Ατομική Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΜΕ ΤΗΝ ΧΡΗΣΗ ΚΑΜΕΡΩΝ TOF ΚΑΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΜΕ ΣΚΟΠΟ ΤΗΝ ΠΑΡΟΧΗ ΒΟΗΘΕΙΑΣ ΣΕ ΑΤΟΜΑ ΜΕ ΠΡΟΒΛΗΜΑΤΑ ΟΡΑΣΗΣ

Ανδρέας Μυλιδώνης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ιανουάριος 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη εφαρμογής με την χρήση καμερών TOF και μηχανικής μάθησης με σκοπό την παροχή βοήθειας σε άτομα με προβλήματα όρασης

Ανδρέας Μυλιδώνης

Επιβλέπων Καθηγητής

Δρ. Χρυσάνθου Γιώργος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Ιανουάριος 2021

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου Δρ. Γιώργο Χρυσάνθου, που με επέλεξε για να ασχοληθώ με το συγκεκριμένο θέμα και για τις κατευθυντήριες γραμμές που μου δόθηκαν για την συγγραφή της πτυχιακής μου εργασίας. Επίσης, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στους Δρ. Μελίνο Αβερκίου και κ. Μάριο Λοΐζου για την συνεχή βοήθεια που μου παρείχαν, τόσο για την υλοποίηση της εφαρμογής, όσο και για τη συγγραφή της παρούσας εργασίας. Επιπλέον, θα ήθελα να δώσω ευχαριστίες στο ερευνητικό κέντρο RISE, που με εμπιστεύτηκε με το πρωτότυπο κινητό που είχαν στην διάθεσή τους και στην εταιρεία SONY για την γρήγορη ανταπόκριση αναφορικά με προβλήματα που αντιμετώπισα κατά την ανάπτυξη της εφαρμογής. Τέλος, ένα μεγάλο ευχαριστώ στην οικογένεια μου, για την ηθική και πρακτική υποστήριξή τους σε κάθε μου επιλογή.

Θα ήθελα να αφιερώσω την εργασία αυτή στον πατέρα μου, Αντώνη, και την γιαγιά μου, Κική, που απεβίωσαν, κατά την διάρκεια των σπουδών μου.

Περίληψη

Τα τελευταία χρόνια οι όροι μηχανική μάθηση και τεχνητή νοημοσύνη εισέρχονται ολοένα και περισσότερο στο λεξιλόγιο μας. Αυτό οφείλεται στην εξαιρετική ανάπτυξη που είχαν οι συγκεκριμένες τεχνολογίες, λόγω του τεράστιου όγκου πληροφοριών που υπάρχει διαθέσιμος στο διαδίκτυο. Οι εφαρμογές αυτών των τεχνολογιών είναι αμέτρητες, με τις πιο γνωστές, στους κλάδους της αυτοκινητοβιομηχανίας με τα self-driving cars, στην ιατρική με αλγόριθμους για αυτόματη ανίχνευση καρκινωμάτων και άλλων ασθενειών και στον τομέα της αστυνόμευσης, με λογισμικό που ανιχνεύει αυτόματα επικίνδυνες συμπεριφορές σε χώρους όπως αεροδρόμια. Παρόλα αυτά, ένας τομέας στον οποίο η μηχανική μάθηση μπορεί να χρησιμοποιηθεί περισσότερο, είναι στην παροχή βοήθειας σε άτομα που αντιμετωπίζουν ανικανότητα σε διάφορους τομείς. Συγκεκριμένα, γίνονται προσπάθειες για να αναπτυχθούν τεχνολογίες για την επίλυση αυτού του προβλήματος όπως αυτόματη ανάγνωση κειμένων για τυφλούς, αυτόματοι υπότιτλοι με μεγάλη ακρίβεια για άτομα με περιορισμένη ακοή και συστήματα μετακίνησης για άτομα με κινητικά προβλήματα. Εντούτοις, η ανάγκη για βοήθεια αυτών των ατόμων εξακολουθεί να υπάρχει.

Στην προσπάθεια μου να συνεισφέρω σε αυτό το σημαντικό έργο, στην παρούσα διπλωματική εργασία, ανέπτυξα ένα λογισμικό για κινητά, που χρησιμοποιεί κάμερες TOF και μηχανική μάθηση. Κύρια λειτουργία της εφαρμογής αυτής, είναι να ανιχνεύει αντικείμενα στον χώρο και να ενημερώνει τον χρήστη για την ύπαρξη τους, ως επίσης και την απόσταση τους από αυτόν. Με αυτό τον τρόπο ο χρήστης μπορεί να “δει” το χώρο γύρω του και να αποφύγει τυχόν κίνδυνους.

Στην παρούσα εργασία περιγράφονται οι αποφάσεις που λήφθηκαν κατά την υλοποίηση της εφαρμογής, τα προβλήματα που αντιμετωπίστηκαν, αλλά και τα αποτελέσματά της και πιθανές μελλοντικές βελτιώσεις της. Η συγκεκριμένη εφαρμογή δουλεύει, προς το παρόν, μόνο στο πρωτότυπο κινητό που χορηγήθηκε από τη Sony, αλλά μελλοντικά θα μπορεί να χρησιμοποιηθεί σε κάθε κινητό που έρχεται εξοπλισμένο με TOF κάμερα. Η υλοποίηση έχει τη δυνατότητα να αναγνωρίσει αντικείμενα σε 80 γενικές κλάσεις και χρησιμοποιώντας ομιλία περιγράφει τα αντικείμενα και την απόσταση τους στον χρήστη.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Γενική Εισαγωγή

Τα τελευταία χρόνια η τεχνολογία έγινε απαραίτητο μέρος της καθημερινότητας μας. Με την εξέλιξη του διαδικτύου κάθε μέρα αναπτύσσονται νέες εφαρμογές, οι οποίες αποσκοπούν στην βελτίωση του βιοτικού επιπέδου, με την αυτοματοποίηση διαδικασιών να είναι ο κύριος στόχος. Σήμερα, χωρίς καμία σωματική μετακίνηση, είναι δυνατή η παράδοση οποιουδήποτε προϊόντος ή υπηρεσίας κατοίκον. Είναι επιπρόσθετα δυνατή η παρακολούθηση των ενδιαφερόντων του χρήστη, μέσω της ανάλυσης των προτιμήσεων και της πρότασης παρόμοιων, μέσω των έξυπνων υπολογιστικών συστημάτων.

Με την ανάπτυξη τεχνολογιών όπως η μηχανική μάθησή και η τεχνητή νοημοσύνη οι υπολογιστές μπορούν πλέον να εκτελέσουν διαδικασίες, όχι μόνο πιο γρήγορα, αλλά πολλές φορές και με μεγαλύτερη ακρίβεια από ότι ο άνθρωπος. Η έρευνα και η καινοτομία σε αυτό το κομμάτι της επιστήμης της πληροφορικής κατάφερε να επιλύσει προβλήματα που στο παρελθόν θεωρούνταν απίθανα και έχει άμεση εφαρμογή στην πραγματική ζωή. Προβλήματα όπως η κατηγοριοποίηση εικόνων και η αναγνώριση αντικειμένων ή προσώπων, που παλαιότερα θα χρειάζονταν υπερβολικά μεγάλο χρόνο για να λυθούν, με τη χρήση της μηχανικής μάθησης εκτελούνται σε κλάσματα δευτερολέπτου και σε μηχανές που μπορούν να χωρέσουν στην τσέπη σου.

Η σημαντικότητα αυτών των ανακαλύψεων θεμελιώθηκε περαιτέρω με την απονομή του βραβείου Turing για το 2018 [1] στους πατέρες των βαθιών νευρωνικών δικτύων, Yoshua Bengio, Geoffrey Hinton, και Yann LeCun, αναγνωρίζοντας έτσι τα νευρωνικά δίκτυα σαν ένα βιώσιμο ερευνητικό πεδίο.

1.2 Περιγραφή και Κίνητρο

Στο μάθημα «Επιχειρηματικότητα και Καινοτομία», η καθηγήτρια σαν παράδειγμα εξαιρετικής καινοτόμου ιδέας ανέφερε την εφαρμογή "be my eyes", μια εφαρμογή όπου εθελοντές μπορούν να συνδεθούν με άτομα με χαμηλή ή καθόλου ορατότητα και βλέποντας από την κάμερα του ατόμου, τους καθοδηγούν με ασφάλεια στον προορισμό τους. Επομένως, μέσω κάποιων εφαρμογών, άτομα με ανίατα προβλήματα μπορούν να βελτιώσουν την καθημερινότητά τους.

Εμπνευσμένος από αυτή την ιδέα και με την ελπίδα να βοηθήσω άτομα με μειωμένη όραση σκέφτηκα να αναπτύξω μια εφαρμογή που θα βοηθά τον χρήστη να «δει» τον κόσμο γύρω του, χρησιμοποιώντας μηχανική μάθηση και τις κάμερες ενός κινητού. Εκτός από την όραση, οι χρήστες τις εφαρμογής έχουν να αντιμετωπίσουν και την πίεση του ότι εξαρτούνται συνεχώς από άλλα άτομα στην καθημερινή τους ζωή. Η λύση που προτείνεται δίνει στον χρήστη την αυτονομία που έχει ανάγκη, αφού θα αντικαθιστά τον ανθρώπινο παράγοντα με σύγχρονες τεχνολογίες.

Η εφαρμογή που θα αναπτυχθεί σε αυτή την εργασία θα χρησιμοποιεί κάμερα, υπέρυθρους αισθητήρες βάθους TOF, state of the art αλγόριθμους αναγνώρισης αντικειμένων και σύνθεση ομιλίας με σκοπό να περιγράφει στον χρήστη τι υπάρχει γύρω του και πόσο μακριά βρίσκεται. Η περιγραφή των αντικειμένων θα μεταδίδεται στον χρήστη με ηχητικά μηνύματα, δηλαδή η εφαρμογή θα μιλά στον χρήστη.

1.3 Στόχος Εργασίας και Συνεισφορά

Αυτή η εργασία αποσκοπεί στην ανάπτυξη ενός λογισμικού που μπορεί να βοηθήσει άτομα με προβλήματα όρασης να αντιλαμβάνονται τι υπάρχει στον περίγυρω τους, χωρίς

να εξαρτούνται από δεύτερο άτομο. Επίσης, μέσω αυτής της εργασίας γίνεται η προσπάθεια να προταθεί ένας καινούριος τρόπος χρήσης των καμερών TOF που τα τελευταία χρόνια άρχισαν να εμφανίζονται στα κινητά τηλέφωνα.

1.4 Δομή Εργασίας

Στο δεύτερο κεφάλαιο γίνονται περιγραφές για όρους και τεχνολογίες οι οποίες χρησιμοποιήθηκαν για να την υλοποίηση της εφαρμογής. Επιπρόσθετα, γίνεται αναφορά στη μηχανική μάθηση, τα νευρωνικά δίκτυα, επεξήγηση των όρων αναγνώριση και ανίχνευση αντικειμένων και επεξήγηση της λειτουργίας των Time Of Flight (TOF) καμερών. Επίσης, αναφέρουμε την σχετική βιβλιογραφία που ενέπνευσε αυτή την εφαρμογή.

Στο τρίτο κεφάλαιο γίνεται αναφορά στην μεθοδολογία της πτυχιακής εργασίας, δηλαδή γίνεται μια περιγραφή των συγκεκριμένων αλγόριθμων και τεχνολογιών που χρησιμοποιήθηκαν.

Στο τέταρτο κεφάλαιο περιγράφεται η υλοποίηση της εφαρμογής, δηλαδή όλα τα στάδια της ανάπτυξης καθώς και τα προβλήματα που αντιμετωπίστηκαν.

Τέλος, στο πέμπτο κεφάλαιο δίνονται τα συμπεράσματα και πώς αυτή η υλοποίηση μπορεί να βελτιωθεί μελλοντικά.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 Μηχανική Μάθηση

Η μηχανική μάθηση είναι υποπεδίο της τεχνητής νοημοσύνης και κατ' επέκταση της επιστήμης της πληροφορικής που μελετά αλγορίθμους που βελτιώνουν την επίδοσή τους, μέσω εμπειριών. Ένας ορισμός για την μηχανική μάθηση δόθηκε από τον Άρθουρ Σάμουελ, το 1959, ως το "Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί" [2]. Μέσω της μηχανικής μάθησης αναπτύσσονται αλγόριθμοι που εκπαιδεύονται με κάποια πειραματικά δεδομένα και βάση της εκπαίδευσης τους μπορούν να κάνουν προβλέψεις ή να δώσουν κάποιο αποτέλεσμα.

Η διαδικασία της μάθησης ξεκινά με δεδομένα όπως παραδείγματα της αναμενόμενης εξόδου ή μέσω εμπειρίας, προκειμένου να βρεθούν μοτίβα στα δεδομένα ή να λαμβάνονται καλύτερες αποφάσεις στο μέλλον. Ο πρωταρχικός στόχος της εκπαίδευσης είναι να επιτρέπεται στους υπολογιστές να μαθαίνουν αυτόματα χωρίς ανθρώπινη παρέμβαση ή βοήθεια και να προσαρμόζουν ανάλογα τις ενέργειές τους. Υπάρχουν τρεις βασικοί μέθοδοι εκμάθησης. Στην επιβλεπόμενη μάθηση, κατά την εκπαίδευση, μαζί με τα δεδομένα εισόδου δίνεται και το αναμενόμενο αποτέλεσμα. Έτσι, το πρόγραμμα αλλάζει τις παραμέτρους του για να πλησιάζει την αναμενόμενη έξοδο και διδάσκει ένα

γενικό κανόνα για την αντιστοιχία εισόδου και αποτελέσματος. Στη μη επιβλεπόμενη μάθηση, το σύστημα χωρίς κάποια εξωτερική βοήθεια προσπαθεί να βρει αντιστοιχίες μεταξύ των δεδομένων και τα ομαδοποιεί. Με αυτό τον τρόπο το σύστημα μπορεί να ανακαλύψει κρυφά μοτίβα που δεν θα διέκρινε ένας άνθρωπος ή να εξάγει χαρακτηριστικά μεταξύ παρομοίων αντικειμένων. Τέλος, στην ενισχυτική μάθηση το πρόγραμμα εκπαιδεύεται με κάποια ανταμοιβή όταν πλησιάζει το σωστό αποτέλεσμα ή τιμωρία για το αντίθετο. Χρησιμοποιώντας αυτό το είδος μάθησης μπορεί να προγραμματιστεί ο υπολογιστής να παίζει καλά κάποιο παιχνίδι ή να οδηγήσει ένα αυτοκίνητο χωρίς να προκαλέσει ατύχημα.

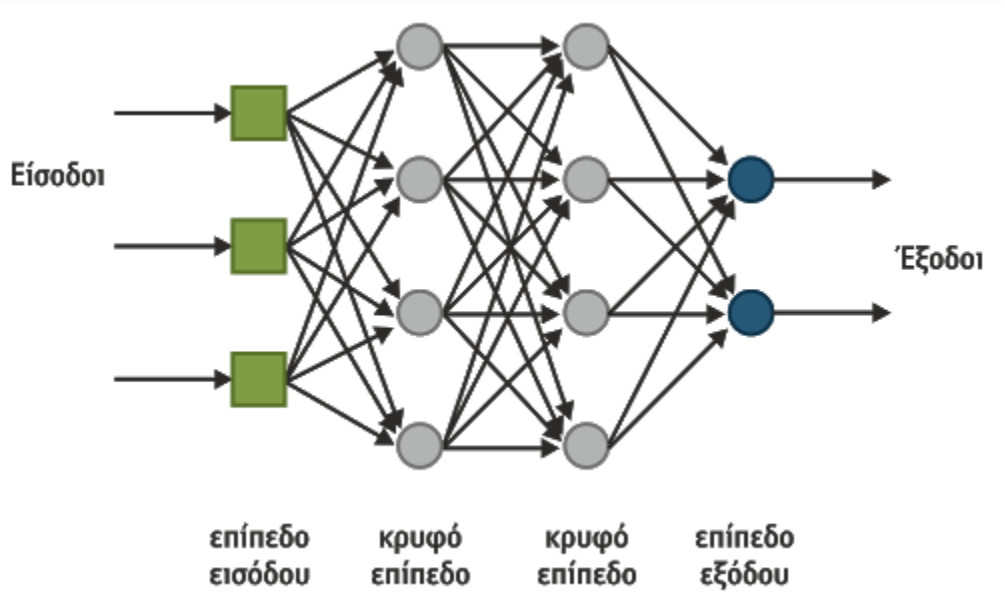
Η μηχανική μάθηση έχει πολλές εφαρμογές σε διάφορους τομείς. Τα πιο γνωστά παραδείγματα εφαρμογών που βασίζονται στην μηχανική μάθηση είναι η οπτική αναγνώριση χαρακτήρων (Optical Character Recognition – OCR), η αναγνώριση αντικειμένων σε μια εικόνα (Object Recognition), οι μηχανές αναζήτησης, οι εξατομικευμένες διαφημίσεις (personalized ads) και τα αυτοκινούμενα αυτοκίνητα (self-driving cars). Αποκαλείται από πολλούς ως το «μέλλον» και είναι ένας από τους πιο ραγδαία αναπτυσσόμενους ερευνητικούς κλάδους στην πληροφορική.

2.2 Νευρωνικά Δίκτυα

Τα νευρωνικά δίκτυα είναι υπολογιστικά συστήματα που μπορούν να εκπαιδευτούν για την επίλυση κάποιου προβλήματος. Το όνομα και η δομή τους είναι εμπνευσμένα από τον ανθρώπινο εγκέφαλο και μιμούνται τον τρόπο που οι βιολογικοί νευρώνες σηματοδοτούν ο ένας τον άλλο. Τα τεχνητά νευρωνικά δίκτυα αποτελούνται από στρώματα νευρώνων, που περιέχουν ένα επίπεδο εισόδου, ένα ή περισσότερα κρυμμένα επίπεδα και ένα επίπεδο εξόδου. Κάθε νευρώνας συνδέεται με έναν άλλο και έχει σχετικό βάρος και κατώφλι. Εάν η έξοδος οποιουδήποτε νευρώνα είναι πάνω από την καθορισμένη τιμή κατωφλίου, αυτός ο νευρώνας ενεργοποιείται και γίνεται αποστολή δεδομένων στο επόμενο επίπεδο του δικτύου. Τα νευρωνικά δίκτυα βασίζονται σε δεδομένα εκπαίδευσης για να μάθουν και να βελτιώσουν την ακρίβειά τους με την πάροδο του χρόνου. Ωστόσο, όταν αυτοί οι αλγόριθμοι ρυθμιστούν σωστά για την ακρίβεια, μας επιτρέπουν να ομαδοποιήσουμε (cluster) και να κατατάξουμε (classify)

δεδομένα με υψηλή ταχύτητα. Ένα από τα πιο γνωστά παραδείγματα χρήσης νευρωνικού δικτύου είναι ο αλγόριθμος αναζήτησης της Google [3].

Κάθε επίπεδο του νευρωνικού δικτυού αποτελείται από ένα σύνολο νευρώνων που ενώνονται με το επόμενο επίπεδο. Στο επίπεδο εισόδου έχουμε νευρώνες εισόδου, στα κρυφά επίπεδα έχουμε κρυφούς νευρώνες και στο επίπεδο εξόδου νευρώνες εξόδου. Οι νευρώνες εισόδου δέχονται δεδομένα από το περιβάλλον και τα διοχετεύουν στο δίκτυο χωρίς να εκτελούν κάποια πράξη. Οι κρυφοί νευρώνες πολλαπλασιάζουν κάθε είσοδο με το αντίστοιχο βάρος, που έχει η σύναψη και υπολογίζουν το άθροισμα όλων των γινομένων. Αυτό στη συνέχεια χρησιμοποιείται σαν είσοδος σε μια συνάρτηση ενεργοποίησης (activation function). Το αποτέλεσμα της ενεργοποίησης διοχετεύεται σαν είσοδος στο επόμενο επίπεδο, εκτός σε περιπτώσεις που έχουμε νευρώνα εξόδου που το αποτέλεσμα επιστρέφεται στο περιβάλλον σαν έξοδος του δικτύου. Το κύριο πλεονέκτημα των νευρωνικών δικτύων είναι η δυνατότητα τους να εκπαιδευτούν. Μπορούμε να ορίσουμε σαν εκπαίδευση την βελτίωση των αποτελεσμάτων για την επίλυση κάποιου προβλήματος. Για να εκπαιδευτεί το δίκτυο εκτελείται πολλές φορές για κάποια εκπαιδευτικά δεδομένα και σε κάθε εκτέλεση προσαρμόζονται οι παραμέτροι (βάρη και κατώφλια), έτσι ώστε η έξοδος να πλησιάζει το επιθυμητό αποτέλεσμα. Η αλλαγή των βαρών μπορεί να επιτευχθεί με τις μεθόδους της ανάστροφης μετάδοσης λάθους (back – propagation) και κατάβασης κλήσης (gradient descent). Όταν το δίκτυο επιλύει το πρόβλημα με ικανοποιητικά αποτελέσματα σταματά η εκπαίδευση και ελέγχεται με κάποια δοκιμαστικά δεδομένα κατά πόσον η υλοποίηση γενικεύει, δηλαδή αν επιλύει το πρόβλημα για οποιαδήποτε είσοδο ή αν υπερ-εκπαιδεύτηκε για τα δεδομένα εκπαίδευσης.



Σχήμα 2.1 Παράδειγμα απλού νευρωνικού δικτύου.

Η ιδέα των νευρωνικών δικτύων προτάθηκε το 1943 από τους Warren McCullough και Walter Pitts [4] αλλά λόγω των περιορισμών στην τεχνολογία της εποχής δεν υλοποιήθηκαν για τα επόμενα 20 χρόνια. Η έρευνα στα νευρωνικά δίκτυα είχε μια αναβίωση κατά τη δεκαετία του 1980, αλλά σύντομα πήγε και πάλι στο παρασκήνιο. Έκανε όμως την επιστροφή της για τα καλά στις αρχές του 21^{ου} αιώνα, τροφοδοτημένη σε μεγάλο βαθμό από την αυξημένη υπολογιστική ισχύ των καρτών γραφικών. Τα τελευταία χρόνια η έρευνα σχετικά με την ανάπτυξη και χρήση των νευρωνικών δικτύων έχει φτάσει στο απόγειο, με όλες τις μεγάλες εταιρείες λογισμικού, όπως Google, IBM, Amazon, να επενδύουν εκατομμύρια σε αυτήν.

Τα τελευταία χρόνια αναπτύσσεται μια καινούρια οικογένεια νευρωνικών δικτύων, τα συνελικτικά νευρωνικά δίκτυα (convolutional networks). Ένα συνελικτικό δίκτυο έχει στο κρυφό στρώμα του συνελικτικά επίπεδα τα οποία εκτελούν συνέλιξη φίλτρων σε μια εικόνα που δέχεται σαν είσοδο. Τα φίλτρα αυτά καθορίζονται κατά την διαδικασία της μάθησης και εξάγουν χαρακτηριστικά που περιγράφουν την κλάση της εικόνας. Σε κάθε συνελικτικό επίπεδο μπορεί να εκτελείται συνέλιξη φίλτρου, που μεγαλώνει τις διαστάσεις της εικόνας που αναλύεται, διαδικασία συγκέντρωσης (pooling) που μειώνει τις διαστάσεις, προσθήκη μη γραμμικής συνθήκης ενεργοποίησης (ReLU) και κανονικοποίηση των δεδομένων. Μέσω της επεξεργασίας που γίνεται στα συνελικτικά

επίπεδα το δίκτυο ξεχωρίζει την ύπαρξη χαρακτηριστικών της εικόνας και βάση αυτών, ανάλογα με τον αλγόριθμο που υλοποιεί στα τελευταία επίπεδα, την κατηγοριοποιεί ή ανιχνεύει αντικείμενα σε αυτή [5].

2.3 Αναγνώριση/ Ανίχνευσή αντικειμένων

Η αναγνώριση αντικειμένων (object recognition) είναι μια τεχνολογία που υπάγεται στον κλάδο της μηχανικής όρασης και ο σκοπός της είναι να εντοπίζει και να ονομάζει αντικείμενα σε μια εικόνα ή βίντεο. Ο ανθρώπινος εγκέφαλος μπορεί να ξεχωρίσει αντικείμενα χωρίς ιδιαίτερη προσπάθεια ακόμα και αν αυτά έχουν διαφορετικό μέγεθος, περιστροφή ή επικαλύπτονται από άλλα αντικείμενα. Αυτή η ανίχνευση εξακολουθεί να αποτελεί πρόκληση για τα συστήματα μηχανικής όρασης και έχουν αναπτυχθεί διάφορες προσεγγίσεις για την επίλυση του προβλήματος τις τελευταίες δεκαετίες. Η αναγνώριση αντικειμένων βρίσκει χρήση σε εφαρμογές ανίχνευσης ή αναγνώρισης προσώπων, αυτόματο σχολιασμό εικόνας (image annotation) ή παρακολούθηση αντικειμένων ή ανθρώπων σε βίντεο.

Η γενική ιδέα είναι ότι κάθε κλάση αντικειμένων έχει τα δικά της χαρακτηριστικά που την κάνουν να ξεχωρίζουν από τις υπόλοιπες. Για παράδειγμα, ένας κύκλος έχει στρογγυλό σχήμα, άρα για ανίχνευση κύκλων αναζητούνται αντικείμενα που έχουν την ίδια απόσταση από κάποιο κέντρο. Ένα άλλο παράδειγμα, στην ανίχνευση προσώπων γίνεται αναζήτηση ανθρώπινων χαρακτηριστικών (μάτια, μύτη, χείλη), αλλά και χαρακτηριστικά όπως η μέτρησης της απόστασης ανάμεσα στα μάτια.

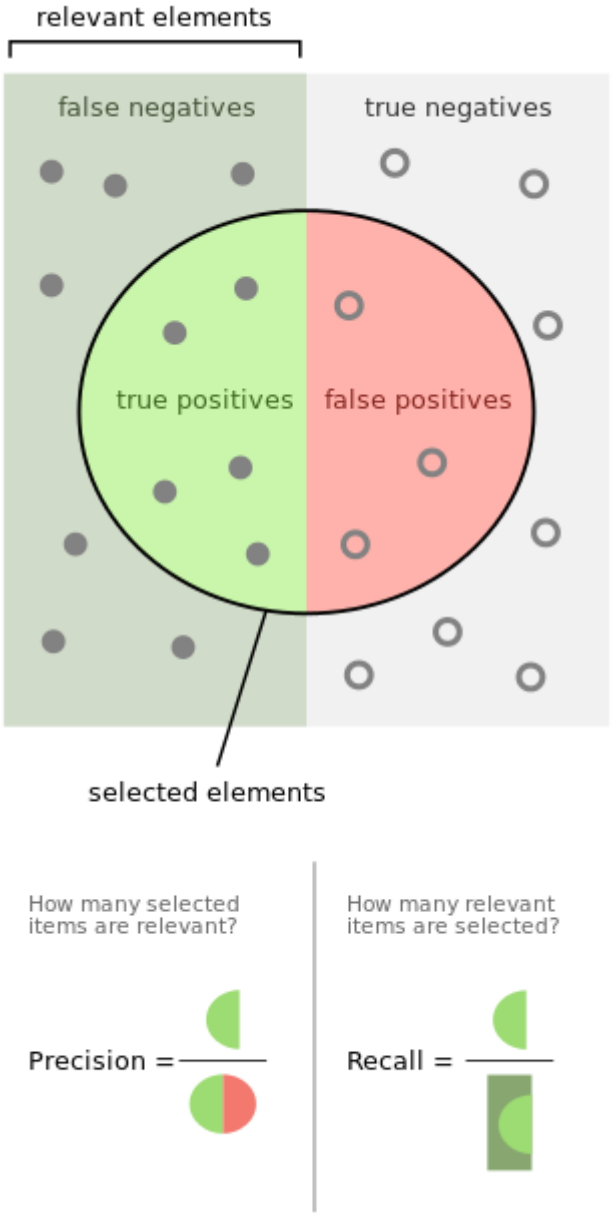
Οι μέθοδοι για την ανίχνευση αντικειμένων, εμπίπτουν γενικά είτε σε προσεγγίσεις που βασίζονται σε μηχανική μάθηση είτε σε βαθιά μάθηση. Για προσεγγίσεις μηχανικής μάθησης πρέπει πρώτα να οριστούν τα χαρακτηριστικά, χρησιμοποιώντας αλγόριθμους όπως Viola–Jones [6], Scale-Invariant Feature Transform (SIFT) [7] και Histogram of Oriented Gradients (HOG) [8], και μετά να περαστούν από δεύτερο σύστημα για να γίνει η κατηγοριοποίηση. Οι αλγόριθμοι μηχανικής μάθησης εκπαιδεύονται να εντοπίζουν συγκεκριμένα χαρακτηριστικά που έχει το κάθε αντικείμενο, για παράδειγμα μάτια και μύτη αν η ανίχνευση αφορά πρόσωπα. Λόγω της πολυπλοκότητας των αντικειμένων προς ανίχνευση χρησιμοποιούνται αλγόριθμοι, όπως το AdaBoost [9], που επιλέγουν τα

καλύτερα χαρακτηριστικά ή συνδυάζουν “αδύναμα” χαρακτηριστικά για να παράξουν “δυνατά”. Από την άλλη πλευρά, οι τεχνικές βαθιάς μάθησης μπορούν να εκτελέσουν την ανίχνευση αντικειμένων χωρίς να ορίζουν συγκεκριμένα χαρακτηριστικά, χρησιμοποιώντας συνελικτικά νευρωνικά δίκτυα. Μέσω των συνελικτικών δικτύων και μεγάλου αριθμού δεδομένων, δίδεται η δυνατότητα να βρεθούν αυτόματα τα χαρακτηριστικά που ορίζουν κάποιο αντικείμενο και χωρίς να χρειάζεται να οριστούν από πριν από κάποιο άλλο σύστημα ή άνθρωπο. Λόγω του τεράστιου όγκου πληροφοριών που είναι άμεσα διαθέσιμες στο διαδίκτυο σήμερα, η εκπαίδευση βαθιών συνελικτικών δικτυών έγινε εφικτή με αποτέλεσμα αυτά τα δίκτυα να αντικαταστήσουν τις απλές τεχνικές μηχανικής μάθησης που χρησιμοποιούνταν στο παρελθόν. Ακόμα ένας λόγος που εξηγεί την αντικατάσταση, είναι ότι η χρήση των συνελικτικών δικτύων παρέχει καλύτερη ακρίβεια και ταχύτητα ανίχνευσης αντικειμένων. Στην εργασία αυτή, θα χρησιμοποιηθεί συνελικτικό δίκτυο που εκπαιδευτικέ για να ξεχωρίζει 80 διαφορετικές κλάσης αντικειμένων και ο αλγόριθμος Single-Shot Detector (SSD) για την ανίχνευση των αντικειμένων που θα περιγράφονται στον χρήστη.

Επειδή όμως, δεν είναι όλοι οι αλγόριθμοι ανίχνευσης το ίδιο αποδοτικοί, χρειάζεται κάποια μετρική που να συγκρίνει την ακρίβεια της αναγνώρισης. Η πιο δημοφιλής μετρική για σύγκριση μοντέλων ανίχνευσης αντικειμένων είναι το mAP (mean average precision). Για να εξηγηθεί το mAP, είναι σημαντικό πρώτα να εξηγηθεί πως υπολογίζεται η ακρίβεια (precision) και η ανάκληση (recall) που χρησιμοποιούνται στον υπολογισμό. Η ακρίβεια ορίζεται ως ο λόγος των πραγματικών θετικών (true positive) και το σύνολο των προβλεπόμενων θετικών (total number of predicted positives). Η ανάκληση ορίζεται ως ο λόγος των πραγματικών θετικών και του συνόλου των αναμενόμενων προβλέψεων (total of ground truth positives). Στους πιο κάτω τύπους TP είναι το True Positive, FP είναι το False Positive και FN είναι το False Negative:

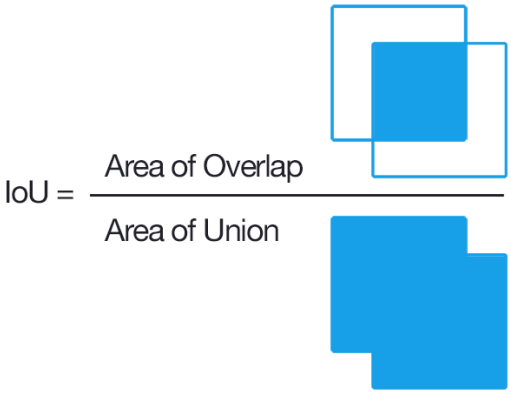
$$\text{Ακρίβεια} = \frac{TP}{TP+FP} \qquad \text{Ανάκληση} = \frac{TP}{TP+FN}$$

Για να βελτιωθεί η ακρίβεια πρέπει να μειωθεί ο αριθμός των λανθασμένων προβλέψεων (false positive). Ομοίως, αν μειωθεί ο αριθμός των false negative αυξάνεται η ανάκληση και μειώνεται η ακρίβεια.



Σχήμα 2.2 Στο πιο πάνω σχήμα φαίνεται γραφικά ο υπολογισμός της ακριβείας και της ανάκλασης. [10]

Για να οριστεί μια ανίχνευση σαν θετική ή αρνητική πρέπει να εντοπιστεί πόσο το bounding box της συμπίπτει με το αληθινό (ground truth). Η ανάθεση γίνεται με την χρήση του Intersection over Union (IoU), δηλαδή τη διαίρεση της τομής των δύο κουτιών με την ένωση τους.



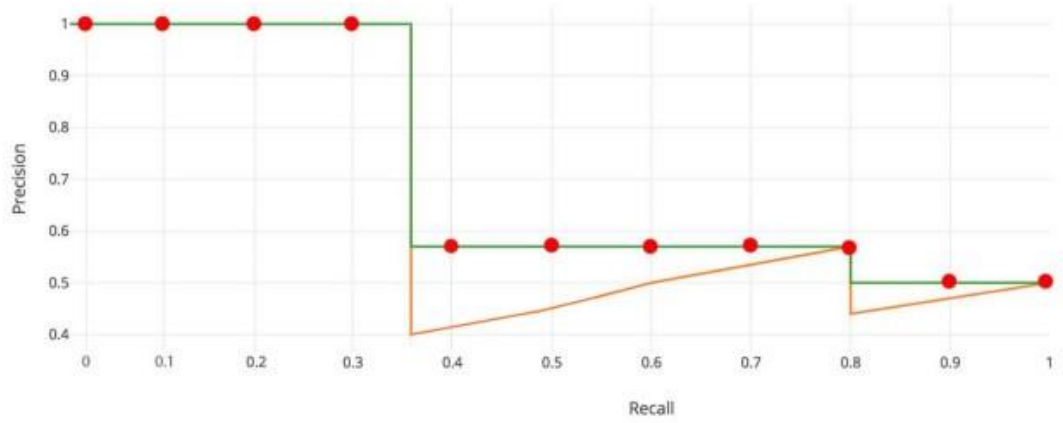
Σχήμα 2.3 Στο πιο πάνω σχήμα φαίνεται γραφικά ο υπολογισμός του IoU, τι σημαίνει τομή και τι ένωση. [10]

Αν το αποτέλεσμα της διαίρεσης είναι μεγαλύτερο από κάποιο threshold, συνήθως 0,5 , τότε η ανίχνευση θεωρείται ως true positive, αλλιώς ως false positive. Περιπτώσεις όπου υπάρχουν διπλά κουτιά θεωρούνται false positive. Περιπτώσεις που έπρεπε να υπάρχει ανίχνευση αλλά δεν λαμβάνεται κάποιο αποτέλεσμα ή η κλάση του αντικειμένου είναι λάθος, ορίζονται σαν false negatives. Αφού ορίστηκαν τα true positive, false positive και false negatives μπορούν τώρα να υπολογιστούν η ακρίβεια και η ανάκληση της ανίχνευσής για μια δεδομένη κατηγορία σε ολόκληρο το σύνολο δοκιμών (test set). Πριν σχεδιαστεί η γραφική παράσταση της ακρίβειας και της ανάκλησης πρέπει να υπολογιστεί η παρεμβολή ακρίβειας (Interpolated precision). Η παρεμβολή υπολογίζεται ως η μέγιστη ακρίβεια για κάθε τιμή ανάκλησης. Τέλος, σχεδιάζεται η γραφική παράστασή της ακριβείας και της παρεμβολής της ακριβείας έναντι της ανάκλησης και για AP (average precision) λαμβάνεται το εμβαδόν κάτω από την καμπύλη που δημιουργείται. Για τον υπολογισμό του εμβαδού χωρίζεται η γραφική παράσταση σε 11 μέρη {0,0.1,0.2,…,0.9,1} και υπολογίζεται ο μέσος όρος της παρεμβολής της ακρίβειας για κάθε μέρος. Στην πιο κάτω εξίσωση για τον υπολογισμό του AP το Pinter(r) είναι η τιμή της παρεμβολής στην θέση r.

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,\ldots 0.9,1\}} Pinter(r)$$

Πιο κάτω παρουσιάζεται ένα παράδειγμα της γραφικής παράστασης ακρίβειας και ανάκλησης με την πορτοκαλιά γραμμή να είναι η ακρίβεια, η πράσινη να είναι η

παρεμβολή της ακρίβειας και τα κόκκινα σημεία να δείχνουν τα 11 σημεία που θα χρησιμοποιηθούν για τον υπολογισμό του AP.



Σχήμα 2.4 Παράδειγμα γραφικής παράστασης για τον υπολογισμό του mean Average Precision [10].

Στις πιο πρόσφατες έρευνες, η σύγκριση μεταξύ αλγορίθμων γίνεται μόνο για το dataset COCO [11] και χρησιμοποιείται ο μέσος της μετρικής AP για διάφορα IoU thresholds. IoU threshold είναι η ελάχιστη τιμή που χρειάζεται να επιστραφεί για να θεωρηθεί μία ανίχνευση αληθής (true positive). Συγκεκριμένα, για το COCO dataset χρησιμοποιούνται τιμές κατωφλίου από το εύρος τιμών 0,5 μέχρι 0,95 με βηματική αλλαγή 0,05 μεταξύ πειραμάτων. Η μετρική COCO mAP ορίζεται ως ο μέσος όρος των 10 επιπέδων IoU στις 80 κλάσεις που περιγράφει το COCO. Για την επιλογή του κατάλληλου νευρωνικού δικτυού που χρησιμοποιήθηκε στην εργασία, οι αποφάσεις λήφθηκαν βάση της πιο πάνω μετρικής και της ταχύτητας της αναγνώρισης.

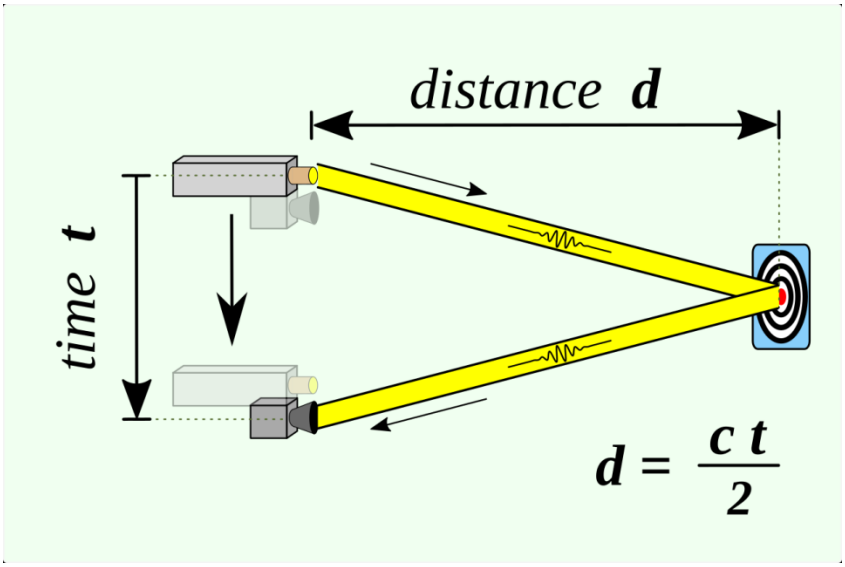
2.4 Time-of-Flight Camera

Μια Time-of-Flight (TOF) κάμερα είναι ένα σύστημα αισθητήρων που υπολογίζει την απόσταση κάθε σημείου στην οθόνη. Χρησιμοποιεί τον χρόνο που χρειάζεται μια ακτίνα φωτός για να ταξιδέψει προς και πίσω από κάποιο αντικείμενο και τη γνωστή ταχύτητα του φωτός για τον υπολογισμό της απόστασης . Η TOF κάμερα αποτελείται από μια πηγή φωτός, ένα φακό, ένα αισθητήρα εικόνας και μια διασύνδεση για τους απαραίτητους υπολογισμούς. Η πηγή φωτός είναι συνήθως υπέρυθρο λέιζερ που εκπέμπει φωτισμό σε παλμούς, ο οποίος αντανακλάται από τα αντικείμενα μπροστά από την κάμερα και

επιστρέφει σε αυτή. Ο φακός μαζεύει το αντανακλασμένο λέιζερ και αφού το φιλτράρει, έτσι ώστε να κρατήσει μόνο τις συχνότητες φωτός που έχουν το ίδιο μήκος κύματος, το διοχετεύει στον αισθητήρα. Ο αισθητήρας με τη σειρά του υπολογίζει την απόσταση χρησιμοποιώντας τον χρόνο που πέρασε από την στιγμή που αποστάλθηκε το σήμα φωτός μέχρι να επιστρέψει σε αυτόν και την ταχύτητα του φωτός και την αναθέτει στο ανάλογο σημείο στην εικόνα. Με αυτό τον τρόπο υπολογίζεται η απόσταση όλων των σημείων στην σκήνη σε πραγματικό χρόνο.

Πλεονέκτημα των καμερών TOF έναντι άλλων τροπών υπολογισμού απόστασης είναι ότι προσφέρουν υπολογισμό σε πραγματικό χρόνο με εύκολη εγκατάσταση και οικονομικά αποδοτική τεχνική. Άλλο πλεονέκτημα της συγκεκριμένης τεχνικής είναι ότι έχει πολύ καλά αποτελέσματα σε περιπτώσεις με λίγο ή καθόλου φωτισμό. Παρά τα πλεονεκτήματα αυτά, υπάρχουν περιορισμοί που πρέπει να ληφθούν υπόψη. Φωτεινές επιφάνειες κοντά στην κάμερα σκορπίζουν πολύ φως μέσα στον φακό και παραμορφώνουν το αποτέλεσμα. Για τον υπολογισμό της απόστασης το φωτεινό σήμα θέλουμε να αντανακλαστεί μόνο μια φορά προς την κάμερα. Αν το φως αντανακλάται περισσότερες από μία φορά, τότε το αποτέλεσμα θα είναι αλλοιωμένο. Επομένως, οι ανακλαστικές και γυαλιστερές επιφάνειες είναι πιθανόν να δώσουν λάθος αποτελέσματα. Το έντονο περιβαλλοντικό φως (ambient light), όπως ο ήλιος, μπορεί να επηρεάσει αρνητικά τον υπολογισμό της απόστασης και έτσι η χρήση της TOF κάμερας σε εξωτερικούς χώρους είναι πιο δύσκολη. Τέλος, η χρήση πολλών TOF καμερών στον ίδιο χώρο μπορεί να προκαλέσει λάθος αποτελέσματα, επειδή μπορεί να λαμβάνεται το φως που έστειλαν οι υπόλοιπες κάμερες.

Η TOF κάμερα σε συνεργασία με μια παραδοσιακή κάμερα, μπορεί να χρησιμοποιηθεί σε διάφορες εφαρμογές μηχανικής όρασης (Machine Vision), αφού η γνώση της απόστασης των σημείων της εικόνας κάνει την εξαγωγή του παρασκηνίου της εικόνας πολύ πιο εύκολη. Σήμερα, οι TOF κάμερες χρησιμοποιούνται στην ρομποτική και στον αυτοματισμό εργοστασίων για να σηκώνουν και να μετακινούν αντικείμενα και στην ιατρική για την επίβλεψη της τοποθέτησης των ασθενών. Στα πλαίσια αυτής της διπλωματική εργασίας, η TOF κάμερα θα χρησιμοποιηθεί για ενημέρωση του χρήστη που έχει προβλήματα όρασης για την απόσταση των αντικειμένων που βρίσκονται γύρω του.



Σχήμα 2.5 Στο πιο πάνω σχήμα φαίνεται υπολογισμός της απόστασης χρησιμοποιώντας τον χρόνο που χρειάζεται να ταξιδέψει ένα φωτεινό σήμα σε κάποιο σημείο και πίσω στον δέκτη. Πολλαπλασιάζουμε τον χρόνο με την ταχύτητα του φωτός (c) και διαιρούμε δια 2. [12]

2.5 Σχετική βιβλιογραφία

Στην ανασκόπηση της βιβλιογραφίας με στόχο την υλοποίηση της εφαρμογής εντοπίστηκαν τρία εξαιρετικά βοηθητικά άρθρα που βοήθησαν τις επιλογές που έγιναν. Αρχικά, το άρθρο για την υλοποίηση της εφαρμογής “be my eyes” [13], που ήταν και η αρχική έμπνευση για την δική μου εφαρμογή, αναφέρει ότι παρέχει στους χρήστες του, οι οποίοι πάσχουν από μερική ή ολική τύφλωση, την δυνατότητα να χρησιμοποιήσουν το κινητό τους με φωνητικά μηνύματα και να συνδεθούν σε πραγματικό χρόνο με κάποιο εθελοντή ο οποίος έχει την όραση του για να τους βοηθήσει με κάποια ασχολία. Το κίνητρο αυτής της εφαρμογής μοιάζει πολύ με το δικό μου, δηλαδή προσπαθεί να βοηθήσει άτομα με προβλήματα όρασης να κάνουν την καθημερινότητα τους πιο εύκολη με την μόνη διαφορά να βρίσκεται στην ανάγκη ύπαρξης εθελοντών διατεθειμένους να βοηθήσουν.

Τα άλλα δύο άρθρα προτείνουν λύσεις για να γίνει πιο εύκολη και ασφαλής η μετακίνηση των ατόμων με δυσκολίες όρασης, κάτι που αρχικά ήταν και στις προδιαγραφές της παρούσας εφαρμογής αλλά λόγω χρόνου δεν μπόρεσε να υλοποιηθεί.

Το πρώτο [14] περιγράφει μια λύση που χρησιμοποιεί συσκευή Kinect για την παροχή εικόνας και βάθους που επεξεργάζεται το σύστημα. Το Kinect είναι ένας αισθητήρας που κατασκευάστηκε από την Microsoft για να καταγράφει τις κινήσεις των παικτών για την κονσόλα Xbox 360 και αποτελείται από μια κάμερα και ένα υπέρυθρο αισθητήρα. Το φορητό σύστημα ενσωματώνει το Kinect στο μπροστά μέρος του χρήστη, περίπου στο ύψος του στήθους, κάτω από αυτό ένα επεξεργαστή και μια οθόνη αφής για να ρυθμίζεται το σύστημα και στην πλάτη ένα βαλιτσάκι με την μπαταρία που τα τροφοδοτεί. Χρησιμοποιώντας τα δεδομένα από το Kinect και κάνοντας τους απαραίτητους υπολογισμούς, δημιουργεί ένα μονοπάτι για τον χρήστη και με ηχητικά μηνύματα τον προτρέπει να κινηθεί μπροστά, να σταματήσει ή να στρίψει προς μια κατεύθυνση. Οι ερευνητές αυτού του άρθρου κατέληξαν στο συμπέρασμα ότι αν και τα αποτελέσματα ήταν υποσχόμενα, ο υπέρυθρος αισθητήρας ήταν αναξιόπιστος όταν ερχόταν σε επαφή με πολύ δυνατό φως.

Το δεύτερο άρθρο για την μετακίνηση ατόμων με οπτικές δυσκολίες [15] προτείνει λύση που αντί Kinect χρησιμοποιεί άλλη εμπορική κάμερα RGB-D. Η βασική διαφορά αυτής της εκδοχής είναι ότι το γενικό σύστημα που χρησιμοποιείται έχει πιο μικρό μέγεθος. Πιο συγκεκριμένα, είναι απλά μια κάμερα κρεμασμένη στο λαιμό του χρήστη με ένα φορητό υπολογιστή σε σακίδιο, για να κάνει τους υπολογισμούς. Άλλη διαφορά είναι ότι τα ηχητικά μηνύματα που παράγονται είναι κανονικές προτάσεις αντί μπιπς στο ανάλογο αυτί και ότι ο αλγόριθμος που χρησιμοποιείται επεξεργάζεται point-clouds αντί depth-map. Τέλος, οι ερευνητές αναφέρουν ότι είχαν καλύτερα αποτελέσματα με πολύ μεγάλη ακρίβεια αλλά αξίζει να σημειωθεί ότι όλα τα αποτελέσματα τους ήταν σε κλειστό χώρο επομένως, δεν είχαν να αντιμετωπίσουν το πρόβλημα του υψηλού φωτισμού.

Κεφάλαιο 3

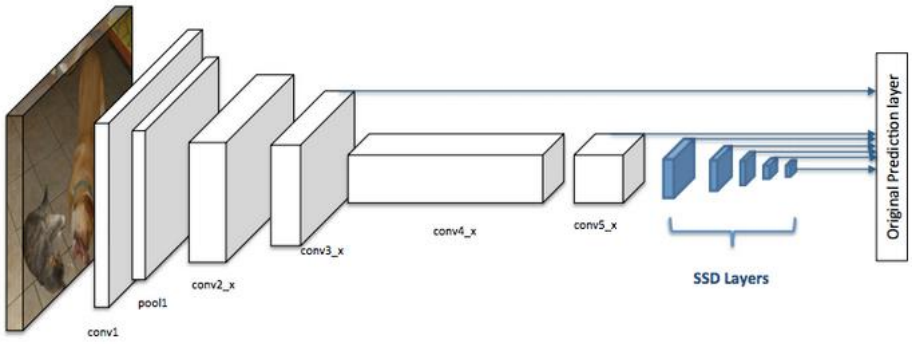
Μεθοδολογία

3.1 Αλγόριθμος Single-Shot MultiBox Detector

Ο αλγόριθμος ανίχνευσης αντικειμένων που θα χρησιμοποιηθεί για την υλοποίηση της εφαρμογής είναι ο Single-Shot MultiBox Detector ή αλλιώς SSD[16]. Από την ονομασία του γίνονται κατανοητές οι κύριες λειτουργίες του αλγόριθμου: Single shot, γιατί η ανίχνευση πραγματοποιείται με ένα πέρασμα του δικτύου, MultiBox, γιατί χρησιμοποιεί την ομώνυμη τεχνική παλινδρόμησης οριοθέτησης κουτιού (bounding box regression) [17] και Detector, γιατί χρησιμοποιείται για ανίχνευση αντικειμένων. Ο αλγόριθμος παίρνει σαν είσοδο μια εικόνα και μετά το πέρασμα της από το νευρωνικό δίκτυο επιστρέφει μια λίστα από εξόδους για τα αντικείμενα που ανιχνευτήκαν. Οι έξοδοι αυτοί περιλαμβάνουν την σιγουριά ύπαρξης αντικειμένου, το όνομα της κλάσης που ανιχνεύτηκε, το πλάτος και το ύψος και τις συντεταγμένες του κουτιού πάνω στην εικόνα. Έπειτα, τα κουτιά σχεδιάζονται στην εικόνα και συγκρίνονται με το ground truth για να υπολογιστεί η ακρίβεια της ανίχνευσης.

Το δίκτυο που υλοποιεί τον αλγόριθμο χωρίζεται σε δύο κύρια μέρη. Το πρώτο μέρος είναι συνήθως ένα προ εκπαιδευμένο συνελικτικό δίκτυο που χρησιμοποιείται σαν feature extractor. Στο άρθρο που περιγράφεται ο αλγόριθμος [16] χρησιμοποιήθηκε δίκτυο VGG16 [18] εκπαιδευμένο στο dataset ImageNet [19], του οποίου αφαιρέθηκε το classification layer για να λειτουργεί μόνο σαν εξαγωγέας χαρακτηριστικών. Στην

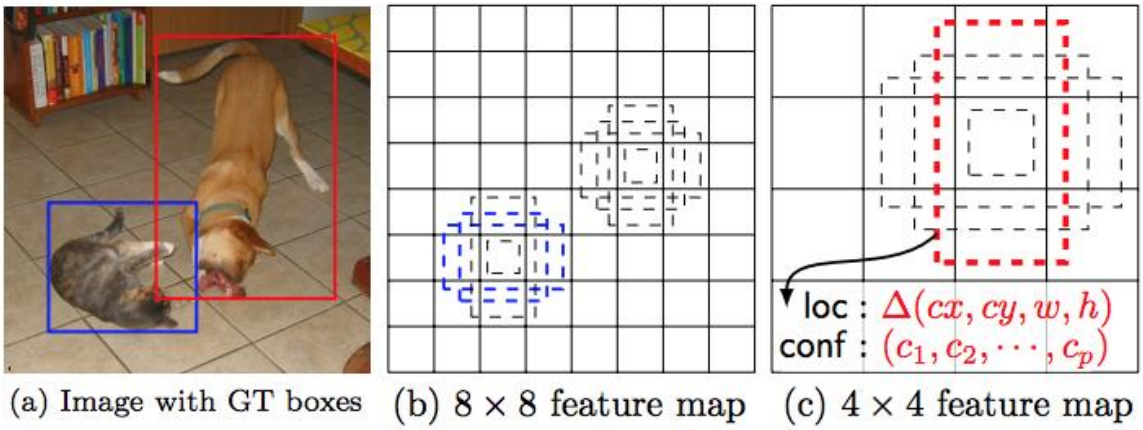
παρούσα υλοποίηση το πρώτο μέρος του δικτύου θα χρησιμοποιεί τον εξαγωγέα MobileNet [20], εκπαιδευμένο στο dataset COCO [11], για να είναι βελτιστοποιημένο για χρήση σε κινητή συσκευή. Στο δεύτερο μέρος ακολουθούν convolutional layers που υλοποιούν τον αλγόριθμο και ερμηνεύουν τις εξόδους σαν bounding boxes πάνω στην εικόνα.



Σχήμα 3.1 Αρχιτεκτονική νευρωνικού δικτύου με τον αλγόριθμο SSD. Τα άσπρα στρώματα ανήκουν στον εξαγωγέα χαρακτηριστικών και τα μπλε στο αλγόριθμο SSD [21]

Ο αλγόριθμος SSD, αντί να χρησιμοποιεί τεχνικές κινητού παραθύρου, όπως άλλες υλοποιήσεις, διαιρεί την εικόνα σε ένα πλέγμα και κάθε κελί του πλέγματος είναι υπεύθυνο για την ανίχνευση αντικειμένων στην περιοχή που του αναλογεί. Ανίχνευση αντικειμένων σημαίνει την πρόβλεψη της κλάσης και της θέσης ενός αντικειμένου εντός αυτής της περιοχής. Εάν δεν υπάρχει κάποιο αντικείμενο, το θεωρεί ως κλάση φόντου και η τοποθεσία αγνοείται. Για κάθε κελί του πλέγματος (grid cell) αναθέτεται ένας αριθμός κουτιών αγκίστρωσης (anchor boxes) τα οποία έχουν προκαθορισμένο μέγεθος και σχήμα μέσα στο κουτί πλέγματος. Ο αλγόριθμος για την εκπαίδευση χρησιμοποιεί μια μέθοδο αντιστοίχισης για να βρει πιο anchor box επικαλύπτει κάθε ένα από τα ground truths της εικόνας εκπαίδευσης. Ουσιαστικά, επιλέγει το κουτί με την μεγαλύτερη επικάλυψη και χρησιμοποιεί την προβλεπόμενη κλάση και την τοποθεσία για την εκπαίδευση. Αυτή η ιδιότητα χρησιμοποιείται για την εκπαίδευση του δικτύου και για την πρόβλεψη των αντικειμένων που εντοπίστηκαν καθώς και την τοποθέτηση τους μετά την εκπαίδευση του δικτύου. Μεταξύ των επιπέδων του δικτύου αλλάζει ο αριθμός των τομών του grid και κατ' επέκταση το μέγεθος των anchor boxes με αποτέλεσμα να ανιχνεύονται αντικείμενα διαφόρων μεγεθών και σχημάτων. Δεδομένου του μεγάλου

αριθμού κουτιών που δημιουργούνται κατά την προώθηση της εικόνας μέσα στο δίκτυο, είναι αναγκαίο να διαγραφτεί ένας μεγάλος αριθμός από αυτά με την τεχνική που ονομάζεται non-maximum suppression. Σε αυτό το επίπεδο, που είναι το τελευταίο από τα επίπεδα του SSD, τα κουτιά που ανιχνεύονται με confidence κάτω από το ορισμένο threshold για ανίχνευση διαγράφονται. Στη συνέχεια, επιλέγεται το bounding box με το μεγαλύτερο confidence και όλα τα κουτιά που έχουν ποσοστό επικάλυψης μεγαλύτερο από 50% με αυτό που επιλεκτικέ, διαγράφονται. Τα βήματα αυτά επαναλαμβάνονται για όλα τα εναπομείναντα κουτιά. Με αυτό τον τρόπο ο αλγόριθμος εγγυάται ότι θα επιστρέψει μόνο ένα κουτί για κάθε αντικείμενο.



Σχήμα 3.2 Παράδειγμα ανίχνευσης του αλγορίθμου. Η γάτα ανιχνεύεται από πλέγμα διαστάσεων 8x8 ενώ ο σκύλος από πλέγμα 4x4 γιατί έχει μεγαλύτερο μέγεθος.

3.2 TensorFlow Lite & MobileNets

Η TensorFlow είναι μια βιβλιοθήκη ανοικτού κώδικα που ανέπτυξε η Google για μηχανική μάθηση. Αρχικά, προοριζόταν για εσωτερική χρήση στην εταιρεία, σαν μαθηματική βιβλιοθήκη για πράξεις αυτόματης παραγωγισής και ροών δεδομένων, αλλά βρήκε χρήση στην εκπαίδευση και την εκτέλεση νευρωνικών δικτιών και το 2017 έγινε διαθέσιμη για το κοινό. Σήμερα, η βιβλιοθήκη χρησιμοποιείται ευρέως από την Google στις εφαρμογές της, όπως στο Google photos για αυτόματη επεξεργασία και στο Voice assistant για την αναγνώριση φωνής, αλλά και από άλλες μεγάλες εταιρείες όπως την Nvidia και Intel για την βελτίωση των εργασιών τους. Η βιβλιοθήκη καθιστά εύκολη την

ανάπτυξη και χρήση υπολογισμών μηχανικής μάθησης και διατίθεται για διάφορα περιβάλλοντα όπως Windows, Linux και macOS, αλλά και σε κινητά με την έκδοση TensorFlow Lite και σε javascript με την έκδοση TensorFlow.js . Στα πλαίσια αυτής της διπλωματικής εργασίας θα χρησιμοποιηθεί η βιβλιοθήκη TensorFlow Lite για την χρήση νευρωνικού δικτυού για αναγνώριση αντικειμένων με κινητή συσκευή.

Τα MobileNets [20] είναι μια οικογένεια από μοντέλα μηχανικής όρασης κατασκευασμένα για την TensorFlow που μεγιστοποιούν την ακρίβεια της ανίχνευσης χρησιμοποιώντας τους περιορισμένους πόρους που διαθέτει μια κινητή συσκευή ή ένα ενσωματωμένο (embedded) σύστημα [22]. Μπορούν επίσης να χρησιμοποιηθούν και σαν εξαγωγής χαρακτηριστικών (Feature extractor) για συνελικτικά νευρωνικά δίκτυα που προορίζονται να χρησιμοποιηθούν σε συσκευές με μικρή υπολογιστική δύναμη. Για την υλοποίηση της εφαρμογής για το κινητό τηλέφωνο θα χρησιμοποιηθεί MobileNet νευρωνικό δίκτυο για αναγνώριση αντικειμένων.

3.3 Σύνθεση Ομιλίας

Η σύνθεση ομιλίας ή αλλιώς Text to Speech αναφέρεται στην τεχνική παραγωγής ανθρώπινης ομιλίας και εφαρμόζεται σε υλικό ή λογισμικό. Ένα σύστημα Text to Speech (TTS) έχει την δυνατότητα να μετατρέψει ένα κείμενο σε φωνητικό μήνυμα – ομιλία. Τα συστήματα αυτά, χρησιμοποιούν βάσεις δεδομένων που περιλαμβάνουν ηχογραφημένους ήχους γραμμάτων ή δίφωνων και ανάλογα με την είσοδο του χρήστη γίνεται μια συνένωση των αντίστοιχων ηχητικών κομματιών που ακούει ο χρήστης. Η αποθήκευση ολόκληρων λέξεων μπορεί να βελτιώσει την ποιότητα της ομιλίας αλλά έχει μεγαλύτερο κόστος σε μνήμη. Μια άλλη προσέγγιση είναι υλοποίηση συστήματος που να μιμείται την φωνητική οδό και χαρακτηριστικά της φωνής και παράγει εξολοκλήρου τεχνητή φωνή. Η ποιότητα της συνθετικής ομιλίας κρίνεται βάση της ομοιότητας του με την ανθρώπινη φωνή και με την ικανότητά της να γίνεται κατανοητή. Μια καλή υλοποίηση σύνθεσης ομιλίας επιτρέπει σε άτομα με προβλήματα όρασης ή με δυσκολίες στην ανάγνωση να ακούσουν κάποιο γραπτό κείμενο. Πολλά λειτουργικά συστήματα υιοθέτησαν τέτοιες λειτουργίες από τη δεκαετία του 1990 και όσο βελτιώνονται οι υλοποιήσεις γίνονται όλο και πιο συνηθισμένες στην καθημερινότητα μας. Virtual assistants, όπως την Siri και την Alexa, μπορούν πλέον να κάνουν τηλεφωνικές κλίσεις,

στις οποίες ο παραλήπτης δυσκολεύεται να καταλάβει ότι δεν μιλά με πραγματικό άνθρωπο.

Το 2009 η Google πρόσθεσε στις Android συσκευές την υλοποίηση της για το Text to Speech, δίνοντας έτσι τη δυνατότητα στις συσκευές να μιλήσουν στο χρήστη σε διάφορες γλώσσες [23]. Αυτή η υλοποίηση χρησιμοποιείται ευρέως από το Google Assistant και το Android σαν ρύθμιση προσβασιμότητας (accessibility setting) και δίνει στον χρήστη την επιλογή να ακούσει κάποιο εισερχόμενο μήνυμα ή τα περιεχόμενα της οθόνης. Για την συγκεκριμένη υλοποίηση θα χρησιμοποιηθεί η native εντολή για το Text to Speech στο Android τηλέφωνο που δόθηκε, για την περιγραφή του αντικείμενου ως επίσης και της απόστασης του.

3.4 Κινητό Sony Xperia XZ1 με TOF κάμερα

Για την υλοποίηση της εφαρμογής, σε αυτή την διπλωματική εργασία, θα χρησιμοποιηθεί ένα πρωτότυπο κινητό τηλέφωνο που είναι εξοπλισμένο με TOF κάμερα. Το μοντέλο του κινητού είναι Sony Xperia XZ1, με επεξεργαστή Snapdragon 835 και κάρτα γραφικών Adreno 540. Αξίζει να σημειωθεί ότι το τηλέφωνο τέθηκε σε κυκλοφορία το 2017 και προφανώς, δεν έχει την υπολογιστική ισχύ ενός τηλεφώνου flagship του 2020. Αυτό αποτελεί μια πρόκληση που πρέπει να ξεπεραστεί και θα έχει ως αποτέλεσμα η εφαρμογή της παρούσας διπλωματικής εργασίας να μπορεί να λειτουργήσει σε κινητά χαμηλότερου κόστους.

Αυτό που κάνει το κινητό να ξεχωρίζει από τα υπόλοιπα του είδους του, είναι η κάμερα TOF που διαθέτει. Έγινε ειδική κατασκευή από την Sony για να προωθήσει την έρευνα τον TOF καμερών με την ελπίδα κάποια μέρα να γίνουν βασικό μέρος στα κινητά τηλέφωνα. Ήδη, τα τηλέφωνα ναυαρχίδες στης Samsung για το 2020 διαθέτουν TOF κάμερες, με κατασκευαστή την Sony. Η TOF κάμερα που θα χρησιμοποιηθεί στην εργασία έχει ανάλυση 240x180 και μπορεί να υπολογίσει την απόσταση τον αντικειμένων στην σκήνη σε πραγματικό χρόνο με 30fps. Επίσης, με το SDK που δόθηκε με το κινητό είναι δυνατή η εύρεση της απόστασης κάθε σημείου στην οθόνη, χωρίς ιδιαίτερο υπολογιστικό κόστος.

Κεφάλαιο 4

Υλοποίηση

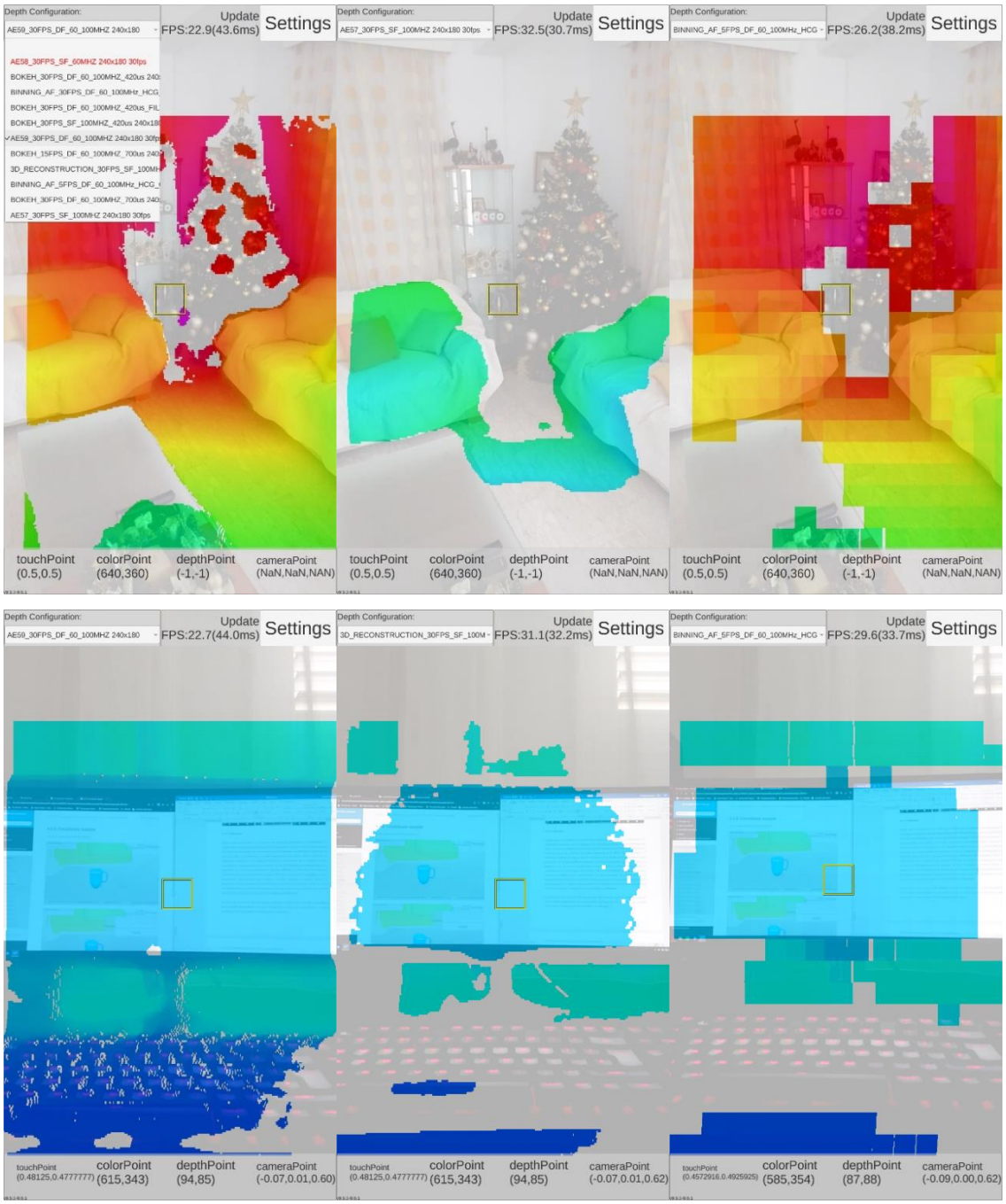
4.1 Εύρεση απόστασης

Πρώτο βήμα στην υλοποίηση της παρούσας εφαρμογής ήταν να ελεγχθεί κατά πόσο υπάρχει η δυνατότητα λήψης της αληθινής απόστασης οποιουδήποτε σημείου στην οθόνη, χρησιμοποιώντας την TOF κάμερα του κινητού. Για να επιτευχθεί αυτό, υλοποιήθηκε μια εφαρμογή η οποία εμφανίζει στην οθόνη την απόσταση που επιστρέφει το sensor για το σημείο που επέλεξε ο χρήστης πάνω στην οθόνη. Η εφαρμογή έπρεπε να δημιουργηθεί στο Unity επειδή το SDK που δόθηκε μαζί με το κινητό είναι γραμμένο αποκλειστικά για την συγκεκριμένη μηχανή παιχνιδιού. Έπειτα, μετρήθηκε η απόσταση από το κινητό σε διάφορα αντικείμενα που επιλέχθηκαν στην οθόνη για να επαληθευτεί ότι η τιμή που αναγράφεται στην οθόνη είναι όντως ορθή. Σε περιπτώσεις που η επιφάνεια δεν αντανακλά το λέιζερ της TOF κάμερας και δεν ήταν δυνατή η λήψη απόστασης, το πρόγραμμα επιστρέφει τιμή -1.

4.1.1 Calibration

Αρχικά, πρέπει να ρυθμιστεί το κινητό για να υπολογίζει με ακρίβεια τις αποστάσεις. Το calibration της κάμερας γίνεται με εφαρμογή που λήφθηκε μαζί με το κινητό της Sony στην οποία πολύ απλά ο χρήστης φωτογραφίζει μια εκτυπωμένη εικόνα σκακιέρας και αυτόματα υπολογίζεται η απόσταση βάση του μεγέθους του κάθε τετραγώνου στην οθόνη. Μόλις τελειώσει το calibration αποθηκεύεται στην μνήμη του τηλεφώνου ένας πίνακας με τις παραμέτρους που πράχτηκαν για να έχουμε ορθά αποτελέσματα. Αυτές οι παράμετροι καταχωρούνται στις εφαρμογές που χρησιμοποιούν την κάμερα TOF για να μην χρειάζεται να γίνεται calibraton σε κάθε εκτέλεση. Αξίζει να σημειωθεί ότι αν δύο κινητά έχουν ίδιες κάμερες, TOF δεν είναι αναγκαίο να παράξουν τις ίδιες παραμέτρους βαθμονόμησης, αφού μπορούν να έχουν διαφορές σε επίπεδο υλικού. Οι εφαρμογές που υλοποιήθηκαν στα πλαίσια αυτής της ατομικής διπλωματικής εργασίας έγιναν με παραμέτρους που αφορούν συγκεκριμένο κινητό τηλέφωνο και για να δουλέψουν σε αλλά κινητά που υποστηρίζονται πρέπει να επαναληφθεί το βήμα του calibration.

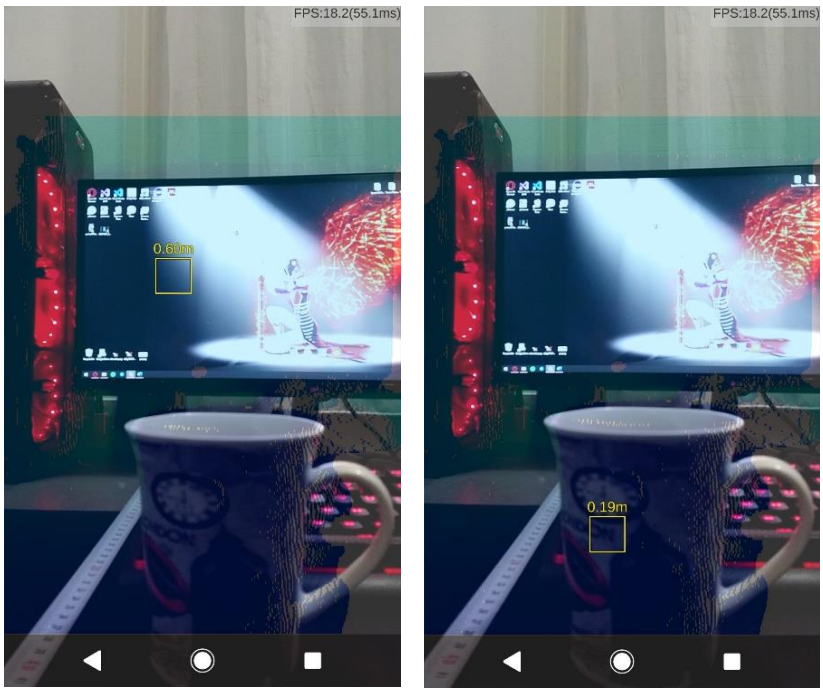
Ανάλογα με το μοντέλο της TOF κάμερας που έχει το κάθε κινητό το SDK περιλαμβάνει κάποια depth configurations τα οποία αλλάζουν την ευστοχία, την απόσταση και τον ρυθμό ανανέωσης των πληροφορίων που επιστρέφει ο αισθητήρας. Αφού δοκιμάστηκαν όλες οι επιλογές, κρίθηκε ότι η επιλογή AE59 είναι η βέλτιστη για την συγκεκριμένη κάμερα, ώστε να υπολογίζεται και η απόσταση για την χρήση της εφαρμογής. Όπως φαίνεται και στις εικόνες στο Σχήμα 4.1 πιο κάτω, εκτός από μεγαλύτερη ακρίβεια, η συγκεκριμένη επιλογή έχει και λιγότερα σημεία που δεν έχουν απόσταση. Στις εικόνες, τα χρώματα υποδηλώνουν την απόσταση των αντικειμένων με τις αποχρώσεις του μπλε να είναι τα κοντινά σημεία, του κόκκινου τα απόμακρα ενώ στα σημεία χωρίς χρώμα σημαίνει δεν έχουμε απόσταση. Όπως και με το calibration, σε περίπτωση που η εφαρμογή εκτελείται με άλλη TOF κάμερα θα πρέπει να βεβαιωθούμε ότι η επιλογή AE59 υπάρχει και επιστρέφει ικανοποιητικά αποτελέσματα ή αν δεν υπάρχει να αντικατασταθεί.



Σχήμα 4.1 Φωτογραφίες από την εκτέλεση της εφαρμογής Coordinate που δόθηκε με το κινητό. Στις δύο αριστερά φωτογραφίες εμφανίζεται το αποτέλεσμα με την επιλογή “AE59”, στις δεξιά το αποτέλεσμα με το preset “Binning”. Στην μεσαία πάνω φωτογραφία χρησιμοποιήθηκε η επιλογή AE57 και στην μεσαία κάτω η 3D-Reconstruction. Το χρώμα υποδηλώνει την απόσταση με το μπλε να σημαίνει μικρή απόσταση ενώ οι αποχρώσεις του κόκκινου και μωβ μεγάλη. Σημεία χωρίς χρώμα δείχνουν ότι η κάμερα δεν ανίχνευσε κάποια απόσταση.

4.1.2 Υλοποίηση εφαρμογής για έλεγχο απόστασης

Η υλοποίηση της εφαρμογής της απόστασης βασίστηκε σε δείγματα κώδικα που λήφθηκαν μαζί με το κινητό και συγκεκριμένα στη σκηνή “Coordinate”. Βάση αυτής, κάνοντας τις απαραίτητες αλλαγές που χρειάστηκαν, υλοποιήθηκε μια εφαρμογή που παρουσιάζει στην οθόνη το colormap των αποστάσεων που επιστρέφει η κάμερα και ανάλογα με το σημείο που επιλέγει ο χρήστης εμφανίζει την αληθινή απόσταση τηλεφώνου και αντικειμένου σε μέτρα. Πιο κάτω, στο Σχήμα 4.2, παρουσιάζεται ένα παράδειγμα εκτέλεσης της εφαρμογής για δύο σημεία στην οθόνη. Το κίτρινο τετράγωνο είναι το σημείο που επέλεξε ο χρήστης και πάνω από αυτό αναγράφεται η απόσταση του. Στην αριστερά εικόνα η απόσταση από το σημείο στην οθόνη του υπολογιστή είναι 0,60 μέτρα και επαληθευτικέ με το μέτρο που βρίσκεται πάνω στο γραφείο. Στην δεξιά εικόνα η απόσταση από το ποτήρι είναι 19 εκατοστόμετρα και είναι δυνατή η επαλήθευση με το μέτρο που φαίνεται στη φωτογραφία.



Σχήμα 4.2 Φωτογραφίες από την εκτέλεση της εφαρμογής για υπολογισμό απόστασης. Το κίτρινο κουτί είναι το σημείο που επιλέχθηκε από τον χρήστη και πάνω από αυτό αναγράφεται η απόσταση που μετρήθηκε.

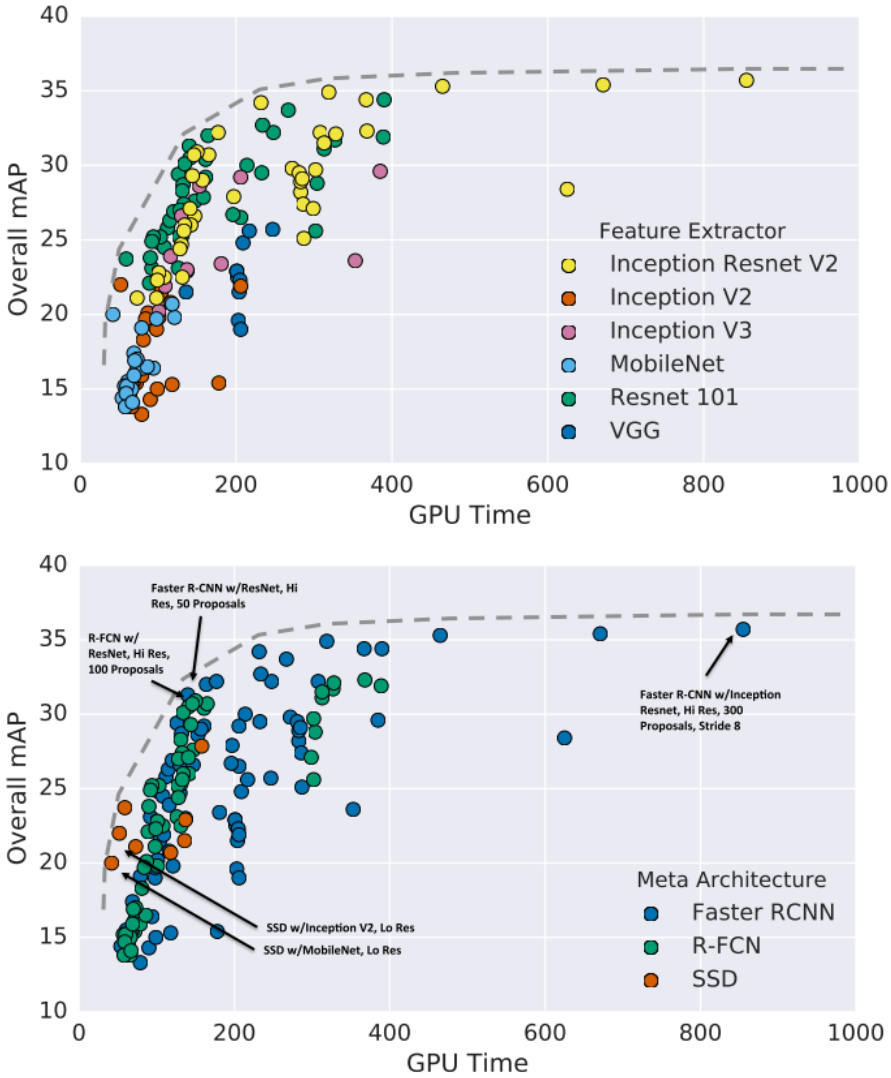
4.2 Αναγνώριση αντικειμένων εικόνας

Το επόμενο βήμα για την υλοποίηση της τελικής εφαρμογής ήταν η επιτυχία της ανίχνευσης αντικειμένων χρησιμοποιώντας την υπολογιστική ισχύ του κινητού. Οι δυσκολίες που έπρεπε να ξεπεραστούν σε αυτό το σημείο ήταν ότι η εφαρμογή έπρεπε να υλοποιηθεί στην μηχανή παιχνιδιών Unity, άρα για να χρησιμοποιηθούν βιβλιοθήκες όπως TensorFlow χρειάζονταν επιπλέων plugins. Επίσης, το γεγονός ότι το κινητό δεν έχει μεγάλη υπολογιστική δύναμη σημαίνει ότι για να υπάρχει ικανοποιητικός αριθμός ανιχνεύσεων σε κάθε frame πρέπει να βρεθεί πολύ γρήγορο νευρωνικό δίκτυο.

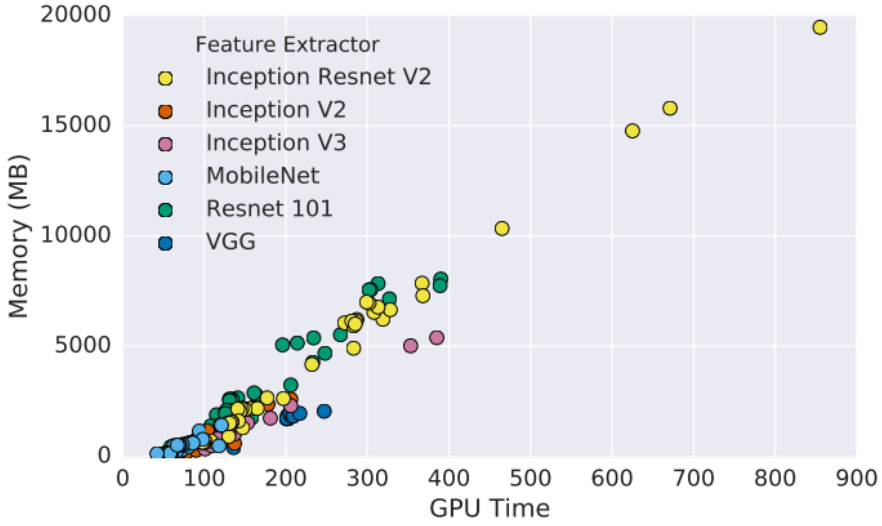
4.2.1 Επιλογή αλγόριθμου αναγνώρισης

Ήταν επιθυμητό η εφαρμογή να μπορεί να ανιχνεύει σε πραγματικό χρόνο αντικείμενα χρησιμοποιώντας τις υπολογιστικές μονάδες που διαθέτει κάποιο κινητό τηλέφωνο. Λόγω της χαμηλής υπολογιστικής ισχύς των κινητών συσκευών, θεωρήθηκε προτεραιότητα το performance έναντι της ακρίβειας της αναγνώρισης στην επιλογή για τον κατάλληλο αλγόριθμο αναγνώρισης αντικειμένων.

Στο άρθρο “Speed/accuracy trade-offs for modern convolutional object detectors” [24] γίνεται σύγκριση τριών κύριων αλγόριθμων ανίχνευσης αντικειμένων με έξι διαφορετικά Feature extractors, ως προς την ακρίβεια, τον χρόνο εκτέλεσης και την μνήμη που χρειάστηκε η ανίχνευση. Οι αλγόριθμοι που συγκρίνονται στο πιο πάνω paper είναι οι Faster R-CNN [25], R-FCN [26] και SSD [16] ενώ τα feature extractors που χρησιμοποιούνται είναι τα VGG-16 [18], Resnet-101 [27], Inception v2 [28], Inception v3 [29], Inception Resnet (v2) [30] και MobileNet [20]. Τα αποτελέσματα της σύγκρισης εμφανίζονται πιο κάτω όπως αναγράφονται στην έρευνα [24].



Σχήμα 4.3 Αποτελέσματα της έρευνας [24] όπως εμφανίζονται σε αυτήν. Οι γραφικές παραστάσεις δείχνουν το mAP κάθε εκτέλεσης ως προς τον χρόνο εκτέλεσης και είναι χρωματισμένα βάση του feature extractor (πάνω) και του αλγόριθμου ανίχνευσης (κάτω).



Σχήμα 4.4 Αποτελέσματα της έρευνας [24] όπως εμφανίζονται σε αυτήν. Στην γραφική παράσταση εμφανίζεται η μνήμη και ο χρόνος που χρειάστηκε η κάθε εκτέλεση, χρωματισμένα βάση το feature extractor της.

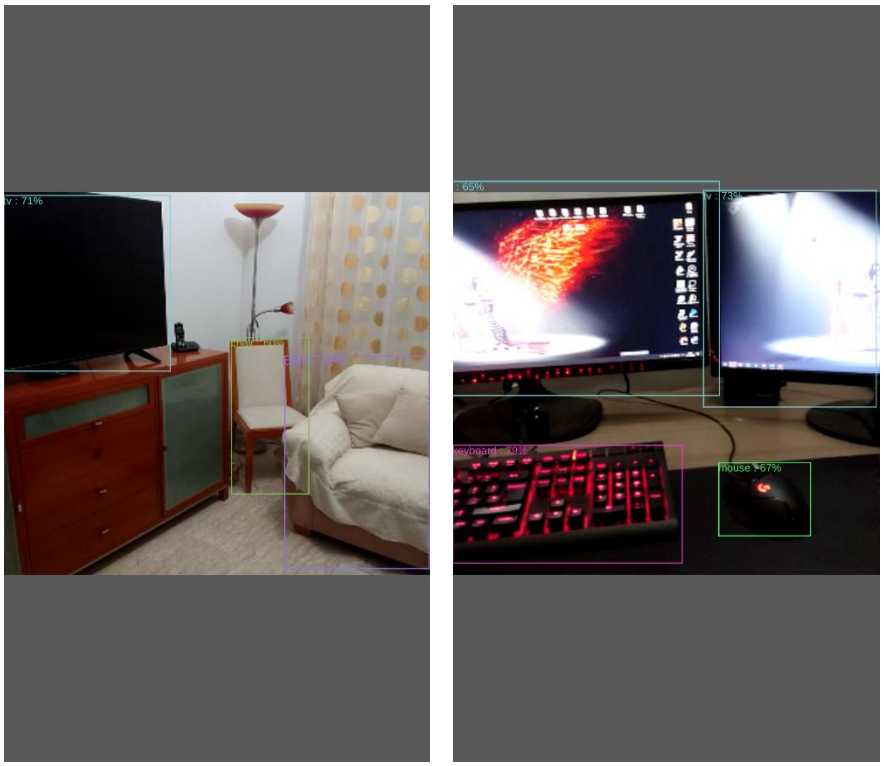
Όπως φαίνεται από τα αποτελέσματα αυτής της έρευνάς το MobileNet χρειάζεται πιο λίγη μνήμη και χρόνο εκτέλεσης από τα υπόλοιπα feature extractors με τον μέσο όρο ευστοχίας ανίχνευσης να φτάνει μέχρι το 22%. Όσον αφορά τους αλγόριθμους ανίχνευσης, από την πρώτη γραφική παράσταση φαίνεται ότι ο SSD έχει μεγαλύτερη ακρίβεια σε ίδιο χρόνο με τους υπόλοιπους αλγόριθμους και όταν συνδυαστεί με το MobileNet έχει την πιο γρήγορη ανίχνευση με ένα ικανοποιητικό mAP=20 που είναι κοντά στο μέγιστο αποτέλεσμα που παρατηρήθηκε στις εκτελέσεις με MobileNet. Από αυτή την ανάλυση συμπεραίνεται ότι, ο συνδυασμός του αλγορίθμου SDD με το feature extractor MobileNet ικανοποιεί τις ανάγκες του προβλήματος για γρήγορη εκτέλεση που χρησιμοποιεί όσο το λιγότερη μνήμη με ικανοποιητική ευστοχία στην ανίχνευση. Η απόφαση επομένως της χρήσης SSD με MobileNet υποστηρίζεται περαιτέρω από την έρευνα “Real-Time Object Detection Using Pre-Trained Deep Learning Models MobileNet-SSD” [31], στην οποία οι ερευνητές υλοποίησαν ένα πρόγραμμα για ανίχνευση αντικειμένων που μπορεί να χρησιμοποιηθεί σε embedded συσκευές. Οι ερευνητές είχαν πολύ καλά αποτελέσματα με τη χρήση ίδιων αλγόριθμων με την εφαρμογή που παρουσιάζεται στην παρούσα διπλωματική εργασία.

4.2.2 Υλοποίηση εφαρμογής για αναγνώριση αντικειμένων

Αφού επιλέγηκε ο αλγόριθμος αναγνώρισης για χρήση στην εφαρμογή έπρεπε να διερευνηθεί αν όντως υπήρχε η επιθυμητή απόδοση. Επιτεύχθηκε η υλοποίηση μιας εφαρμογής στο Unity η οποία περνά από το βαθύ νευρωνικό δίκτυο κάθε frame που λαμβάνει από την κάμερα και εμφανίζει στην οθόνη τα bounding boxes γύρω από τα αντικείμενα που αναγνωρίζει. Σε αυτή την υλοποίηση χρησιμοποιήθηκε ένα προ-εκπαιδευμένο συνελικτικό νευρωνικό δίκτυο [32], το οποίο εκπαιδεύτηκε να αναγνωρίζει τις κλάσεις του COCO [11] dataset με τον αλγόριθμο SSD-MobileNet, χρησιμοποιώντας την βιβλιοθήκη TensorFlow Lite για να είναι optimized για κινητές συσκευές. Λόγω του ότι η μηχανή Unity δεν έχει την δυνατότητα να χρησιμοποιήσει μοντέλα της TensorFlow natively χρησιμοποιήθηκε ένα πειραματικό plugin [33] που μεταφράζει τον κώδικα της TensorFlow Lite που είναι σε γλώσσα C, σε C# που αναγνωρίζει το Unity. Τέλος, για την υλοποίηση της μεθόδου που σχεδιάζει τα κουτιά γύρω από το κάθε αντικείμενο που ανιχνεύτηκε για την εφαρμογή που παρουσιάζεται, δόθηκε έμπνευση από μια παρόμοια υλοποίηση από το GitHub [34] με προσθήκη διαφορετικών χρωμάτων των κουτιών για κάθε κλάση αντικειμένων.

Η συγκεκριμένη υλοποίηση είχε πολύ υποσχόμενα αποτελέσματα, αφού ανιχνεύει τα πλείστα αντικείμενα στην οθόνη σε πραγματικό χρόνο με πολλά detections per frame. Το νευρωνικό δίκτυο που χρησιμοποιήθηκε μπορεί να ξεχωρίσει αντικείμενα σε 80 διαφορετικές κλάσεις που πιθανόν να αντικρίσει ο χρήστης στην καθημερινότητα του, με 24 mAP, όπως αναγράφεται στην ιστοσελίδα από την οποία λήφθηκε [32]. Επειδή χρησιμοποιείται πολύ μικρό νευρωνικό δίκτυο η πιθανότητα για λάθος ανίχνευση είναι αυξημένη, γι’ αυτό ορίζεται ως threshold ανίχνευσης 60% confidence. Έτσι, ανιχνεύεται μικρότερος αριθμός αντικειμένων σε κάθε frame αλλά η πιθανότητα για λάθος ανίχνευση είναι μειωμένη. Εντέλει, η επιλογή χρήσης του MobileNet-SSD ήταν σωστή εφόσον επιτεύχθηκε ικανοποιητική ανίχνευση, σε πραγματικό χρόνο, χρησιμοποιώντας μόνο την υπολογιστική ισχύ του κινητού που δόθηκε για την εργασία. Αξίζει να σημειωθεί ότι αυτή η εφαρμογή επειδή δεν συσχετίζεται με την TOF κάμερα μπορεί να εκτελεστεί σε οποιαδήποτε Android συσκευή που έχει μια κάμερα, με την ταχύτητα της ανίχνευσης να εξαρτάται από την υπολογιστική δύναμη της συσκευής.

Πιο κάτω, στο Σχήμα 4.5, φαίνονται φωτογραφίες από την εκτέλεση αυτής της εφαρμογής. Στην αριστερά εικόνα ανιχνεύονται η τηλεόραση με 71%, η καρέκλα με 60% και ο καναπές με 62%. Στην δεξιά εικόνα η εφαρμογή βρίσκει τις δύο οθόνες του υπολογιστή και τις ονομάζει τηλεοράσεις με 65% και 73% confidence, το πληκτρολόγιο με 79% και το ποντίκι με 67%.



Σχήμα 4.5 Παράδειγμα εκτέλεσης της εφαρμογής για ανίχνευσης αντικειμένων με τη βιβλιοθήκη TensorFlow Lite

4.3 Συνδυασμός των δύο εφαρμογών

Σε αυτό το σημείο της ανάπτυξης της εφαρμογής υπολόγιζα να συνδυάσω τις δύο υλοποιήσεις που δημιουργήθηκαν και να δημιουργήσω μια εφαρμογή που μπορεί να ανιχνεύει αντικείμενα σε πραγματικό χρόνο και παράλληλα να ξέρει την απόσταση τους χρησιμοποιώντας την κάμερα TOF. Η ιδέα ήταν για κάθε αντικείμενο που ανιχνεύεται στην οθόνη να βρίσκεται η απόσταση του από το κινητό και να χρησιμοποιείται αυτή την πληροφορία για να περιγράφει καλύτερα ο χώρος. Η μέθοδος που επιστρέφει την απόσταση έχει για παράμετρο ένα σημείο στην οθόνη για αυτό επιλέχτηκε να δίνεται το

κέντρο του κάθε bounding box, με την ελπίδα ότι ακόμα και όταν το bounding box δεν περιτριγυρίζει σωστά το αντικείμενο το κέντρο του θα βρίσκετέ πάνω του.

4.3.1 Προβλήματα που αντιμετωπίστηκαν

Στην προσπάθεια συνδυασμού των δύο εφαρμογών έγινε αντιληπτό ότι το SDK για την TOF κάμερα έκανε conflict με την βιβλιοθήκη για την αναγνώριση αντικειμένων. Συγκεκριμένα, για να δουλέψει η βιβλιοθήκη, χρειαζόταν η επιλογή για το scripting backend να είναι "IL2CPP" αντί "Mono" κάτι που προκαλούσε τα assets του SDK να μην δουλεύουν. Μετά από επικοινωνία με τη Sony, εξηγήθηκε ότι η έκδοση του SDK που χρησιμοποιήθηκε ήταν παλιά και δεν υποστηρίζει την επιλογή "IL2CPP". Επιπρόσθετα, ενημέρωσαν ότι μπορούν να αποστείλουν την καινούρια έκδοση, η οποία όμως δεν θα λειτουργούσε με το τηλέφωνο που χρησιμοποιείτο μέχρι στιγμής. Η καινούρια έκδοση αφορά τηλέφωνα που είναι εξοπλισμένα με τις τελευταίες TOF κάμερες της SONY όπως για παράδειγμα το Samsung S20 Plus. Η συνομιλία με την Sony ξεκίνησε μέσα Οκτωβρίου και η καινούρια έκδοση λήφθηκε στις αρχές Δεκεμβρίου. Εξαιτίας των μέτρων που επικρατούσαν για τη πανδημία και του λιγοστού χρόνου που παρέμενε, η ανάπτυξη εφαρμογής με την καινούρια έκδοση ήταν αδύνατη.

4.3.2 Υλοποίηση εφαρμογής με TensorFlowSharp

Στον χρόνο αναμονής αναφορικά με την έκδοση του SDK, διερευνήθηκε ο τρόπος για ταυτόχρονη ανίχνευση και υπολογισμό απόστασης. Η λύση τελικά βρέθηκε με την βιβλιοθήκη TensorFlowSharp [35] και το πειραματικό plugin [36] που την υλοποιεί για την μηχανή παιχνιδιών Unity. Για τις μεθόδους που φορτώνουν το νευρωνικό δίκτυο και που σχεδιάζουν τα bounding boxes στην οθόνη δόθηκε έμπνευση από μια παρόμοια υλοποίηση στο GitHub [37]. Αφού υλοποιήθηκε η αναγνώριση, προστέθηκαν μεθόδοι για τον υπολογισμό της απόστασης και όλα τα απαραίτητα assets για να λειτουργεί η κάμερα TOF.

Το πρόβλημα με αυτή την βιβλιοθήκη, εκτός από το ότι οι δημιουργοί της σταμάτησαν να την υποστηρίζουν το 2018, είναι ότι δεν αναγνωρίζει μοντέλα της TensorFlow Lite. Ως εκ τούτου, χρησιμοποιήθηκε μοντέλο κατασκευασμένο για να εκτελείται σε κάρτα

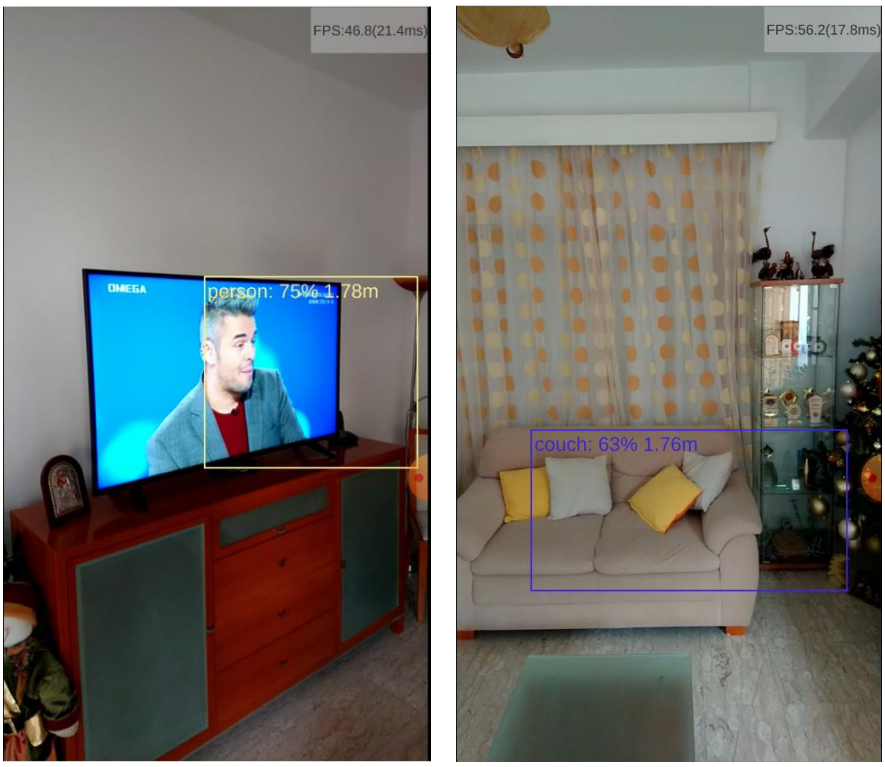
γραφικών ηλεκτρονικού υπολογιστή. Αυτό είχε ως αποτέλεσμα η κάθε ανίχνευση να χρειάζεται μερικά δευτερόλεπτα για να ολοκληρωθεί. Για να μειωθεί ο χρόνος ανίχνευσης, έγινε προσπάθεια ανεύρεσης ενός μοντέλου με λιγότερο υπολογιστικό κόστος από το κανονικό Mobile-SSD που χρησιμοποιήθηκε στην προηγούμενη υλοποίηση. Από τα προ-εκπαιδευμένα μοντέλα στη βιβλιογραφία [32], αυτό με τον πιο μικρό χρόνο εκτέλεσης ήταν το "ssd_mobilenet_v1_0.75_depth_coco" και το χρησιμοποιήθηκε για τις ανάγκες εκτέλεσης της παρούσας εφαρμογής. Η διαφορά αυτού του μοντέλου από το κανονικό είναι ότι έχει πολλαπλασιαστή βάθους 0,75 για το feature extractor και έχει mAP=18, έναντι του 24 που είχε το μοντέλο της TFlite. Αυτή η υλοποίηση εκτός από πιο αργή ανίχνευση έχει και πιο μεγάλο ποσοστό ανίχνευσης λάθους. Για threshold επιλέχτηκε για άλλη μια φορά το 60% διότι ναι μεν περιορίζει τις λάθος ανιχνεύσεις αλλά δε συνεχίζει να βρίσκει ένα ικανοποιητικό αριθμό αντικειμένων.

Με το μικρότερο μοντέλο που δόθηκε στην εφαρμογή επιτεύχθηκε η μείωση του χρόνου ανίχνευσης σε περίπου 2,5 δευτερόλεπτα ανά ανίχνευση. Αυτή η καθυστέρηση έχει ως αποτέλεσμα τα bounding boxes να εμφανίζονται εκεί που ήταν το αντικείμενο την ώρα που δόθηκε το frame για επεξεργασία, ακόμα και αν το αντικείμενο δεν υπάρχει πλέον στην οθόνη. Επίσης, παρατηρήθηκε ότι αν δίνονται όλα τα frames για επεξεργασία η εικόνα που βλέπει ο χρήστης γίνεται πολύ αργή και κολλάει το πρόγραμμα μέχρι να τελειώσει η κάθε ανίχνευση. Αυτό επιλύθηκε δίνοντας μόνο ένα frame για επεξεργασία κάθε φορά και μέσω παραλληλοποίησης συνεχίζει να δείχνει στην οθόνη το feed της κάμερας χωρίς αναμονή του αποτελέσματος της ανίχνευσης. Ουσιαστικά, η εφαρμογή παίρνει ένα frame από την κάμερα και ξεκινά να το επεξεργάζεται για να εντοπίσει αντικείμενα. Παράλληλα, συνεχίζει να δείχνει το camera feed και μόλις έχει το αποτέλεσμα της ανίχνευσης σχεδιάζει στην οθόνη τα κουτιά στο σημείο που τα βρήκε και αναγράφει την απόσταση που είχε το κέντρο τους την στιγμή που ξεκίνησε η ανίχνευση.

Σε σύγκριση με την υλοποίηση με την βιβλιοθήκη TFlite, το αποτέλεσμα δεν είναι ικανοποιητικό. Η ανίχνευση είναι πολύ αργή και με μικρότερη ακρίβεια. Όμως επειδή, ο κώδικας για την απόσταση λειτουργεί με την συγκεκριμένη υλοποίηση, το αποτέλεσμα θεωρήθηκε αρκετό και ως μια καλή ένδειξή για τη διαχείριση της απόστασης όταν υπάρχει το καινούριο SDK και πιο γρήγορη ανίχνευση αντικειμένων.

Για μελλοντική χρήση, είναι σημαντικό να κρατηθούν οι δύο εφαρμογές που αναπτύχθηκαν, αυτήν με τη TFlite χωρίς μέθοδο για απόσταση και αυτήν με την TFsharp με απόσταση, σαν σημεία αναφοράς. Η πρώτη για την ταχύτητα της ανίχνευσης και η δεύτερη για την διαχείριση της πληροφορίας της απόστασης.

Πιο κάτω, στο Σχήμα 4.5 εμφανίζονται εικόνες από την εκτέλεση αυτής της εφαρμογής με περιστροφή του τηλεφώνου με μικρή ταχύτητα προς τα δεξιά. Στην δεξιά εικόνα αναγνωρίζει τον άνθρωπο στην τηλεόραση με 75% βεβαιότητα αλλά δεν αναγνωρίζει την τηλεόραση. Στην αριστερή εικόνα αναγνωρίζει τον καναπέ με 63% σιγουριά και απόσταση 1,76 μέτρα που είναι πάρα πολύ κοντά στην αληθινή απόσταση. Και στις δύο περιπτώσεις το κουτί είναι σε λάθος σημείο, αλλά όταν έγινε επεξεργασία του βίντεο παρατηρήθηκε ότι 2 δευτερόλεπτα πριν τα αντικείμενα ήταν εκεί που τα ανιχνεύει η εφαρμογή.



Σχήμα 4.5 Παράδειγμα εκτέλεσης της εφαρμογής για ανίχνευσης αντικειμένων με τη βιβλιοθήκη TensorFlow Sharp

4.4 Υλοποίηση text to speech

Τελευταίο βήμα της παρούσας εργασίας είναι η μετάδοση της πληροφορίας στον χρήστη. Τα πολύχρωμα κουτιά που σχεδιάζονται στην οθόνη είναι πολύ καλά για έλεγχο της εφαρμογής, αλλά δεν εξυπηρετούν καθόλου τους χρήστες με προβλήματα όρασης, στους οποίους επιδιώκεται να δοθεί βοήθεια. Για να επιλυθεί αυτό, έγιναν σκέψεις που αφορούσαν τη λεκτική επικοινωνία της συσκευής με τον χρήστη, συγκεκριμένα ονομασία των αντικειμένων ανίχνευσης και της απόστασης τους από τον χρήστη. Ο πιο εύκολος τρόπος υλοποίησης της λεκτικής επικοινωνίας με τον χρήστη είναι μέσω ενός αλγόριθμου Text to Speech.

Η συσκευή που χρησιμοποιήθηκε είναι android, με αποτέλεσμα να είναι δυνατή η χρήση της native εντολής για text to speech που δημιούργησε η Google. Όμως, η ανάπτυξη της εφαρμογής γίνεται σε Unity αντί Android studio και για αυτό χρειαζόμαστε ένα plugin [38] που θα δίνει την δυνατότητα να χρήσης της native μεθόδου στο Unity.

Βάση της κλάσης του αντικειμένου και της απόστασης του, δημιουργείται η πρόταση που θα ακούει ο χρήστης. Για να μην είναι μονότονο επιλέγηκε η παραγωγή τεσσάρων διαφορετικών προτάσεων και η τυχαία επιλογή αυτής που θα ακούσει ο χρήστης. Σε κάθε πρόταση αναφέρεται ένα αντικείμενο με την απόσταση του και σε περίπτωση που η απόσταση είναι -1, δηλαδή η TOF κάμερα δεν μπόρεσε να επιστρέψει κάποια απόσταση, το κομμάτι που αναφέρεται στην απόσταση αγνοείται. Λόγω του ότι σε κάθε ανίχνευση είναι δυνατή η παρουσία περισσότερων του ενός αντικειμένων, επιλέγεται η αναφορά αυτού με τον μεγαλύτερο βαθμό βεβαιότητας. Σε περίπτωση που το τελευταίο αντικείμενο που σχολιάστηκε ανήκει στη ίδια κλάση με αυτό που είναι πρώτο στη λίστα προτεραιότητας τότε προχωρεί στο δεύτερο. Με αυτό τον τρόπο, εγγυάται ότι δεν θα αναφέρεται πάντα το ίδιο αντικείμενο όταν ένα πολύ καλό αποτέλεσμα παραμένει στην οθόνη για μεγάλο χρονικό διάστημα. Τέλος, επειδή το κάθε ηχητικό μήνυμα μπορεί να διαρκέσει μερικά δευτερόλεπτα επιλέγεται το επόμενο αντικείμενο που θα σχολιαστεί μόλις τελειώσει το μήνυμα χρησιμοποιώντας την τελευταία αναγνώριση.

Κεφάλαιο 5

Συμπεράσματα

5.1 Σχολιασμός αποτελεσμάτων

Στον προγραμματισμό των φάσεων της πτυχιακής εργασίας υπολογίστηκε ότι μετά την υλοποίηση της εφαρμογής θα ακολουθούσε αξιολόγηση της εφαρμογής από χρήστες για να βρεθούν τρόποι βελτίωσης της. Επειδή όμως η εφαρμογή λειτουργεί μόνο στο πρωτότυπο τηλέφωνο που δόθηκε από το RISE για την εργασία, θα έπρεπε να πραγματοποιηθεί συνάντηση με τους χρήστες, κάτι που δεν ήταν εφικτό λόγο της πανδημίας και των μέτρων αντιμετώπισης της. Για αυτό το λόγο τα αποτελέσματα που θα αναλυθούν σε αυτό το κεφάλαιο είναι σχετικά με την απόδοση των δυο εφαρμογών και τους περιορισμούς της κάμερας TOF, αντί της ικανοποίησης των αναγκών των χρηστών που ήταν το αρχικό πλάνο.

Οι υλοποιήσεις που θα σχολιαστούν περιλαμβάνουν την εφαρμογή που υλοποιήθηκε για να εκλεχθεί ο υπολογισμός της απόστασης, οι 2 εφαρμογές για ανίχνευση αντικειμένων και η υλοποίηση της ομιλίας για τα αντικείμενα που ανιχνεύονται.

5.1.1 Αποτελέσματα TOF

Όσον αφορά την κάμερα TOF επαληθεύθηκε ότι παρουσιάζεται στη θεωρία. Επιφάνειες με έντονη αντανάκλαση και περιβάλλοντα με δυνατή ηλιοφάνεια επηρεάζουν αρνητικά τις μετρήσεις της απόστασης. Η χρήση της συγκεκριμένης κάμερας σε κλειστούς χώρους έδωσε αρκετά ακριβή αποτελέσματα, απόκλιση από την πραγματική απόσταση της τάξεως μερικών εκατοστών. Όπως παρατηρήθηκε στις εικόνες πιο κάτω είναι εμφανές ότι η χρήση της TOF κάμερας σε εξωτερικούς χώρους παράγει ανακριβή αποτελέσματα. Τα χρώματα στις φωτογραφίες υποδηλώνουν την απόσταση κάθε σημείου με τα πράσινα να περιγράφουν αποστάσεις ένα με δύο μέτρα ενώ αποχρώσεις του μοβ μεγαλύτερες από 4. Στην πρώτη φωτογραφία από τα αριστερά φαίνεται να επηρεάζεται η μέτρηση της απόστασης από το φως του ήλιου. Μέχρι το σημείο που υπάρχει σκιά η απόσταση υπολογίζεται μια χαρά, με ορθές τιμές σε όλα τα σημεία της οθόνης. Οι μετρήσεις σταματούν να είναι ορθές όταν περαστεί η νοητή γραμμή της σκιάς με αποτελέσματα που δεν επιστρέφουν τιμή η αν επιστρέφουν είναι εντελώς λάθος, με παράδειγμα τη πρώτη φωτογραφία που αναφέρει την απόσταση του σημείου σαν 7 μέτρα ενώ στην πραγματικότητα ήταν λιγότερο από 3. Το ίδιο φαινόμενο παρουσιάζεται και στην μεσαία φωτογραφία, που δείχνει λάθος απόσταση στα σημεία κοντά στην καρέκλα. Επίσης, η δεύτερη εικόνα έχει ένα μεγάλο σημείο χωρίς αποστάσεις, στο κέντρο κοντά στο δέντρο, το οποίο ευθύνεται στο ότι εισχωρεί έντονη ηλιακή ακτινοβολία μέσα στο φακό της κάμερας TOF. Από την τρίτη φωτογραφία φαίνεται ότι η κάμερα, αποτυγχάνει να υπολογίσει απόσταση για επιφάνειες που είναι φτιαγμένες από γυαλί, όπως η βιτρίνα και το τραπεζάκι, επειδή αντανακλούν το υπέρυθρο φως της κάμερας. Τέλος, τα στολίδια στο χριστουγεννιάτικο δέντρο αντανακλούν το φως που αποστέλλεται και το σκορπίζουν σε διάφορες κατευθύνσεις με αποτέλεσμα να μην υπολογίζεται ούτε η απόσταση του δέντρου.

Σχήμα 5.1 Φωτογραφίες από την εκτέλεση της εφαρμογής για υπολογισμό απόστασης.

Ένας άλλος περιορισμός που έχει η κάμερα TOF είναι η απόσταση. Ένας από τους λόγους που δεν μπορεί να χρησιμοποιηθεί η εφαρμογή σε εξωτερικούς ανοικτούς χώρους είναι το ότι για μεγάλες αποστάσεις δεν υπάρχουν επιφάνειες που μπορεί να αντανακλάσουν το φως. Για να υπολογιστεί η μέγιστη απόσταση που μπορεί να υπολογίσει η TOF κάμερα που διαθέτει το κινητό ορίστηκε πείραμα στο οποίο εκτελούταν η εφαρμογή στον ίδιο χώρο και σε κάθε εκτέλεση άλλαζε η απόσταση από ένα γνωστό σημείο. Πιο συγκεκριμένα, μέτρησα την απόσταση από ένα τοίχο σε ένα δωματίου του σπιτιού και εκτέλεσα την εφαρμογή σε 3 διαφορετικά σημεία στο χώρο. Σε κάθε εκτέλεση επιλέχτηκαν διάφορα σημεία στην οθόνη για να διαπιστωθεί αν η απόσταση που επιστρέφει η εφαρμογή ήταν ορθή. Πιο κάτω, στο Σχήμα 5.2, φαίνονται τα αποτελέσματα αυτών των εκτελέσεων και για πρακτικούς λόγους οι εικόνες συνδυάστηκαν έτσι ώστε να εμφανίζονται περισσότερες από 1 αποστάσεις ανά εικόνα. Η τρίτη εικόνα είναι κάπως θολωμένη γιατί το κινητό δεν ήταν πολύ σταθερό κατά την εκτέλεση του πειράματος και οι εικόνες που συνδυάστηκαν δεν ήταν ακριβώς ίδιες. Αξίζει να σημειωθεί ότι όλες οι αποστάσεις που υπολογίστηκαν από την εφαρμογή ήταν πολύ κοντά στις αληθινές, με μια πολύ μικρή απόκλιση της τάξεως μερικών εκατοστόμετρων.

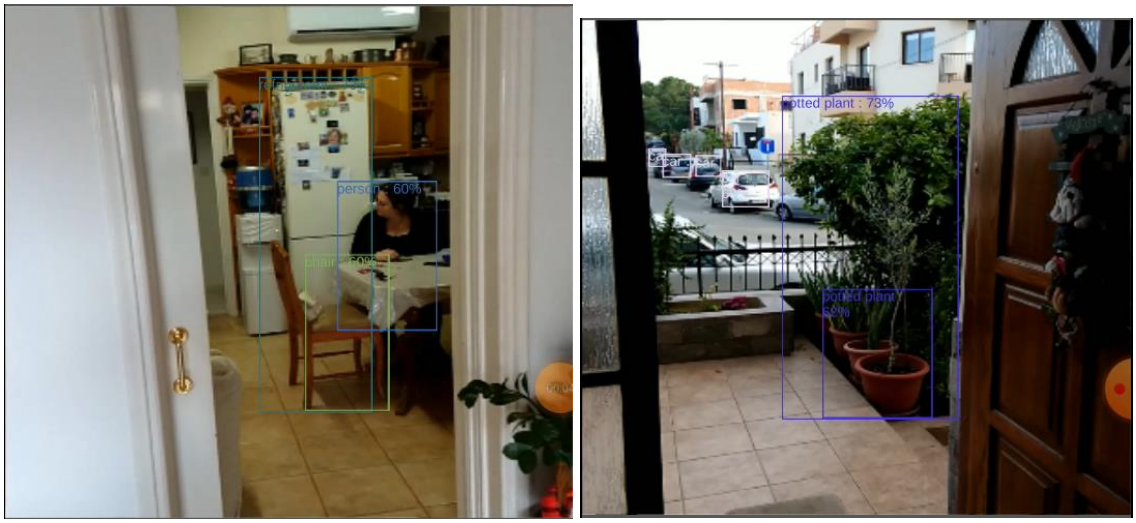
Σχήμα 5.2 Κάθε φωτογραφία είναι συνδυασμός πολλών εκτελέσεων της εφαρμογής για τον υπολογισμό απόστασης για να φαίνεται η απόσταση που υπολογίζεται σε διάφορα σημεία στην οθόνη.

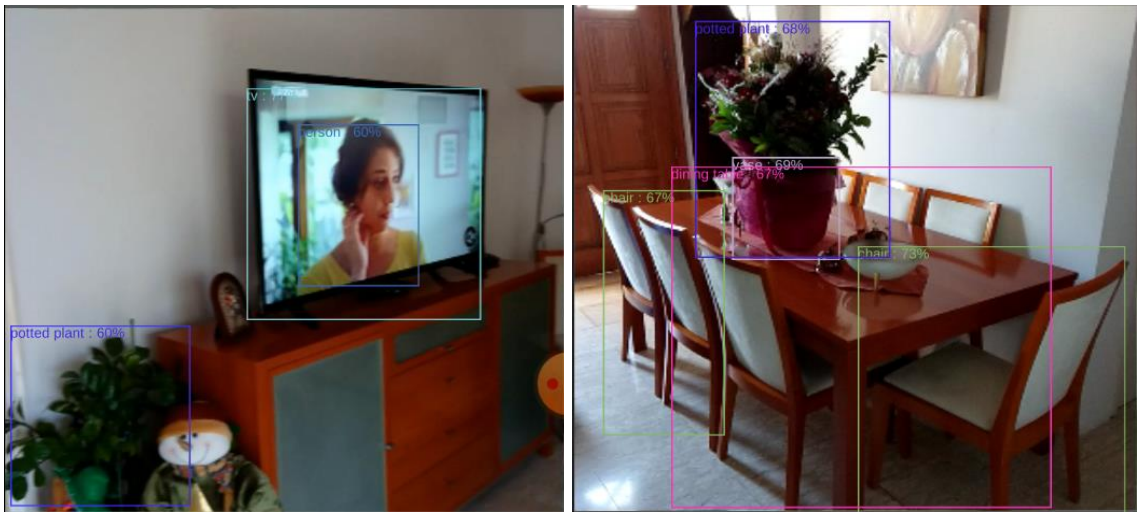
Από αυτή τη δοκιμή παρατηρήθηκε ότι για αποστάσεις κάτω των τριών μέτρων η TOF κάμερα δίνει τιμές απόστασης για τα πλείστα σημεία με τους μόνους περιορισμούς του υπολογισμού να είναι αυτοί που αναφέρθηκαν πιο πριν, δηλαδή οι γυαλιστερές επιφάνειες. Επίσης παρατηρήθηκε ότι όσο μεγαλώνει η απόσταση ο αριθμός των σημείων χωρίς απόσταση αυξάνεται. Η κλήση της κάμερας επηρεάζει την εμβέλεια που έχει το φωτεινό σήμα και στην περίπτωση της μεσαίας εικόνας του σχήματος 5.2, όπου το τηλέφωνο έχει κλήση 90 μοίρες από το έδαφος, τα σημεία στο πάτωμα που δεν έχουν απόσταση ξεκινούν μετά τα 3 μέτρα. Τα χρώματα που εμφανίζονται στη οθόνη του τηλεφώνου υποδεικνύουν την απόσταση και όπως φαίνεται από το πιο πάνω σχήμα αποστάσεις πέραν των 4,5 μέτρων δεν χρωματίζονται. Σημεία που είναι παράλληλα με την κάμερα έχουν μεγαλύτερη πιθανότητα να πάρουν απόσταση, ενώ όσο αυξάνεται η απόσταση η παραμικρή κλίση μπορεί να εκτροχιάσει το φωτεινό σήμα και ως αποτέλεσμα να μην υπάρχει μέτρηση. Από την δεξιά εικόνα του σχήματος 5.2 φαίνεται ότι η κάμερα μπορεί να υπολογίσει αποστάσεις πέραν των 5 μέτρων αλλά μόνο σε ιδανικές συνθήκες, για παράδειγμα η επιφάνεια να είναι μεγάλη, ομαλή και ακριβώς απέναντι από την κάμερα. Στο πάτωμα αυτής της εικόνας υπολογίζονται αποστάσεις

μέχρι τα 3,5 μέτρα διότι το τηλέφωνο είχε μια μικρή κλίση προς τα κάτω κατά την εκτέλεση του πειράματος.

5.1.2 Αποτελέσματα ανίχνευσης με TensorFlow Lite

Η πρώτη υλοποίηση για την ανίχνευση αντικειμένων στην εφαρμογή ήταν με την χρήση της βιβλιοθήκης TensorFlow Lite για το περιβάλλον Unity. Με αυτή την υλοποίηση επιτεύχθηκε ανίχνευση σε πραγματικό χρόνο, ενώ παράλληλα υπήρχε ακρίβεια στην τοποθέτηση των bounding boxes και στην ονομασία των αντικειμένων. Πιο κάτω, στο Σχήμα 5.3 είναι εμφανή κάποια παραδείγματα της εκτέλεσης της εφαρμογής. Από αυτά είναι κατανοητό ότι τα πλείστα αντικείμενα που μπορούν να ανιχνευτούν από το δίκτυο ανιχνεύονται, ακόμα και σε μεγάλες αποστάσεις, όπως τα αυτοκίνητα στην πάνω δεξιά φωτογραφία. Για να μειωθεί ο αριθμός των λανθασμένων ανιχνεύσεων ορίστηκε το κατώφλι ανίχνευσης στο 60% και γι' αυτό πιθανόν να μην ανιχνεύονται όλες οι καρέκλες στην κάτω δεξιά φωτογραφία. Το ψηλό κατώφλι βοηθά από την άλλη στο να μειωθεί η πιθανότητα αναφοράς κάποιου λάθους αντικειμένου στον χρήστη, μέσω της σύνθεσης ομιλίας.



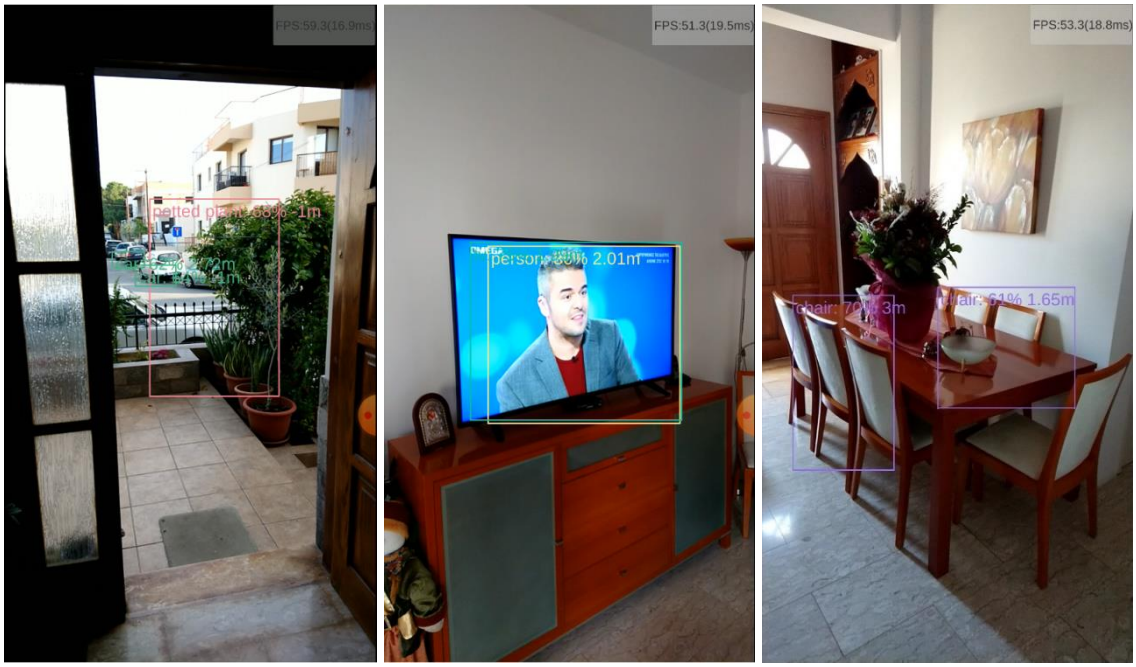


Σχήμα 5.3 Φωτογραφίες από την εκτέλεση της εφαρμογής για ανίχνευση αντικειμένων με τη βιβλιοθήκη TensorFlow Lite.

5.1.3 Αποτελέσματα ανίχνευσης με TensorFlow Sharp

Για λόγους που εξηγήθηκαν στην προηγούμενη ενότητα κρίθηκε αναγκαία η υλοποίηση της εφαρμογής με δεύτερη βιβλιοθήκη ανίχνευσης αντικειμένων που θα μπορεί να χρησιμοποιηθεί παράλληλα με την μέθοδο υπολογισμού απόστασης. Αυτή η υλοποίηση έγινε με τη χρήση της βιβλιοθήκης TensorFlow Sharp, αλλά τα αποτελέσματα υστερούν πολύ σε σύγκριση με την πρώτη υλοποίηση. Επειδή, η ανίχνευση είχε μεγάλη χρονική καθυστέρηση παρουσιάστηκε ανάγκη να χρησιμοποιηθεί νευρωνικό δίκτυο που θυσιάζει ακρίβεια για ταχύτητα, με αποτέλεσμα, όπως φαίνεται και από τις εικόνες, να χάνεται μεγάλος αριθμός ανιχνεύσεων. Κάθε ανίχνευση σε αυτή την υλοποίηση χρειάζεται περίπου 2,5 δευτερόλεπτα για να ολοκληρωθεί και έτσι οποιαδήποτε μετακίνηση αντικειμένων στην οθόνη έχει ως αποτέλεσμα τα κουτιά που θα εμφανιστούν στην οθόνη να μην περιτριγυρίζουν τα αντικείμενα που ανιχνευτήκαν. Αυτό φαίνεται από τις πρώτες δύο εικόνες που λήφθηκαν ενώ η συσκευή περιστρεφόταν στον χώρο με μικρή ταχύτητα. Στην τρίτη φωτογραφία έγινε προσπάθεια δημιουργίας ανίχνευσης, όπως παρουσιάζεται στην κάτω δεξιά φωτογραφία της προηγούμενης υλοποίησης (Σχήμα 5.3). Η διαφορά μεταξύ των δύο ανιχνεύσεων είναι τεράστια με την καινούρια υλοποίηση να υστερεί κατά πολύ. Το θετικό αποτέλεσμα αυτής της υλοποίησης είναι η δυνατότητα ανάθεσης απόστασης σε κάθε αντικείμενο που ανιχνεύτηκε. Σε κάθε κουτί που σχεδιάζεται στην

οθόνη αναγράφεται η απόσταση που είχε το κέντρο του κουτιού την ώρα που ξεκίνησε η ανίχνευση και σε περίπτωση που δεν μπορεί αν υπολογιστεί αναγράφεται -1.



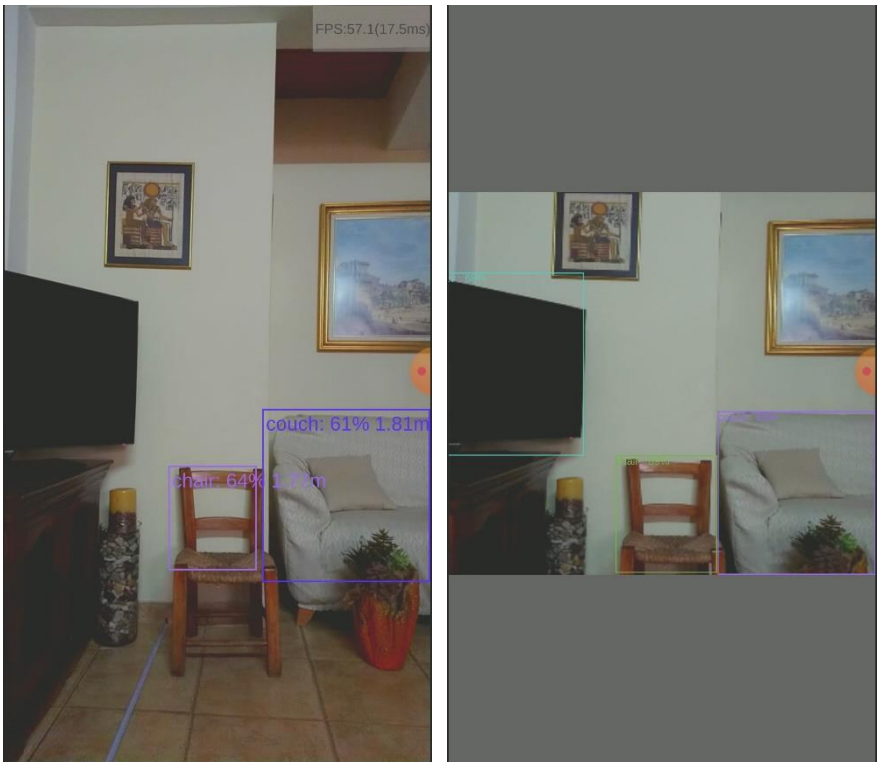
Σχήμα 5.4 Φωτογραφίες από την εκτέλεση της εφαρμογής για ανύχνευση αντικημένων με τη βιβλιοθήκη TensorFlow Sharp.

5.1.4 Αποτελέσματα σύγκρισης εφαρμογών ως προς την απόσταση

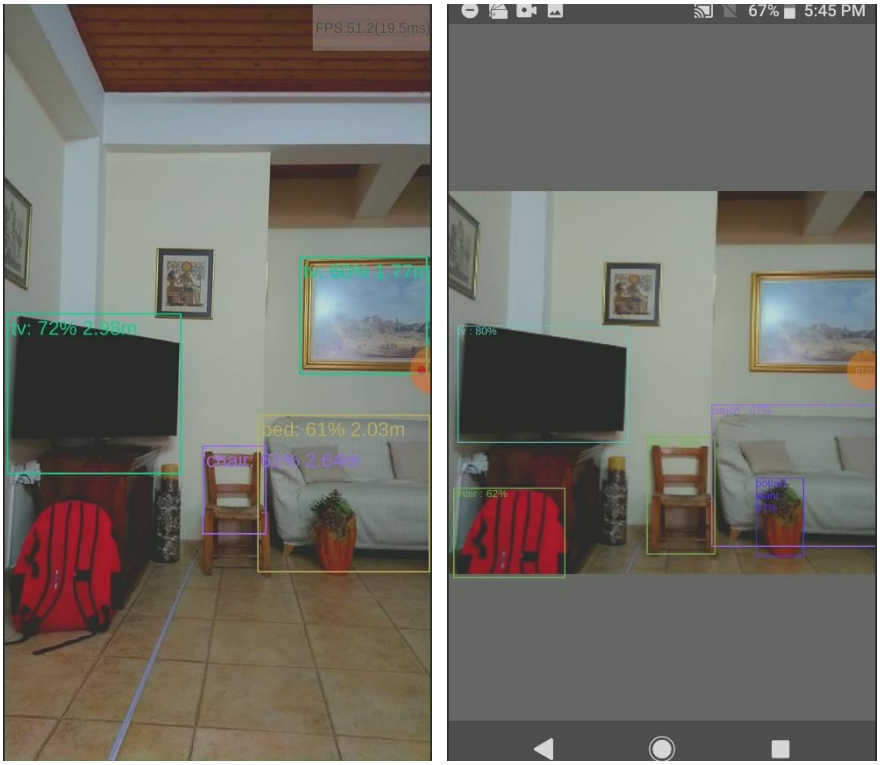
Για να γίνει καλύτερη σύγκριση των δύο εφαρμογών δημιουργήθηκε ένα περιβάλλον όπου το τηλέφωνο και τα αντικείμενα προς ανίχνευση ήταν σταθερά για τις εκτελέσεις πειραμάτων. Τα αντικείμενα προς ανίχνευση ήταν μια κλειστή τηλεόραση, μια κόκκινη βαλίτσα, μια μικρή καρέκλα, μια γλάστρα με φυτό και ένας καναπές. Το κάδρο που ανιχνεύεται σε κάποιες εκτελέσεις δεν ήταν μέρος του πειράματος αλλά το γεγονός ότι ανιχνεύεται σαν τηλεόραση είναι μεν λάθος αλλά είναι λογικό. Συνολικά έγιναν τέσσερις εκτελέσεις, με απόσταση δύο, τρία τέσσερα και πέραν των τεσσάρων μέτρων από τον τοίχο απέναντι και το κινητό δεν μετακινήθηκε μεταξύ των εκτελέσεων.. Σκοπός αυτού του πειράματος ήταν να βρεθεί η μέγιστη απόσταση που μπορούν οι εφαρμογές να ανιχνεύσουν αντικείμενα και να γίνει σύγκριση για των δύο υλοποιήσεων χρησιμοποιώντας τις ίδιες εισόδους.

Στις πιο κάτω εικόνες εμφανίζονται τα αποτελέσματα του πειράματος. Σε όλες τις περιπτώσεις η εκτέλεση με την βιβλιοθήκη TensorFlow Lite έχει καλύτερα αποτελέσματα, δηλαδή περισσότερες ορθές ανιχνεύσεις και τα κουτιά περικυκλώνουν καλύτερα τα αντικείμενα. Στο Σχήμα 5.5 φαίνεται ότι η υλοποίηση με την κάμερα TOF ανιχνεύει δύο τρίτα των αντικείμενα που ανιχνεύει η άλλη υλοποίηση. Οι αποστάσεις που δίνονται από την εφαρμογή για την καρέκλα και τον καναπέ είναι ορθές. Στο Σχήμα 5.6 φαίνεται ότι στα 3 μέτρα η υλοποίηση χωρίς την TOF κάμερα ανιχνεύει όλα τα αντικείμενα του πειράματος, με μόνο μια λάθος ανίχνευση που αντί για backpack επιστρέφει chair. Τα κουτιά γύρω από κάθε αντικείμενο είναι σχεδόν τέλεια σε αυτή την εκτέλεση. Η υλοποίηση με την κάμερα TOF κάνει λιγότερες ανιχνεύσεις από αυτή την απόσταση αφού δεν βρίσκει την βαλίτσα και το φυτό. Επίσης αναγνωρίζει λάθος τον πίνακα σαν τηλεόραση και τον καναπέ σαν κρεβάτι. Στο σχήμα 5.7 άρχισε να φαίνεται ότι η απόσταση επηρεάζει σε μεγαλύτερο βαθμό την υλοποίηση με την βιβλιοθήκη TensorFlow Sharp. Στα 4 μέτρα η εφαρμογή αποτυγχάνει να ανιχνεύσει οποιοδήποτε αντικείμενο σωστά για το συγκεκριμένο πείραμα αλλά η απόσταση που επιστράφηκε ήταν ορθή. Από την άλλη η υλοποίηση με την βιβλιοθήκη TensorFlow Lite ανιχνεύει με μεγάλη ακρίβεια τέσσερα από τα πέντε αντικείμενα. Στην τελευταία εκτέλεση του πειράματος επαναλαμβάνεται ξανά η παρατήρηση ότι η υλοποίηση με την βιβλιοθήκη TensorFlow Sharp δυσκολεύεται να ανιχνεύσει αντικείμενα ή όταν το κάνει είναι λάθος. Η διαφορά όμως με την προηγούμενη εκτέλεση είναι ότι αυτή τη φόρα η απόσταση που επιστρέφεται είναι εντελώς λανθασμένη.

Από το πείραμα αυτό, συμπεραίνεται ότι για αποστάσεις πέραν τον τριών μέτρων η υλοποίηση με την βιβλιοθήκη TensorFlow Sharp παράγει ανακριβή αποτελέσματα και πέραν των τεσσάρων μέτρων η απόσταση που επιστρέφει η κάμερα TOF πιθανόν να είναι λανθασμένη. Για την υλοποίηση χωρίς την TOF φαίνεται ότι σε μεγαλύτερες αποστάσεις τα μικρά αντικείμενα δεν ανιχνεύονται αλλά διατηρείται μια καλού επιπέδου ακρίβεια. Τέλος, από αυτό το πείραμα επιβεβαιώνεται ότι το μέγεθος του αντικειμένου στην οθόνη παίζει μεγάλο ρόλο στην ανίχνευση του.



Σχήμα 5.5 Φωτογραφίες από την εκτέλεση των δύο εφαρμογών με απόσταση 2 μέτρα από τον τοίχο. Στην αριστερά φαίνεται η υλοποίηση με την βιβλιοθήκη TensorFlow Sharp και της κάμερας TOF, ενώ δεξιά με την βιβλιοθήκη TensorFlow Lite χωρίς TOF.



Σχήμα 5.6 Φωτογραφίες από την εκτέλεση των δύο εφαρμογών με απόσταση 3 μέτρα από τον τοίχο. Αριστερά TensorFlow Sharp με TOF, δεξιά TensorFlow Lite χωρίς TOF.

Σχήμα 5.7 Φωτογραφίες από την εκτέλεση των δύο εφαρμογών με απόσταση 3 μέτρα από τον τοίχο. Αριστερά TensorFlow Sharp με TOF, δεξιά TensorFlow Lite χωρίς TOF.

Σχήμα 5.8 Φωτογραφίες από την εκτέλεση των δύο εφαρμογών με απόσταση μεγαλύτερη από 4 μέτρα από τον τοίχο. Αριστερά TensorFlow Sharp με TOF, δεξιά TensorFlow Lite χωρίς TOF.

5.1.5 Αποτελέσματα σύνθεσης ομιλίας

Για την επικοινωνία με τον χρήστη επιλέγηκε η παραγωγή διάφορων μηνυμάτων που ονομάζουν κάποιο αντικείμενο και να αναφέρουν την απόσταση του, αν υπάρχει. Έπειτα επιλέγεται τυχαία κάποιο από αυτά και μεταδίδεται στον χρήστη με μέθοδο Text to Speech. Η εφαρμογή δεν στέλνει κατ' επιλογήν δεύτερο μήνυμα για ομιλία αν δεν έχει ολοκληρωθεί το πρώτο για να εξαλειφθεί το ενδεχόμενο να ξεκινήσει να μιλά για κάποιο άλλο αντικείμενο που ανιχνεύτηκε καθώς γίνεται περιγραφή άλλου αντικειμένου. Μέσα στις υποθέσεις που έγιναν για την εφαρμογή θεωρήθηκε ότι το αντικείμενο με τη μεγαλύτερη βεβαιότητα ανίχνευσης έχει τη λιγότερη πιθανότητα να είναι λάθος άρα επιλέγεται για να αναφερθεί. Άλλη υπόθεση που έγινε είναι ότι αν αναφερθεί κάποιο αντικείμενο, στην επόμενη ομιλία θέλουμε να αναφέρουμε κάποιο άλλο αντικείμενο, αν το πρώτο εξακολουθεί αν έχει την πιο μεγάλη σιγουριά. Με αυτό τον τρόπο έγινε δυνατή η αποφυγή της συνεχόμενης αναφοράς στο ίδιο αντικείμενο στην περίπτωση που ο χρήστης παραμένει ακίνητος. Οι μόνες περιπτώσεις που η εφαρμογή αναφέρει λάθος

πληροφορίες είναι όταν ανιχνεύεται κάποιο αντικείμενο του οποίου η κλάση είναι εντελώς λάθος ή η ανάθεση της κλάσης αλλάζει ανάλογα με την οπτική γωνιά, κάτι για το οποίο δεν ευθύνεται η υλοποίηση σύνθεσης γλώσσας αλλά η ανίχνευση.

5.2 Συμπεράσματα

Είναι σημαντικό να αναφερθεί πως πέραν της θεωρητικής γνώσης που αποκτήθηκε κατά την έρευνα και ανάπτυξη των εφαρμογών που υλοποιήθηκαν, τέθηκαν οι βάσεις για την ανάπτυξη μιας εφαρμογής που θα μπορεί να βοηθήσει τα άτομα με προβλήματα όρασης να αντιλαμβάνονται καλύτερα το περιβάλλον τους. Αρχικά, αποδείχθηκε ότι χρησιμοποιώντας την υπολογιστική δύναμη ενός κινητού τηλεφώνου με σχετικά παλιό επεξεργαστή είναι δυνατή η ανίχνευση αντικειμένων σε πραγματικό χρόνο, νοουμένου ότι χρησιμοποιείται η κατάλληλη βιβλιοθήκη στην υλοποίηση. Από την άλλη, τέθηκαν οι βάσεις για το πώς μια τέτοια εφαρμογή μπορεί να χρησιμοποιηθεί παράλληλα με TOF κάμερα και τεχνολογία σύνθεσης ομιλίας για να μεταδώσει επιπλέον πληροφορίες, όπως την απόσταση του αντικειμένου, σε φωνητικά μηνύματα. Τέλος, χρησιμοποιήθηκε η κάμερα TOF που ήταν διαθέσιμη ώστε να ανευρεθούν τα σενάρια που αποφέρουν λάθος αποτελέσματα. Μέσα από αυτή την έρευνα ορίστηκε ότι η συγκεκριμένη υλοποίηση είναι ιδανική για κλειστούς χώρους και σε περιπτώσεις που χρησιμοποιείται σε εξωτερικούς χώρους προβλέπονται ανακρίβειες στην απόσταση.

Μπορεί η υλοποίηση που χρησιμοποιεί την κάμερα TOF να υστερεί από το αναμενόμενο αποτέλεσμα, άλλα θέτει τα θεμέλια για μια ενδεικτική λύση που συνδυάζει state of the art αλγόριθμους ανίχνευσης αντικειμένων και τεχνολογία Time of Flight. Λόγω διαφορών καταστάσεων δεν έγινε εφικτή η υλοποίηση μιας τελικής εφαρμογής που να χρησιμοποιεί όλες τις τεχνολογίες που αναφέρθηκαν και η λύση που παρουσιάστηκε αφορά μόνο στη λειτουργία με το συγκεκριμένο κινητό. Στο μέλλον σίγουρα μπορεί να υλοποιηθεί η εφαρμογή που προτείνεται και με την χρήση του καινούριου SDK που λήφθηκε πρόσφατα η υλοποίηση είναι θέμα μερικών μηνών.

5.3 Μελλοντική εργασία

Όπως αναφέρθηκε και προηγουμένως, λόγω τεχνικών δυσκολιών και της πανδημίας κατέληξα να έχω 2 εφαρμογές, εκ των οποίων καμία δεν εξυπηρετεί πλήρως τις ανάγκες του προβλήματος. Η πρώτη που χρησιμοποιεί την απόσταση έχει πολύ μεγάλη καθυστέρηση στην ανίχνευση αντικειμένων και η δεύτερη ενώ έχει πολύ καλή απόδοση στις ανιχνεύσεις δεν εκμεταλλεύεται την TOF κάμερα του κινητού.

Ως μελλοντική εργασία επιβάλλεται να υλοποιηθεί ξανά η εφαρμογή, αυτή τη φορά με το καινούριο SDK της Sony, το οποίο δεν θα συγκρούεται με την βιβλιοθήκη της TensorFlow Lite για το Unity. Σε αυτή την υλοποίηση μπορούν να ελεγχθούν τα όρια της εφαρμογής και εφόσον υπάρχει η δυνατότητα να χρησιμοποιηθεί κάποιο μοντέλο με μεγαλύτερη ακρίβεια. Μπορεί επίσης, να εκπαιδευτεί από την αρχή κάποιο νευρωνικό δίκτυο που να ανιχνεύει μόνο τις κλάσεις που είναι πιθανών να συναντήσει ο χρήστης, έναντι στην πληθώρα των αντικειμένων που υπάρχουν στο dataset COCO[11], όπως ελέφαντας ή καμηλοπάρδαλη που κατά πάσα πιθανότητα δεν θα χρειαστεί να ανιχνευτούν ποτέ από την συγκεκριμένη εφαρμογή. Με το προσαρμοσμένο νευρωνικό δίκτυο οι ανιχνεύσεις θα έχουν μεγαλύτερη ακρίβεια και αν ο αριθμός των κλάσεων που ανιχνεύονται είναι μικρότερος θα έχει και μεγαλύτερη ταχύτητα.

Στο μέλλον μπορούν να υλοποιηθούν επιπλέον λειτουργίες στην εφαρμογή. Αυτές οι λειτουργίες μπορούν να περιλαμβάνουν ενημέρωση του χρήστη για το χρώμα του αντικειμένου. Αυτή η λειτουργία μπορεί να υλοποιηθεί με χρωματική ανάλυση του αντικειμένου, αφού αφαιρεθεί το παρασκήνιο και μπορεί να βοηθήσει τον χρήστη να συνδέσει την υφή ή την μυρωδιά του αντικείμενου με το χρώμα που δεν είδε ποτέ. Άλλη λειτουργία που θα ήταν καλό να διαθέτει η εφαρμογή είναι η ανάγνωση κειμένου όταν ανιχνεύεται. Μέσω αυτής, προσδίδεται στον χρήστη επιπλέον αυτονομία, με τη δυνατότητα χρήσης της εφαρμογής για να διαβάσει κάποια πινακίδα ή την ετικέτα κάποιου προϊόντος που αγόρασε. Επειδή η χρήση menu και κουμπιών στην εφαρμογή δεν είναι εφικτή, σε περίπτωση που προστεθούν κι άλλες λειτουργίες θα ήταν σοφό να υλοποιηθεί κάποιου είδους επικοινωνία με τον χρήστη χρησιμοποιώντας φωνητικά μηνύματα, όπως συμβαίνει και με τα virtual assistants, ή με gestures – κινήσεις των χεριών μπροστά στην κάμερα. Μέσω της επικοινωνίας αυτής ο χρήστης μπορεί για

παράδειγμα να σταματήσει την ανίχνευση αντικειμένων ή να ενημερώσει την εφαρμογή ότι θέλει να του διαβάσει κάποιο κείμενο. Τέλος, θα ήταν καλή ιδέα να υλοποιηθεί ομιλία και σε άλλες γλώσσες, κυρίως ελληνικά, μιας και οι χρήστες δεν είναι όλοι γνώστες της αγγλικής γλώσσας.

Για άλλες βελτιώσεις ή επιπλέον λειτουργίες της εφαρμογής μπορούμε να αποταθούμε στα άτομα που στοχεύουμε να βοηθήσουμε με την εφαρμογή. Λόγω της η συνάντηση με άτομα που έχουν προβλήματα όρασης για συζήτηση των αναγκών τους και του βαθμού βοήθειας που τυχόν θα λάμβαναν από την εφαρμογή δεν ήταν εφικτή. Το πιο σημαντικό μελλοντικό βήμα για την ανάπτυξη της εφαρμογής που προτάθηκε σε αυτή την εργασία είναι η ανάλυση των αναγκών και η γνώμη των ατόμων που θα χρησιμοποιήσουν την εφαρμογή. Με την πρώτη ευκαιρία θα ήταν ιδανικό η εφαρμογή που υλοποιήθηκε μέχρι τώρα να χρησιμοποιηθεί από κάποιο τυφλό ή άτομο με προβλήματα όρασης, ώστε να παρουσιαστούν οι πραγματικές αδυναμίες της εφαρμογής και με σκοπό την εποικοδομητική βελτίωση της εφαρμογής.

Βιβλιογραφία

[1] "2018 Turing Award." https://awards.acm.org/about/2018-turing (accessed Dec. 07, 2020).

[2] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, Jul. 1959, doi: 10.1147/rd.33.0210.

[3] W. Serrano, "Neural Networks in Big Data and Web Search," *Data*, vol. 4, no. 1, p. 7, Dec. 2018, doi: 10.3390/data4010007.

[4] W. S. Mcculloch and W. Pitts, "A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY," 1943.

[5] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, Mar. 2018, vol. 2018-January, pp. 1–6, doi: 10.1109/ICEngTechnol.2017.8308186.

[6] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," 2001.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, 2005, vol. I, pp. 886–893, doi: 10.1109/CVPR.2005.177.

[9] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," 1999. Accessed: Dec. 30, 2020. [Online]. Available: www.research.att.com/.

[10] "Breaking Down Mean Average Precision (mAP) | by Ren Jie Tan | Towards Data Science." https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52#1a59 (accessed Dec. 28, 2020).

[11] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, May 2014, vol. 8693 LNCS, no. PART 5, pp. 740–755, doi: 10.1007/978-3-319-10602-1_48.

[12] “1920px-20200501_Time_of_flight.svg.png (1920×1280).” https://upload.wikimedia.org/wikipedia/commons/thumb/f/f1/20200501_Time_of_flight.svg/1920px-20200501_Time_of_flight.svg.png (accessed Dec. 28, 2020).

[13] R. Doiphode, M. Ganore, A. Garud, T. Ghuge, and P. K. Guide, “Be My Eyes : Android Voice Application for Visually Impaired People,” no. April, 2017, doi: 10.13140/RG.2.2.12307.48164.

[14] N. Kanwal, E. Bostanci, K. Currie, and A. F. Clark, “A Navigation System for the Visually Impaired: A Fusion of Vision and Depth Sensor,” *Appl. Bionics Biomech.*, vol. 2015, 2015, doi: 10.1155/2015/479857.

[15] A. Aladren, G. Lopez-Nicolas, L. Puig, and J. J. Guerrero, “Navigation Assistance for the Visually Impaired Using RGB-D Sensor with Range Expansion,” *IEEE Syst. J.*, vol. 10, no. 3, pp. 922–932, 2016, doi: 10.1109/JSYST.2014.2320639.

[16] W. Liu *et al.*, “SSD: Single shot multibox detector,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9905 LNCS, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.

[17] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, and S. Ioffe, “Scalable, High-Quality Object Detection,” Dec. 2014, Accessed: Dec. 30, 2020. [Online]. Available: http://arxiv.org/abs/1412.1441.

[18] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” Sep. 2015, Accessed: Dec. 22, 2020. [Online]. Available: http://www.robots.ox.ac.uk/.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” Mar. 2010, pp. 248–255, doi: 10.1109/cvpr.2009.5206848.

[20] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017, [Online]. Available: http://arxiv.org/abs/1704.04861.

[21] “How single-shot detector (SSD) works? | ArcGIS for Developers.” https://developers.arcgis.com/python/guide/how-ssd-works/ (accessed Dec. 30, 2020).

[22] “Google AI Blog: MobileNets: Open-Source Models for Efficient On-Device

Vision.” https://ai.googleblog.com/2017/06/mobilenets-open-source-models-for.html (accessed Dec. 27, 2020).

[23] “Android Developers Blog: An introduction to Text-To-Speech in Android.” https://android-developers.googleblog.com/2009/09/introduction-to-text-to-speech-in.html (accessed Dec. 27, 2020).

[24] J. Huang *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-Janua, pp. 3296–3305, doi: 10.1109/CVPR.2017.351.

[25] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” Accessed: Dec. 22, 2020. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/results.

[26] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object detection via region-based fully convolutional networks,” in *Advances in Neural Information Processing Systems*, May 2016, pp. 379–387, Accessed: Dec. 22, 2020. [Online]. Available: https://github.com/daijifeng001/r-fcn.

[27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-December, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[28] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *32nd International Conference on Machine Learning, ICML 2015*, Feb. 2015, vol. 1, pp. 448–456, Accessed: Dec. 22, 2020. [Online]. Available: https://arxiv.org/abs/1502.03167v3.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-December, pp. 2818–2826, doi: 10.1109/CVPR.2016.308.

[30] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-ResNet and the impact of residual connections on learning,” in *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, Feb. 2017, pp. 4278–4284, Accessed: Dec. 22, 2020. [Online]. Available: https://arxiv.org/abs/1602.07261v2.

[31] A. Younis, L. Shixin, S. Jn, and Z. Hai, “Real-Time Object Detection Using Pre-Trained Deep Learning Models MobileNet-SSD,” in *Proceedings of 2020 the 6th International Conference on Computing and Data Engineering*, Jan. 2020, pp. 44–48, doi: 10.1145/3379247.3379264.

[32] “models/tf1_detection_zoo.md at master · tensorflow/models · GitHub.” https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md (accessed Dec. 22, 2020).

[33] “tensorflow/tensorflow/lite/experimental/examples/unity/TensorFlowLitePlugin at master · tensorflow/tensorflow · GitHub.” https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/experimental/examples/unity/TensorFlowLitePlugin (accessed Dec. 08, 2020).

[34] “GitHub - asus4/tf-lite-unity-sample: TensorFlow Lite Samples on Unity.” https://github.com/asus4/tf-lite-unity-sample (accessed Dec. 08, 2020).

[35] “GitHub - migueldeicaza/TensorFlowSharp: TensorFlow API for .NET languages.” https://github.com/migueldeicaza/TensorFlowSharp (accessed Dec. 08, 2020).

[36] “unity-ml-agents/Using-TensorFlow-Sharp-in-Unity.md at master · miyamotok0105/unity-ml-agents · GitHub.” https://github.com/miyamotok0105/unity-ml-agents/blob/master/docs/Using-TensorFlow-Sharp-in-Unity.md#using-tensorflowsharp-with-ml-agents (accessed Dec. 08, 2020).

[37] “GitHub - Syn-McJ/TFClassify-Unity: An example of using Tensorflow with Unity for image classification and object detection.” https://github.com/Syn-McJ/TFClassify-Unity (accessed Dec. 08, 2020).

[38] “GitHub - PingAK9/Speech-And-Text-Unity-iOS-Android: Speed to text in Unity iOS use Native Speech Recognition.” https://github.com/PingAK9/Speech-And-Text-Unity-iOS-Android (accessed Dec. 08, 2020).

Παράρτημα Α

Σε αυτό το παράρτημα επισυνάπτεται ο κώδικας που χρησιμοποιήθηκε για την υλοποίηση με την βιβλιοθήκη TensorflowSharp και την κάμερα TOF. Είδη η συμφωνία με την Sony αναφέρει ότι δεν επιτρέπεται να δημοσιευτεί κώδικας που δόθηκε, οι μέθοδοι που χρησιμοποιούν εντολές του SDK παραλείπονται. Αυτό περιλαμβάνει τις μεθόδους getDepth, getColorTexture και τα initialization των παραμέτρων της κάμερας στην μέθοδο start.

```
1.  using System;
2.  using System.Collections;
3.  using System.Collections.Generic;
4.  using UnityEngine;
5.  using UnityEngine.UI;
6.  using System.IO;
7.  using System.Linq;
8.  using System.Text;
9.  using System.Text.RegularExpressions;
10. using TFClassify;
11. using System.Diagnostics;
12. using System.Threading.Tasks;
13. using Debug = UnityEngine.Debug;
14. using TensorFlow;
15. using TextSpeech;
16.
17. public class DetectWithTOF : MonoBehaviour
18. {
19.     [SerializeField] TextAsset modelFile;
20.     [SerializeField] TextAsset labelsFile;
21.     [SerializeField] TextAsset colorFile;
22.     [SerializeField] RawImage background;
23.
24.     [SerializeField, Range(0.1f, 1)]float MINIMUM_CONFIDENCE = 0.5f;
25.     [SerializeField, Range(0.5f, 2)] float pitch = 1f; //[0.5 - 2] Default 1
26.     [SerializeField, Range(0.5f, 2)] float rate = 1f; //[min - max] android:[0.5 - 2] iOS:[0 - 1]
27.     [SerializeField] string code = "en-US";
28.
29.
30.     private const int detectImageSize = 300;
31.
32.     private static Texture2D boxOutlineTexture;
33.     private static GUIStyle labelStyle;
34.
35.     private Detector detector;
36.
37.     private List<BoxOutline> boxOutlines;
38.     private Vector2 backgroundSize;
39.     private Vector2 backgroundOrigin;
40.     private bool isSpeaking = false;
41.
42.     private async Task UpdateAsync()
```

```
43.     {
44.         while (true)
45.         {
46.             await TFDetect();
47.         }
48.     }
49.
50.     public void OnGUI()
51.     {
52.         if (this.boxOutlines != null && this.boxOutlines.Any())
53.         {
54.             foreach (var outline in this.boxOutlines)
55.             {
56.                 DrawBoxOutline(outline);
57.             }
58.         }
59.     }
60.
61.     private void LoadWorker()
62.     {
63.         try
64.         {
65.             LoadDetector();
66.         }
67.         catch (TFException ex)
68.         {
69.             if (ex.Message.EndsWith("is up to date with your GraphDef-generating binary.)."))
70.             {
71.                 Debug.Log("Error: TFException. Make sure you model trained with same version of T
    ensorFlow as in Unity plugin.");
72.             }
73.
74.             throw;
75.         }
76.     }
77.
78.     private void LoadDetector()
79.     {
80.         // Labels
81.         String[] labels = labelsFile.text.Split('\n');
82.
83.         // Colors
84.         string[] lines = colorFile.text.Split('\n');
85.         Color[] colors = new Color[labels.Length];
86.         for (int i = 0; i < labels.Length; i++)
87.         {
88.             string[] line = lines[i].Split(' ');
89.             colors[i] = new Color(float.Parse(line[0])/255, float.Parse(line[1])/255, float.Parse(line[2
    ])/255, 1f);
90.         }
91.
92.         this.detector = new Detector(
93.             this.modelFile.bytes,
94.             labels,colors, MINIMUM_CONFIDENCE,
95.             detectImageSize);
96.     }
97.
98.     private async Task TFDetect()
99.     {
```

```
100.        float ctime = Time.realtimeSinceStartup;
101.        //Debug.Log("<-- Detect starts " + (Time.realtimeSinceStartup - ctime));
102.        UpdateBackgroundOrigin();
103.        //Debug.Log("<-
      - UpdateBackgroundOrigin returned " + (Time.realtimeSinceStartup - ctime));
104.        var snap = TakeTextureSnap();
105.        //Debug.Log("<-- Snap taken " + (Time.realtimeSinceStartup - ctime));
106.        var scaled = Scale(snap, detectImageSize);
107.        //Debug.Log("<-- Scaled " + (Time.realtimeSinceStartup - ctime));
108.
109.        int angle = 90;
110.        switch (Screen.orientation)
111.        {
112.            case ScreenOrientation.Portrait:
113.                angle = 90;
114.                break;
115.            case ScreenOrientation.PortraitUpsideDown:
116.                angle = -90;
117.                break;
118.            case ScreenOrientation.LandscapeRight:
119.                angle = 180;
120.                break;
121.            case ScreenOrientation.LandscapeLeft:
122.                angle = 0;
123.                break;
124.        }
125.        var rotated = await RotateAsync(scaled.GetPixels32(), scaled.width, scaled.height, angle);
126.        //Debug.Log("<-- Rotated " + (Time.realtimeSinceStartup - ctime));
127.        //Debug.Log("<- Preproccessing done " + (Time.realtimeSinceStartup - ctime));
128.        this.boxOutlines = await this.detector.DetectAsync(rotated);
129.        Debug.Log("<-- Detect returned " + (Time.realtimeSinceStartup - ctime));
130.        Debug.Log("Boxes detected " + boxOutlines.Count);
131.        foreach (var outline in this.boxOutlines)
132.        {
133.            outline.Depth = GetDepth(new Vector2(outline.XCentre, outline.YCentre));
134.            //Debug.Log(outline.XMin + " " + outline.XMax + " " + outline.YMin + " " + outline.Y
      Max);
135.            Debug.Log(outline.Label + " " + outline.Score * 100 + "% " + outline.Depth + "m " + out
      line.XCentre + "," + outline.YCentre);
136.        }
137.        if (!isSpeaking && this.boxOutlines.Any()) {
138.            string txt = GenerateText(this.boxOutlines[0].Label,this.boxOutlines[0].Depth);
139.            TextToSpeech.instance.StartSpeak(txt);
140.        }
141.
142.        Destroy(snap);
143.        Destroy(scaled);
144.        //Debug.Log("<-- Memory clean " + (Time.realtimeSinceStartup - ctime));
145.    }
146.
147.    private void UpdateBackgroundOrigin()
148.    {
149.        var backgroundPosition = this.background.transform.position;
150.        if (backgroundPosition.x < backgroundPosition.y)
151.        {
152.            this.backgroundOrigin = new Vector2(0, 420);
153.        }
154.        else
155.        {
```

```
156.            this.backgroundOrigin = new Vector2(420, 0);
157.        }
158.
159.        //Debug.Log("backgroundPosition " + backgroundPosition.x + " , " + backgroundPosition.y
    );
160.        //Debug.Log("backgroundOrigin " + backgroundOrigin.x + " , " + backgroundOrigin.y);
161.
162.    }
163.
164.    private void DrawBoxOutline(BoxOutline outline)
165.    {
166.
167.        var xMin = outline.XMin * this.backgroundSize.x + this.backgroundOrigin.x;
168.        var xMax = outline.XMax * this.backgroundSize.x + this.backgroundOrigin.x;
169.        var yMin = outline.YMin * this.backgroundSize.y + this.backgroundOrigin.y;
170.        var yMax = outline.YMax * this.backgroundSize.y + this.backgroundOrigin.y;
171.
172.        DrawRectangle(new Rect(xMin, yMin, (xMax - xMin), yMax - yMin), 4, outline.Clr);
173.
174.        DrawLabel(new Rect(xMin + 10, yMin + 10, 200, 20), $"{outline.Label}: {(int)(outline.Sc
    ore * 100)}% {outline.Depth}m", outline.Clr);
175.    }
176.
177.    public static void DrawRectangle(Rect area, int frameWidth, Color color)
178.    {
179.        // Create a one pixel texture with the right color
180.        if (boxOutlineTexture == null)
181.        {
182.            var texture = new Texture2D(1, 1);
183.            texture.SetPixel(0, 0, color);
184.            texture.Apply();
185.            boxOutlineTexture = texture;
186.        }
187.        else
188.        {
189.            boxOutlineTexture.SetPixel(0, 0, color);
190.            boxOutlineTexture.Apply();
191.        }
192.
193.        Rect lineArea = area;
194.        lineArea.height = frameWidth;
195.        GUI.DrawTexture(lineArea, boxOutlineTexture); // Top line
196.
197.        lineArea.y = area.yMax - frameWidth;
198.        GUI.DrawTexture(lineArea, boxOutlineTexture); // Bottom line
199.
200.        lineArea = area;
201.        lineArea.width = frameWidth;
202.        GUI.DrawTexture(lineArea, boxOutlineTexture); // Left line
203.
204.        lineArea.x = area.xMax - frameWidth;
205.        GUI.DrawTexture(lineArea, boxOutlineTexture); // Right line
206.    }
207.
208.    private static void DrawLabel(Rect position, string text,Color color)
209.    {
210.        if (labelStyle == null)
211.        {
212.            var style = new GUIStyle();
```

```
213.            style.fontSize = 50;
214.            style.normal.textColor = color;
215.            labelStyle = style;
216.        }
217.        else
218.        {
219.            labelStyle.normal.textColor = color;
220.        }
221.
222.        GUI.Label(position, text, labelStyle);
223.    }
224.
225.    private Texture2D TakeTextureSnap()
226.    {
227.        var smallest = 720;
228.        var tex =  getColorTexture();
229.        var snap = TextureTools.CropWithRect(tex,
230.            new Rect(0, 0, smallest, smallest),
231.            TextureTools.RectOptions.Center, 0, 0);
232.
233.        return snap;
234.    }
235.
236.    private Texture2D Scale(Texture2D texture, int imageSize)
237.    {
238.        var scaled = TextureTools.scaled(texture, imageSize, imageSize, FilterMode.Bilinear);
239.
240.        return scaled;
241.    }
242.
243.    private Task<Color32[]> RotateAsync(Color32[] pixels, int width, int height, int angle)
244.    {
245.        return Task.Run(() =>
246.        {
247.            //return TextureTools.RotateImageMatrix(pixels, width, height, -90);
248.            return TextureTools.RotateImageMatrix(pixels, width, height, angle);
249.        });
250.    }
251.
252.    void onSpeakingStart()
253.    {
254.        isSpeaking = true;
255.        Debug.Log("Speaking started");
256.    }
257.
258.    void onSpeakingStop()
259.    {
260.        isSpeaking = false;
261.        Debug.Log("Speaking stopped");
262.    }
263.
264.    string GenerateText(string id,double d)
265.    {
266.        System.Random rnd = new System.Random();
267.        string aid;
268.        if (id[0] == 'a' || id[0] == 'e' || id[0] == 'i' || id[0] == 'o' || id[0] == 'u' || id[0] == 'y')
269.            aid = "an " + id;
270.        else
271.            aid = "a " + id;
```

```
272.
273.        string[] txt = {$"There is {aid} in front of you" ,
274.                        $"There is {aid}",
275.                        $"I can see {aid}",
276.                        $"{id}" };
277.
278.        if (d != -1)
279.        {
280.            txt[0] += $"at approximately {d} meters";
281.            txt[1] += $" {d} meters away";
282.            txt[2] += $", {d} meters ahead";
283.            txt[3] += $", {d} meters away";
284.        }
285.        else
286.        {
287.            txt[3] += " ahead";
288.        }
289.
290.        return txt[(int)rnd.Next(4)];
291.    }
292.}
```

Παράρτημα Β

Σε αυτό το παράρτημα επισυνάπτεται ο κώδικας που χρησιμοποιήθηκε για την υλοποίηση με την βιβλιοθήκη TensorFlow Lite. Επειδή αυτή η υλοποίηση δεν χρησιμοποιεί την κάμερα TOF συμπεριλαμβάνεται ολόκληρος ο κώδικας που χρησιμοποιήθηκε για την ανίχνευση αντικειμένων.

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using System.IO;
4.  using UnityEngine;
5.  using UnityEngine.UI;
6.  using TensorFlowLite;
7.  using TextSpeech;
8.  using System;
9.  using System.Linq;
10.
11. public class SsdSample : MonoBehaviour
12. {
13.     [SerializeField, FilePopup("*.tflite")] string fileName = "coco_ssd_mobilenet_quant.tflite";
14.     [SerializeField] RawImage cameraView = null;
15.     [SerializeField] Text framePrefab = null;
16.     [SerializeField, Range(0f, 1f)] float scoreThreshold = 0.5f;
17.     [SerializeField] TextAsset labelMap = null;
18.     [SerializeField] TextAsset colorMap = null;
19.
20.     [SerializeField, Range(0.5f, 2)] float pitch = 1f; //[0.5 - 2] Default 1
21.     [SerializeField, Range(0.5f, 2)] float rate = 1f; //[min - max] android:[0.5 - 2] iOS:[0 - 1]
22.     [SerializeField] string code = "en-US";
23.
24.     WebCamTexture webcamTexture;
25.     SSD ssd;
26.
27.     Text[] frames;
28.     private string[] labels;
29.     private Color[] colors;
30.
31.     private bool isSpeaking = false;
32.     private double[] obj;
33.     private int winner;
34.
35.     void Start()
36.     {
37.         string path = Path.Combine(Application.streamingAssetsPath, fileName);
38.         ssd = new SSD(path);
39.         TextToSpeech.instance.Setting(code, pitch, rate);
40.         TextToSpeech.instance.onStartCallBack = onSpeakingStart;
41.         TextToSpeech.instance.onDoneCallback = onSpeakingStop;
42.
43.         // Init camera
44.         string cameraName = WebCamUtil.FindName();
45.         webcamTexture = new WebCamTexture(cameraName, 1280, 720, 30);
46.         cameraView.texture = webcamTexture;
```

```
47.         webcamTexture.Play();
48.         Debug.Log($"Starting camera: {cameraName}");
49.
50.         // Init frames
51.         frames = new Text[10];
52.         var parent = cameraView.transform;
53.         for (int i = 0; i < frames.Length; i++)
54.         {
55.             frames[i] = Instantiate(framePrefab, Vector3.zero, Quaternion.identity, parent);
56.         }
57.
58.         // Labels
59.         labels = labelMap.text.Split('\n');
60.         obj = new double[labels.Length];
61.
62.         // Colors
63.         string[] lines = colorMap.text.Split('\n');
64.         colors = new Color[labels.Length];
65.         for(int i= 0; i < labels.Length; i++){
66.             string[] line = lines[i].Split(' ');
67.             colors[i] = new Color32(byte.Parse(line[0]), byte.Parse(line[1]), byte.Parse(line[2]),255);
68.         }
69.     }
70.
71.     void OnDestroy()
72.     {
73.         webcamTexture?.Stop();
74.         ssd?.Dispose();
75.     }
76.
77.     void Update()
78.     {
79.         ssd.Invoke(webcamTexture);
80.         var results = ssd.GetResults();
81.         var size = cameraView.rectTransform.rect.size;
82.         for (int i = 0; i < 10; i++)
83.         {
84.             SetFrame(frames[i], results[i], size);
85.             CountResult(results[i]);
86.         }
87.
88.         obj[winner] = 0;
89.         if (!isSpeaking && obj.Max() > 0)
90.         {
91.             isSpeaking = true;
92.             winner = obj.ToList().IndexOf(obj.Max());
93.             Array.Clear(obj, 0, obj.Length);
94.             string txt = GenerateText(GetLabelName(winner));
95.             TextToSpeech.instance.StartSpeak(txt);
96.         }
97.         // cameraView.material = ssd.transformMat;
98.          cameraView.texture = ssd.inputTex;
99.     }
100.
101.    void SetFrame(Text frame, SSD.Result result, Vector2 size)
102.    {
103.        if (result.score < scoreThreshold)
104.        {
105.            frame.gameObject.SetActive(false);
```

```
106.            return;
107.        }
108.        else
109.        {
110.            frame.gameObject.SetActive(true);
111.            Debug.Log(result.classID + " " + GetLabelName(result.classID) + " " + result.score);
112.        }
113.
114.        frame.text = $"{GetLabelName(result.classID)} : {(int)(result.score * 100)}%";
115.        var rt = frame.transform as RectTransform;
116.        rt.anchoredPosition = result.rect.position * size - size * 0.5f;
117.        rt.sizeDelta = result.rect.size * size;
118.
119.        frame.color = colors[result.classID]; //text color
120.        rt.GetChild(0).GetComponent<Image>().color = colors[result.classID]; //border color
121.
122.    }
123.
124.    void CountResult(SSD.Result result)
125.    {
126.        if (result.score < scoreThreshold) return;
127.        else
128.        {
129.            int index;
130.            if (result.classID < 0 || result.classID > labels.Length)
131.                index = 0;
132.            else
133.                index = result.classID;
134.            obj[index] += result.score;
135.        }
136.    }
137.
138.    string GenerateText(string id)
139.    {
140.        System.Random rnd = new System.Random();
141.        string aid;
142.        if (id[0] == 'a' || id[0] == 'e' || id[0] == 'i' || id[0] == 'o' || id[0] == 'u' || id[0] == 'y')
143.            aid = "an " + id;
144.        else
145.            aid = "a " + id;
146.
147.        string[] txt = {$"There is {aid} in front of you" ,
148.                        $"There is {aid}",
149.                        $"I can see {aid}",
150.                        $"{id} ahead"};
151.
152.        return txt[(int)rnd.Next(4)];
153.    }
154.
155.    string GetLabelName(int id)
156.    {
157.        if (id < 0 || id >= labels.Length - 1)
158.        {
159.            return "?";
160.        }
161.        return labels[id + 1];
162.    }
163.
164.    void onSpeakingStart()
```

```
165.    {
166.        isSpeaking = true;
167.        Debug.Log("Speaking started");
168.    }
169.
170.    void onSpeakingStop()
171.    {
172.        isSpeaking = false;
173.        Debug.Log("Speaking stopped");
174.    }
175.}
```