι κατάλληλα σχεδιασμένοι να λειτουργούν συνδυαστικά. Από τους αλγορίθμους που αναφέρθηκαν στο κεφάλαιο 3.2, οι SIFT, SURF, AKAZE και BRISK έχουν αντίστοιχο αλγόριθμο περιγραφής των σημείων τους, ενώ οι FAST και STAR χρησιμοποιούνται μόνο στην ανίχνευση σημείων.

3.4 – Ταίριασμα

Αφού υπολογιστούν οι περιγραφείς για όλα τα keypoints και για τις δύο εικόνες, πρέπει να γίνει σύγκριση μεταξύ τους. Η ομοιότητα ανάμεσα σε δύο descriptors ονομάζεται απόσταση (distance), η χαμηλότερη τιμή της οποίας δηλώνει καλύτερο ταίριασμα. Ο έλεγχος περιλαμβάνει την εύρεση του ζευγαριού του κάθε descriptor της train image στην query image με την μικρότερη απόσταση. Αυτό πρακτικά σημαίνει ότι γίνεται ταίριασμα των πιο όμοιων χαρακτηριστικών περιοχών ανάμεσα στις δύο εικόνες.

Οι αλγόριθμοι διεξαγωγής της αναζήτησης ζευγαριών χωρίζονται σε αλγορίθμους τύπου “ωμής βίας” (brute force) και “κοντινότερου γείτονα” (nearest neighbor). Πιο συγκεκριμένα για την δεύτερη περίπτωση χρησιμοποιείται ο αλγόριθμος της βιβλιοθήκης FLANN (Fast Library for Approximate Nearest Neighbors). Η διαφορά των δύο προαναφερθέντων κατηγοριών είναι ότι ο αλγόριθμος brute force κάνει έλεγχο όλων των descriptors της train image με όλους της query image, συνολικά NxM ελέγχους, για να βρει τα ζευγάρια. Αντιθέτως ο αλγόριθμος της FLANN πραγματοποιεί λιγότερες συγκρίσεις για να την εύρεση ενός ταιριάσματος, γεγονός που τον καθιστά ιδανικό για μεγάλα data sets όπου η διαφορά στην ταχύτητα εκτέλεσης είναι φανερή.



Εικόνα 3.5: Εύρεση ταιριασμάτων σε δύο εικόνες. Οι γραμμές ενώνουν τα σημεία που ταίριαξαν, ενώ το μέγεθος των κύκλων υποδηλώνει το μέγεθος (δηλαδή την ποιότητα) ενός keypoint.

3.5 – Εύρεση αντικειμένου με χρήση της ομογραφίας

Κατά τις διαδικασίες που αναλύθηκαν στις προηγούμενες παραγράφους βρέθηκαν χαρακτηριστικά σημεία από δύο εικόνες και έγινε αντιστοίχηση μεταξύ τους. Στη συνέχεια, πρέπει να υπολογιστούν οι συντεταγμένες του αντικειμένου στην query image. Για να επιτευχθεί αυτό, θα γίνει χρήση της ομογραφίας (homography) που υπάρχει ανάμεσα στις δύο εικόνες. Η ομογραφία είναι ένας πίνακας 3x3, συνήθως συμβολισμένος με το γράμμα Η, ο οποίος αποτελεί μαθηματική αναπαράσταση του προβολικού μετασχηματισμού μιας εικόνας. Έστω δύο εικόνες Α και Β, με την Β να αποτελεί προβολικό μετασχηματισμό της Α βάσει του πίνακα ομογραφίας Η. Τότε ο τελευταίος μπορεί να χρησιμοποιηθεί για να βρεθούν για οποιοδήποτε σημείο της Α, οι συντεταγμένες στις οποίες μετατοπίστηκε στη Β μετά την εκτέλεση του μετασχηματισμού. Αυτή η ιδιότητα είναι πολύ χρήσιμη καθώς σημαίνει πως ο Η μπορεί να χρησιμοποιηθεί για να βρεθούν οι τέσσερις γωνίες του αντικειμένου μέσα στην εικόνα σκηνής, αφού αυτές είναι γνωστές στην ουδέτερη εικόνα του αντικειμένου.

Εικόνα 3.6: Με χρήση της ομογραφίας, βρέθηκαν οι γωνίες του αντικειμένου και ενώθηκαν με κίτρινες γραμμές.

Η μετατροπή ενός σημείου $A(x_1, y_1)$ μιας εικόνας στο σημείο $B(x_2, y_2)$ μιας άλλης με χρήση του πίνακα ομογραφίας Η γίνεται με τον παρακάτω μαθηματικό τύπο:

$$B = H \cdot A \Rightarrow \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

Για να ορίζονται οι πράξεις μεταξύ των πινάκων προστέθηκε μια τρίτη συντεταγμένη z στα σημεία Α και Β η οποία μπορεί να αγνοηθεί.

Εικόνα 3.7: Συσχέτιση των γωνιών του αντικειμένου με βάση την ομογραφία.

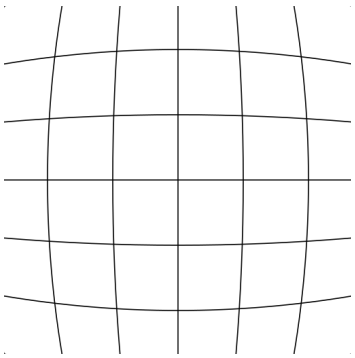
3.6 – Εύρεση γωνιών περιστροφής αντικειμένου

3.6.1 – Εσωτερικές παράμετροι της κάμερας

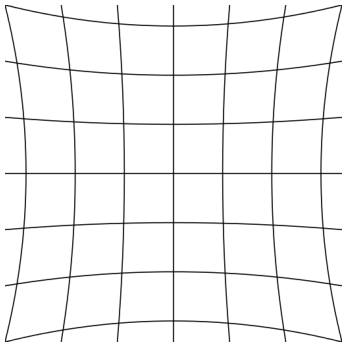
Μια κάμερα δεν μπορεί να αναπαραστήσει τον κόσμο τέλεια. Πάντα προκαλείται κάποιο είδος παραμόρφωσης στη φωτογραφία. Αυτό το φαινόμενο, παρόλο που συνήθως δεν είναι προφανές στο ανθρώπινο μάτι, πρέπει να ληφθεί υπ' όψιν σε πολλές εφαρμογές υπολογιστικής όρασης. Στη προκειμένη περίπτωση οι παράμετροι χρειάζονται για τον υπολογισμό των γωνιών περιστροφής του αντικειμένου που ανιχνεύθηκε σε σχέση με την κάμερα. Οι παράμετροι παραμόρφωσης που εξαρτώνται από την κάμερα ονομάζονται εσωτερικές παράμετροι (intrinsic parameters) και αποτελούνται από:

- Τους συντελεστές ακτινικής (radial) ή εφαπτομενικής (tangential) παραμόρφωσης.
- Την εστιακή απόσταση (focal length) του φακού της κάμερας
- Το οπτικό κέντρο (optical center) της κάμερας.

Ως παράδειγμα, στις παρακάτω εικόνες φαίνεται πως απεικονίζεται ένα πλέγμα τετραγώνων όταν υπάρχει θετική ή αρνητική ακτινική παραμόρφωση.



Εικόνα 3.8: Θετική ακτινική παραμόρφωση.



Εικόνα 3.9: Αρνητική ακτινική παραμόρφωση.

Έχουν επινοηθεί πολλές μέθοδοι για τον υπολογισμό των εσωτερικών παραμέτρων. Ένας από τους πιο βασικούς είναι τραβώντας φωτογραφίες μιας σκακιέρας γνωστών διαστάσεων από διάφορες οπτικές γωνίες και χρησιμοποιώντας αλγορίθμους που με αυτά τα στοιχεία μπορούν να βρουν τις παραμέτρους. Η διαδικασία εύρεσης των εσωτερικών παραμέτρων συχνά ονομάζεται “ρύθμιση” της κάμερας (camera calibration), με μια κάμερα της οποίας οι παράμετροι είναι γνωστοί να αναφέρεται ως “ρυθμισμένη” (calibrated). Η εστιακή απόσταση και το οπτικό κέντρο από μαθηματικής άποψης παρουσιάζονται ως ένας πίνακας 3x3, συνήθως συμβολισμένος με το γράμμα K, με την παρακάτω μορφή:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Όπου:

- $\mathbf{f_x}$, $\mathbf{f_y}$: Οι αποστάσεις από το κέντρο του φακού της κάμερας μέχρι το εστιακό του σημείο στους άξονες x και y αντίστοιχα.
- $\mathbf{c_x}$, $\mathbf{c_y}$: Το οπτικό κέντρο της κάμερας

3.6.2 – Εξωτερικές παράμετροι της κάμερας

Μια άλλη κατηγορία παραμέτρων της κάμερας είναι οι εξωτερικές παράμετροι ή με τον συνώνυμο όρο “camera pose”. Σε αντίθεση με τις εσωτερικές, αυτές δεν εξαρτώνται από τα κατασκευαστικά χαρακτηριστικά της κάμερας, αλλά από την θέση και τον προσανατολισμό της μέσα σε μια σκηνή. Αυτά τα μεγέθη είναι σχετικά ως προς ένα νοητό σύστημα συντεταγμένων εντός της σκηνής, γεγονός που τα καθιστά αυθαίρετα και περιττά για αρκετές εφαρμογές. Ωστόσο, στη προκειμένη περίπτωση οι εξωτερικές παράμετροι είναι χρήσιμες, καθώς ο προσανατολισμός της κάμερας σε σχέση με το αντικείμενο που ανιχνεύθηκε απαιτείται για την σωστή αναπαράσταση ενός οπτικού ψηφιακού στοιχείου στο χώρο της σκηνής.

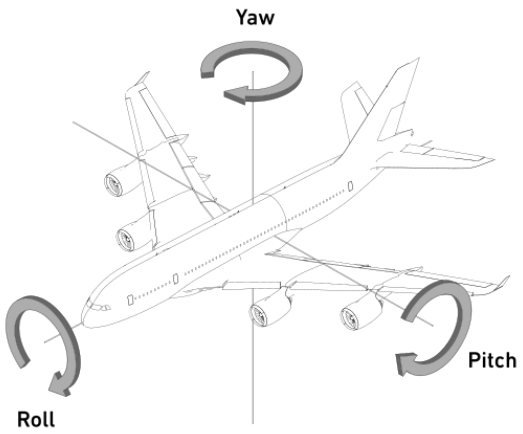
Οι εξωτερικές παράμετροι αποτελούνται από έναν πίνακα 3x3, συνήθως συμβολισμένος ως R, που ονομάζεται “πίνακας περιστροφής” (rotation matrix) και από ένα τρισδιάστατο διάνυσμα t με όνομα “διάνυσμα μετατόπισης” (translation vector). Αυτά τα δύο σύνολα συνήθως ενοποιούνται σε έναν πίνακα P 3x4, ή σε έναν 4x4 του οποίου η τελευταία γραμμή είναι αυτή του μοναδιαίου πίνακα.
Από τα παραπάνω δεδομένα, χρήσιμος για την συγκεκριμένη εφαρμογή είναι μόνο ο πίνακας περιστροφής, καθώς οι αλλαγές που προκαλεί η μετατόπιση στην εμφάνιση του ψηφιακού αντικειμένου είναι αμελητέες.

$$P = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Για τον υπολογισμό των εξωτερικών παραμέτρων είναι απαραίτητες οι εσωτερικές παράμετροι καθώς και προβολές σημείων του αντικειμένου από το νοητό, τρισδιάστατο σύστημα συντεταγμένων πάνω στο δισδιάστατο σύστημα της κάμερας. Το πρόβλημα εύρεσης των παραμέτρων βάσει αυτών των στοιχείων ονομάζεται Perspective-n-Point (PnP). Στην πιο απλή μορφή του, το πρόβλημα απαιτεί n = 3 σημεία να λυθεί, ωστόσο με μόνο 3 σημεία μπορούν να παραχθούν τέσσερις πιθανές λύσεις από τις οποίες οι τρεις θα πρέπει να απορριφθούν, για να προκύψει το σωστό αποτέλεσμα. Στη γενική περίπτωση όπου n ≥ 4 έχουν επινοηθεί αρκετοί αλγόριθμοι που λύνουν το πρόβλημα σε γραμμικό χρόνο (O(n)) ως προς τον αριθμό των σημείων, γεγονός που καθιστά την χρήση τους σε εφαρμογές πραγματικού χρόνου εφικτή. Μερικοί από τους αλγόριθμους επίλυσης είναι οι EPnP, DLS, OPnP, GPnP και UPnP.

3.6.3 – Υπολογισμός των γωνιών περιστροφής

Επιλύοντας το πρόβλημα PnP προκύπτει ο πίνακας περιστροφής. Απομένει η εξαγωγή των γωνιών περιστροφής (yaw, pitch, roll) για τον κάθε άξονα από αυτόν ώστε αυτές τελικά να εφαρμοστούν στο ψηφιακό αντικείμενο.



Εικόνα 3.10: Οπτική αναπαράσταση των αξόνων περιστροφής. (Aashmango4793 / CC BY-SA (https://creativecommons.org/licenses/by-sa/4.0))

Ο πίνακας περιστροφής προκύπτει από τον πολλαπλασιασμό 3 επιμέρους πινάκων, ένα για κάθε άξονα.

$$R_x(\theta_x)=\begin{bmatrix}1 & 0 & 0\\ 0 & cos\theta_x & -sin\theta_x\\ 0 & sin\theta_x & cos\theta_x\end{bmatrix}$$

$$R_y(\theta_y)=\begin{bmatrix}cos\theta_y & 0 & sin\theta_y\\ 0 & 1 & 0\\ -sin\theta_y & 0 & cos\theta_y\end{bmatrix}$$

$$R_z(\theta_z)=\begin{bmatrix}cos\theta_z & -sin\theta_z & 0\\ sin\theta_z & cos\theta_z & 0\\ 0 & 0 & 1\end{bmatrix}$$

Όπου θ η γωνία περιστροφής σε κάθε άξονα.

Τόσο ο τελικός όσο και οι επιμέρους πίνακες μπορούν να χρησιμοποιηθούν για να βρεθεί η θέση ενός σημείου, όταν το σύστημα αξόνων περιστραφεί κατά κάποια

γωνία. Δεδομένου του σημείου (x, y, z), για να υπολογιστεί η νέα θέση του (x΄, y΄, z΄) μετά από περιστροφή του άξονα y κατά 90°, πρέπει να γίνει η πράξη:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos 90^o & 0 & \sin 90^o \\ 0 & 1 & 0 \\ -\sin 90^o & 0 & \cos 90^o \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Σε αυτό το σημείο φανερώνεται μια σημαντική αδυναμία αυτού του τρόπου αναπαράστασης της περιστροφής. Η σειρά εκτέλεσης του πολλαπλασιασμού των επιμέρους πινάκων περιστροφής επηρεάζει το αποτέλεσμα, καθώς η επιμεριστική ιδιότητα δεν ισχύει στον πολλαπλασιασμό πινάκων. Αυτό μπορεί να οδηγήσει σε ασάφειες, αφού ο τελικός πίνακας R δεν επαρκεί για να οριστεί πλήρως μία περιστροφή, αλλά χρειάζεται και η σειρά των επιμέρους περιστροφών. Οι 6 πιθανές περιπτώσεις (XYZ, XZY, YXZ, YZX, ZXY, ZYX) είναι όλες σωστές, αρκεί μόλις επιλεχθεί μια να διατηρηθεί η χρήση της καθ' όλη τη διάρκεια των πράξεων. Επιπλέον, αποδεικνύεται ότι κάθε πιθανός προσανατολισμός στον χώρο μπορεί να περιγραφεί περιστρέφοντας τον ένα από τους άξονες δύο φορές, αρκεί αυτές οι περιστροφές να μην είναι διαδοχικές και να χωρίζονται από την περιστροφή άλλου άξονα. Αυτό το γεγονός δημιουργεί άλλες 6 εξίσου σωστές σειρές περιστροφής (XYX, XZX, YXY, YZY, ZXZ, ZYZ). Παρακάτω αναλύεται η περίπτωση ZYX, καθώς είναι η πιο καθιερωμένη και χρησιμοποιείται από τη βιβλιοθήκη OpenCV.

$$R = R_z \cdot R_y \cdot R_x$$

Μετά την εκτέλεση των πολλαπλασιασμών προκύπτει ο παρακάτω τελικός πίνακας περιστροφής:

$$R=\begin{bmatrix} cos\theta_z \cdot cos\theta_y & -sin\theta_z \cdot cos\theta_x + cos\theta_z \cdot sin\theta_y \cdot sin\theta_x & sin\theta_z \cdot sin\theta_x + cos\theta_z \cdot sin\theta_y \cdot cos\theta_x \\ sin\theta_z \cdot cos\theta_y & cos\theta_z \cdot cos\theta_x + sin\theta_z \cdot sin\theta_y \cdot sin\theta_x & -cos\theta_z \cdot sin\theta_x + sin\theta_z \cdot sin\theta_y \cdot cos\theta_x \\ -sin\theta_y & cos\theta_y \cdot sin\theta_x & cos\theta_y \cdot cos\theta_x \end{bmatrix}$$

Παρατηρώντας τον πίνακα, φαίνεται ότι οι γωνίες μπορούν να εξαχθούν με χρήση των εξής τύπων:

$$\theta_y = -\arcsin(R_{31})$$

$$\theta_x = \arctan\frac{R_{32}}{R_{33}}$$

$$\theta_z = \arctan\frac{R_{21}}{R_{11}}$$

Αυτοί οι τύποι αποδεικνύονται εύκολα όπως φαίνεται παρακάτω:

$$-\arcsin(R_{31}) = -\arcsin(-\sin(\theta_y)) = \arcsin(\sin(\theta_y)) = \theta_y$$

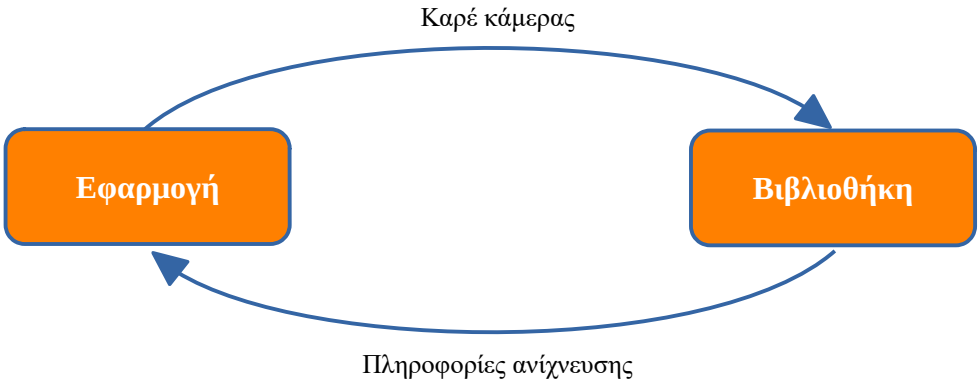
$$\arctan\frac{R_{32}}{R_{33}} = \arctan\frac{\cancel{cos\theta_y} \cdot sin\theta_x}{\cancel{cos\theta_y} \cdot cos\theta_x} = \arctan(\tan(\theta_x)) = \theta_x$$

$$\arctan\frac{R_{21}}{R_{11}} = \arctan\frac{\cancel{cos\theta_y} \cdot sin\theta_z}{\cancel{cos\theta_y} \cdot cos\theta_z} = \arctan(\tan(\theta_z)) = \theta_z$$

ΚΕΦΑΛΑΙΟ 4 - ΥΛΟΠΟΙΗΣΗ

4.1 – Γενική δομή

Το λογισμικό που κατασκευάζεται στα πλαίσια της παρούσας διπλωματικής αποτελείται από δύο μέρη. Το πρώτο είναι μία εφαρμογή για το λειτουργικό σύστημα Android υλοποιημένη με C# στη μηχανή Unity. Το δεύτερο είναι μια βιβλιοθήκη κώδικα σε μορφή αρχείου κατάληξης .so (Shared Object) γραμμένη σε C++, η οποία θα φορτώνεται δυναμικά από την εφαρμογή κατά την εκτέλεση. Η εφαρμογή αναλαμβάνει την ανάγνωση των αρχείων παραμετροποίησης και την εμφάνιση ή αναπαραγωγή του ψηφιακού περιεχομένου. Η βιβλιοθήκη θα εμπεριέχει την OpenCV και θα την χρησιμοποιεί, για να ελέγξει αν υπάρχει κάποια από τις σελίδες του βιβλίου στο καρέ. Τα αποτελέσματα της ανίχνευσης επιστρέφονται στην εφαρμογή, ώστε αυτή να δράσει ανάλογα. Η διαδικασία αποστολής καρέ, εύρεσης σελίδας και εμφάνισης ψηφιακού περιεχομένου επαναλαμβάνεται έως ότου τον τερματισμό της από τον χρήστη.



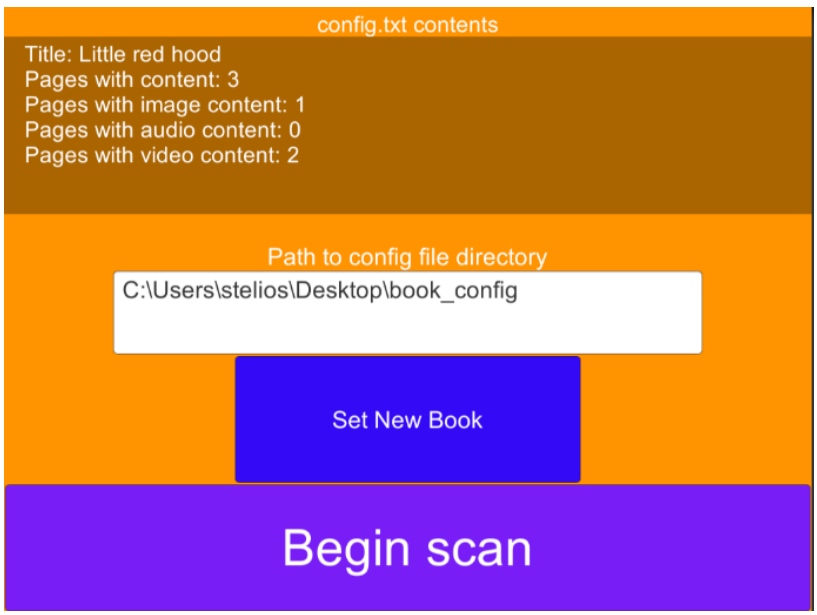
Εικόνα 4.1: Μετάφορα δεδομένων ανάμεσα σε βιβλιοθήκη και εφαρμογή.

4.2 – Εφαρμογή

Η εφαρμογή έχει δυνατότητα να παραμετροποιηθεί ως προς το βιβλίο το οποίο θα αναγνωρίζει και το ψηφιακό περιεχόμενο που θα εμφανίζει ή θα αναπαράγει. Η επιλογή αυτών γίνεται μέσω αρχείων στα οποία θα οδηγείται από τον χρήστη. Αυτά πρέπει να περιέχουν όλη την απαραίτητη πληροφορία για την λειτουργία της εφαρμογής. Πιο συγκεκριμένα, τα στοιχεία που πρέπει να περιλαμβάνονται στα αρχεία είναι τα παρακάτω:

- Σκαναρισμένες εικόνες των σελίδων του βιβλίου στις οποίες θα εμφανίζεται το ψηφιακό περιεχόμενο καθώς και σε ποια σελίδα αντιστοιχεί η κάθε εικόνα.
- Τα μέσα που θα αναπαράγονται ψηφιακά σε μορφή αρχείων εικόνας, ήχου και βίντεο.
- Μία σύνδεση μεταξύ των μέσων και των σελίδων στις οποίες θα εμφανίζονται.

Οι εικόνες των σελίδων αποστέλλονται μαζί με την αρίθμηση τους στη βιβλιοθήκη, ώστε να γίνει η αρχικοποίηση της και να τεθεί σε θέση να ανιχνεύσει τις σελίδες στο καρέ.



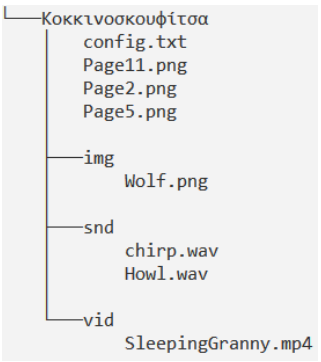
Εικόνα 4.2: Το περιβάλλον χρήστη της εφαρμογής.

4.3 – Δομή αρχείων παραμετροποίησης εφαρμογής

Για να επιτευχθεί η παραμετροποίηση, ο χρήστης πρέπει να ορίσει το μονοπάτι (path) στο φάκελο με τα δεδομένα ενός συγκεκριμένου βιβλίου. Ο φάκελος θα περιέχει τις εικόνες των σελίδων του βιβλίου, όλα τα μέσα που θα αναπαράγονται για αυτό και ένα αρχείο κείμενου με το όνομα “config.txt” για την σύνδεση αυτών μεταξύ τους. Το αρχείο θα περιέχει στην πρώτη γραμμή την συμβολοσειρά “title:” ακολουθούμενη από τον τίτλο του βιβλίου. Στις υπόλοιπες γραμμές θα φαίνονται τα πολυμέσα που θα χρησιμοποιούνται ανά σελίδα. Για καθεμία από τις τελευταίες, θα υπάρχει η λέξη “page:” ακολουθούμενη από τον αριθμό της. Από κάτω θα τοποθετούνται το πολύ δύο γραμμές με τα σχετικά, ως προς τον φάκελο του config.txt, μονοπάτια των πολυμέσων που θα αναπαράγονται στη συγκεκριμένη σελίδα. Η κάθε γραμμή θα αφορά ένα διαφορετικό τύπο υποστηριζόμενου πολυμέσου (εικόνα, ήχος, βίντεο), με τον περιορισμό ότι μια σελίδα δεν μπορεί να περιέχει συγχρόνως εικόνα και βίντεο σαν ψηφιακό περιεχόμενο. Η δομή της σελίδας θα επαναλαμβάνεται για καθεμιά από αυτές που έχουν ψηφιακό περιεχόμενο. Ένα παράδειγμα ενός πιθανού φακέλου δεδομένων για το παραμύθι της Κοκκινοσκουφίτσας καθώς και του αντίστοιχου αρχείου παραμετροποίησης φαίνεται στις παρακάτω εικόνες:

```
title: Little red hood

page 2:
    sound:snd/chirp.wav

page 5:
    image:img/Wolf.png
    sound:snd/Howl.wav
page 11:
    video:vid/SleepingGranny.mp4
```

Εικόνα 4.3: Παράδειγμα αρχείου config.txt



Εικόνα 4.4: Φάκελος δεδομένων

Τα δεδομένα του παραπάνω παραδείγματος αντιστοιχούν σε ένα βιβλίο το οποίο περιέχει σαν ψηφιακό περιεχόμενο 2 κομμάτια ήχου στις σελίδες 2 και 5, μία εικόνα στη σελίδα 5 και ένα βίντεο στη σελίδα 11.

4.4 – Βιβλιοθήκη

Η βιβλιοθήκη αναλαμβάνει την αναζήτηση των σελίδων εντός του καρέ της κάμερας χρησιμοποιώντας τους αλγόριθμους επεξεργασίας εικόνας που περιλαμβάνει η OpenCV. Για να επιτευχθεί αυτό, πρέπει πριν την εκκίνηση της ανίχνευσης να δοθούν στη βιβλιοθήκη οι σελίδες του βιβλίου ώστε να τις περάσει από την διαδικασία εύρεσης των σημείων κλειδιών και των περιγραφέων αυτών. Όταν βρεθεί μια σελίδα τότε στο κυρίως πρόγραμμα επιστρέφονται:

- Ο αριθμός της σελίδας.
- Οι συντεταγμένες ως προς την κάμερα του κέντρου της σελίδας.
- Ο πίνακας περιστροφής R που περιγράφει την περιστροφή του βιβλίου σε σχέση με την κάμερα.

Αυτά τα στοιχεία επαρκούν για να αναπαραχθεί το ψηφιακό περιεχόμενο από το κυρίως πρόγραμμα.

Η γλώσσα υλοποίησης της βιβλιοθήκης είναι η C++. Ο λόγος αυτής της επιλογής είναι το γεγονός ότι ο κώδικας της τρέχει απευθείας στον επεξεργαστή της εκάστοτε συσκευής, σε αντίθεση με την C# που εκτελείται σε ξεχωριστό απομονωμένο περιβάλλον. Το τελευταίο έχει αντίκτυπο στην ταχύτητα εκτέλεσης, το οποίο δεν μπορεί να αγνοηθεί σε μια εφαρμογή που στοχεύει φορητές συσκευές. Επίσης, η ανάπτυξη της βιβλιοθήκης σε C++ βοηθάει στην ευκολότερη ενσωμάτωση της OpenCV, καθώς και αυτή είναι προγραμματισμένη στην ίδια γλώσσα.

4.5 – Επεξεργασία δεδομένων ανίχνευσης

4.5.1 – Ανάγκη επεξεργασίας

Κατά την διάρκεια φόρτωσης των εικόνων και των σταδίων ανίχνευσης, όπως αυτά επεξηγήθηκαν στο κεφάλαιο 3, είναι σημαντική η σωστή επεξεργασία των δεδομένων, ώστε επιτευχθεί η βέλτιστη ποιότητα ανίχνευσης. Στις ακόλουθες παραγράφους θα αναλυθούν οι μέθοδοι επεξεργασίας των δεδομένων που οδηγούν σε βελτίωση της ανίχνευσης, τόσο από άποψη ακρίβειας, όσο και από άποψη ταχύτητας.

4.5.2 – Προ επεξεργασία των train images

Το στάδιο της επεξεργασίας ξεκινά από τις train images, δηλαδή τις εικόνες των σελίδων του βιβλίου, πριν αυτές σταλθούν στον αλγόριθμο ανίχνευσης των keypoints. Η προ επεξεργασία περιλαμβάνει:

1. Μετατροπή των τριών καναλιών χρώματος της εικόνας σε ένα (grayscale).
2. Μείωση του μεγέθους της εικόνας.
3. Εφαρμογή φίλτρου Gaussian blur με πίνακα 3x3 στην εικόνα.

Η μετατροπή της εικόνας σε grayscale είναι απαιτούμενο από τους αλγόριθμους ανίχνευσης, καθώς στις περισσότερες περιπτώσεις η πληροφορία του χρώματος δεν είναι χρήσιμη στην εύρεση γωνιών ή άλλων χρήσιμων χαρακτηριστικών. Επιπλέον, οι εικόνες με ένα μόνο κανάλι είναι γρηγορότερες στη διαχείριση και χρησιμοποιούν λιγότερη από τη μνήμη του συστήματος με αποτέλεσμα όλα τα υπόλοιπα στάδια να πραγματοποιούνται γρηγορότερα.

Το επόμενο βήμα της μείωσης των διαστάσεων γίνεται για δύο λόγους. Πρώτον, μια εικόνα μικρού μεγέθους είναι πολύ γρηγορότερη στην επεξεργασία. Δεύτερον, η σμίκρυνση οδηγεί σε καλύτερη ανίχνευση, καθώς η ύπαρξη πολλών pixels έχει ως αποτέλεσμα την εμφάνιση χαρακτηριστικών τα οποία αποτελούνται από μικρές και

ασήμαντες λεπτομέρειες της εικόνας. Αυτά τα keypoints είναι δύσκολο να ταιριάξουν σε μια ελαφρώς αλλαγμένη εκδοχή της που θα εμφανιστεί στην σκηνή. Συνεπώς αυτό το στάδιο βελτιώνει την ανίχνευση, παρά την αφαίρεση πληροφορίας.

Τέλος, το φίλτρο Gaussian θολώνει την εικόνα, προκαλώντας μείωση του θορύβου. Ο τελευταίος μπορεί να δημιουργήσει χρωματική αντίθεση σε ομοιόμορφες περιοχές, γεγονός που δύναται να εμφανίσει ανούσια keypoints όταν η εικόνα περνάει από το στάδιο ανίχνευσής τους. Ωστόσο, τα σημαντικά χαρακτηριστικά, όπως οι άκρες και οι γωνίες, μένουν ανεπηρέαστα από το φίλτρο.

4.5.3 – Επεξεργασία των keypoints στην train image

Το πλήθος των keypoints μπορεί να διαφέρει σημαντικά ανάμεσα σε διάφορες εικόνες. Είναι λογικό η ποιότητα της ανίχνευσης να είναι ανάλογη με τον αριθμό των keypoints, καθώς περισσότερα χαρακτηριστικά σημεία αυξάνουν τις πιθανότητες να βρεθούν κάποια από αυτά στην εικόνα της σκηνής. Ωστόσο, στη περίπτωση που τα keypoints είναι υπερβολικά υπεράριθμα δημιουργείται ανάγκη για πολύ επεξεργασία κατά την διάρκεια της σύγκρισης τους, με αποτέλεσμα να προκαλούνται καθυστερήσεις. Δεδομένου ότι η εφαρμογή στοχεύει συσκευές Android, οι οποίες έχουν περιορισμένη επεξεργαστική ισχύ, εμφανίζεται η ανάγκη περιορισμού των keypoints με βάση την αξία τους. Αν η εικόνα της σελίδας έχει πάνω από 800 keypoints, τότε κρατούνται μόνο αυτά για τα οποία ισχύει η παρακάτω σχέση:

$$Αξία > Factor \cdot Max$$

Όπου:

- Αξία: Η αξία του keypoint όπως αυτή υπολογίζεται από τον αλγόριθμο που το ανίχνευσε.
- Factor: Ένας συντελεστής με τιμή στο εύρος [0,1]. Στη συγκεκριμένη περίπτωση, ορίστηκε στο 0.6.
- Max: Η αξία του καλύτερου keypoint που βρέθηκε.

4.5.4 – Επεξεργασία αποτελεσμάτων ταιριάσματος

Μετά το ταίριασμα των περιγραφέων ανάμεσα στις δύο εικόνες, δημιουργούνται ζευγάρια αυτών. Αυτά πρέπει στη συνέχεια να ελεγχθούν για να αξιολογηθεί η ποιότητά τους, ώστε να απορριφθούν όσα κριθούν αναξιόπιστα. Το κύριο κριτήριο έγκρισης τους είναι η απόσταση μεταξύ των descriptor που ταίριαξαν. Θα εφαρμοστούν δύο φίλτρα πάνω στα ζευγάρια.

Το πρώτο φίλτρο αναλαμβάνει την αφαίρεση ζευγαριών που προέκυψαν από ταίριασμα μεταξύ επαναλαμβανόμενων περιοχών στις εικόνες. Αυτές οι περιοχές προκαλούν την εμφάνιση ταιριασμάτων τα οποία αν και έχουν χαμηλή απόσταση στην πραγματικότητα είναι αναξιόπιστα, καθώς θα μπορούσαν να προκύψουν ανάμεσα σε οποιοδήποτε τμήμα του επαναλαμβανόμενου μοτίβου. Αυτό το φαινόμενο φαίνεται πιο καθαρά στην παρακάτω εικόνα:

Εικόνα 4.5: Παράδειγμα λανθασμένων ταιριασμάτων. Οι περιοχές P1 και P2 θα μπορούσαν να θεωρηθούν όμοιες από έναν αλγόριθμο.

Για να αποφευχθεί αυτό, για κάθε περιγραφέα της train image θα υπολογίζονται οι δύο καλύτερα ταιριαστοί από την query. Αν αυτοί οι δύο περιγραφείς προέκυψαν από επαναλαμβανόμενο μοτίβο, τότε θα έχουν πολύ μικρή διαφορά στο distance, καθώς θα αντιστοιχούν σε παρόμοια περιοχή. Με βάση αυτό, το φίλτρο θα συγκρίνει τα δύο ζευγάρια και αν η μεταξύ τους διαφορά είναι μικρότερη από 10%, θα απορρίπτονται και τα δύο.

Το δεύτερο φίλτρο εφαρμόζεται στα ταιριάσματα που πέρασαν από το πρώτο. Σκοπός του είναι να απορρίπτει όσα ζευγάρια δεν έχουν ικανοποιητική απόσταση. Το όριο αυτού του ελέγχου είναι δυναμικό και ισούται με το τριπλάσιο της χαμηλότερης απόστασης που βρέθηκε σε όλα τα ζευγάρια. Ο συγκεκριμένος τρόπος υπολογισμού του κατωφλιού είναι ιδανικός, καθώς προσφέρει στην εφαρμογή αυξημένη προσαρμοστικότητα σε ακραία δεδομένα εισόδου, συγκριτικά με ένα στατικό όριο.

4.6 – Υπολογισμός ομογραφίας και έλεγχος σωστών ταιριασμάτων

Έχοντας υπολογίσει τα τελικά ζευγάρια, το επόμενο βήμα, όπως αναφέρθηκε στο κεφάλαιο 3, είναι να βρεθεί η ομογραφία. Αυτό πραγματοποιείται με τον αλγόριθμο RANSAC (Random Sample Consensus). Πρόκειται για έναν αλγόριθμο που μπορεί να κάνει εκτίμηση των παραμέτρων ενός μαθηματικού μοντέλου, όπως του πίνακα της ομογραφίας, αν του δοθούν πειραματικά δεδομένα. Ένα σημαντικό χαρακτηριστικό του συγκεκριμένου αλγορίθμου είναι ότι αναμένει την ύπαρξη λανθασμένων στοιχείων (outliers), και στις περισσότερες περιπτώσεις μπορεί να τα ξεχωρίσει από τα σωστά (inliers). Αυτό είναι χρήσιμο, καθώς δεν είναι σπάνιο φαινόμενο να προκύψουν λανθασμένα ταιριάσματα κατά την διάρκεια της ανίχνευσης, παρά τις προσπάθειες που έγιναν για το αντίθετο. Η OpenCV έχει ενσωματωμένο τον αλγόριθμο RANSAC για τον υπολογισμό της ομογραφίας και μπορεί, με βάση αυτόν, να χαρακτηρίσει το κάθε ζευγάρι ως inlier ή outlier. Σε αυτό το σημείο πραγματοποιείται ένας τελευταίος έλεγχος για το αν τα inliers αποτελούν

τουλάχιστον το 65% όλων των στοιχείων, ώστε να κριθεί αν η σελίδα υπάρχει όντως στη σκηνή. Σε αυτή την περίπτωση, χρησιμοποιείται η ομογραφία για τον υπολογισμό της τοποθεσίας της σελίδας σε σχέση με την κάμερα. Αυτό το τελευταίο δεδομένο τοποθετείται μαζί με τον αριθμό της σελίδας που ανιχνεύθηκε στις μεταβλητές επιστροφής της βιβλιοθήκης.

4.7 – Υπολογισμός του πίνακα περιστροφής

Επόμενο και τελευταίο βήμα είναι η εύρεση του πίνακα περιστροφής. Αυτό, όπως αναφέρθηκε στο κεφάλαιο 3.6.2, πραγματοποιείται με χρήστη ενός αλγορίθμου επίλυσης του προβλήματος PnP. Η βιβλιοθήκη OpenCV παρέχει υλοποιήσεις για αρκετούς από τους αλγόριθμους που έχουν επινοηθεί για το συγκεκριμένο θέμα. Μετά από δοκιμές, παρατηρήθηκε ότι ο πιο κατάλληλος για τη συγκεκριμένη περίπτωση είναι ο EPnP (Efficient Perspective-n-Point) από απόψεις ταχύτητας και αποτελεσματικότητας.

Προαπαιτούμενες για όλους τους αλγόριθμους είναι οι εσωτερικές παράμετροι της κάμερας. Δεδομένου ότι η διαδικασία ρύθμισης της κάμερας είναι σχετικά εξειδικευμένη, δεν μπορούσε να θεωρηθεί ότι ο χρήστης της εφαρμογής θα είναι ικανός να την ολοκληρώσει επιτυχώς. Συνεπώς η εφαρμογή χρησιμοποιεί προσεγγιστικές τιμές για τις εσωτερικές παραμέτρους, θυσιάζοντας λίγη ακρίβεια στα αποτελέσματα. Πιο συγκεκριμένα, το οπτικό κέντρο ορίστηκε ως το κέντρο της εικόνας της κάμερας, η εστιακή απόσταση ως το μήκος της, ενώ οι συντελεστές παραμόρφωσης θεωρήθηκαν μηδενικοί. Παρά τις προσεγγίσεις η ακρίβεια της εφαρμογής δεν επηρεάζεται σε σημαντικό βαθμό. Ο πίνακας περιστροφής που υπολογίστηκε από τον αλγόριθμο αντιγράφεται στο τμήμα μνήμης που έχει δεσμευτεί για αυτόν από το κυρίως πρόγραμμα. Με το τελευταίο βήμα, ολοκληρώθηκε ο κύκλος ανίχνευσης για ένα καρέ της κάμερας.

4.8 – Επιτάχυνση ανίχνευσης

Κατά την διάρκεια εκτέλεσης δοκιμών παρατηρήθηκε σημαντική πτώση στην ταχύτητα ανίχνευσης ακόμα και όταν οι σελίδες προς ανίχνευση ήταν λίγες. Αυτό είναι αναμενόμενο, καθώς ο αλγόριθμος ανίχνευσης με όλα τα στάδια που περιγράφηκαν σε αυτό το κεφάλαιο πρέπει να εκτελείται για κάθε καρέ της κάμερας, δηλαδή περίπου 30 ή 60 φορές το δευτερόλεπτο. Όταν αυτό δεν είναι δυνατό, τότε μερικά καρέ δεν περνούν από επεξεργασία και δεν εμφανίζονται ποτέ στην οθόνη, γεγονός που οδηγεί σε πτώση του F.P.S. (Frames Per Second). Οι αλγόριθμοι της OpenCV είναι ήδη βελτιστοποιημένοι και δεν υπάρχουν σημαντικά περιθώρια επιτάχυνσης στην υπόλοιπη διαδικασία, οπότε πρέπει να βρεθεί εναλλακτική μέθοδος αύξησης της ταχύτητας ανίχνευσης.

Για την επίλυση αυτού του προβλήματος έγινε τροποποίηση του αλγόριθμου έτσι ώστε όταν ανιχνευθεί επιτυχώς μια σελίδα, στα επόμενα 3 καρέ θα γίνει αναζήτηση μόνο της συγκεκριμένης. Δεδομένου ότι ο μέσος χρήστης θα παραμένει σε μια σελίδα για μερικά λεπτά, ώστε να διαβάσει το περιεχόμενό της, δεν υπάρχει λόγος να γίνεται πάντα αναζήτηση όλων των σελίδων. Επιπλέον, όταν ο αναγνώστης αλλάξει σελίδα, οι 3 εσφαλμένες αναζητήσεις δεν προκαλούν σημαντική καθυστέρηση, καθώς πραγματοποιούνται πάνω μόνο σε μια εικόνα και όχι σε όλο το σύνολο. Η φαινομενική επιτάχυνση που προκαλεί αυτό το σύστημα είναι φανερή, παρόλο που το τμήμα του αλγορίθμου που ασχολείται άμεσα με την ανίχνευση δεν τροποποιήθηκε.

ΚΕΦΑΛΑΙΟ 5 – ΑΝΑΠΤΥΞΗ

5.1 – Εισαγωγή

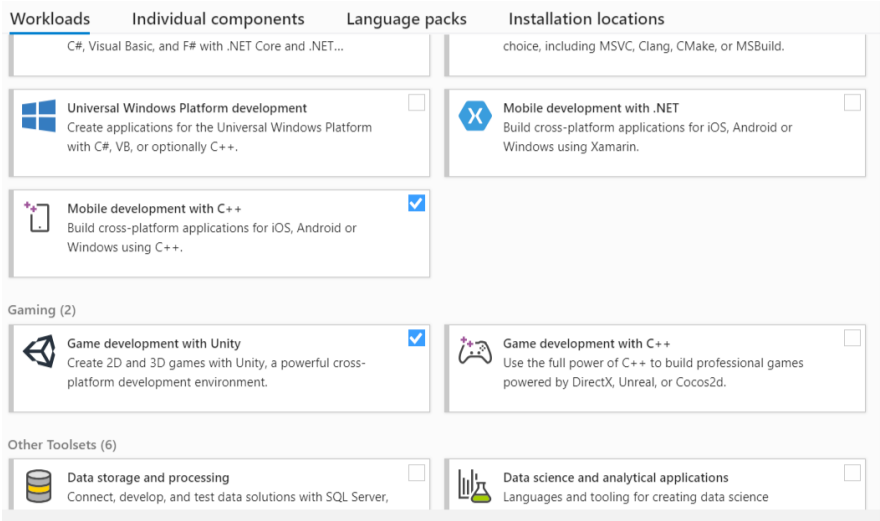
Το πέμπτο κεφάλαιο αφορά το πρακτικό μέρος της ανάπτυξης. Θα γίνει περιγραφή του τρόπου κατασκευής των συστημάτων που αναλύθηκαν στο προηγούμενο κεφάλαιο, επιλεκτική ανάλυση του κώδικα και επεξήγηση άλλων θεμάτων που προέκυψαν κατά την ανάπτυξη. Ο πλήρης κώδικας ανά αρχείο της εφαρμογής και της βιβλιοθήκης βρίσκεται στο παράρτημα.

5.2 – Λογισμικά που χρησιμοποιήθηκαν

Για την ανάπτυξη της εφαρμογής, χρησιμοποιήθηκαν τα παρακάτω λογισμικά/πακέτα:

- Η μηχανή Unity 2019.3.6f1
- Το περιβάλλον ανάπτυξης Visual Studio Community
- Η Android έκδοση της βιβλιοθήκης OpenCV 3.4.10

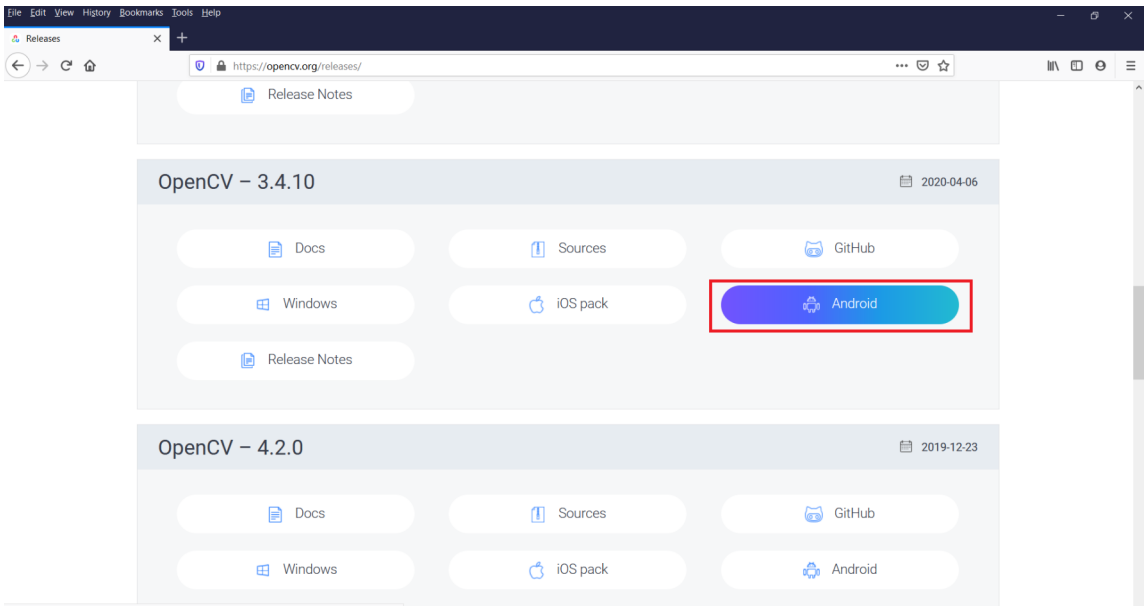
Επιπλέον στο Visual Studio εγκαταστάθηκαν τα πακέτα “Mobile development with C++” για την ανάπτυξη της βιβλιοθήκης και “Game development with Unity” για την σωστή συνεργασία του Visual Studio με την Unity.



Εικόνα 5.1: Επιπλέον πακέτα επέκτασης του Visual Studio

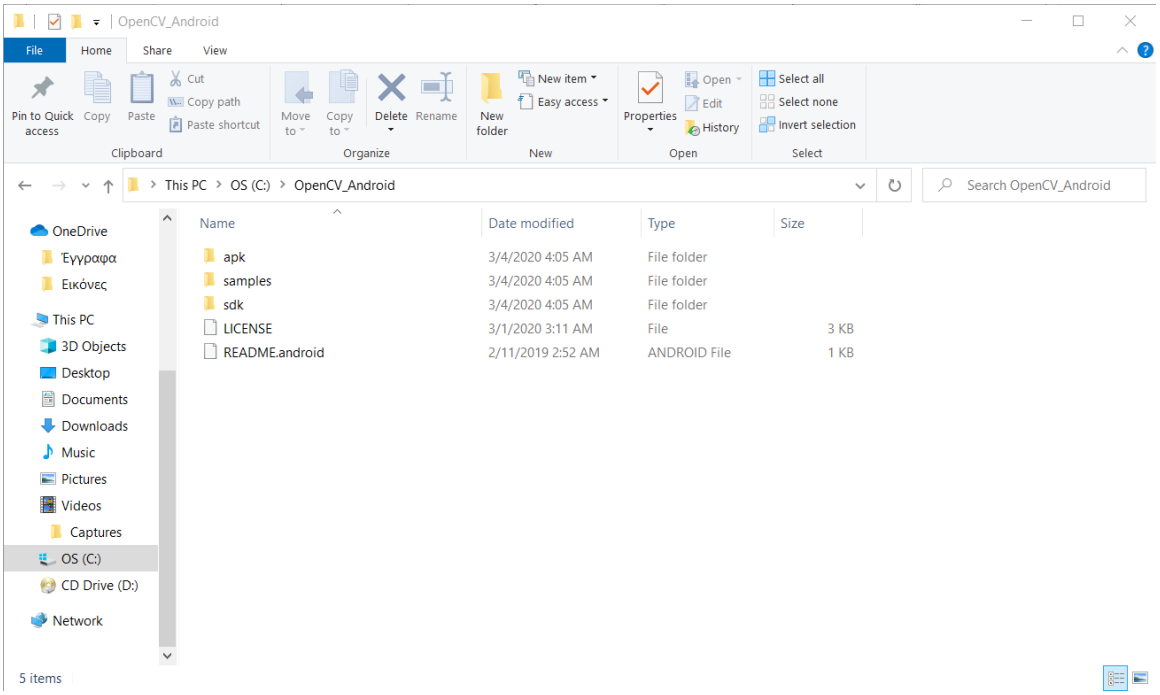
5.3 – Δημιουργία του project της βιβλιοθήκης

Για να δημιουργηθεί το project της βιβλιοθήκης, ακολουθείται η εξής διαδικασία: Αρχικά γίνεται κατέβασμα της OpenCV για το Android από την επίσημη ιστοσελίδα της βιβλιοθήκης (https://opencv.org/releases/)



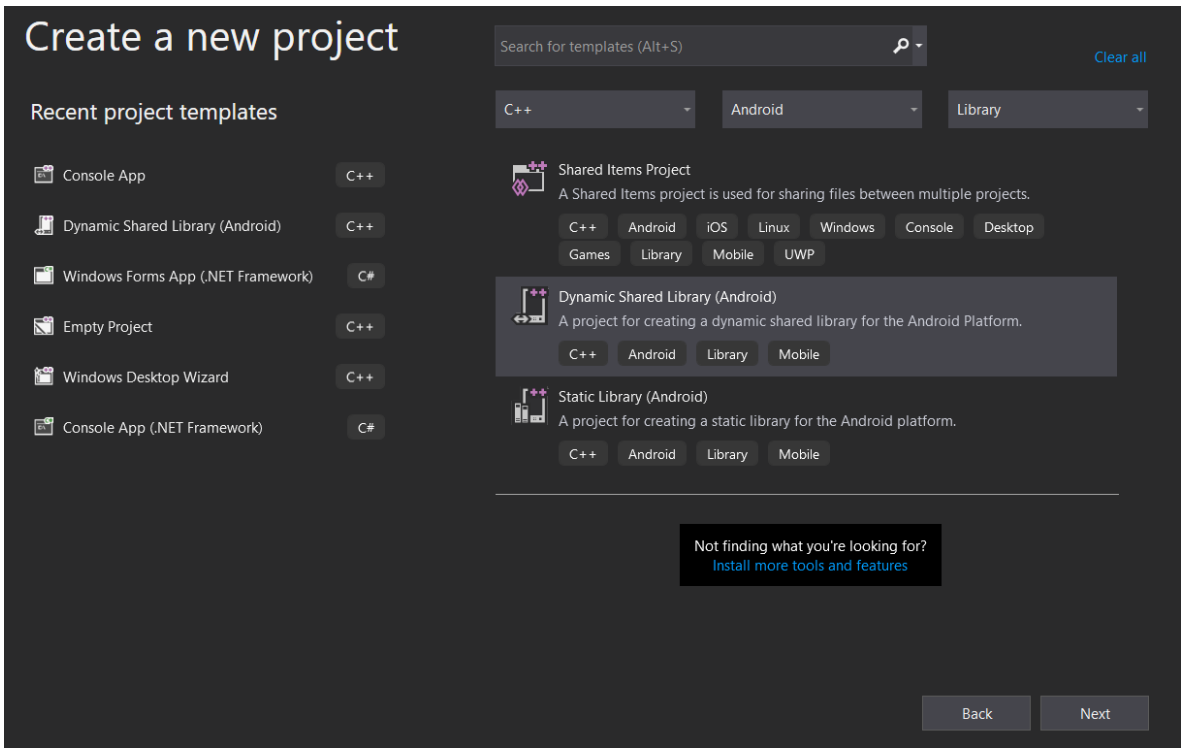
Εικόνα 5.2: Κατέβασμα της OpenCV

Μετά την εξαγωγή του αρχείου, προκύπτει ένας φάκελος με 2 αρχεία και 3 υποφακέλους.



Εικόνα 5.3: Περιεχόμενα του φακέλου της OpenCV

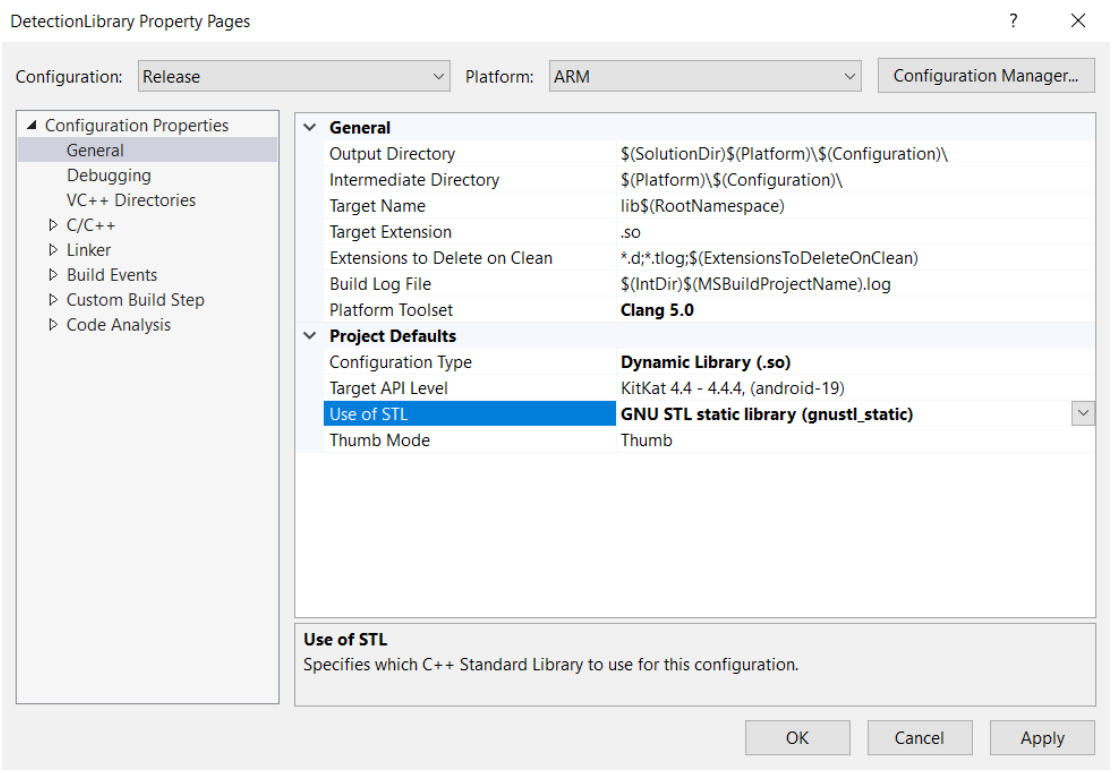
Η βιβλιοθήκη περιλαμβάνεται στον φάκελο sdk, οπότε τα υπόλοιπα μπορούν να διαγραφούν. Επόμενο βήμα είναι η δημιουργία ενός νέου project στο Visual Studio κατηγορίας “Dynamic Shared Library (Android)”. Αν αυτή η κατηγορία δεν υπάρχει, τότε δεν έχει εγκατασταθεί το πακέτο “Mobile Development with C++" που αναφέρθηκε προηγουμένως.



Εικόνα 5.4: Περιβάλλον δημιουργίας project στο Visual Studio

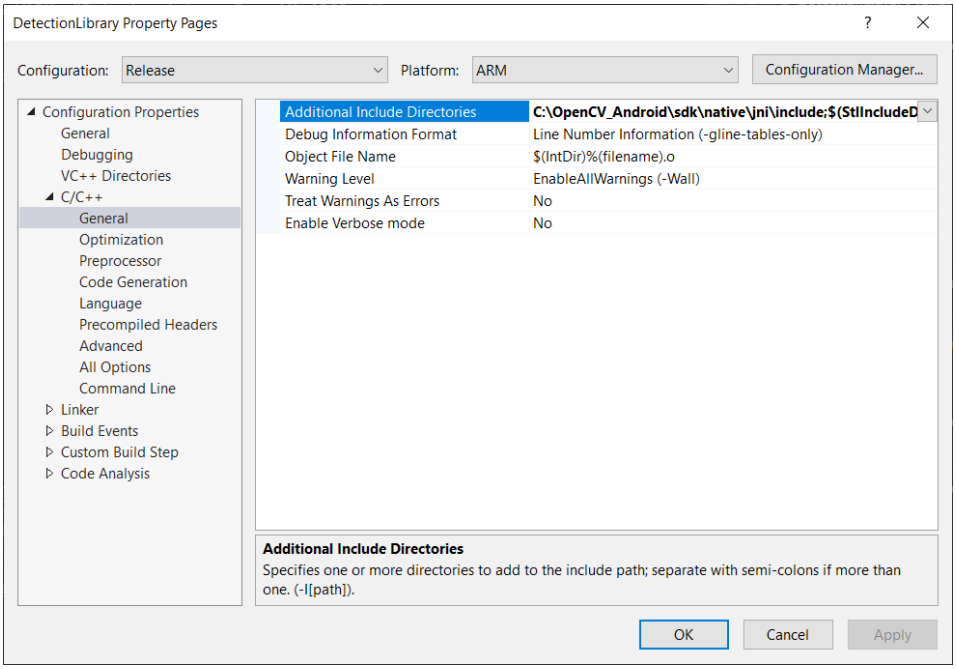
Σε αυτό το σημείο θα πρέπει να γίνουν κάποιες αλλαγές στις ρυθμίσεις του project, ώστε να μπορέσει να συνδεθεί επιτυχώς με την OpenCV.

1) Ανοίγοντας τις ρυθμίσεις η πρώτη αλλαγή βρίσκεται στη καρτέλα General, όπου η STL (Standard Template Library) της C++ πρέπει να οριστεί ως “GNU STL static library”.



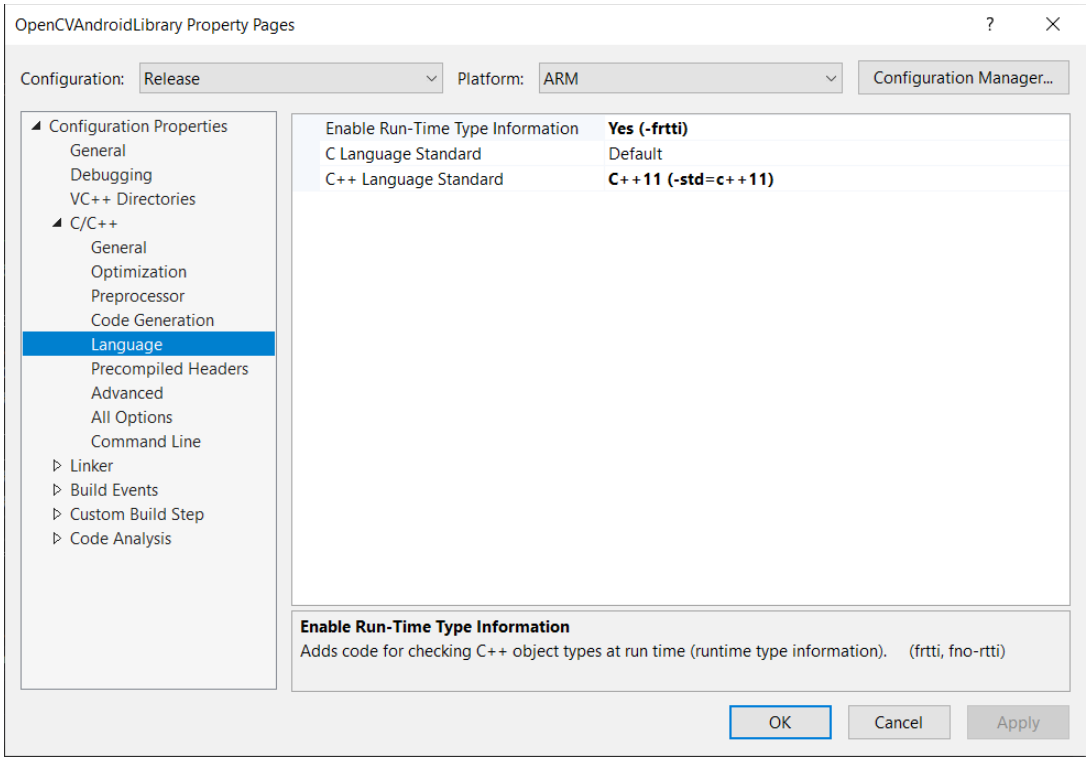
Εικόνα 5.5: Ρύθμιση της Standard Template Library

2) Η επόμενη αλλαγή βρίσκεται στην καρτέλα C/C++ → General, όπου στα **Additional Include Directories** πρέπει να προστεθεί το μονοπάτι προς τα header files της OpenCV, ώστε να μπορούν να γίνουν include στον κώδικα. Τα header files βρίσκονται στον φάκελο **sdk\native\jni\include**.



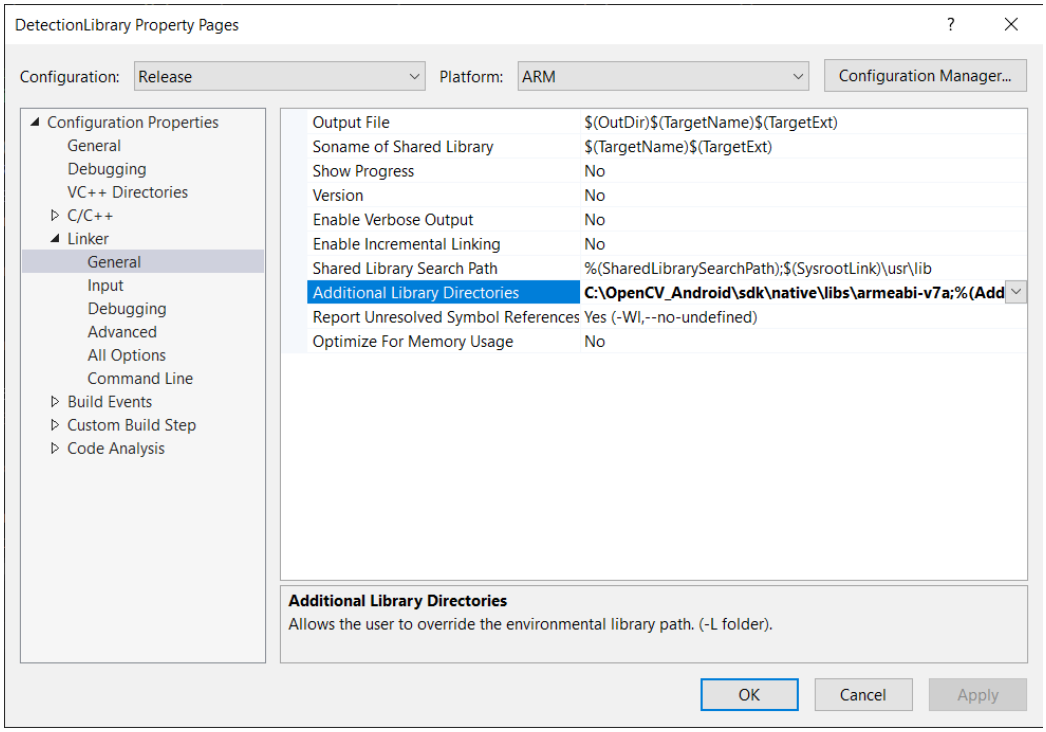
Εικόνα 5.6: Ρύθμιση του φακέλου με τα header files

3) Στην καρτέλα C/C++ → Language η ρύθμιση "**Enable Run-Time Type Information**" ορίζεται σε **Yes (-frtti)** και, επιπλέον, επιλέγεται η έκδοση **C++ 11**. Η τελευταία δεν είναι απαραίτητη για την ενσωμάτωση της OpenCV, αλλά χρειάζεται διότι αξιοποιείται η συνάρτηση std::move, η οποία δεν υπάρχει στην προεπιλεγμένη έκδοση.



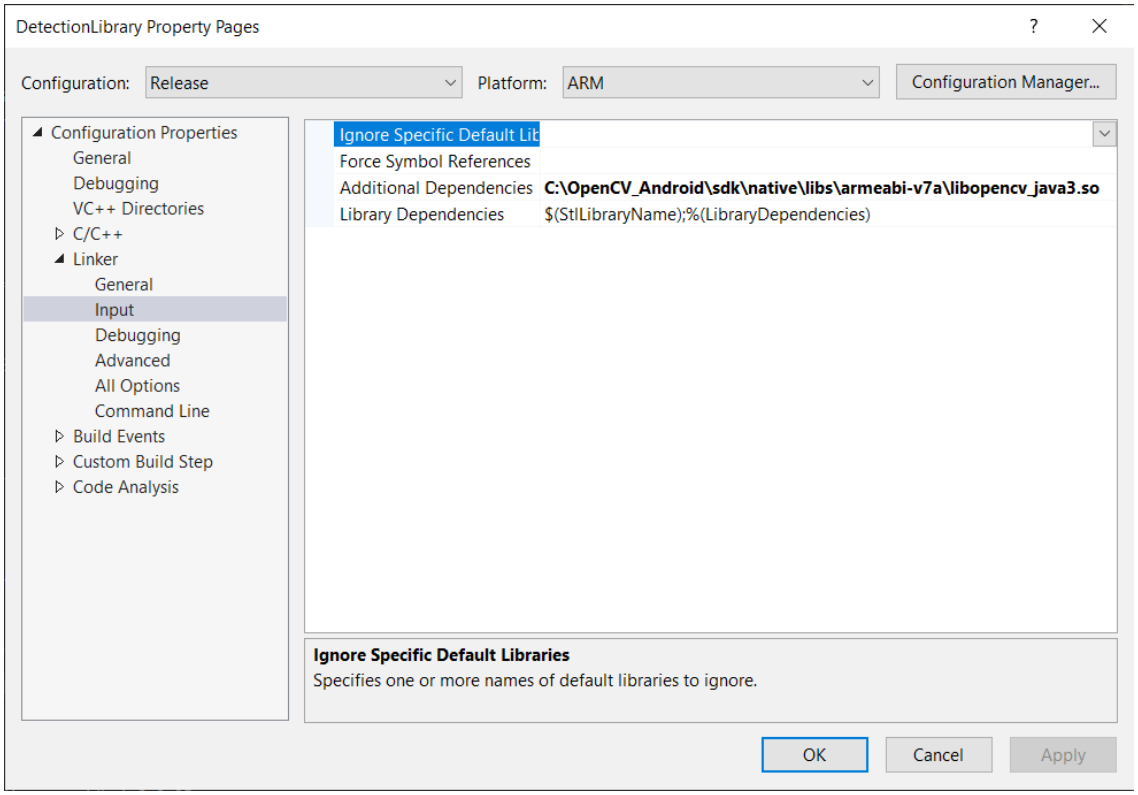
Εικόνα 5.7: Ρυθμίσεις της καρτέλας language

4) Στη συνέχεια ο Linker της C++ πρέπει να οδηγηθεί στα binaries της βιβλιοθήκης, τα οποία βρίσκονται στον φάκελο **sdk\native\libs\armeabi-v7a**. Αυτό επιτυγχάνεται με τις επόμενες δύο ρυθμίσεις. Στην καρτέλα Linker → General θα προστεθεί το παραπάνω μονοπάτι στην ρύθμιση “**Additional Library Directories**”



Εικόνα 5.8: Επιλογή του φακέλου με τις βιβλιοθήκες

Και τέλος, στην αμέσως επόμενη καρτέλα Linker → Input στη ρύθμιση "**Additional Dependencies**" θα τοποθετηθεί το πλήρες μονοπάτι του binary αρχείου της OpenCV. Πλέον, το project είναι έτοιμο να λειτουργήσει μαζί με την OpenCV.



Εικόνα 5.9: Επιλογή του μονοπατιού της βιβλιοθήκης OpenCV

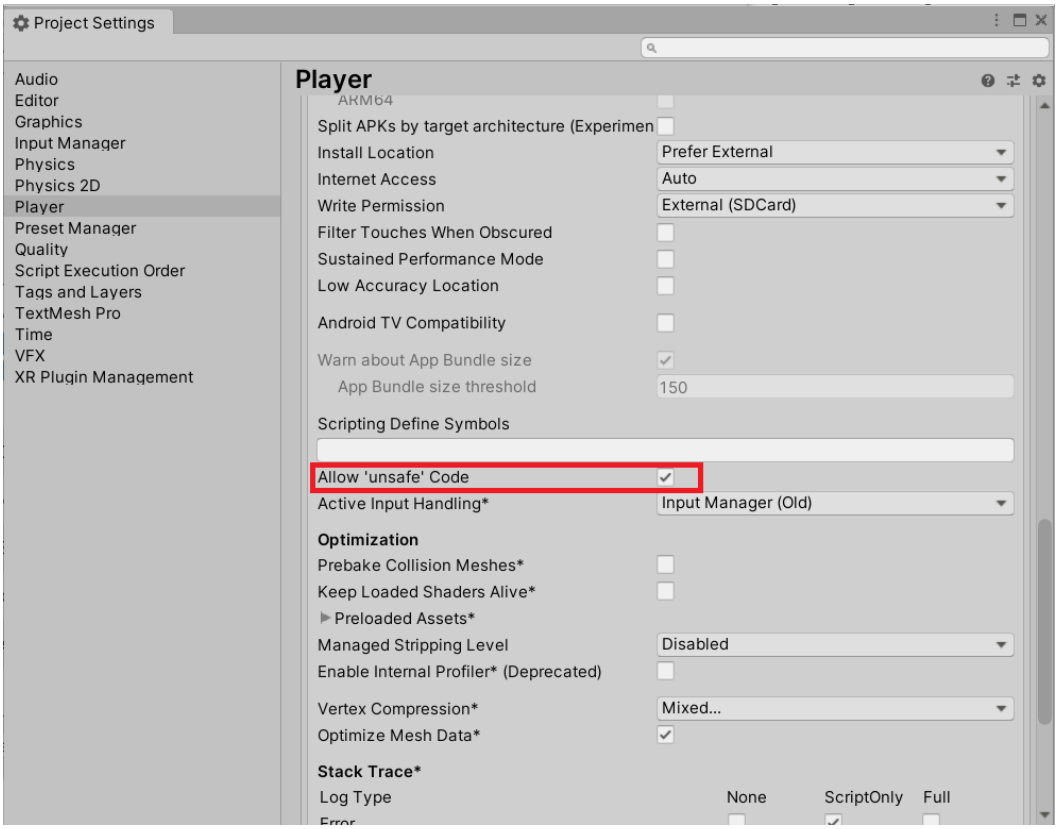
Οι συναρτήσεις που η βιβλιοθήκη φανερώνει στην κυρίως εφαρμογή δηλώνονται ως **extern "C".** Αυτό δείχνει το παρακάτω παράδειγμα μιας δοκιμαστικής συνάρτησης που υπολογίζει το τετράγωνο της μεταβλητής εισόδου:

```
extern "C" int pow2(int x) {
    return x * x;
}
```

Εικόνα 5.10: Παράδειγμα συνάρτησης βιβλιοθήκης

5.4 – Δημιουργία Unity project

Το Unity project απαιτεί λιγότερη παραμετροποίηση, συγκριτικά με της βιβλιοθήκης, για να λειτουργήσει. Αρχικά, πρέπει να επιτραπεί η χρήση “μη-ασφαλούς” (unsafe) κώδικα. Με αυτόν το όρο γίνεται αναφορά στον κώδικα που έχει απευθείας πρόσβαση στην μνήμη, μέσω δεικτών και του τελεστή διεύθυνσης ‘&’. Αυτό είναι αναγκαίο, διότι οι πίνακες στέλνονται στη βιβλιοθήκη ως δείκτες στο πρώτο στοιχείο τους. Η ρύθμιση αυτή βρίσκεται στο μενού File → Build Settings → Player Settings και στη συνέχεια στην καρτέλα Player.



Εικόνα 5.11: Ρύθμιση για μη-ασφαλή κώδικα

Στη συνέχεια, θα ενημερωθεί η Unity ότι στο τελικό εκτελέσιμο πρέπει να συμπεριλάβει τις απαραίτητες βιβλιοθήκες. Αυτό γίνεται δημιουργώντας ένα φάκελο με το όνομα “Plugins” μέσα στον κατάλογο του project, και μέσα σε αυτόν ένα φάκελο

με όνομα “Android”. Εντός του τελευταίου τοποθετούνται οι βιβλιοθήκες που χρειάζεται ένα project. Στη προκειμένη περίπτωση, απαραίτητα είναι το αρχείο που παράγεται από το project της βιβλιοθήκης καθώς και το binary της OpenCV, με μονοπάτι το “sdk\native\libs\armeabi-v7a\libopencv_java3.so” εντός φακέλου της.

Εικόνα 5.12: Τοποθέτηση των βιβλιοθηκών

5.5 – Συνεργασία Unity με βιβλιοθήκη

Για να μπορέσει η Unity να καλέσει συναρτήσεις που είναι δηλωμένες μέσα στη βιβλιοθήκη, πρέπει να χρησιμοποιηθεί η οδηγία **[DllImport(“lib_name”)]** πριν την δήλωση του πρωτότυπου της. Επιπλέον, στο πρωτότυπο πρέπει να προστεθούν οι λέξεις-κλειδιά “unsafe” και “extern”. Αυτό φαίνεται στην παρακάτω εικόνα:

```
[DllImport("DetectionLibrary")]
public unsafe static extern void addImage(Color32* queryImage, int width, int height, int page);
```

Εικόνα 5.13: Δήλωση συνάρτησης βιβλιοθήκης

Όπως δείχνει η εικόνα, στο όνομα της βιβλιοθήκης παραλείπεται το πρόθεμα “lib” καθώς και η κατάληξη του αρχείου. Συνεπώς αντί για “libDetectionLibrary.so” γράφτηκε “DetectionLibrary”. Οι απλοί τύποι δεδομένων της C# σχετίζονται ένας-προς-ένα με τους αντίστοιχους στη C++, άρα δεν απαιτείται εξειδικευμένη μέθοδος μετατροπής. Το ίδιο ισχύει και για τον τύπο Color32*, διότι είναι δείκτης, οπότε στην ουσία είναι ένας απλός ακέραιος. Η κλήση της συνάρτησης πρέπει να γίνει σε ένα unsafe block με χρήση της οδηγίας fixed όπως φαίνεται εδώ:

```
// Send the pixel data to the library to be scanned.
unsafe {
    fixed (Color32* p1 = inPixels) {
        addImage(p1, image.width, image.height, page);
    }
}
```

Εικόνα 5.14: Κλήση συνάρτησης βιβλιοθήκης

Στο παραπάνω τμήμα κώδικα, η δομή Color32 περιέχει πληροφορίες για ένα pixel της εικόνας, και η μεταβλητή inPixels είναι ένας πίνακας με στιγμιότυπα αυτής της δομής. Σκοπός είναι όλος ο πίνακας να περάσει στη βιβλιοθήκη, ώστε γίνει επεξεργασία της εικόνας. Για να επιτευχθεί αυτό, δημιουργείται ένας δείκτης προς τον πίνακα μέσω της οδηγίας fixed. Η τελευταία υποχρεώνει το περιβάλλον εκτέλεσης της C# να κρατήσει τον πίνακα σε σταθερή θέση μνήμης, καθώς, υπό κανονικές συνθήκες, θα μπορούσε να τον μετακινήσει με αποτέλεσμα ένας απλός δείκτης να έδειχνε σε σκουπίδια.

Όταν η βιβλιοθήκη λαμβάνει εικόνες σε μορφή Color32 διανυσμάτων, τα μετατρέπει σε αντικείμενα Mat. Η κλάση Mat είναι η βασική δομή εικόνας ή πίνακα στην OpenCV. Η μετατροπή είναι εξαιρετικά εύκολη, καθώς η Mat αποθηκεύει στη μνήμη τα δεδομένα με παρόμοιο τρόπο με την Unity, οπότε μια αντιγραφή επαρκεί.

```
extern "C" void addImage(void* trainImageInput, int width, int height, int page) {

    // Create a new mat object to store the input data.
    Mat newImage(height, width, CV_8UC4);

    // Copy the data to the mat.
    memcpy(newImage.data, trainImageInput, width * height * 4);
```

Εικόνα 5.15: Αντιγραφή δεδομένων εικόνας σε αντικείμενο Mat

Αρχικά δημιουργείται ένα νέα αντικείμενο Mat τεσσάρων καναλιών. Τα 4 κανάλια αναφέρονται στο γεγονός ότι μπορεί να αποθηκεύσει 4 στοιχεία (RGBA) σε κάθε κελί

του πίνακα. Στη συνέχεια, αντιγράφονται στη μνήμη που δεσμεύτηκε για το Mat τα δεδομένα της εικόνας με κλήση της memcpy.

5.6 – Δομή βιβλιοθήκης

5.6.1 – Γενικά στοιχεία

Η βιβλιοθήκη αποτελείται από ένα cpp αρχείο το οποίο περιέχει 4 δημόσιες συναρτήσεις:

- void addImage(void* data, int width, int height, int page)
- void initScan()
- void processImage(void* image, int width, int height, int& foundPage, int& centerX, int& centerY, double* rotationMat)
- void removeImages()

Η **addImage** επεξεργάζεται την προς-ανίχνευση εικόνα που λαμβάνει (μετατροπή σε grayscale, σμίκρυνση, θόλωση), και την προσθέτει σε διάνυσμα (vector) μαζί με την σελίδα στην οποία βρίσκεται.

Η **initScan** καλείται αφού έχουν προστεθεί όλες οι εικόνες των σελίδων. Χρησιμοποιεί τον αλγόριθμο BRISK, για να υπολογίσει για κάθε εικόνα τα keypoints και τους descriptors. Μετά τον υπολογισμό, αποθηκεύει όλα όσα υπολόγισε σε διανύσματα.

Η **removeImages** αδειάζει όλα τα διανύσματα και επιστρέφει τη βιβλιοθήκη στη μη αρχικοποιημένη της κατάσταση.

Η **processImages** λαμβάνει τα καρέ της κάμερας και αναζητά σε αυτά τις εικόνες που δόθηκαν στην addImage. Αν βρει κάποια από αυτές, τοποθετεί την σελίδα που της αντιστοιχεί, τις συντεταγμένες στις οποίες βρέθηκε και τον πίνακα περιστροφής που υπολογίστηκε στις μεταβλητές που πέρασαν με αναφορά. Στη περίπτωση που καμία σελίδα δεν ανιχνευθεί στο καρέ επιστρέφεται ως σελίδα η τιμή -1.

Πέρα από τις εξωτερικές συναρτήσεις, η βιβλιοθήκη περιλαμβάνει και άλλες τρεις βοηθητικές, τις οποίες δεν φανερώνει έξω. Αυτές είναι οι εξής:

bool doMatching: Επιστρέφει αν η τωρινή εικόνα της κάμερας ταιριάζει με αυτή που βρίσκεται σε μια συγκεκριμένη θέση του διανύσματος με τις εικόνες εισόδου.

Mat getCameraMatrix: Επιστρέφει ένα αντικείμενο Mat με τον προσεγγιστικό πίνακα εσωτερικών παραμέτρων της κάμερας, όπως αυτός αναλύθηκε στο κεφάλαιο 4.7, βάσει των διαστάσεών της.

Mat getRot: Επιλύει το πρόβλημα PnP με τον αλγόριθμο EPnP και επιστρέφει τον πίνακα περιστροφής που προέκυψε.

5.6.2 – Μεταβλητές και σταθερές

Οι σταθερές της βιβλιοθήκης έχουν ονόματα με κεφαλαίους χαρακτήρες και χρησιμοποιούνται για ευκολότερη παραμετροποίηση διαφόρων σταδίων της ανίχνευσης.

```
// BRISK variables.
const int THRESHOLD = 40; // A bigger value makes the algorithm faster but less accurate.
const int OCTAVES = 3; // Determines the scale invariance of the algorithm.
const float PATTERN_SCALE = 1.0f;

const int TOTAL_PREDICTION_ATTEMPTS = 3; // How many attempts are made at the predicted image before others are tried.
const int REQUIRED_MATCHES = 4; // Minimum number of matches required to consider the object found.
const int MIN_KEYPOINTS_FOR_REDUCTION = 800; // How many keypoints there have to be before some are deleted to save on processing power.
const float MIN_RESPONSE_FACTOR = 0.6f; // Multiplied with the min response to form the limit above which keypoints are removed.
const int QUERY_IMAGE_REQUIRED_SIZE = 360; // Dimension to which the query image will be scaled.
const int TRAIN_IMAGE_REQUIRED_SIZE = 240; // Dimension to which the train images will be scaled.
const float MIN_DISTANCE_FACTOR = 3.0f; // Multiplied with the min distance to form the limit above which matches are removed.
const float KNN_MATCHES_SIMILARITY_FACTOR = 0.9f; // Similarity ratio between the 2 best matches of each descriptor above which the match is ignored.
const float REQUIRED_INLIERS = 0.65f; // Percentage of inliers in total matches to consider the object found.
const double RANSAC_THRESHOLD = 3.0; // Threshold used in the RANSAC algorithm.
```

Εικόνα 5.16: Σταθερές παραμετροποίησης

Πιο αξιοσημείωτες σταθερές είναι αυτές που έχουν να κάνουν με τον αλγόριθμο ανίχνευσης BRISK. Στις δοκιμές που πραγματοποιήθηκαν βγήκε το συμπέρασμα ότι η παράμετρος THRESHOLD είναι αντιστρόφως ανάλογη με την αξιοπιστία της ανίχνευσης και ανάλογη της ταχύτητας εκτέλεσης του αλγορίθμου. Η τιμή 40 προσφέρει μια καλή ισορροπία ανάμεσα σε αυτά. Από την άλλη, η παράμετρος OCTAVES αφορά με την ανεκτικότητα του αλγορίθμου σε αλλαγές κλίμακας της

σελίδας. Παρατηρήθηκε ότι η τιμή 3 καλύπτει τις ανάγκες της εφαρμογής, δεδομένου ότι το βιβλίο δεν θα απέχει πάνω από μερικά εκατοστά του μέτρου από την κάμερα.

Οι μεταβλητές που είναι ορατές σε όλη τη βιβλιοθήκης (global) ξεχωρίζουν από το πρόθεμα *m_* που έχουν στο όνομα τους (συντομογραφία για το member).

```
// Train image.
std::vector<Mat> m_trainImages;
std::vector<std::vector<KeyPoint> > m_trainImagesKeypoints;
std::vector<Mat> m_trainImagesDescriptors;
std::vector<int> m_imagePages;
int m_trainImageNum = 0; // Number of train images given.

// Query image.
std::vector<KeyPoint> m_queryImageKeypoints;
Mat m_queryImageDescriptors;

// Detection.
Ptr<DescriptorMatcher> m_matcher;
Ptr<Feature2D> m_briskDetector;
Mat m_H; // Homography matrix.
int m_predictedImage = -1; // Which image is expected to appear next. Used for optimization.
int m_predictionCount = TOTAL_PREDICTION_ATTEMPTS; // How many times only the predicted image will be searched for.

bool m_initialized = false; // Whether the library is initialized.
```

Εικόνα 5.17: Global μεταβλητές βιβλιοθήκης

Η πρώτη ομάδα μεταβλητών αποθηκεύουν τις πληροφορίες των εικόνων ανίχνευσης, ενώ η δεύτερη του καρέ της κάμερας. Η τρίτη ομάδα αποτελείται από μεταβλητές που έχουν να κάνουν με την ανίχνευση, όπως τα αντικείμενα που υλοποιούν τους αλγορίθμους και τον πίνακα ομογραφίας. Τέλος, υπάρχει μια μεταβλητή που δείχνει αν η βιβλιοθήκη έχει αρχικοποιηθεί, δηλαδή αν έχει γίνει κλήση της συνάρτησης initScan.

5.7 – Δομή Unity project

5.7.1 – Γενικά στοιχεία

Το project της εφαρμογής από πλευρά της Unity αποτελείται από 2 σκηνές και 4 κλάσεις. Οι σκηνές είναι οι:

- BookSettings: Η αρχική σκηνή που περιλαμβάνει την επιλογή αρχείου παραμετροποίησης.

- PlayMode: Η κύρια σκηνή που τρέχει όταν έχει επιλεχθεί το βιβλίο και έχει ξεκινήσει η διαδικασία εύρεσης των σελίδων του.

Οι κλάσεις είναι οι:

- NativeFunctions: Αυτή η κλάση υπάρχει για οργανωτικούς σκοπούς. Είναι στατική και περιλαμβάνει την δήλωση όλων των συναρτήσεων της βιβλιοθήκης.
- BookConfigData: Περιλαμβάνει τις συναρτήσεις φόρτωσης του αρχείου παραμετροποίησης του βιβλίου μέσω κανονικών εκφράσεων και αποθηκεύει τα στοιχεία του σε δομές δεδομένων για ευκολότερη πρόσβαση.
- BookSettingsUIController: Περιλαμβάνει κώδικα για τον χειρισμό του περιβάλλοντος χρήστη της σκηνής BookSettings.
- ProcessImages: Περιλαμβάνει κώδικα για το χειρισμό του περιβάλλοντος χρήστη της σκηνής PlayMode. Επιπλέον, σε αυτή τη κλάση εκτελείται η κύρια λειτουργία της εφαρμογής όπως η λήψη εικόνων από την κάμερα, η κλήση των συναρτήσεων βιβλιοθήκης και η αναπαραγωγή του ψηφιακού περιεχομένου του εκάστοτε βιβλίου.

5.7.2 – Φόρτωση αρχείων ψηφιακού περιεχομένου

Για να μπορέσει η εφαρμογή να προσαρμόζεται σε διαφορετικά βιβλία πρέπει όλα τα αρχεία του ψηφιακού περιεχομένου (εικόνες, βίντεο, ήχος) να φορτώνονται δυναμικά κατά την ώρα εκτέλεσης. Η Unity παρέχει πλήρης υποστήριξη για φόρτωση αρχείων εικόνων και βίντεο κατά την ώρα εκτέλεσης για πολλές μορφές συμπίεσης. Το ίδιο δεν ισχύει για τα αρχεία ήχου, τα οποία μπορούν να φορτώνονται συμπιεσμένα πριν την μεταγλώττιση, αλλά όχι κατά την εκτέλεση. Συνεπώς θα πρέπει να γραφεί κώδικας φόρτωσης των μη-συμπιεσμένων δειγμάτων ήχου. Τα τελευταία πρέπει να έχουν την ένταση τους κωδικοποιημένη στο εύρος τιμών [-1.0, 1.0], καθώς μόνο αυτή η μορφή υποστηρίζεται από την Unity. Με βάση αυτόν τον περιορισμό, και δεδομένου ότι η αποσυμπίεση αρχείων ήχου είναι εκτός του θέματος

της παρούσας διπλωματικής, η εφαρμογή θα μπορεί να αναπαράγει ήχο μόνο από ασυμπίεστα αρχεία ήχου κατάληξης wav.

Παρακάτω φαίνεται ο κώδικας που φορτώνει το αρχείο ήχου:

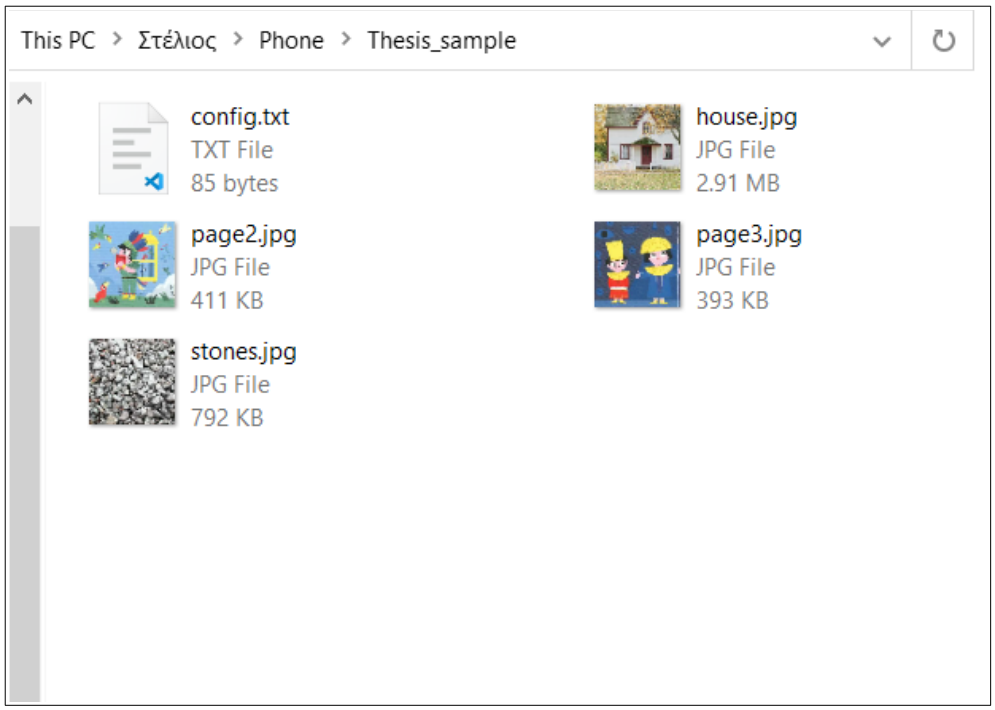
```
float[] samples;
int fileSize;
short channels;
int sampleRate;

using (FileStream fs = new FileStream(file, FileMode.Open, FileAccess.Read)) {
    BinaryReader reader = new BinaryReader(fs);

    // Read file size.
    fs.Seek(4, SeekOrigin.Begin);
    fileSize = reader.ReadInt32();
    samples = new float[(fileSize - 36) / 4]; // File size also accounts for remaining header, so it is subtracted by 36.

    // Read channels and sample rate.
    fs.Seek(22, SeekOrigin.Begin);
    channels = reader.ReadInt16();
    sampleRate = reader.ReadInt32();

    // Read samples.
    fs.Seek(44, SeekOrigin.Begin);
    for (int i = 0; i < samples.Length; i++)
        samples[i] = reader.ReadSingle();
}

clip = AudioClip.Create(name, samples.Length, channels, sampleRate, false);
clip.SetData(samples, 0);
```

Εικόνα 5.18: Φόρτωση αρχείου ήχου.

Αρχικά διαβάζονται τα δεδομένα του αρχείου (μέγεθος, συχνότητα και αριθμός καναλιών) και στη συνέχεια αυτά χρησιμοποιούνται για την κατασκευή του πίνακα samples με τα δείγματα και αργότερα για την δημιουργία του αντικειμένου της κλάσης AudioClip που χρησιμοποιεί η Unity για την αναπαραγωγή ήχου.

ΚΕΦΑΛΑΙΟ 6 – ΔΟΚΙΜΕΣ

Στα πλαίσια των δοκιμών χρησιμοποιήθηκαν εικόνες από το βιβλίο “Ο μαγικός αυλός” των εκδόσεων Πατάκη με ISBN 978-960-16-7715-6. Παρακάτω φαίνονται η δομή του φακέλου εισόδου, τα περιεχόμενα του config.txt και εικόνες δύο σελίδων του βιβλίου.



Εικόνα 6.1: Περιεχόμενα φακέλου εισόδου

```
Book title: The magic flute

page 2:
    image:stones.jpg
page 3:
    image:house.jpg
```

Εικόνα 6.2: Περιεχόμενα του config.txt

Εικόνα 6.3: Σελίδα 2 του δοκιμαστικού βιβλίου

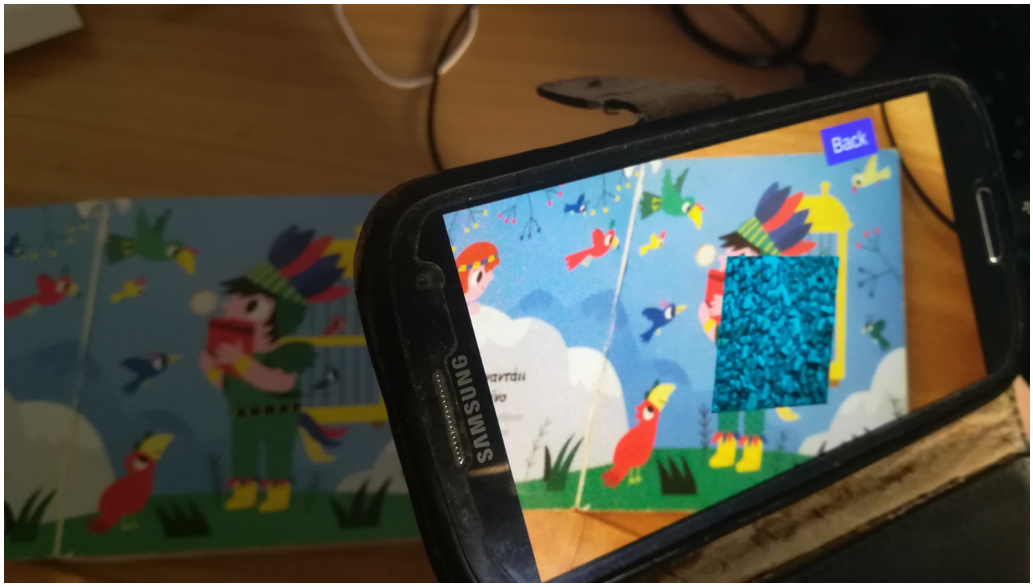
Εικόνα 6.4: Σελίδα 3 του δοκιμαστικού βιβλίου

Σύμφωνα με τα δεδομένα εισόδου, η εφαρμογή θα εμπλουτίσει αυτές τις σελίδες του βιβλίου με τις παρακάτω δύο εικόνες:

Εικόνα 6.5: Ψηφιακό περιεχόμενο για την σελίδα 2

Εικόνα 6.6: Ψηφιακό περιεχόμενο για την σελίδα 3

Χρησιμοποιώντας τα συγκεκριμένα δεδομένα εισόδου, λήφθηκαν οι παρακάτω φωτογραφίες κατά την εκτέλεση της εφαρμογής.



Εικόνα 6.7: Ανίχνευση της σελίδας 2 του δοκιμαστικού βιβλίου

Εικόνα 6.8: Ανίχνευση της σελίδας 3 του δοκιμαστικού βιβλίου

Παρατηρείται ότι μόλις ανιχνευτεί κάποια από τις σελίδες του βιβλίου, η εφαρμογή εμφανίζει στο κέντρο αυτής της σελίδας την εικόνα που είχε δηλωθεί ως ψηφιακό στοιχείο στο config.txt. Επιπλέον, η εικόνα αυτή περιστρέφεται με τρόπο που δείχνει να εφάπτεται στη σελίδα του βιβλίου.

Συμπεράσματα

Η επαυξημένη πραγματικότητα είναι μια τεχνολογία με πολλές προσδοκίες για μελλοντικές εφαρμογές. Πλέον, μια συσκευή που να μπορεί να την υποστηρίξει αγοράζεται σε προσιτή τιμή. Αυτό παρατηρήθηκε στην παρούσα διπλωματική, που επιτεύχθηκε μια αποδεκτή ταχύτητα εκτέλεσης παρόλο που δεν χρησιμοποιήθηκε κάποιο επαγγελματικό πακέτο. Αυτό το γεγονός φανερώνει τι μπορεί να επιτευχθεί όταν πίσω από μια εφαρμογή υπάρχει μεγάλη ομάδα εξειδικευμένων μηχανικών. Επιπλέον, στις δοκιμές φάνηκε ότι η επαυξημένη πραγματικότητα είναι πιο προσβάσιμη από άλλες μορφές τεχνολογίας, γεγονός που οδηγεί σε ευρεία αποδοχή της από άτομα που δεν είναι εξοικειωμένα με την τεχνολογία.

Μελλοντική επέκταση

Το σύστημα που κατασκευάστηκε στα πλαίσια αυτής της διπλωματικής έχει πολλά περιθώρια για επέκταση. Από άποψη δυνατοτήτων, θα μπορούσε να υποστηρίζει την εμφάνιση τρισδιάστατων μοντέλων ως ψηφιακό περιεχόμενο. Ένα άλλο σημείο βελτίωσης είναι η ταχύτητα των αλγορίθμων ανίχνευσης, καθώς τη δεδομένη στιγμή δε είναι πρακτική η χρήση της εφαρμογής σε ένα βιβλίο με πολλές δυναμικές σελίδες. Επιπλέον, η παραμετροποίησή της εφαρμογής θα μπορούσε να γίνεται με πιο φιλικό προς τον χρήστη τρόπο, ώστε να μην πρέπει να δημιουργεί ο ίδιος χειροκίνητα τα αρχείο παραμετροποίησης, αλλά να γίνεται από βοηθητικό πρόγραμμα.

Παράρτημα – Κώδικας

Κώδικας βιβλιοθήκης:

```
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/calib3d.hpp>

#define EXTERNAL_FUNC extern "C" void

using namespace cv;

// BRISK variables.
const int THRESHOLD = 40; // A bigger value makes the algorithm faster but less
accurate.
const int OCTAVES = 3; // Determines the scale invariance of the algorithm.
const float PATTERN_SCALE = 1.0f;

const int TOTAL_PREDICTION_ATTEMPTS = 3; // How many attempts are made at the
predicted image before others are tried.
const int REQUIRED_MATCHES = 4; // Minimum number of matches required to consider
the object found.
const int MIN_KEYPOINTS_FOR_REDUCTION = 800; // How many keypoints there have to
be before some are deleted to save on processing power.
const float MIN_RESPONSE_FACTOR = 0.6f; // Multiplied with the min response to
form the limit above which keypoints are removed.
const int QUERY_IMAGE_REQUIRED_SIZE = 360; // Dimension to which the query images
will be scaled.
const int TRAIN_IMAGE_REQUIRED_SIZE = 240; // Dimension to which the train images
will be scaled.
const float MIN_DISTANCE_FACTOR = 3.0f; // Multiplied with the min distance to
form the limit above which matches are removed.
const float KNN_MATCHES_SIMILARITY_FACTOR = 0.9f; // Similarity ratio between the
2 best matches of each descriptor above which the match is ignored.
const float REQUIRED_INLIERS = 0.65f; // Percentage of inliers in total matches to
consider the object found.
const double RANSAC_THRESHOLD = 3.0; // Threshold used in the RANSAC algorithm.

// Train image.
std::vector<Mat> m_trainImages;
std::vector<std::vector<KeyPoint> > m_trainImagesKeypoints;
std::vector<Mat> m_trainImagesDescriptors;
std::vector<int> m_imagePages;
int m_trainImageNum = 0; // Number of train images given.

// Query image.
std::vector<KeyPoint> m_queryImageKeypoints;
Mat m_queryImageDescriptors;
```

```
// Detection.
Ptr<DescriptorMatcher> m_matcher;
Ptr<Feature2D> m_briskDetector;
Mat m_H; // Homography matrix.
int m_predictedImage = -1; // Which image is expected to appear next. Used for
optimization.
int m_predictionCount = TOTAL_PREDICTION_ATTEMPTS; // How many times only the
predicted image will be searched for.

bool m_initialized = false; // Whether the library is initialized.

// Function prototypes.
bool doMatching(int imageIndex);
Mat getCameraMatrix(int width, int height);
Mat getRot(const std::vector<Point2f>& imagePoints, const std::vector<Point2f>&
objectPoints, int width, int height);

EXTERNAL_FUNC addImage(void* trainImageInput, int width, int height, int page) {

	// Create a new mat object to store the input data.
	Mat newImage(height, width, CV_8UC4);

	// Copy the data to the mat.
	memcpy(newImage.data, trainImageInput, width * height * 4);

	// Turn the mat to grayscale.
	cvtColor(newImage, newImage, CV_RGBA2GRAY);

	// Resize the input image.
	int smallSide = newImage.cols > newImage.rows ? newImage.rows :
newImage.cols;
	float scaleFactor = (float)TRAIN_IMAGE_REQUIRED_SIZE / smallSide;
	resize(newImage, newImage, Size((int)(width * scaleFactor), (int)(height *
scaleFactor)));

	GaussianBlur(newImage, newImage, Size(3, 3), 0);

	// Save new image data.
	m_trainImages.push_back(newImage);
	m_imagePages.push_back(page);
	m_trainImageNum++;
}

EXTERNAL_FUNC initScan() {
	m_matcher = DescriptorMatcher::create("BruteForce-Hamming");
	m_briskDetector = BRISK::create(THRESHOLD, OCTAVES, PATTERN_SCALE);

	for (int i = 0; i < m_trainImageNum; i++) {

		std::vector<KeyPoint> currentImageKeypoints;
```

```
			m_briskDetector->detect(m_trainImages[i], currentImageKeypoints);

#pragma region KeypointReduction
			if (currentImageKeypoints.size() > MIN_KEYPOINTS_FOR_REDUCTION) {
				int num = currentImageKeypoints.size();
				float max = currentImageKeypoints[0].response;

				// Find the max keypoint response.
				for (int j = 1; j < num; j++) {
					if (max < currentImageKeypoints[j].response)
						max = currentImageKeypoints[j].response;
				}

				// Use the max value to filter out keypoints.
				std::vector<KeyPoint> temp;
				for (int j = 0; j < num; j++) {
					int response = currentImageKeypoints[j].response;
					if (max * MIN_RESPONSE_FACTOR < response)
						temp.push_back(currentImageKeypoints[j]);
				}
				currentImageKeypoints = std::move(temp);
			}
#pragma endregion

			Mat tempDescriptors;
			m_briskDetector->compute(m_trainImages[i], currentImageKeypoints,
tempDescriptors);

			m_trainImagesKeypoints.push_back(std::move(currentImageKeypoints));
			m_trainImagesDescriptors.push_back(std::move(tempDescriptors));
		}
		m_initialized = true;
}

EXTERNAL_FUNC removeImages() {
		m_initialized = false;
		m_imagePages.clear();
		m_trainImageNum = 0;

		if (m_matcher != NULL) {
			m_matcher.release();
		}

		if (m_briskDetector)
			m_briskDetector.release();

		m_trainImages.clear();
		m_trainImagesKeypoints.clear();
		m_trainImagesDescriptors.clear();
```

```
	m_queryImageKeypoints.clear();
}

EXTERNAL_FUNC processImage(void* queryImage, int width, int height, int&
foundPage, int& centerX, int& centerY, double* rotData) {
	foundPage = -1;

	if (!m_initialized)
		return;

	Mat webcamFrame = Mat(height, width, CV_8UC4);
	webcamFrame.data = (uchar*)queryImage;

	// Grayscale the input.
	Mat webcamFrameGray(webcamFrame.rows, webcamFrame.cols, CV_8UC1);
	cvtColor(webcamFrame, webcamFrameGray, CV_RGBA2GRAY);

	// Resize.
	int smallSide = webcamFrameGray.cols > webcamFrameGray.rows ?
webcamFrameGray.rows : webcamFrameGray.cols;
	float scaleFactor = (float)QUERY_IMAGE_REQUIRED_SIZE / smallSide;
	resize(webcamFrameGray, webcamFrameGray, Size((int)(width * scaleFactor),
(int)(height * scaleFactor)));

	// Do the detection.
	m_briskDetector->detectAndCompute(webcamFrameGray, Mat(),
m_queryImageKeypoints, m_queryImageDescriptors);

	if (m_queryImageKeypoints.size() == 0 || m_queryImageDescriptors.cols == 0)
{
		return;
	}

	// Look for the predicted image first if there is one.
	int foundImage = -1;
	if (m_predictedImage != -1) {
		bool found = doMatching(m_predictedImage);
		if (found) {
			foundImage = m_predictedImage;
			m_predictionCount = TOTAL_PREDICTION_ATTEMPTS;
		}
		else {
			m_predictionCount--;
			if (m_predictionCount == 0) {
				m_predictedImage = -1;
			}
			return;
		}
	}

	// If the predicted image was not found, look for the other images.
```

```
	if (foundImage == -1) {
		for (int i = 0; i < m_trainImageNum; i++) {
			if (i == m_predictedImage)
				continue;
			bool found = doMatching(i);
			if (found) {
				foundImage = i;
				m_predictedImage = i;
				m_predictionCount = TOTAL_PREDICTION_ATTEMPTS;
				break;
			}
		}
	}

	if (foundImage != -1) {

		std::vector<Point2f> queryImageCorners(4);
		std::vector<Point2f> trainImageCorners(4);

		trainImageCorners[0] = Point2f(0, 0);
		trainImageCorners[1] =
Point2f((float)m_trainImages[foundImage].cols, 0);
		trainImageCorners[2] =
Point2f((float)m_trainImages[foundImage].cols,
(float)m_trainImages[foundImage].rows);
		trainImageCorners[3] = Point2f(0,
(float)m_trainImages[foundImage].rows);

		perspectiveTransform(trainImageCorners, queryImageCorners, m_H);

		Mat rotationMatrix = getRot(queryImageCorners, trainImageCorners,
webcamFrameGray.cols, webcamFrameGray.rows);

		// Copy the rotation data to the return pointer.
		memcpy(rotData, rotationMatrix.data, 9 * 8);

		// Scale the points to fit the image and calculate center of mass
for the query image corners.
		float xScale = (float)width / webcamFrameGray.cols;
		float yScale = (float)height / webcamFrameGray.rows;
		int sumX = 0, sumY = 0;
		for (int i = 0; i < 4; i++) {
			queryImageCorners[i].x *= xScale;
			queryImageCorners[i].y *= yScale;

			sumX += queryImageCorners[i].x;
			sumY += queryImageCorners[i].y;
		}
		centerX = sumX / 4;
		centerY = sumY / 4;
```

```
			foundPage = m_imagePages[foundImage];
		}
}

// Generate the default camera matrix.
Mat getCameraMatrix(int width, int height) {

		Mat mat = Mat(3, 3, CV_64F);
		mat.at<double>(0, 0) = width;
		mat.at<double>(0, 1) = 0;
		mat.at<double>(0, 2) = width / 2.0;
		mat.at<double>(1, 0) = 0;
		mat.at<double>(1, 1) = width;
		mat.at<double>(1, 2) = height / 2.0;
		mat.at<double>(2, 0) = 0;
		mat.at<double>(2, 1) = 0;
		mat.at<double>(2, 2) = 1;
		return mat;
}

Mat getRot(const std::vector<Point2f>& imagePoints, const std::vector<Point2f>&
objectPoints, int width, int height) {

		std::vector<Point3f> obj;
		for (int i = 0; i < objectPoints.size(); i++)
			obj.push_back(Point3f(objectPoints[i]));

		Mat R(3, 1, CV_64F);
		Mat T(3, 1, CV_64F);

		Mat distCoef = Mat::zeros(4, 1, CV_64F);
		solvePnP(obj, imagePoints, getCameraMatrix(width, height), distCoef, R, T,
false, SOLVEPNP_EPNP);

		// Use the Rodrigues function to convert the rotation vector to the
rotation matrix.
		Mat rotMatrix(3, 3, CV_64F);
		Rodrigues(R, rotMatrix);

		return rotMatrix;
}

bool doMatching(int imageIndex) {
		std::vector<std::vector<DMatch> > tempMatches;

		m_matcher->knnMatch(m_trainImagesDescriptors[imageIndex],
m_queryImageDescriptors, tempMatches, 2);

		std::vector<DMatch> goodMatches;

		// --- Filter matches ---
```

```
	// Filter 1
	for (size_t j = 0; j < tempMatches.size(); j++) {

		// If there was only one match, then add it to the vector.
		if (tempMatches[j].size() == 1) {
			goodMatches.push_back(tempMatches[j][0]);
			continue;
		}

		// Check the similarity between the two best matches and only add if
it isn't greater than a threshold.
		float distance1 = tempMatches[j][0].distance;
		float distance2 = tempMatches[j][1].distance;

		if (distance2 * KNN_MATCHES_SIMILARITY_FACTOR > distance1) {
			goodMatches.push_back(tempMatches[j][0]);
		}
	}
	if (goodMatches.size() == 0)
		return false;

	// Filter 2
	std::sort(goodMatches.begin(), goodMatches.end());
	float minDistance = goodMatches[0].distance * MIN_DISTANCE_FACTOR;
	for (size_t j = 0; j < goodMatches.size(); j++) {
		if (goodMatches[j].distance > minDistance) {
			goodMatches.erase(goodMatches.begin() + j, goodMatches.end());
			break;
		}
	}
	// --- Filters end ---

	if (goodMatches.size() < REQUIRED_MATCHES) {
		return false;
	}

	// Calculate homography.
	std::vector<Point2f> points1, points2;
	for (size_t j = 0; j < goodMatches.size(); j++) {
		points1.push_back(m_trainImagesKeypoints[imageIndex]
[goodMatches[j].queryIdx].pt);

	points2.push_back(m_queryImageKeypoints[goodMatches[j].trainIdx].pt);
	}

	Mat mask;
	Mat Htemp = findHomography(points1, points2, RANSAC, RANSAC_THRESHOLD,
mask);

	if (!Htemp.empty()) {
```

```
                unsigned int inliers = 0;
                for (int r = 0; r < mask.rows; r++) {
                        if ((unsigned int)mask.at<uchar>(r, 0))
                                inliers++;
                }

                float inlierPercentage = (float)inliers / mask.rows;

                if (inlierPercentage >= REQUIRED_INLIERS) {
                        // Image found.
                        m_H = std::move(Htemp);
                        return true;
                }
                return false;
        }
        else {
                return false;
        }
}
```

Κώδικας Unity project:

NativeFunctions.cs:

```
using System.Collections;
using System.Collections.Generic;
using System.Runtime.InteropServices;
using UnityEngine;

public static class NativeFunctions {

    private const string libraryName = "OpenCVAndroidLibrary";

        // Library methods.
        [DllImport(libraryName)]
        public unsafe static extern void initScan();

        [DllImport(libraryName)]
        public unsafe static extern void removeImages();

        [DllImport(libraryName)]
        public unsafe static extern void addImage(void* queryImage, int width, int
height, int page);

        [DllImport(libraryName)]
        public unsafe static extern void processImage(void* trainImage, int width,
int height, ref int detectedPage, ref int foundPageX, ref int foundPageY, double*
outRotMatrix);
}
```

BookConfigData.cs:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text.RegularExpressions;
using UnityEngine;

public class BookConfigData {

	// Singleton instance.
	private static BookConfigData m_instance = null;

	private string m_bookTitle;
	private int m_pages;
	private int m_imageElements, m_soundElements, m_videoElements;

	private string[] m_ARImagesPaths;
	private string[] m_pagesPaths;
	private string[] m_audioPaths;
	private string[] m_videoPaths;

	private Dictionary<int, int> m_pageToImagePathIndex;
	private Dictionary<int, int> m_pageToAudioPathIndex;
	private Dictionary<int, int> m_pageToVideoPathIndex;

	public static BookConfigData Instance {
		get {
			if (m_instance == null)
				m_instance = new BookConfigData();
			return m_instance;
		}
	}

	public string[] PagePaths {
		get {
			string[] tmp = new string[m_pagesPaths.Length];
			m_pagesPaths.CopyTo(tmp, 0);
			return tmp;
		}
	}

	public string BookTitle => m_bookTitle;

	public int Pages => m_pages;

	public int ImageElements => m_imageElements;

	public int SoundElements => m_soundElements;
```

```
        public int VideoElements => m_videoElements;

        private BookConfigData() {

                m_bookTitle = "";
                m_pages = 0;
                m_imageElements = m_soundElements = m_videoElements = 0;

                m_ARImagesPaths = new string[0];
                m_pagesPaths = new string[0];
                m_audioPaths = new string[0];
                m_videoPaths = new string[0];

                m_pageToImagePathIndex = new Dictionary<int, int>();
                m_pageToAudioPathIndex = new Dictionary<int, int>();
                m_pageToVideoPathIndex = new Dictionary<int, int>();
        }

        public bool ParseDirectory(string rootPath) {

                // Reset values.
                Clear();

                // Check if a valid directory was given.
                if (!Directory.Exists(rootPath)) {
                        return false;
                }

                // Create temporary variables to hold data.
                List<string> arImagePaths = new List<string>();
                List<string> pagePaths = new List<string>();
                List<string> audioPaths = new List<string>();
                List<string> videoPaths = new List<string>();

                // Get the files in root folder.
                string[] files = Directory.GetFiles(rootPath);

                foreach (string path in files) {

                        // Filename without path.
                        string filename = path.Substring(path.LastIndexOf('/') + 1);

                        // Page file.
                        if (filename.StartsWith("page")) {
                                pagePaths.Add(path);
                        }

                        // Config file.
                        else if (filename == "config.txt") {

                                // Read the file.
```

```
                        string configFileContents = "";
                        using (StreamReader sr = new StreamReader(path)) {
                                configFileContents = sr.ReadToEnd();
                        }

                        // Get the title of the book.
                        string titleLinePattern = @"title:\s*[\w\s]+\n";
                        Match titleLineMatch = Regex.Match(configFileContents,
titleLinePattern);
                        if (titleLineMatch.Success) {
                                string line = titleLineMatch.Value;
                                int index = line.IndexOf(':');
                                m_bookTitle = line.Substring(index + 1,
line.Length - index - 1).Trim();
                        }
                        else {
                                return false;
                        }

                        // Regex for parsing the file.
                        string linePattern = @"page\s\d:([\r\n]*\s*\w+:\s*[\w/\
s]+\.\w+)+"; // Matches the entire entry for a page.
                        string mediaPattern = @"(image:|sound:|video:)[\w/\s]
+\.\w+"; // Matches a media of the page.
                        string pathPattern = @"[\w/\s]+\.\w+"; // The path of a
media of the page.

                        Match lineMatch = Regex.Match(configFileContents,
linePattern);
                        while (lineMatch.Success) {
                                m_pages++;
                                string page = lineMatch.Value;

                                // Find the page number.
                                int pageDigits = 0;
                                while(true) {
                                        char c = page[5 + pageDigits];
                                        if (char.IsDigit(c))
                                                pageDigits++;
                                        else
                                                break;
                                }
                                int currentPage =
Convert.ToInt32(page.Substring(5, pageDigits));

                                Match matchMedia = Regex.Match(page,
mediaPattern);
                                while (matchMedia.Success) {
                                        string media = matchMedia.Value;
                                        Match matchPath = Regex.Match(media,
pathPattern);
```

```
                                        if (matchPath.Success) {
                                              string mediaPath = rootPath + '/' +
matchPath.Value;

                                              if (File.Exists(mediaPath)) {
                                                    if (media.Contains("image:"))
{
                                                          m_imageElements++;

      arImagePaths.Add(mediaPath);

      m_pageToImagePathIndex[currentPage] = arImagePaths.Count - 1;

                                                    }
                                                    else if
(media.Contains("sound:")) {
                                                          m_soundElements++;

      audioPaths.Add(mediaPath);

      m_pageToAudioPathIndex[currentPage] = audioPaths.Count - 1;
                                                    }
                                                    else if
(media.Contains("video:")) {
                                                          m_videoElements++;

      videoPaths.Add(mediaPath);

      m_pageToVideoPathIndex[currentPage] = videoPaths.Count - 1;
                                                    }
                                              }
                                        }
                                        matchMedia = matchMedia.NextMatch();
                                  }
                                  lineMatch = lineMatch.NextMatch();
                            }
                      }
            }

            // Save the data to member variables.
            m_ARImagesPaths = arImagePaths.ToArray();
            m_pagesPaths = pagePaths.ToArray();
            m_audioPaths = audioPaths.ToArray();
            m_videoPaths = videoPaths.ToArray();

            return true;
      }

      public void Clear() {
            // Reset values.
            m_pagesPaths = new string[0];
```

```
            m_ARImagesPaths = new string[0];
            m_audioPaths = new string[0];
            m_videoPaths = new string[0];

            m_pageToImagePathIndex.Clear();
            m_pageToAudioPathIndex.Clear();
            m_pageToVideoPathIndex.Clear();

            m_bookTitle = "";
            m_pages = 0;
            m_imageElements = m_soundElements = m_videoElements = 0;
      }

      public string GetARimagePath(int page) {

            // If this page does not have an image to show, return null.
            if (!m_pageToImagePathIndex.ContainsKey(page))
                  return null;

            // Look at the dictionary for the correct index in the array.
            int index = m_pageToImagePathIndex[page];
            return m_ARImagesPaths[index];
      }

      public string GetAudioPath(int page) {

            // If this page does not have an audio to play, return null.
            if (!m_pageToAudioPathIndex.ContainsKey(page))
                  return null;

            // Look at the dictionary for the correct index in the array.
            int index = m_pageToAudioPathIndex[page];
            return m_audioPaths[index];
      }

      public string GetVideoPath(int page) {

            // If this page does not have a video to play, return null.
            if (!m_pageToVideoPathIndex.ContainsKey(page))
                  return null;

            // Look at the dictionary for the correct index in the array.
            int index = m_pageToVideoPathIndex[page];
            return m_videoPaths[index];
      }

}
```

BookSettingsUIController.cs:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class BookSettingsUIController : MonoBehaviour {

	[SerializeField] private Text m_txtConfigContent;
	[SerializeField] private InputField m_inptPath;
	[SerializeField] private Button m_btnBegin;

	private BookConfigData m_configData;

	private void Awake() {

		m_configData = BookConfigData.Instance;

		m_inptPath.text = GetAndroidExternalStoragePath();

	}

	private void Update() {
		if (Input.GetKeyDown(KeyCode.Escape)) {
			Application.Quit();
		}
	}

	private string GetAndroidExternalStoragePath() {
		string path = "";
		try {
			AndroidJavaClass jc = new
AndroidJavaClass("android.os.Environment");
			path =
jc.CallStatic<AndroidJavaObject>("getExternalStorageDirectory").Call<String>("getA
bsolutePath");
			return path;
		}
		catch (Exception e) {
			Debug.Log(e.Message);
			return path;
		}
	}

	public void SetButton() {
```

```
		string dir = m_inptPath.text;
		bool success = m_configData.ParseDirectory(dir);
		string text = "";

		if (success) {

			// Set the text based on the config file.
			text += "Title: " + m_configData.BookTitle + '\n';
			text += "Pages with content: " + m_configData.Pages + '\n';
			text += "Pages with image content: " +
m_configData.ImageElements + '\n';
			text += "Pages with audio content: " +
m_configData.SoundElements + '\n';
			text += "Pages with video content: " +
m_configData.VideoElements + '\n';

			// Enable the button.
			m_btnBegin.interactable = true;
		}
		else {
			text = "Error reading book data.";
			m_btnBegin.interactable = false;
		}
		m_txtConfigContent.text = text;
	}

	public void BeginButton() {
		SceneManager.LoadScene("PlayMode");
	}
}
```

ProcessImages.cs:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine.Video;

public class ProcessImages : MonoBehaviour {

	[SerializeField] private RawImage m_background; // Background of the
canvas.
	[SerializeField] private RectTransform m_canvas; // The canvas' transform.
	[SerializeField] private RawImage m_contentImage; // The image containing
the AR video and image elements.
```

```
	private AudioSource m_audioSource; // Play the audio.
	private WebCamTexture m_camTexture; // Texture onto which the camera frame
is drawn.
	private Color32[] m_pixels; // Array to hold the camera frame's pixel data.
	private Camera m_mainCamera;

	private VideoPlayer m_videoPlayer; // Used to play video.
	private RenderTexture m_renderTexture; // The AR video is drawn on this
texture.
	private Texture2D m_imageTexture; // The AR image is drawn on this texture.

	private BookConfigData configData;

	private bool m_readyToProcess; // True if the initialization was
successful.

	// Last audio and video file played so that they are not repeated.
	private string m_lastAudioFile;
	private string m_lastVideoFile;

	private void Start() {

		configData = BookConfigData.Instance;

		m_readyToProcess = false;
		m_lastAudioFile = m_lastVideoFile = "";

		m_mainCamera = FindObjectOfType<Camera>();
		m_videoPlayer = FindObjectOfType<VideoPlayer>();
		m_audioSource = GetComponent<AudioSource>();

		m_imageTexture = new Texture2D(1, 1);

		m_renderTexture = new RenderTexture(256, 256, 24);
		m_videoPlayer.targetTexture = m_renderTexture;

		// Make the device not go to sleep on idle.
		Screen.sleepTimeout = SleepTimeout.NeverSleep;

		// Get the camera devices.
		WebCamDevice[] devices = WebCamTexture.devices;
		if (devices.Length == 0) {
			Debug.Log("No camera found");
			return;
		}

		// Start the camera recording.
		m_camTexture = new WebCamTexture(devices[0].name, Screen.width,
Screen.height);
		m_camTexture.Play();
		m_background.texture = m_camTexture;
```

```
            // Create space for the library input and output pixels.
            int totalPixels = m_camTexture.width * m_camTexture.height;
            m_pixels = new Color32[totalPixels];

            // Set the aspect ratio.
            AspectRatioFitter fit =
m_background.GetComponent<AspectRatioFitter>();
            float ratio = (float)m_canvas.rect.width /
(float)m_canvas.rect.height;
            fit.aspectRatio = ratio;

            // Get the paths for the images to be scanned.
            string[] resourceNames = configData.PagePaths;

            for (int i = 0; i < resourceNames.Length; i++) {

                  byte[] fileData = File.ReadAllBytes(resourceNames[i]);
                  int page = GetPageNumberFromPath(resourceNames[i]);

                  // Load the image into a Texture2D object.
                  Texture2D image = new Texture2D(1, 1);
                  image.LoadImage(fileData); // This will automatically resize
the texture dimensions.

                  // Get the pixel data of the image.
                  Color32[] inPixels = image.GetPixels32();

                  // Send the pixel data to the library to be scanned.
                  unsafe {
                        fixed (Color32* p1 = inPixels) {
                              NativeFunctions.addImage(p1, image.width,
image.height, page);
                        }
                  }
            }

            // Make the library to initiate the scan for all the target images.
            NativeFunctions.initScan();

            // The library is now ready.
            m_readyToProcess = true;
      }

      void Update() {

            // Respond to the back/esc button.
            if (Input.GetKeyDown(KeyCode.Escape)) {
                  GoBack();
                  return;
            }
```

```
            // Do not continue if the library is not ready to process images or
if no camera was detected on the device.
            if (!m_readyToProcess)
                  return;

            // Do not continue if the frame hasn't changed.
            if (!m_camTexture.didUpdateThisFrame)
                  return;

            // Get pixel data from the camera.
            m_camTexture.GetPixels32(m_pixels);

            int detectedPage = -1;
            int centerx = 0;
            int centery = 0;

            double[] rotationMatrix = new double[9];

            // Process current frame.
            unsafe {
                  fixed (Color32* p1 = m_pixels) {
                        fixed (double* p2 = rotationMatrix) {
                              NativeFunctions.processImage(p1,
m_camTexture.width, m_camTexture.height, ref detectedPage, ref centerx, ref
centery, p2);
                        }
                  }
            }

            if (detectedPage != -1) {

                  // Check if there are digital elements for the detected page.
                  string image = configData.GetARimagePath(detectedPage);
                  string sound = configData.GetAudioPath(detectedPage);
                  string video = configData.GetVideoPath(detectedPage);

                  if (image != null) {

                        // Stop the video if it was playing since they share
the surface.
                        if (m_videoPlayer.isPlaying)
                              m_videoPlayer.Stop();
                        m_lastVideoFile = "";

                        // Set the texture.
                        LoadTextureFromFile(m_imageTexture, image);
                        m_contentImage.texture = m_imageTexture;

                        Vector2 centerPoint = new Vector2(centerx, centery);
                        Vector3 eulerAngles = GetEulerAngles(rotationMatrix);
```

```
                        SetObjectTransform(m_contentImage.transform,
centerPoint, eulerAngles);
                        m_contentImage.gameObject.SetActive(true);
                }
                else if (video != null) {

                        Vector2 centerPoint = new Vector2(centerx, centery);
                        Vector3 eulerAngles = GetEulerAngles(rotationMatrix);

                        if (video != m_lastVideoFile) {

                                m_videoPlayer.url = video;
                                m_contentImage.texture = m_renderTexture;
                                m_videoPlayer.Play();

                                // Remember this file so it is not played twice
in a row.
                                m_lastVideoFile = video;
                        }

                        SetObjectTransform(m_contentImage.transform,
centerPoint, eulerAngles);
                        m_contentImage.gameObject.SetActive(true);
                }

                if (sound != null) {

                        if (!m_audioSource.isPlaying && sound !=
m_lastAudioFile) {
                                AudioClip clip = LoadAudioClip(sound);
                                if (clip != null) {
                                        m_audioSource.clip = clip;
                                        m_audioSource.Play();

                                        // Remember this file so it is not played
twice in a row.
                                        m_lastAudioFile = sound;
                                }
                        }
                }
        }
        else {
                m_contentImage.gameObject.SetActive(false);
        }
    }

    public void BackButton() {
            GoBack();
    }
```

```
	private void GoBack() {
		m_readyToProcess = false;
		m_camTexture.Stop();
		NativeFunctions.removeImages();
		configData.Clear();
		SceneManager.LoadScene("BookSettings");
	}

	private Vector3 GetEulerAngles(double[] rotMatrix) {

		// Result variables.
		float thetaX = 0;
		float thetaY = 0;
		float thetaZ = 0;

		// Helper variables for the matrix.
		float r11 = (float)rotMatrix[0];
		float r21 = (float)rotMatrix[3];
		float r31 = (float)rotMatrix[6];
		float r32 = (float)rotMatrix[7];
		float r33 = (float)rotMatrix[8];

		// Calculations.
		thetaY = -Mathf.Asin(r31);
		thetaX = Mathf.Atan2(r32 / Mathf.Cos(thetaY), r33 /
Mathf.Cos(thetaY));
		thetaZ = Mathf.Atan2(r21 / Mathf.Cos(thetaY), r11 /
Mathf.Cos(thetaY));

		// Convert to degrees.
		thetaX *= Mathf.Rad2Deg;
		thetaY *= Mathf.Rad2Deg;
		thetaZ *= Mathf.Rad2Deg;

		thetaX *= 100; thetaX = Mathf.Round(thetaX); thetaX /= 100f;
		thetaY *= 100; thetaY = Mathf.Round(thetaY); thetaY /= 100f;
		thetaZ *= 100; thetaZ = Mathf.Round(thetaZ); thetaZ /= 100f;

		return new Vector3(thetaX, thetaY, thetaZ);
	}

	// Set the transform of the object based on the data received from the
library.
	private void SetObjectTransform(Transform obj, Vector2 imgPos, Vector3 rot)
{

		float canvasX = m_canvas.rect.width * (imgPos.x /
m_camTexture.width);
		float canvasY = m_canvas.rect.height * (imgPos.y /
m_camTexture.height);
		Vector3 canvasPos = new Vector3(canvasX, canvasY);
```

```
		Vector3 worldPos = m_mainCamera.ScreenToWorldPoint(canvasPos);
		worldPos.z = 0;

		obj.transform.position = worldPos;
		obj.transform.rotation = Quaternion.Euler(rot.x, rot.y, rot.z);
	}

	private int GetPageNumberFromPath(string path) {
		int numberIndex = path.LastIndexOf("page") + 4;
		int dotIndex = numberIndex;
		while (true) {
			if (char.IsDigit(path[dotIndex]))
				dotIndex++;
			else
				break;
		}
		return Convert.ToInt32(path.Substring(numberIndex, dotIndex -
numberIndex));
	}

	private bool LoadTextureFromFile(Texture2D tex, string file) {

		byte[] imageData;
		try {
			imageData = File.ReadAllBytes(file);
		}
		catch (IOException) {
			return false;
		}

		tex.LoadImage(imageData);
		return true;
	}

	private AudioClip LoadAudioClip(string file) {

		if (!File.Exists(file))
			return null;

		AudioClip clip;
		string name;
		{
			int indx1 = file.LastIndexOf('\\') + 1;
			int indx2 = file.LastIndexOf('.');
			name = file.Substring(indx1, indx2 - indx1);
		}

		float[] samples;
		int fileSize;
		short channels;
```

```
            int sampleRate;

            using (FileStream fs = new FileStream(file, FileMode.Open,
FileAccess.Read)) {
                  BinaryReader reader = new BinaryReader(fs);

                  // Read file size.
                  fs.Seek(4, SeekOrigin.Begin);
                  fileSize = reader.ReadInt32();
                  samples = new float[(fileSize - 36) / 4]; // File size also
accounts for remaining header, so it is subtracted by 36.

                  // Read channels and sample rate.
                  fs.Seek(22, SeekOrigin.Begin);
                  channels = reader.ReadInt16();
                  sampleRate = reader.ReadInt32();

                  // Read samples.
                  fs.Seek(44, SeekOrigin.Begin);
                  for (int i = 0; i < samples.Length; i++)
                        samples[i] = reader.ReadSingle();
            }

            clip = AudioClip.Create(name, samples.Length, channels, sampleRate,
false);
            clip.SetData(samples, 0);

            return clip;
      }

#if UNITY_EDITOR
      // Not necessary when built, only for editor.
      private void OnApplicationQuit() {
            NativeFunctions.removeImages();
      }
#endif
}
```

Βιβλιογραφία

1. Michael Isberto (2018). "*The history of augmented reality".*Ανακτήθηκε Σεπτέμβριο 2020 από: https://www.colocationamerica.com/blog/history-of-augmentedreality
2. Ivan Sutherland (1968). "*A head-mounted three dimensional display*". Ανακτήθηκε Σεπτέμβριο 2020 από: http://cacs.usc.edu/education/cs653/Sutherland-HeadmountedDisplay-AFIPS68.pdf
3. https://aboutmyronkrueger.weebly.com/videoplace.html Ανακτήθηκε Σεπτέμβριο 2020.
4. "*Louis Rosenberg Develops Virtual Fixtures, the First Fully Immersive Augmented Reality System*". Ανακτήθηκε Σεπτέμβριο 2020 από: https://www.historyofinformation.com/detail.php?entryid=4696
5. Wikipedia contributors. (2020, July 23). "*1st & Ten (graphics system)*". In Wikipedia, The Free Encyclopedia. Ανακτήθηκε Σεπτέμβριο 2020 από: https://en.wikipedia.org/w/index.php?title=1st_%26_Ten_(graphics_system)&oldid=969101613
6. https://www.hitl.washington.edu/artoolkit/ Ανακτήθηκε Σεπτέμβριο 2020.
7. http://imdl.naist.jp/people/hirokazukato/ Ανακτήθηκε Σεπτέμβριο 2020.
8. Nicole Lee (2013). "*Volkswagen develops augmented reality service manual for the XL1*". Ανακτήθηκε Σεπτέμβριο 2020 από: https://www.engadget.com/2013-10-01-volkswagen-augmented-reality-ipad-manual-xl1.html
9. "*AR/MR Devices*" Ανακτήθηκε Σεπτέμβριο 2020 από: https://www.augmented-minds.com/en/augmented-reality/ar-hardware-devices/
10. Διαφάνειες από το Πανεπιστήμιο της Ουάσινγκτον. Ανακτήθηκε Σεπτέμβριο 2020 από: https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect6.pdf
11. Shaharyar Khan, Zahra Saleem (2018). "*A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK*" doi: 10.1109/ICOMET.2018.8346440
12. Şahin Işık, Kemal Özkan (2014). "*A Comparative Evaluation of Well-known Feature Detectors and Descriptors*". Ανακτήθηκε Σεπτέμβριο 2020 από: https://dergipark.org.tr/en/download/article-file/89429
13. Marius Muja, Davic Lowe (2012). "*Fast Matching of Binary Features*". doi: 10.1109/CRV.2012.60

14. Δήμητρα Πάνου 2016. “*Αφαίρεση περιθωρίου και διόρθωση παραμόρφωσης σε έγγραφα από κάμερα*”. (Πτυχιακή Εργασία, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών). Ανακτήθηκε από: https://pergamos.lib.uoa.gr/uoa/dl/object/1325010
15. Adrian Penate-Sanchez, Juan Andrade-Cetto, Francesc Moreno-Noguer (2013). “*Exhaustive Linearization for Robust Camera Pose and Focal Length Estimation*”. doi: 10.1109/TPAMI.2013.36
16. Laurent Kneip, Hongdong li, Yongduek Seo (2014). “*UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability*” doi: 10.1007/978-3-319-10590-1_9
17. Satya Mallick (2016). “*Rotation Matrix To Euler Angles*”. Ανακτήθηκε Σεπτέμβριο 2020 από: https://www.learnopencv.com/rotation-matrix-to-euler-angles/
18. Satya Mallick (2016). “*Head Pose Estimation using OpenCV and Dlib*”. Ανακτήθηκε Σεπτέμβριο 2020 από: https://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/
19. https://www.mathworks.com/help/vision/ug/camera-calibration.html Ανακτήθηκε Σεπτέμβριο 2020.
20. Kaustubh Sadekar, Satya Mallick (2020). “*Camera Calibration using OpenCV*”. Ανακτήθηκε Σεπτέμβριο 2020 από: https://www.learnopencv.com/camera-calibration-using-opencv/