3D Scene Segmentation & Detection

Καλαϊτζόπουλος Βασίλειος, Α.Μ. 1066670, 5ο Έτος

Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών, Πανεπιστήμιο Πατρών

3Δ Υπολογιστική Γεωμετρία & Όραση

9 Ιουλίου, 2023

Table of Contents

Περίληψη

Η εργασία αυτή εξετάζει τις διαφορές που προκύπτουν κατά την επεξεργασία των point cloud που δημιουργούνται με την χρήση του disparity map το οποίο υπολογίζεται για ζεύγη στερεοσκοπικών εικόνων και αυτών που παίρνουμε με την χρήση lidar αισθητήρων ή είναι έτοιμα μοντέλα. Στο stereo reconstruction εφαρμόζονται και κάποιες τεχνικές denoising στα δεδομένα του point cloud, στις αρχικές εικόνες αλλά και στο disparity map. Έπειτα, παρουσιάζονται τα αποτελέσματα από τον εντοπισμό των επίπεδων επιφανειών και από το clustering που υλοποιείται στα point cloud που παίρνουμε και από τις δύο τεχνικές. Επιπλέον, παρουσιάζεται ένα τύπου virtualization των αντικειμένων της σκηνής με την χρήση των Axis Aligned Bounding Boxes (AABB) τους και μίας μπάλας που προστίθεται και μετακινείται στην σκηνή. Τέλος, υλοποιείται και ένας αλγόριθμος triangulation, Ball Pivot Algorithm, ο οποίος προσπαθεί να δημιουργήσει ένα mesh από ένα point cloud.

Keywords: Stereo reconstruction, disparity map, denoising, plane detection, RANSAC, clustering, DBSCAN, scene segmentation, scene virtualization, object-based triangulation, ball pivot algorithm.

3D Scene Segmentation & Detection

Stereo Reconstruction

Σε αυτό το μέρος της εργασίας θα αναλυθεί η διαδικασία που ακολουθήθηκε για την δημιουργία ενός point cloud από ένα ζεύγος στερεοσκοπικών εικόνων.

Επιλογή στερεοσκοπικών εικόνων

Χρησιμοποιήθηκαν τέσσερα stereo pairs για την εξαγωγή point cloud στην εργασία αυτή. Τα τρία από αυτά είναι από το stereo evaluation 2015 dataset του KITTI. Το τέταρτο είναι αυτό που μας δόθηκε για το vision εργαστήριο. Για το ζεύγος που μας δόθηκε στο εργαστήριο οι παράμετροι που χρειαζόμασταν ήταν γνωστές (fov = 1.2, baseline = 0.2) ενώ για τα ζεύγη από το dataset του KITTI βρέθηκε ότι έχουμε fov = 1.0 και baseline = 0.54.

Υπολογισμός disparity map

Για τον υπολογισμό του disparity map χρησιμοποιήθηκε ο block matching αλγόριθμος. Η λογική του αλγόριθμού αυτού είναι αφού έχουν εισάγει τις εικόνες στον κώδικα μας, να παίρνουμε ένα window ή block από την αριστερή εικόνα και έπειτα να ψάχνουμε πάνω στην επιπολική ευθεία πάνω στην δεξιά εικόνα για το block το οποίο κάνει καλύτερο matching με το block της αριστερής εικόνας. Το βολικό είναι ότι οι κάμερες είναι παράλληλες οπότε οι επιπλοκές ευθείες είναι οριζόντιες και έτσι απλά κινούμαστε στις γραμμές των πινάκων που έχουμε για την αριστερή και την δεξιά εικόνα για να βρούμε το καλύτερο match. Παρακάτω φαίνονται αναλυτικά τα βήματα του αλγόριθμου.

Block matching algorithm:

1. *Πάρε ένα template από την αριστερή εικόνα ανάλογα με το block size που έχει οριστεί*
2. *Βρες την αρχική και την τελική στήλη από την ίδια γραμμή της δεξιάς εικόνας για να πάρεις τα blocks με τα οποία θα συγκρίνεις το template*

3. *Υπολόγισε την διαφορά μεταξύ του template και όλων των blocks από το κομμάτι της γραμμής της δεξιάς εικόνας*
4. *Βρες το block με το οποίο η διαφορά είναι μικρότερη*
5. *Βρες πόσες θέσεις απέχει το column του template από το column του block (disparity)*
6. *Τοποθέτησε την απόσταση μεταξύ των columns στην θέση που αντιστοιχεί στο αρχικό template (row, column) σε έναν πίνακα ίδιων διαστάσεων με την εικόνα (disparity map)*
7. *Επανέλαβε την διαδικασία για όλα τα template της αριστερής εικόνας*

Αποτελέσματα υπολογισμού disparity map. Παρακάτω παρουσιάζονται τα disparity map που προέκυψαν για το εκάστοτε stereo pair.

- Πρώτο ζεύγος:

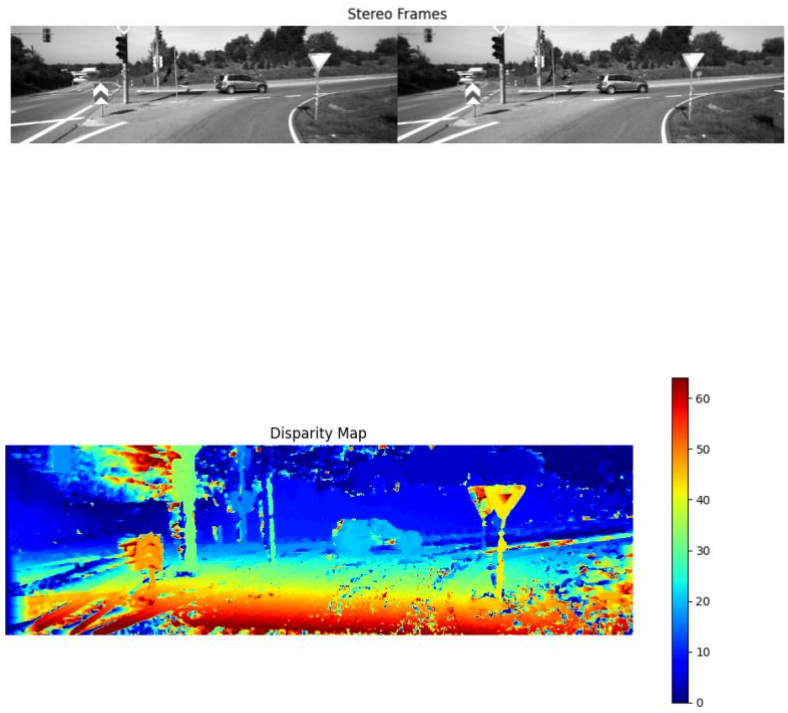

Figure 1: Πρώτο stereo pair και disparity map.

- Δεύτερο ζεύγος:

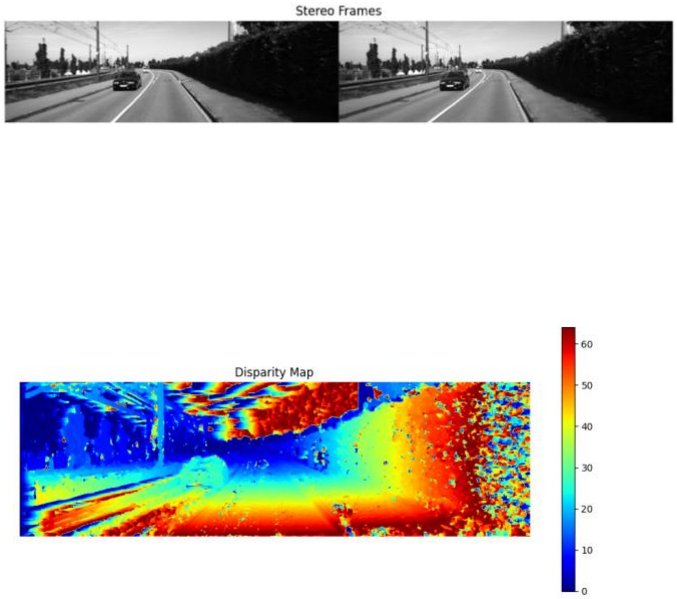

Figure 2: Δεύτερο stereo pair και disparity map.

- Τρίτο ζεύγος:

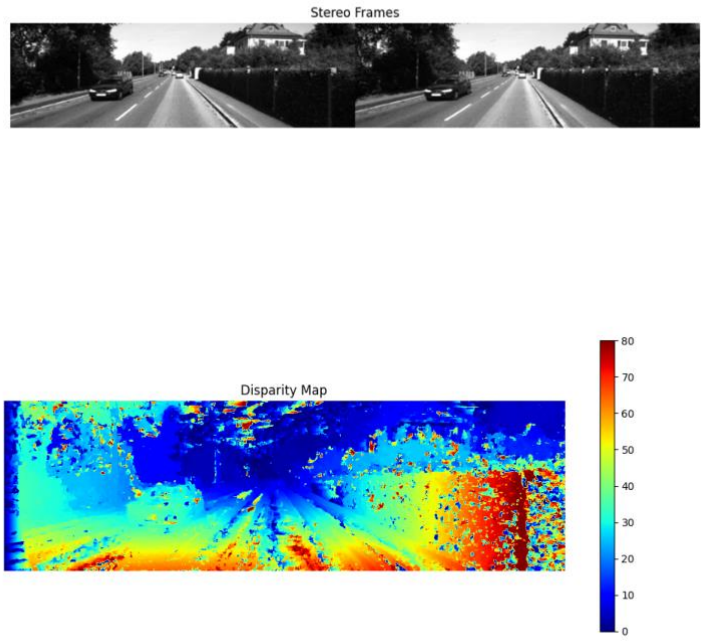

Figure 3: Τρίτο stereo pair και disparity map.

- Τέταρτο ζεύγος:

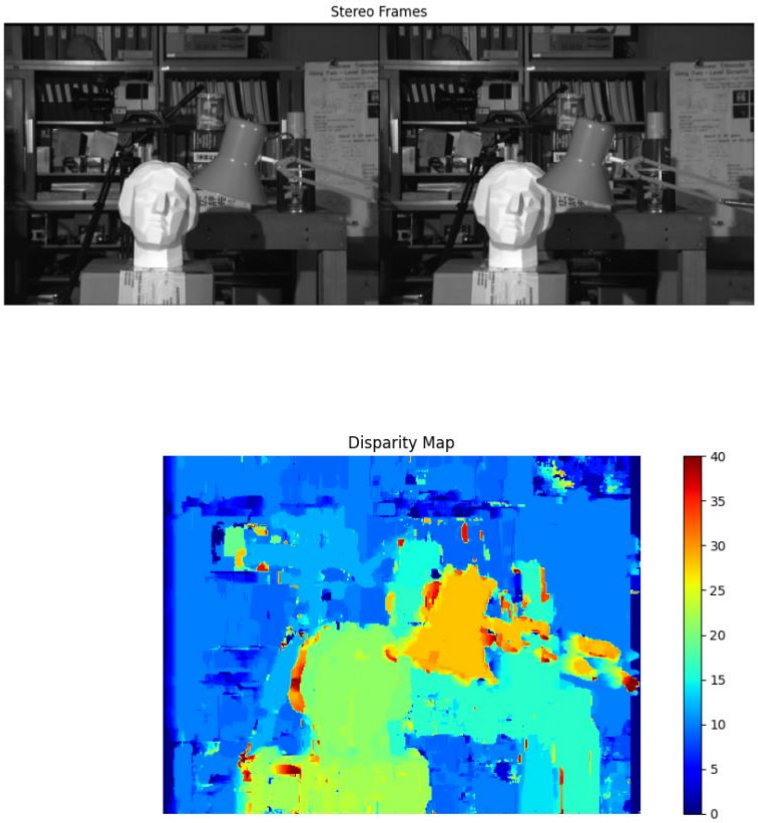

Figure 4: Τέταρτο stereo pair και disparity map.

Μετατροπή του disparity map σε point cloud

Σε αυτό το σημείο έχουμε υπολογίσει το disparity map για το stereo pair που έχουμε επιλέξει. Για να πάμε τώρα σε συντεταγμένες στον τρισδιάστατο χώρο χρησιμοποιούμε είτε το baseline (απόσταση μεταξύ των δύο καμερών) και το focal length το οποίο υπολογίζουμε μέσω του fov που έχουμε για το ζεύγος που της αποθήκης και από τους παρακάτω τύπους παίρνουμε τις συντεταγμένες στον τρισδιάστατο χώρο:

- Συντεταγμένη z (depth) :

$$z = depth = focal\ length * \frac{baseline}{disparity}$$

- Συντεταγμένη x:

$$x = \frac{disparity\ column - center\ column}{number\ of\ columns} * \frac{depth}{focal\ length}$$

- Συντεταγμένη y:

$$y = \frac{disparity\ row - center\ row}{number\ of\ rows} * \frac{depth}{focal\ length}$$

Να σημειωθεί ότι κρατάμε μόνο τα σημεία για τα οποία προκύπτει θετικό z και τους δίνουμε το χρώμα που είχαμε στην αριστερή εικόνα για να τα δείξουμε στο visualization.

Σε αντίθεση με το stereo pair της αποθήκης για τα pairs από το dataset του KITTI δεν γνωρίζουμε το fov έτσι ώστε να κάνουμε την μετατροπή στον τρισδιάστατο χώρο με το focal length. Γνωρίζουμε όμως τα calibration matrices των καμερών οπότε χρησιμοποιούμε αυτά για υπολογίσουμε τον projection matrix και έπειτα τις συντεταγμένες στον τρισδιάστατο χώρο. Τα matrices αυτά βρίσκονται στο directory /data/calib/00XXXX_XX.txt στις μεταβλητές P2 και P3 για την αριστερή και την δεξιά κάμερα αντίστοιχα.

Αποτελέσματα μετατροπής σε point cloud. Παρακάτω παρουσιάζονται τα point cloud που προέκυψαν με από το εκάστοτε disparity map πριν από οποιοδήποτε denoising.

- Πρώτο point cloud:

Figure 5: Πρώτο point cloud από stereo pair.

- Δεύτερο point cloud:

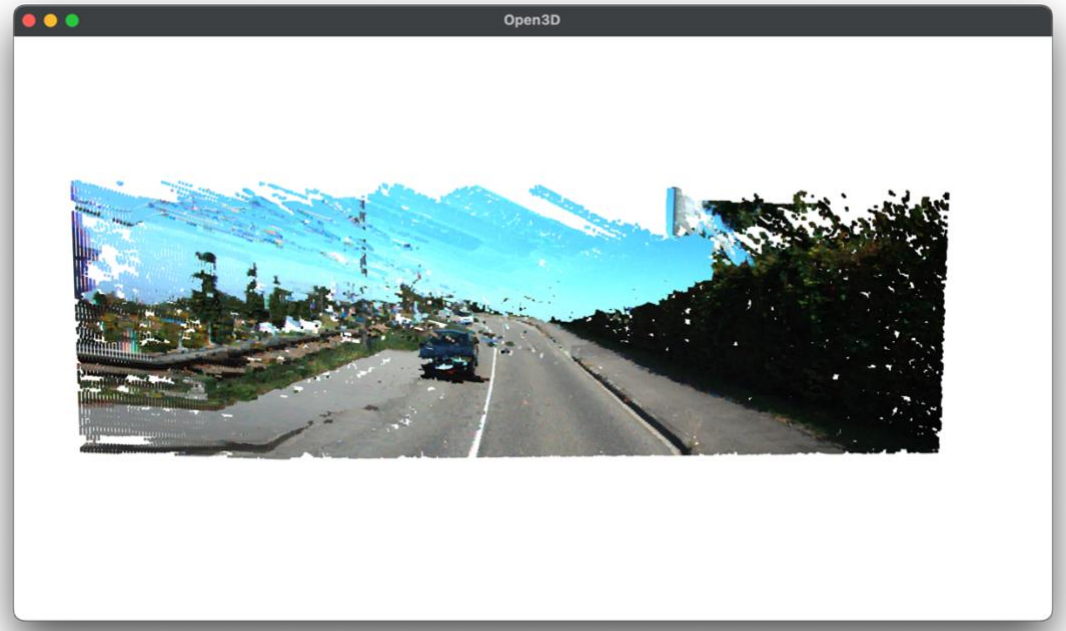

Figure 6: Δεύτερο point cloud από stereo pair.

- Τρίτο point cloud:

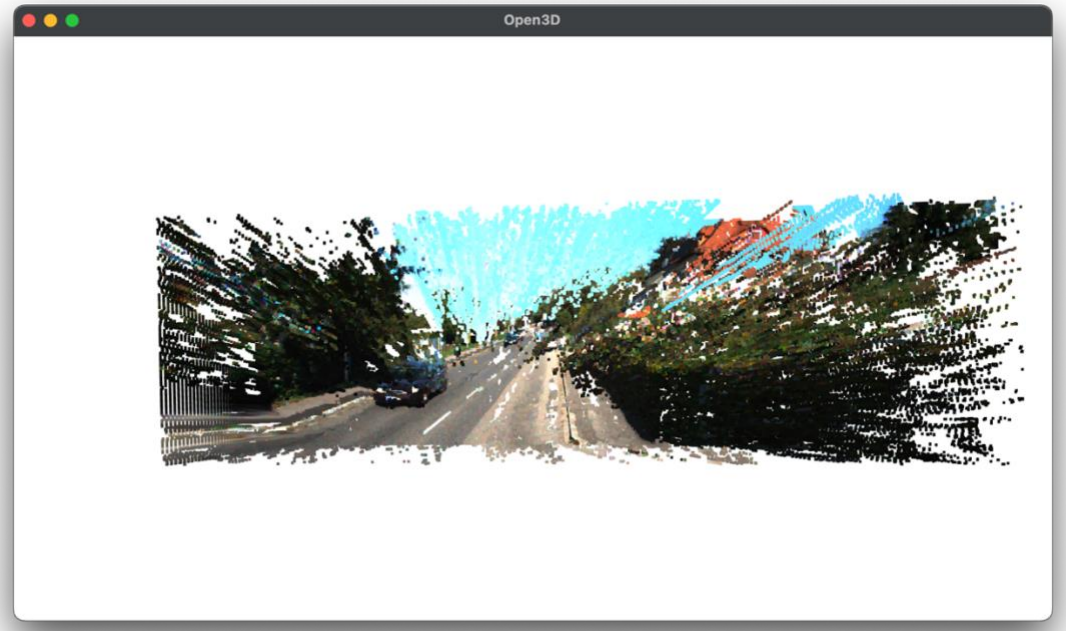

Figure 7: Τρίτο point cloud από stereo pair.

- Τέταρτο point cloud:

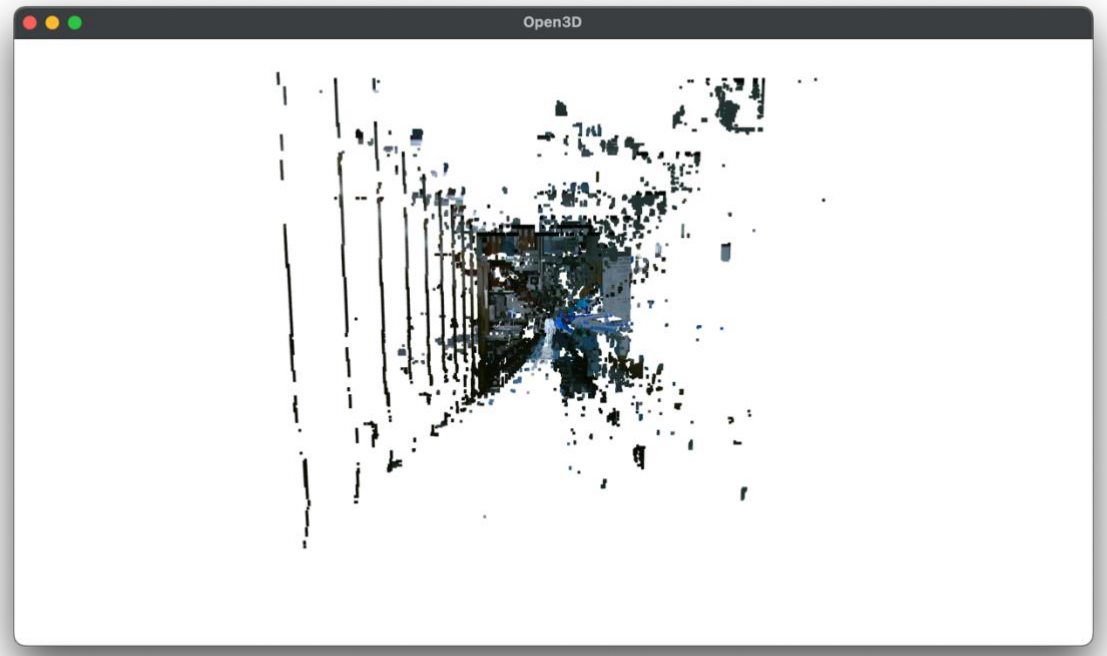

Figure 8: Τέταρτο point cloud από stereo pair.

Από τα αποτελέσματα φαίνεται ξεκάθαρα πως επηρεάζει το μέγεθος της εικόνας και τα disparity values που βρίσκουμε. Συγκεκριμένα, βλέπουμε ότι για τις εικόνες από το dataset του KITTY έχουμε πολλά περισσότερα σημεία το οποίο ήταν αναμενόμενο γιατί είναι εικόνες με μεγαλύτερη ανάλυση από αυτές τις αποθήκης. Επιπλέον βλέπουμε ότι για τις εικόνες του KITTI προκύπτουν πολλά περισσότερα disparity values με αποτέλεσμα να έχουμε πιο πυκνές «φέτες» μετά την μετατροπή σε συντεταγμένες στον τρισδιάστατο χώρο. Οι περισσότερες τιμές αυτές οφείλονται στο βάθος που εντοπίζεται στις εξωτερικές σκηνές σε αντίθεση με τη σκηνή τις αποθήκης που οι αλλαγές βάθους δεν είναι τόσες πολλές.

Αποθορυβοποίηση

Εφαρμόστηκαν τρεις μέθοδοι denoising. Συγκεκριμένα εφαρμόστηκε gaussian smoothing πάνω στις αρχικές φωτογραφίες για να δούμε πως επηρεάζει ο θόρυβος κατά τον υπολογισμό του disparity map, gaussian smoothing πάνω στο ίδιο τι disparity map με σκοπό να

εξομαλυνθούν στα spikes που παρατηρήθηκαν (τα οποία είναι θόρυβος) και μια μέθοδος clustering πάνω στο point cloud έτσι ώστε να εντοπιστούν τα σημεία που είναι ομαδοποιημένα και τα σημεία που είναι θόρυβος και να πεταχτούν.

Gaussian smoothing στα frames. Εδώ πρακτικά εφαρμόστηκε ένα 3x3 kernel και στα δύο αρχικά frame το οποίο έκανε μία gaussian εξομάλυνση στις εικόνες για να μειώσει τυχόν θόρυβο. Αυτή η μέθοδος δεν προκάλεσε καμία διαφορά στα disparity map γι' αυτό και δεν παρουσιάζονται καθώς είναι ίδια με τα αρχικά. Η μέθοδος αυτή ίσως να ήταν πιο αποτελεσματική αν οι φωτογραφίες είχαν αρκετό θόρυβο έτσι ώστε να είναι ουσιαστική η αλλαγή που θα προκαλούσε το φίλτρο.

Gaussian smoothing στα disparity maps. Η λογική πίσω από την δεύτερη μέθοδο ήταν να χρησιμοποιηθεί ένα gaussian φίλτρο έτσι ώστε να μειωθεί ο θόρυβος που πρόκυπτε σε «δύσκολες» περιοχές όπου δεν μπορούσε να υπολογιστεί σωστά το disparity, όπως δέντρα, ουρανός κλπ. Ο λόγος που σε αυτές τις περιοχές δε υπολογίζεται σωστά το disparity value οφείλεται στην χρήση μικρού παραθύρου το οποίο επιλέγουμε για να μεγαλύτερη λεπτομέρεια στο disparity map. Ωστόσο το μικρό παράθυρο πολλές φορές κάνει match με λάθος περιοχές από το δεξί frame γιατί δεν έχει τόση πληροφορία όσο ένα μεγαλύτερο παράθυρο το οποίο παρόλο που θα οδηγούσε σε πιο smooth αποτελέσματα θα θυσίαζε αρκετή λεπτομέρεια με αποτέλεσμα να μην καταλαβαίνουμε τι απεικονίζεται στο disparity map. Στα αποτελέσματα παρουσιάζονται μόνο τα disparity map καθώς η βελτίωση που προκάλεσε η μέθοδος δεν ήταν τόσο δραματική έτσι ώστε να γίνεται αισθητή και κατά την μετατροπή σε point cloud.

Αποτελέσματα smoothing στα disparity maps. Παρακάτω παρουσιάζονται τα disparity map έπειτα από την εφαρμογή gaussian φίλτρου.

- Πρώτο denoised disparity map:

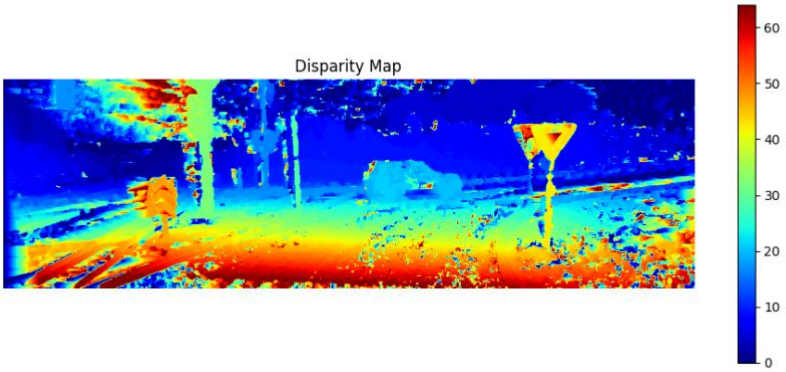

Figure 9:Πρώτο disparity map denoised με gaussian smoothing.

- Δεύτερο denoised disparity map:

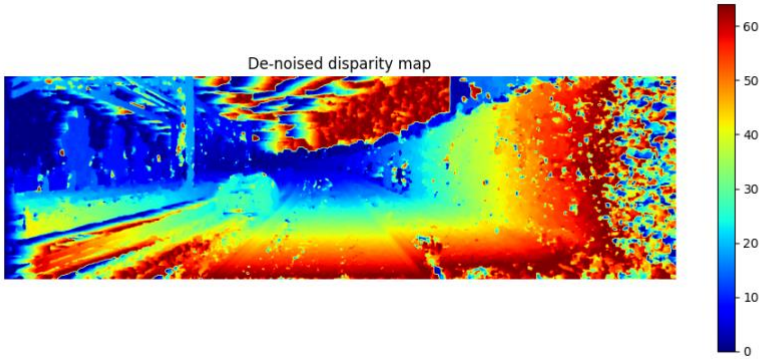

Figure 10: Δεύτερο disparity map denoised με gaussian smoothing.

- Τρίτο denoised disparity map:

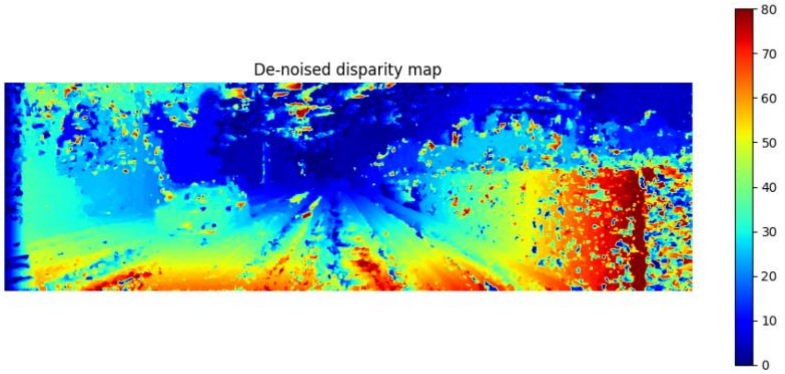

Figure 11:Τρίτο disparity map denoised με gaussian smoothing.

- Τέταρτο denoised disparity map:

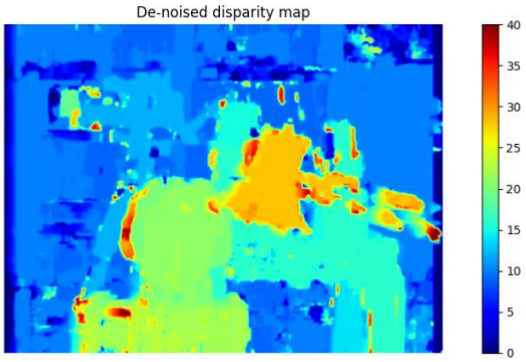

Figure 12: Τέταρτο disparity map denoised με gaussian smoothing.

Όπως φαίνεται στα αποτελέσματα την καλύτερη βελτίωση την βλέπουμε στο τέταρτο set με μικρή απώλεια ths αρχικής λεπτομέρειας του disparity map. Είναι λογικό να είναι πιο δραστικό το αποτέλεσμα σε αυτό το ζεύγος γιατί οι φωτογραφίες του είναι μικρότερης ανάλυσης από τις άλλες οπότε το 5x5 φίλτρο έχει μεγαλύτερη «βαρύτητα». Ωστόσο, άμα είχε χρησιμοποιηθεί μεγαλύτερο φίλτρο, ώστε να δούμε κάποια ουσιαστική βελτίωση και στα υπόλοιπα ζεύγη θα χανόταν περισσότερη λεπτομέρεια το οποίο δεν είναι επιθυμητό.

Ομαδοποίηση σημείων. Στην τελευταία μέθοδο denoising που εφαρμόστηκε χρησιμοποιήθηκε ο αλγόριθμος ομαδοποίησης DBSCAN. Ο αλγόριθμος αυτός χρειάζεται δύο παραμέτρους. Την παράμετρο epsilon η οποία είναι η ακτίνα της γειτονιάς που θέλουμε να είναι οι γείτονες του σημείου που εξετάζεται και την παράμετρο minimum samples η οποία δηλώνει πόσα σημεία τουλάχιστον χρειάζονται για να κρατήσουμε μία ομάδα σημείων και να μην θεωρηθούν θόρυβος. Τα βήματα του αλγόριθμου περιγράφονται παρακάτω.

Αλγόριθμος DBSCAN:

1. *Πάρε το σύνολο με όλα τα σημεία*
2. *Πήγαινε σε ένα τυχαίο σημείο που δεν έχεις ξαναπάει*
3. *Δες πόσα σημεία βρίσκονται σε απόσταση μικρότερη από το epsilon από το σημείο αυτό*

4. *Αν είναι περισσότερα από το minimum samples θεώρησε τα μία ομάδα και κάνε το ίδιο για όλα τα σημεία που προστέθηκαν στην ομάδα μέχρι να μην προστεθούν άλλα*
5. *Αν είναι λιγότερα από το minimum samples θεώρησε το σημείο σαν θόρυβο και πήγαινε σε ένα καινούργιο σημείο*

Αποτελέσματα ομαδοποίησης των σημείων. Παρακάτω παρουσιάζονται τα αποτελέσματα από την ομαδοποίηση.

- Πρώτο point cloud:

Figure 13: Πρώτο denoised point cloud.

- Δεύτερο point cloud:

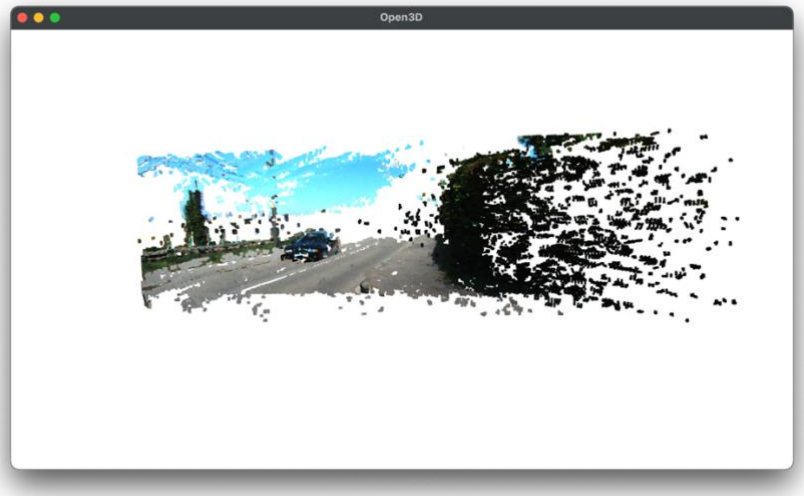

Figure 14: Δεύτερο denoised point cloud.

- Τρίτο point cloud:

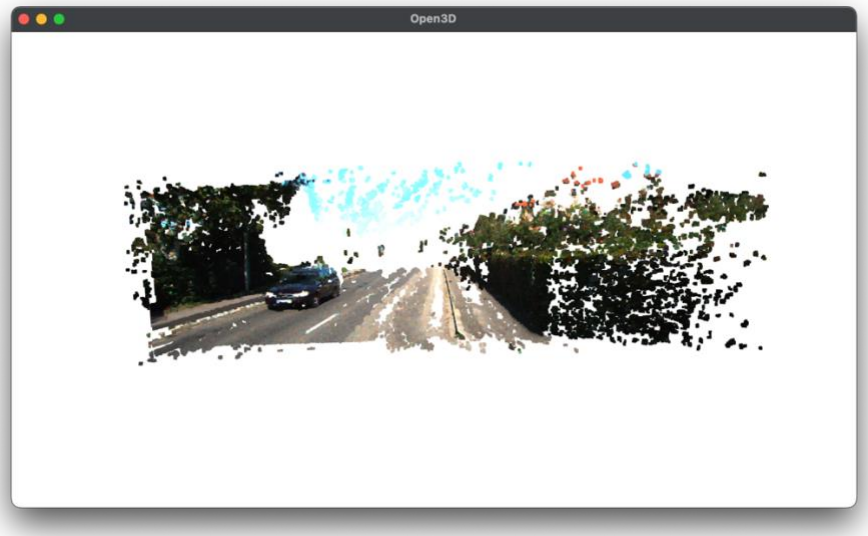

Figure 15: Τρίτο denoised point cloud.

- Τέταρτο point cloud:

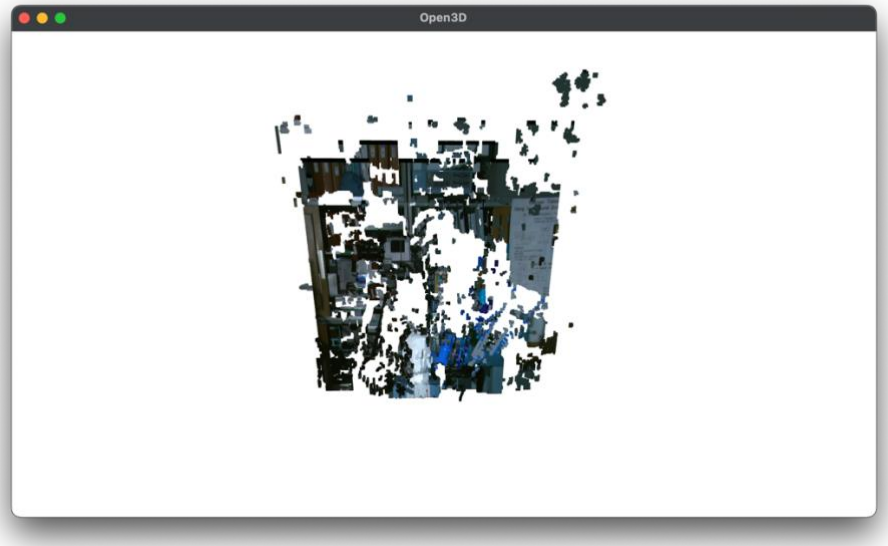

Figure 16: Τέταρτο denoised point cloud.

Παρατηρούμε ότι αυτή η μέθοδος είναι η πιο αποτελεσματική από όλες τις μεθόδους. Στις σκηνές με τα αυτοκίνητα βλέπουμε πως είναι αρκετά καλή έτσι ώστε να πετάξει ένα μεγάλο ποσοστό του θορύβου που προέκυψε από τα δύσκολα σημεία στις φωτογραφίες που αναφέρθηκαν και παραπάνω ενώ κρατάει τα αντικείμενα όπως τα αμάξια άθικτα. Επίσης

στην σκηνή με την αποθήκη βλέπουμε ότι ο περισσότερος θόρυβος έχει φύγει ενώ επειδή και το disparity map δεν είχε πολύ θόρυβο τα αντικείμενα της σκηνής είναι αρκετά ευδιάκριτα το οποίο φαίνεται και καλύτερα στην ενότητα object detection.

Scene Segmentation

Σε αυτό το κομμάτι της εργασίας σχολιάζονται οι διαφορές εντοπισμού επίπεδων επιφανειών (π.χ. τοίχους, δρόμους) μεταξύ ενός point cloud που έχει υπολογιστεί μέσω stereo reconstruction και ενός που είτε έχει υπολογιστεί με την χρήση ενός lidar scanner είτε το έχει φτιάξει κάποιος. Έπειτα, θα αναπτυχθεί και μία μέθοδος εντοπισμού αντικειμένων στην σκηνή αφού έχουν αφαιρεθεί οι επίπεδες επιφάνειες.

Plane detection

Για την ανίχνευση επίπεδων χρησιμοποιήθηκε ένας αλγόριθμος RANSAC (random sample consensus). Η λογική πίσω από τον αλγόριθμο είναι να εντοπίσει ποια σημεία του point cloud «ταιριάζουν» καλύτερα σε κάποιο συγκεκριμένο μοντέλο, στην δική μας περίπτωση ένα επίπεδο. Οπότε, για να εφαρμόσουμε τον αλγόριθμο πρέπει να ορίσουμε το επίπεδο έτσι ώστε να το χρησιμοποιήσουμε. Συγκεκριμένα, μπορούμε να υπολογίσουμε την εξίσωση ενός επιπέδου αν γνωρίζουμε τρία σημεία τα οποία ανήκουν σε αυτό. Αρχικά, παίρνουμε τρία σημεία, έστω P0, P1, P2 και υπολογίζουμε το normal του επιπέδου παίρνοντας το εξωτερικό γινόμενο των διανυσμάτων που σχηματίζουν το P0 με το P1 και το P0 με το P2. Το normal αποτελείται από τις παραμέτρους a, b, c της εξίσωσης του επιπέδου. Αφού γνωρίζουμε αυτές και ένα σημείο του επιπέδου (από τα τρία αρχικά) μπορούμε να υπολογίσουμε και το d όπως φαίνεται παρακάτω.

$$d = -(ax_1 + by_1 + cz_1)\text{ , όπου}$$

$$normal = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \text{ και } P1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

Τέλος, μπορούμε να υπολογίσουμε την απόσταση ενός σημείου P από το επίπεδο παίρνοντας το μήκος της προβολής του διανύσματος του σημείου P (x_0, y_0, z_0) στο μοναδιαίο normal του επιπέδου.

$$distance = \frac{|\vec{n} \cdot \vec{p}|}{|\vec{n}|} = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}}$$

Έτσι μπορούμε να παίρνουμε τρία τυχαία σημεία από αυτά του point cloud, να βρίσκουμε το επίπεδο που περνάει από αυτά και έπειτα να υπολογίζουμε την απόσταση όλων το σημείων από αυτό το plane. Τέλος, θα ελέγχουμε ποια σημεία έχουν απόσταση μικρότερη από ένα threshold που ορίζουμε εμείς τα οποία είναι οι inliers και τα σημεία που έχουν απόσταση μεγαλύτερη από το threshold είναι οι outliers. Αφού δοκιμάσουμε αρκετά επίπεδα θα κρατάμε αυτό με το καλύτερο score το οποίο είναι αυτό με τους περισσότερους inliers και θα θεωρούμε ότι η επίπεδη επιφάνεια (τοίχος, δρόμος) αποτελείται από αυτούς τους inliers.

Αποτελέσματα RANSAC για point clouds από stereo pairs. Εφαρμόστηκε εντοπισμός επιπέδων μόνο στα point cloud που προέκυψαν από τα πρώτα τρία stereo pairs καθώς στο τέταρτο δεν υπήρχε κάποιος δρόμος η τοίχος να εντοπιστεί.

- Πρώτο stereo point cloud:

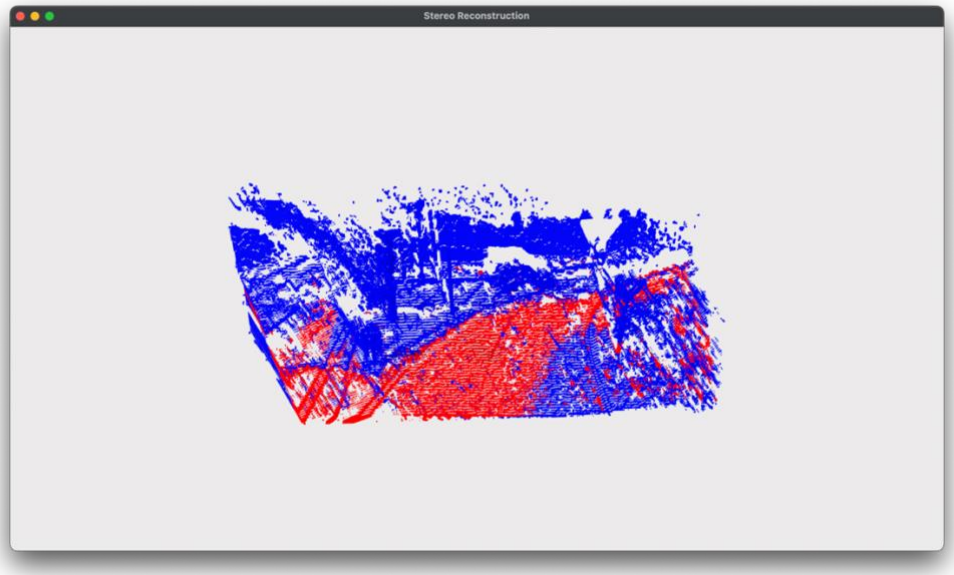

Figure 17: Εντοπισμός μίας επίπεδης επιφάνειας στο πρώτο stereo point cloud.

- Δεύτερο stereo point cloud:

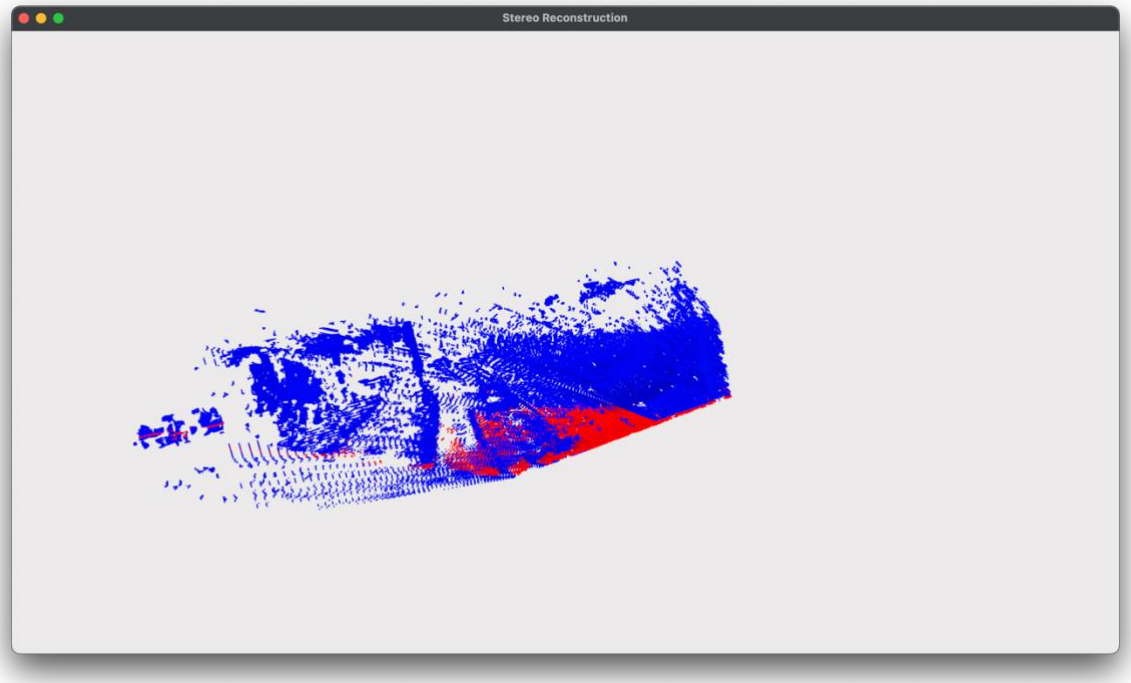

Figure 18: Εντοπισμός μίας επίπεδης επιφάνειας στο δεύτερο stereo point cloud.

- Τρίτο stereo point cloud:

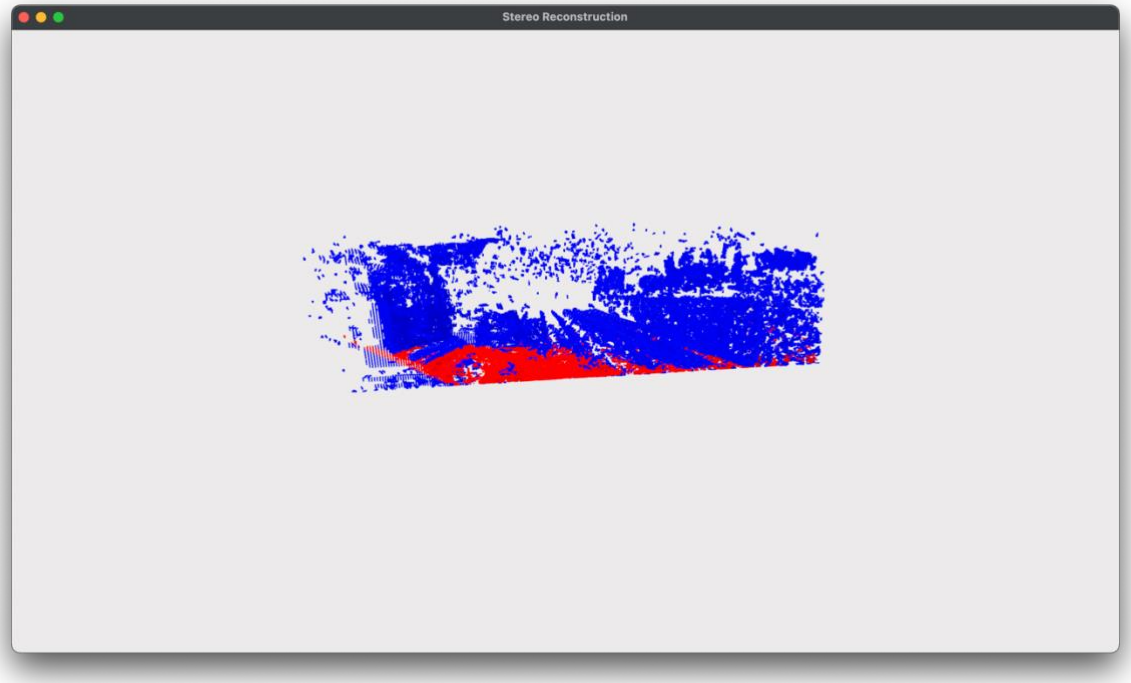

Figure 19: Εντοπισμός μίας επίπεδης επιφάνειας στο τρίτο point cloud.

Παρατηρούμε πως η μέθοδος βρίσκει πολύ καλά τους δρόμους και στις τρεις περιπτώσεις και ιδιαίτερα στην πρώτη όπου κρατάει και την στροφή. Να σημειωθεί ότι δεν παίρνουμε πάντα αυτά τα αποτελέσματα δεδομένης της τυχαιότητας του αλγόριθμου και ότι η μέθοδος εφαρμόζεται στα point clouds αφού εφαρμοστεί denoising με την μέθοδο της ομαδοποίησης που αναφέρθηκε παραπάνω. Είναι σημαντικό να εφαρμόζεται μετά από το denoising καθώς οι δρόμοι διατηρούνται σε αρκετά καλή κατάσταση και φεύγουν πολλά σημεία τα οποία ήταν θόρυβος. Τέλος παρατηρούμε ότι χάνουμε κάποια κομμάτια του δρόμου το οποίο οφείλεται στο μικρό distance threshold που έχουμε ορίσει. Ωστόσο, αν αυξήσουμε το threshold για να κρατήσουμε και αυτά τα κομμάτια είναι πιθανό planes τα οποία δεν ανήκουν σε κάποια επίπεδη επιφάνεια να πάρουν μεγαλύτερο score γιατί θα έχουν πολλά σημεία κοντά ντους τα οποία θα θεωρούνται inliers λόγω του μεγάλου distance threshold.

Αποτελέσματα RANSAC για artificial point clouds. Η μέθοδος εφαρμόστηκε και σε artificial point clouds. Σε αυτού του είδους τα point cloud τα σημεία είναι τοποθετημένα τέλεια οπότε έχουμε σκηνές, όπως ένα δωμάτιο με ένα γραφείο, στις οποίες είναι πολύ εύκολο για τον αλγόριθμο να εντοπίσει τις επίπεδες επιφάνειες.

- Πρώτο artificial point cloud

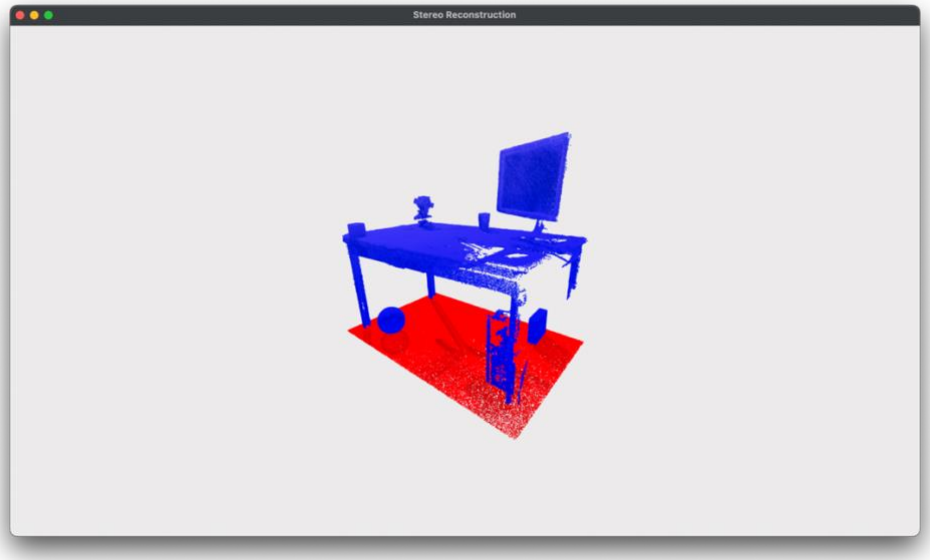

Figure 20: Εντοπισμός μίας επίπεδης επιφάνειας στο πρώτο artificial point cloud.

- Δεύτερο artificial point cloud

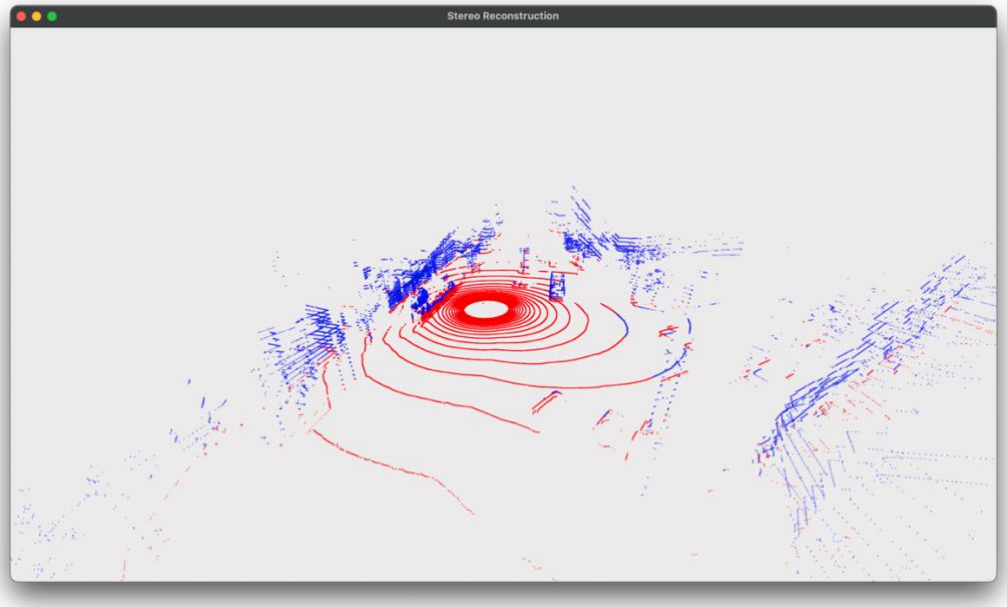

Figure 21: Εντοπισμός μίας επίπεδης επιφάνειας στο δεύτερο artificial point cloud.

- Τρίτο artificial point cloud

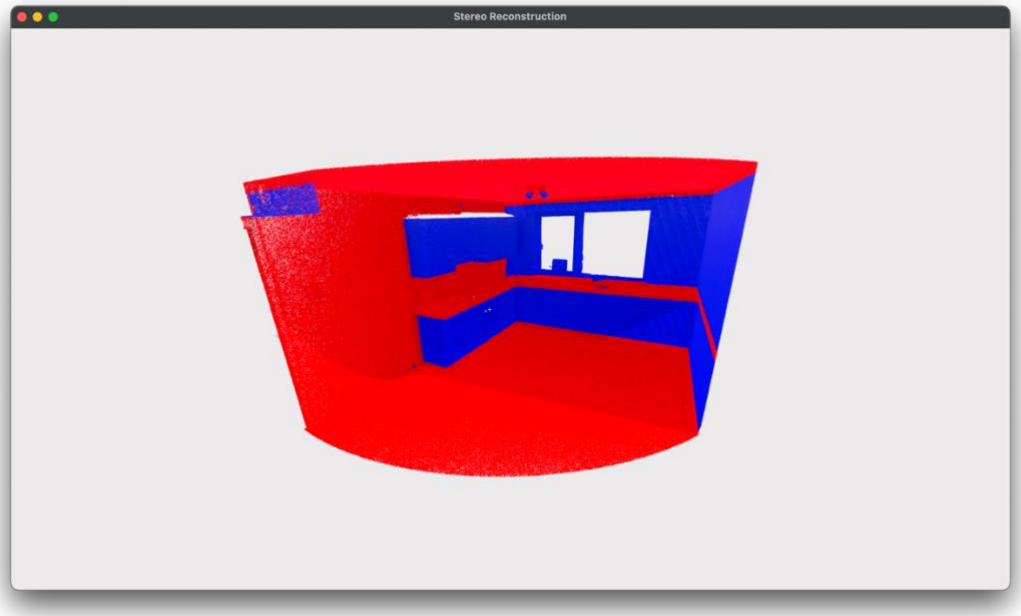

Figure 22 (α): Εντοπισμός τεσσάρων επιφανειών στο τρίτο artificial point cloud.

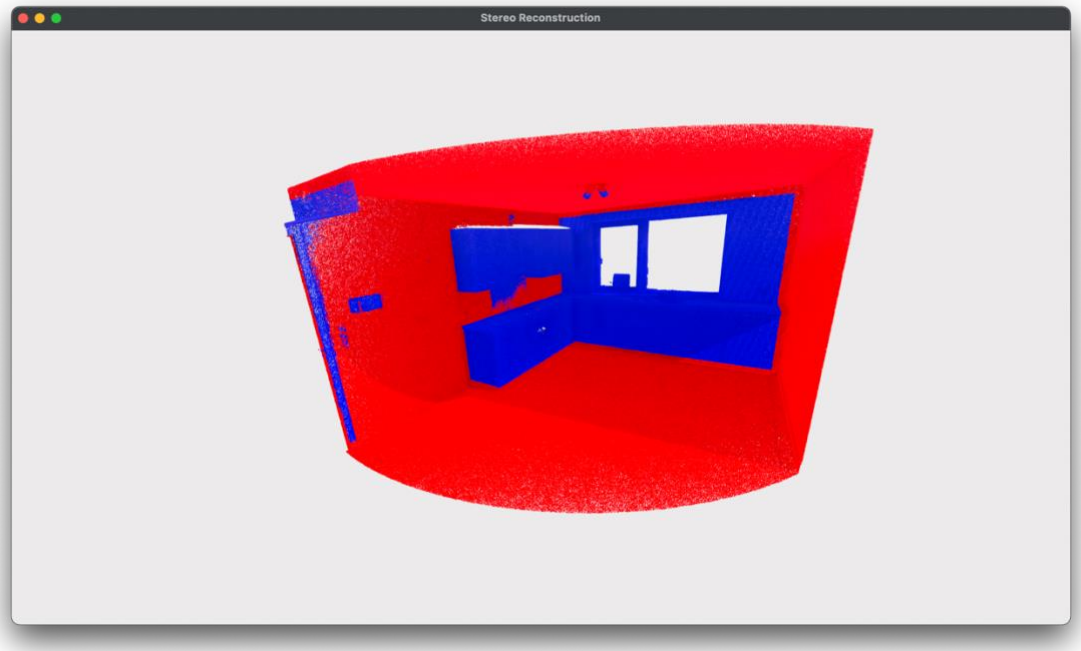

Figure 22 (β): Εντοπισμός τεσσάρων επιφανειών στο τρίτο artificial point cloud.

Στα artificial point clouds παρατηρήθηκε ότι για να εντοπιστούν τα σημεία που θεωρούμε τοίχους και πατώματα είναι καλή πρακτική να χρησιμοποιούμε σαν distance threshold περίπου την διπλάσια μέση απόσταση των 7 γειτονικών σημείων του κάθε σημείου του point cloud. Έτσι, χρησιμοποιήθηκε ένα kd tree έτσι ώστε να βρούμε τις αποστάσεις όλων των 7 γειτονικών σημείων για κάθε σημείο που είχαμε στο point cloud. Με αυτόν το τρόπο ξέρουμε επίσης και ότι άμα θέσουμε distance threshold μικρότερο αυτής της τιμής είναι πιθανό να μην δούμε κάποιο ουσιαστικό αποτέλεσμα αλλά πολύ λίγα σημεία τα οποία τέμνει το επίπεδο που ελέγχουμε. Επίσης, έτσι αποφεύγουμε το να θεωρούμε σημεία τα οποία δεν ανήκουν στις επίπεδες επιφάνειες τα οποία θα θεωρούσαμε μέρος τους άμα χρησιμοποιούσαμε μεγαλύτερο threshold και θα χάναμε κομμάτια από την υπόλοιπη σκηνή. Να σημειωθεί ότι αυτό το distance threshold δεν ισχύει και για τα point clouds που προέκυψαν από τα stereo pairs και εκεί το threshold υπολογίστε με trial and error. Τέλος όπως παρατηρούμε στο *figure 22 (α)* σε point clouds που έχουν και άλλες μεγάλες επίπεδες επιφάνειες, όπως είναι ο πάγκος τις κουζίνας ενδέχεται ο αλγόριθμος να μας επιστρέψει και

κάποιες από αυτές. Ωστόσο το να εντοπίσουμε και αυτές τις επιφάνειες έχει και κάποιες θετικές επιπτώσεις οι οποίες αναφέρονται παρακάτω.

Object Detection

Αφού έχουμε αφαιρέσει τις επίπεδες επιφάνειες από τις σκηνές που παρουσιάστηκαν θέλουμε να εντοπίσουμε τα αντικείμενα στην σκηνή (π.χ. αμάξια, κτήρια κλπ.). Για να επιτευχθεί αυτό χρησιμοποιήθηκε ο ίδιος αλγόριθμος ομαδοποίησης (DBSCAN) που χρησιμοποιήθηκε και για το denoising των point clouds. Η λογική πίσω από αυτή την επιλογή έχει να κάνει με το ότι αφού «αφαιρέθηκαν» οι επίπεδες επιφάνειες τα αντικείμενα όπως τα αμάξια θα έχουν μείνει στον «αέρα» δηλαδή θα είναι απομονωμένα και ο αλγόριθμος θα τα εντοπίσει.

Αποτελέσματα object detection σε point clouds από stereo pairs. Οι παράμετροι (epsilon, minimum samples) που χρησιμοποιήθηκαν για το κάθε point cloud φαίνεται στο Table 2 στην ενότητα Tables

- Πρώτο stereo point cloud:

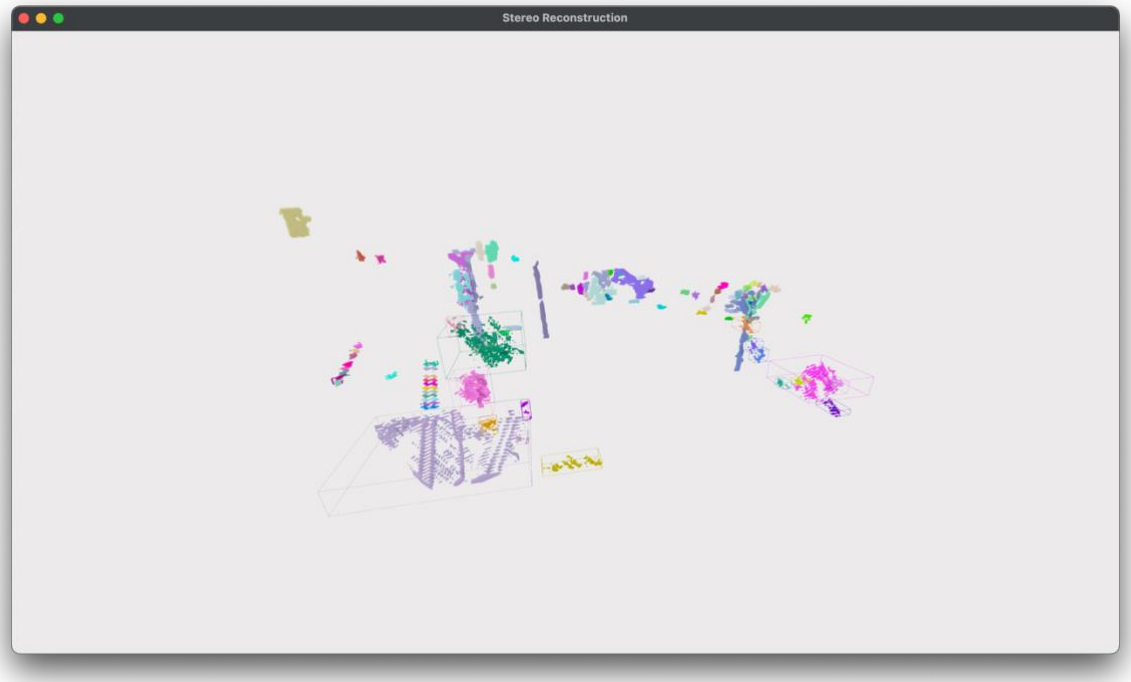

Figure 23: Αντικείμενα πρώτου stereo point cloud.

- Δεύτερο stereo point cloud:

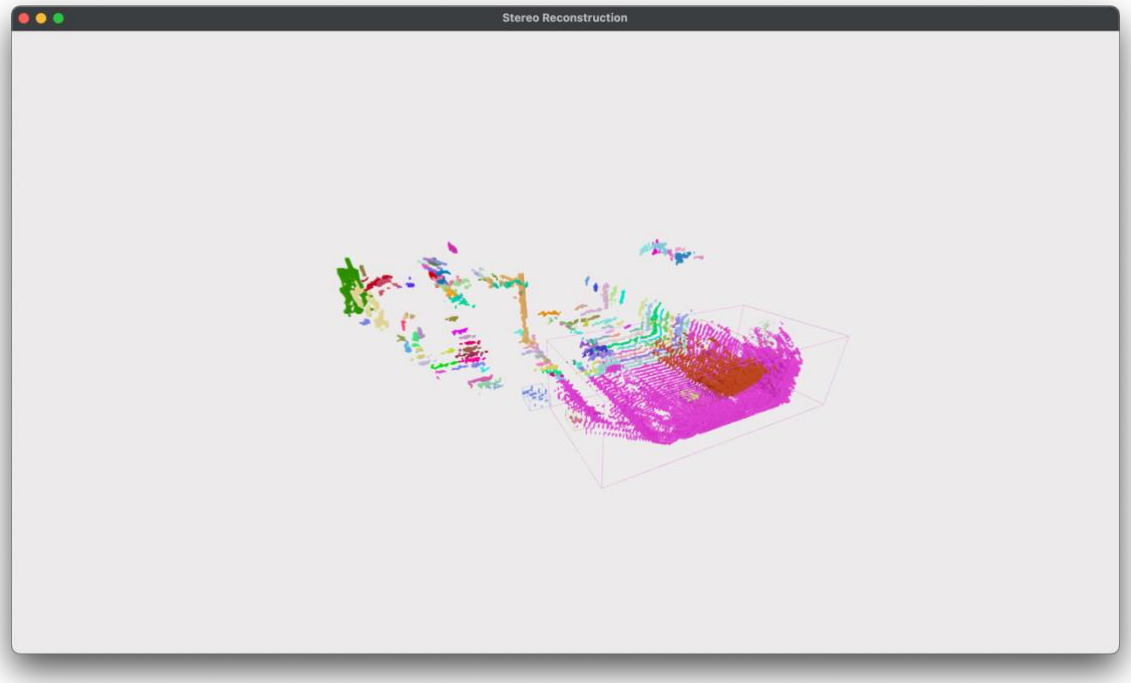

Figure 24: Αντικείμενα δεύτερου stereo point cloud.

- Τρίτο stereo point cloud:

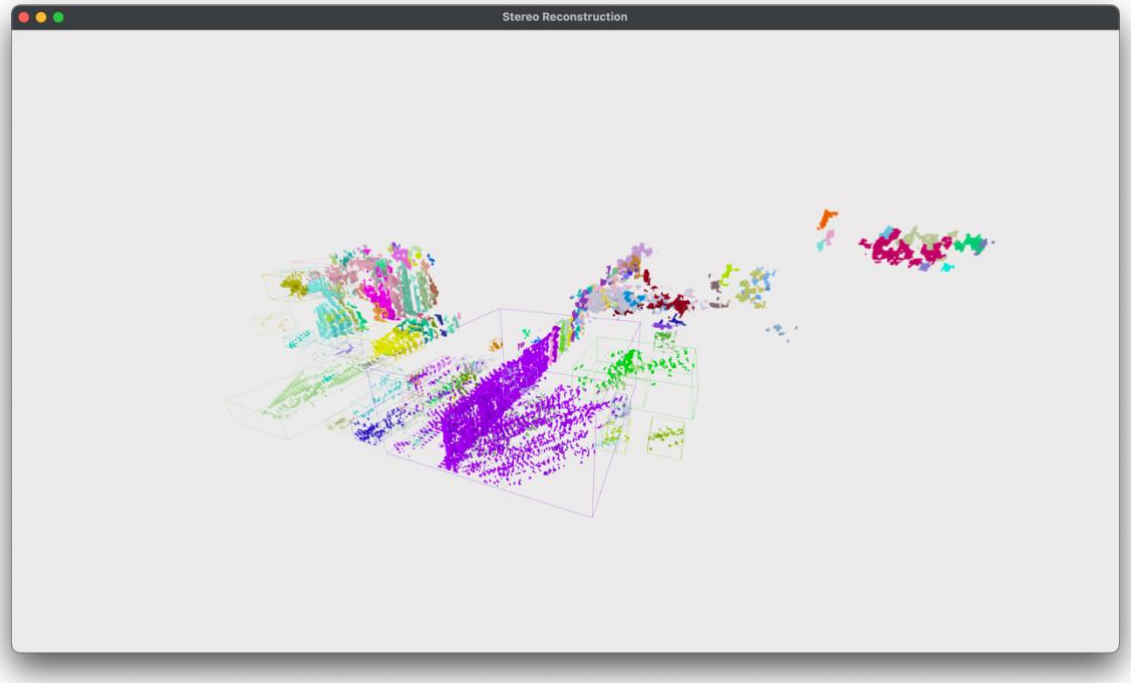

Figure 25: Αντικείμενα τρίτου stereo point cloud.

- Τέταρτο stereo point cloud:

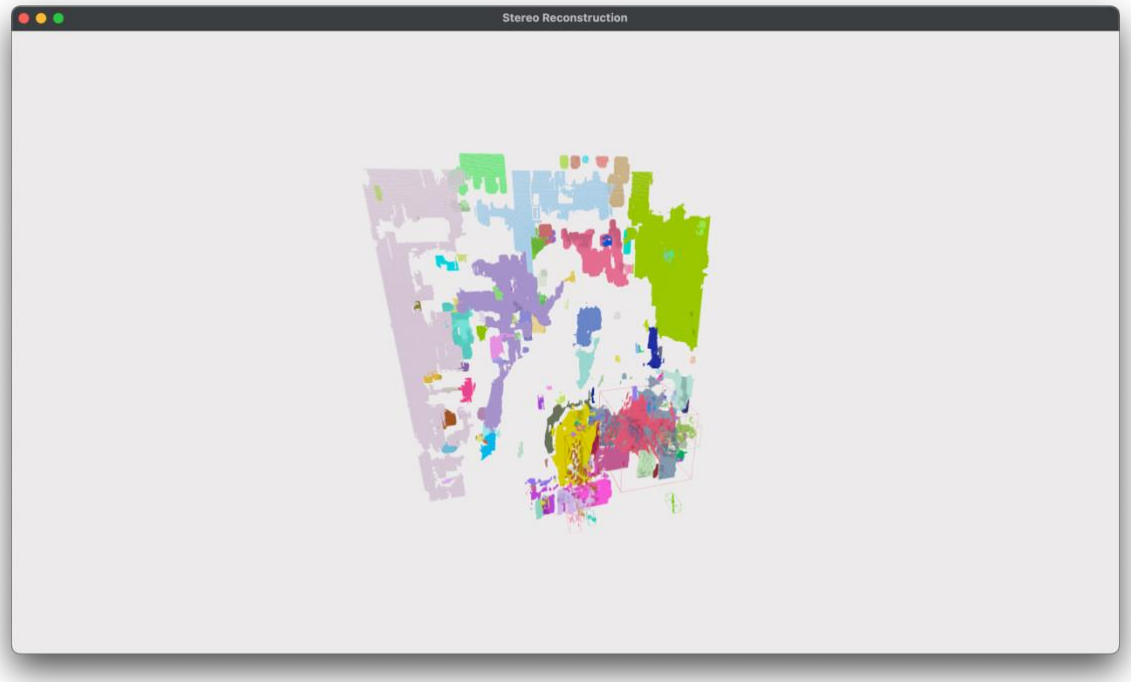

Figure 26: Αντικείμενα τέταρτου stereo point cloud.

Όπως φαίνεται από τα πρώτα τρία point clouds με την συγκεκριμένη μέθοδο προκύπτουν δύο βασικά προβλήματα. Αρχικά επειδή δεν έχουμε συνέχεια καθώς τα σημεία είναι σε επίπεδα που αντιστοιχούν στο disparity που προκύπτει για αυτά ο αλγόριθμος εντοπίζει σαν αντικείμενα κομμάτια από το κάθε επίπεδο. Αυτό μπορεί να διορθωθεί μεγαλώνοντας την παράμετρο epsilon ωστόσο έτσι προκαλείται ένα πρόβλημα στα point clouds που δεν έχει εντοπιστεί πλήρως ο δρόμος και έχουν μείνει κάποια κομμάτια του. Συγκεκριμένα, επειδή υπάρχει ο δρόμος κατά την «εξάπλωση» του αλγορίθμου μπορεί να ομαδοποιηθεί μαζί μέχρι και όλη η σκηνή καθώς βρίσκει συνεχώς σημεία έτσι ώστε να εξαπλωθεί. Ωστόσο, για το τέταρτο point cloud αλλά και για το πρώτο παίρνουμε καλά αποτελέσματα για κάποια αντικείμενα (κάμερα, κεφάλι, λάμπα, πινακίδα, φανάρι) τα οποία είναι σε ένα επίπεδο ή έστω τα σημεία τους είναι πολύ κοντά.

Αποτελέσματα object detection σε artificial point clouds. Με την ίδια λογική του DBSCAN προκύψαν και τα παρακάτω αποτελέσματα για τα artificial point clouds.

- Πρώτο artificial point cloud:

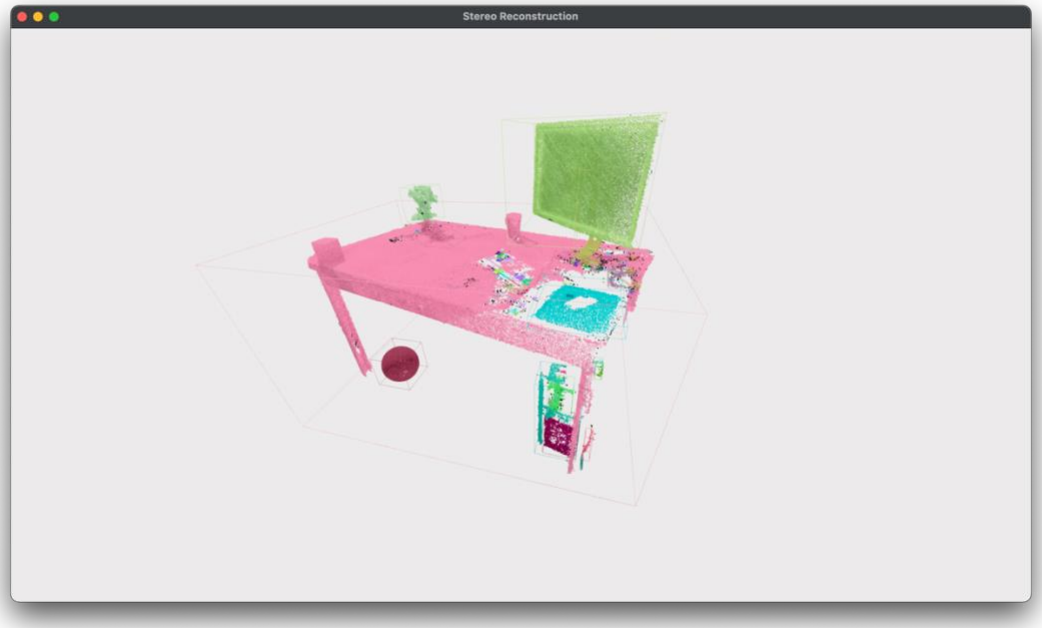

Figure 27: Αντικείμενα πρώτου artificial point cloud.

- Δεύτερο artificial point cloud:

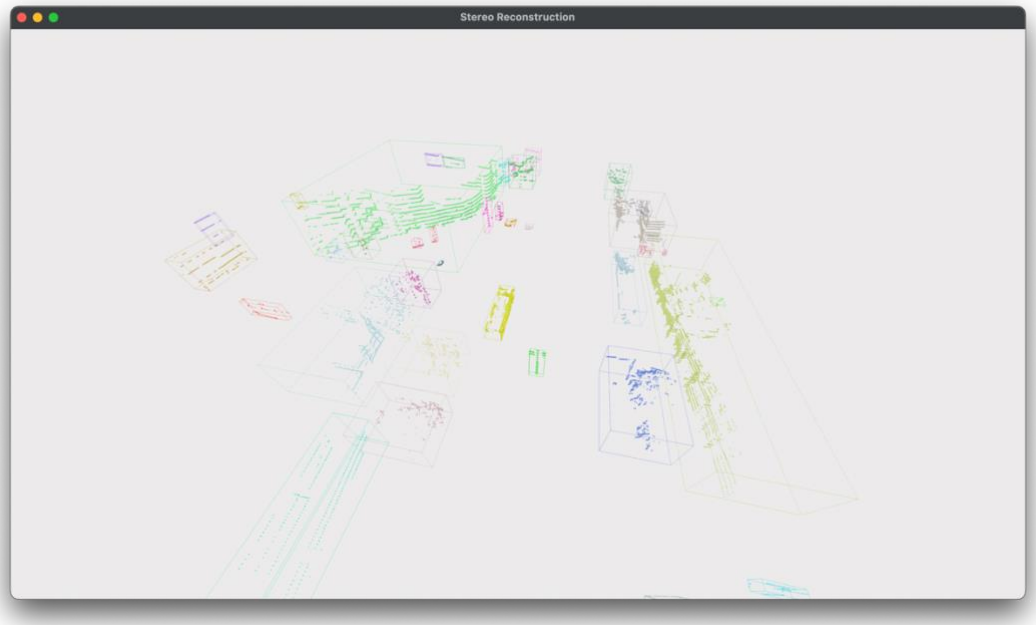

Figure 28: Αντικείμενα δεύτερου artificial point cloud.

- Τρίτο artificial point cloud:

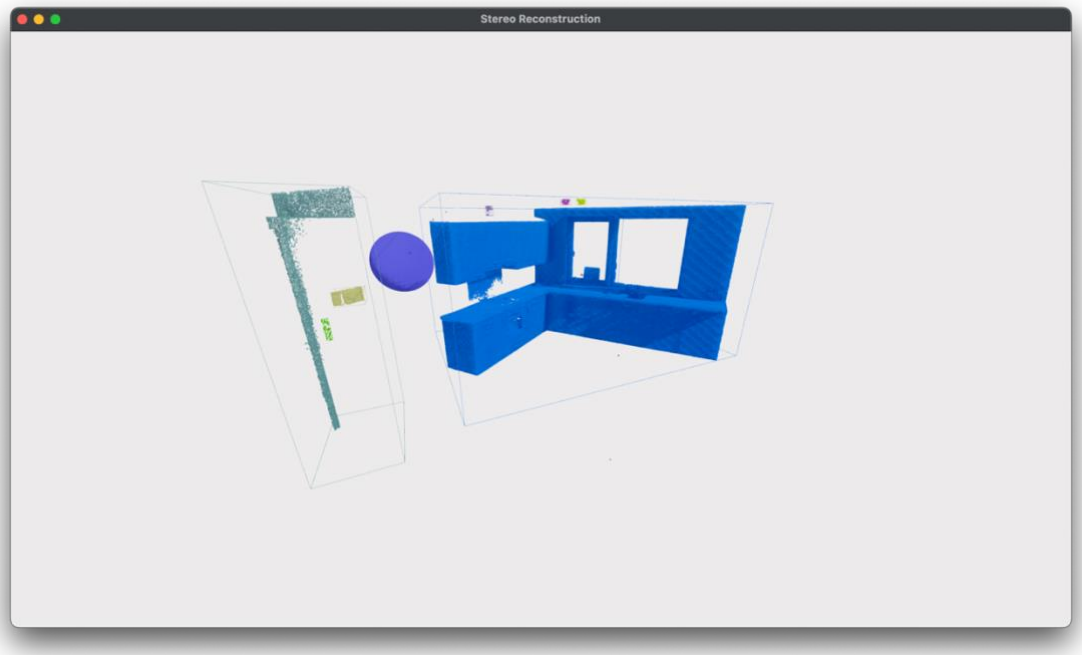

Figure 29 (α): Αντικείμενα τρίτου artificial point cloud.

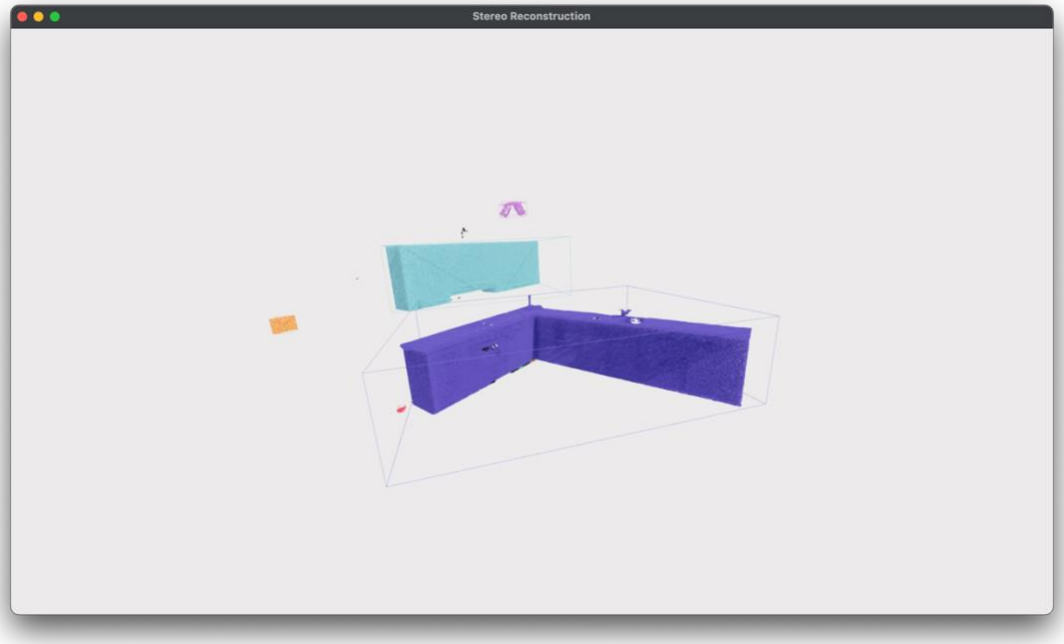

Figure 30 (β): Αποτελέσματα τρίτου artificial point cloud (χωρίς κανέναν τοίχο).

Για τα artificial point clouds βλέπουμε ότι η μέθοδος δουλεύει πολύ καλύτερο γιατί όντως τα πιο πολλά αντικείμενα έχουν κάποια απόσταση από την υπόλοιπη σκηνή αφού αφαιρεθούν οι επίπεδες επιφάνειες. Και για σε αυτή την περίπτωση χρησιμοποιήθηκε η μέση απόσταση των

7 γειτόνων του κάθε σημείου για να βρούμε μια μέση απόσταση έτσι ώστε να μην χρησιμοποιηθεί πολύ μεγάλο η πολύ μικρό epsilon. Το πολύ μεγάλο θα οδηγούσε σε ομαδοποίηση όλων των σημείων της σκηνής και το πολύ μικρό δεν θα επέστρεφε καμία ομάδα σημείων. Ωστόσο, και εδώ πολλά αντικείμενα δεν εντοπίστηκαν γιατί τα σημεία είναι έχουν μια σταθερή απόσταση μεταξύ τους επειδή είναι τεχνητά point cloud οπότε ο αλγόριθμος τα ομαδοποιεί όλα μαζί. Στο point cloud από τον αισθητήρα lidar βέβαια, έχουμε τα καλύτερα αποτελέσματα καθώς τα σημεία είναι κοντά μόνο με αυτά που όντως ανήκουν στο ίδιο αντικείμενο. Από τις διάφορες δοκιμές που γίνανε φαίνεται ότι για point cloud που είναι αραιά πρέπει να έχουμε μεγαλύτερο epsilon και μεγαλύτερο minimum samples για να εντοπιστούν τα αντικείμενα ενώ για τα πυκνά point cloud και οι δύο παράμετροι πρέπει να είναι μικρότεροι. Τέλος, μια πρακτική που φάνηκε να έχει ένα καλό αποτέλεσμα είναι αν αφαιρούμε και άλλες επίπεδες επιφάνειες (όπως ο πάγκος από την κουζίνα) έτσι ώστε να μένουν αντικείμενα στον «αέρα» και να τα εντοπίζει πιο εύκολα ο αλγόριθμος όπως φαίνεται παρακάτω.

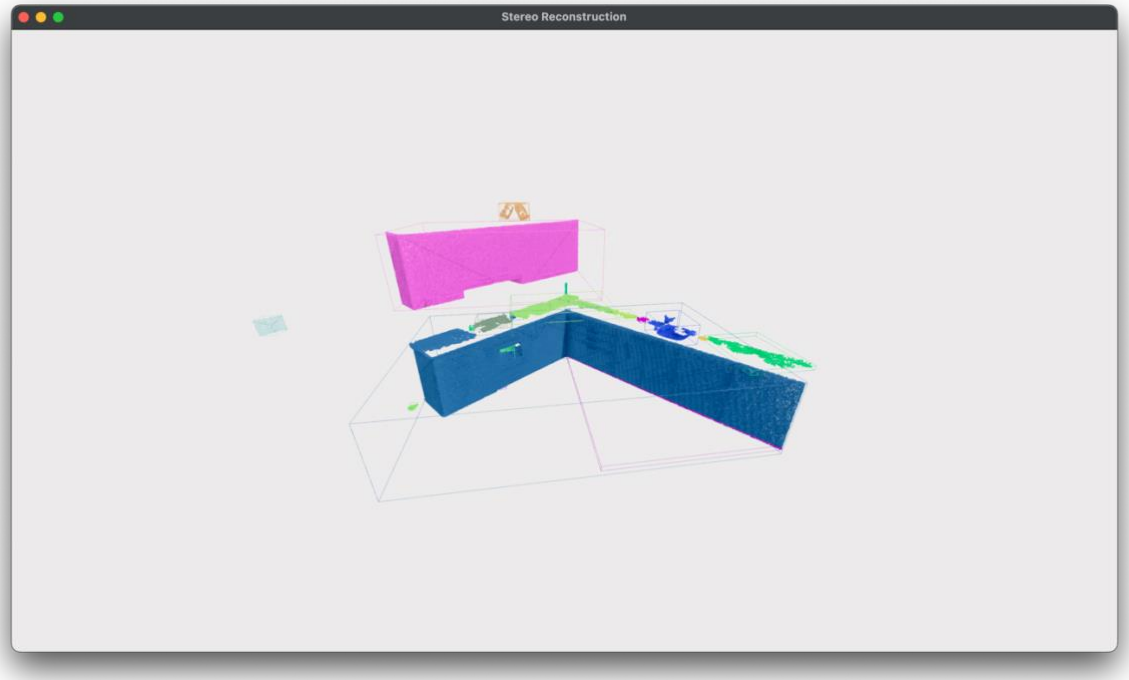

Figure 31: Αντικείμενα τρίτου artificial point cloud έπειτα από την αφαίρεση και του πάγκου.

Scene Virtualization

Σε αυτό το κομμάτι της εργασίας θέλουμε να κάνουμε «διαδραστικά» τα αντικείμενα της σκηνής. Συγκεκριμένα θέλουμε να έχουμε μία σφαίρα στην σκηνή η οποία θα αντιδράει με το Axis Aligned Bounding Box (AABB) του κάθε αντικειμένου.

Προσθήκη σφαίρας στην σκηνή

Επειδή ο τρόπος που λειτουργεί το widget της open3d είναι να γίνεται ένα event και να γίνεται ένα render την σφαίρα πρακτικά την κουνάμε προς μία κατεύθυνση. Αρχικά τοποθετούμε την σφαίρα στην σκηνή. Έπειτα προσθέτουμε στο διάνυσμα που αντιστοιχεί στη θέση του κέντρου της, το διάνυσμα της ταχύτητας. Σβήνουμε την προηγούμενη σφαίρα και κάνουμε μία καινούργια στην επόμενη θέση και έτσι έχουμε την κίνηση της σφαίρας σε μία κατεύθυνση. Να σημειωθεί ότι η ταχύτητα της σφαίρας είναι σταθερή προς την κατεύθυνση που ορίζουμε.

Collision detection

Για να κάνουμε την σφαίρα να αντιδρά όταν συγκρούεται με κάποιο bounding box πρέπει πρώτα να εντοπίσουμε την σύγκρουση. Άρα πρέπει να εφαρμόσουμε collision detection μεταξύ σφαίρας και AABB. Η διαδικασία αυτή είναι πολύ απλή καθώς έχουμε υπολογίσει όλα τα bounding boxes τον αντικειμένων κατά την διαδικασία εντοπισμού τους. Κάνουμε loop το dictionary που έχουμε δημιουργήσει με τα AABB και υπολογίζουμε για το καθένα από αυτά την απόσταση του κοντινότερου σημείου τους από το κέντρο της σφαίρας. Έτσι άμα κάποια από αυτές τις αποστάσεις είναι μικρότερη από την ακτίνα της σφαίρας τότε έχουμε collision.

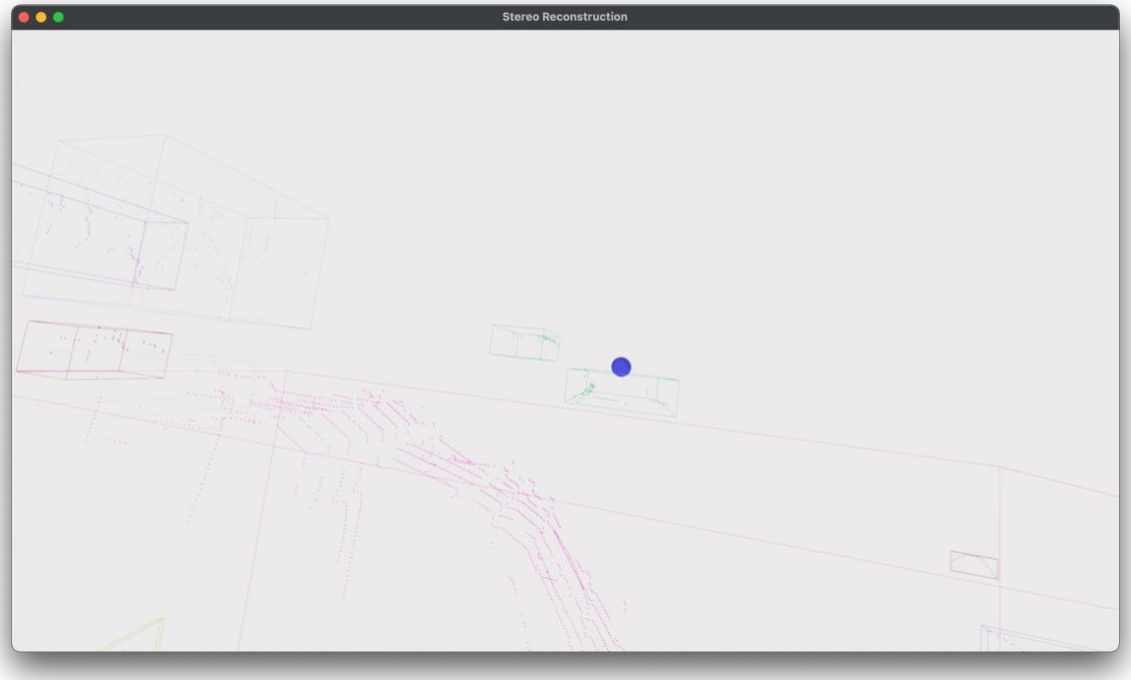

Figure 32: Collision μεταξύ σφαίρας και AABB

Στο παράδειγμα απεικονίζεται η σφαίρα που έχει σταματήσει πάνω στο AABB. Το κομμάτι τις σφαίρας που φαίνεται να μπαίνει εντός του AABB έχει να κάνει με το γεγονός ότι η σφαίρα μετακινείται βήμα-βήμα. Οπότε επειδή η ταχύτητα της σφαίρας είναι λίγο μεγάλη για να μην παίρνει πολύ ώρα μέχρι να φτάσει σε κάποιο bounding box η άκρη της κάνει clip στην πλευρά.

Collision handling

Αφού εντοπίσουμε το collision πρέπει να κάνουμε εφαρμόσουμε τις κατάλληλες αλλαγές στην κατεύθυνση μεταφοράς της σφαίρας. Εφόσον έχουμε σύγκρουση σφαίρας με επίπεδο θα χρησιμοποιήσουμε το concept του reflection. Το μόνο που χρειαζόμαστε για να υπολογίσουμε την καινούργια ταχύτητα της σφαίρας είναι το διάνυσμα κατεύθυνσης της σφαίρας, το σημείο σύγκρουσης και το normal της επιφάνειας. Για την κατεύθυνση της σφαίρας και το σημείο σύγκρουσης δεν χρειάζεται να κάνουμε κάτι επιπλέον καθώς το πρώτο είναι ορισμένο από τον χρήστη και το δεύτερο υπολογίζεται κατά την διάρκεια του collision detection. Για το normal του plane η διαδικασία είναι πολύ απλή καθώς έχουμε

AABB. Το μόνο που πρέπει να κάνουμε είναι να δούμε ποια από τις συντεταγμένες του min bound ή του max bound του AABB ισούται με την αντίστοιχη συντεταγμένη του σημείου σύγκρουσης. Αν ισούται κάποια από τις συντεταγμένες του min bound με κάποια από αυτές του σημείου σύγκρουσης τότε το normal του plane είναι το μοναδιαίο τις συντεταγμένης που ισούται. Ενώ αν ισούται κάποια του max bound τότε το normal είναι το αντίθετο του μοναδιαίου. Έπειτα μένει να υπολογίσουμε την καινούργια κατεύθυνση (reflection) όπως φαίνεται παρακάτω

$$reflection = speed - 2 * (speed \cdot normal) * normal$$

όπου speed είναι το normalized διάνυσμα ταχύτητας και normal το normal τις πλευράς του bounding box που γίνεται το collision.

Object-based Triangulation

Το τελευταίο ζητούμενο της εργασίας είναι να εφαρμόσουμε object-based triangulation σε ένα point cloud. Για την επίτευξη αυτού του σκοπού επιλέχθηκε ο ball pivot algorithm. Ο αλγόριθμος ονομάζεται έτσι για τον τρόπο που αντιμετωπίζει το πρόβλημα. Συγκεκριμένα είναι σαν να έχουμε μία μπάλα την οποία κάνουμε pivot γύρω από τα edges των τριγώνων έτσι ώστε να δημιουργήσουμε καινούργια τρίγωνα και τελικά το mesh. Συγκεκριμένα ο αλγόριθμος έχει δύο βασικές λειτουργίες. Αρχικά, να εντοπίζει τρίγωνα τα οποία είναι κατάλληλα για να ξεκινήσει από εκεί την τριγωνοποίηση, το οποία είναι τα seed triangles. Δεύτερον, να εφαρμόζει την pivot κίνηση γύρω από τα edge των τριγώνων, η οποία είναι πρακτικά ένα rotate της εικονικής σφαίρας γύρω από το edge και έπειτα να ελέγχει άμα μπορεί να επεκτείνει το τρίγωνο από το edge γύρω από το οποίο κάνει pivot. Οι παράμετροι που επηρεάζουν την συμπεριφορά του αλγόριθμου είναι η ακτίνα της σφαίρας και ο ανώτατος αριθμός προσπαθειών τριγωνοποίησης

Εύρεση seed triangle

Αρχικά επιλέγεται ένα τυχαίο σημείο (p1) το οποίο δεν χρησιμοποιείται. Έπειτα για να βρεθούν τα επόμενα δύο σημεία ακολουθείται η εξής λογική. Βρίσκουμε τους γείτονες του p1 σε ακτίνα ίση με αυτή της σφαίρας που θα κάνουμε pivot. Έπειτα, παίρνουμε τους 5 πιο κοντινούς, σαν υποψήφιους για το δεύτερο σημείο του τριγώνου (p2). Βρίσκουμε για αυτούς τους 5, ξεκινώντας από τον πιο κοντινό στο p1, τους δικούς τους γείτονες σε ακτίνα ίση με της σφαίρας. Ελέγχουμε αν κάποιος από αυτούς έχει κοινούς γείτονες με το p1. Αυτοί οι κοινοί γείτονες του p1 και του p2 είναι η υποψήφιοι για το τρίτο και τελευταίο σημείο έτσι ώστε να σχηματιστεί seed triangle. Οπότε, υπολογίζουμε την απόσταση όλων αυτών των κοινών γειτόνων από το p1 και επιλέγουμε τον πιο κοντινό για το τρίτο σημείο του τριγώνου (p3). Αν έχουμε εξαντλήσει όλες τις επιλογές για το κάθε p2 και p3 και δεν έχουμε βρει κάποιο τρίγωνο τότε απλά πηγαίνουμε σε ένα καινούργιο τυχαίο p1 που δεν χρησιμοποιείται και επαναλαμβάνουμε την διαδικασία.

Ball pivot γύρω από edge

Αυτή η διαδικασία εφαρμόζεται αφού έχουμε βρει κάποιο seed triangle. Η λογική είναι η εξής. Καθώς κάνουμε pivot την σφαίρα γύρω από κάποιο edge του τριγώνου θα σχηματίσουμε καινούργιο τρίγωνο αν αυτή «χτυπήσει» με κάποιο άλλο σημείο. Για να βρεθεί αν η σφαίρα «χτυπάει» κάποιο σημείο πρέπει να υπολογιστεί η ακτίνα του εσωτερικού κύκλου του τριγώνου που σχηματίζει το υποψήφιο σημείο με τα δύο άκρα του edge. Την ακτίνα αυτή την βρίσκουμε από την παρακάτω εξίσωση

$$iner\ circle\ radius = \sqrt{\frac{(s-|e_1|)*(s-|e_2|)*(s-|e_3|)}{s}}$$

$$s = \frac{|e_1|+|e_2|+|e_3|}{2}$$

όπου s είναι η ημιπερίμετρος και $|e_i|$ για i=1,2,3 είναι το μήκος τις εκάστοτε πλευράς του τριγώνου.

Τα βήματα που ακολουθήσαμε κατά την διαδικασία επέκτασης του see triangle περιγράφονται αναλυτικά παρακάτω.

1. Επιλογή ενός από τα edges που δεν έχουμε ελέγξει για να κάνουμε pivot γύρω του
2. Εύρεση κοινών γειτόνων των δύο άκρων του edge σε ακτίνα ίση με αυτή της σφαίρας
3. Για κάθε έναν από αυτούς τους γείτονες δημιουργία ενός υποθετικού τριγώνου
4. Έλεγξε αν κάποιο από αυτά είναι έγκυρο το οποίο συμβαίνει όταν η ακτίνα του εσωτερικού του κύκλου είναι μικρότερη από αυτή της σφαίρας
5. Κράτα το τρίγωνο του πιο κοντινού γείτονα το είναι έγκυρο
6. Αν βρεθεί καινούργιο τρίγωνο επανέλαβε την διαδικασία θεωρώντας εκείνο σαν seed triangle αλλιώς, επανέλαβε την διαδικασία εύρεσης seed triangle

Αποτελέσματα ball pivot algorithm

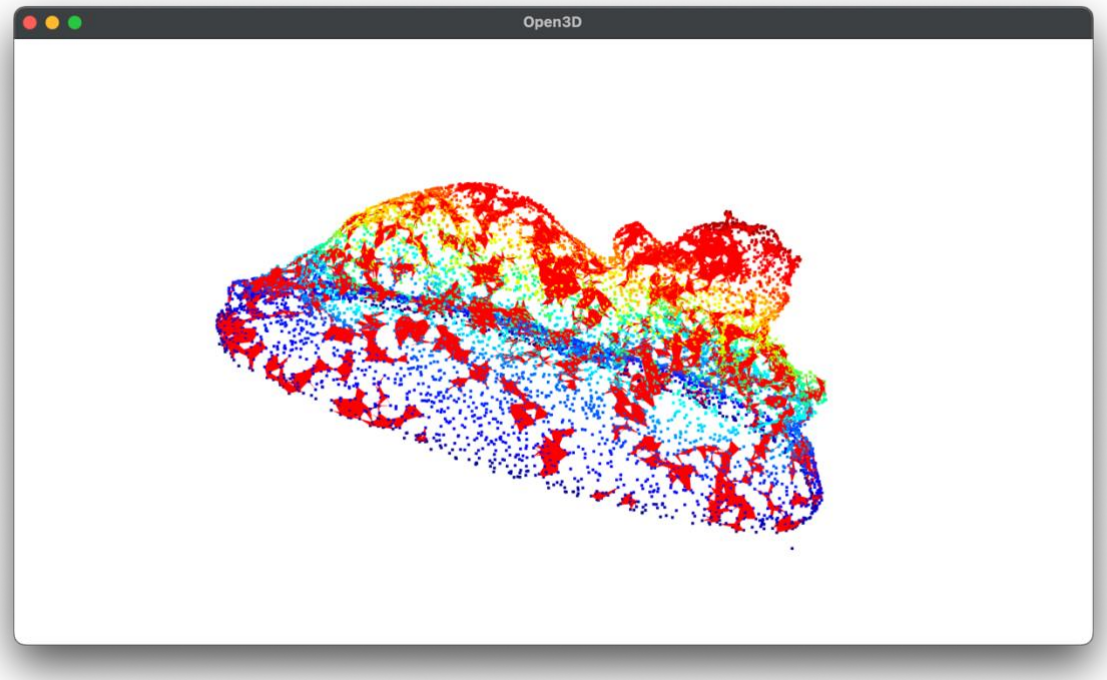

Figure 33: Αποτελέσματα ball pivot algorithm.

Η υλοποίηση του αλγόριθμου δεν είναι άρτια και αυτό φαίνεται και στα αποτελέσματα. Ωστόσο στο παράδειγμα βλέπουμε ότι δημιουργούνται αρκετά τρίγωνα για το point cloud

10000 σημείων που προσπαθούμε να τριγωνοποιήσουμε. Επίσης, να σημειωθεί πως αν ο αλγόριθμος τρέξει για περισσότερη ώρα ενώνει σχεδόν όλα τα σημεία ωστόσο όχι με τον σωστό τρόπο καθώς υπάρχουν overlaps μεταξύ τριγώνων. Αυτό λογικά οφείλεται στους ελλιπείς ελέγχους για την επιλογή σωστών σημείων καθώς δεν βρήκα κάποια ξεκάθαρη πηγή για να καταλάβω πως ακριβώς να το υλοποιήσω.

Οδηγίες Χρήσης Εφαρμογής

Dependencies Εφαρμογής

Για να μπορέσει ο χρήστης να τρέξει την εφαρμογή πρέπει πρώτα να έχει το κατάλληλο περιβάλλον το οποίο χρειάζεται. Συστήνεται χρήση ενός anaconda environment το οποίο και αναλύεται παρακάτω. Ωστόσο, ο χρήστης μπορεί να χρησιμοποιήσει οποιοδήποτε python environment προτιμάει αρκεί να κατεβάσει τις απαραίτητες βιβλιοθήκες.

Anaconda

Αρχικά ο χρήστης πρέπει να κατεβάσει το anaconda. Αφού ολοκληρωθεί αυτή η εγκατάσταση μένει η δημιουργία του environment. Αυτό μπορεί μέσω του anaconda navigator ή μέσω του command line. Συγκεκριμένα, μέσω του command line ο χρήστης πρέπει να γράψει την εντολή ‘conda create --name myenv’ όπου αντί για myenv ο χρήστης μπορεί να χρησιμοποιήσει όποιο όνομα θέλει για το environment. Αφού έχει ολοκληρωθεί το environment ο χρήστης το μόνο που πρέπει να κάνει είναι να το κάνει activate αν έχει σκοπό να τρέξει την εφαρμογή από το terminal και για να κατεβάσει τις βιβλιοθήκες στο environment και όχι στο base.Αυτό γίνεται με την χρήση της εντολής ‘conda activate myenv’ όπου myenv είναι το όνομα που έχει επιλέξει ο χρήστης.

Packages

Τα libraries που πρέπει να κατεβάσει ο χρήστης αναφέρονται παρακάτω. Οδηγίες για την εγκατάσταση τους για το εκάστοτε λογισμικό μπορούν να βρεθούν στο anaconda.org αναζητώντας το όνομα το package.

Packages:

1. Numpy
2. Open3d
3. OpenCV
4. Matplotlib

5. Tqdm
6. Scipy
7. Sklearn

Συστήνεται ο χρήστης να κατεβάσει τα packages με την σειρά που παρουσιάζεται έτσι ώστε να μην υπάρχουν conflicts. Επίσης αφού κατεβούν τα πρώτα πέντε καλό θα ήταν να ελέγξει ο χρήστης μήπως τα δύο τελευταία κατέβηκαν με κάποιο από τα πρώτα οπότε δεν υπάρχει λόγος να τα κατεβάσει. Αφού έχουν κατεβεί τα packages ο χρήστης μπορεί να τρέξει την εφαρμογή μέσω του terminal αφού έχει κάνει activate το environment όπως αναφέρθηκε και βρίσκεται στο path που έχει κατεβάσει την εφαρμογή χρησιμοποιώντας το command ‘python main.py’. Αν είναι επιθυμητή η χρήση κάποιου IDE (π.χ. Visual Studio Code) πρέπει να ρυθμιστεί έτσι ώστε να χρησιμοποιείται σαν interpreter το anaconda environment που δημιουργήθηκε.

Επιλογή Δεδομένων & Εισαγωγή τους στην Σκηνή

Αφού ο χρήστης έχει τρέξει το αρχείο main.py πρέπει να επιλέξει πια αρχεία θέλει να χρησιμοποιήσει για να δοκιμάσει την εφαρμογή. Συγκεκριμένα ποιο ζεύγος στερεοσκοπικών εικόνων και ποιο artificial point cloud.

Επιλογή δεδομένων

Η επιλογή των δεδομένων γίνεται μέσω του terminal. Όταν ο χρήστης τρέξει το αρχείο main.py θα δει δύο ερωτήσεις σχετικά με το τι θα επιλέξει. Η πρώτη θα τον ρωτάει πιο από τα artificial point cloud θέλει να χρησιμοποιήσει. Οι επιλογές του χρήστη είναι από το 1 μέχρι το 5. Ωστόσο όπως ενημερώνεται και από το terminal το point cloud 1 δεν λειτουργεί σωστά καθώς έχει πάρα πολλά σημεία και επιβαρύνει πολύ τον υπολογιστή. Συστήνεται η επιλογή του Adas lidar point cloud (**3**) γιατί φαίνεται καλύτερα το virtualization της σκηνής με την μπάλα. Έπειτα θα του ζητηθεί να επιλέξει ένα stereo pair. Οι πρώτες τρείς επιλογές είναι σκηνές από το dataset το KITTI και σε αυτές έχει νόημα να τις χρησιμοποιήσει για να

δει τα αποτελέσματα του RANSAC. Η τέταρτη επιλογή είναι καλύτερη για την προβολή εντοπισμού αντικειμένων στην σκηνή με τον αλγόριθμο DBSCAN. Αν αφού δοκιμάσει τις λειτουργίες της εφαρμογής με τα επιλεγμένα δεδομένα ο χρήστης θέλει να δοκιμάσει κάποια άλλα πρέπει απλά να κλείσει το παράθυρο να την ξανατρέξει και να επιλέξει αυτά που θέλει.

Εισαγωγή δεδομένων στην σκηνή και reset σκηνής – A, S, R

Ο χρήστης αφού έχει επιλέξει τα δεδομένα θα δει μία άδεια σκηνή. Αυτό γίνεται για να μπορεί χωρίς να επανεκκινήσει την εφαρμογή να δει τα αποτελέσματα για τους δύο τύπους point cloud. Οπότε, προτού κάνει οποιαδήποτε διεργασία από αυτές που έχουν αναφερθεί πρέπει να επιλέξει με ποιο από τα δύο θέλει να ασχοληθεί. Τέλος, αφού έχει τελειώσει με όποια διεργασία εφάρμοσε στο point cloud που επέλεξε μπορεί να επαναφέρει την σκηνή στην αρχική της κατάσταση για να ξαναχρησιμοποιήσει το ίδιο ή το άλλο πατώντας το πλήκτρο **R**.

Artificial point cloud – A. Για να εισάγει το artificial point cloud στην σκηνή έτσι ώστε να το επεξεργαστεί ο χρήστης πρέπει να πατήσει το πλήκτρο **A**.

Stereo point cloud – S. Ο χρήστης να σημειωθεί ότι με την επιλογή που κάνει στην αρχή επιλέγει το ζεύγος των στερεοσκοπικών εικόνων. Οπότε, αν θέλει να χρησιμοποιήσει το point cloud που προκύπτει από αυτές θα πρέπει να περιμένει και ένα πολύ μικρό χρονικό διάστημα έτσι ώστε να υπολογιστεί. Για να το κάνει αυτό λοιπόν, πρέπει να πατήσει το πλήκτρο **S** και μόλις τελειώσει η διαδικασία το point cloud θα εμφανιστεί στην σκηνή.

Επεξεργασία Point Cloud

Παρακάτω αναφέροντα όλες οι διεργασίες που μπορεί να εκτελέσει ο χρήστης αφού εισάγει ένα από τα δύο point cloud στην σκηνή. Προφανώς για να εφαρμοστούν κάποιες διεργασίες πρέπει πρώτα να έχουν προηγηθεί κάποιες άλλες και κάποιες πρέπει να εφαρμοστούν μόνο στα stereo point clouds. Αυτές οι περιπτώσεις επισημαίνονται παρακάτω.

Denoising with clustering – D

Η διαδικασία του denoising προφανώς πρέπει να χρησιμοποιηθεί μόνο πάνω στα stereo point clouds. Ο χρήστης αφού έχει δει το stereo point cloud και μπορεί να αλληλοεπιδράσει πάλι με την σκηνή μπορεί να πατήσει το πλήκτρο **D** και έπειτα από ένα μικρό χρονικό διάστημα που παίρνει η διαδικασία του denoising θα δει το point cloud να έχει μειωθεί αισθητά λόγω της αφαίρεσης του θορύβου. Έπειτα μπορεί να εφαρμόσει όποια διεργασία θέλει.

Plane detection – 1:9

Το plane detection μπορεί να εφαρμοστεί και στις δύο επιλογές του χρήστη. Ο χρήστης μπορεί να επιλέξει τον αριθμό των plane που θέλει να βρεθούν και να «αφαιρεθούν» από την σκηνή επιλέγοντας ένα πλήκτρο από το **1** μέχρι το **9**. Συνιστάται η αφαίρεση ενός plane από τις σκηνές του KITTI το οποίο θα αντιστοιχεί στον δρόμο. Το ίδιο ισχύει και για το researcher’s desk και το Adas lidar point cloud όπου η επιφάνεια που θέλουμε να εντοπίσουμε είναι μία, στην πρώτη περίπτωση το πάτωμα και στην δεύτερη ο δρόμος. Για την σκηνή της κουζίνας με τους τοίχους ενθαρρύνεται και η επιλογή αφαίρεσης τεσσάρων επιπέδων με σκοπό να εντοπιστούν και οι τέσσερεις τοίχοι. Καλό είναι να μην γίνεται επιλογή πολλών επιπέδων γιατί καθυστερεί πολύ και δεν έχει κάποιο ουσιαστικό αποτέλεσμα.

Object detection – C

Ο χρήστης μπορεί ανά πάσα στιγμή να εφαρμόσει την διεργασία εντοπισμού αντικειμένων πατώντας το πλήκτρο **C**. Ωστόσο, συστήνεται για τα stereo point clouds να εφαρμοστεί έπειτα από την αφαίρεση του επιπέδου (στα πρώτα τρία stereo pairs) και έπειτα από την εφαρμογή του denoising (και στα τέσσερα). Να σημειωθεί ότι αυτή η διεργασία επιστρέφει και τα AABB των αντικειμένων που εντοπίζονται.

Virtualization Σκηνής

Παρακάτω αναφέρονται οι λειτουργείες με τις οποίες μπορεί ο χρήστης να αλληλοεπιδράσει με την σκηνή με την χρήση της μπάλας. Καλό είναι για να δει καλύτερα αποτελέσματα ο χρήστης να επιλέξει το Adas lidar point cloud όπως αναφέρθηκε και παραπάνω.

Κίνηση σφαίρας – T, F

Ο χρήστης μπορεί να μετακινήσει την σφαίρα εντός της σκηνής χρησιμοποιώντας τα πλήκτρα **T** και **F**. Συγκεκριμένα, το πλήκτρο T «κουνάει» την σφαίρα προς την κατεύθυνση που έχει οριστεί ενώ το F προς την αντίθετη. Να σημειωθεί πως για να αλληλοεπιδράσει η σφαίρα με την σκηνή *πρέπει να έχει προηγηθεί το object detection* έτσι ώστε να υπάρχουν τα AABB γύρω από τα αντικείμενα με τα οποία και θα αλληλοεπιδράσει.

Object-based Triangulation

Το object-based triangulation μπορεί να εφαρμοστεί πατώντας το πλήκτρο **G**. Ωστόσο, δεν προτείνεται καθώς είναι πάρα πολύ αργό και μπορεί να κρασάρει την εφαρμογή καθώς το widget της open3d είναι αρκετά buggy. Γι’ αυτό και τα αποτελέσματα που παρουσιάστηκαν υπολογίστηκαν σε ξεχωριστό python file και μπορούν να εντοπιστούν στον φάκελο Triangulation Results. Επίσης στο παραδοτέο υπάρχει και το python file (triangulation.py) το οποίο μπορεί να εκτελεστεί με την εντολή ‘python triangulation.py’. Και πάλι όμως δεν συστήνεται ούτε αυτό καθώς παίρνει πολύ ώρα για να τελειώσει η εκτέλεση.

Επιπρόσθετες Λειτουργίες

Η εφαρμογή έχει και μερικές λειτουργίες οι οποίες δεν επηρεάζουν κάπως τα αποτελέσματα αλλά απλά δίνουν την δυνατότητα στον χρήστη να εμφανίζει και να κρύβει κάποια πράγματα από την σκηνή.

Εμφάνιση και απόκρυψη inliers και outliers – I, O, P

Ο χρήστης μπορεί να επιλέξει αν θα βλέπει μόνο τους inliers, δηλαδή τα σημεία που βρίσκονται εντός του distance threshold από το επίπεδο του RANSAC αλγόριθμου, πατώντας το πλήκτρο **I**, ή μόνο τους outliers πατώντας το πλήκτρο **O**, δηλαδή τα υπόλοιπα σημεία. Αν θέλει να προστεθούν και ξανά όλα τα σημεία στην σκηνή απλά πρέπει να πατήσει το πλήκτρο **P**. Προφανώς, αυτές οι λειτουργίες είναι διαθέσιμες μόνο μετά από τον εντοπισμό των επιπέδων με τον τρόπο που αναφέρθηκε παραπάνω.

Εμφάνιση και απόκρυψη των AABB – B, H

Αυτή ή λειτουργία είναι διαθέσιμη στον χρήστη έπειτα από τον εντοπισμό των αντικειμένων ο οποίος προσθέτει στην σκηνή και τα AABB τους. Οπότε, αφού έχουν εντοπιστεί τα αντικείμενα και τα AABB είναι εμφανισμένα ο χρήστης μπορεί να τα κρύψει πατώντας το πλήκτρο **H** και μετά να τα εμφανίσει ξανά πατώντας το πλήκτρο **B**. Είναι καλό να είναι εμφανισμένα κατά την κίνηση της σφαίρας στην σκηνή έτσι ώστε να είναι ορατή και πιο κατανοητή όποια σύγκρουση που μπορεί να προκύψει.

Εμφάνιση και απόκρυψη θορύβου – N, Z

Άλλη μία δυνατότητα που έχει ο χρήστης έπειτα από τον εντοπισμό των αντικειμένων είναι να κρύψει τον θόρυβο ο οποίος δεν ομαδοποιήθηκε σε κανένα αντικείμενο. Συγκεκριμένα, ο χρήστης μπορεί να πατήσει το πλήκτρο **N** για να κρύψει τον θόρυβο ενώ άμα θέλει να τον ξαναεμφανίσει το πλήκτρο **Z**.

Εμφάνιση denoised disparity map – M

Μια απλή λειτουργία που έχει στην διάθεσή του ο χρήστης είναι οπτικοποίηση του denoised disparity map που stereo pair που επέλεξε. Για να το κάνει αυτό απλά πρέπει να πατήσει το πλήκτρο **M** και εμφανίζει το disparity map σε ένα εξωτερικό figure της pyplot. Προφανώς, για να χρησιμοποιηθεί αυτή η λειτουργία πρέπει ο χρήστης να έχει προσθέσει το point cloud από το stereo pair στην σκηνή έτσι ώστε να έχει υπολογιστεί το αρχικό disparity map.

Υπολογισμός μέσης απόστασης – X

Ο χρήστης μπορεί να υπολογίσει μία μέση απόσταση χρησιμοποιώντας την τεχνική με τους επτά γείτονες που αναφέρθηκε άμα θέλει πατώντας το πλήκτρο **X**. Ωστόσο, δεν υπάρχει κάποιος πρακτικός λόγος να το κάνει αυτό για τα point clouds που έχουν αναφερθεί καθώς αυτές η πληροφορίες παρέχονται στο τέλος της αναφοράς. Η λειτουργία αυτή υλοποιήθηκε καθαρά για την χρήση της κατά την ανάπτυξη της εφαρμογής έτσι ώστε να υπολογιστούν οι μέσες αποστάσεις και δεν παρέχει κάποια έξτρα χρησιμότητα.

Τύπωση των γεωμετριών της σκηνής – V

Τέλος, άλλη μία απλή λειτουργία που προστέθηκε απλά για χρήση κατά την ανάπτυξη της εφαρμογής ήταν το τύπωμα των γεωμετριών που έχουν προστεθεί στην σκηνή. Ο χρήστης άμα θέλει να δει ποιες είναι αυτές μπορεί απλά να πατήσει το πλήκτρο **V**.

References

3D collision detection - Game development | MDN. (2023, June 28). https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_collision_detection

Anaconda. (2023, July 6). *Anaconda | The World's Most Popular Data Science Platform*. https://www.anaconda.com/

Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C. A., & Taubin, G. (1999). The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, *5*(4), 349–359. https://doi.org/10.1109/2945.817351

Open3D – A Modern Library for 3D Data Processing. (n.d.). http://www.open3d.org/

Poux, F., PhD. (2021, December 14). Generate 3D meshes from point clouds with Python | Towards Data Science. *Medium*. https://towardsdatascience.com/5-step-guide-to-generate-3d-meshes-from-point-clouds-with-python-36bad397d8ba

Science, B. O. C., & Science, B. O. C. (2022). Disparity Map in Stereo Vision | Baeldung on Computer Science. *Baeldung on Computer Science*. https://www.baeldung.com/cs/disparity-map-stereo-vision

Shah, U. (2021, December 13). Stereo Vision for 3D Reconstruction - Analytics Vidhya - Medium. *Medium*. https://medium.com/analytics-vidhya/depth-sensing-and-3d-reconstruction-512ed121aa60

The KITTI Vision Benchmark Suite. (n.d.). https://www.cvlibs.net/datasets/kitti/index.php

Wikipedia contributors. (2023a). Random sample consensus. *Wikipedia*. https://en.wikipedia.org/wiki/Random_sample_consensus

Wikipedia contributors. (2023b). Incircle and excircles. *Wikipedia*. https://en.wikipedia.org/wiki/Incircle_and_excircles

Wikipedia contributors. (2023c). DBSCAN. *Wikipedia*. https://en.wikipedia.org/wiki/DBSCAN

Tables

Table 1

RANASAC Distance Thresholds

Point Cloud	Mean Distance	Distance Threshold
Kitti 78	0.027	0.15
Kitti 89	0.034	0.2
Kitti 102	0.027	0.2
Storage Room	1.022	0.1
Researcher’s Desk	0.04	0.01
Adas Lidar	0.4	0.7
Kitchen	0.013	0.05

Note: Η στήλη mean distance περιέχει τα mean distances που υπολογίστηκαν με τους 7 κοντινότερους γείτονες όλων των σημείων του εκάστοτε point cloud όπως αναφέρθηκε. Η στήλη distance threshold έχεις τις τιμές που χρησιμοποιήθηκαν για να βρούμε τους inliers στα plane του αλγόριθμου RANSAC για το κάθε point cloud.

Table 2

Object detection clustering parameters

Point Cloud	Epsilon	Minimum Samples
Kitti 78	0.2	100
Kitti 89	0.5	100
Kitti 102	0.4	100
Storage Room	3.5	30
Researcher’s Desk	0.01	5
Adas Lidar	2.0	20
Kitchen	0.05	5

Note: Η στήλη epsilon είναι οι παράμετροι που ορίζουμε για την ακτίνα της γειτονιάς του αλγόριθμου DBSCAN ενώ η στήλη minimum samples είναι τα ελάχιστα σημεία που θέλουμε για να θεωρούμε μία γειτονία dense και να την επεκτείνουμε.

Table 3

Denoising clustering parameters

Point Cloud	Epsilon	Minimum Samples
Kitti 78	0.1	10
Kitti 89	0.1	10
Kitti 102	0.1	10
Storage Room	10	50

Note: Η στήλη epsilon είναι οι παράμετροι που ορίζουμε για την ακτίνα της γειτονιάς του αλγόριθμου DBSCAN ενώ η στήλη minimum samples είναι τα ελάχιστα σημεία που θέλουμε για να θεωρούμε μία γειτονία dense και να την επεκτείνουμε.