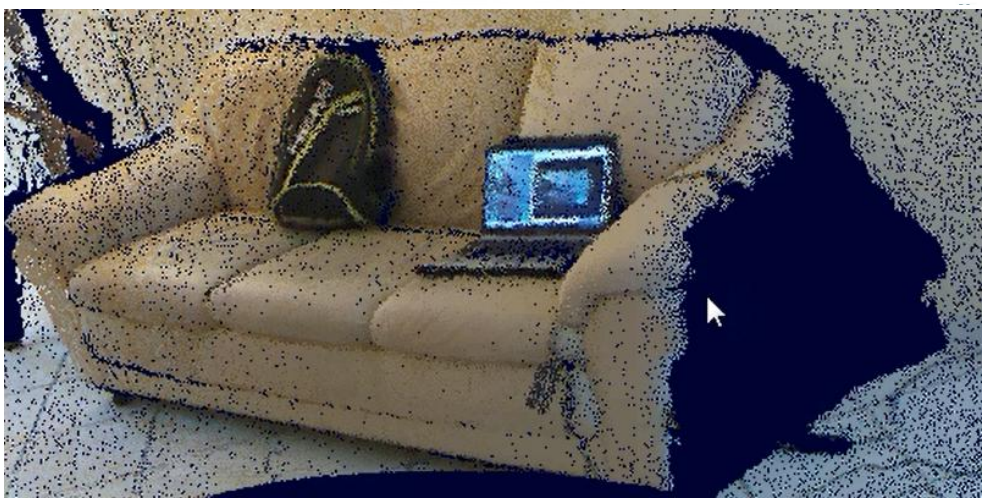
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ

Τρισδιάστατη Χαρτογράφηση Εσωτερικών Χώρων σε Πραγματικό Χρόνο με Χρήση Αισθητήρα Microsoft Kinect for Windows (v1)



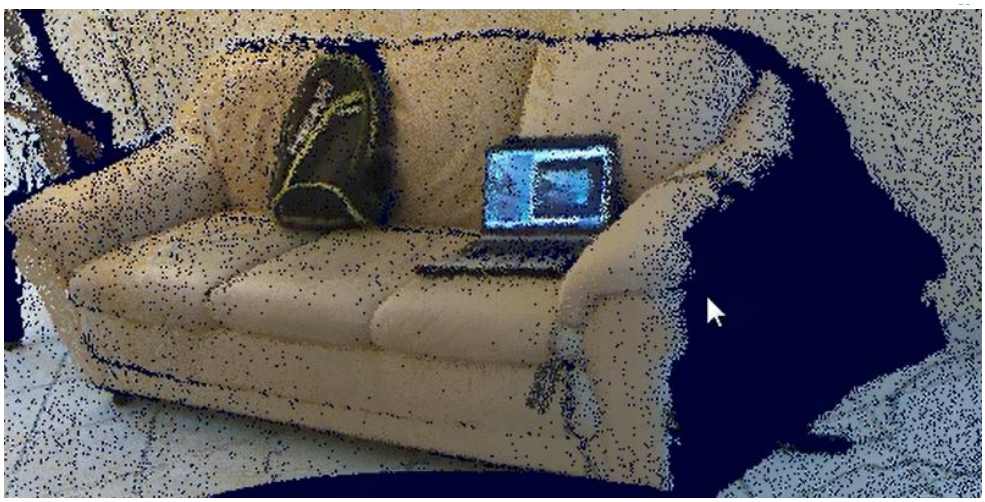
Διπλωματική Εργασία
του
Χριστόπουλου Ανδρέα
(ΑΜ:06112210)

Επιβλέπων:
Δουλάμης Νικόλαος
Επίκ. Καθ. ΕΜΠ

Ακαδημαϊκό έτος 2014 – 2015
Αθήνα, Οκτώβριος 2015

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF RURAL AND SURVEYING ENGINEERING
DEPARTMENT OF TOPOGRAPHY

Real Time Three-Dimensional (3D) Indoor Mapping Using Kinect for Windows (v1)



Diploma Thesis

Christopoulos Andreas

Supervisor:

Doulamis Nikolaos,
Assistant Professor NTUA

Athens, October 2015

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

ΕΥΧΑΡΙΣΤΙΕΣ

Εν αρχή, θα ήθελα να ευχαριστήσω ιδιαιτέρως τον καθηγητή κ. Δουλάμη Νικόλαο για την ανάθεση της παρούσης διπλωματικής εργασίας, τις συμβουλές και την πολύτιμη συνεργασία του.

Επιθυμώ, επίσης, να εκφράσω τις ευχαριστίες μου στα μέλη της τριμελούς επιτροπής, τους καθηγητές κ. Γεωργόπουλο Ανδρέα και Βεσκούκη Βασίλειο, για την αξιολόγηση της εργασίας μου και τη διαθεσιμότητά τους προς επίλυση οιασδήποτε απορίας εγέρθη κατά τη διάρκεια της εκπόνησής της.

Ολοκληρώνοντας, θα ήθελα να ευχαριστήσω τη Δήμητρα, τους φίλους και συναδέλφους μου, καθώς και τα μέλη της οικογενείας μου που στηρίζουν, διαχρονικά και ανελλιπώς, την ακαδημαϊκή και επαγγελματική μου σταδιοδρομία.

Η διπλωματική αυτή εργασία ολοκληρώνει το κύκλο της φοίτησης μου στη Σχολή Αγρονόμων και Τοπογράφων Μηχανικών του Εθνικού Μετσόβιου Πολυτεχνείου. Σημειώνεται πως η φοίτηση αυτή κατέστη δυνατή υπό την αιγίδα του Γεωγραφικού Σώματος το οποίο υπερήφανα υπηρετώ και του είμαι ευγνώμων.

ΠΕΡΙΛΗΨΗ

Η συσκευή Kinect for Windows (v1) είναι αποτέλεσμα πολυετούς και πολυδάπανης έρευνας στο τομέα της υπολογιστικής όρασης. Διετέθη στην αγορά από την εταιρεία Microsoft ως φυσική διεπαφή μεταξύ της δημοφιλούς παιχνιδομηχανής XBOX και των οικιακών χρηστών. Εντούτοις, η συστοιχία αισθητήρων και η τεχνολογία επεξεργασίας ήχου και εικόνας που διαθέτει η συσκευή, ενέπνευσαν πληθώρα ερευνητών να διερευνήσουν το ενδεχόμενο χρησιμοποίησης της για διαφορετικούς σκοπούς – λιγότερο ή περισσότερο δημοφιλείς στο ευρύ κοινό.

Η παρούσα εργασία εξετάζει το ενδεχόμενο χρησιμοποίησης του Kinect για τη τρισδιάστατη χαρτογράφηση εσωτερικών χώρων σε πραγματικό χρόνο, εκμεταλλευόμενοι, μεταξύ άλλων, την έγχρωμη κάμερα και τον πομποδέκτη υπέρυθρης ακτινοβολίας της συσκευής. Δεδομένου ότι τα εργαλεία ανάπτυξης λογισμικού για το Kinect που διαθέτει στην αγορά η Microsoft τυγχάνουν περιορισμών ως προς την εκμετάλλευση και τη διάθεση τους, γίνεται προσπάθεια επίτευξης του στόχου με χρήση ελεύθερου λογισμικού/λογισμικού ανοιχτού κώδικα (ΕΛ/ΛΑΚ) και με τους ελάχιστους δυνατούς συμβιβασμούς σε ότι αφορά τη ποιότητα του τελικού αποτελέσματος. Η προσπάθεια αυτή περιλαμβάνει την υιοθέτηση ικανού πλήθους frameworks, οδηγών (drivers), APIs, middleware και βιβλιοθηκών (libraries), την επίλυση των σχετικών προβλημάτων συμβατότητας προτού στεφθεί με επιτυχία.

Η ποιότητα του παραγόμενου νέφους σημείων εξετάζεται, εν συνεχεία κατ' αντιπαραβολή με τον σαρωτή (laser scanner) FARO focus 3D, προκειμένου να εξαχθούν χρήσιμα συμπεράσματα. Τέλος, τα αποτελέσματα της σύγκρισης αυτής επιτρέπουν να προσδιοριστούν τα δυνητικά πεδία εφαρμογής και τη διατύπωση σχετικών προτάσεων.

Λέξεις κλειδιά: Τρισδιάστατη, Μοντελοποίηση, Kinect, Νέφος σημείων, Αισθητήρας

ABSTRACT

The Kinect for Windows (v1) device is the result of many years of costly research in the field of computer vision. It was marketed by Microsoft as a physical interface between the popular XBOX video games console and home users. However, the sensor array and the audio and video processing technology available to the device, inspired numerous researchers to explore the possibility of alternative uses - more or less popular with the general public.

This paper examines the possible use of Kinect for three-dimensional indoor mapping in real time, by taking advantage of the color camera and the device's infrared transceiver. Since the Kinect software development tools (SDK) marketed by Microsoft suffers from licensing restrictions, an effort is made to achieve the objective by using free and open source software and with the minimum possible compromises in terms of quality. This effort requires the adoption of a number of frameworks, drivers, APIs, middleware and libraries, plus the resolution of certain compatibility problems before it is deemed with success.

The quality of the point cloud is subsequently examined in comparison with the laser scanner FARO focus 3D, in order to draw conclusions regarding the potential fields of application. Finally, the results of this comparison allow for the identification of potential areas of application and the presentation of relevant proposals.

Key Words: 3D, Modeling, Kinect, Point cloud, Sensor

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΧΑΡΙΣΤΙΕΣ **1**

ΠΕΡΙΛΗΨΗ **2**

ABSTRACT **3**

1. ΕΙΣΑΓΩΓΗ **9**

1.1 ΣΤΟΧΟΣ ΤΗΣ ΕΡΓΑΣΙΑΣ 10
1.2 ΔΟΜΗ ΤΗΣ ΕΡΓΑΣΙΑΣ 10

2. Ο ΑΙΣΘΗΤΗΡΑΣ KINECT **11**

2.1 ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ 11
2.2 ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ 13
2.2.1 ΤΟ ΣΥΣΤΗΜΑ ΑΝΙΧΝΕΥΣΗΣ ΒΑΘΟΥΣ 14
2.2.2 Η ΕΓΧΡΩΜΗ ΚΑΜΕΡΑ 15
2.2.3 Η ΣΥΣΤΟΙΧΙΑ ΜΙΚΡΟΦΩΝΩΝ 16
2.2.4 Ο ΚΙΝΗΤΗΡΑΣ ΚΛΙΣΗΣ ΚΑΙ ΤΟ ΕΠΙΤΑΧΥΝΣΙΟΜΕΤΡΟ 16
2.2.5 ΕΜΒΕΛΕΙΑ - ΘΟΡΥΒΟΣ 17

3. ΒΑΣΙΚΕΣ ΑΡΧΕΣ ΛΕΙΤΟΥΡΓΙΑΣ **18**

3.1 TIME - OF - FLIGHT 18
3.2 ΜΟΝΤΕΛΟ ΚΑΜΕΡΑΣ ΜΙΚΡΗΣ ΟΠΗΣ 19
3.3 ΜΑΘΗΜΑΤΙΚΟ ΜΟΝΤΕΛΟ ΕΞΑΓΩΓΗΣ ΒΑΘΟΥΣ 20
3.4 ΔΙΑΚΡΙΤΙΚΗ ΙΚΑΝΟΤΗΤΑ ΚΑΙ ΠΥΚΝΟΤΗΤΑ ΣΗΜΕΙΩΝ 22
3.5 ΒΑΘΜΟΝΟΜΗΣΗ ΑΙΣΘΗΤΗΡΑ 23
3.6 ΜΟΝΤΕΛΟ ΣΦΑΛΜΑΤΟΣ 24

4. ΤΟ ΛΟΓΙΣΜΙΚΟ **26**

4.1 RGBDEMO 0.7.0 26
4.1.1 ΒΙΒΛΙΟΘΗΚΕΣ 27
4.1.2 FRAMEWORKS 28
4.1.3 API 29
4.2 ΕΓΚΑΤΑΣΤΑΣΗ 30
4.2.1 ΔΙΑΔΙΚΑΣΙΑ 30

5. ΝΕΦΗ ΣΗΜΕΙΩΝ **33**

5.1 ΒΑΘΜΟΝΟΜΗΣΗ ΜΕ ΤΟ RGBDEMO 33
5.2 ΔΗΜΙΟΥΡΓΙΑ ΝΕΦΟΥΣ ΣΗΜΕΙΩΝ ΜΕ ΧΡΗΣΗ ΤΗΣ PCL 37
5.3 ΠΑΡΑΓΩΓΗ ΝΕΦΟΥΣ ΣΗΜΕΙΩΝ ΑΠΟ ΕΙΚΟΝΑ ΒΑΘΟΥΣ 38
5.4 ΑΠΟΔΟΣΗ ΧΡΩΜΑΤΟΣ ΣΤΟ ΝΕΦΟΣ ΣΗΜΕΙΩΝ 39
5.5 ΟΠΤΙΚΟΠΟΙΗΣΗ ΤΟΥ ΝΕΦΟΥΣ ΣΗΜΕΙΩΝ 40
5.6 ΑΓΚΙΣΤΡΩΣΗ (REGISTRATION) 42
5.7 Ο ΑΛΓΟΡΙΘΜΟΣ ICP 45
5.8 ΔΙΑΧΕΙΡΙΣΗ ΑΚΡΑΙΩΝ ΤΙΜΩΝ 45
5.9 SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM) 47

6. Η ΕΦΑΡΜΟΓΗ **49**

6.1 ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΧΡΗΣΤΗ 49

6.2 ΜΟΡΦΟΤΥΠΟΣ (FORMAT)....51
6.3 ΠΑΡΑΔΕΙΓΜΑΤΑ....52

7. ΠΟΙΟΤΙΚΟΣ ΕΛΕΓΧΟΣ....57

7.1 ΣΑΡΩΤΗΣ....57
7.2 CLOUDCOMPARE....59
7.3 ΜΕΘΟΔΟΛΟΓΙΑ....59
7.4 ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΥΓΚΡΙΣΗΣ....61
7.4.1 ΜΕ ΚΡΙΤΗΡΙΟ ΤΗΝ ΑΠΟΣΤΑΣΗ....61
7.4.1.1 ΑΠΟΣΤΑΣΗ 1Μ....61
7.4.1.2 ΑΠΟΣΤΑΣΗ 2Μ....70
7.4.1.3 ΑΠΟΣΤΑΣΗ 3Μ....76
7.4.2 ΜΕ ΚΡΙΤΗΡΙΟ ΤΟΝ ΦΩΤΙΣΜΟ....82
7.4.2.1 ΑΠΟΣΤΑΣΗ 1Μ....83
7.4.2.2 ΑΠΟΣΤΑΣΗ 2Μ....86
7.4.2.3 ΑΠΟΣΤΑΣΗ 3Μ....89

8. ΣΥΜΠΕΡΑΣΜΑΤΑ - ΠΡΟΤΑΣΕΙΣ....92

9. ΒΙΒΛΙΟΓΡΑΦΙΑ....95

ΠΑΡΑΡΤΗΜΑ....98

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

Εικόνα 2–1: Το πρωτότυπο Project Natal 12
Εικόνα 2–2: Το MS Kinect for Windows (v1) 13
Εικόνα 2–3: Τα συστατικά μέρη του Kinect for Windows (v1) 13
Εικόνα 2–4: Το οπτικό πεδίο (FoV) του MS Kinect for Windows (v1) 14
Εικόνα 2–5: Το μοτίβο υπερύθρων του MS Kinect for Windows (v1) 15
Εικόνα 2–6: Εντοπισμός πηγής ήχου από τη συστοιχία μικροφώνων του Kinect 16
Εικόνα 2–7: Τα περιθώρια κλίσης του Kinect for Windows (v1) 17
Εικόνα 3–1: Η αρχή λειτουργίας time-of-flight 19
Εικόνα 3–2: Το μοντέλο μικρής οπής 19
Εικόνα 3–3: Το μοτίβο υπερύθρων και η εξαγόμενη εικόνα βάθους 20
Εικόνα 3–4: Σχέση μεταξύ βάθους και μετατόπισης 21
Εικόνα 4–1: Το RDBDemo στο αποθετήριο GitHub 27
Εικόνα 4–2: Τα λογότυπα των επιμέρους λογισμικών. 29
Εικόνα 5–1: Κάναβος τύπου πλάκας ζατρικίου 34
Εικόνα 5–2: Ένα παράδειγμα θέσης λήψης εικόνας κατά τη βαθμονόμηση 35
Εικόνα 5–3: Ο μετασχηματισμός στερεού σώματος 39
Εικόνα 5–4: Απλή επίθεση (αριστερά) έναντι αγκίστρωσης (δεξιά) 42
Εικόνα 5–5: Διάγραμμα ροής της μεθόδου SLAM σε συνδυασμό με το Kinect 47
Εικόνα 6–1: Τα γραφικά στοιχεία του παραθύρου RGB-D Capture 50
Εικόνα 6–2: Τα γραφικά στοιχεία του παραθύρου 3D View 50
Εικόνα 6–3: Το πρώτο καρέ 52
Εικόνα 6–4: Βελτίωση νέφους σημείων προϊόντος του χρόνου 53
Εικόνα 6–5: Επόμενες λήψεις 53
Εικόνα 6–6: Το μοντέλο προϊόντος του χρόνου 54
Εικόνα 6–7: Εναλλακτική όψη 54
Εικόνα 6–8: Επόμενες λήψεις 55
Εικόνα 6–9: Αποτέλεσμα 55
Εικόνα 7–1: Ο στόχος (πεζογέφυρα Ραφήνας) και το αντίστοιχο νέφος σημείων 57
Εικόνα 7–2: Ο σαρωτής laser FARO Focus 3D 58
Εικόνα 7–3: Το λογότυπο του λογισμικού CloudCompare 59
Εικόνα 7–4: Τα αντικείμενα - στόχοι 60
Εικόνα 7–5: Η επιφάνεια εργασίας του CloudCompare με τα προαναφερθέντα εργαλεία 61

Εικόνα 7–6: Η διαδικασία αγκίστρωσης των δύο νεφών 62
Εικόνα 7–7: Το αντίστοιχο C2C γράφημα 63
Εικόνα 7–8: Η εφαρμογή του αλγορίθμου ICP 63
Εικόνα 7–9: Η διαδικασία αγκίστρωσης των δύο νεφών 64
Εικόνα 7–10: Το αντίστοιχο C2C γράφημα 65
Εικόνα 7–11: Η εφαρμογή του αλγορίθμου ICP 65
Εικόνα 7–12: Η διαδικασία αγκίστρωσης των δύο νεφών 66
Εικόνα 7–13: Το αντίστοιχο C2C γράφημα 67
Εικόνα 7–14: Η εφαρμογή του αλγορίθμου ICP 67
Εικόνα 7–15: Η διαδικασία αγκίστρωσης των δύο νεφών 68
Εικόνα 7–16: Το αντίστοιχο C2C γράφημα 69
Εικόνα 7–17: Η εφαρμογή του αλγορίθμου ICP 69
Εικόνα 7–18: Η διαδικασία αγκίστρωσης των δύο νεφών 70
Εικόνα 7–19: Το αντίστοιχο C2C γράφημα 70
Εικόνα 7–20: Η εφαρμογή του αλγορίθμου ICP 71
Εικόνα 7–21: Η διαδικασία αγκίστρωσης των δύο νεφών 71
Εικόνα 7–22: Το αντίστοιχο C2C γράφημα 72
Εικόνα 7–23: Η εφαρμογή του αλγορίθμου ICP 72
Εικόνα 7–24: Η διαδικασία αγκίστρωσης των δύο νεφών 73
Εικόνα 7–25: Το αντίστοιχο C2C γράφημα 74
Εικόνα 7–26: Η εφαρμογή του αλγορίθμου ICP 74
Εικόνα 7–27: Η διαδικασία αγκίστρωσης των δύο νεφών 75
Εικόνα 7–28: Το αντίστοιχο C2C γράφημα 75
Εικόνα 7–29: Η εφαρμογή του αλγορίθμου ICP 76
Εικόνα 7–30: Η διαδικασία αγκίστρωσης των δύο νεφών 76
Εικόνα 7–31: Το αντίστοιχο C2C γράφημα 77
Εικόνα 7–32: Η εφαρμογή του αλγορίθμου ICP 77
Εικόνα 7–33: Η διαδικασία αγκίστρωσης των δύο νεφών 78
Εικόνα 7–34: Το αντίστοιχο C2C γράφημα 78
Εικόνα 7–35: Η εφαρμογή του αλγορίθμου ICP 79
Εικόνα 7–36: Η διαδικασία αγκίστρωσης των δύο νεφών 79
Εικόνα 7–37: Το αντίστοιχο C2C γράφημα 80
Εικόνα 7–38: Η εφαρμογή του αλγορίθμου ICP 80

Εικόνα 7–39: Η διαδικασία αγκίστρωσης των δύο νεφών 81
Εικόνα 7–40: Το αντίστοιχο C2C γράφημα 81
Εικόνα 7–41: Η εφαρμογή του αλγορίθμου ICP 82
Εικόνα 7–42: Η διαδικασία αγκίστρωσης των δύο νεφών 83
Εικόνα 7–43: Το αντίστοιχο C2C γράφημα 83
Εικόνα 7–44: Η εφαρμογή του αλγορίθμου ICP 84
Εικόνα 7–45: Η διαδικασία αγκίστρωσης των δύο νεφών 84
Εικόνα 7–46: Το αντίστοιχο C2C γράφημα 85
Εικόνα 7–47: Η εφαρμογή του αλγορίθμου ICP 85
Εικόνα 7–48: Η διαδικασία αγκίστρωσης των δύο νεφών 86
Εικόνα 7–49: Το αντίστοιχο C2C γράφημα 86
Εικόνα 7–50: Η εφαρμογή του αλγορίθμου ICP 87
Εικόνα 7–51: Η διαδικασία αγκίστρωσης των δύο νεφών 87
Εικόνα 7–52: Το αντίστοιχο C2C γράφημα 88
Εικόνα 7–53: Η εφαρμογή του αλγορίθμου ICP 88
Εικόνα 7–54: Η διαδικασία αγκίστρωσης των δύο νεφών 89
Εικόνα 7–55: Το αντίστοιχο C2C γράφημα 89
Εικόνα 7–56: Η εφαρμογή του αλγορίθμου ICP 90
Εικόνα 7–57: Η διαδικασία αγκίστρωσης των δύο νεφών 90
Εικόνα 7–58: Το αντίστοιχο C2C γράφημα 91
Εικόνα 7–59: Η εφαρμογή του αλγορίθμου ICP 91
Πίνακας 1: Συγκεντρωτικά αποτελέσματα του ποιοτικού ελέγχου 93

1. ΕΙΣΑΓΩΓΗ

Η τρισδιάστατη (3D) απεικόνιση δεν αποτελεί νέο πεδίο επιστημονικής έρευνας, καθώς οι επιστήμονες ασχολούνται με την τρισδιάστατη πληροφορία τις τελευταίες δεκαετίες. Εντούτοις, μόλις πρόσφατα έγινε δημοφιλής στο ευρύ κοινό χάρη στην έλευση σχετικών καταναλωτικών και βιομηχανικών προϊόντων.

Οι τρισδιάστατες ταινίες υπάρχουν στη ζωή μας από την εποχή των γνωστών πολωτικών γυαλιών με μπλε και κόκκινους φακούς. Εντούτοις, το φιλοθεάμων κοινό έστρεψε τη πλάτη του στην τεχνολογία, καθώς η εικόνα υστερούσε σε ποιότητα. Η απεικόνιση της τρισδιάστατης πληροφορίας μειονεκτούσε, καθώς οι ερευνητές περιορίζοντο στη χρήση χαμηλής υπολογιστικής ισχύος και μνήμης. Η πρόοδος της επιστήμης της πληροφορικής και η συνακόλουθη έλευση πολυπύρηνων επεξεργαστών, ισχυρών καρτών γραφικών και μεγάλων μονάδων μνήμης, επέτρεψε στην τεχνολογία της τρισδιάστατης απεικόνισης να καλύψει το χαμένο έδαφος ετών.

Οι τεχνολογίες τρισδιάστατης απεικόνισης διακρίνονται σε τεχνολογίες επαφής (contact), μετάδοσης (transmissive) και ανάκλασης (reflective). Οι μεν πρώτες, όπως προδίδει το όνομα τους, απαιτούν την φυσική επαφή με το αντικείμενο προκειμένου να εξάγουν το τρισδιάστατο μοντέλο του και ως εκ τούτου είναι χρονοβόρες και πολυδάπανες. Οι τεχνολογίες μετάδοσης δεν απαιτούν την επαφή με το αντικείμενο και είναι πολύ δημοφιλείς στον χώρο της ιατρικής. Σε αυτές οφείλεται η ύπαρξη μαγνητικών και αξονικών τομογράφων που έφεραν την επανάσταση της ιατρικής επιστήμης.

Οι ανακλαστικές μέθοδοι βασίζονται σε οπτικές ή μη οπτικές πηγές, ανάλογα την περίπτωση χρήσης. Οι μη οπτικές πηγές περιλαμβάνουν συσκευές ραδιοεντοπισμού (radar), ηχοεντοπισμού (sonar) και υπερήχων. Είναι προφανές πως η τεχνολογία αυτή συναντάται σε μεγάλο εύρος εφαρμογών με κόστος που ποικίλει από μερικές εκατοντάδες έως μερικές χιλιάδες ευρώ. Οι ανακλαστικές μέθοδοι που βασίζονται σε οπτικές πηγές είναι αυτές που κατά τεκμήριο συναντώνται στη καθημερινότητα του μέσου ανθρώπου. Σε αυτές βασίζονται δημοφιλείς καθημερινές συσκευές όπως 3D TVs και βιντεοπαιχνίδια.

Οι ανακλαστικές μέθοδοι που βασίζονται σε οπτικές πηγές μπορούν να διακριθούν περαιτέρω σε ενεργές και παθητικές. Οι παθητικές χρησιμοποιούν ενδείξεις βάθους όπως εστίαση (focus), αποεστίαση (defocus), υφή (texture), κίνηση (motion), στερεοσκοπία (stereo) και σκίαση (shading). Οι ενεργές μέθοδοι επιστρατεύουν τη προβολή ακτινοβολίας και μοτίβων για την εξαγωγή της τρισδιάστατης πληροφορίας. Οι τελευταίες, βρίσκονται και στο επίκεντρο της τεχνολογίας που αναλύεται στη παρούσα διπλωματική εργασία και με εφαρμογή στη τρισδιάστατη χαρτογράφηση εσωτερικών χώρων. [1]

1.1 Στόχος της εργασίας

Αντικείμενο της διπλωματικής αυτής εργασίας αποτελεί η εξερεύνηση της δυνατότητας για τη τρισδιάστατη χαρτογράφηση εσωτερικών χώρων σε πραγματικό χρόνο με χρήση του (πολύ)αισθητήρα Kinect for Windows (v1).

Βασική επιδίωξη της εργασίας είναι επίτευξη του παραπάνω στόχου με χρήση ελεύθερου λογισμικού/λογισμικού ανοικτού κώδικα (RGBDemo), και τους ελάχιστους δυνατούς συμβιβασμούς σε ότι αφορά τη ποιότητα του τελικού αποτελέσματος. Η προϋπόθεση αυτή επιτυγχάνεται με την εφαρμογή τεχνικών (SLAM) και αλγορίθμων (ICP) στη στάθμη της τεχνικής (state of the art).

Δευτερεύον στόχο της εργασίας αποτελεί η αξιολόγηση της ποιότητας των αποτελεσμάτων σε συνάρτηση με τις τεχνικές δυνατότητες της συσκευής και το σχετικό κόστος. Στη κατεύθυνση αυτή, έγινε σύγκριση των αποτελεσμάτων με αντίστοιχες μετρήσεις από συσκευή laser scanner, πολλαπλάσιας αξίας. Η σύγκριση αυτή επιτρέπει την εξαγωγή συμπερασμάτων για το είδος και εύρος των εφαρμογών που δύναται να εξυπηρετηθούν από το συνδυασμό Kinect και RGBDemo.

1.2 Δομή της εργασίας

Η διπλωματική αυτή εργασία αποτελείται από δέκα (10) συνολικά κεφάλαια, που αναλύονται ως εξής:

- Το 1^{ο} κεφάλαιο αποτελεί την εισαγωγή της εργασίας και επιπλέον προσδιορίζει το αντικείμενο της.
- Το 2^{ο} κεφάλαιο επικεντρώνεται στο Kinect και τα τεχνικά του χαρακτηριστικά.
- Το 3^{ο} κεφάλαιο αναλύει τις βασικές αρχές λειτουργίας της συσκευής και τη μαθηματική τους τεκμηρίωση.
- Το 4^{ο} κεφάλαιο είναι αφιερωμένο στο λογισμικό που επελέγη για την εκπόνηση της εργασίας, με έμφαση στα βασικά του μέρη και τη διαδικασία εγκατάστασης.
- Το 5^{ο} κεφάλαιο αναφέρεται διεξοδικά στα νέφη σημείων και δη, στον τρόπο δημιουργίας και διαχείρισης τους με το επιλεχθέν λογισμικό.
- Το 6^{ο} κεφάλαιο αναλώνεται στην επίδειξη της εφαρμογής και των αποτελεσμάτων που παράγει.
- Το 7^{ο} κεφάλαιο περιλαμβάνει τη σύγκριση των παραπάνω αποτελεσμάτων με τα αντίστοιχα του laser scanner.
- Στο 8^{ο} κεφάλαιο εξάγονται συμπεράσματα από τη παραπάνω σύγκριση, σε ότι αφορά την ποιότητα των αποτελεσμάτων και τα προτεινόμενα πεδία εφαρμογής.
- Το 9^{ο} κεφάλαιο εμφαίνει αναλυτικά τη βιβλιογραφία.
- Τέλος, το παράρτημα περιέχει βασικά τμήματα του χρησιμοποιούμενου κώδικα και επιπλέον φωτογραφικό υλικό.

2. Ο ΑΙΣΘΗΤΗΡΑΣ KINECT

Ο αισθητήρας Kinect είναι αποτέλεσμα έρευνας δεκαετιών και δισεκατομμυρίων δολαρίων και ηγείται της επανάστασης της υπολογιστικής όρασης (computer vision - CV). Μολονότι το φάσμα των εφαρμογών του ποικίλει, μπορεί να συνοψιστεί ως εξής: για πρώτη φορά οι υπολογιστές μπορούν να δουν στερεοσκοπικά και να ερμηνεύσουν το περιβάλλον τους όπως ακριβώς ο άνθρωπος. Ως εκ τούτου, οι δυνατότητες που ανοίγονται, είναι απεριόριστες.

2.1 Ιστορική αναδρομή

Μολονότι ο ιστορικός του μέλλοντος θα μπορεί να εντοπίσει τις ρίζες του Kinect σε τεχνολογίες που αναπτύχθηκαν από προγράμματα έρευνας και ανάπτυξης που χρηματοδοτήθηκαν προκειμένου οι υπηρεσίες πληροφοριών να έχουν τη δυνατότητα της αναγνώρισης τρομοκρατικών στοιχείων με χρήση αισθητήρων σε δημόσια μέρη, η ιστορική αυτή αναδρομή θα περιοριστεί στον αισθητήρα Kinect καθαυτό, όπως αναπτύχθηκε στα εργαστήρια της Microsoft.

Το 2005, στο Tokyo Game Show, η εταιρεία Nintendo παρουσίασε την κονσόλα ηλεκτρονικών παιχνιδιών Wii της οποίας το χειριστήριο, ονόματι Wii Remote, είχε τη δυνατότητα ανίχνευσης κίνησης σε τρεις (3) άξονες. Το χειριστήριο αυτό διέθετε επίσης έναν οπτικό αισθητήρα που δύνατο να ανιχνεύσει που στοχεύει ανά πάσα στιγμή. Ένα επιπλέον πλεονέκτημα του χειριστηρίου αυτού έναντι του ανταγωνισμού, ήταν το γεγονός πως λειτουργούσε με μπαταρίες και κατά συνέπεια καταργούσε τα καλώδια και επέτρεπε επιπλέον ελευθερία κίνησης για το χρήστη.

Ο τότε επικεφαλής του τομέα ανάπτυξης ηλεκτρονικών παιχνιδιών της Microsoft, ονόματι Peter Moore, αποφάσισε την ανάπτυξη μιας ανταγωνιστικής τεχνολογίας. Η εταιρεία συνέθεσε δύο (2) ανταγωνιστικές ομάδες ανάπτυξης, οι οποίες θα εργαζόταν στη κατεύθυνση αυτή, η μεν πρώτη σε συνεργασία με την εταιρεία PrimeSense, η δε δεύτερη σε συνεργασία με την εταιρεία 3DV. Μολονότι το έτος - στόχος 2007 παρήλθε δίχως απτά αποτελέσματα και ο επικεφαλής του προγράμματος παραιτήθηκε, ο ιδρυτής της Microsoft, Bill Gates δήλωσε σε συνέντευξή του στο συνέδριο D: All Things Digital πως το μέλλον είναι η μηχανική όραση και η αναγνώριση προτύπων (vision recognition), φανερώνοντας πως κάτι σημαντικό βρίσκεται στα σπάργανα.

Ο αντικαταστάτης του Peter Moore, ονόματι Don Matrick, έδωσε το πράσινο φως στη τεχνολογία της PrimeSense και έθεσε επικεφαλής τον Alex Kipman, ενώ και η 3DV εξαγοράστηκε έναντι 35 εκατομμυρίων δολαρίων για το σύνολο των ευρεσιτεχνιών της. Έτσι γεννήθηκε το Project Natal, όπως ονομάστηκε από τη γενέτειρα πόλη του

Alex Kipman στη Βραζιλία, με στόχο τη κατασκευή μιας συσκευής ικανής για αναγνώριση βάθους (depth recognition), παρακολούθηση κίνησης (motion tracking), αναγνώριση προσώπου (facial recognition) και αναγνώρισης ομιλίας (speech recognition).

Η πρωτότυπη συσκευή περιείχε μια RGB κάμερα, έναν υπέρυθρο αισθητήρα βάθους και ένα πομπό υπερύθρων, καθώς και ένα PS1080 chip - ικανό να επεξεργαστεί δεδομένα βάθους σε 30fps. Η επεξεργασία αυτή ήταν πλέον εφικτή με χαμηλό υπολογιστικό κόστος, χάριν μιας νέας μεθόδου ονόματι «time of flight", η οποία υπολόγιζε το χρόνο που χρειάζεται μια δέσμη φωτός να διανύσει την απόσταση από και προς τον αισθητήρα. Εν συνεχεία προστέθηκαν τέσσερα κατευθηντικά μικρόφωνα για την αναγνώριση ομιλίας σε μεγάλους χώρους, καθώς και ένας κινητήρας κλίσης (tilt motor).

Εικόνα 2–1: Το πρωτότυπο Project Natal
Πηγή: www.xboxprojectnatalblog.com

Το project Natal πήρε το πράσινο φως για τη κυκλοφορία του στην παγκόσμια αγορά τα Χριστούγεννα του 2010 ως περιφερειακό εξάρτημα για τη παιχνιδομηχανή της Microsoft, ονόματι XBOX 360. [2] [7] [29] [30]

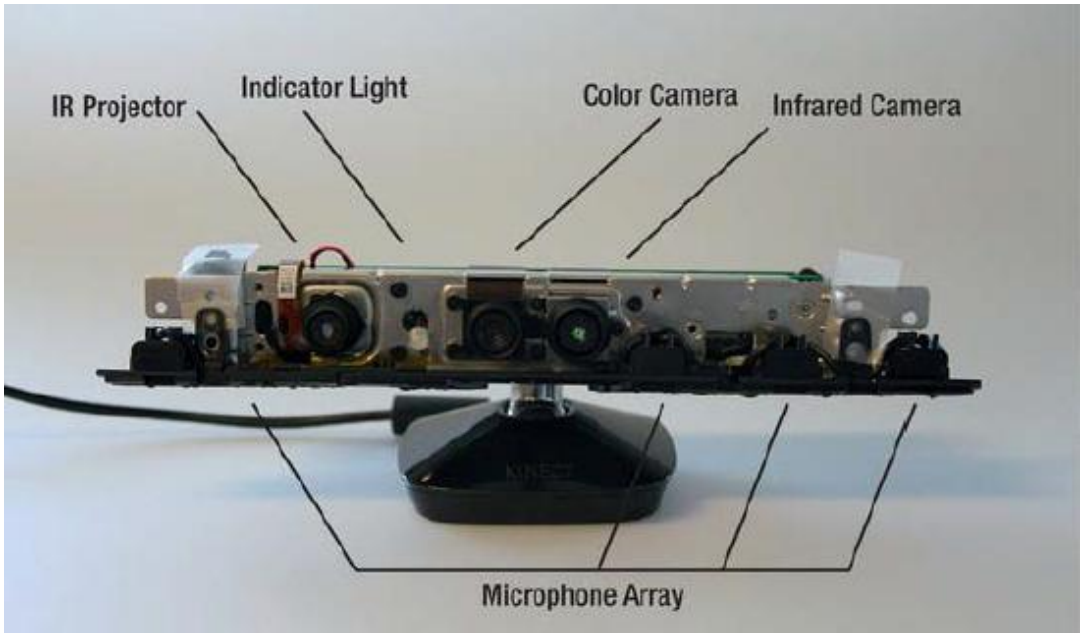
Τα τεχνικά χαρακτηριστικά του τελικού προϊόντος παρατίθενται αναλυτικά παρακάτω.

2.2 Τεχνικά Χαρακτηριστικά

Το Kinect for Windows παρουσιάστηκε την 7η Ιουνίου 2011 και αποτελεί μια πανομοιότυπη συσκευή με τη πρωτότυπη, όσον αφορά τα εξωτερικά και τεχνικά του χαρακτηριστικά, ενώ συνοδεύεται από οδηγούς (drivers) για την απευθείας σύνδεση της με Η/Υ, δίχως να απαιτείται χρήση προσαρμογέα (adaptor).

Εικόνα 2–2: Το MS Kinect for Windows (v1)
Πηγή: www.ubergizmo.com

Αφαιρώντας το περίβλημα (δεν συστήνεται) διακρίνονται, από αριστερά προς τα δεξιά, ο πομπός υπερύθρων (IR projector), το φως ένδειξης της κατάστασης λειτουργίας (indicator light), η έγχρωμη κάμερα RGB (color camera) και ο αισθητήρας υπερύθρων (IR camera), ενώ κατά μήκος της κάτω πλευράς της συσκευής συναντάται μια συστοιχία τεσσάρων (4) μικροφώνων.

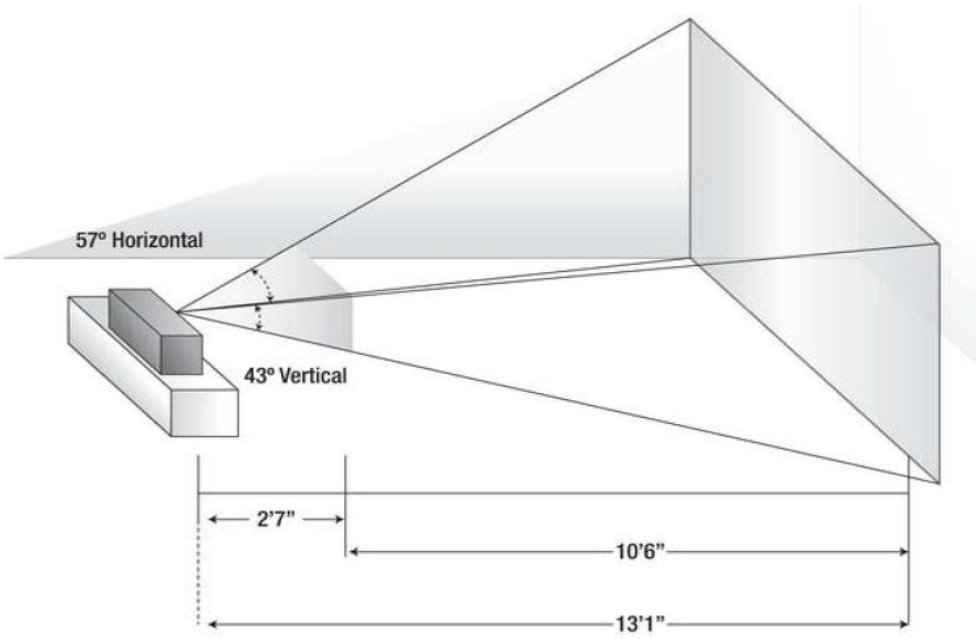


Εικόνα 2–3: Τα συστατικά μέρη του Kinect for Windows (v1)
Πηγή: www.ifixit.com

Πιο αναλυτικά:

2.2.1 Το σύστημα ανίχνευσης βάθους

Το σύστημα ανίχνευσης βάθους αποτελείται από τον πομπό και την κάμερα υπερύθρων. Ο πομπός υπέρυθρης ακτινοβολίας εκπέμπει ένα ψευδοτυχαίο μοτίβο 34.749 υπέρυθρων κουκκίδων στα 830nm - και ως εκ τούτου αόρατων δια γυμνού οφθαλμού - στο σύνολο του οπτικού πεδίου της συσκευής, ήτοι 43^{o} στον κάθετο και 57^{o} στον οριζόντιο άξονα για σταθερό μηχανισμό κλίσης. [2] [29] [30]

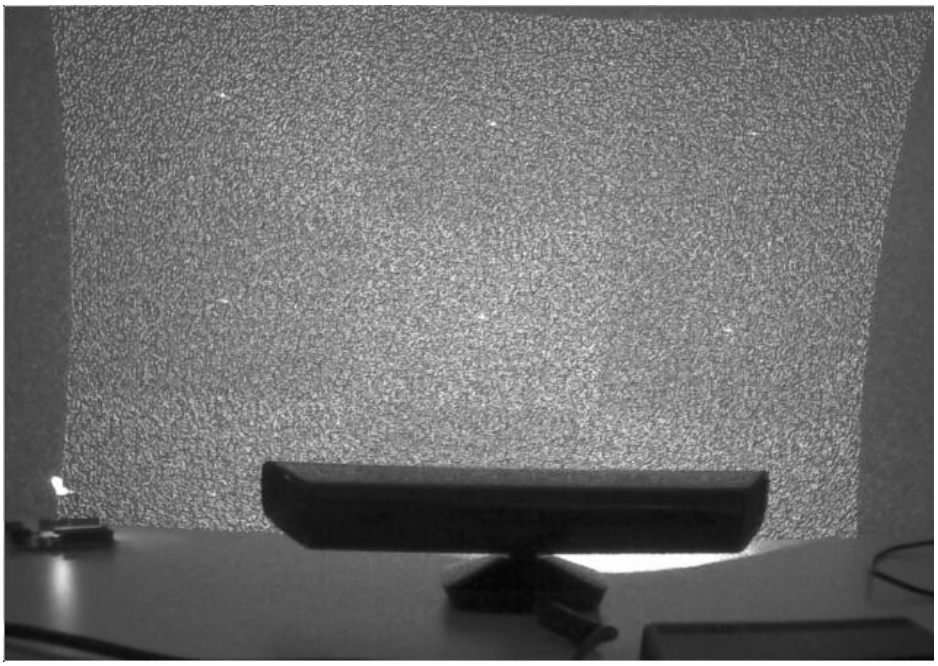


Εικόνα 2–4: Το οπτικό πεδίο (FoV) του MS Kinect for Windows (v1)
Πηγή: www.cnblogs.com

Οι κουκκίδες αυτές ανακλώνται στα αντικείμενα του χώρου και προσλαμβάνονται από τον αισθητήρα υπερύθρων, οπότε και συγκρίνονται με το θεωρητικό μοντέλο και τυχούσες διαφορές ερμηνεύονται από τη συσκευή ως προς τη θέση τους στο χώρο (εγγύτερα ή μακρύτερα), για τη εξαγωγή μιας εικόνας βάθους (depth image). Η προσέγγιση αυτή προϋποθέτει σταθερό μήκος κύματος για την εκπεμπόμενη ακτινοβολία και περιορίζει την εμβέλεια της συσκευής σύμφωνα την ισχύ του πομπού, ενώ το περιβάλλον φως μπορεί να οδηγήσει σε σφάλματα.

Η σταθερότητα του εκπεμπόμενου μήκους κύματος επιτυγχάνεται από τη συσκευή με χρήση μιας μονάδας ψύξης/θέρμανσης της διόδου laser, η οποία την διατηρεί σε σταθερή θερμοκρασία. Σε ότι αφορά τη διαχείριση των συνεπειών του περιβάλλοντος φωτισμού, αυτή επιτυγχάνεται με την εφαρμογή ενός υψιπεράτου φίλτρου στα 830nm επί της κάμερας υπερύθρων, το οποίο βελτιώνει αλλά δεν θεραπεύει το πρόβλημα. Τέλος, σε ότι αφορά την εμβέλεια και δεδομένου ότι η ισχύς των περίπου 70mW δεν

είναι ασφαλής για το ανθρώπινο μάτι, χρησιμοποιείται μια ευρεσιτεχνία της PrimeSense, η οποία διανέμει το φωτεινό κέντρο της εκπομπής σε εννέα (9) επιμέρους κατανεμημένα (υπό)κέντρα, επιτρέποντας τη χρήση ισχυρότερης διόδου laser. [2] [29] [30]



Εικόνα 2–5: Το μοτίβο υπερύθρων του MS Kinect for Windows (v1)
Πηγή: http://livingplace.informatik.haw-hamburg.de/

Η παραγόμενη εικόνα βάθους μπορεί να είναι ανάλυσης 80 x 60, 320 x 240 και 640 x 480 px, ανάλογα με τις επιθυμίες του χρήστη. Η ανάλυση αυτή υπαγορεύει πως το εκπεμπόμενο μοτίβο καθαυτό είναι μεγαλύτερης ανάλυσης, ενδεχομένως 1600 x 1200 px όπως υποδηλώνεται σε έγγραφα τεκμηρίωσης (documentation papers) της PrimeSense, χωρίς ωστόσο να επιβεβαιώνεται από τη Microsoft.

2.2.2 Η έγχρωμη κάμερα

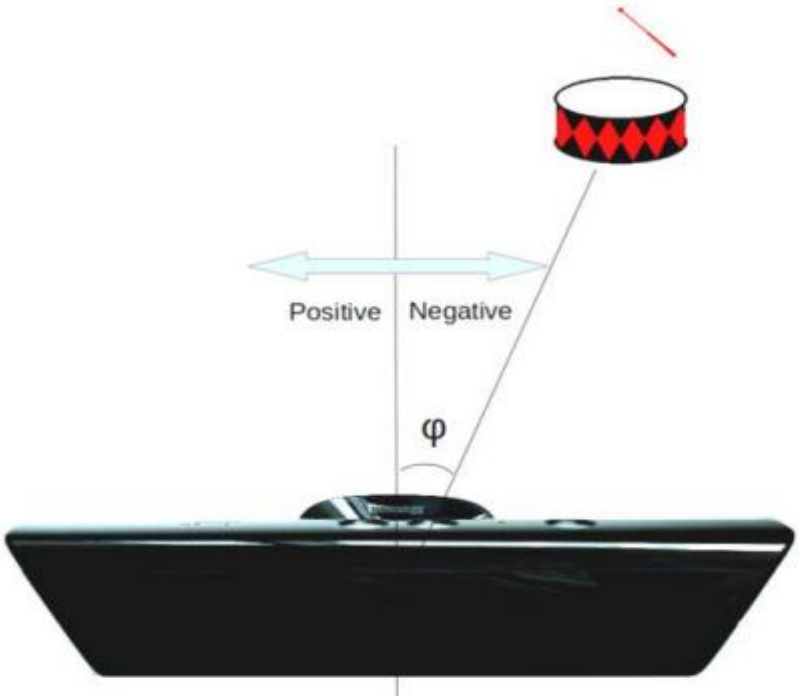
Η έγχρωμη κάμερα (RGB - color camera) δύναται να συλλάβει 30 καρέ ανά δευτερόλεπτο για εικόνα ανάλυσης 80 x 60, 320 x 240 και 640 x 480 px. Εναλλακτικά μπορεί να συλλάβει 12 καρέ ανά δευτερόλεπτο για τη μέγιστη ανάλυση εικόνας των 1280 x 960 px. Σε κάθε περίπτωση το οπτικό πεδίο παραμένει ίδιο με αυτό της της υπέρυθρης κάμερας.

Η κάμερα, αν και συνηθισμένη ως προς τα τεχνικά χαρακτηριστικά, διαθέτει ένα εξαιρετικό σύνολο λειτουργιών, όπως automatic white balancing, black reference, flicker avoidance, color saturation και defect correction. [2] [29] [30]

Μπορεί η έγχρωμη κάμερα αφ' εαυτού δεν παρουσιάζει ιδιαίτερο ενδιαφέρον, εντούτοις σε συνδυασμό με τον αισθητήρα υπερύθρων μπορούν να αποτελέσουν βάση για πλήθος εφαρμογών μεταξύ των οποίων και το αντικείμενο της παρούσης.

2.2.3 Η συστοιχία μικροφώνων

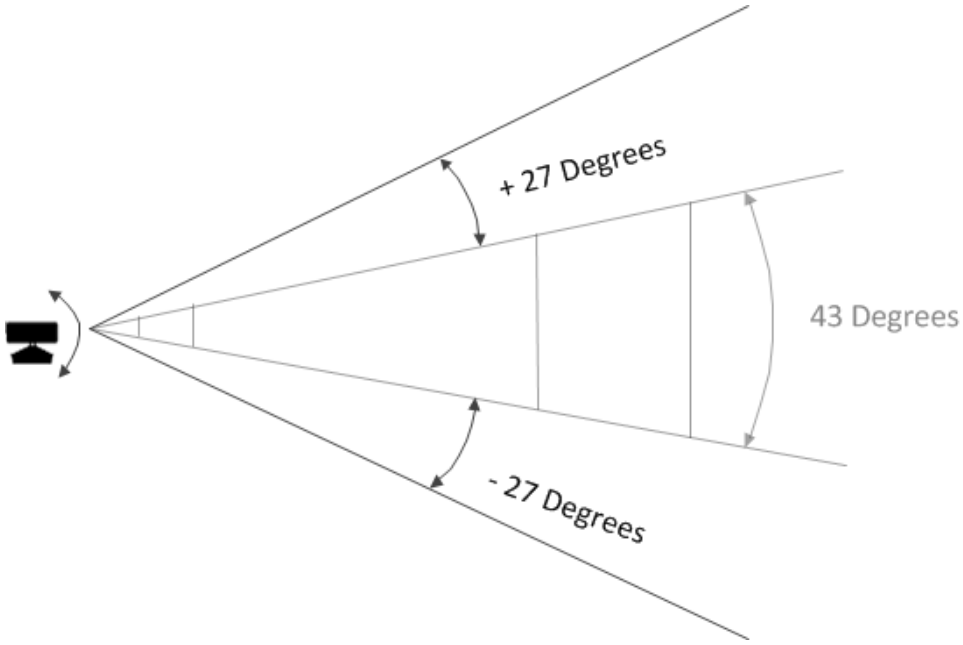
Τα μικρόφωνα στη βάση της συσκευής αποσκοπούν στη σύλληψη του φυσικού ήχου εντός του δωματίου. Ωστόσο, αυτό θα μπορούσε να επιτευχθεί με χρήση ενός μόνο μικροφώνου. Η συστοιχία τεσσάρων μικροφώνων επιτρέπει στη συσκευή τον εντοπισμό στο χώρο της πηγής του ήχου, επιτρέποντας φωνητικές εντολές από αρκετούς χρήστες παράλληλα. [2] [29] [30]



Εικόνα 2–6: Εντοπισμός πηγής ήχου από τη συστοιχία μικροφώνων του Kinect
Πηγή: G. Galatas, S. Ferdous, F. Makedon, "Multimodal Person Localization and Emergency Detection Using The Kinect", 2013

2.2.4 Ο κινητήρας κλίσης και το επιταχυνσιόμετρο

Ο κινητήρας κλίσης του Kinect είναι το συνδετικό στοιχείο μεταξύ του στελέχους των αισθητήρων και της βάσης της συσκευής. Παρέχει τη δυνατότητα χειροκίνητης κλίσης του αισθητήρα κατά $\pm 27^{o}$ από το οριζόντιο επίπεδο. Ο προσδιορισμός της κλίσης του στελέχους γίνεται με χρήση ενός τυπικού επιταχυνσιόμετρου (accelerometer). Εντούτοις, το επιταχυνσιόμετρο μπορεί να χρησιμοποιηθεί και για την τρισδιάστατη ανακατασκευή του χώρου (3d scene reconstruction), όπως θα δούμε σε επόμενο κεφάλαιο. [2] [29] [30]



Εικόνα 2–7: Τα περιθώρια κλίσης του Kinect for Windows (v1)
Πηγή: www.msdn.microsoft.com

2.2.5 Εμβέλεια - Θόρυβος

Ολοκληρώνοντας την ενότητα αυτή, γίνεται μια σύντομη αναφορά στην εμβέλεια των αισθητήρων, η οποία ορίζεται μεταξύ 0.5m (σε near mode) και 4m, ενώ συστήνεται η χρήση τους για αποστάσεις 0.8m έως 3.5m, δεδομένου ότι τα σφάλματα λόγω θορύβου είναι μη γραμμικά και ανέρχονται από 1.5mm στο 0.5m έως 5cm στα 5m. Εντούτοις, οι τιμές αυτές προέρχονται από την PrimeSense και δεν επιβεβαιώνονται από τη Microsoft.

3. ΒΑΣΙΚΕΣ ΑΡΧΕΣ ΛΕΙΤΟΥΡΓΙΑΣ

Μολονότι ο (πολύ)αισθητήρας Kinect σχεδιάστηκε από τη Microsoft ως φυσική διεπαφή για παιχνιδομηχανές ή/και Η/Υ, σύντομα προσέλκυσε το ενδιαφέρον επιστημόνων από τα πεδία της χαρτογραφίας και της τρισδιάστατης μοντελοποίησης χάριν κυρίως της ικανότητας του να καταγράφει δεδομένα βάθους.

Ο συνδυασμός δεδομένων βάθους και χρώματος για κάθε λήψη, παράγει ένα νέφος 300.000 περίπου σημείων - πυκνότητα που μπορεί να αυξηθεί περαιτέρω με χρήση διαδοχικών επικαλυπτόμενων εικόνων. Η διαδικασία αυτή είναι ικανή να παραγάγει νέφη σημείων για τη χαρτογράφηση εσωτερικών χώρων σε πραγματικό χρόνο, που είναι και το αντικείμενο της παρούσης.

Ακολουθεί μια παρουσίαση της μεθοδολογίας που ακολουθείται για τη συλλογή, διαχείριση και επεξεργασία των δεδομένων βάθους που συλλέγει η συσκευή.

3.1 Time - of - flight

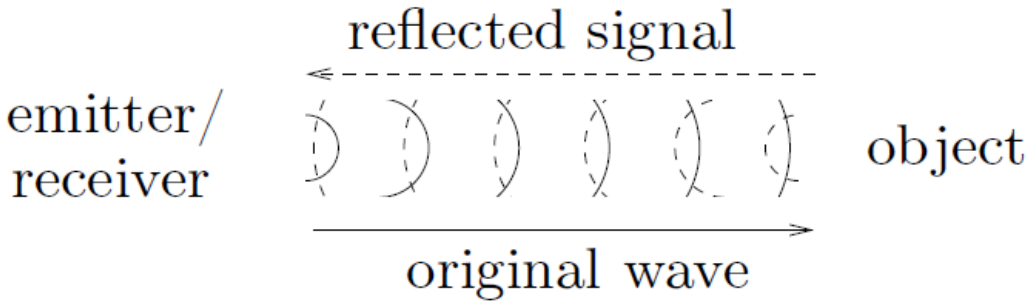
Τα συστήματα time - of - flight βασίζονται στη μέτρηση της διαφοράς χρόνου μεταξύ εκπεμπόμενου και λαμβανόμενου σήματος. Αν t είναι ο μετρηθείς χρόνος και u η ταχύτητα του σήματος, τότε η διανυθείσα απόσταση s υπολογίζεται ως εξής:

$$s = u * t \quad (1)$$

Συνήθως, τόσο ο πομπός όσο και ο αισθητήρας / δέκτης είναι επί της ιδίας συσκευής και σε μικρή μεταξύ τους απόσταση (εν προκειμένω περίπου 7,5 cm). Επομένως, μπορεί να υποτεθεί πως η απόσταση πομπού – αντικειμένου ισούται με την απόσταση αντικειμένου – δέκτη. Η απόσταση αυτή υπολογίζεται ως εξής:

$$D = u\frac{t}{2} \quad (2)$$

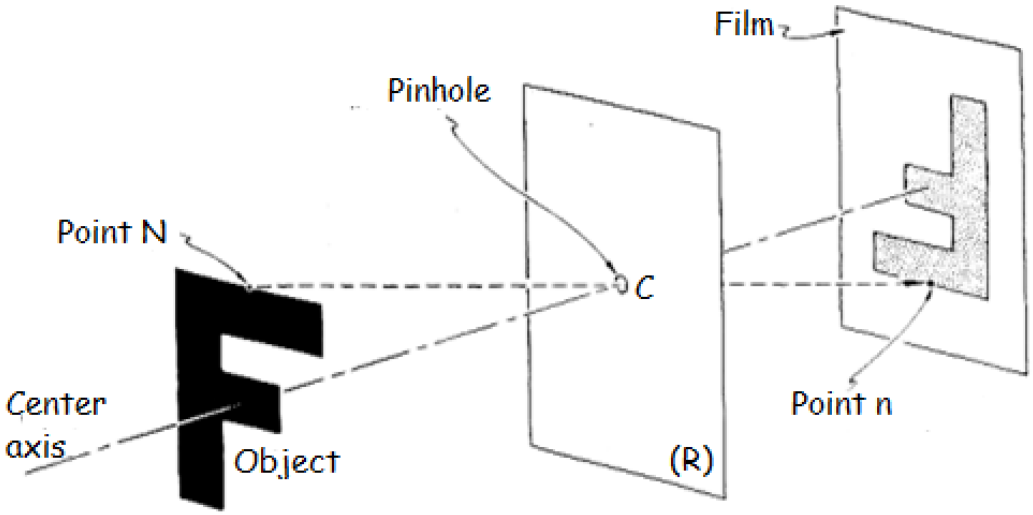
Ενδεικτικά, η ταχύτητα του ήχου στον αέρα είναι u = 343 m/s και η ταχύτητα του φωτός στο κενό ισούται με c = 299792458 m/s. Η ταχύτητα του φωτός είναι τέτοια που επιβάλει τη χρήση χρονομέτρων υψηλής ακρίβειας, καθώς η μέτρηση διαφοράς μήκους 1 cm απαιτεί ακρίβεια χρόνου της τάξης των picosecond. [3] [9] [11]



Εικόνα 3–1: Η αρχή λειτουργίας time-of-flight
Πηγή: A. Nuchter, "3D Roboting Mapping", 2009

3.2 Μοντέλο κάμερας μικρής οπής

Η βαθμονόμηση της συσκευής γίνεται με χρήση του μοντέλου κάμερας μικρής οπής (pinhole camera model). Πρόκειται για ένα απλό γεωμετρικό μοντέλο προβολικού μετασχηματισμού του τρισδιάστατου χώρου στο δισδιάστατο επίπεδο της εικόνας. Οι οπτικές ακτίνες που ανακλώνται στο αντικείμενο, διέρχονται από μια μικρή οπή στην αδιαφανή οθόνη και προσπίπτουν στο επίπεδο της εικόνας όπου δημιουργείται ένα ανεστραμμένο είδωλο.



Εικόνα 3–2: Το μοντέλο μικρής οπής
Πηγή: www.wesjones.com

Το σημείο N του τρισδιάστατου χώρου απεικονίζεται στο σημείο n της εικόνας, ήτοι στο σημείο όπου τέμνει η οπτική ακτίνα το επίπεδο της εικόνας ή επίπεδο της ίριδας όπως είναι γνωστό. Το σημείο C όπου εντοπίζεται η οπή, καλείται οπτικό κέντρο ή εστία της κάμερας και η απόστασή του από το επίπεδο της εικόνας καλείται εστιακή

απόσταση (f). Η ευθεία που διέρχεται από το C και είναι κάθετη στο επίπεδο της ίριδας (R) καλείται οπτικός άξονας και το επίπεδο που περιέχει το C και είναι παράλληλο στο επίπεδο της ίριδας καλείται εστιακό επίπεδο.

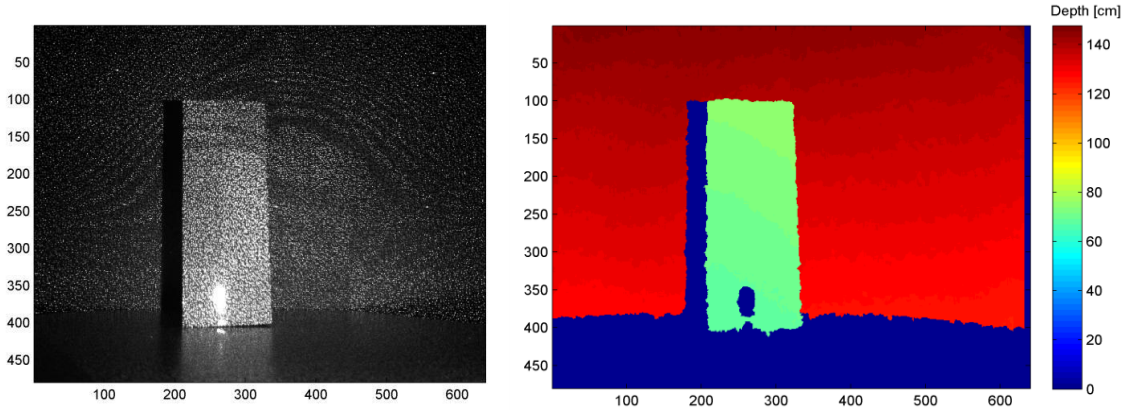
Οι μη γραμμικές παραμορφώσεις που εισάγουν οι φακοί έχουν ως αποτέλεσμα τα N, C και n να μην είναι συνευθειακά, εντούτοις το απλό αυτό μοντέλο, σε συνδυασμό με τις αρχές της προβολικής γεωμετρίας, περιγράφει ικανοποιητικά τη λειτουργία της κάμερας. Περαιτέρω ανάλυση του μοντέλου γίνεται στην ενότητα της βαθμονόμησης.

3.3 Μαθηματικό μοντέλο εξαγωγής βάθους

Η διαδικασία προσδιορισμού της τιμής του βάθους είναι γνωστή ως τριγωνισμός (triangulation). Η δίοδος laser εκπέμπει μια δέσμη υπέρυθρης ακτινοβολίας η οποία εν συνεχεία περιθλάται, για τη δημιουργία ενός μοτίβου. Το στιγμιότυπο (instance) του μοτίβου αυτού σε ορισμένη απόσταση από το δέκτη, αποτελεί και το πρότυπο επίπεδο αναφοράς του δέκτη. Κάθε κουκίδα υπερύθρων που προβάλλεται σε αντικείμενο που βρίσκεται εγγύτερα ή μακρύτερα του επιπέδου αυτού, θα μετατοπιστεί προς τη κατεύθυνση της γραμμής βάσης (baseline) μεταξύ της διόδου laser και του προοπτικού κέντρου της υπέρυθρης κάμερας. Οι μετατοπίσεις για το σύνολο των κουκίδων μετρώνται και μέσω μιας απλής διαδικασίας συσχέτισης (correlation), υπολογίζεται ένας πίνακας διασποράς (disparity matrix).

Ισχύει: $$disparity = \frac{baseline * f}{Z} \quad (3)$$

Η απόσταση κάθε εικονοστοιχείου (pixel) από τον αισθητήρα μπορεί να υπολογιστεί από αυτόν ακριβώς το πίνακα, σύμφωνα με το μαθηματικό μοντέλο υπολογισμού του βάθους που ακολουθεί. [3]



Εικόνα 3–3: Το μοτίβο υπερύθρων και η εξαγόμενη εικόνα βάθους
Πηγή: K. Khoshelham, S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications", 2012

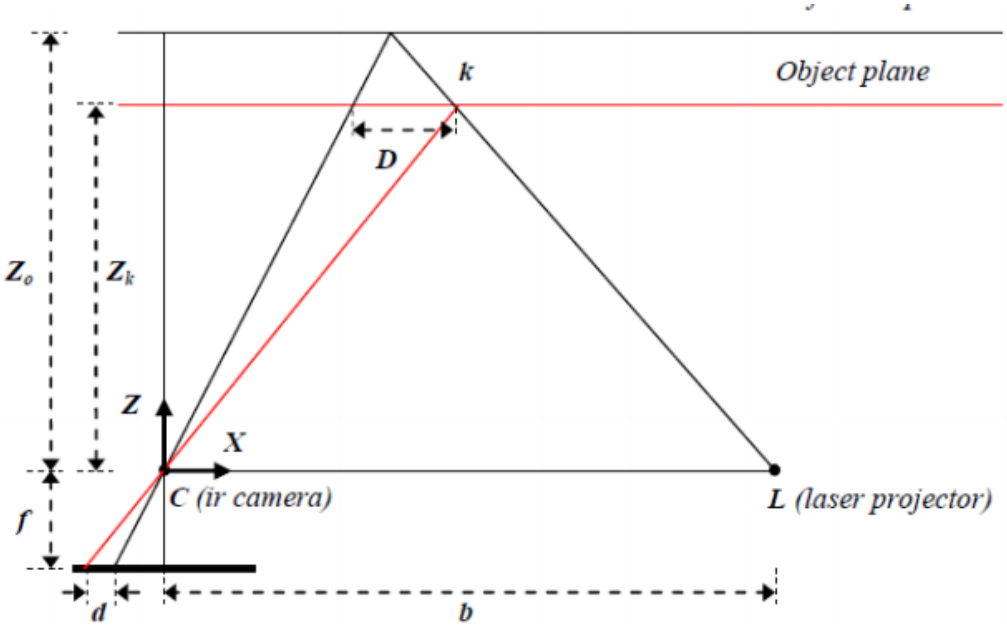
Η σχέση μεταξύ της απόστασης ενός σημείου k από τον αισθητήρα ως προς το επίπεδο αναφοράς και τη μετρηθείσα μετατόπιση d, εμφαίνεται στην εικόνα 2-2. Για τον υπολογισμό των τρισδιάστατων συντεταγμένων των σημείων ενός αντικειμένου, θεωρούμε ένα τρισδιάστατο σύστημα συντεταγμένων με αρχή των αξόνων στο προοπτικό κέντρο της υπέρυθρης κάμερας, όπως στο σχήμα.

Έστω ότι ένα αντικείμενο βρίσκεται στο επίπεδο αναφοράς (reference plane), σε απόσταση Z_0 από τον αισθητήρα, και μια κουκίδα υπέρυθρου φωτός καταγράφεται στο επίπεδο του αντικειμένου (object plane). Αν το αντικείμενο μετατοπιστεί εγγύτερα ή μακρύτερα από τον αισθητήρα, η θέση της κουκίδας στο επίπεδο αντικειμένου θα μετατοπιστεί κατά D, στον άξονα X. Αυτή η απόσταση D αναλογεί στη μετατόπιση d (disparity), η οποία υπολογίζεται από τα όμοια τρίγωνα ως εξής:

$$\frac{D}{b} = \frac{Z_0 - Z_k}{Z_0} \quad (4) \qquad \text{και} \qquad \frac{d}{f} = \frac{D}{Z_k} \quad (5)$$

Όπου:

- Z_k είναι η απόσταση (βάθος) του σημείου k
- b είναι η απόσταση μεταξύ της υπέρυθρης κάμερας και του πομπού υπερύθρων
- f είναι το εστιακή απόσταση της κάμερας
- D είναι η μετατόπιση του σημείου k στο επίπεδο του αντικειμένου
- d είναι η παρατηρούμενη μετατόπιση στο επίπεδο της εικόνας.



Εικόνα 3–4: Σχέση μεταξύ βάθους και μετατόπισης
Πηγή: K. Khoshelham, S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications", 2012

Συνδυάζοντας τις εξισώσεις (4) και (5) και λύνοντας ως προς το Zk προκύπτει:

$$Z_k = \frac{Zo}{1+\frac{Zo}{fb}d} \quad (6)$$

Η εξίσωση (6) αποτελεί το βασικό μαθηματικό μοντέλο για την εξαγωγή του βάθους από την παρατηρούμενη μετατόπιση d, με την προϋπόθεση ότι οι όροι Zo, f και b είναι γνωστοί από τη διαδικασία βαθμονόμησης.

Από το πεδίο της προοπτικής προβολής ισχύουν:

$$x_k = \frac{f}{Z_k}X_k \quad (7) \quad \text{και} \quad y_k = \frac{f}{Z_k}Y_k \quad (8)$$

Επομένως οι συντεταγμένες κάθε σημείου του αντικειμένου μπορούν να υπολογιστούν σύμφωνα με τις εξισώσεις:

$$X_k = \frac{Z_k}{f}(x_k - x_0 + \delta x) \quad (9) \quad \kappa\alpha\iota \quad Y_k = \frac{Z_k}{f}(y_k - y_0 + \delta x) \quad (10)$$

Όπου:

- x_k, y_k είναι οι συντεταγμένες του σημείου στο επίπεδο της εικόνας (image plane)
- x_o, y_o είναι οι συντεταγμένες του κέντρου της εικόνας στο επίπεδο της εικόνας (principal point)
- δx, δy οι διορθώσεις από την παραμόρφωση του φακού. [4] [26]

3.4 Διακριτική ικανότητα και πυκνότητα σημείων

Η ανάλυση της υπέρυθρης κάμερας καθορίζει τη πυκνότητα των σημείων της εικόνας βάθους, στο επίπεδο XY – κάθετα στον άξονα της κάμερας (Z). Δεδομένου ότι κάθε εικόνα βάθους αποτελείται από 640 x 480 px, η πυκνότητα των σημείων είναι αντιστρόφως ανάλογη από την απόσταση του υπό εξέταση αντικειμένου από τον αισθητήρα.

Η διακριτική ικανότητα σε ότι αφορά το βάθος, αναφέρεται στο ελάχιστο βήμα με το οποίο μετράται η απόσταση στον άξονα Z. Το βήμα αυτό καθορίζεται από τον αριθμό των bits ανά pixel που απαιτούνται για την αποθήκευση των τιμών της μετατόπισης (disparity). Το Kinect αποθηκεύει τις τιμές αυτές με ακέραιους αριθμούς 11 bit, όπου 1 bit προορίζεται για τη καταγραφή των σημείων για τα οποία δεν καταγράφηκε μετατόπιση, γνωστά και ως no data. Επομένως, τα 10 εναπομείναντα bits αρκούν για την αποθήκευση 1024 επιπέδων μετατόπισης.

Όπως προαναφέρθηκε κατά την ανάλυση του μαθηματικού μοντέλου εξαγωγής του βάθους, η μετατόπιση είναι αντιστρόφως ανάλογη αυτού. Το ίδιο ισχύει και για τη διακριτική ικανότητα σε σχέση με τα επίπεδα της μετατόπισης, με τη διακριτική ικανότητα του βάθους να ορίζεται απλά ως τη διαφορά που αντιστοιχεί σε δύο διαδοχικά επίπεδα της μετατόπισης.

Ισχύει: $$\Delta_Z = \left(\frac{m}{fb}\right) Z^2 \quad (11)$$

Είναι προφανές πως η διακριτική ικανότητα του βάθους είναι τετραγωνική συνάρτηση του βάθους που αυξάνεται, καθώς μειώνεται η απόσταση από τον αισθητήρα. [4]

3.5 Βαθμονόμηση αισθητήρα

Οι μετρήσεις, είτε προέρχονται από ένα οποιοδήποτε μετρητικό όργανο είτε από το Kinect, χαρακτηρίζονται από δύο ειδών σφάλματα:

- Συστηματικά, που εμφανίζονται στο σύνολο των παρατηρήσεων και μπορούν να μοντελοποιηθούν μαθηματικά.
- Τυχαία η Στοχαστικά, που εμφανίζονται σε όλες σχεδόν τις μετρήσεις και ακολουθούν συνήθως κανονική κατανομή.

Τα τυχαία σφάλματα χαρακτηρίζονται από τη μέση τιμή και τη μεταβλητότητα τους. Η τελευταία μπορεί να μειωθεί λαμβάνοντας υπόψη το ΜΟ πολλαπλών μετρήσεων, ενώ η μέση τιμή μπορεί να ενσωματωθεί στο συστηματικό σφάλμα. [18]

Η βαθμονόμηση (calibration) ως διαδικασία, στοχεύει ευθέως στη μείωση του συστηματικού σφάλματος, ενώ δευτερευόντως επιδρά και στη μέση τιμή του τυχαίου σφάλματος και τη μεταβλητότητα της.

Εν προκειμένω, οι παράμετροι της βαθμονόμησης που συμμετέχουν στο μαθηματικό μοντέλο υπολογισμού των τρισδιάστατων συντεταγμένων από τις πρωτογενείς μετρήσεις, είναι:

- ✓ Η εστιακή απόσταση (f)
- ✓ Οι συντεταγμένες του κέντρου της εικόνας στο επίπεδο αυτής (x_0, y_0)
- ✓ Οι συντελεστές παραμόρφωσης του φακού (δx, δy)
- ✓ Η απόσταση μεταξύ πομπού και δέκτη υπερύθρων (b)
- ✓ Η απόσταση (Z_0) του μοτίβου αναφοράς

Θεωρητικά, από τις παραπάνω παραμέτρους, οι πρώτες τρείς μπορούν να καθοριστούν με μια απλή βαθμονόμηση της υπέρυθρης κάμερας. Εντούτοις, στη πράξη οι παράμετροι βαθμονόμησης της υπέρυθρης κάμερας δεν αντιστοιχούν ευθέως στις τιμές της εικόνας διασποράς. Αυτό συμβαίνει γιατί η εικόνα διασποράς, όπως υπολογίζεται από το Kinect, έχει μέγεθος 640x480 px, ενώ η υπέρυθρη κάμερα έχει ανάλυση 1280x1024 px.

Η χαμηλή χωρητικότητα του διαύλου USB είναι ο λόγος που οι εικόνες της υπέρυθρης κάμερας περικόπτονται στο μέγεθος των 640x480 px, οπότε και αντιστοιχούν στο μέγεθος της εικόνας διασποράς. Επομένως είναι μάλλον βολικότερο, ο υπολογισμός των παραμέτρων της βαθμονόμησης να γίνεται στις περικομμένες εικόνες που επιτρέπουν την ευθεία αντιστοίχιση των εικονοστοιχείων τους με αυτά της έγχρωμης εικόνας.

Σε ότι αφορά τις δύο εναπομείναντες παραμέτρους, την απόσταση μεταξύ πομπού και δέκτη υπερύθρων (baseline) και την απόσταση του μοτίβου αναφοράς, ο καθορισμός τους στο πλαίσιο της βαθμονόμησης είναι μάλλον δυσχερέστερος. Αυτό οφείλεται στο γεγονός πως ο επεξεργαστής του Kinect υπολογίζει το βάθος για κάθε σημείο με διαφορετικό βήμα.

Ως εκ τούτου, η μετατόπιση d στην εξίσωση (3) αντικαθίσταται από τον όρο md’ + n, όπου d’ η κανονικοποιημένη ανισότητα και m,n οι παράμετροι της γραμμικής κανονικοποίησης. Η εξίσωση που προκύπτει εκφράζει τη γραμμική σχέση μεταξύ βάθους και κανονικοποιημένης ανισότητας. [4] [10] [11]

$$Z^{-1} = \frac{m}{fb} d' + \left({Z_0}^{-1} + \frac{n}{fb} \right) \quad (12)$$

Παρατηρούμε ότι για σημεία γνωστού βάθους, οι παράμετροι της γραμμικής σχέσης μπορούν να υπολογιστούν με χρήσης της μεθόδου ελαχίστων τετραγώνων (Μ.Ε.Τ).

3.6 Μοντέλο σφάλματος

Θεωρώντας πως οι παράμετροι της βαθμονόμησης προσδιορίστηκαν με ακρίβεια και αντιμετωπίστηκαν τα συστηματικά σφάλματα, στην ενότητα αυτή αναφερόμαστε στο θεωρητικό μοντέλο του τυχαίου σφάλματος.

Έστω d’ η κανονικοποιημένη μετατόπιση (disparity) - μια τυχαία μεταβλητή με κανονική κατανομή – τότε χρησιμοποιείται η μεταβλητότητα της τιμής της μετατόπισης για τον προσδιορισμό της μεταβλητότητας της τιμής του βάθους ως εξής:

$$\sigma^2{}_Z = \left(\frac{\partial Z}{\partial d}\right)^2 \sigma^2{}_{d'} \quad (13)$$

Έπειτα από απλοποίηση προκύπτει η συνάρτηση της τυπικής απόκλισης του βάθους:

$$\sigma_Z = \left(\frac{m}{fb}\right)^2 Z^2 \sigma_{d'} \quad (14)$$

Όπου σd' και σz η τυπική απόκλιση της μετρηθείσας κανονικοποιημένης μετατόπισης και του υπολογισμένου βάθους αντίστοιχα. Η τελευταία εξίσωση εκφράζει το τυχαίο σφάλμα της τιμής του βάθους ως ανάλογο του τετραγώνου της απόστασης του αντικειμένου από τον αισθητήρα.

Όμοια το τυχαίο σφάλμα των X και Y προκύπτει από τις εξισώσεις [4]:

$$\sigma_X = \left(\frac{mx}{f^2 b}\right)^2 Z^2 \sigma_{d'} \quad (15)$$

$$\sigma_Y = \left(\frac{my}{f^2 b}\right)^2 Z^2 \sigma_{d'} \quad (16)$$

4. ΤΟ ΛΟΓΙΣΜΙΚΟ

Προκειμένου να επιτευχθεί ο στόχος της τρισδιάστατης χαρτογράφησης εσωτερικών χώρων σε πραγματικό χρόνο με χρήση του MS Kinect for Windows (v1), επελέγη το λογισμικό RGBDemo, σε συνδυασμό με ικανό πλήθος οδηγών (drivers), βιβλιοθηκών (libraries), APIs, frameworks, middleware και modules που είναι απαραίτητα για την ορθή εγκατάσταση και λειτουργία του. Ακολουθεί συνοπτική περιγραφή/καταγραφή των βασικών στοιχείων του απαραίτητου λογισμικού, καθώς και τα προτεινόμενα βήματα για την εγκατάστασή του.

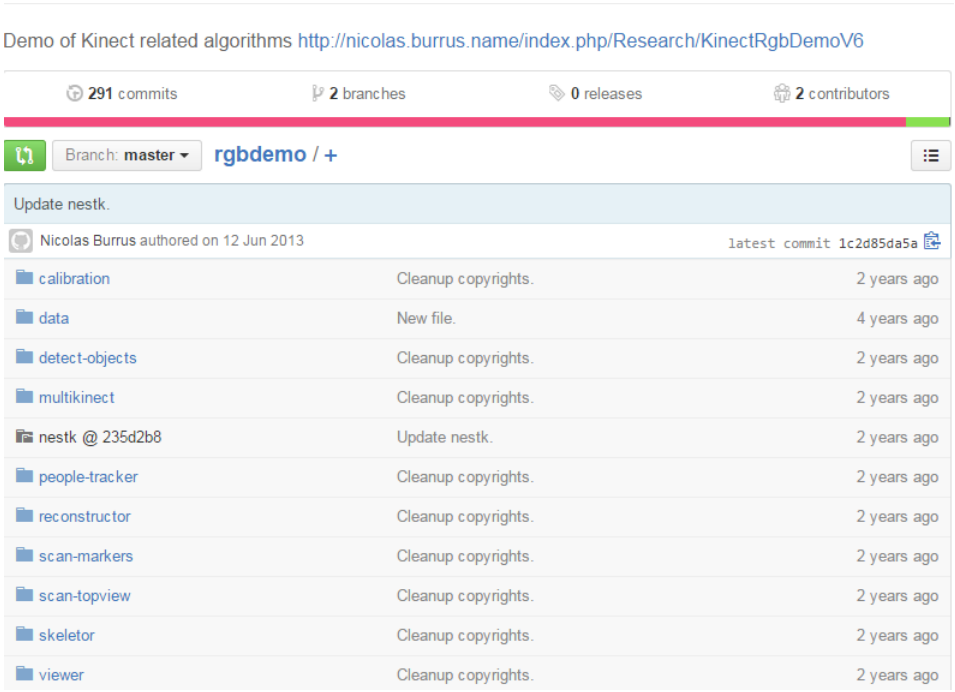
Σημειώνεται πως απαραίτητες προϋποθέσεις για την επιλογή του λογισμικού ήταν να είναι ελεύθερο και ανοιχτού κώδικα στο σύνολο του.

4.1 RGBDemo 0.7.0

Το RGBDemo είναι ένα ελεύθερο λογισμικό/λογισμικό ανοιχτού κώδικά (ΕΛ/ΛΑΚ), γραμμένο ως επί το πλείστον σε C kai C++. Αναπτύχθηκε από τον Γάλλο μεταδιδακτορικό ερευνητή Nicolas Burrus σε συνεργασία με την ερευνητική ομάδα RoboticsLab. Η ονομασία του λογισμικού προδίδει και τη λειτουργικότητα του καθώς τα γράμματα RGB αντιστοιχούν στα τρία (3) βασικά χρώματα Red, Green, Blue μιας έγχρωμης κάμερας, ενώ το D υποδηλώνει τον όρο Depth για τα δεδομένα βάθους που επεξεργάζεται. Αποτελεί ένα απλό πακέτο εργαλείων (toolkit) για την οπτικοποίηση (visualization), βαθμονόμηση (calibration) και επεξεργασία των δεδομένων που συλλέγει η συσκευή Kinect. Διακρίνεται σε δύο (2) κυρίως μέρη: τη βιβλιοθήκη Nestk και τα προγράμματα που τη χρησιμοποιούν. Σημειώνεται πως υποστηρίζει την τεχνολογία γραφικών CUDA και τη χρήση του αλγόριθμου ICP.

Η βασική λειτουργικότητα των προγραμμάτων του πακέτου περιλαμβάνει μεταξύ άλλων:

- ✓ Εγγραφή εικόνας και οπτικοποίηση/αναπαραγωγή της.
- ✓ Βαθμονόμηση της κάμερας για την εξαγωγή νέφους σημείων στο μετρικό χώρο.
- ✓ Εξαγωγή δεδομένων σκελετού (skeleton data) από εικόνες.
- ✓ Ταυτόχρονη υποστήριξη και βαθμονόμηση πολλαπλών συσκευών Kinect.
- ✓ Τρισδιάστατη ανακατασκευή του χώρου σε πραγματικό χρόνο.
- ✓ Εξαγωγή τρισδιάστατου μοντέλου αντικειμένων.
- ✓ Ανίχνευση κίνησης και υπολογισμός θέσης ατόμων.



Εικόνα 4–1: Το RDBDemo στο αποθετήριο GitHub
Πηγή: www.github.com

Σημειώνεται πως το λογισμικό RGBDemo δεν αναπτύσσεται πλέον ενεργά. [19]

4.1.1 Βιβλιοθήκες

Nestk

Όπως προαναφέρθηκε, η Nestk είναι η βιβλιοθήκη που βρίσκεται στον πυρήνα του λογισμικού RGBDemo και είναι σχεδιασμένη να προσφέρει εύκολη και γρήγορη πρόσβαση στα δεδομένα που συλλέγει το Kinect. Η Nestk βασίζεται στις βιβλιοθήκες OpenCV (Open Computer Vision) και Qt (Q toolkit) για τα γραφικά μέρη της. Ένα επίσης μέρος της βασίζεται στην βιβλιοθήκη PCL (Point Cloud Library), ενώ περιέχει, μεταξύ άλλων, και τη βιβλιοθήκη libfreenect που υποστηρίζει η κοινότητα OpenKinect. [19]

OpenCV

Η Open Computer Vision είναι μια βιβλιοθήκη λογισμικού ανοιχτού κώδικα για εφαρμογές υπολογιστικής όρασης και μηχανικής μάθησης. Παρέχει μια κοινή υποδομή προκειμένου να επιταχύνει την υιοθέτηση και παραγωγή σχετικών εμπορικών προϊόντων. Η βιβλιοθήκη περιλαμβάνει περισσότερους από 2500 αλγόριθμους, τόσο κλασικούς όσο και στη στάθμη της τεχνικής (state of the art), για

την αναγνώριση προσώπων, εντοπισμό αντικειμένων, ανίχνευση κίνησης, εξαγωγή τρισδιάστατων μοντέλων και νεφών σημείων κλπ. Υποστηρίζει όλα τα λειτουργικά συστήματα (βλ. Windows, Linux, Android, Mac OS) και διεπαφές (βλ. C++, C, Python, Java και MATLAB), ενώ χρησιμοποιείται και από τις μεγαλύτερες εταιρείες του χώρου της πληροφορικής και όχι μόνο (βλ. Google, Yahoo, Microsoft, Intel, IBM, Sony και Honda). [20]

PCL

Η Point Cloud Library είναι μια βιβλιοθήκη μεγάλης κλίμακας για την επεξεργασία δισδιάστατων (2D) και τρισδιάστατων (3D) εικόνων καθώς και νεφών σημείων. Περιέχει, μεταξύ άλλων, μεγάλο πλήθος αλγορίθμων για την εξαγωγή στοιχείων, αναδόμηση, κατάτμηση, εφαρμογή φίλτρων και απαλοιφή θορύβου. Η χρήση της είναι ελεύθερη τόσο για ερευνητικούς σκοπούς, όσο και για εμπορική χρήση. Όπως και η OpenCV, υποστηρίζει το σύνολο των πλατφορμών και διεπαφών και τυγχάνει ευρείας αποδοχής και χρήσης. [21]

4.1.2 Frameworks

Qt

Το Q toolkit είναι ένα ελεύθερο πακέτο εργαλείων για την ανάπτυξη εφαρμογών για κάθε πλατφόρμα και διεπαφή. Αναπτύσσεται από την εταιρεία Qt Company και τη θυγατρική της Digia. Χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών με γραφικά περιβάλλοντα χρήστη (graphical user interfaces - GUIs). Βασικό του συστατικό αποτελεί μια ομάδα βιβλιοθηκών γραμμένη σε C++. Προωθεί τη διαλειτουργικότητα μεταξύ συστημάτων και τυγχάνει μεγάλης αποδοχής καθώς συναντάται σε 70 κλάδους της βιομηχανίας. [22]

OpenNI

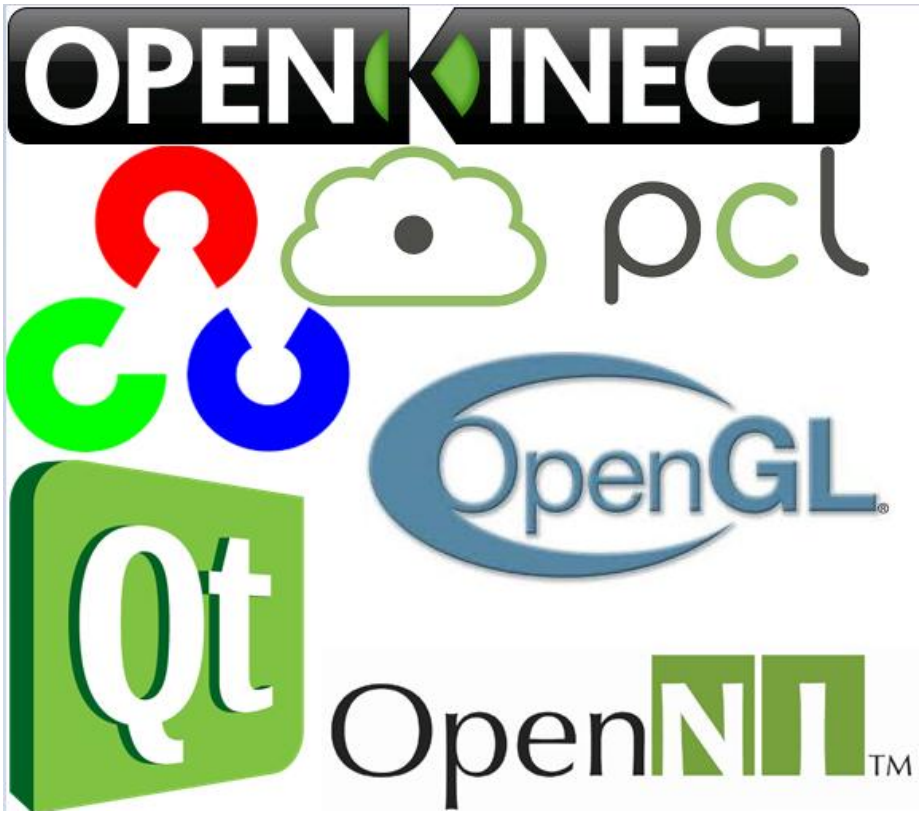
Η Open Natural Interaction είναι μια μη-κερδοσκοπική κοινότητα με ρίζες στη βιομηχανία, η οποία παράγει λογισμικό ανοικτού κώδικα για τη βελτίωση της διαλειτουργικότητας των φυσικών διεπαφών χρήστη (natural user interfaces), αυτών δηλαδή που δεν απαιτούν τη χρήση περιφερειακών όπως πληκτρολόγιο ή/και ποντίκι. Διαθέτει την πλειοψηφία των βιβλιοθηκών για πρόσβαση στα δεδομένα του Kinect, είτε πρόκειται για δεδομένα σκελετού (skeleton data) και τις χειρονομίες (gestures) με χρήση του middleware NITE, είτε για πρωτόλεια δεδομένα με χρήση του hardware driver SensorKinect. Το βασικό ιδρυτικό μέλος της κοινότητας αυτής ήταν η

ισραηλινή εταιρεία PrimeSense που σε συνεργασία με τη Microsoft ανέπτυξε το Kinect και η οποία εν συνεχεία, τον Νοέμβριο του 2013, εξαγοράστηκε από την εταιρεία Apple έναντι 350 εκατομμυρίων δολαρίων. Το αποτέλεσμα της εξαγοράς αυτής ήταν η παύση της ενεργής υποστήριξης της κοινότητας OpenNI. [23]

4.1.3 API

OpenGL

Η Open Graphics Library αποτελεί ένα περιβάλλον ανάπτυξης εφαρμογών (API) για την απόδοση δισδιάστατων (2D) και τρισδιάστατων (3D) διανυσματικών γραφικών, σε όλες τις γλώσσες προγραμματισμού και για το σύνολο των λειτουργικών συστημάτων. Χρησιμοποιείται συνήθως για την αλληλεπίδραση με μια μονάδα επεξεργασίας γραφικών (GPU), προκειμένου να επιταχυνθεί η απόδοση των γραφικών μέσω αυτής. Περιλαμβάνει ένα πακέτο εντολών και μεθόδων για την δημιουργία γραφικών στοιχείων, επαναλήψιμων και μη. Η ανάπτυξη της OpenGL ξεκίνησε το 1991 από την Silicon Graphics και διετέθη στην αγορά ένα χρόνο αργότερα. Τυγχάνει μεγάλης αποδοχής από τη βιομηχανία στους τομείς της ψηφιακής σχεδίασης (CAD), εικονικής πραγματικότητας (VR), οπτικοποίησης δεδομένων, εξομοιωτές και ηλεκτρονικά παιχνίδια. [24]



Εικόνα 4–2: Τα λογότυπα των επιμέρους λογισμικών.
Πηγή: Ίδια εικόνα

4.2 Εγκατάσταση

Η διαδικασία εγκατάστασης του λογισμικού RGBDemo ποικίλει. Εξαρτάται από το εκάστοτε λειτουργικό σύστημα και τις απαιτήσεις του χρήστη ως προς τη λειτουργικότητα. Εν προκειμένω επελέγη η εγκατάσταση σε υπολογιστή με λειτουργικό σύστημα Microsoft Windows 7 32 Bit Home Premium Service Pack 1, με χρήση Microsoft Visual Studio 2012. Ακολουθεί μια αναλυτική παράθεση των επιμέρους βημάτων που απαιτήθηκαν.

Σημειώνεται πως οι εκδόσεις των επιμέρους βιβλιοθηκών, οδηγών, frameworks, middlewares και APIs που χρησιμοποιούνται παρακάτω καθώς και η σειρά εγκατάστασης τους αποτελούν συνδυασμό, ο οποίος επελέγη κατόπιν διαδοχικών δοκιμών, ως ο πλέον λειτουργικός. Μολονότι ο συνδυασμός αυτός δεν είναι μοναδικός, το σύνολο των εναλλακτικών δεν στέφθηκε με επιτυχία.

4.2.1 Διαδικασία

Η επιτυχής εγκατάσταση του RGBDemo v 0.7.0 με υποστήριξη OpenCV, OpenNI και PCL, απαιτεί την κατά σειρά εκτέλεση των παρακάτω βημάτων:

- Λήψη του Zigfu Installer και εγκατάστασή του.
- Έλεγχος της ορθής λειτουργίας των RGBDemo Binaries.
- Αν λειτουργούν σωστά, εκτέλεση του επόμενου βήματος. Στην αντίθετη περίπτωση, απαιτείται έλεγχος του Device Manager και μόνιμη απεγκατάσταση των υπαρχόντων οδηγών (drivers) του Kinect. Έπειτα επανασύνδεση του Kinect και έλεγχος του Device Manager για τις εγγραφές Kinect Camera, Kinect Audio και Kinect Motor. Επανέλεγχος της λειτουργίας των RGBDemo Binaries.
- Λήψη των Qt Source Files.
- Αποσυμπίεση των Qt Source Files στο: “C:\DEV\Qt\”
- Εκκίνηση της γραμμής εντολών του Visual Studio: Start → All Programs → Microsoft Visual Studio 2012 → Visual Studio Tools → Visual Studio Command Prompt.
- Μετάβαση στο: “C:\DEV\Qt\4.8.2”
- Εισαγωγή της εντολής: “configure -release -no-webkit -no-phonon no-phonon-backend -no-script -no-scripttools -no-qt3support -no-multimedia -no-ltcg”
- Κατόπιν εισαγωγή: “nmake” και επιβεβαίωση με enter.

- Μόλις ολοκληρωθεί η παραπάνω διαδικασία, εισάγεται η εξής εντολή: configure -debug -no-webkit -no-phonon no-phonon-backend -no-script -no-scripttools -no-qt3support -no-multimedia -no-ltcg”
- Εκ νέου εισαγωγή: “nmake” και επιβεβαίωση με enter.
- Μόλις ολοκληρωθεί και αυτή η διαδικασία, κλείνει η γραμμή εντολών.
- Προσθήκη του: “C:\DEV\Qt\4.8.2\bin” στις μεταβλητές συστήματος.
- Προσθήκη του: “C:\DEV\Qt\4.8.2\bin” ως “QTDIR”στις μεταβλητές περιβάλλοντος.
- Λήψη της OpenCV 2.3.1
- Αποσυμπίεση της OpenCV 2.3.1 στο: “C:\DEV\ OpenCV\2.3.1\”
- Λήψη και εγκατάσταση του CMake.
- Εκκίνηση του CMake GUI.
- Στο πεδίο του εργαλείου όπου γράφει: “Where is the source code”, εισάγετε: “C:\DEV\OpenCV\2.3.1\”
- Στο πεδίο του εργαλείου όπου γράφει: “Where to build the binaries”, εισάγετε: “C:\DEV\OpenCV\2.3.1\build”
- Επιλογή: “Configure”
- Προσδιορισμός του compiler: “Visual Studio 2012”, επιλογή: “Use default native compiler” και αναμονή για την ένδειξη: “Configuring done”
- Επιλογή: “Generate”, αναμονή για την ένδειξη: “Generating done” και κλείσιμο του CMake.
- Μετάβαση στο: “C:\DEV\OpenCV\2.3.1\build” και άνοιγμα του αρχείου: “OpenCV.sln” με το Microsoft Visual Studio 2012.
- Εντοπισμός του Combobox που περιέχει την ένδειξη: “Debug”
- Επιλογή της εντολής “Compile” και αναμονή για την ολοκλήρωση της διαδικασίας.
- Αλλαγή της ένδειξης: “Debug” σε: “Release”
- Επιλογή εκ νέου της εντολής: “Compile” και αναμονή για την ολοκλήρωση της διαδικασίας, οπότε και κλείνει το Visual Studio.
- Προσθήκη των: “C:\DEV\OpenCV\2.3.1\build\bin\Debug” και “C:\DEV\OpenCV\2.3.1\build\bin\Release” στις μεταβλητές συστήματος.
- Λήψη και εγκατάσταση της PCL.

- Λήψη του GLUT και αποσυμπίεση των αρχείων: "*.dll" στο: "C:\Windows\"
- Επανεκκίνηση του Η/Υ.
- Λήψη των RGBDemo v 0.7.0 Sources.
- Εκκίνηση του: CMake GUI
- Στο πεδίο του εργαλείου όπου γράφει: "Where is the source code", εισάγετε: "C:\DEV\RGBDemo 0.7\"
- Στο πεδίο του εργαλείου όπου γράφει: "Where to build the binaries", εισάγετε: "C:\DEV\RGBDemo 0.7\build"
- Επιλογή: "Configure"
- Έλεγχος για τυχόντα λάθη στις μεταβλητές και διόρθωσή τους.
- Εκ νέου επιλογή: "Configure"
- Αν δεν εμφανίστηκαν λάθη, επιλέγεται: "Generate"
- Μετάβαση στο: "C:\DEV\RGBDemo 0.7\build" και άνοιγμα του: "RGBDemo.sln"
- Εντοπισμός του Combobox που περιέχει την ένδειξη: "Debug"
- Επιλογή της εντολής "Compile" και αναμονή για την ολοκλήρωση της διαδικασίας.
- Αλλαγή της ένδειξης: "Debug" σε: "Release"
- Επιλογή εκ νέου της εντολής: "Compile" και αναμονή για την ολοκλήρωση της διαδικασίας, οπότε και κλείνει το Visual Studio.
- Ορισμός του επιθυμητού project ως start-project.
- Επιλογή Compile και αναμονή για την ολοκλήρωση της διαδικασίας.

5. ΝΕΦΗ ΣΗΜΕΙΩΝ

Το νέφος σημείων (point cloud) είναι μια συλλογή από σημεία στον τρισδιάστατο χώρο. Ο όρος νέφος οφείλεται στην αίσθηση πως τα σημεία αιωρούνται στο χώρο, λόγω της έλλειψης μεταξύ τους σύνδεσης. Στην απλούστερη μορφή τους, τα νέφη σημείων περιέχουν μόνο χωρική πληροφορία (X, Y, Z), εντούτοις είναι δυνατό να συμπεριλάβουν επιπλέον πληροφορίες, όπως π.χ. χρώμα και προσανατολισμό.

Χρησιμοποιούνται από τη βιομηχανία σε πλήθος εφαρμογών. Ενδεικτικά, αναφέρονται εφαρμογές οπτικοποίησης, τρισδιάστατης ανακατασκευής, ποιοτικού ελέγχου, πλοήγησης στον χώρο και σχεδιαστικές εφαρμογές. Εντούτοις, προτού παραχθεί ένα νέφος σημείων, οφείλει κανείς να γνωρίζει πώς να διαχειρίζεται, επεξεργάζεται και αναπαριστά τη τρισδιάστατη πληροφορία.

Τα νέφη σημείων αποτελούν τον κατεξοχήν τρόπο αναπαράστασης της πληροφορίας που συλλέγει το Kinect, καθώς κάθε εικονοστοιχείο στην παραγόμενη εικόνα βάθους μπορεί να αναχθεί σε ένα σημείο στο χώρο. [13]

5.1 Βαθμονόμηση με το RGBDemo

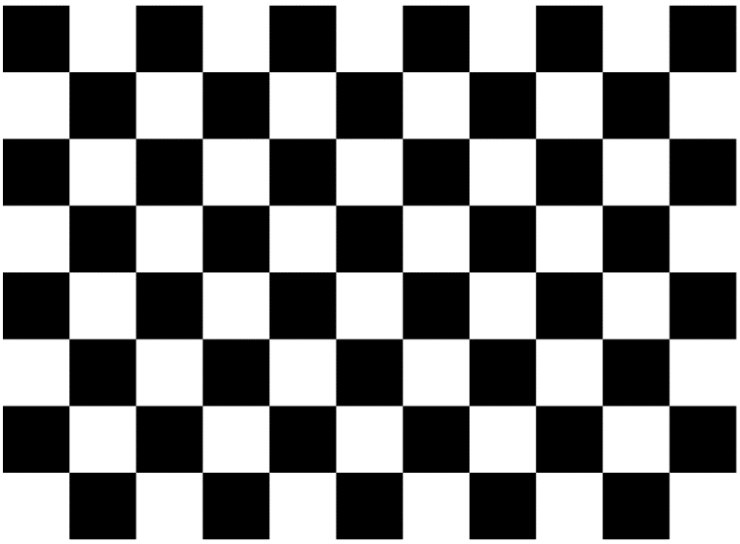
Προτού προβεί κανείς σε οποιαδήποτε μέτρηση, συμπεριλαμβανομένης της δημιουργίας νέφους σημείων, οφείλει να προβεί σε βαθμονόμηση του αισθητήρα για λόγους που αναλύθηκαν σε προηγούμενο κεφάλαιο.

Η βαθμονόμηση είναι ουσιαστικά μια διαδικασία αντιστοίχισης ενός γνωστού στόχου σε μια ομάδα πιθανών εικόνων και ο υπολογισμός των σχετικών διαφορών. Σε κάθε περίπτωση, είναι απαραίτητη διαδικασία για τον ακριβή εντοπισμό αντικειμένων στο χώρο.

Μολονότι το Kinect προσφέρεται στο καταναλωτικό κοινό βαθμονομημένο (factory set), συστήνεται οι πιο εξειδικευμένοι χρήστες να βαθμονομήσουν εκ νέου τη συσκευή και το λογισμικό RGBDemo παρέχει τη δυνατότητα αυτή.

Πρώτα ο χρήστης οφείλει να κατασκευάσει ένα στόχο της μορφής κανάβου πλάκας ζατρικίου (σκακιέρας), μεγέθους A4. Το σχετικό μοτίβο είναι διαθέσιμο στο φάκελο της εγκατάστασης και δη στη θέση /data, υπό την ονομασία chessboard_a4.pdf.

Ο κάναβος του προτεινόμενου αρχείου έχει βήμα 23mm. Σε περίπτωση που ο χρήστης επιλέξει να δημιουργήσει δικό του στόχο, τότε οφείλει σε κάθε περίπτωση να χρησιμοποιήσει χρώματα με μεγάλη αντίθεση και να γνωρίζει το ακριβές βήμα, καθώς θα χρειαστεί στη συνέχεια. [5] [15]



Εικόνα 5–1: Κάναβος τύπου πλάκας ζατρικίου
Πηγή: www.rgbdemo.org

Έπειτα η διαδικασία της βαθμονόμησης έχει ως εξής:

- ✓ Εκκίνηση του rgbd-viewer από τη θέση /build/bin.
- ✓ Επιλογή του IR/RGB Mode από το μενού Capture.
- ✓ Λήψη 30 τουλάχιστον εικόνων από σχετικά μικρή απόσταση, σε διαφορετικές γωνίες και πάντα με τα άκρα του στόχου εντός καρέ.
- ✓ Έξοδος από τον rgbd-viewer και εκτέλεση σε περιβάλλον command line της εντολής build/bin/calibrate_kinect_ir σε συνδυασμό με τις παραμέτρους:
  - --pattern-size number_in_meters
  - --input directory_where_your_images_are
- ✓ Αφού ολοκληρωθούν οι διεργασίες, γίνεται έλεγχος του τελικού σφάλματος, όπου τιμές που είναι μικρότερες του 1 pixel είναι αποδεκτές.

Εικόνα 5–2: Ένα παράδειγμα θέσης λήψης εικόνας κατά τη βαθμονόμηση
Πηγή: J. Kramer, N.Burrus, D. Herrera C., F. Echtler, M. Parker, "Hacking the Kinect", 2012

Η παραπάνω διαδικασία βαθμονόμησης παράγει το αρχείο kinect_calibration.yml, το οποίο περιέχει χρήσιμες πληροφορίες. Μια ενδεικτική μορφή του περιεχομένου του, όπως παρήχθη για της ανάγκες της παρούσης εργασίας, είναι η παρακάτω:

```
%YAML:1.0
rgb_intrinsics: !!opencv-matrix
    rows: 3
    cols: 3
    dt: d
    data: [ 5.1949264458725597e+02, 0., 3.2538790034152803e+02, 0.,
        5.1689335524191112e+02, 2.5164152041182755e+02, 0., 0., 1. ]
rgb_distortion: !!opencv-matrix
    rows: 1
    cols: 5
    dt: d
    data: [ 2.4242340694285675e-01, -8.4427732173142570e-01,
        -1.9170692121966231e-03, 5.5158456701865192e-03, 9.7454412755454362e-01 ]
depth_intrinsics: !!opencv-matrix
    rows: 3
    cols: 3
    dt: d
    data: [ 5.8189378818359510e+02, 0., 3.1145158291339540e+02, 0.,
        5.8026607093655313e+02, 2.5011989404977429e+02, 0., 0., 1. ]
depth_distortion: !!opencv-matrix
    rows: 1
    cols: 5
    dt: d
```

```
    data: [ -2.398766091027847e-01, 1.599626095975791e+00,
            -8.426185476727272e-04, 1.108454678946856e-03, -4.101822656557877e+00 ]
R: !!opencv-matrix
    rows: 3
    cols: 3
    dt: d
    data: [ 9.998910620782972e-01, -1.933773241880584e-03, 1.463299343892394e-02,
            1.953951487267514e-03, 9.999715971478453e-01, -1.364784213407723e-03,
            -1.463031269385618e-02, 1.393227695945112e-03, 9.998920006015985e-01 ]
T: !!opencv-matrix
    rows: 3
    cols: 1
    dt: d
    data: [ 1.981723807543234e-02, -1.916979935401025e-03, -2.745059180211685e-03 ]
rgb_size: !!opencv-matrix
    rows: 1
    cols: 2
    dt: i
    data: [ 640, 480 ]
raw_rgb_size: !!opencv-matrix
    rows: 1
    cols: 2
    dt: i
    data: [ 640, 480 ]
depth_size: !!opencv-matrix
    rows: 1
    cols: 2
    dt: i
    data: [ 640, 480 ]
raw_depth_size: !!opencv-matrix
    rows: 1
    cols: 2
    dt: i
    data: [ 640, 480 ]
depth_base_and_offset: !!opencv-matrix
    rows: 1
    cols: 2
    dt: f
    data: [ 1.33541569e-01, 1.55009338e+03 ]
```

Η βαθμονόμηση γίνεται σύμφωνα με το μοντέλο κάμερας μικρής οπής που αναλύθηκε διεξοδικά σε προηγούμενο κεφάλαιο. Ως εκ τούτου, οι τιμές των πεδίων rgb_intrinsics και depth_intrinsics αναφέρονται στις παραμέτρους του εσωτερικού προσανατολισμού της κάμερας σε τιμές pixel. Με μια πιο προσεκτική ανάγνωση, αντιλαμβάνεται κανείς πως πρόκειται για πίνακες μεγέθους 3x3 της μορφής:

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

Όπου c_x, c_y οι συντεταγμένες του κέντρου της εικόνας στο επίπεδο της εικόνας (principal point) και f_x, f_y οι εστιακές αποστάσεις

Οι παραμορφώσεις του φακού, ακτινική (k) και εφαπτομενική (P), προστίθενται στο μοντέλο με τη μορφή διανύσματος, ήτοι:

$$[k_1 \quad k_2 \quad P_1 \quad P_2 \quad k_3]$$

Οι τιμές των k_1, k_2, k_3, P_1, P_2 εντοπίζονται στα πεδία rgb_distortion και depth_distortion.

Τέλος, οι τιμές της στροφής και της μετάθεσης εντοπίζονται στα πεδία R και T αντίστοιχα και αθροίζονται με τη μορφή ενός πίνακα R/T, ως εξής [5]:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

5.2 Δημιουργία νέφους σημείων με χρήση της PCL

Ο βασική δομή δεδομένων που χρησιμοποιεί η PCL είναι ένα απλό διάνυσμα (vector) σημείων (βλ. pcl::PointCloud<PointT>), που η βιβλιοθήκη του παρέχει επιπλέον δυνατότητες όπως file I/O (άνοιγμα και κλείσιμο ροής, ανάγνωση, εγγραφή και προσάρτηση), οπτικοποίηση και normal estimation.

Η κλάση (class) PointCloud δημιουργήθηκε με κριτήριο την ευελιξία αποθήκευσης του επιθυμητού τύπου σημείου. Ο απλούστερος τύπος σημείου είναι ο pcl::PointXYZ, που ουσιαστικά αποθηκεύει τις συντεταγμένες ενός σημείου στο χώρο, και μόνον. Ο τύπος pcl::PointXYZRGB προσθέτει τις τιμές RGB για την απόδοση του χρώματος του σημείου κ.ο.κ. Η βιβλιοθήκη περιλαμβάνει μεγάλο πλήθος κλάσεων για την ικανοποίηση των συνηθέστερων αναγκών των χρηστών και παράλληλα παρέχει τη δυνατότητα δημιουργίας νέων κλάσεων για την κάλυψη ιδιαιτέρων αναγκών.

Όπως συνηθίζεται για τους περισσότερους τύπους μεταβλητών, αν ο αριθμός των σημείων είναι γνωστός, το διάνυσμα δύναται να αρχικοποιηθεί και η μνήμη να δεσμευτεί εκ των προτέρων. Εναλλακτικά, το μέγεθος του μπορεί να αυξάνεται δυναμικά με χρήση π.χ. της μεθόδου push_back, με δυναμική δέσμευση της απαιτούμενης μνήμης.

Οι περισσότερες συσκευές, συμπεριλαμβανομένου του Kinect, αποθηκεύουν τη τρισδιάστατη πληροφορία σε δισδιάστατους πίνακες. Η ανάθεση τιμών μπορεί λοιπόν εκτός από γραμμικά, να γίνει και σε δύο (2) διαστάσεις. [5]

Η δημιουργία ενός απλού νέφους σημείων, σύμφωνα με τα παραπάνω, έχει ως εξής [21]:

```
pcl::PointCloud<pcl::PointXYZ> cloud;

cloud.width = 20;  //Dimensions must be initialized to use 2-D indexing
cloud.height = 20;
cloud.resize(cloud.width*cloud.height);
for(int v = 0; v < cloud.height; v++) //2-D indexing
    for(int ui = 0; u < cloud.width; u++) {
        cloud(u,v).x = u;
        cloud(u,v).y = v;
        cloud(u,v).z = rand() / (float) RAND_MAX; //Assign a random value just for testing
    }
for(unsigned int i = 0; i < cloud.size(); i++) //Linear indexing accesses the same points
    do_something_with_point(cloud[i]);
```

5.3 Παραγωγή νέφους σημείων από εικόνα βάθους

Τα δεδομένα βάθους του Kinect συλλέγονται με τη μορφή δισδιάστατης εικόνας έντασης (2D intensity image). Η ένταση είναι μια μέτρηση που συλλέγεται για κάθε σημείο και αναφέρεται στην ισχύ της υπέρυθρης ακτινοβολίας στο σημείο αυτό, η οποία, εν πολλοίς, εξαρτάται από την ανακλαστικό ήτα αυτού και το μήκος κύματος της ακτινοβολίας. Η εικόνα έντασης είναι ένας ορθογώνιος πίνακας ακεραίων, αποθηκευμένος στη προσωρινή μνήμη της συσκευής.

Η ερμηνεία του πίνακα εξαρτάται, σε μεγάλο βαθμό, από τα χαρακτηριστικά της κάμερας και δη, τους παράγοντες του εσωτερικού και εξωτερικού προσανατολισμού. Η ορθή βαθμονόμηση της συσκευής, επιτρέπει την αντιστοίχιση του βάθους με τις τιμές της διασποράς (disparity units). Η μετατροπή γίνεται με χρήση της εξίσωσης:

$$z = 1/(value * dc1 + dc2) \quad (17)$$

Όπου dc1 και dc2 οι παράμετροι του προσανατολισμού που καθορίζουν τη κλίμακα (scale) και τη μετατόπιση (offset) αντίστοιχα. Αφού προσδιοριστεί η τιμή του Z, θα χρησιμοποιηθεί το μοντέλο κάμερας μικρής οπής (pinhole camera model) για τον προσδιορισμό των τιμών X,Y. [5]

Η μετατροπή μιας εικόνας βάθους σε νέφος σημείων γίνεται ως εξής [21]:

```
pcl::PointCloud<pcl::PointXYZ> cloud;
cv::Mat1s depth_image;

const float dc1= -0.0030711021;
const float dc2=3.3309495157;
float fx_d,fy_d,px_d,py_d; //From calibration

cloud.width = depth_image.cols; //Dimensions must be initialized to use 2-D indexing
cloud.height = depth_image.rows;
cloud.resize(cloud.width*cloud.height);
for(int v = 0; v < depth_image.rows; v++) //2-D indexing
    for(int ui = 0; u < depth_image.cols; u++) {
        float z = 1.0f / (depth_image(v,u)*dc1+dc2);

        cloud(u,v).x = z*(u-px_d)/fx_d;
        cloud(u,v).y = z*(v-py_d)/fy_d;
        cloud(u,v).z = z;
    }
```

5.4 Απόδοση χρώματος στο νέφος σημείων

Προκειμένου να αποδοθεί χρώμα στα τρισδιάστατα σημεία του νέφους, πρέπει να γίνει ο μετασχηματισμός τους στο σύστημα συντεταγμένων της έγχρωμης εικόνας. Επειδή οι κάμερες έχουν διαφορετικά χαρακτηριστικά (βλ. εστιακή απόσταση κλπ.), ο μετασχηματισμός αυτός θα γίνει σε δύο (2) βήματα. Αρχικά, θα γίνει των σημείων στο επίπεδο της ίριδας με χρήση ενός μετασχηματισμού στερεού σώματος (rigid transformation) - έξι (6) βαθμοί ελευθερίας (μετάθεση και στροφή).

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} X'_O \\ Y'_O \\ Z'_O \end{bmatrix} \quad \text{είτε} \quad \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X + X_O \\ Y + Y_O \\ Z + Z_O \end{bmatrix}$$

Εικόνα 5–3: Ο μετασχηματισμός στερεού σώματος
Πηγή: Γ. Καρράς, "Γραμμικοί Μετασχηματισμοί Συντεταγμένων στη Φωτογραμμετρία", 1998

Εν συνεχεία, θα γίνει προβολή των σημείων στο επίπεδο της εικόνας, σύμφωνα με το μοντέλο κάμερας μικρής οπής. Το μοντέλο επεκτείνεται για να συμπεριλάβει τις παραμορφώσεις, ακτινική και εφαπτομενική [5].

Η απόδοση χρώματος στο νέφος σημείων γίνεται ως εξής [21]:

```
pcl::PointCloud<pcl::PointXYZRGB> cloud;
cv::Mat1s depth_image;
cv::Mat3ub color_image;
const float dc1= -0.0030711016;
const float dc2=3.3309495161;
float fx_d,fy_d,px_d,py_d; //From calibration
float fx_c,fy_c,px_c,py_c; //From calibration
cv::Matx33f R; //From calibration
cv::Matx31f T; // From calibration

cloud.width = depth_image.cols; //Dimensions must be initialized to use 2-D indexing
cloud.height = depth_image.rows;
cloud.resize(cloud.width*cloud.height);
for(int v=0; v< depth_image.rows; v++) //2-D indexing
    for(int ui=0; u< depth_image.cols; u++) {
        cv::Matx31f &Xd = *(cv::Matx31f*)&cloud(u,v).x; //Access the point as a cv::Matx

        //3-D position
        float z = 1.0f / (depth_image(v,u)*dc1+dc2);
        Xd(0) = z*(u-px_d)/fx_d;
        Xd(1) = z*(v-py_d)/fy_d;
        Xd(2) = z;

        //Project to color image
        cv::Matx31f Xc = R*Xd + T; //Rigid transformation
        int uc,vc;
        uc = (int)(Xc(0)*fx_rgb/Xc(2) + px_rgb); //Pinhole projection
        vc = (int)(Xc(1)*fy_rgb/Xc(2) + py_rgb); //Truncate to nearest neighbor

        //Copy color
        cloud(u,v).r = color_image(vc,uc)[0];
        cloud(u,v).g = color_image(vc,uc)[1];
        cloud(u,v).b = color_image(vc,uc)[2];
    }
```

5.5 Οπτικοποίηση του νέφους σημείων

Την κατασκευή του έγχρωμου νέφους σημείων ακολουθεί η οπτικοποίηση του που θα επιτρέψει μεταξύ άλλων και έναν πρόχειρο εποπτικό έλεγχο. Το νέφος σημείων μπορεί να οπτικοποιηθεί είτε με χρήση της PCL, είτε με χρήση της OpenGL. Εν προκειμένω επελέγη η OpenGL, καθώς παρέχει αυξημένη ελευθερία κατά την υλοποίηση. Το πακέτο εργαλείων της OpenGL (OpenGL Utilities Toolkit – GLUT) επιτρέπει την εύκολη δημιουργία παραθύρων, την απόδοση των γραφικών στοιχείων και τον χειρισμό γεγονότων (event handling).

Μετά την αρχικοποίηση της OpenGL, η απόδοση των σημείων μπορεί να γίνει είτε κατά μόνας - που δεν συστήνεται καθώς το μεγάλο πλήθος σημείων του νέφους έχει μεγάλο υπολογιστικό κόστος – είτε με χρήση πινάκων (arrays) [13].

Η αρχικοποίηση της OpenGL γίνεται ως εξής [21]:

```
int main(int argc, char **argv) {
    glutInit(&argc, argv); //Initialize GLUT library

    //Create glut window
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(0, 0);
    int main_window = glutCreateWindow("Cloud viewer");

    //Glut callbacks
    glutDisplayFunc(do_glutDisplay); //These functions are defined by you
    glutIdleFunc(do_glutIdle); // to handle the different events
    glutReshapeFunc(do_glutReshape);
    glutKeyboardFunc(do_glutKeyboard);
    glutMotionFunc(do_glutMotion);
    glutMouseFunc(do_glutMouse);

    //Default settings
    glMatrixMode(GL_TEXTURE);
    glLoadIdentity();
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glEnable(GL_DEPTH_TEST);
    do_glutReshape(Width, Height);
    glutMainLoop(); //Main event loop, will never return
}

// do_glutReshape: called when the window is resized
void do_glutReshape(int Width, int Height) {
    glViewport(0,0,Width,Height); //Sets the viewport matrix to match the window size
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60, 4/3., 0.3, 200); //Sets up a regular camera in the projection matrix
    glMatrixMode(GL_MODELVIEW);
}

void do_glutDisplay() {
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    //Modelview matrix
    glLoadIdentity();
    glScalef(zoom,zoom,1); //Matrix transformations to alter where the object is rendered
    glTranslatef(0,0,-3.5);
    glRotatef(rotangles[0], 1,0,0);
    glRotatef(rotangles[1], 0,1,0);
    glTranslatef(0,0,1.5);

    //Show point cloud here
    glutSwapBuffers(); //Show the newly rendered buffer
}
```

Εν προκειμένω επελέγη η χρήση πινάκων και η οπτικοποίηση του νέφους γίνεται ως εξής:

```
//Set up arrays
unsigned int indices[] = {0,1,2,10,11,12}; //Render only these 6 points
int index_count = 6;
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, sizeof(point_cloud[0]), &point_cloud[0].x);
glEnableClientState(GL_COLOR_ARRAY);
glColorPointer(3, GL_FLOAT, sizeof(point_cloud[0]), &point_cloud[0].r);

//Render
glPointSize(1.0f);
glDrawElements(GL_POINTS, index_count, GL_UNSIGNED_INT, indices);
```

5.6 Αγκίστρωση (Registration)

Η απλή επίθεση της εικόνας βάθους επί της έγχρωμης εικόνας είναι ένας μάλλον αφελής τρόπος δημιουργίας νέφους σημείων, δεδομένου πως η έγχρωμη κάμερα και η κάμερα βάθους βρίσκονται σε διαφορετικές θέσεις πάνω στη συσκευή και κατά συνέπεια δεν καταγράφουν την ίδια εικόνα. Η χαμηλή ποιότητα του χάρτη βάθους που προκύπτει από απλή επίθεση είναι ιδιαίτερα εμφανής στα όρια των διαφόρων αντικειμένων στο χώρο που αναμενόμενα δεν ευθυγραμμίζονται πλήρως. Προκειμένου να αποφευχθεί το παραπάνω φαινόμενο, γίνεται, τρόπον τινά, μια ευθυγράμμιση των δύο καμερών, γνωστή ως αγκίστρωση (registration).

Εικόνα 5–4: Απλή επίθεση (αριστερά) έναντι αγκίστρωσης (δεξιά)
Πηγή: www.cs.princeton.edu

Η αγκίστρωση είναι μια θεμελιώδης αρχή της όρασης των υπολογιστών (computer vision - cv) με την οποία επιτυγχάνεται η ευθυγράμμιση dataset που ελήφθησαν σε διαφορετικά συστήματα συντεταγμένων. Αν πρόκειται για εικόνες, η αγκίστρωση γίνεται σε δύο διαστάσεις, ενώ τα νέφη σημείων αγκιστρώνονται σε τρεις διαστάσεις. Η διαδικασία ποικίλει από απλή μετάθεση έως πολύπλοκους μετασχηματισμούς. Πρόκειται, πρακτικά, για τη διαδικασία εύρεσης του πλέον κατάλληλου μετασχηματισμού για την ευθυγράμμιση και κατόπιν εφαρμογής του στο σύνολο των δεδομένων [26].

Εν προκειμένω, για την αγκίστρωση των σημείων του νέφους, επιλέγεται ο τρισδιάστατος μετασχηματισμός στερεού σώματος (3D rigid transformation) που περιλαμβάνει μια μετάθεση και μια στροφή, η εφαρμογή των οποίων δεν αλλοιώνει τη δομή των αντικειμένων. Μεταξύ δύο μη ομογενών συστημάτων συντεταγμένων, ο μετασχηματισμός έχει ως εξής:

$$p' = Rp + t \quad (18)$$

Όπου R ένας ορθογώνιος πίνακας διαστάσεων 3x3 για την εφαρμογή της στροφής και t ένα διάνυσμα διάστασης 3x1 για την εφαρμογή της στροφής. Για δύο νέφη σημείων P και P’ αντίστοιχα, όπου κάθε σημείο ρ_i αντιστοιχεί σε ένα σημείο ρ_i’, ο μετασχηματισμός γίνεται με εφαρμογή της στροφής και κατόπιν της μετάθεσης.

Για τον εφαρμογή της μετάθεσης, πρώτα αφαιρείται η μέση θέση από κάθε νέφος σημείων:

$$m = \frac{1}{N}\sum_{j=1}^{N} p_j \quad (19)$$

$$\check{p}_i = p_i - m \quad (20)$$

Έτσι, απλοποιείται το πρόβλημα καθώς τα "νέα" νέφη διαφέρουν πλέον μόνο ως προς στροφή. Τα σημεία κάθε νέφους εισάγονται σε ένα πίνακα $\check{P}$ διαστάσεων 3xN, όπου κάθε στήλη είναι ένα σημείο και τα πολλαπλασιάζουμε:

$$A = \check{P}\left(\check{P}'\right)^T \quad (21)$$

Κατόπιν εξάγουμε τον πίνακα στροφής μέσω ανάλυσης ιδιαζουσών τιμών (Singular Value Decomposition - SVD) που απλώνει ένα πίνακα σε τρεις (3) νέους, έστω U, D, V:

$$UDV^T = A \quad (22)$$

Οπότε προκύπτει στροφή: $R = VU^T$ (23)

Και μετάθεση από τη σχέση: $t = m' - Rm$ (24)

[5]

Σημειώνεται πως ένας εναλλακτικός τρόπος είναι η χρήση της PCL και η εφαρμογή πιο προηγμένων μεθόδων αγκίστρωσης όπως του αλγορίθμου ICP (Iterative Closest Point). Η μέθοδος αυτή αναλύεται διεξοδικά στο επόμενο κεφάλαιο.

Εν προκειμένω, η εφαρμογή των παραπάνω σε περιβάλλον προγραμματισμού είναι μάλλον απλούστερη. Ο απόλυτος προσανατολισμός δύο νεφών σημείων έχει ως εξής [21]:

```
//Inputs
cv::Mat1f X; //Nx3 matrix of points
cv::Mat1f Y; //Nx3 matrix of points
//Outputs
cv::Matx33f R; //3x3 rotation matrix (X=R*Y+T)
cv::Matx31f T; //3x1 translation matrix

cv::Matx31f meanX(0,0,0),meanY(0,0,0);
int point_count = X.rows;

//Calculate mean
for(int i=0; i<point_count; i++) {
    meanX(0) += X(i,0);
    meanX(1) += X(i,1);
    meanX(2) += X(i,2);
    meanY(0) += Y(i,0);
    meanY(1) += Y(i,1);
    meanY(2) += Y(i,2);
}
meanX *= 1.0f / point_count;
meanY *= 1.0f / point_count;

//Subtract mean
for(int i=0; i<point_count; i++) {
    X(i,0) -= meanX(0);
    X(i,1) -= meanX(1);
    X(i,2) -= meanX(2);
    Y(i,0) -= meanY(0);
    Y(i,1) -= meanY(1);
    Y(i,2) -= meanY(2);
}
//Rotation
cv::Mat1f A;
A = Y.t() * X;

cv::SVD svd(A);

cv::Mat1f Rmat;
Rmat = svd.vt.t() * svd.u.t();
Rmat.copyTo(R);

//Translation
T = meanX - R*meanY;
```

5.7 Ο αλγόριθμος ICP

Ο αλγόριθμος προτάθηκε από τους Besl και McKay και επιλύει ένα από τα σημαντικότερα προβλήματα βελτιστοποίησης σε N διαστάσεις. Έστω δύο (2) νέφη σημείων P και P', αποτελούμενα από σημεία ρ_i και ρ_i' αντίστοιχα, με i = 1 …N. Ο αλγόριθμος αναζητεί το εγγύτερο όμοιο σημείο μεταξύ των νεφών και εν συνεχεία το ολικό βέλτιστο. Προσπαθεί δηλαδή με διαδοχικές επαναλήψεις να βρει τη βέλτιστη μετατόπιση και στροφή ούτως ώστε το σύνολο των σημείων των νεφών να έχει την ελάχιστη δυνατή απόσταση.

Ως απόσταση χρησιμοποιείται η ευκλείδεια απόσταση μεταξύ σημείων στο χώρο:

$$D = \left|\left|p_i - {p'}_i\right|\right| = \sqrt{(x_i - {x'}_i)^2 + \left(y_i - {y'}_i\right)^2 + (z_i - {z'}_i)^2 + \cdots + (n_i - {n'}_i)^2} \quad (25)$$

Και ως προϋπόθεση για τη ταύτιση των σημείων λογίζεται μια τιμή κατωφλίου T:

$$\sum_{i=1}^{n} \left|\left|{p'}_i - (Rp_i + T)\right|\right|^2 < T \quad (26)$$

Το πρόβλημα είναι δυσεπίλυτο λόγω των τοπικών ελαχίστων που εμφανίζονται και για τη λύση του έχουν προταθεί πολλές μεθοδολογίες. Εν προκειμένω, όπως παραπάνω, υιοθετείται η ανάλυση ιδιαζουσών τιμών (SVD). [6]

5.8 Διαχείριση ακραίων τιμών

Η λανθασμένη αντιστοίχιση σημείων μεταξύ νεφών κατά την αγκίστρωση, μπορεί να οδηγήσει σε μεγάλα σφάλματα. Τα σφάλματα αυτά μπορούν να θεωρηθούν ακραίες τιμές - σημεία που δεν ταιριάζουν στο μοντέλο. Σε περίπτωση που τέτοια σημεία συμμετέχουν στο μετασχηματισμό, το αποτέλεσμα θα είναι εσφαλμένο. Επομένως, πρέπει να υπάρξει σχετική πρόβλεψη.

Ένας ενδεχόμενος τρόπος καταπολέμησης του φαινομένου είναι η απόδοση βαρών και η εφαρμογή κατωφλίων, ώστε τα σημεία αυτά να η λαμβάνονται υπόψιν. Εντούτοις, τέτοιες μέθοδοι, συνήθως μη γραμμικές, έχουν μεγάλο υπολογιστικό κόστος. Ένας εναλλακτικός και υπολογιστικά αποδοτικός τρόπος, είναι η εφαρμογή του αλγορίθμου RANSAC (RANdom SAmple Consensus) για την ανίχνευση των ακραίων τιμών. Ο αλγόριθμος αυτός προϋποθέτει πως τα σημεία αυτά αποτελούν μικρό ποσοστό του συνολικού δείγματος. Υπολογίζονται όλες οι εναλλακτικές αντιστοιχίσεις και προκρίνεται αυτή που παράγει το μικρότερο σφάλμα. [5] [14]

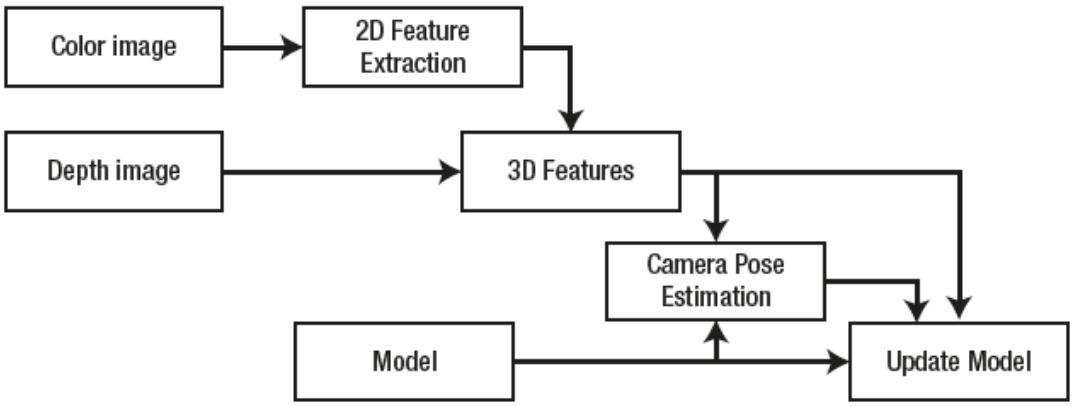
Η εφαρμογή του αλγόριθμου RANSAC για τον απόλυτο προσανατολισμό των νεφών γίνεται ως εξής:

```
void ransac_orientation(const cv::Mat1f &X, const cv::Mat1f &Y, cv::Matx33f &R, cv::Matx31f&T)
    {
    const int max_iterations = 200;
    const int min_support = 3;
    const float inlier_error_threshold = 0.2f;

    const int pcount = X.rows;
    cv::RNG rng;
    cv::Mat1f Xk(min_support,3), Yk(min_support,3);
    cv::Matx33f Rk;
    cv::Matx31f Tk;
    std::vector<int> best_inliers;

    for(int k=0; k<max_iterations; k++) {
        //Select random points
        for(int i=0; i<min_support; i++) {
            int idx = rng(pcount);
            Xk(i,0) = X(idx,0);
            Xk(i,1) = X(idx,1);
            Xk(i,2) = X(idx,2);
            Yk(i,0) = Y(idx,0);
            Yk(i,1) = Y(idx,1);
            Yk(i,2) = Y(idx,2);
    }

    //Get orientation
    absolute_orientation(Xk,Yk,Rk,Tk);

    //Get error
    std::vector<int> inliers;
    for(int i=0; i<pcount; i++) {
        float a,b,c,errori;
        cv::Matx31f py,pyy;
        py(0) = Y(i,0);
        py(1) = Y(i,1);
        py(2) = Y(i,2);
        pyy = Rk*py+T;
        a = pyy(0)-X(i,0);
        b = pyy(1)-X(i,1);
        c = pyy(2)-X(i,2);
        errori = sqrt(a*a+b*b+c*c);
        if(errori < inlier_error_threshold) {
            inliers.push_back(i);
        }
    }

    if(inliers.size() > best_inliers.size()) {
        best_inliers = inliers;
    }
}
std::cout << "Inlier count: " << best_inliers.size() << "/" << pcount << "\n";

//Do final estimation with inliers
Xk.resize(best_inliers.size());
Yk.resize(best_inliers.size());
for(unsigned int i=0; i<best_inliers.size(); i++) {
    int idx = best_inliers[i];
    Xk(i,0) = X(idx,0);
    Xk(i,1) = X(idx,1);
    Xk(i,2) = X(idx,2);
    Yk(i,0) = Y(idx,0);
    Yk(i,1) = Y(idx,1);
    Yk(i,2) = Y(idx,2);
}
absolute_orientation(Xk,Yk,R,T);
}
```

5.9 Simultaneous Localization and Mapping (SLAM)

Υπάρχουν πολλές διαφορετικές προσεγγίσεις της μεθόδου SLAM, ανάλογα με το είδος των αισθητήρων και των αλγορίθμων που χρησιμοποιούνται. Σε κάθε περίπτωση, η μέθοδος SLAM ακολουθεί πέντε (5) βασικά βήματα. Αρχικά, δέχεται ως είσοδο ένα καρέ (frame) της έγχρωμης κάμερας και τη σχετική πληροφορία βάθους που τη συνοδεύει. Ακολουθεί ο εντοπισμός χαρακτηριστικών σημείων ενδιαφέροντος (key points) επί της εικόνας και ταύτιση τους με τα ανάλογα σημεία σε παρελθοντικά καρέ. Έπειτα, γίνεται υπολογισμός της σχετικής μετατόπισης του συστήματος στον τρισδιάστατο χώρο σύμφωνα με τη σχετική πληροφορία βάθους. Το τελευταίο βήμα αφορά την διόρθωση της σχετικής κίνησης και του συσσωρευμένου σφάλματος (drift).

Η εικόνα που ακολουθεί καταδεικνύει πως η μέθοδος SLAM σε συνδυασμό με το Kinect ενσωματώνει και βελτιώνει τις παραπάνω αρχές:



Εικόνα 5–5: Διάγραμμα ροής της μεθόδου SLAM σε συνδυασμό με το Kinect
Πηγή: J. Kramer, N.Burrus, D. Herrera C., F. Echtler, M. Parker, "Hacking the Kinect", 2012

Η βελτίωση αφορά την λήψη μέτρων για τη διόρθωση του σφάλματος που προκύπτει από τη κίνηση της κάμερας, ήτοι του προσανατολισμού της λήψης (camera pose estimation) με αγκίστρωση του νέφους απευθείας στις τρεις (3) διαστάσεις (3D registration), σε σχέση με τις συμβατικές κάμερες.

Το μεγαλύτερο μειονέκτημα της μεθόδου είναι αναμφισβήτητα η συσσώρευση του σφάλματος (drift accumulation). Καθώς ο μετασχηματισμός γίνεται με διαδοχικούς πολλαπλασιασμούς πινάκων και παρά το γεγονός πως το σφάλμα που παράγεται είναι μικρό, οι συνεχείς επαναλήψεις το διογκώνουν. Το αποτέλεσμα είναι εμφανές αν και μετριάζεται από τη διόρθωση του προσανατολισμού της λήψης. Ο καλύτερος τρόπος εξάλειψης του σφάλματος είναι το κλείσιμο του βρόγχου (loop closure) – η επιστροφή δηλαδή της κάμερας στην αρχική θέση λήψης για τη συνολική διόρθωση

του μοντέλου. Δυστυχώς η μέθοδος αυτή δεν υποστηρίζεται από το λογισμικό RGBDemo v 0.7.0.

Τέλος, προκειμένου η μέθοδος να είναι αποτελεσματική σε πραγματικό χρόνο, πρέπει η αναβάθμιση του μοντέλου που γίνεται μετά από τη λήψη κάθε καρέ, να απεμπλακεί από τη συνολική διόρθωση του σφάλματος. Η τελευταία πρέπει να γίνεται κάθε λίγα δευτερόλεπτα δεδομένου πως είναι υπολογιστικά ακριβή, όπως άλλωστε και το σύνολο της μεθόδου. [1] [5]

6. Η ΕΦΑΡΜΟΓΗ

Η χρήση της εφαρμογής διακρίνεται ουσιαστικά σε δύο μέρη, τη πλοήγηση στο γραφικό της περιβάλλον και την παραγωγή του τελικού προϊόντος. Η μεν διεπαφή αποτελείται από δύο παράθυρα χρήστη, το δε προϊόν συνίσταται ουσιαστικά από ένα αρχείο PLY.

6.1 Γραφικό Περιβάλλον Χρήστη

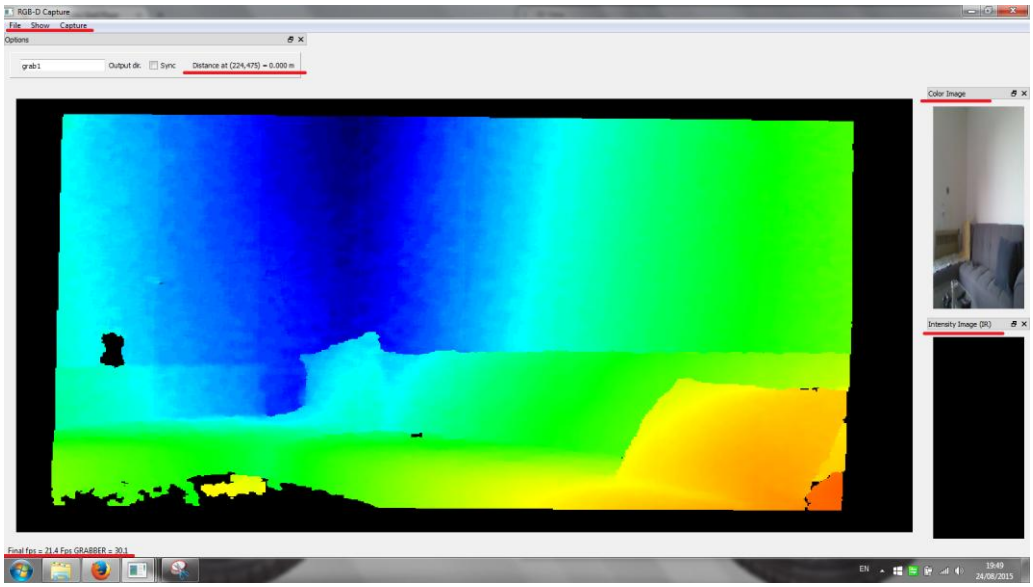
Το Γραφικό Περιβάλλον Χρήστη (Graphical User Interface – GUI) είναι μια διεπαφή, ένα σύνολο γραφικών στοιχείων στην οθόνη του χρήστη που του επιτρέπουν την αλληλεπίδραση με τη συσκευή προκειμένου να εκτελέσει τις λειτουργίες που επιθυμεί. Επί της ουσίας, η διεπαφή ερμηνεύει τα γεγονότα των συσκευών εισόδου (π.χ. πληκτρολόγιο, ποντίκι) στη γλώσσα της συσκευής και επιτρέπει στο χρήστη την αλληλεπίδραση με αυτή, με τρόπο που ταιριάζει στην ανθρώπινη εμπειρία και φύση.

Εν προκειμένω, η εκκίνηση του rgbd-reconstructor.exe εμφανίζει ως έξοδο στην οθόνη του χρήστη δύο παράθυρα εργασίας. Στο πρώτο παράθυρο, ονόματι RGB-D Capture, κυριαρχεί η εικόνα βάθους που καταγράφει ο αισθητήρας Kinect, ενώ διακρίνονται και η αντίστοιχη εικόνα της έγχρωμης κάμερας και εικόνα έντασης. Επιπλέον, παρέχεται πληροφόρηση σχετικά με τον τρέχοντα αριθμό καρέ ανά δευτερόλεπτο (frames per second - fps) καθώς και η απόσταση από το στόχο.

Τέλος, διακρίνεται ένα μενού επιλογών που παρέχουν περεταίρω λειτουργικότητα. Το μενού αυτό επιτρέπει τη λήψη ενός μόνο καρέ, την εισαγωγή φίλτρων και την εναλλαγή των παραθύρων - και δεν αποτελούν αντικείμενο της παρούσης.

Σημειώνεται πως η πλειοψηφία των επιλογών (options) που αφορούν στη λειτουργικότητα της εφαρμογής, δεν συναντώνται στο παραπάνω μενού αλλά εισάγονται ως παράμετροι (flags) κατά την εκκίνηση της εφαρμογής σε περιβάλλον γραμμής εντολών (command line) και δεν δύνανται να αλλάξουν πριν τη λήξη της εκάστοτε χρήσης. Οι επιλογές αφορούν, μεταξύ άλλων, στη χρήση του αλγορίθμου ICP για βελτίωση της ποιότητας του παραγόμενου νέφους, την επιθυμητή θέση αποθήκευσης των παραγόμενων αρχείων, επιλογή μεταξύ οδηγών (drivers) και της ανάλυσης της έγχρωμης κάμερας.

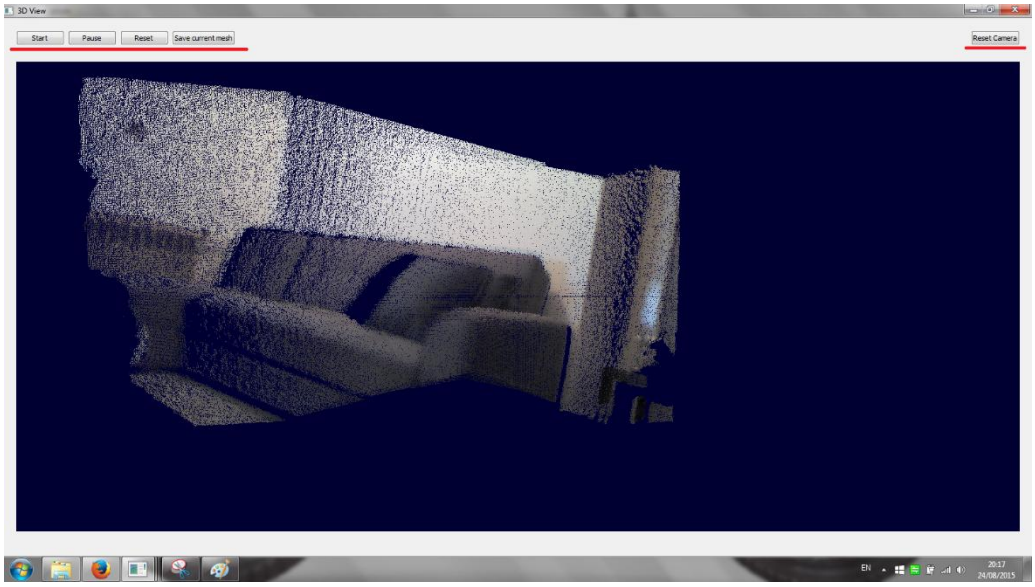
Στην εικόνα που ακολουθεί, τα σχετικά γραφικά στοιχεία έχουν υπογραμμιστεί με κόκκινο χρώμα προκειμένου να είναι ευδιάκριτα.



Εικόνα 6–1: Τα γραφικά στοιχεία του παραθύρου RGB-D Capture
Πηγή: Ίδια εικόνα

Στο δεύτερο και πλέον σημαντικό παράθυρο, ονόματι 3D View, διακρίνεται δυναμικά το νέφος σημείων που δημιουργείται με την ελεύθερη κίνηση του αισθητήρα σε πραγματικό χρόνο. Ο χρήστης έχει τη δυνατότητα περιστροφής του μοντέλου, εντούτοις στερείται τη δυνατότητα μεγέθυνσης ή σμίκρυνσης (zoom in/out) καθώς και οριζόντιας ή κάθετης μετακίνησης (pan) αυτού. Επίσης παρέχεται η δυνατότητα εκκίνησης (start), παύσης (pause), επαναφοράς (reset) της λήψης και του προσανατολισμού της κάμερας, καθώς και αποθήκευσης της λήψης.

Όμοια, στην εικόνα που ακολουθεί, τα σχετικά γραφικά στοιχεία έχουν υπογραμμιστεί με κόκκινο χρώμα προκειμένου να είναι ευδιάκριτα.



Εικόνα 6–2: Τα γραφικά στοιχεία του παραθύρου 3D View
Πηγή: Ίδια εικόνα

6.2 Μορφότυπος (format)

Ο μορφότυπος PLY (Polygon File Format) σχεδιάστηκε εξ αρχής με γνώμονα την αναπαραγωγή τρισδιάστατης πληροφορίας. Διακρίνεται σε δύο επιμέρους μορφότυπους: ASCII για ευκολία χρήσης και δυαδικό (binary) για γρήγορη εκκίνηση και αποθήκευση.

Η αναπαράσταση των αντικείμενων του χώρου γίνεται με τη μιας μορφή συλλογής επιπέδων πολυγώνων, στα οποία μπορούν να αποδοθούν ιδιότητες όπως χρώμα, βαθμός διαφάνειας, υφή, συντεταγμένες και βαθμός εμπιστοσύνης. Σημειώνεται πως ο μορφότυπος δεν διακρίνει μεταξύ αντικειμένων στο χώρο. Αντίθετα θεωρεί όλη τη σκηνή ως ένα αντικείμενο.

Ως εκ τούτου, δεδομένης και της φύσης της εφαρμογής, τα αρχεία αυτά είναι συνήθως ικανού μεγέθους. Επιπλέον, δύνανται να υποστούν περεταίρω επεξεργασία και βελτιστοποίηση με χρήση κατάλληλου λογισμικού όπως τα MeshLab και Blender. Η επεξεργασία συνήθως περιλαμβάνει τον καθορισμό του νέφους από τυχόν θόρυβο ή/και ακραίες τιμές, συρραφή διαδοχικών νεφών κλπ. - και δεν αποτελεί αντικείμενο της παρούσης εργασίας. [8] [23]

Η τυπική δομή ενός τέτοιου αρχείου είναι η παρακάτω:

```
ply
format ascii 1.0
element vertex 897349
property float x
property float y
property float z
property float nx
property float ny
property float nz
property uchar red
property uchar green
property uchar blue
end_header
-0.00296346 0.00217294 -1.68876 -0.627798 0.652503 0.424393 18 6 41
-0.00251487 -0.000797849 -1.68132 -0.772982 0.414607 0.480208 8 23 35
-0.00195516 -0.00382787 -1.66623 -0.854148 0.0849686 0.513041 28 4 30
0.00303925 -0.000404546 -1.65413 -0.413118 0.84334 0.343673 15 16 11
0.00236304 -0.00327914 -1.6474 -0.646249 -0.500563 0.576021 30 17 21
-0.00343543 -0.000232736 -1.64179 -0.69992 0.0097747 0.714154 22 3 18
0.00397352 0.00284384 -1.64498 -0.126689 0.830903 0.541803 21 29 34
8.78299e-005 -0.000661007 -1.63653 -0.56901 0.0966898 0.816626 25 25 31
0.000591348 0.000355976 -1.63196 -0.676114 0.32705 0.660234 29 18 30
0.000491831 0.00115166 -1.6266 -0.518243 -0.361072 0.775275 42 28 33
```

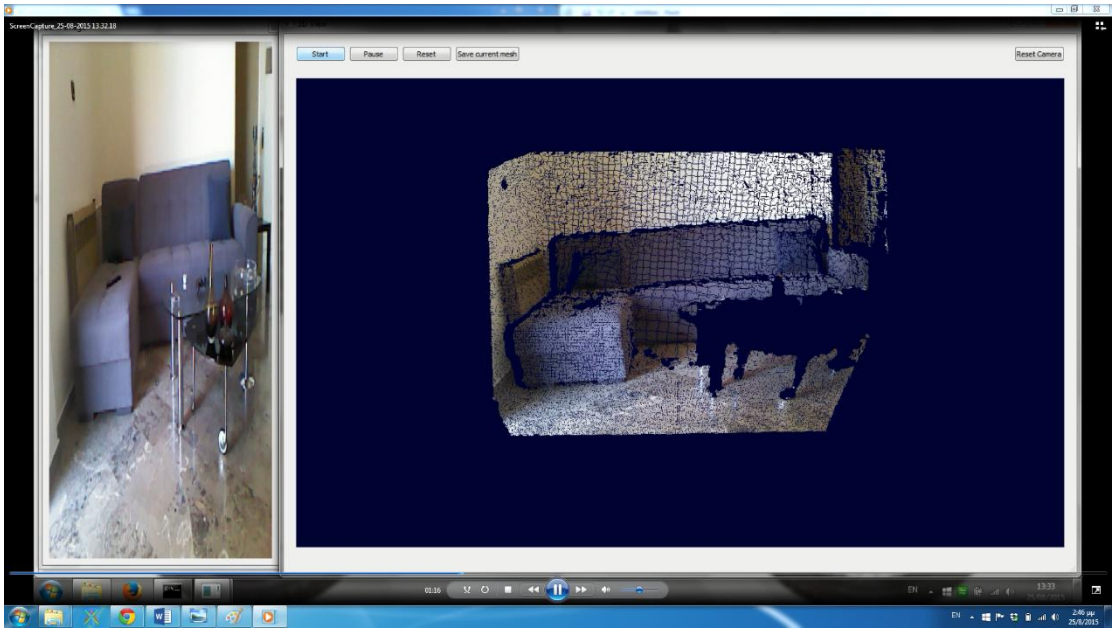
Η κεφαλίδα (header) περιγράφει τον τύπο του αρχείου και των μεταβλητών, τις ιδιότητες που περιέχει η λίστα, καθώς και το πλήθος των στοιχείων.

Ο μορφότυπος θεωρείται γενικά απλός και ευέλικτος, καθώς δύναται να αναγνωστεί επιμέρους από προγράμματα που δεν αναγνωρίζουν το περιεχόμενο στο σύνολο του.

6.3 Παραδείγματα

Παρακάτω, γίνεται προσπάθεια απόδοσης του τρόπου λειτουργίας της εφαρμογής με χρήση εικόνων. Οι επιλεγείσες εικόνες είναι χαρακτηριστικές της λειτουργικότητας της εφαρμογής και αφορούν στη μοντελοποίηση μέρους ενός διαμερίσματος σε πραγματικό χρόνο και επισκόπηση του αποτελέσματος, ενώ συνοδεύονται από σύντομο σχολιασμό αναφορικά με την ποιότητα του νέφους σε συνάρτηση με τα αντικείμενα του χώρου και της συνθήκες της λήψης. Αναλυτικός σχολιασμός και συγκριτικά αποτελέσματα ακολουθούν στο επόμενο κεφάλαιο.

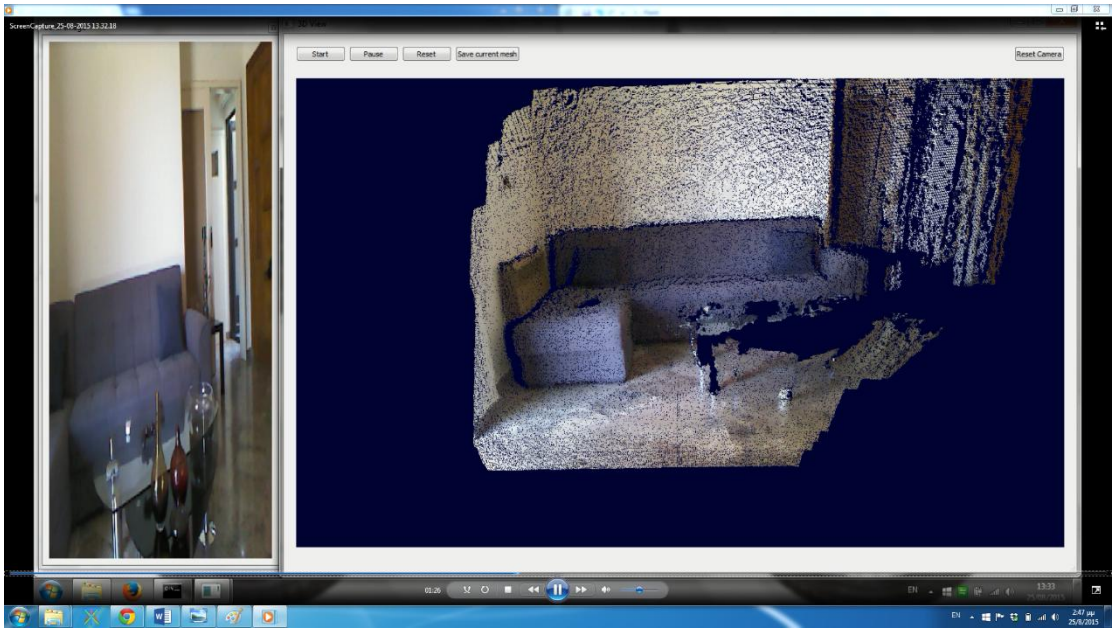
Επελέγη η διατήρηση της λήψης (παραμορφωμένης έστω) στο αριστερό μέρος, ούτως ώστε ο αναγνώστης να έχει επίγνωση της εικόνας που λαμβάνει ο αισθητήρας Kinect τη δεδομένη στιγμή και πως αυτή αποτυπώνεται/προστίθεται στο μοντέλο. Το μεγαλύτερο μέρος της οθόνης καταλαμβάνει το παράθυρο 3D View, στο οποίο αναπαρίσταται σε πραγματικό χρόνο (δυναμικά) το νέφος σημείων.



Εικόνα 6–3: Το πρώτο καρέ
Πηγή: Ίδια εικόνα

Με την επιλογή του κομβίου εκκίνησης (βλ. Start), η πρώτη εικόνα που καταγράφει το Kinect και το αντίστοιχο νέφος είναι αυτή του καθιστικού. Διακρίνει κανείς πως τόσο ο τοίχος, το πάτωμα και ο καναπές, όσο και το καλοριφέρ, μολονότι διαφορετικής υφής και χρώματος, αποτυπώνονται σε ικανοποιητικό βαθμό.

Εντούτοις, το γυάλινο τραπέζι εξαιτίας της ανακλαστικότητας του, δεν καταγράφεται – τουλάχιστον σε αυτή τη λήψη.

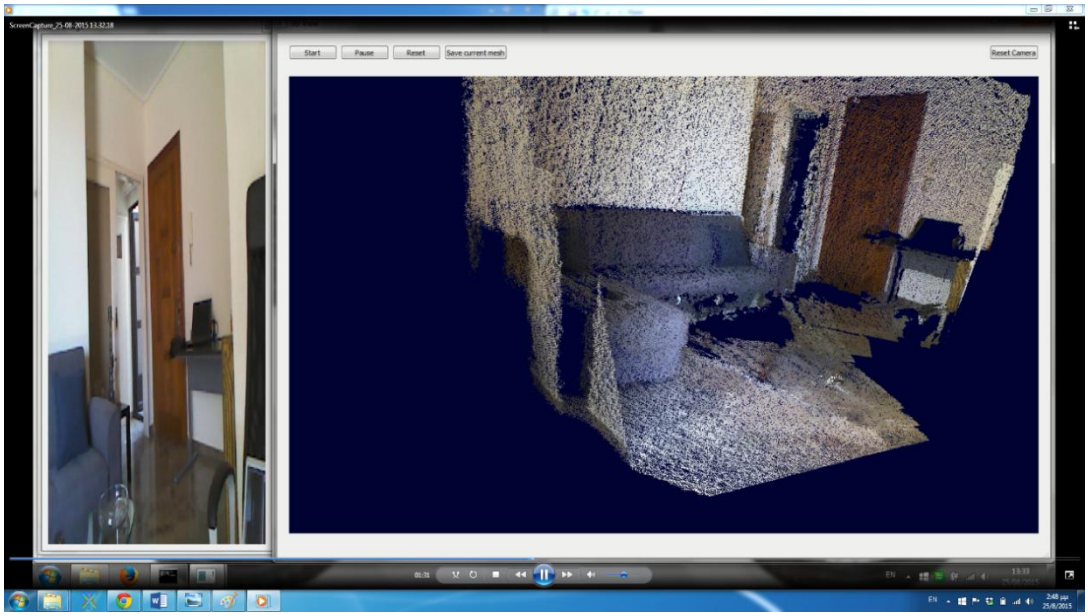


Εικόνα 6–4: Βελτίωση νέφους σημείων προϊόντος του χρόνου
Πηγή: Ίδια εικόνα

Λίγα δευτερόλεπτα αργότερα και ενώ ο αισθητήρας κινείται δεξιόστροφα, οι επικαλυπτόμενες λήψεις σε συνδυασμό με την εφαρμογή του αλγορίθμου ICP (αναλύθηκε σε προηγούμενο κεφάλαιο) σε πραγματικό χρόνο, η ποιότητα του νέφους βελτιώνεται, τα σημεία πυκνώνουν και το αποτέλεσμα είναι αισθητικά και λειτουργικά καλύτερο.

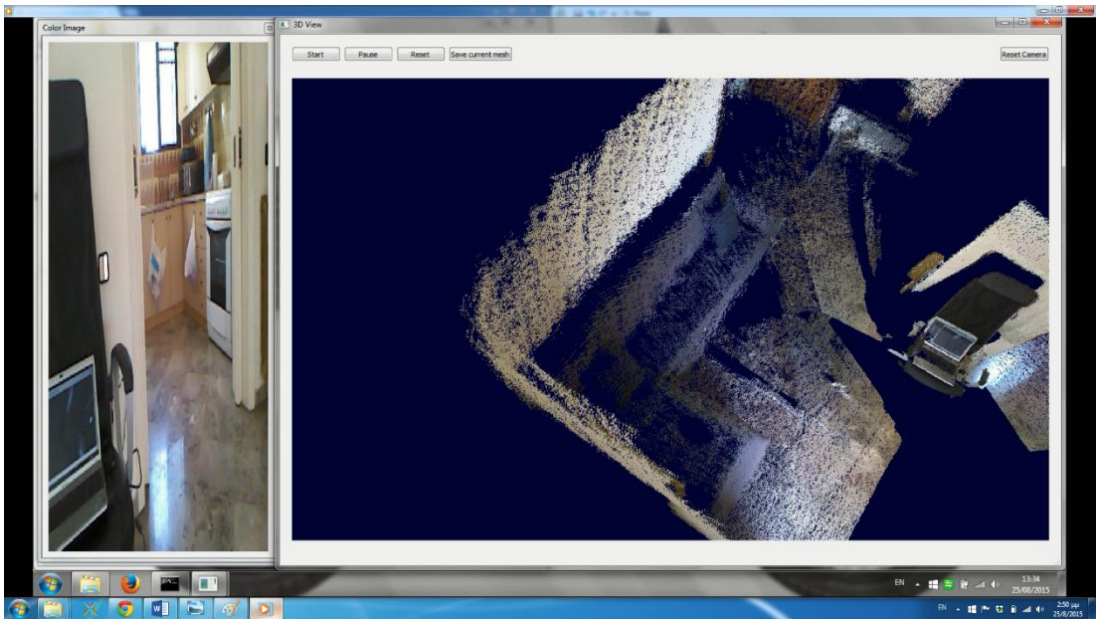
Εικόνα 6–5: Επόμενες λήψεις
Πηγή: Ίδια εικόνα

Οι επόμενες λήψεις και ενώ ο αισθητήρας συνεχίζει την ελεύθερη κίνηση του στο χώρο (δεξιόστροφα για τους σκοπούς της παρούσης), τα αντικείμενα του χώρου καταγράφονται και αναπαριστώνται με τη μορφή νέφους σημείων και προστίθενεται στο μοντέλο.



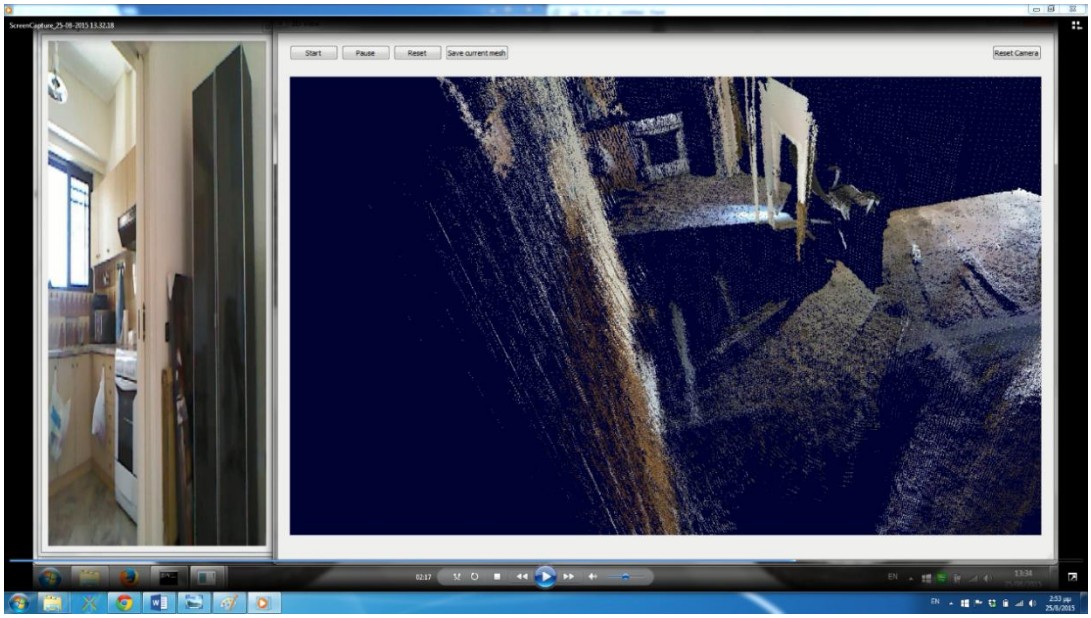
Εικόνα 6–6: Το μοντέλο προϊόντος του χρόνου
Πηγή: Ίδια εικόνα

Το μοντέλο συμπληρώνεται και αναβαθμίζεται προϊόντος του χρόνου, με το βαθμό λεπτομέρειας του να εξαρτάται από τη διάρκεια της λήψης και τα χαρακτηριστικά του αισθητήρα. Παρατηρείται πως η νέα λήψη καταγράφει εν μέρει και το γυάλινο τραπέζι.



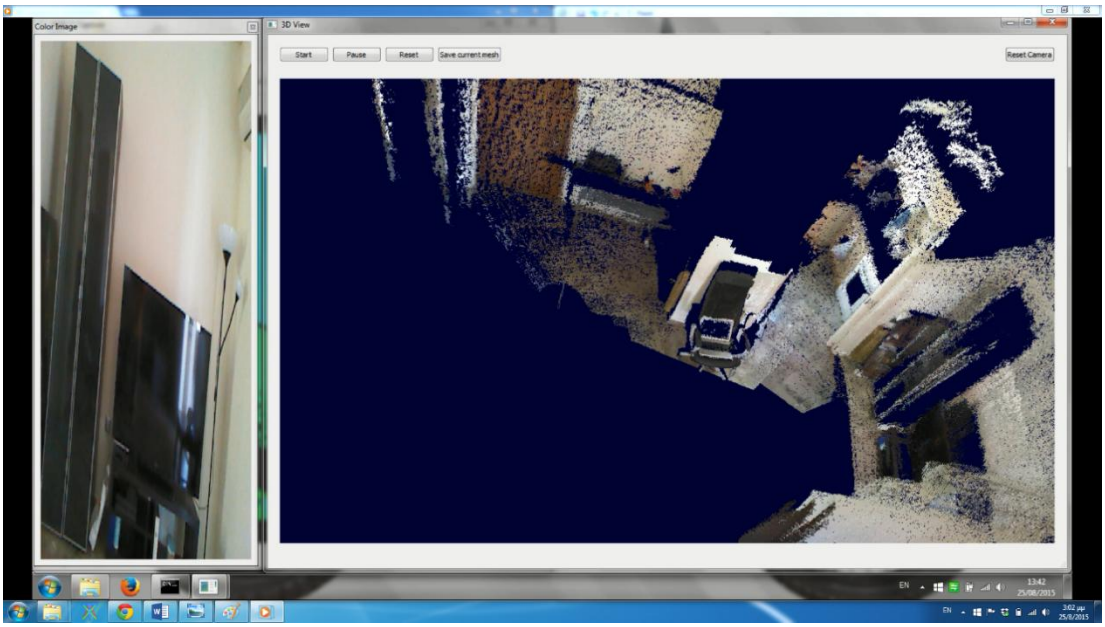
Εικόνα 6–7: Εναλλακτική όψη
Πηγή: Ίδια εικόνα

Το λογισμικό παρέχει τη δυνατότητα περιστροφής του μοντέλου ούτως ώστε ο χρήστης να έχει τη δυνατότητα εποπτείας του μοντέλου από εναλλακτικές όψεις. Εντούτοις η αδυναμία μετακίνησης στον οριζόντιο και κάθετο άξονα δυσχεραίνει το έργο του χρήστη καθώς το μοντέλο μεγαλώνει σε μέγεθος.



Εικόνα 6–8: Επόμενες λήψεις
Πηγή: Ίδια εικόνα

Στην παραπάνω εικόνα από μια εναλλακτική όψη επιτρέπει στο χρήστη να παρακολουθήσει σε πραγματικό χρόνο τη προσάρτηση ενός νέου δωματίου (σ.σ. Κουζίνα στο βάθος), όπως άλλωστε διακρίνεται στην έγχρωμη κάμερα.



Εικόνα 6–9: Αποτέλεσμα
Πηγή: Ίδια εικόνα

Το αποτέλεσμα έπειτα από μια σύντομη σάρωση όπως διακρίνεται στη παραπάνω εικόνα, είναι αντιπροσωπευτικό του χώρου στο βαθμό που λαμβάνονται υπόψη ο ελάχιστος χρόνος που δαπανήθηκε και το κόστος του εξοπλισμού. Το μοντέλο μπορεί να βελτιωθεί σημαντικά αφιερώνοντας επιπλέον χρόνο και κατά συνέπεια περισσότερες λήψεις, εντούτοις πρέπει να ληφθούν υπόψη οι απαιτήσεις της εφαρμογής σε υπολογιστική ισχύ και το μέγεθος του δημιουργούμενου αρχείου PLY που μπορεί να οδηγήσουν σε μη επιθυμητά αποτελέσματα.

Εναλλακτικά, μπορεί να γίνει συρραφή επιμέρους, πιο λεπτομερών μοντέλων με χρήση σχετικών προγραμμάτων (βλ. MeshLab, Blender). Η εναλλακτική αυτή λύση επιτρέπει την χρήση εργαλείων για το καθαρισμό του νέφους από θόρυβο και εκ των υστέρων κλείσιμο του βρόγχου, γεγονός που αποτελεί και τη συνήθη πρακτική. Είναι χαρακτηριστικό στην εικόνα 6-7, πως η γυάλινη επιφάνεια του φωριαμού έχει ως αποτέλεσμα τα ανακλώμενα σε αυτή στοιχεία να εμφανίζονται στο μοντέλο εν είδη αρνητικής εικόνας.

Σε κάθε περίπτωση, το αντικείμενο της παρούσης εξαντλείται στην υλοποίηση μιας μεθόδου τρισδιάστατης χαρτογράφησης εσωτερικών χώρων με χρήση του αισθητήρα Kinect for Windows (v1) και επί της αρχής κρίνεται επιτυχημένο.

Το κεφάλαιο που ακολουθεί υπεισέρχεται σε λεπτομέρειες αναφορικά με την ποιότητα του νέφους σημείων που δημιουργείται.

7. ΠΟΙΟΤΙΚΟΣ ΕΛΕΓΧΟΣ

Επί της αρχής, ο στόχος της τρισδιάστατης χαρτογράφησης εσωτερικών χώρων με χρήση του αισθητήρα Kinect for Windows (v1), επιτυγχάνεται σε συνδυασμό με το λογισμικό RGBDemo 0.7.0, της PCL, της OpenCV και του OpenNI. Εντούτοις, παραμένει το ζήτημα της ποιότητας του τελικού αποτελέσματος (βλ. νέφους σημείων), που θα καθορίσει εν πολλοίς και το πεδίο εφαρμογής της παρούσης.

Ακολουθεί, κατά σειρά, μια αναλυτική παρουσίαση του εξοπλισμού, του λογισμικού και της μεθοδολογίας που χρησιμοποιήθηκε για τη διενέργεια τη σύγκρισης και την εξαγωγή των σχετικών συμπερασμάτων.

7.1 Σαρωτής

Για τη διενέργεια της σύγκρισης επιστρατεύθηκε σαρωτής - laser scanner. Γενικά, οι σαρωτές laser (Light Amplification by Stimulated Emission of Radiation) είναι μηχανήματα υψηλής τεχνολογίας και κόστους, τα οποία τυγχάνουν συνεχώς μεγαλύτερης υιοθέτησης τα τελευταία χρόνια, χάρη στην ευκολία χρήσης και τα εξαιρετικά ακριβή αποτελέσματα που προσφέρουν.

Οι σαρωτές βασίζουν τη λειτουργία τους στη τεχνολογία LIDAR (Light Detection And Ranging) που συνίσταται στην εκπομπή ακτινοβολίας και ανίχνευση της ανάκλασης της με ένα μηχανισμό κατόπτρων για τη ταχεία συλλογή μετρητικής (X, Y, Z) και χρωματικής (grayscale ή RGB) πληροφορίας. Ο χειριστής επιλέγει απλά τον επιθυμητό στόχο (σ.σ. αντικείμενο) και ορίζει την επιθυμητή πυκνότητα σημείων και ο σαρωτής παράγει το αντίστοιχο νέφος ομοιόμορφα κατανεμημένων σημείων.

Εικόνα 7–1: Ο στόχος (πεζογέφυρα Ραφήνας) και το αντίστοιχο νέφος σημείων
Πηγή: Α. Β. Αναστασίου, Δ. Πρέκα, "Η Χρήση της Τρισδιάστατης Σάρωσης στις Τοπογραφικές Αποτυπώσεις», 2011

Τα προϊόντα της σάρωσης δύνανται επιπλέον του νέφους σημείων να είναι πλέγμα τριγώνων (polygon meshes), εικόνες απόστασης (range images) και παραμετρικές επιφάνειες ή μοντέλα. Αναμενόμενα, η παρούσα εργασία επικεντρώνεται στα νέφη σημείων.

Εν προκειμένω, επελέγη ο σαρωτής FARO Focus 3D με τα εξής τεχνικά χαρακτηριστικά [31]:

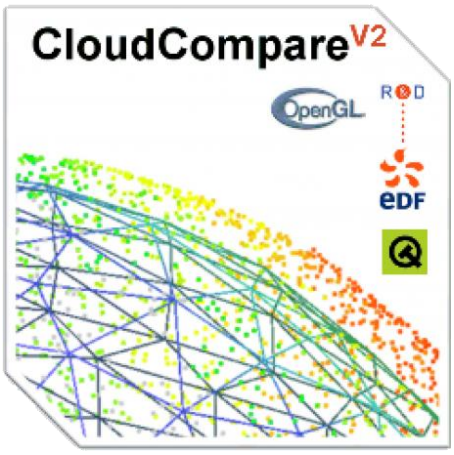
- Εμβέλεια: 0,6 – 130 m
- Ταχύτητα καταγραφής: έως 976.000 σημεία ανά δευτερόλεπτο
- Σφάλμα: $\mp 2\ mm$
- Κλάση: laser class 1
- Βάρος: 5,2 kg
- Αισθητήρες: GPS, Πυξίδα, Υψομετρικός, Αξονικός
- Μέγεθος: 240 x 200 x 100 mm
- Χειρισμός: έγχρωμη οθόνη αφής και WLAN
- Μορφότυπος αρχείων εξόδου: FLS/FWS

Εικόνα 7–2: Ο σαρωτής laser FARO Focus 3D
Πηγή: www.faro.com

7.2 CloudCompare

Για την επεξεργασία των νεφών που προέκυψαν ως αποτέλεσμα της σάρωσης τόσο με τον αισθητήρα Kinect, όσο και από τον σαρωτή laser, επιστρατεύθηκε το λογισμικό CloudCompare v2.6.1.

Το CloudCompare δημιουργήθηκε ως αποτέλεσμα της συνεργασίας μεταξύ της Telecom ParisTech και του τμήματος έρευνας και ανάπτυξης του EDF και βασίζεται στη διδακτορική διατριβή του Daniel Giradeu-Montaut το 2003. Σήμερα διατίθεται ως ανοικτό και ελεύθερο. [32]



Εικόνα 7–3: Το λογότυπο του λογισμικού CloudCompare
Πηγή: www.cloudcompare.org

Επί της ουσίας, παρέχει ένα σύνολο αλγορίθμων και εργαλείων για την επεξεργασία νεφών σημείων και πλεγμάτων τριγώνων. Ορισμένες από τις δυνατότητες που παρέχει είναι η αγκίστρωση νεφών, η προβολή, η κατάτμηση, η εξαγωγή στατιστικών και γεωμετρικών χαρακτηριστικών.

Δέχεται ως είσοδο και αντίστοιχα παράγει ως έξοδο, αρχεία στους παρακάτω μορφότυπους: BIN, ASCII, **PLY**, OBJ, VTK, STL,E57, LAS, PCD, FBX, SHP, OFF, PTX, **FLS**, **FWS** και DP.

7.3 Μεθοδολογία

Η μεθοδολογία που ακολουθήθηκε προκειμένου να γίνει σύγκριση των παραγόμενων νεφών σημείων μεταξύ του αισθητήρα Kinect for Windows (v1) και του laser scanner FARO Focus 3D αναλύεται στα κάτωθι:

Επελέγησαν τέσσερα καθημερινά αντικείμενα με διαφορετικά χαρακτηριστικά έκαστο ως προς το μέγεθος, το υλικό, το χρώμα, την υφή, και την ανακλαστικότητα. Τα αντικείμενα αυτά τοποθετήθηκαν επί βάθρου και αποτέλεσαν στόχο σάρωσης των

δύο συσκευών επί δύο (2) λεπτά και από απόσταση ενός (1), δύο (2), τριών (3) και τεσσάρων (4) μέτρων – σύμφωνα με την εμβέλεια του Kinect.

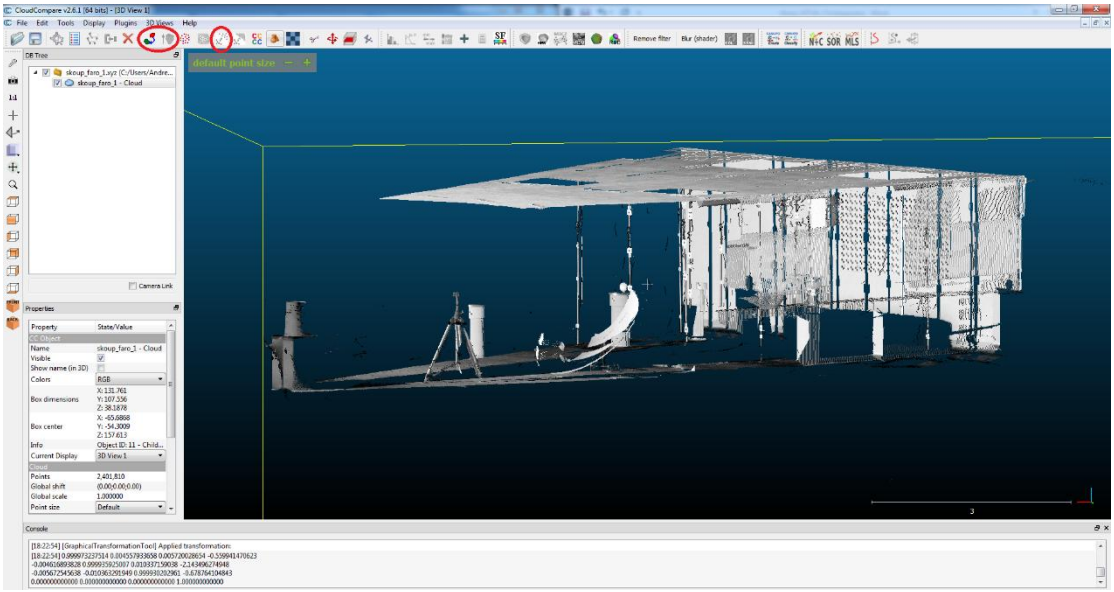
Εικόνα 7–4: Τα αντικείμενα - στόχοι
Πηγή: Ίδια εικόνα

Η ίδια μεθοδολογία ακολουθήθηκε και με μειωμένες συνθήκες φωτισμού για δύο αντικείμενα και μόνο για τον αισθητήρα Kinect, προκειμένου να διαπιστωθεί κατά πόσο επηρεάζει ο περιβάλλον φωτισμός το αποτέλεσμα της σάρωσης.

Σημειώνεται πως η σύγκριση αποσκοπεί στον προσδιορισμό της τάξης μεγέθους του σφάλματος που παράγει η σάρωση με το Kinect και όχι για την ευθεία σύγκριση μεταξύ των δύο συσκευών/τεχνολογιών, δεδομένης της διαφοράς κόστους και αντικειμένου (φυσική διεπαφή έναντι εξειδικευμένου τοπογραφικού εργαλείου). Με γνώμονα τα παραπάνω η επιθυμητή πυκνότητα της σάρωσης του σαρωτή ορίστηκε στο ¼ της μέγιστης δυνατής.

Όπως προαναφέρθηκε, η επεξεργασία των νεφών σημείων που προέκυψαν από τις σαρώσεις, έγινε με χρήση του λογισμικού CloudCompare και περιελάμβανε τα παρακάτω βήματα:

- ✓ Κατάτμηση (segmentation) του τμήματος του νέφους (σ.σ. αντικείμενο) προς σύγκριση.
- ✓ Αγκίστρωση (point-pair registration) των αντίστοιχων νεφών σημείων και εξαγωγή του μέσου τετραγωνικού σφάλματος (RMS).
- ✓ Εξαγωγή γραφήματος με τις απόλυτες αποστάσεις μεταξύ των δύο νεφών (C2C absolute distance).
- ✓ Εφαρμογή του αλγορίθμου ICP στα δύο νέφη και επανυπολογισμός του μέσου τετραγωνικού σφάλματος (RMS).



Εικόνα 7–5: Η επιφάνεια εργασίας του CloudCompare με τα προαναφερθέντα εργαλεία
Πηγή: Ίδια εικόνα

Παρακάτω παρατίθενται τα αποτελέσματα της σύγκρισης με σύντομο σχολιασμό, ενώ σε επόμενο κεφάλαιο ακολουθεί περεταίρω ανάλυση και εξαγωγή σχετικών συμπερασμάτων.

7.4 Αποτελέσματα σύγκρισης

Παρακάτω διακρίνονται κατά σειρά τα αποτελέσματα της σύγκρισης των νεφών σημείων που παρήχθησαν από Kinect ή/και σαρωτή, για κάθε αντικείμενο σε κάθε απόσταση.

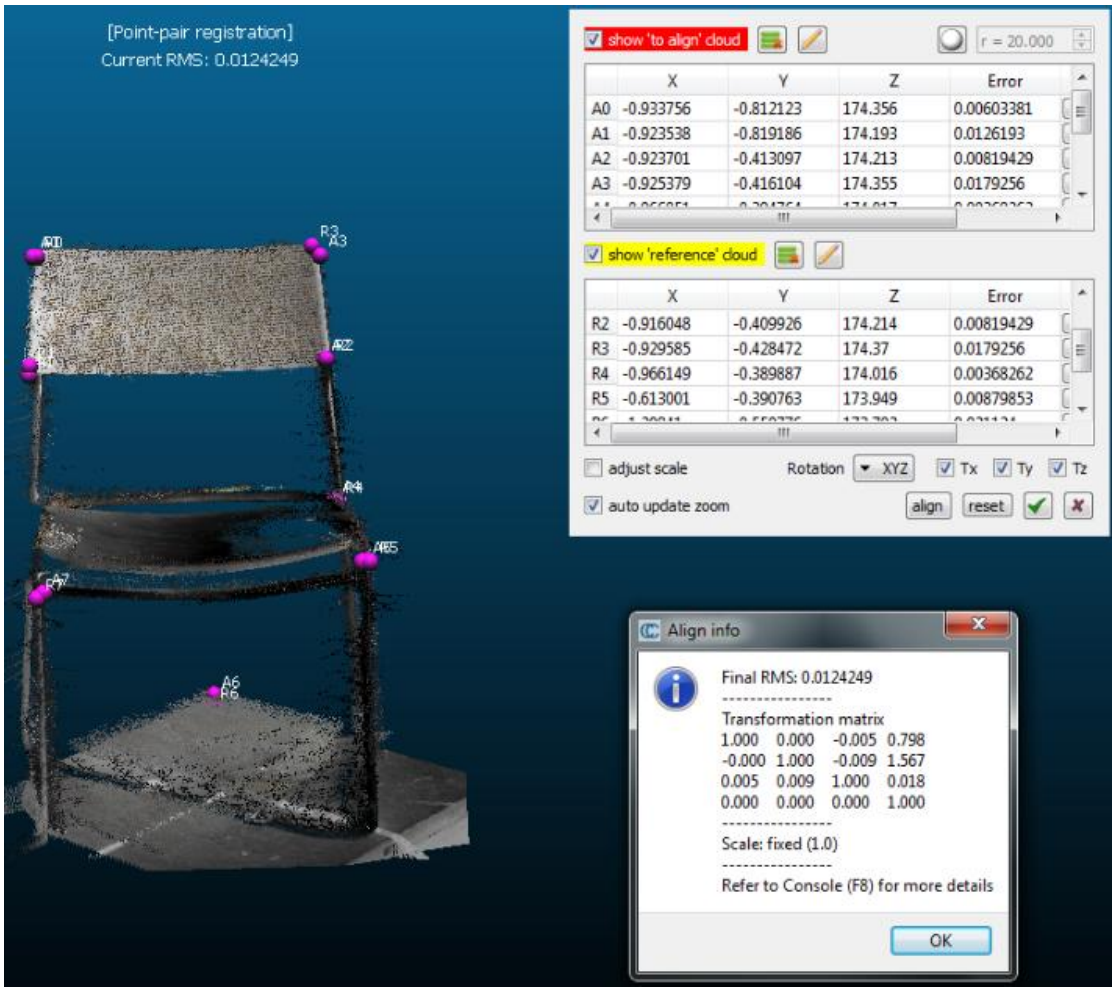
Σημειώνεται πως τα αποτελέσματα σχολιάζονται αναλυτικά και παρατίθενται συγκεντρωτικά με τη μορφή πίνακα στο επόμενο κεφάλαιο.

7.4.1 Με κριτήριο την απόσταση

Σκοπός είναι ο προσδιορισμός της επίδρασης της απόστασης στη ποιότητα του παραγόμενου νέφους σημείων του αισθητήρα Kinect. Σε κάθε περίπτωση, ως νέφος αναφοράς επελέγη το νέφος σημείων που παρήχθη από το σαρωτή.

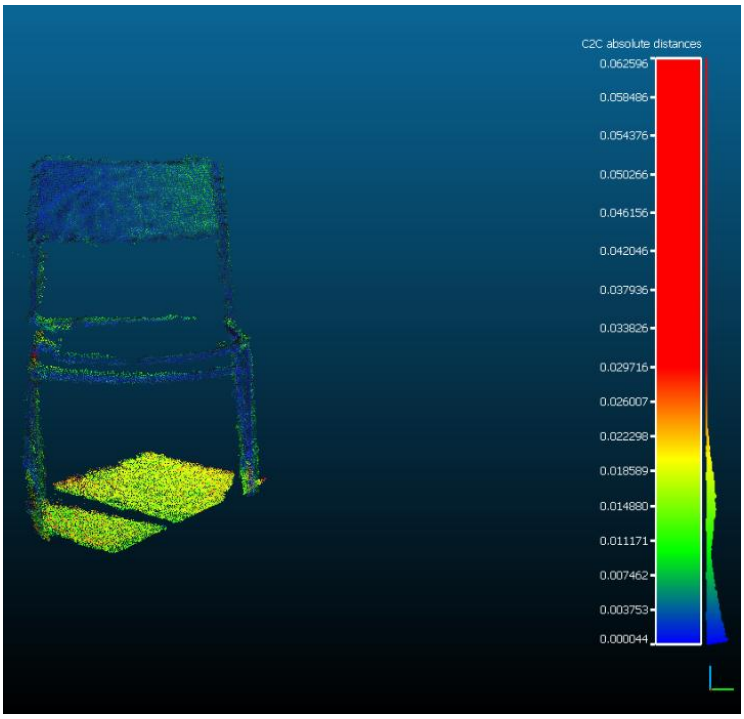
7.4.1.1 Απόσταση 1m

Καρέκλα



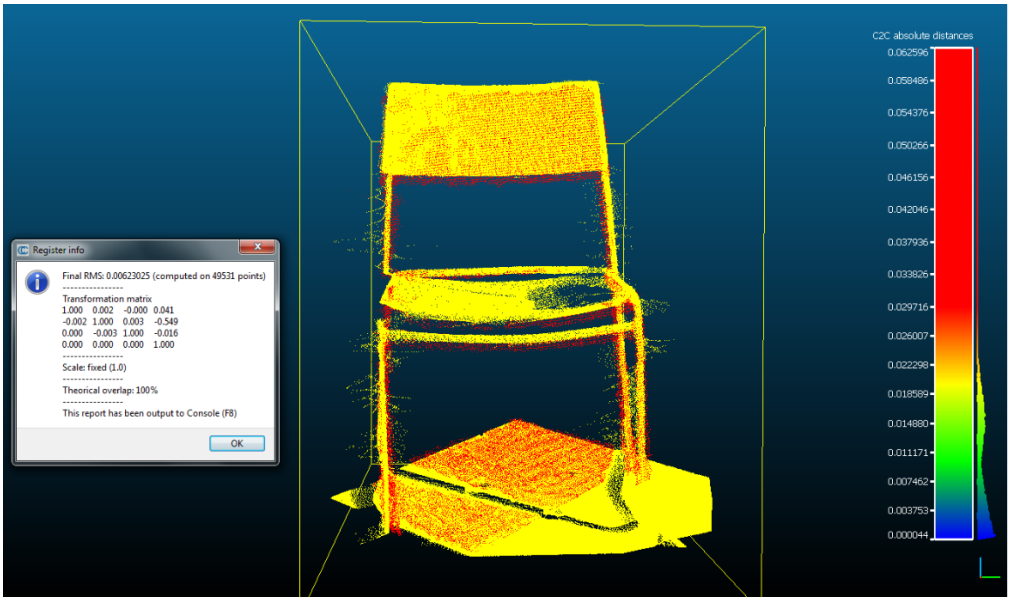
Εικόνα 7–6: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Στη παραπάνω εικόνα διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D. Αρχικά, τα δύο νέφη ευθυγραμμίστηκαν και κατόπιν έγινε κατάλληλη επιλογή ομόλογων σημείων. Το αποτέλεσμα της όλης διαδικασίας χαρακτηρίζεται από μέσο τετραγωνικό σφάλμα (εφεξής RMS) της τάξης του 1cm.



Εικόνα 7–7: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα

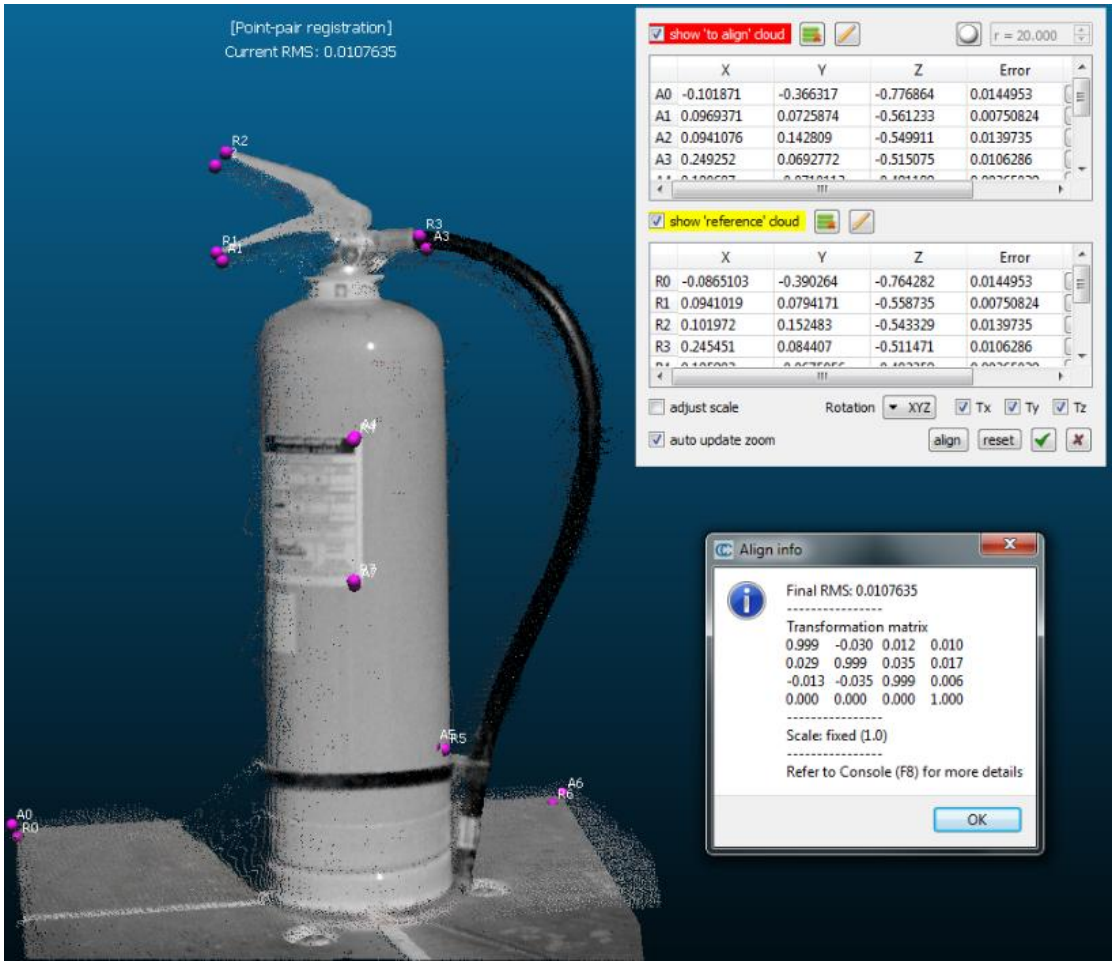
Στην εικόνα 7-7 διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, οι οποίες κυμαίνονται στη μεγάλη πλειοψηφία τους κάτω από το 1cm.



Εικόνα 7–8: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Αναμενόμενα, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει το RMS στο επίπεδο των 6mm (Εικόνα 7-8).

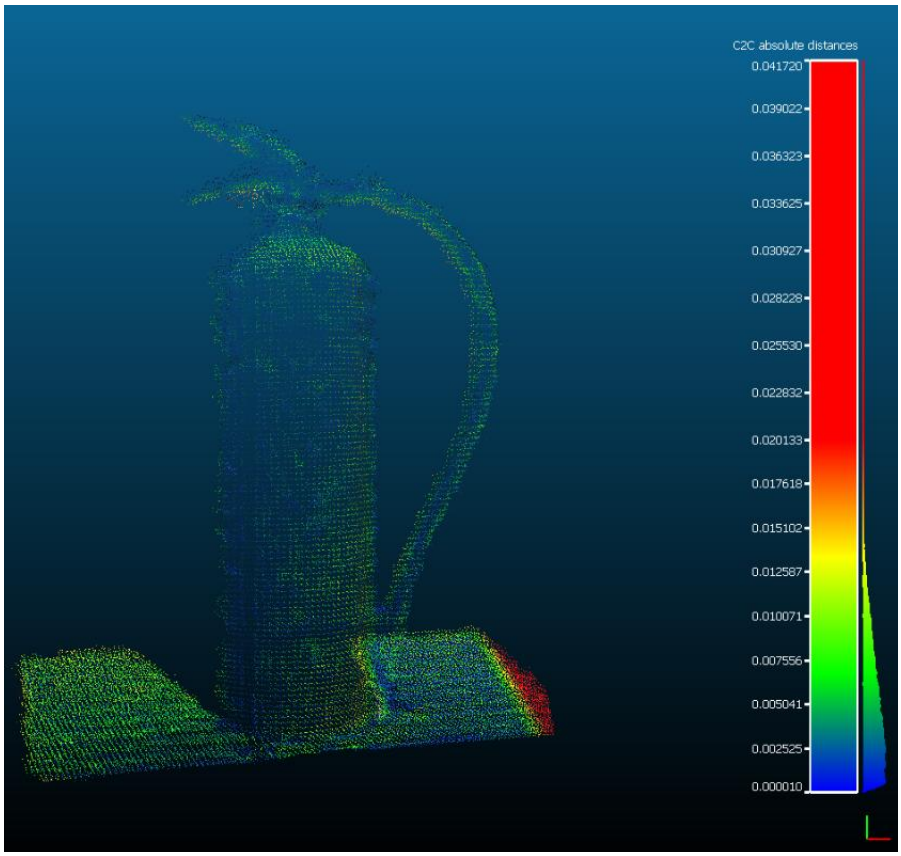
Πυροσβεστήρας



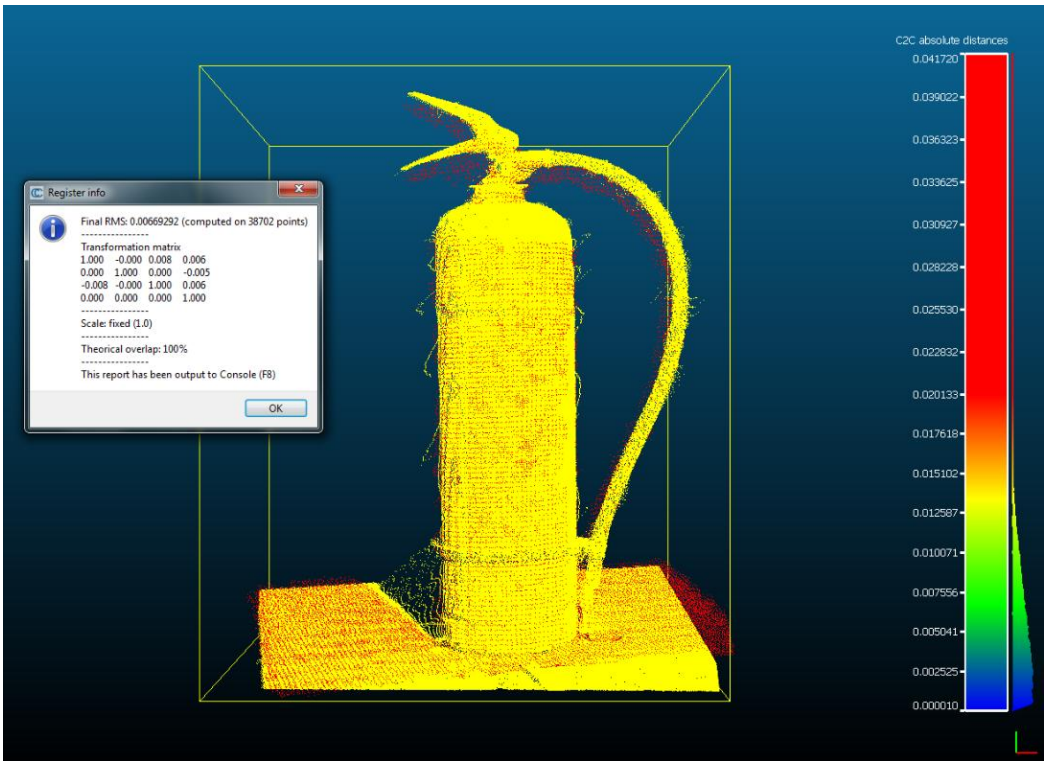
Εικόνα 7–9: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Όμοια, στη παραπάνω εικόνα, διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, που αφορούν στον πυροσβεστήρα. Η κατάλληλη επιλογή ομόλογων σημείων στα δύο νέφη, έχει ως αποτέλεσμα ένα RMS της τάξης του 1cm.

Παρακάτω (εικόνα 7-10) διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το πρόγραμμα και οι οποίες, στη πλειοψηφία τους, κυμαίνονται κάτω από το 1cm



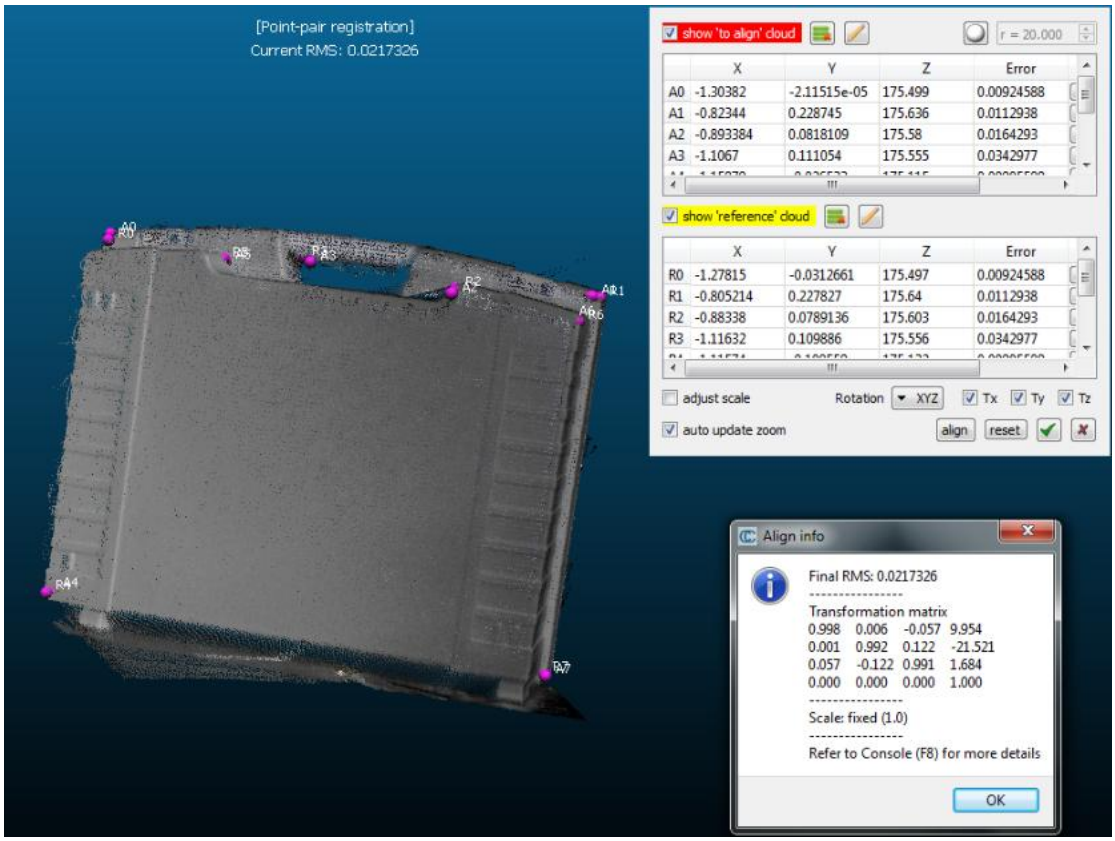
Εικόνα 7–10: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–11: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Εξίσου αναμενόμενα, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει το RMS στο επίπεδο των 7mm (Εικόνα 7-11).

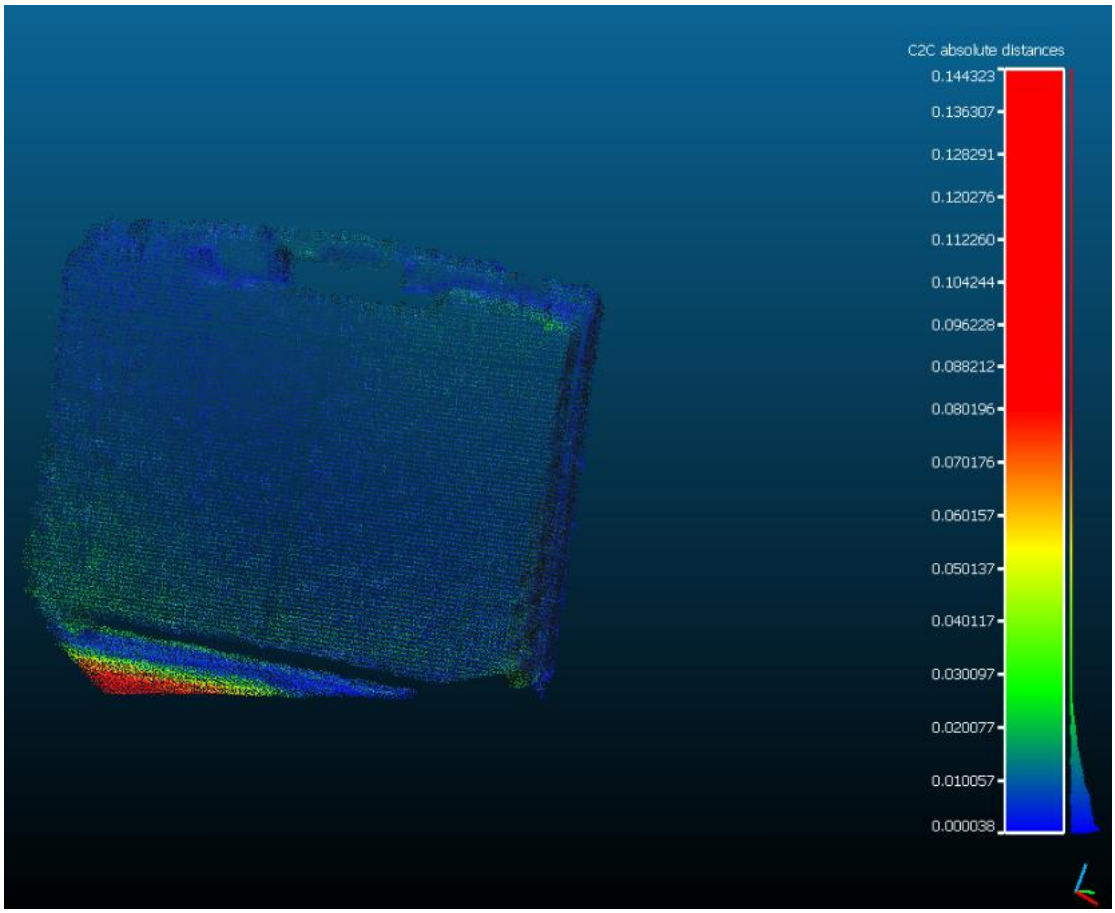
Θήκη GPS



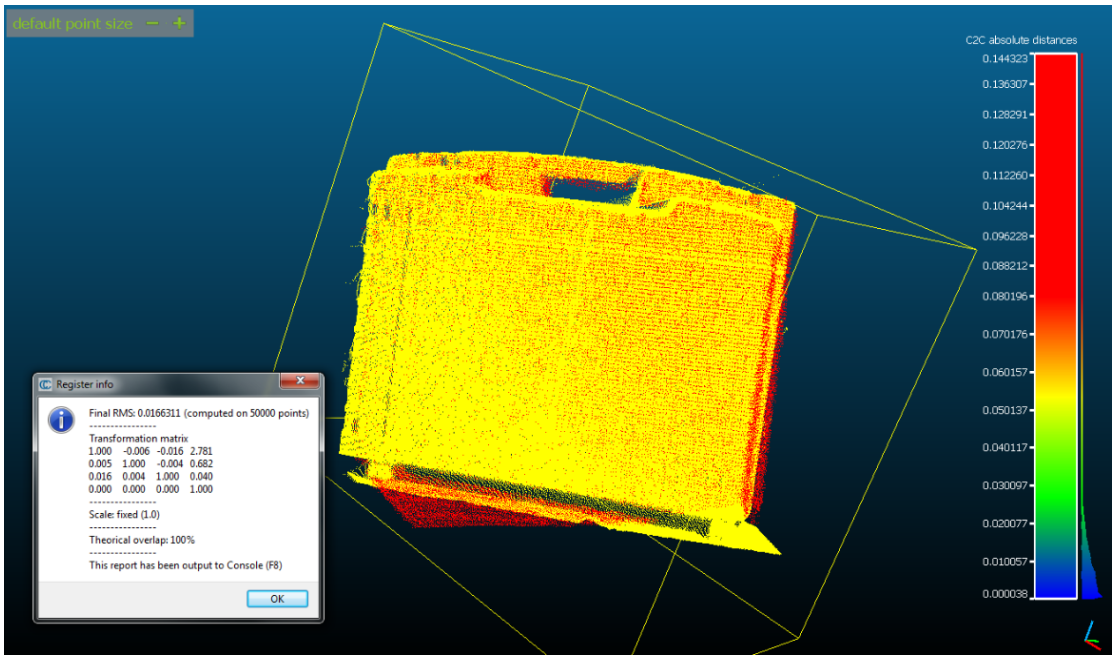
Εικόνα 7–12: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Στη παραπάνω εικόνα, διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, που αφορούν στη θήκη GPS. Κατόπιν κατάλληλης επιλογής ομόλογων σημείων στα δύο νέφη, το παραγόμενο μέσο τετραγωνικό σφάλμα είναι της τάξης των 2cm.

Παρακάτω (εικόνα 7-13), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το λογισμικό και οι οποίες, στη μεγάλη πλειοψηφία τους, κυμαίνονται κάτω από τα 2cm.



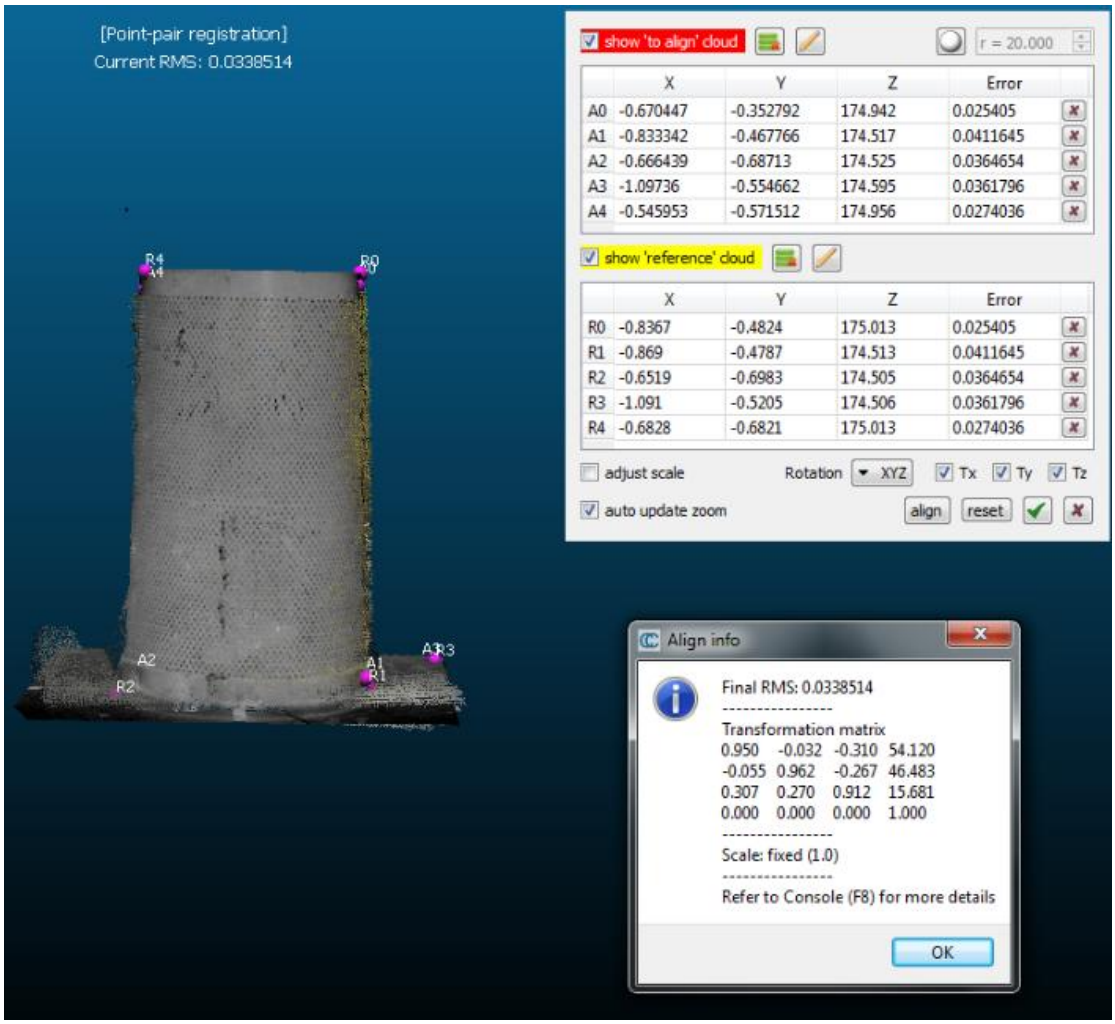
Εικόνα 7–13: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–14: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Όμοια, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει το RMS στο επίπεδο του 1,5cm (Εικόνα 7-14).

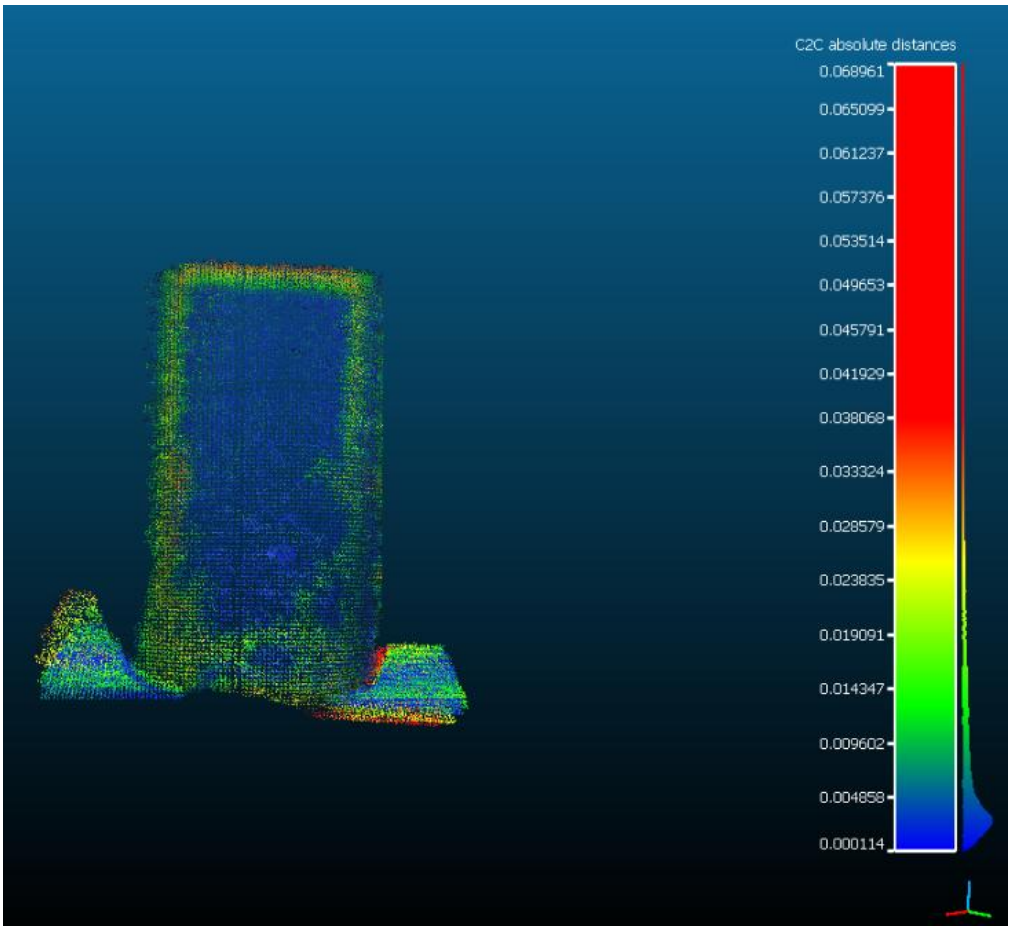
Κάδος απορριμμάτων



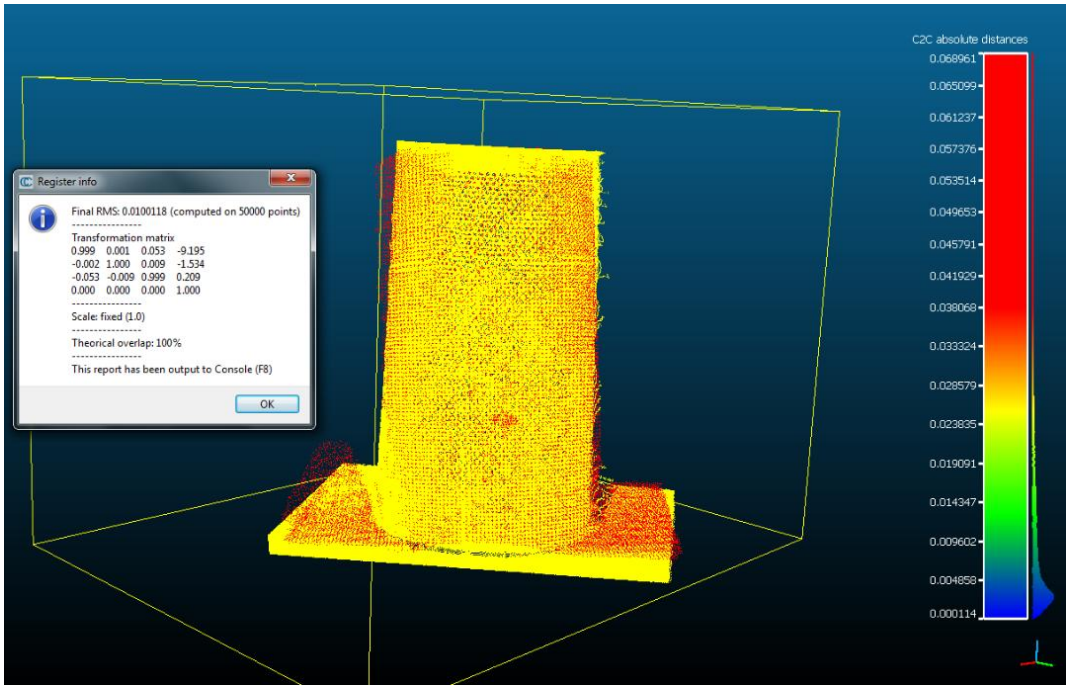
Εικόνα 7–15: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Όπως διακρίνεται στη παραπάνω εικόνα, η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, με κατάλληλη επιλογή ομόλογων σημείων, χαρακτηρίζεται από μέσο τετραγωνικό σφάλμα (RMS) της τάξης των 3cm.

Παρακάτω (εικόνα 7-16), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το λογισμικό και οι οποίες, στη μεγάλη πλειοψηφία τους, κυμαίνονται κάτω από τα 5mm.



Εικόνα 7–16: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα

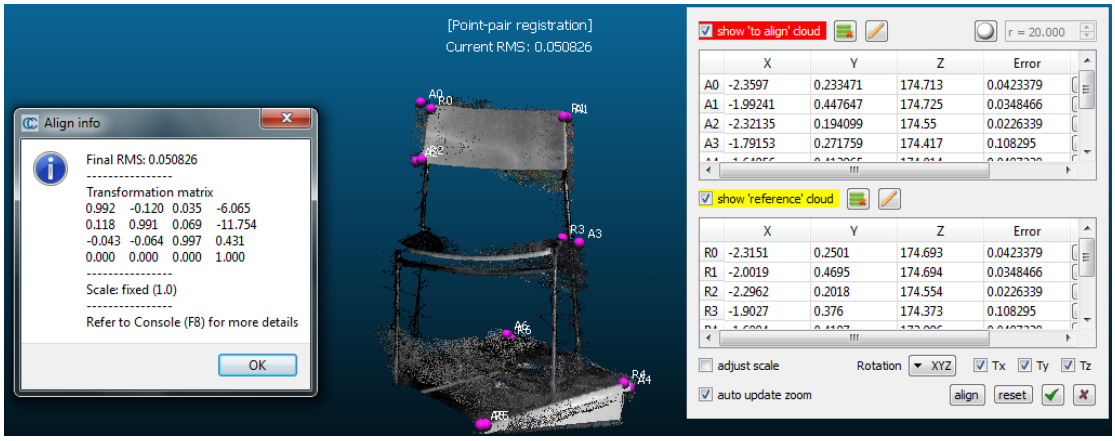


Εικόνα 7–17: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Στην ίδια λογική, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει το RMS στο επίπεδο του 1cm (Εικόνα 7-14).

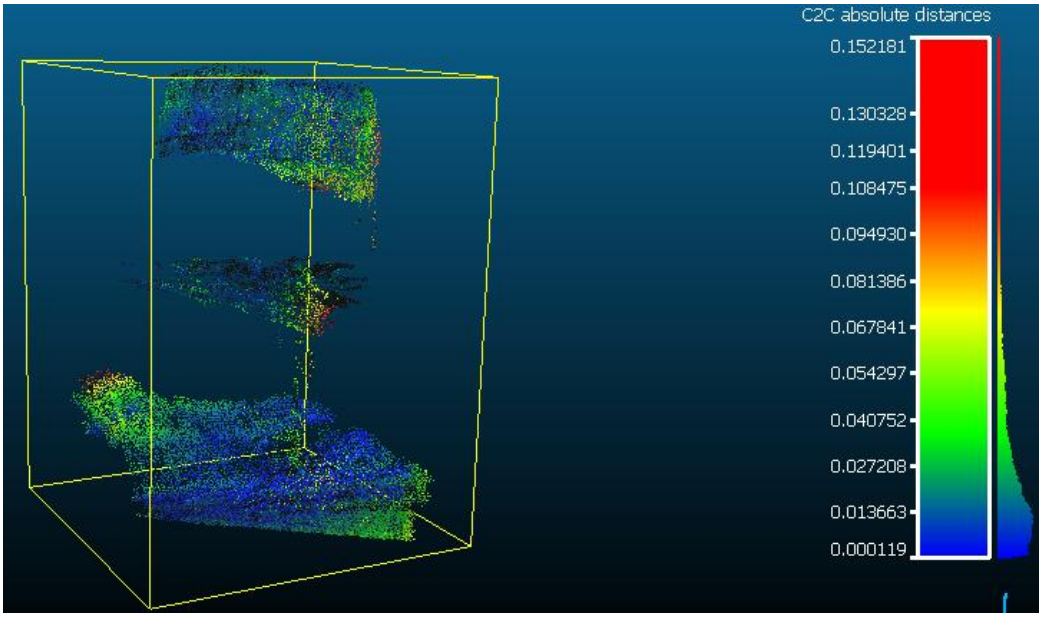
7.4.1.2 Απόσταση 2m

Καρέκλα



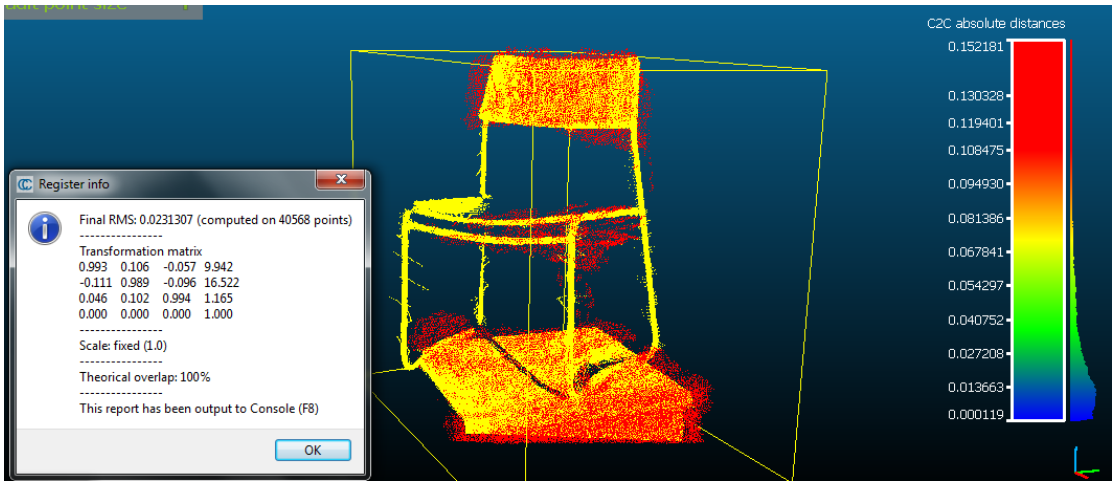
Εικόνα 7–18: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Όπως διακρίνεται στη παραπάνω εικόνα, η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, με κατάλληλη επιλογή ομόλογων σημείων, χαρακτηρίζεται από μέσο τετραγωνικό σφάλμα (RMS) της τάξης του 5cm, πενταπλάσιο αυτού της απόστασης 1m. Είναι επίσης χαρακτηριστικό πως τα στελέχη της καρέκλας διαμέτρου λόγων εκατοστών, δεν έχουν καταγραφεί.



Εικόνα 7–19: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα

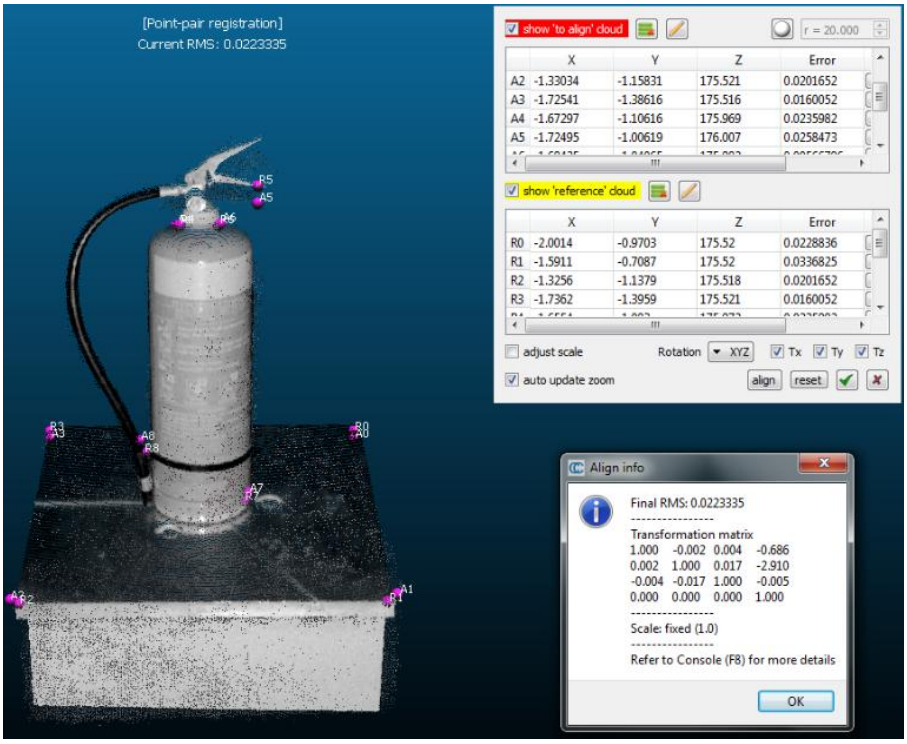
Στην εικόνα 7-19 διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, οι οποίες κυμαίνονται στη μεγάλη πλειοψηφία τους κάτω από το 4cm, έναντι 1cm σε απόσταση 1m.



Εικόνα 7–20: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Αναμενόμενα, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει το RMS στο επίπεδο των 2cm (Εικόνα 7-20).

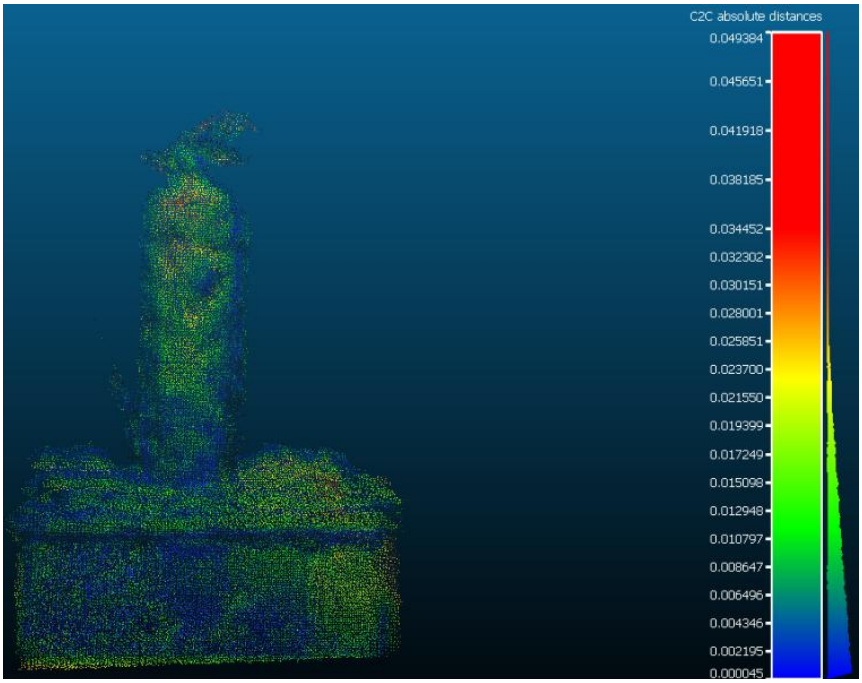
Πυροσβεστήρας



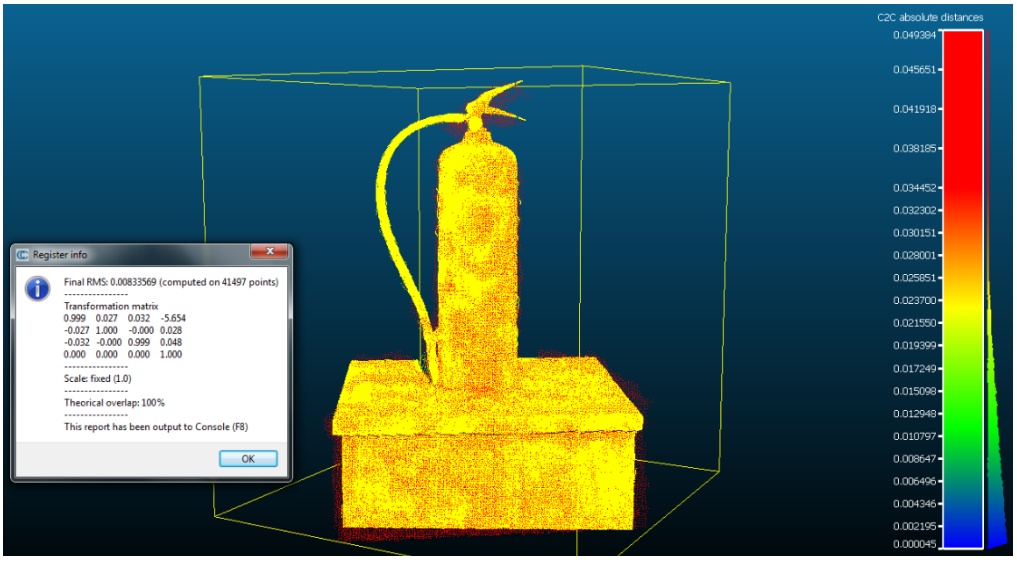
Εικόνα 7–21: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Όμοια, στη παραπάνω εικόνα, διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, που αφορούν στον πυροσβεστήρα και σε απόσταση 2m. Η κατάλληλη επιλογή ομόλογων σημείων στα δύο νέφη, έχει ως αποτέλεσμα ένα RMS της τάξης των 2cm, υπερδιπλάσια αυτού στο 1m.

Παρακάτω (εικόνα 7-22) διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το πρόγραμμα και οι οποίες κυμαίνονται ως επί το πλείστον, κάτω από το 2cm, ενώ το στέλεχος (σωλήνας) του πυροσβεστήρα δεν έχει καταγραφεί, όμοια με το πλαίσιο της καρέκλας.



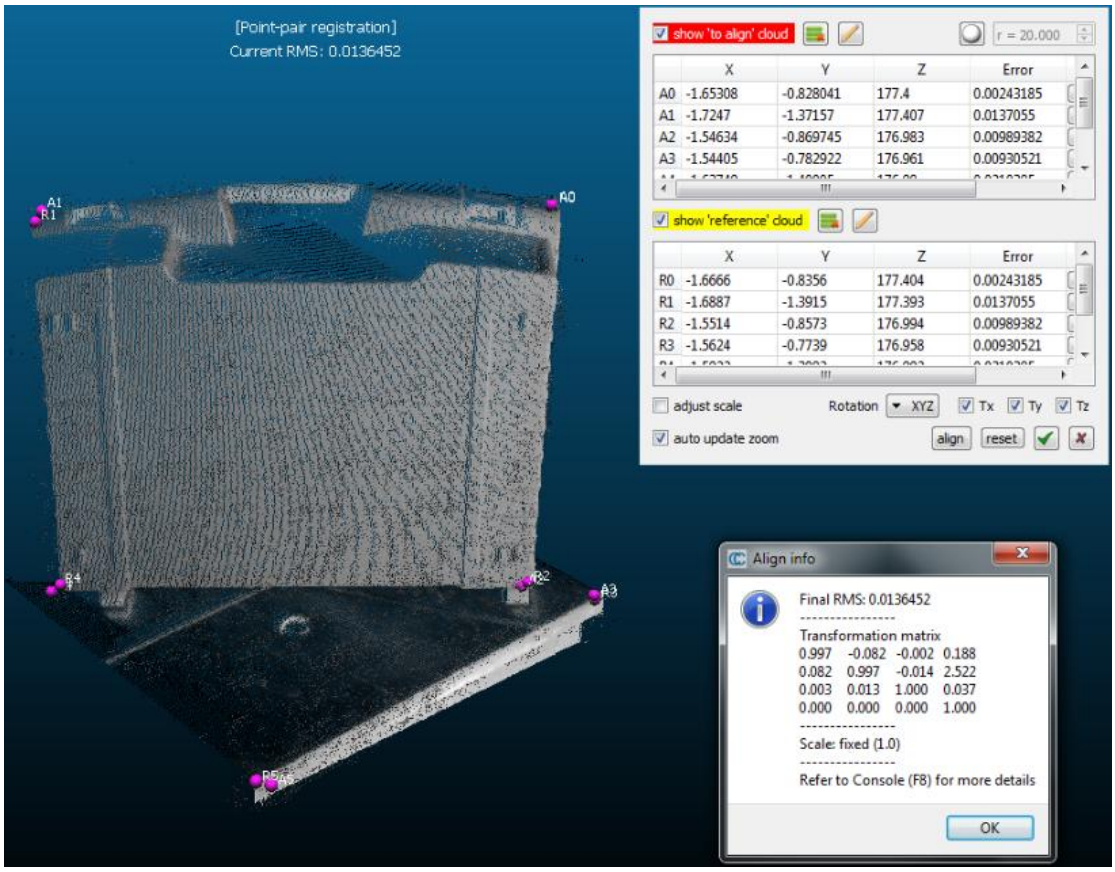
Εικόνα 7–22: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–23: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Εξίσου αναμενόμενα, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει το RMS στο επίπεδο του 1cm (Εικόνα 7-23).

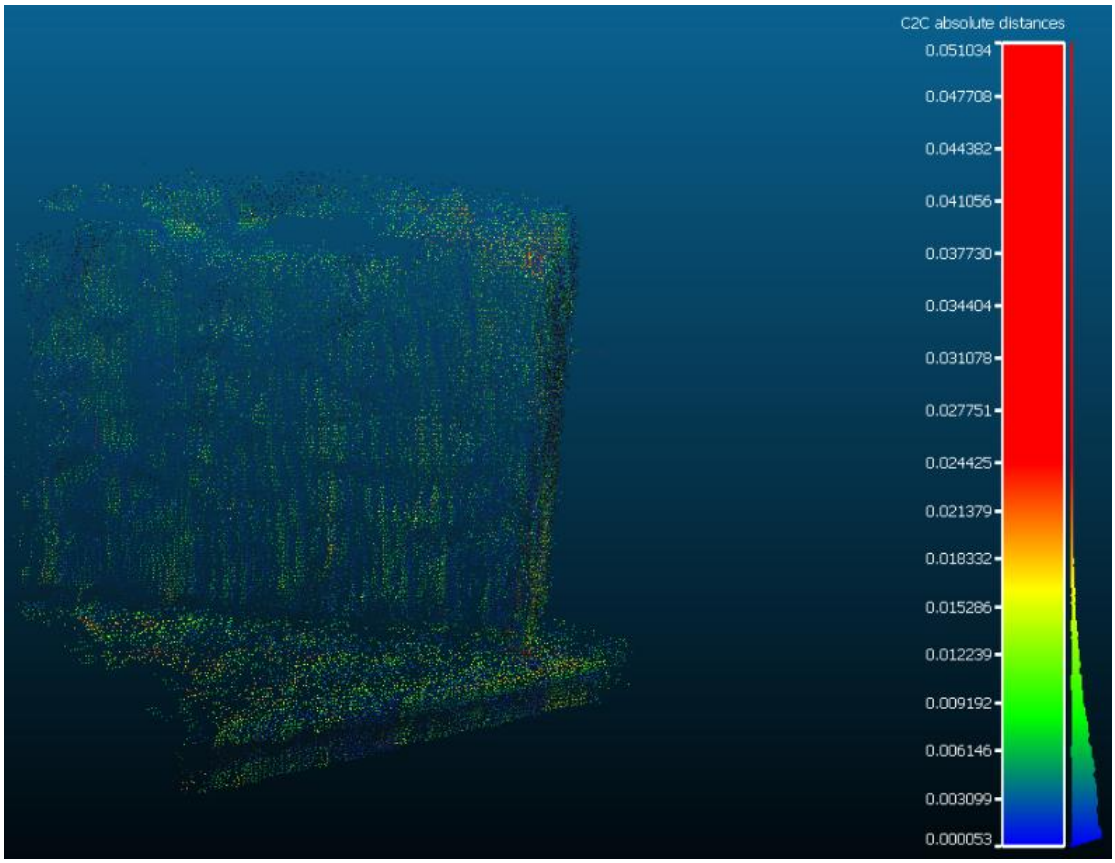
Θήκη GPS



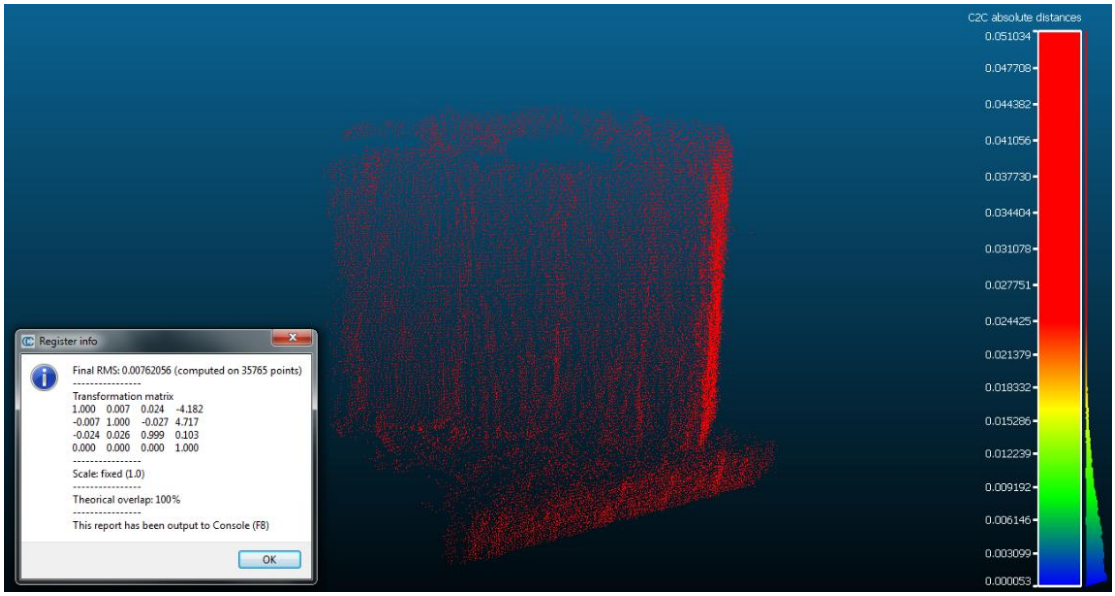
Εικόνα 7–24: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Στη παραπάνω εικόνα, διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, που αφορούν στη θήκη GPS. Κατόπιν κατάλληλης επιλογής ομόλογων σημείων στα δύο νέφη, το παραγόμενο μέσο τετραγωνικό σφάλμα είναι μεν μεγαλύτερο του 1cm, αλλά αισθητά μικρότερο αυτού στο 1m.

Παρακάτω (εικόνα 7-25), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το λογισμικό και οι οποίες, στη μεγάλη πλειοψηφία τους, κυμαίνονται κάτω από τα 1cm.



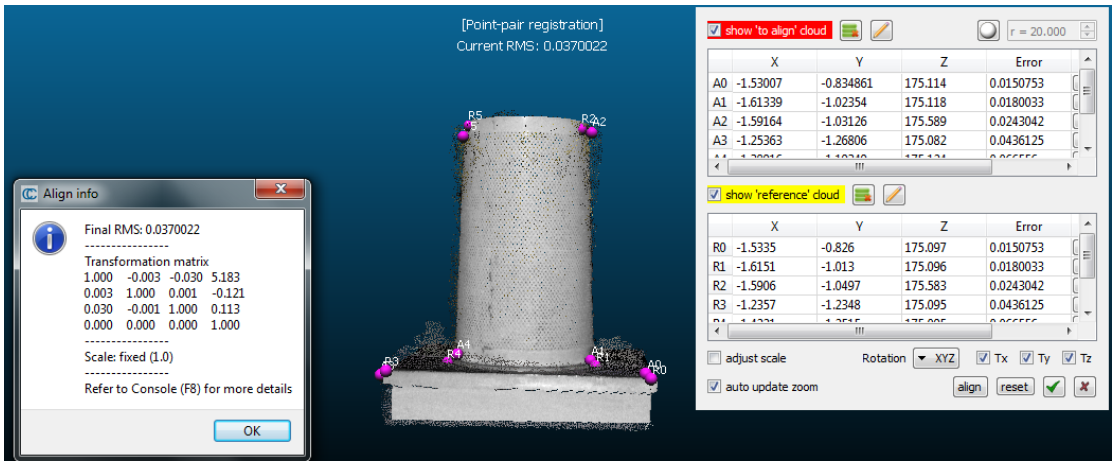
Εικόνα 7–25: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–26: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Όμοια, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, υποδιπλασιάζει το RMS στο επίπεδο των 7mm (Εικόνα 7-26).

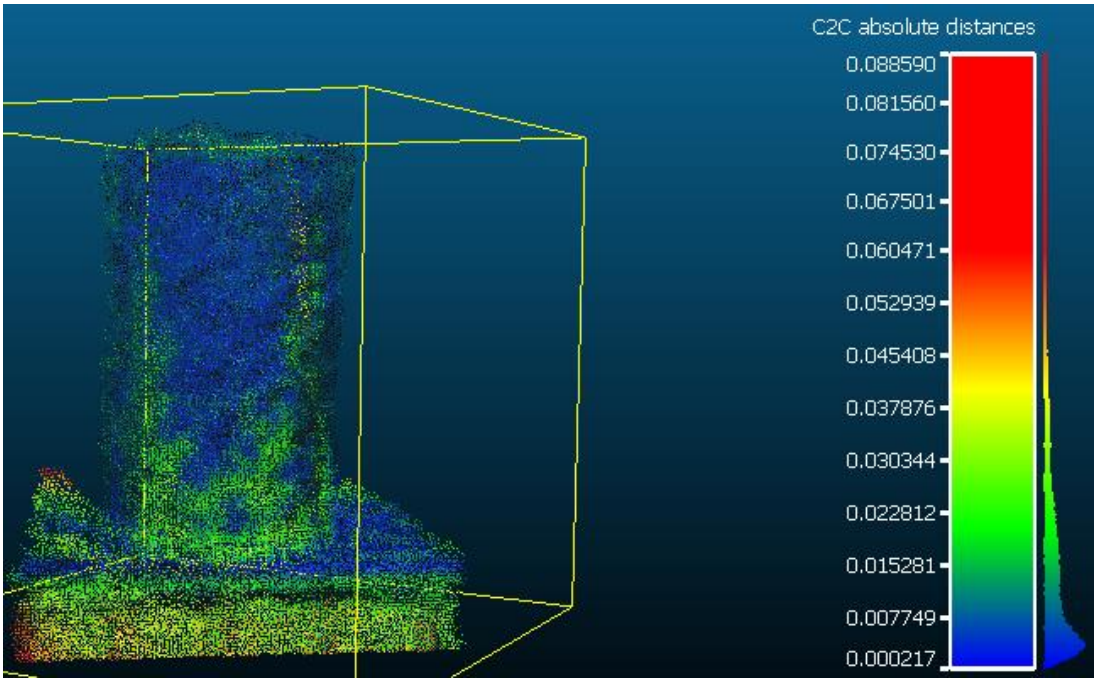
Κάδος απορριμμάτων



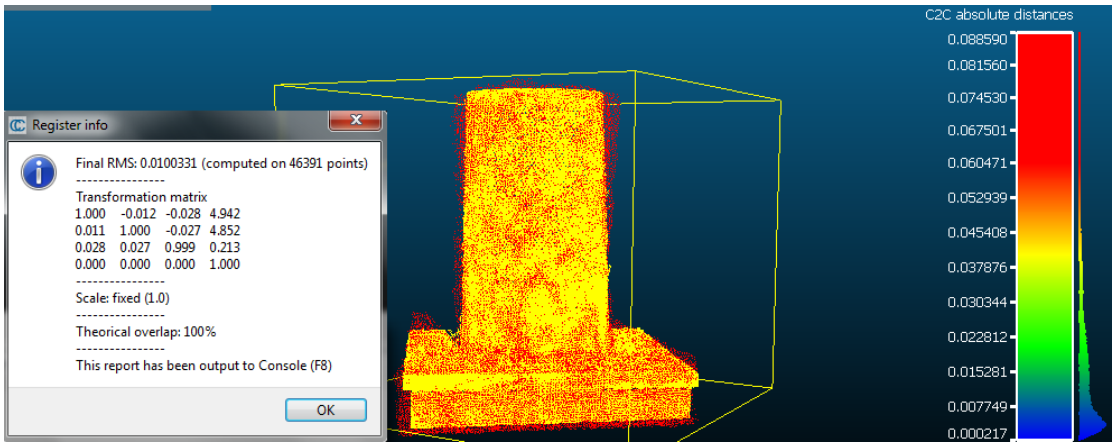
Εικόνα 7–27: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Όπως διακρίνεται στη παραπάνω εικόνα, η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, με κατάλληλη επιλογή ομόλογων σημείων, χαρακτηρίζεται από μέσο τετραγωνικό σφάλμα (RMS) της τάξης των 4cm, λίγο μεγαλύτερο αυτού στο 1m.

Παρακάτω (εικόνα 7-28), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το λογισμικό και οι οποίες, στη μεγάλη πλειοψηφία τους, κυμαίνονται κάτω από τα 1cm, αισθητά μεγαλύτερες αυτών στο 1m.



Εικόνα 7–28: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα

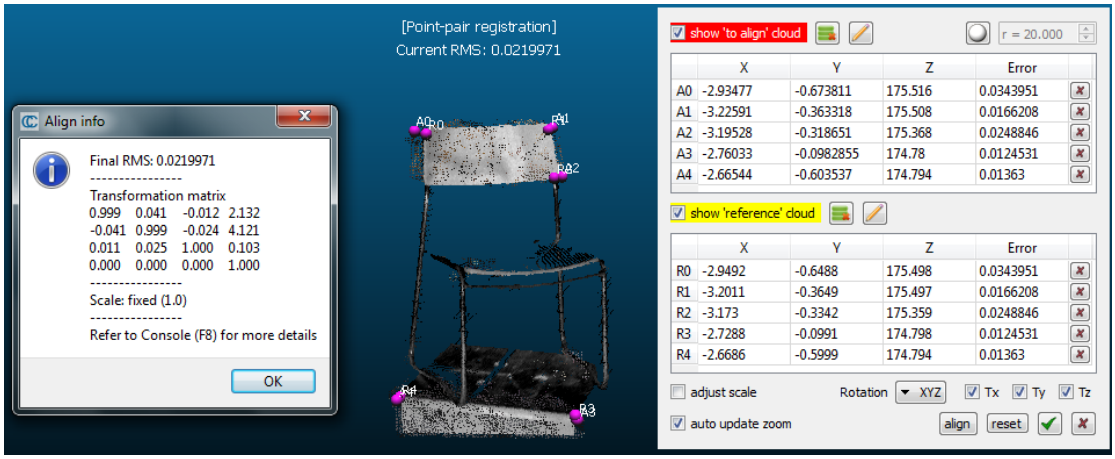


Εικόνα 7–29: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Στην ίδια λογική, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει το RMS στο επίπεδο του 1cm (Εικόνα 7-29).

7.4.1.3 Απόσταση 3m

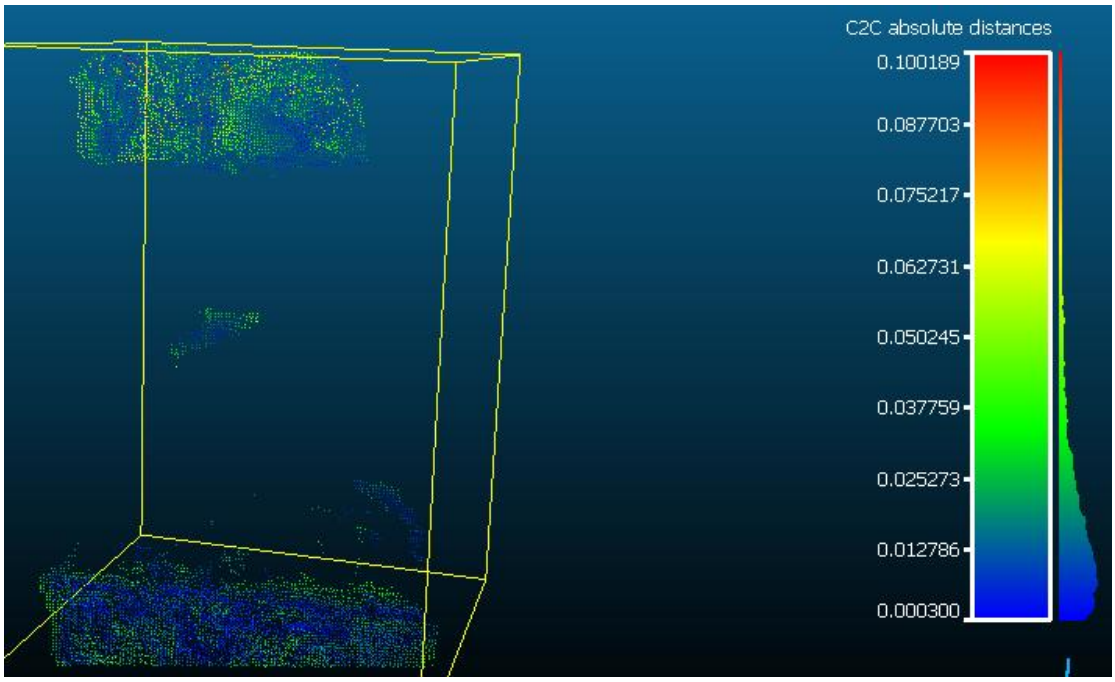
Καρέκλα



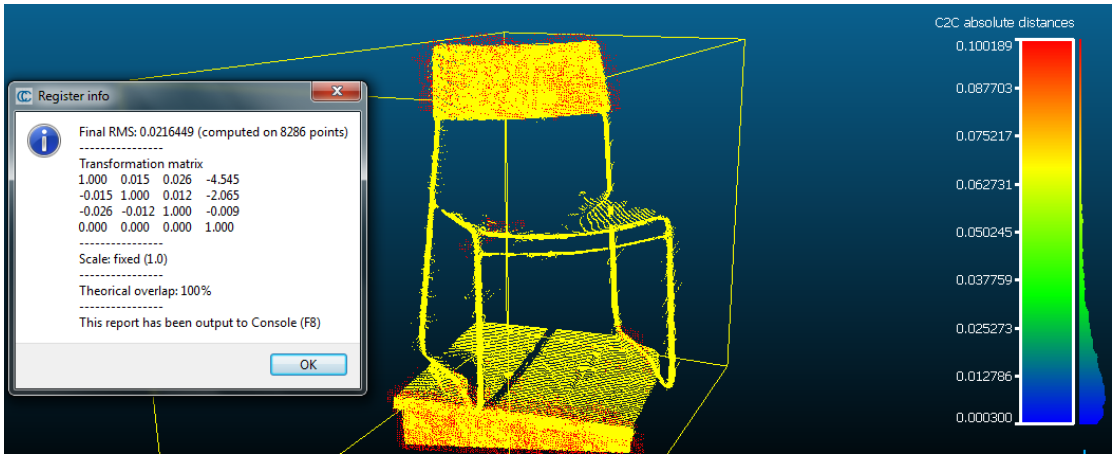
Εικόνα 7–30: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Όπως διακρίνεται στη παραπάνω εικόνα, η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, με κατάλληλη επιλογή ομόλογων σημείων, χαρακτηρίζεται από μέσο τετραγωνικό σφάλμα (RMS) της τάξης των 2cm, αισθητά μικρότερου αυτού της απόστασης των 2m. Όμοια, το πλαίσιο της καρέκλας δεν έχει καταγραφεί.

Στην εικόνα που ακολουθεί, διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, οι οποίες κυμαίνονται στη μεγάλη πλειοψηφία τους κάτω από το 4cm, όμοια με τα αποτελέσματα που προηγήθηκαν.



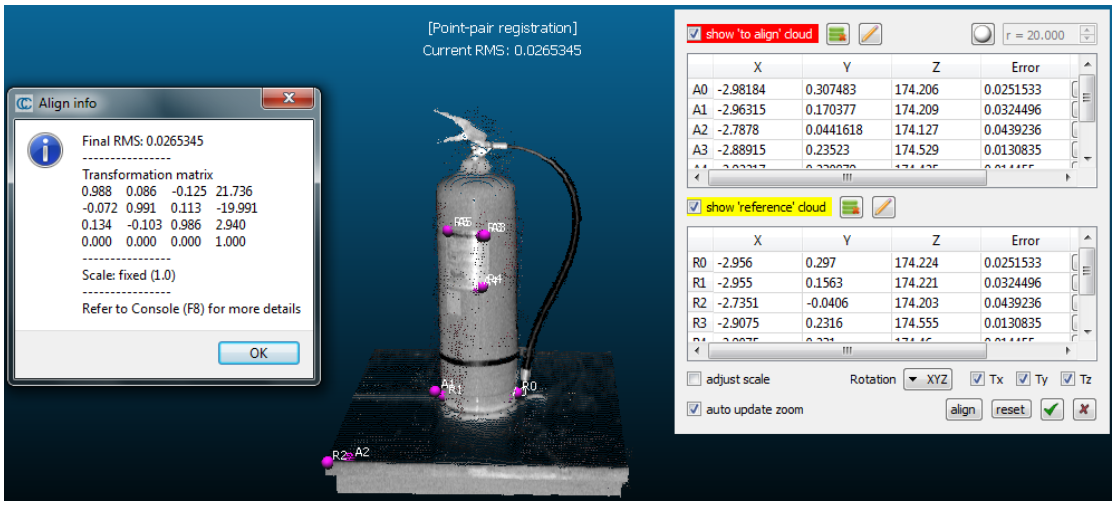
Εικόνα 7–31: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–32: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Εν προκειμένω, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, διατηρεί το RMS στο επίπεδο των 2cm (Εικόνα 7-32).

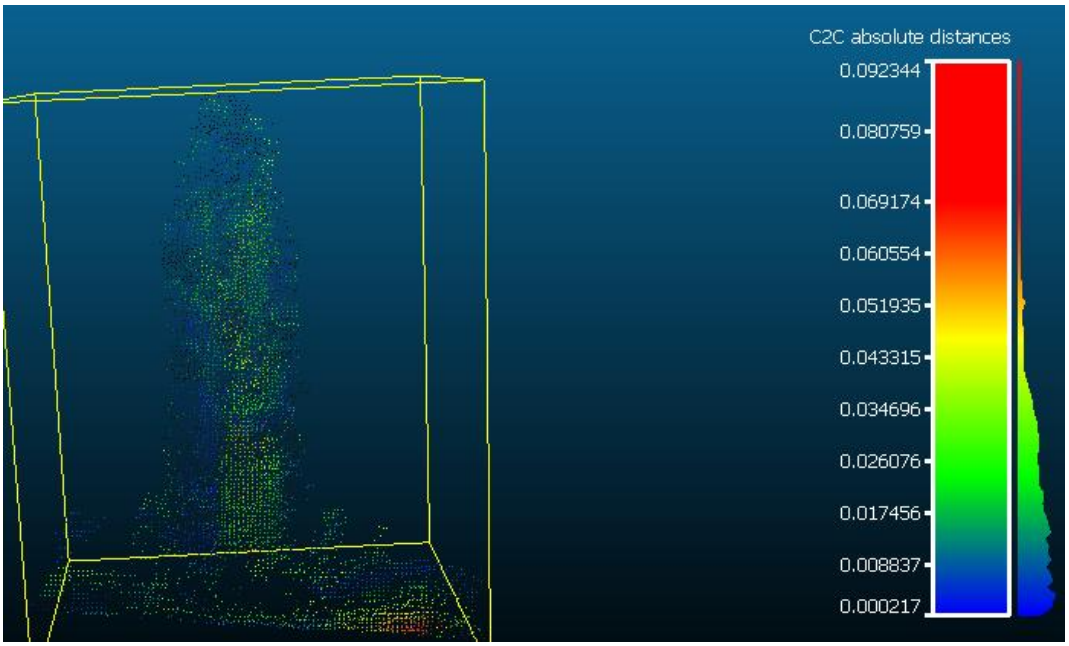
Πυροσβεστήρας



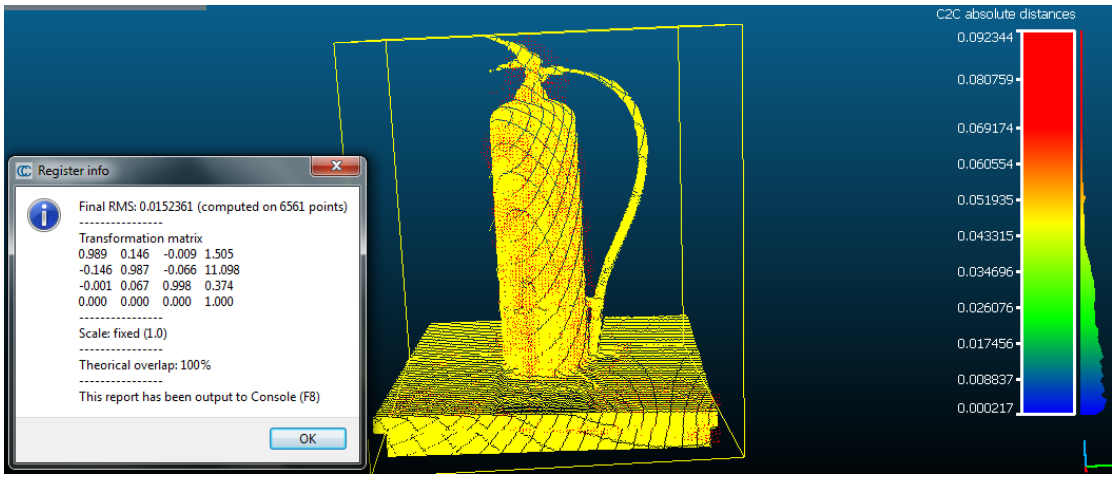
Εικόνα 7–33: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Όμοια, στη παραπάνω εικόνα, διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, που αφορούν στον πυροσβεστήρα και σε απόσταση 3m. Η κατάλληλη επιλογή ομόλογων σημείων στα δύο νέφη, έχει ως αποτέλεσμα ένα RMS της τάξης των 2,5cm, ελαφρώς μεγαλύτερο αυτού στα 2m.

Παρακάτω (εικόνα 7-34) διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το πρόγραμμα και οι οποίες κυμαίνονται, ως επί το πλείστον, κάτω από το 3cm, ενώ το στέλεχος (σωλήνας) του πυροσβεστήρα δεν έχει καταγραφεί, όμοια με το πλαίσιο της καρέκλας.



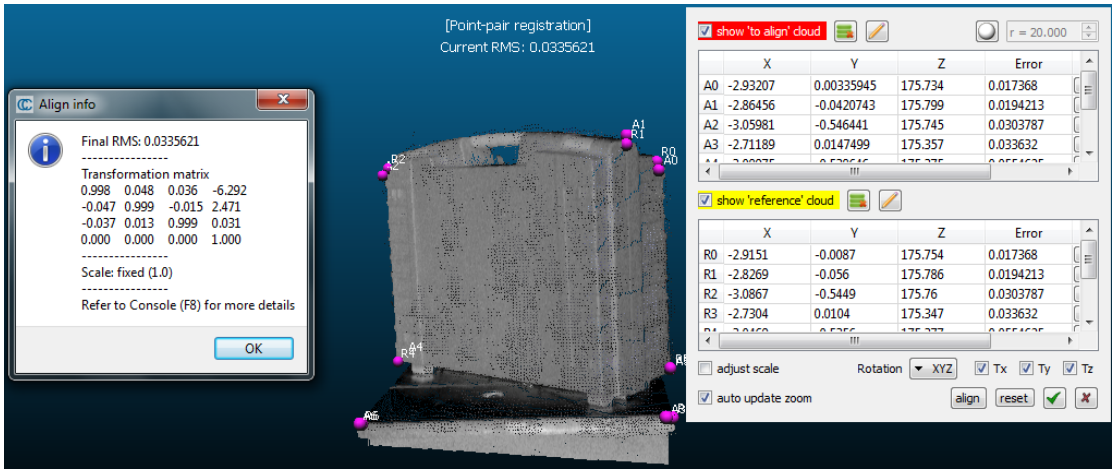
Εικόνα 7–34: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–35: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Εξίσου αναμενόμενα, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει το RMS στο επίπεδο του 1,5cm (Εικόνα 7-35).

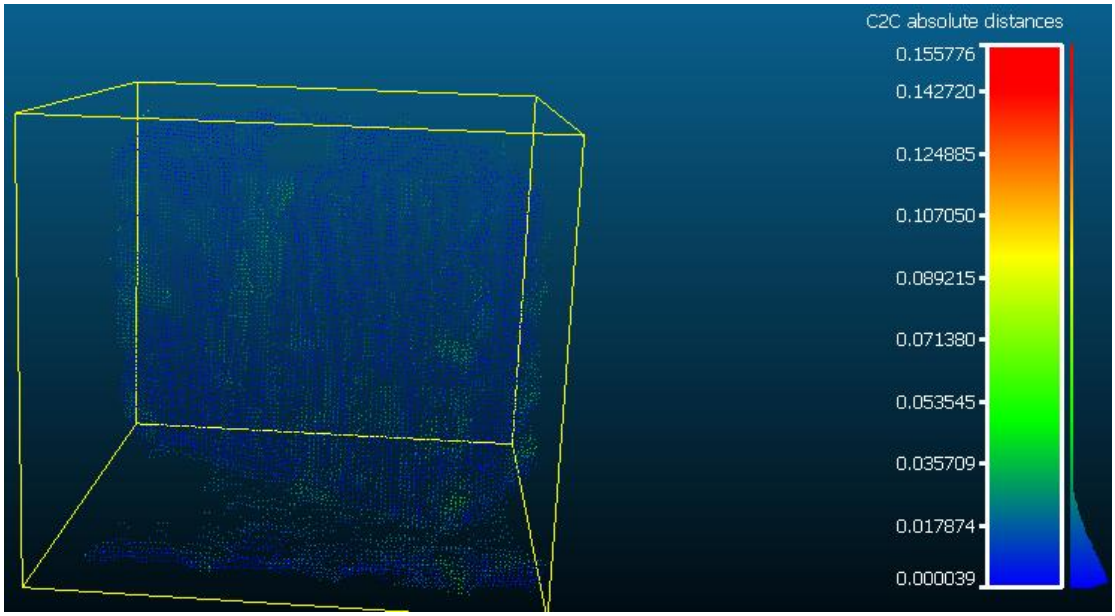
Θήκη GPS



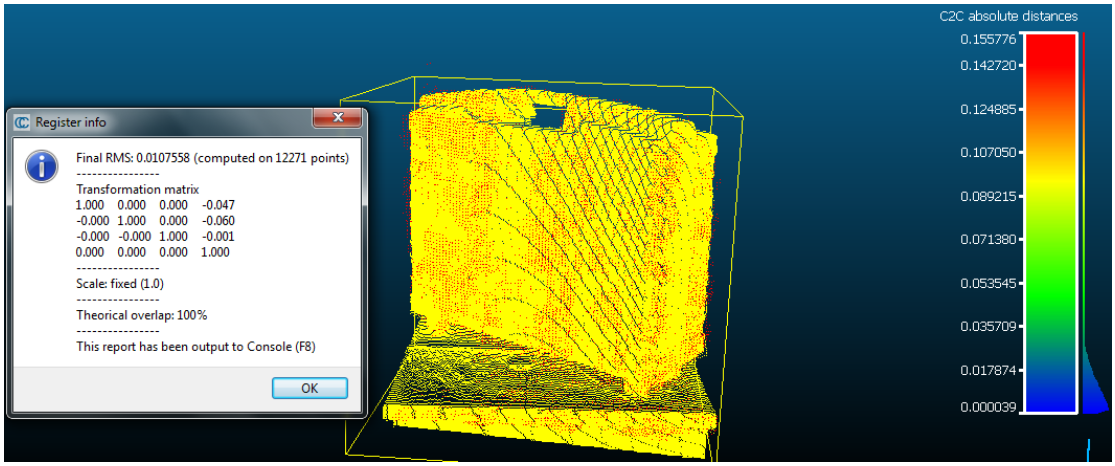
Εικόνα 7–36: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Στη παραπάνω εικόνα, διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, που αφορούν στη θήκη GPS. Κατόπιν κατάλληλης επιλογής ομόλογων σημείων στα δύο νέφη, το παραγόμενο μέσο τετραγωνικό σφάλμα είναι της τάξης των 3cm, σχεδόν τριπλάσιο του αντίστοιχου σφάλματος στην απόσταση των 2m.

Παρακάτω (εικόνα 7-37), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το λογισμικό και οι οποίες, στη μεγάλη πλειοψηφία τους, κυμαίνονται κάτω από τα 2cm.



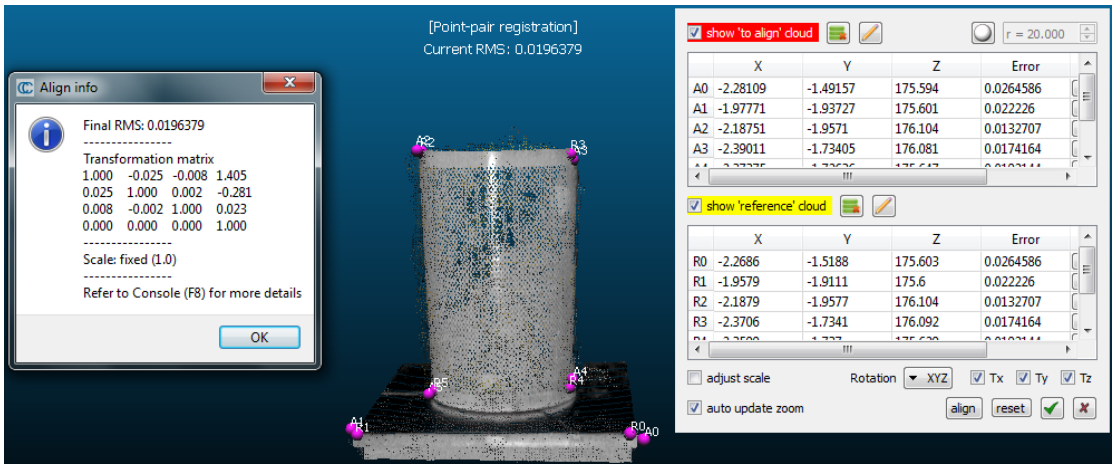
Εικόνα 7–37: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–38: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Όμοια, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, υποδιπλασιάζει το RMS στο επίπεδο των 1cm (Εικόνα 7-38).

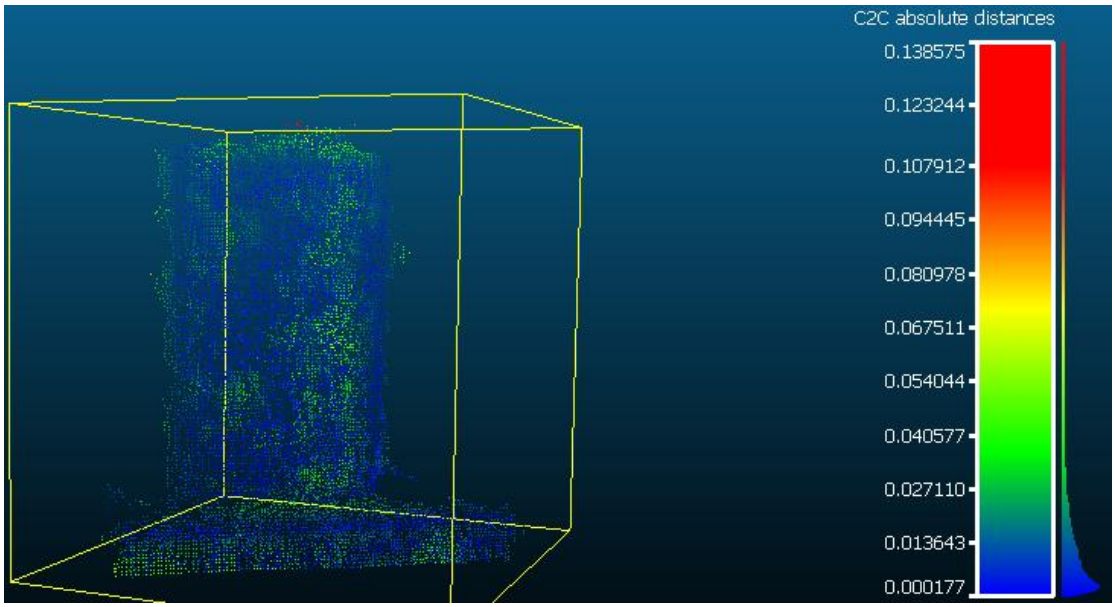
Κάδος απορριμμάτων



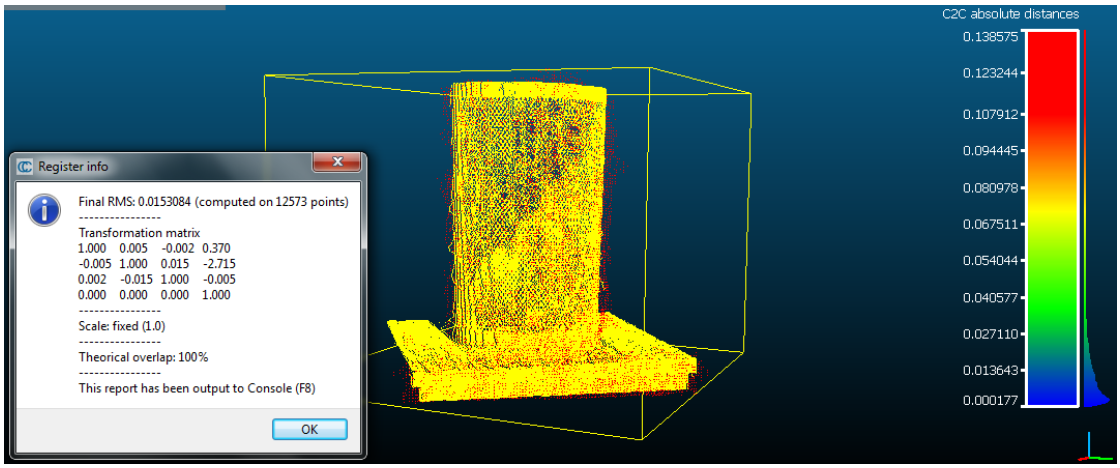
Εικόνα 7–39: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Όπως διακρίνεται στη παραπάνω εικόνα, η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect και του σαρωτή Faro Focus 3D, με κατάλληλη επιλογή ομόλογων σημείων, χαρακτηρίζεται από μέσο τετραγωνικό σφάλμα (RMS) της τάξης των 2cm, υποδιπλάσιο του αντίστοιχου σφάλματος σε απόσταση 2m.

Παρακάτω (εικόνα 7-40), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το λογισμικό και οι οποίες, στη μεγάλη πλειοψηφία τους, κυμαίνονται κάτω από τα 2cm, πρακτικά διπλάσιες αυτών στα 2m.



Εικόνα 7–40: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–41: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Στην ίδια λογική, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει το RMS στο επίπεδο του 1,5cm (Εικόνα 7-41).

Σημειώνεται πως με βάση τα αποτελέσματα που προέκυψαν από τη μέτρηση στις τρεις πρώτες θέσεις, η παράθεση των αποτελεσμάτων για απόσταση 4m κρίνεται μη σκόπιμη δεδομένου ότι η ποιότητα της καταγραφής είναι τέτοια που δεν εξυπηρετεί τον σκοπό της παρούσης, πόσο μάλλον συγκρινόμενη με φωτογραμμετρικές μεθόδους.

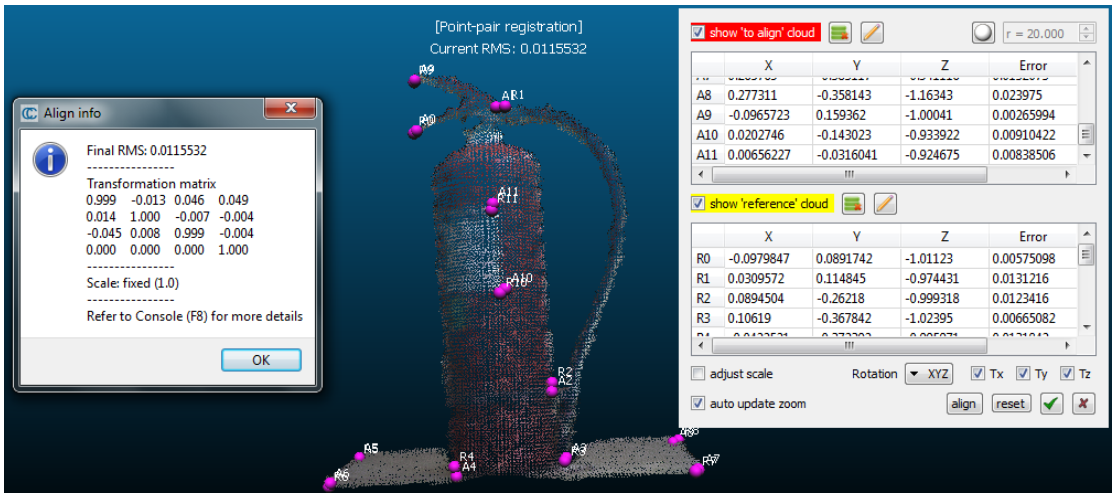
7.4.2 Με κριτήριο τον φωτισμό

Το υποκεφάλαιο αυτό αναλώνεται στον προσδιορισμό της επίδρασης του σχετικού φωτισμού στη ποιότητα του νέφους σημείων που παράγει ο αισθητήρας Kinect. Τα συγκρινόμενα νέφη αφορούν δύο αντικείμενα (πυροσβεστήρας, θήκη GPS) σε τρεις αποστάσεις αποτυπωμένα μόνον από το Kinect, τόσο σε συνθήκες φυσικού όσο και σε συνθήκες τεχνητού φωτισμού.

Σημειώνεται ότι ως νέφος αναφοράς, για κάθε σύγκριση χρησιμοποιήθηκε αυτό που κατεγράφη με συνθήκες τεχνητού φωτισμού.

7.4.2.1 Απόσταση 1m

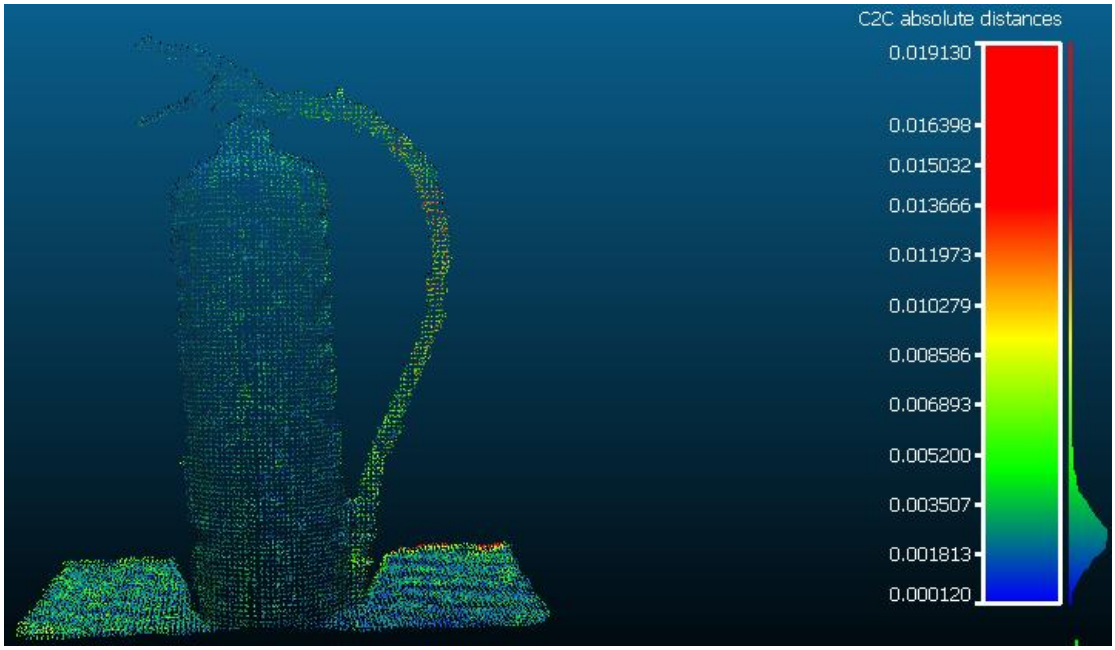
Πυροσβεστήρας



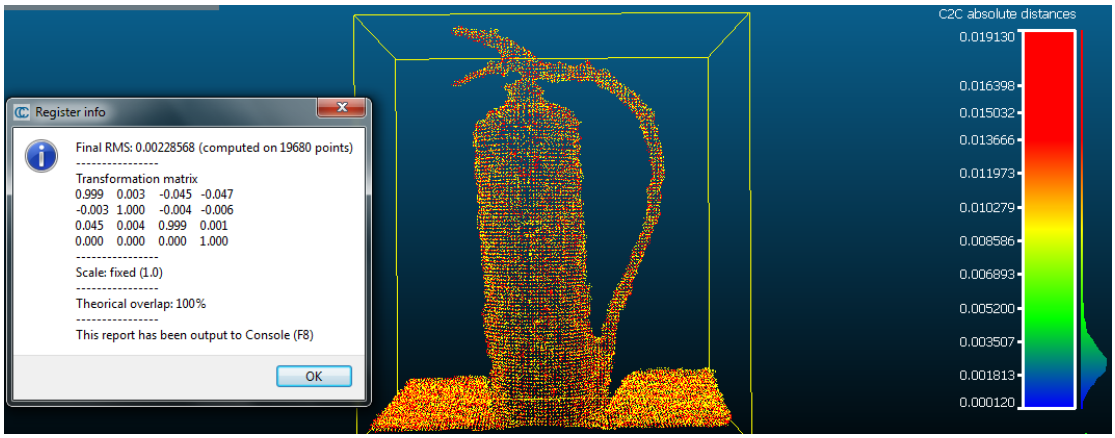
Εικόνα 7–42: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Όπως διακρίνεται στη παραπάνω εικόνα, η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect για διαφορετικές συνθήκες φωτισμού, με κατάλληλη επιλογή ομόλογων σημείων, χαρακτηρίζεται από μέσο τετραγωνικό σφάλμα (RMS) της τάξης του 1cm.

Στην εικόνα που ακολουθεί, διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, οι οποίες κυμαίνονται στη μεγάλη πλειοψηφία τους κάτω από το 3mm.



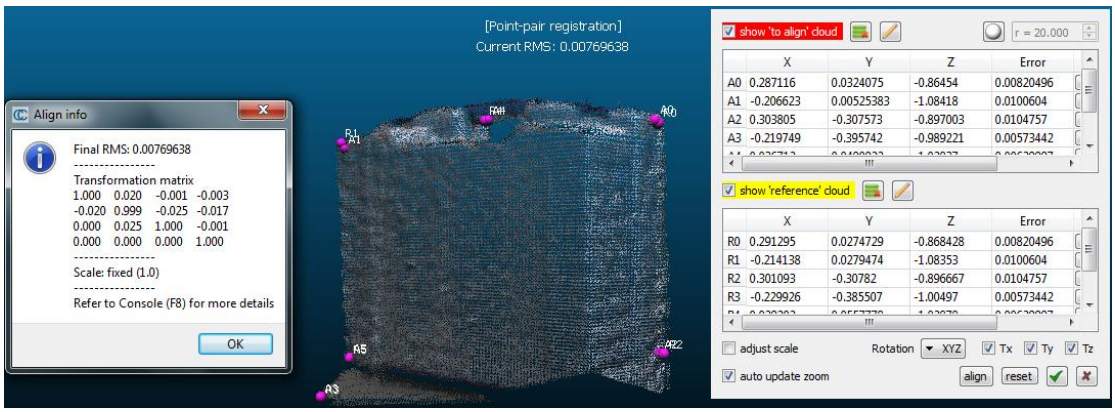
Εικόνα 7–43: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–44: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Αναμενόμενα, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει θεαματικά το RMS στο επίπεδο των 2mm (Εικόνα 7-44).

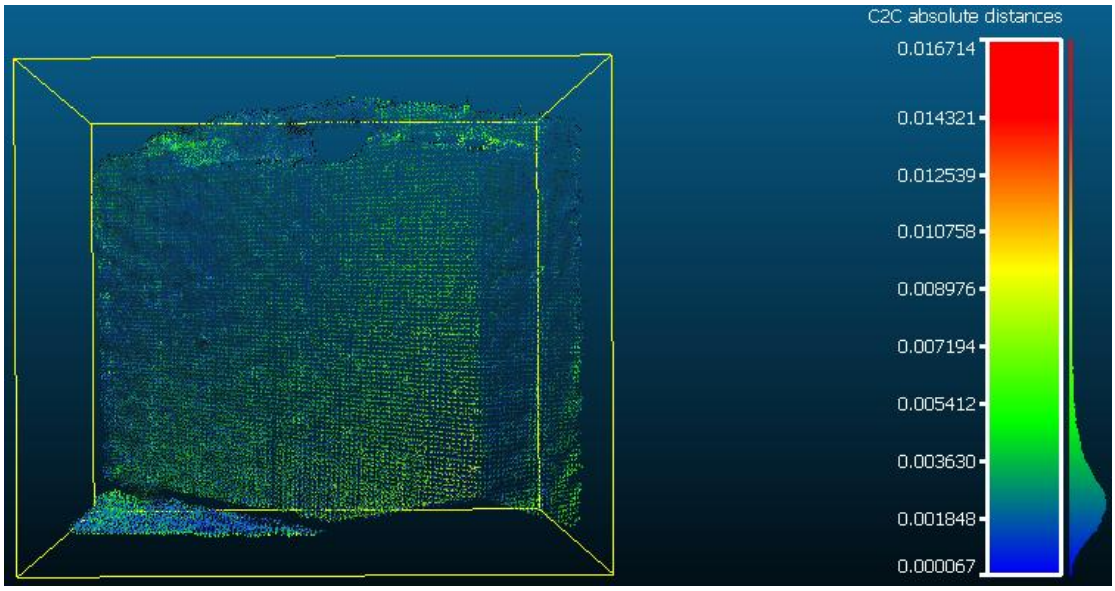
Θήκη GPS



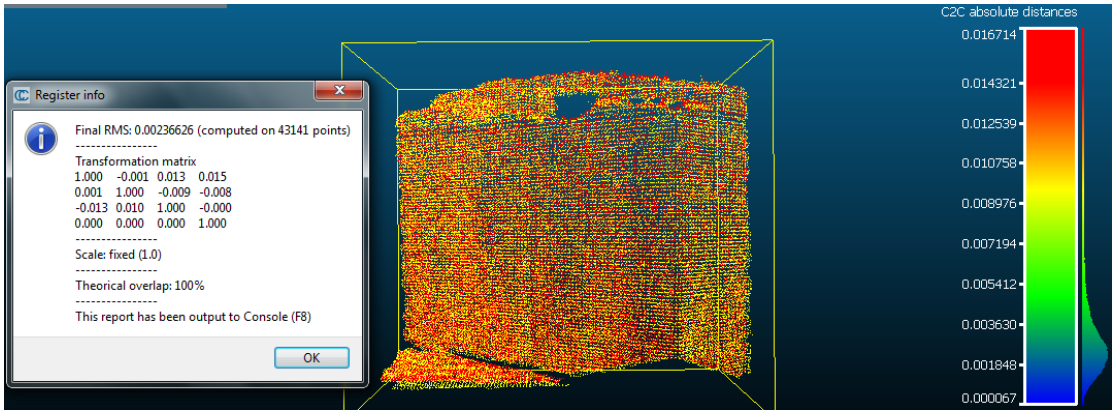
Εικόνα 7–45: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Στη παραπάνω εικόνα, διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect για διαφορετικές συνθήκες φωτισμού, που αφορούν στη θήκη GPS. Κατόπιν κατάλληλης επιλογής ομόλογων σημείων στα δύο νέφη, το παραγόμενο μέσο τετραγωνικό σφάλμα είναι της τάξης των 8mm.

Παρακάτω (εικόνα 7-46), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το λογισμικό και οι οποίες, στη μεγάλη πλειοψηφία τους, κυμαίνονται κάτω από τα 4mm.



Εικόνα 7–46: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα

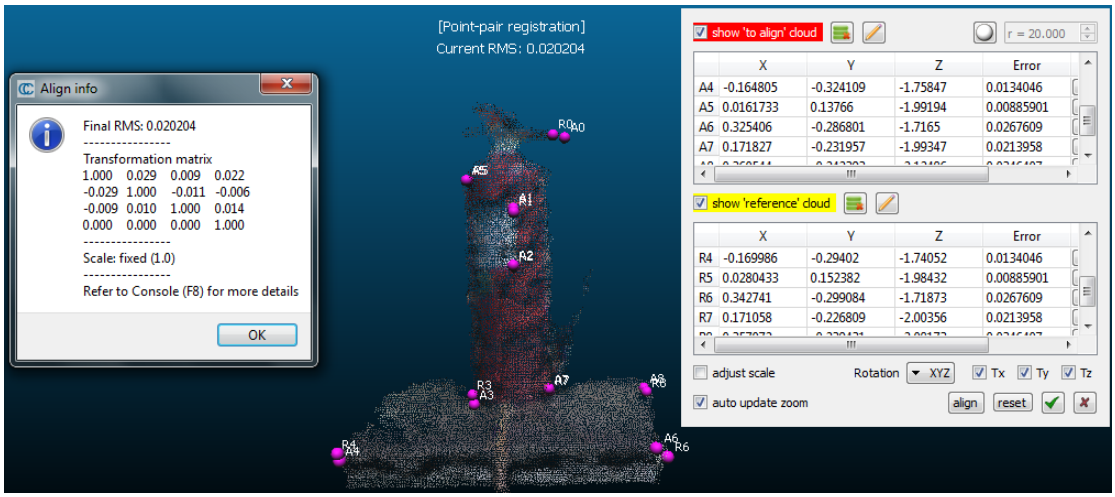


Εικόνα 7–47: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Όμοια, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, υποδιπλασιάζει το RMS στο επίπεδο των 2mm (Εικόνα 7-47).

7.4.2.2 Απόσταση 2m

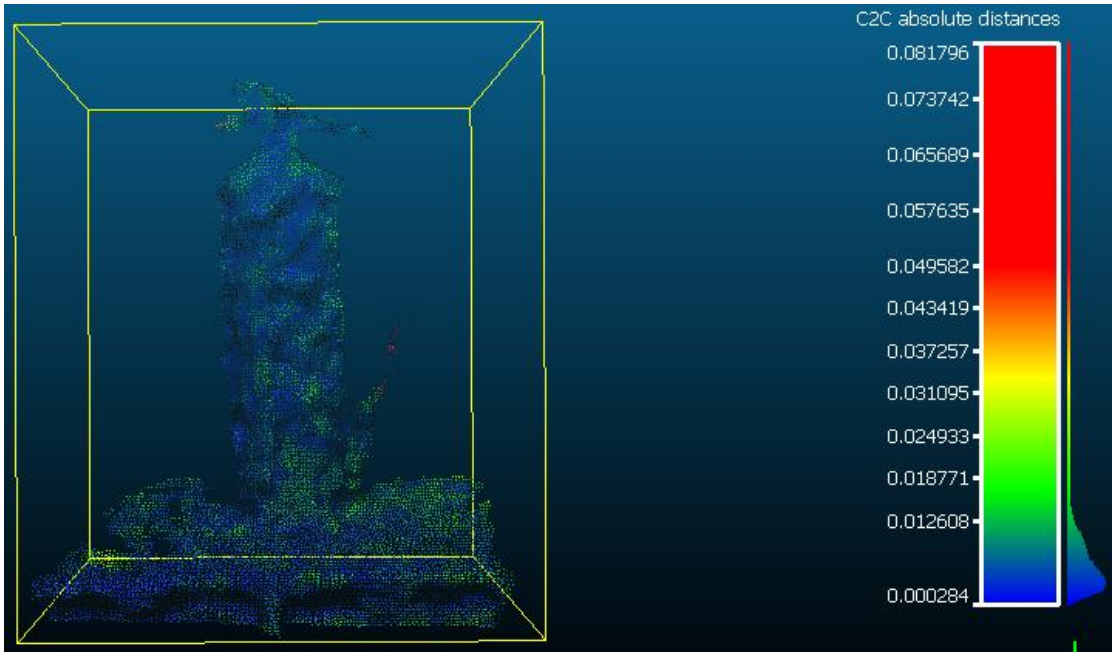
Πυροσβεστήρας



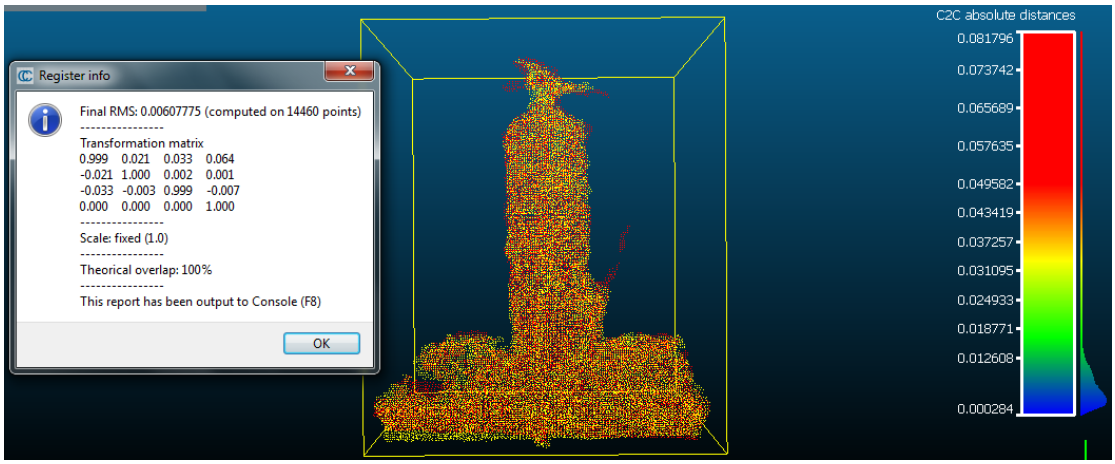
Εικόνα 7–48: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Στη παραπάνω εικόνα διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect για διαφορετικές συνθήκες φωτισμού, όπου με κατάλληλη επιλογή ομόλογων σημείων, χαρακτηρίζεται από μέσο τετραγωνικό σφάλμα (RMS) της τάξης των 2cm, σχεδόν διπλάσιο του αντίστοιχου σφάλματος στην απόσταση του 1m.

Παρακάτω (εικόνα 7-49), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, οι οποίες κυμαίνονται στη μεγάλη πλειοψηφία τους κάτω από το 1cm.



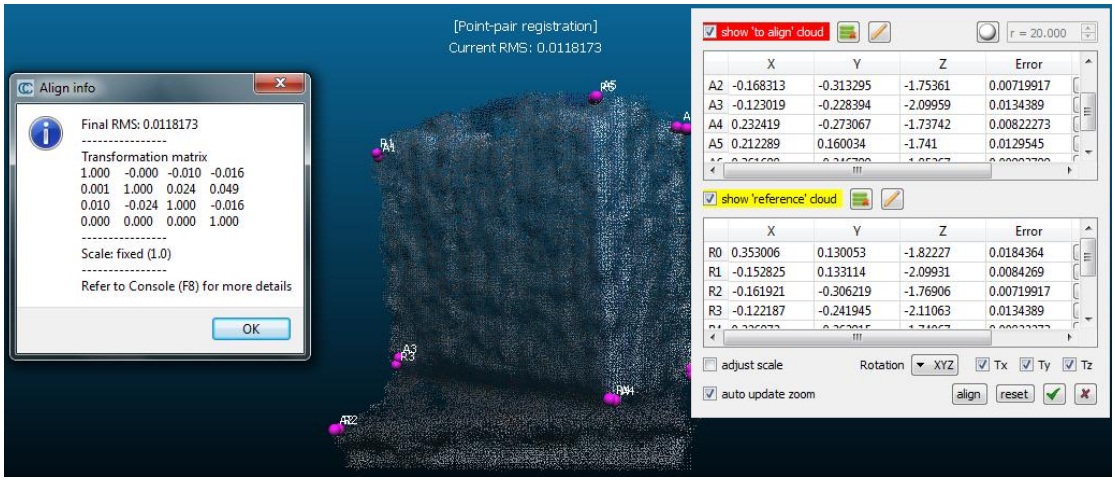
Εικόνα 7–49: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–50: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Αναμενόμενα, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει θεαματικά το RMS στο επίπεδο των 6mm (Εικόνα 7-50).

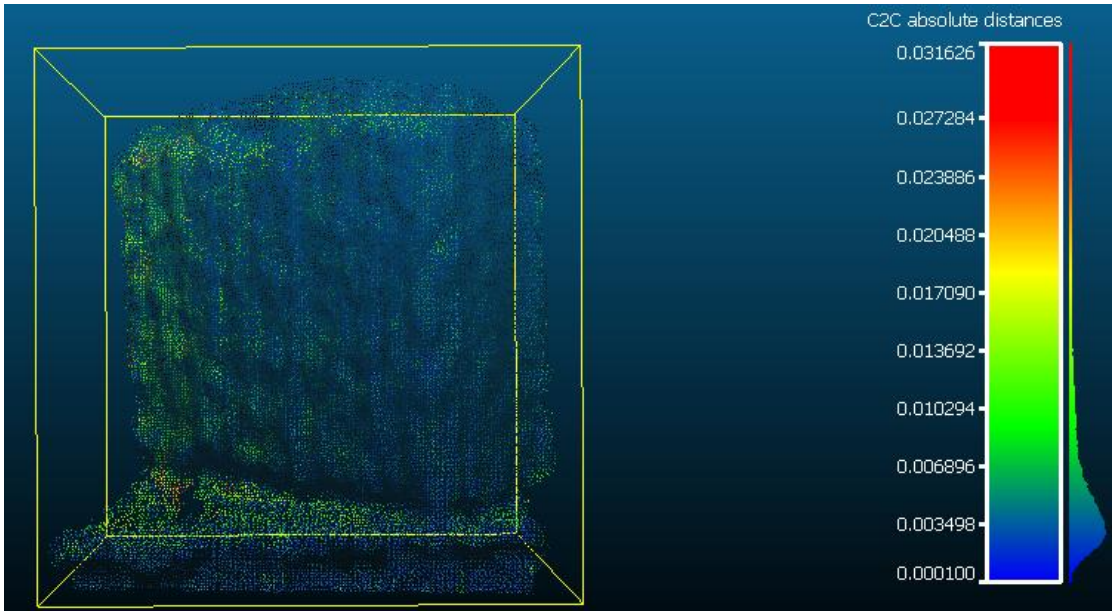
Θήκη GPS



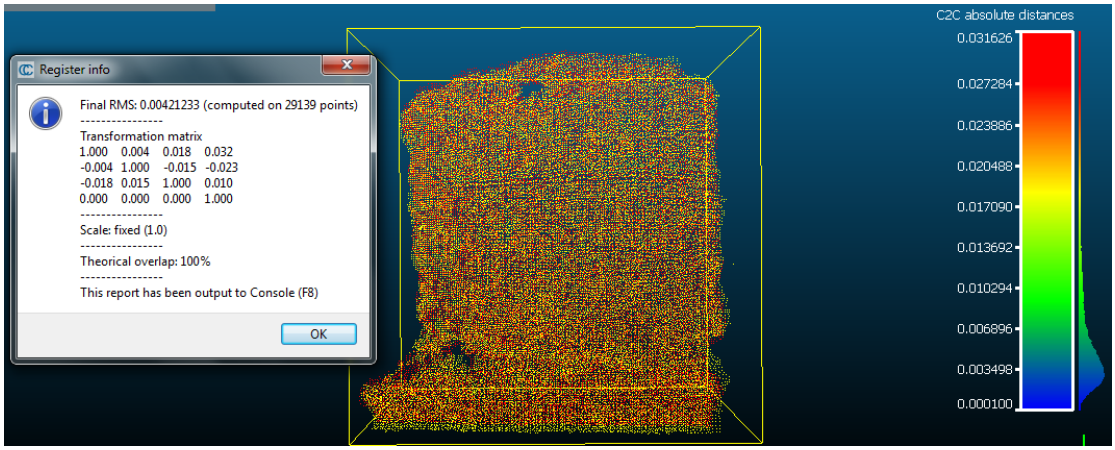
Εικόνα 7–51: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Στη παραπάνω εικόνα, διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect για διαφορετικές συνθήκες φωτισμού, που αφορούν στη θήκη GPS. Κατόπιν κατάλληλης επιλογής ομόλογων σημείων στα δύο νέφη, το παραγόμενο μέσο τετραγωνικό σφάλμα είναι της τάξης του 1cm, ελαφρώς μεγαλύτερο από το αντίστοιχο σφάλμα στην απόσταση του 1m.

Παρακάτω (εικόνα 7-52), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το λογισμικό και οι οποίες κυμαίνονται ως επί το πλείστον, κάτω από τα 7mm.



Εικόνα 7–52: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα

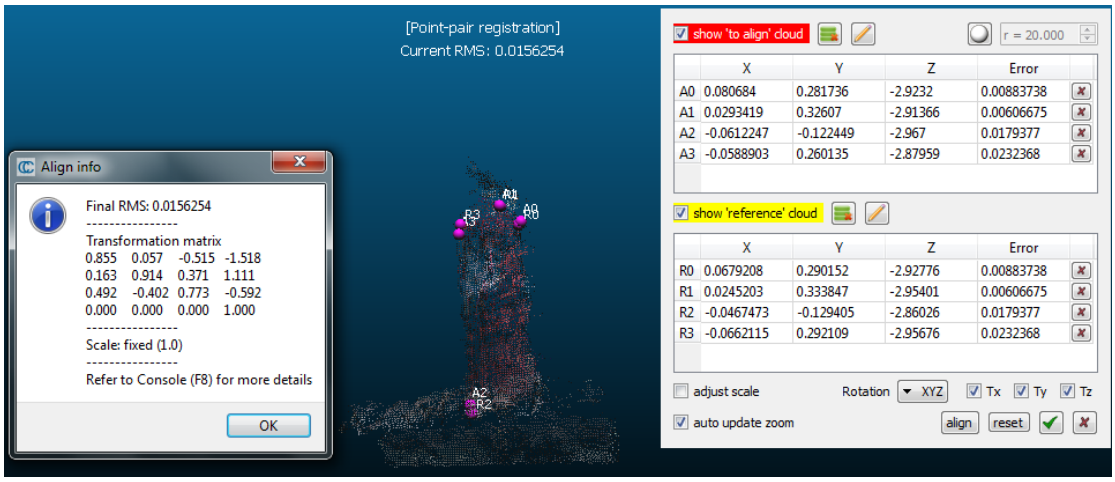


Εικόνα 7–53: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Εξίσου αναμενόμενα, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, υποδιπλασιάζει το RMS στο επίπεδο των 4mm (Εικόνα 7-53), σχεδόν διπλάσιο από το αντίστοιχο σφάλμα στην απόσταση του 1m.

7.4.2.3 Απόσταση 3m

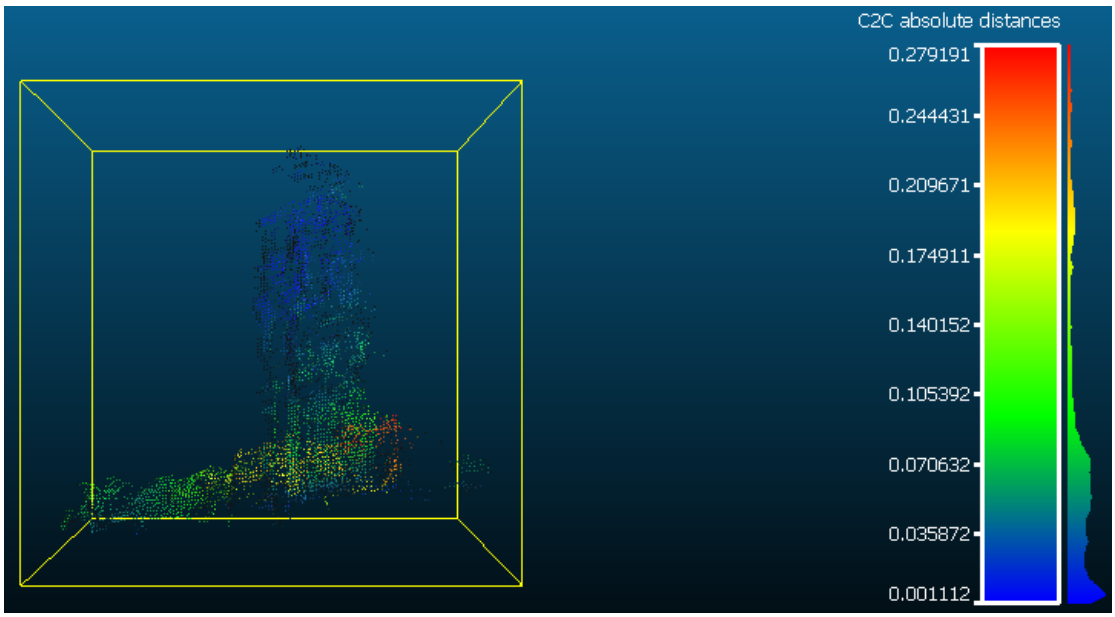
Πυροσβεστήρας



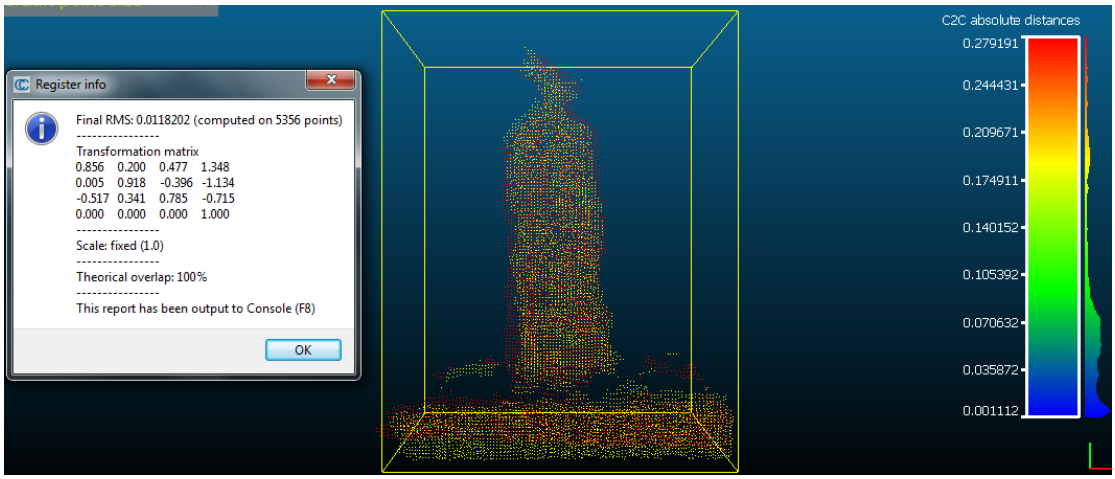
Εικόνα 7–54: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Στη παραπάνω εικόνα διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect για διαφορετικές συνθήκες φωτισμού, όπου με κατάλληλη επιλογή ομόλογων σημείων, χαρακτηρίζεται από μέσο τετραγωνικό σφάλμα (RMS) της τάξης του 1,5cm, λίγο μικρότερο του αντίστοιχου σφάλματος στην απόσταση των 2m.

Παρακάτω (εικόνα 7-55), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, οι οποίες κυμαίνονται στη μεγάλη πλειοψηφία τους κάτω από τα 7cm.



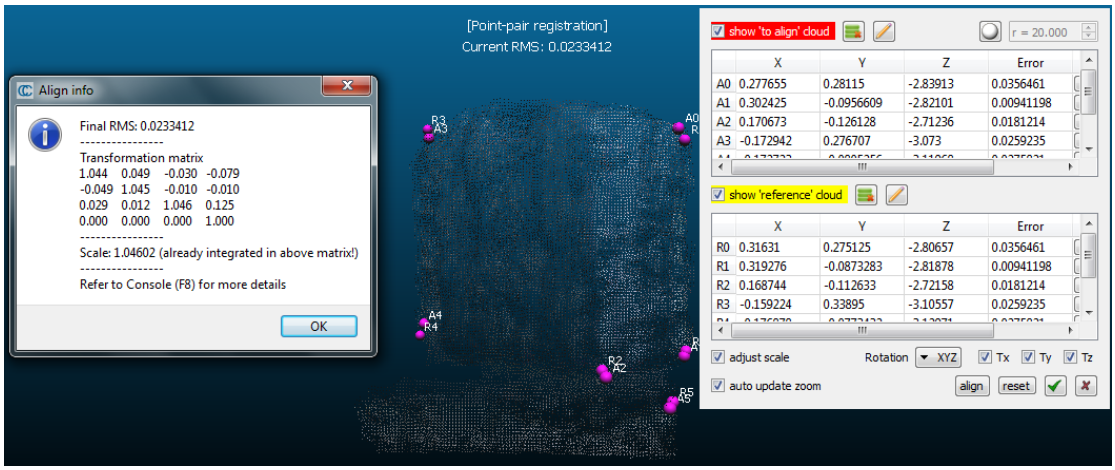
Εικόνα 7–55: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–56: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Αναμενόμενα, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, μειώνει αισθητά το RMS στο επίπεδο του 1cm (Εικόνα 7-56).

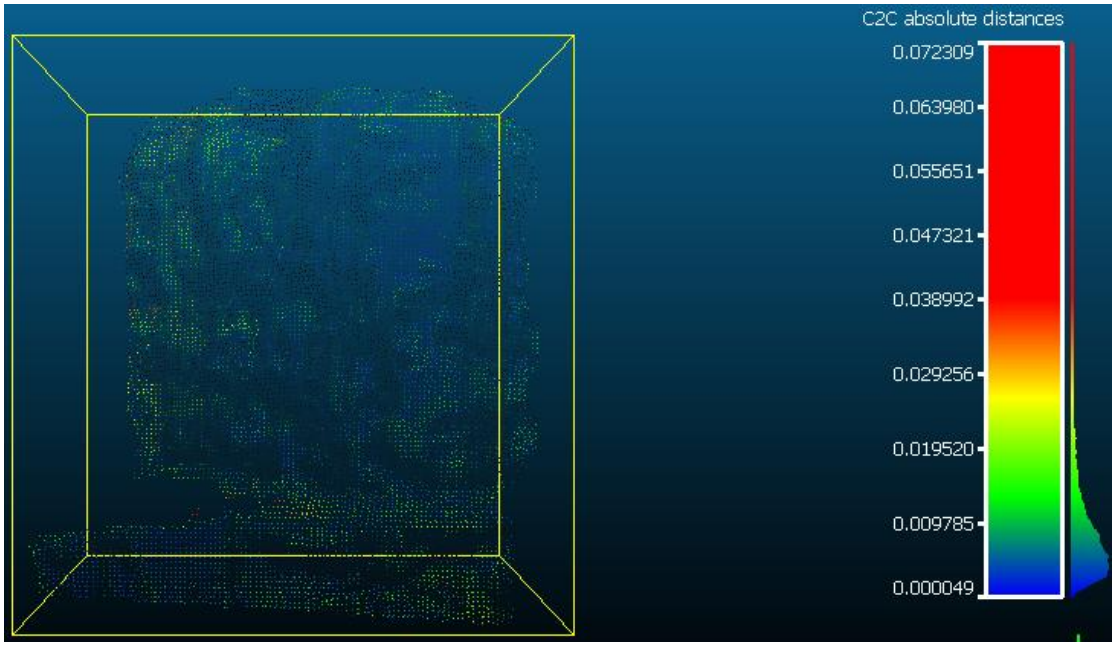
Θήκη GPS



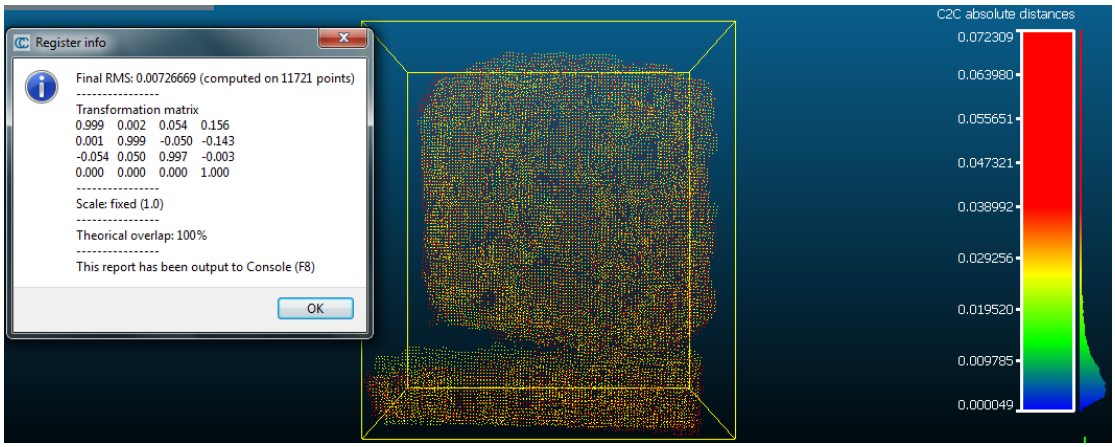
Εικόνα 7–57: Η διαδικασία αγκίστρωσης των δύο νεφών
Πηγή: Ίδια εικόνα

Στη παραπάνω εικόνα, διακρίνεται η διαδικασία αγκίστρωσης των νεφών του αισθητήρα Kinect για διαφορετικές συνθήκες φωτισμού, που αφορούν στη θήκη GPS σε απόσταση 3m. Κατόπιν κατάλληλης επιλογής ομόλογων σημείων στα δύο νέφη, το παραγόμενο μέσο τετραγωνικό σφάλμα είναι της τάξης των 2cm, σχεδόν διπλάσιο από το αντίστοιχο σφάλμα στην απόσταση των 2m.

Παρακάτω (εικόνα 7-58), διακρίνονται οι απόλυτες αποστάσεις μεταξύ των δύο νεφών, όπως υπολογίστηκαν από το λογισμικό και οι οποίες κυμαίνονται ως επί το πλείστον, κάτω από το 1cm.



Εικόνα 7–58: Το αντίστοιχο C2C γράφημα
Πηγή: Ίδια εικόνα



Εικόνα 7–59: Η εφαρμογή του αλγορίθμου ICP
Πηγή: Ίδια εικόνα

Όμοια, η εφαρμογή του αλγορίθμου ICP σε ένα ήδη αγκιστρωμένο ζεύγος νεφών, υποδιπλασιάζει το RMS στο επίπεδο των 7mm (Εικόνα 7-59), αισθητά μεγαλύτερο από το αντίστοιχο σφάλμα στην απόσταση των 2m.

8. Συμπεράσματα - Προτάσεις

Στο κεφάλαιο αυτό επιχειρείται η εξαγωγή συμπερασμάτων αναφορικά με το βαθμό εκπλήρωσης του στόχου της εργασίας, με γνώμονα την ποιότητα του τελικού αποτελέσματος και τη συνολική εμπειρία χρήστη.

Ο στόχος της εργασίας συνίσταται στη τρισδιάστατη χαρτογράφηση εσωτερικών χώρων σε πραγματικό χρόνο με χρήση του αισθητήρα Kinect for Windows (v1). Στο σημείο αυτό υπενθυμίζεται στον αναγνώστη πως το Kinect αποτελεί ένα πακέτο αισθητήρων, σχεδιασμένο να αποτελέσει τη φυσική διεπαφή μεταξύ ενός Η/Υ και του ανθρώπου, και όχι εξειδικευμένο τοπογραφικό εργαλείο. Συνεπώς, ο σκοπός ανάπτυξης του και το σχετικό του κόστος των €150 πρέπει να ληφθούν υπόψη κατά την αξιολόγηση των αποτελεσμάτων της παρούσης.

Αρχικά, σε ότι αφορά την καθαυτή επίτευξη του στόχου που τέθηκε, το αποτέλεσμα κρίνεται επιτυχές. Ο συνδυασμός του ελεύθερου λογισμικού RGBDemo 0.7.0, της PCL της OpenCV και του ICP με τον αισθητήρα Kinect, δύναται να παραγάγει νέφη σημείων για την αναπαράσταση εσωτερικών (και όχι μόνον) χώρων.

Σημαντικό πλεονέκτημα της διαδικασίας, πέραν του χαμηλότατου κόστους, αποτελεί η ιδιαίτερα σύντομη διάρκεια και η επόπτευση του αποτελέσματος σε πραγματικό χρόνο. Στα μειονεκτήματα αυτής καταλογίζονται οι υψηλές απαιτήσεις σε εμπειρία χρήστη, υπολογιστική ισχύ και τα ασυνεπή αποτελέσματα.

Αναλυτικότερα, το γεγονός πως ο οικιακός χρήστης μπορεί εντός ολίγων λεπτών να παραγάγει ένα νέφος σημείων που αναπαριστά το αντικείμενο/χώρο της επιλογής του, χρησιμοποιώντας ένα περιφερειακό Η/Υ σε συνδυασμό με το φιλικό γραφικό περιβάλλον, αποτελεί το κατεξοχήν πλεονέκτημα της υλοποίησης αυτής. Σε κάθε περίπτωση, το δημιουργούμενο νέφος σημείων επαρκεί σε ποιότητα για τις όποιες ανάγκες του μέσου χρήστη. Το γεγονός αυτό είναι η αιτία που συνεχώς περισσότεροι χρήστες πειραματίζονται με τον αισθητήρα Kinect και στάθηκε ως αφορμή για την ανάθεση και εκτέλεση της παρούσας εργασίας. Εντούτοις, η κατάσταση περιπλέκεται όταν από το πεδίο του οικιακού χρήστη περάσουμε σε αυτό της τοπογραφικής τέχνης και επιστήμης.

Βασικό μειονέκτημα της υλοποίησης αποτελεί η δεδομένη ανάγκη εξοικείωσης του χρήστη με περιβάλλοντα και γλώσσες προγραμματισμού με τα οποία ενδεχομένως να μην είναι συνηθισμένος. Η διαδικασία εγκατάστασης περιλαμβάνει πλήθος βημάτων για την εγκατάσταση μιας πλειάδας software, middleware, framework, APIs και drivers – πολλά εκ των οποίων είναι τόσο δυσεύρετα όσο και δύσχρηστα υπό την έννοια πως η μεταξύ τους συμβατότητα εξαρτάται τόσο από τη σειρά (!) εγκατάστασης, το λειτουργικό σύστημα και έτερα εγκατεστημένα προγράμματα. Η δυσχέρεια αυτή εντείνεται περεταίρω από τη περιορισμένη σχετική βιβλιογραφία.

Σε επόμενο στάδιο, η χρήση του λογισμικού, μολονότι φιλική εκ πρώτης όψεως, δυσχεραίνει καθώς μεγαλώνει η κλίμακα και οι απαιτήσεις. Η έλλειψη δυνατότητας μετακίνησης στον κάθετο και οριζόντιο άξονα (pan) δυσκολεύει το έργο της εποπτείας σε πραγματικό του δημιουργούμενου νέφους, καθώς νέα αντικείμενα/δωμάτια προστίθενται στο μοντέλο. Επίσης, οι απαιτήσεις σε υπολογιστική ισχύ (βλ. επεξεργαστής, μνήμη RAM) οδηγούν ενίοτε σε ανεπιθύμητα αποτελέσματα και έχουν αρνητικές επιπτώσεις στην ποιότητα του τελικού αποτελέσματος. Η χρήση του αλγόριθμου ICP επιβάλλει τη λήψη επικαλυπτόμενων καρέ και μάλλον αναμενόμενα, περισσότερες εικόνες ισοδυναμούν με καλύτερο ποιοτικά αποτέλεσμα. Ωστόσο, από τη φύση του το αντικείμενο της επεξεργασίας εικόνων είναι υπολογιστικά απαιτητικό, συνεπώς ο χρήστης πρέπει να βρει τη χρυσή τομή μεταξύ ποιότητας και διάρκειας καταγραφής - γεγονός που αναμφίβολα υποβαθμίζει το τελικό αποτέλεσμα.

Τέλος, σε ότι αφορά τη ποιότητα καθαυτή του παραγόμενου νέφους σημείων, η σύγκριση που προηγήθηκε (βλ. Κεφάλαιο 7), οδηγεί σε σημαντικά συμπεράσματα. Τα αποτελέσματα αυτής παρατίθενται παρακάτω με τη μορφή πίνακα και ακολουθεί η ερμηνεία τους.

	Κριτήριο					
	Απόσταση (Kinect vs Laser Scanner)			**Φωτισμός (Kinect vs Kinect)**		
Αντικείμενο / Απόσταση	**1m**	**2m**	**3m**	**1m**	**2m**	**3m**
	Μέσο Τετραγωνικό Σφάλμα (RMS) σε mm					
Καρέκλα	6	23	22	–	–	–
Πυροσβεστήρας	7	8	15	2	6	12
Θήκη GPS	16	8	11	2	4	7
Κάδος απορριμμάτων	10	10	15	–	–	–

Πίνακας 1: Συγκεντρωτικά αποτελέσματα του ποιοτικού ελέγχου
Πηγή: Ίδια εργασία

Τα συγκριτικά αποτελέσματα μεταξύ Kinect και laser scanner για τα τέσσερα αντικείμενα της δοκιμής στις τρεις αποστάσεις καθιστούν σαφές πως η υλοποίηση δεν μπορεί να συγκριθεί ευθέως σε όρους απόλυτης ακρίβειας με τον σαρωτή. Τα αποτελέσματα για αποστάσεις έως 2m κυμαίνονται μεταξύ 0,5 και 1cm (με εξαιρέσεις) και σε συνδυασμό με την ασυνέπεια στη καταγραφή, κρίνονται μη ανταγωνιστικά για τη συνήθεις φωτογραμμετρικές διαδικασίες που μετέρχονται οι Αγρονόμοι – Τοπογράφοι μηχανικοί. Σε μεγαλύτερες αποστάσεις το μέσο τετραγωνικό σφάλμα αυξάνεται περεταίρω ενώ και ο αριθμός των καταγραφόμενων

σημείων μειώνεται σημαντικά, ενισχύοντας την παραπάνω κρίση και το προβάδισμα του σαρωτή.

Εντούτοις, πρέπει να σημειωθεί πως οι λήψεις για τη μετέπειτα σύγκριση έγιναν από σταθερό σημείο γεγονός που καταστρατηγεί την χρήση του αλγόριθμου ICP για τη βελτίωση του παραγόμενου νέφους. Πιθανώς, μια διαφορετική - μη στατική - προσέγγιση ως προς τη μεθοδολογία της σύγκρισης να οδηγούσε σε κολακευτικότερα αποτελέσματα για το Kinect, με παράλληλη ελάττωση της σχετικής ασυνέπειας. Ωστόσο, σε περίπτωση που ληφθεί υπόψη ο παράγοντας κόστος, τα αποτελέσματα που παράγει το Kinect με σφάλμα - έστω μη γραμμικό - μικρότερο των 2cm για αποστάσεις μικρότερες των 3m, μπορούν να χαρακτηριστούν εντυπωσιακά. Το παραπάνω εύρος των 0,5 έως 3m είναι και η προτεινόμενη εμβέλεια χρήσης της εφαρμογής.

Σε ότι αφορά την επίδραση του περιβάλλοντος φωτισμού στη ποιότητα του νέφους, τα αποτελέσματα του πίνακα φανερώνουν μια μικρή αλλά αισθητή υποβάθμιση για χειρότερες συνθήκες φωτισμού, η οποία εντείνεται καθώς αυξάνεται η απόσταση καταγραφής. Μια άλλη ανάγνωση του πίνακα, σε συνδυασμό με την εμπειρία χρήσης του γράφοντος και το οπτικό αποτέλεσμα που διακρίνεται στις εικόνες του κεφαλαίου που προηγήθηκε, φανερώνει πως η ανακλαστικότητα του αντικειμένου σε συνδυασμό με το μέγεθος του, επιδρούν σημαντικά στη ποιότητα του παραγόμενου νέφους. Η μεν ανακλαστικότητα οδηγεί σε αδυναμία καταγραφής από τη συσκευή, το δε μέγεθος δυσχεραίνει τη καταγραφή η οποία όμως βελτιώνεται με τη καταγραφή επιπλέον καρέ από διαφορετικές γωνίες θέασης.

Συμπερασματικά, η προτεινόμενη υλοποίηση μπορεί να μην ενδείκνυται για φωτογραμμετρικές εργασίες όπως π.χ. αποτυπώσεις μνημείων [17], ωστόσο υπερκαλύπτει τις ανάγκες του μέσου χρήστη είτε πρόκειται για απλές εφαρμογές, είτε για πιο σύνθετες.

Σε μια αγορά όπου η τρισδιάστατη πληροφορία κερδίζει συνεχώς έδαφος, η υλοποίηση αυτή θα μπορούσε να τύχει εφαρμογής και σε άλλους τομείς όπου η ακρίβεια είναι μεν επιθυμητή, δεν αποτελεί όμως αυτοσκοπό. Τέτοιες εφαρμογές είναι μεταξύ άλλων, η ταχεία δημιουργία γραφικών, η πλοήγηση ρομπότ στο χώρο, η ασφαλής προσγείωση μη επανδρωμένων αεροσκαφών (drone) [12], τα προηγμένα συστήματα ασφαλείας, συστήματα επιθεώρησης εγκαταστάσεων και οι τρισδιάστατες εκτυπώσεις (3D printing).

Τέλος, όσον αφορά τυχούσες προσθήκες – βελτιώσεις στην υφιστάμενη λειτουργικότητα που θα μπορούσαν να διευρύνουν σημαντικά τα δυνητικά πεδία εφαρμογής της παρούσης, η κατάτμηση (segmentation) και η αναγνώριση προτύπων (pattern recognition) κατέχουν εξέχουσες θέσεις. Η υιοθέτηση τους θα προσέθετε, κυριολεκτικά, μια νέα διάσταση σε δύο ήδη πολλά υποσχόμενα επιστημονικά πεδία.

9. ΒΙΒΛΙΟΓΡΑΦΙΑ

Διεθνής

1. S. Malik, T. S. Chol, H. Nisar "*Depth Map and 3D Imaging Applications – Algorithms and Technologies*", 2012

2. J. Webb, J. Ashley "*Beginning Kinect Programming with the Microsoft Kinect SDK*", 2012

3. A. Nuchter "*3D Roboting Mapping – The SLAM Problem with Six Degrees of Freedom*", 2009

4. K. Khoshelham, S. O. Elberink "*Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications*", Sensors – Open Access Journal, 2012

5. J. Kramer, N.Burrus, D. Herrera C. F. Echtler, M. Parker, "*Hacking the Kinect*", 2012

6. S. Rusinkiewicz, M. Levoy "*Efficient Variants of the ICP Algorithm*", IEEE Computer Soc., 2001

7. G. Galatas, S. Ferdous, F. Makedon "*Multimodal Person Localization and Emergency Detection Using The Kinect*", 2013

8. S. Kean, J. Hall, P. Perry "*Meet the Kinect – An Introduction to Programming Natural User Interfaces*", 2011

9. C. Dal. Mutto, P. Zannutigh, G. M. Cortelazzo "*Time-of-Flight Cameras and Microsoft Kinect*", 2013

10. T. Kahlmann, H. Ingensand "*Calibration and development for increased accuracy of 3D range imaging cameras*", Journal of Applied. Geodesy, 2008

11. M. Lindner, I. Schiller, A. Kolb, R. Koch "*Time-of-flight sensor calibration for accurate range sensing*", Computer Vision and Image Understanding, 2010

12. P. Benavidez, M. Jamshidi "*Mobile robot navigation and target tracking system*", Proceedings of the 6th International Conference on System of Systems Engineering: SoSE in Cloud Computing, Smart Grid, and Cyber Security, SoSE, 2011

13. P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox *"RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments"*, Proceedings of International Symposium on Experimental Robotics (ISER), 2010.

14. E. Herbst, X. Ren, D. Fox *"Toward object discovery and modeling via 3-D scene comparison"*, Proceedings of IEEE International Conference on Robotics and Automation, 2011.

15. J. M. Gottfried, J. Fehr, C. S. Garbe *"Computing range flow from multi-modal Kinect data"*, Proceedings of the 7th International Symposium on Visual Computing (ISVC), 2011

Ελληνική

16. Α. Β. Αναστασίου, Δ. Πρέκα "*Η Χρήση της Τρισδιάστατης Σάρωσης στις Τοπογραφικές Αποτυπώσεις*", 2011

17. Γ. Κοντογιάννη, Α. Γεωργόπουλος "*Τρισδιάστατη εικονική ανακατασκευή τμήματος της μεσαίας στοάς στην αρχαία αγορά των Αθηνών*" 2014

18. Γ. Καρράς "*Γραμμικοί Μετασχηματισμοί Συντεταγμένων στη Φωτογραμμετρία*", 1998

Ιστοχώροι

19. RGBDemo Homepage - www.rgbdemo.org

20. OpenCV Homepage - www.opencv.org

21. PCL Homepage - www.pointclouds.org

22. Qt Homepage - www.qt.io

23. Wikipedia Homepage - www.wikipedia.org

24. OpenGL Homepage - www.opengl.org

25. www.midnight-tech.de

26. Princeton Homepage - www.cs.princeton.edu

27. Stack Overflow Homepage - www.stackoverflow.com

28. GitHub Homepage - www.github.com

29. Microsoft Homepage - www.microsoft.com

30. XBOX Homepage - www.xbox.com

31. Faro Homepage - www.faro.com

32. CloudCompare Homepage - www.cloudcompare.org

ΠΑΡΑΡΤΗΜΑ

Κώδικας

calibrate_kinect.cpp

```
#include <ntk/ntk.h>
#include <ntk/camera/calibration.h>
#include <ntk/geometry/pose_3d.h>
// #include <opencv/cv.h>
#include <fstream>

#include <QDir>
#include <QDebug>

using namespace ntk;
using namespace cv;

// example command line (for copy-n-paste):
// calibrate_one_camera -w 8 -h 6 -o camera.yml images

namespace global
{
  ntk::arg<const char*> opt_image_directory(0, "RGBD images directory", 0);
  ntk::arg<const char*> opt_ref_points_file("--pattern-ref", "Ref points file", "patt
  ntk::arg<const char*> opt_output_file("--output", "Output YAML filename", "kinect_c
  ntk::arg<int> opt_pattern_width("--pattern-width", "Pattern width (number of inner
  ntk::arg<int> opt_pattern_height("--pattern-height", "Pattern height (number of inn
  ntk::arg<float> opt_square_size("--pattern-size", "Square size in used defined scal

  const cv::Size image_size(640,480);

  QDir images_dir;
  QStringList images_list;
  std::vector<Point3f> pattern_ref;

  cv::Mat1d depth_intrinsics;
  cv::Mat1d depth_distortion;

  cv::Mat1d rgb_intrinsics;
  cv::Mat1d rgb_distortion;

  // stereo transform.
  cv::Mat1d R, T;
}

// Find depth intrinsics and return undistorted corners location
void calibrate_kinect_depth(std::vector< std::vector<Point2f> >& stereo_corners)
{
  std::vector< std::vector<Point2f> > good_corners;
  stereo_corners.resize(global::images_list.size());
  for (int i_image = 0; i_image < global::images_list.size(); ++i_image)
  {
    QString filename = global::images_list[i_image];
    QDir cur_image_dir (global::images_dir.absoluteFilePath(filename));
    std::string full_filename = cur_image_dir.absoluteFilePath("raw/depth.png.calib")
    cv::Mat3b image = imread(cur_image_dir.absoluteFilePath("raw/depth.png").toStdStr
    ntk_ensure(image.data, "Could not read image");
    ntk_dbg_print(full_filename, 1);

    std::vector<Point2f> current_view_corners;

    std::ifstream f(full_filename.c_str());
    if (!f.good())
```

```
    {
      ntk_dbg(0) << "Warning: Could not load calib file.";
      stereo_corners[i_image].resize(0);
      continue;
    }

    current_view_corners.resize(global::pattern_ref.size());
    foreach_idx(i, current_view_corners)
    {
      f >> current_view_corners[i].x >> current_view_corners[i].y;
    }
    stereo_corners[i_image] = current_view_corners;
    good_corners.push_back(current_view_corners);

    show_corners(image, current_view_corners);
  }

  std::vector< std::vector<Point3f> > pattern_points (good_corners.size());
  for(int i = 0; i < good_corners.size(); ++i)
  {
    pattern_points[i] = global::pattern_ref;
  }

  std::vector<Mat> rvecs, tvecs; // not used.
  calibrateCamera(pattern_points, good_corners, global::image_size,
                  global::depth_intrinsics, global::depth_distortion,
                  rvecs, tvecs);

  // Undistort depth corners.
  foreach_idx(i, stereo_corners)
  {
    if (stereo_corners[i].size() > 0)
    {
      cv::undistortPoints(Mat(stereo_corners[i]), stereo_corners[i],
                          global::depth_intrinsics, global::depth_distortion,
                          cv::Mat(), global::depth_intrinsics);
    }
  }
}

// Find rgb intrinsics and return undistorted corners location
void calibrate_kinect_rgb(std::vector< std::vector<Point2f> >& stereo_corners)
{
  std::vector< std::vector<Point2f> > good_corners;
  stereo_corners.resize(global::images_list.size());
  for (int i_image = 0; i_image < global::images_list.size(); ++i_image)
  {
    QString filename = global::images_list[i_image];
    QDir cur_image_dir (global::images_dir.absoluteFilePath(filename));

    std::string full_filename = cur_image_dir.absoluteFilePath("raw/color.png").toStd
    ntk_dbg_print(full_filename, 1);
    cv::Mat3b image = imread(full_filename);
    ntk_ensure(image.data, "Could not load color image");

    std::vector<Point2f> current_view_corners;
    calibrationCorners(full_filename, "corners",
                       global::opt_pattern_width(), global::opt_pattern_height(),
                       current_view_corners, image, 1);

    if (current_view_corners.size() == global::opt_pattern_height()*global::opt_patte
    {
      good_corners.push_back(current_view_corners);
      // FIXME: compute corners automatically.
      stereo_corners[i_image].resize(global::pattern_ref.size());
      show_corners(image, current_view_corners);
    }
```

```
148       else
149       {
150         ntk_dbg(0) << "Warning: corners not detected";
151         stereo_corners[i_image].resize(0);
152       }
153     }
154
155
156     std::vector< std::vector<Point3f> > pattern_points;
157     calibrationPattern(pattern_points,
158                        global::opt_pattern_width(),  global::opt_pattern_height(), glob
159                        good_corners.size());
160
161     ntk_assert(pattern_points.size() == good_corners.size(), "Invalid points size");
162
163     std::vector<Mat> rvecs(good_corners.size()), tvecs(good_corners.size());
164     double error = calibrateCamera(pattern_points, good_corners, global::image_size,
165                                    global::rgb_intrinsics, global::rgb_distortion,
166                                    rvecs, tvecs);
167
168     int good_i = 0;
169     foreach_idx(stereo_i, stereo_corners)
170     {
171       if (stereo_corners[stereo_i].size() > 0)
172       {
173         QString filename = global::images_list[stereo_i];
174         QDir cur_image_dir (global::images_dir.absoluteFilePath(filename));
175         std::string full_filename = cur_image_dir.absoluteFilePath("raw/color.png").toS
176         cv::Mat3b image = imread(full_filename);
177         ntk_ensure(image.data, "Could not load color image");
178
179         cv::Mat3b undistorted_image;
180         undistort(image, undistorted_image, global::rgb_intrinsics, global::rgb_distort
181
182         std::vector<Point2f> current_view_corners;
183         calibrationCorners(full_filename, "corners",
184                            global::opt_pattern_width(), global::opt_pattern_height(),
185                            current_view_corners, undistorted_image, 1);
186
187         if (current_view_corners.size() != global::opt_pattern_height()*global::opt_pat
188         {
189           stereo_corners[stereo_i].resize(0);
190           continue;
191         }
192
193         cv::Mat1f H;
194         estimate_checkerboard_pose(pattern_points[0],
195                                    current_view_corners,
196                                    global::rgb_intrinsics,
197                                    H);
198         Pose3D pose;
199         pose.setCameraParametersFromOpencv(global::rgb_intrinsics);
200         ntk_dbg_print(pose, 1);
201         pose.setCameraTransform(H);
202
203         // FIXME: why rvecs and tvecs from calibrateCamera seems to be nonsense ?
204         // calling findExtrinsics another time to get good chessboard estimations.
205
206         foreach_idx(pattern_i, global::pattern_ref)
207         {
208           ntk_dbg_print(global::pattern_ref[pattern_i], 1);
209           Point3f p = pose.projectToImage(global::pattern_ref[pattern_i]);
210           ntk_dbg_print(p, 1);
211           stereo_corners[stereo_i][pattern_i] = Point2f(p.x, p.y);
212         }
```

```
215     }
216   }
217 }
218
219 void calibrate_kinect_stereo(const std::vector< std::vector<Point2f> >& undistorted_r
220                              const std::vector< std::vector<Point2f> >& undistorted_d
221 {
222   ntk_assert(undistorted_depth_corners.size() == undistorted_rgb_corners.size(), "Siz
223   std::vector< std::vector<Point2f> > undistorted_good_rgb;
224   std::vector< std::vector<Point2f> > undistorted_good_depth;
225
226   foreach_idx(i, undistorted_depth_corners)
227   {
228     if (undistorted_depth_corners[i].size() > 0 && undistorted_rgb_corners[i].size()
229     {
230       ntk_assert(undistorted_depth_corners[i].size() == undistorted_rgb_corners[i].si
231                  "Sizes should be equal.");
232       undistorted_good_depth.push_back(undistorted_depth_corners[i]);
233       undistorted_good_rgb.push_back(undistorted_rgb_corners[i]);
234     }
235   }
236
237   std::vector< std::vector<Point3f> > pattern_points (undistorted_good_depth.size());
238   for(int i = 0; i < pattern_points.size(); ++i)
239     pattern_points[i] = global::pattern_ref;
240
241   cv::Mat E(3,3,CV_64F),F(3,3,CV_64F);
242   cv::Mat zero_dist (global::depth_distortion.size(), global::depth_distortion.type()
243   zero_dist = Scalar(0);
244
245   stereoCalibrate(pattern_points,
246                   undistorted_good_rgb, undistorted_good_depth,
247                   global::rgb_intrinsics, zero_dist,
248                   global::depth_intrinsics, zero_dist,
249                   global::image_size,
250                   global::R, global::T, E, F,
251                   TermCriteria(TermCriteria::COUNT+TermCriteria::EPS, 50, 1e-6),
252                   CALIB_FIX_INTRINSIC);
253
254   double error = computeError(F,
255                               undistorted_good_rgb, undistorted_good_depth);
256   std::cout << "Average pixel reprojection error: " << error << std::endl;
257 }
258
259 void writeNestkMatrix()
260 {
261   FileStorage output_file (global::opt_output_file(),
262                            CV_STORAGE_WRITE);
263   writeMatrix(output_file, "rgb_intrinsics", global::rgb_intrinsics);
264   writeMatrix(output_file, "rgb_distortion", global::rgb_distortion);
265   writeMatrix(output_file, "depth_intrinsics", global::depth_intrinsics);
266   writeMatrix(output_file, "depth_distortion", global::depth_distortion);
267   writeMatrix(output_file, "R", global::R);
268   writeMatrix(output_file, "T", global::T);
269   cv::Mat1i size_matrix(1,2);
270   size_matrix(0,0) = global::image_size.width;
271   size_matrix(0,1) = global::image_size.height;
272   writeMatrix(output_file, "rgb_size", size_matrix);
273   writeMatrix(output_file, "raw_rgb_size", size_matrix);
274   writeMatrix(output_file, "depth_size", size_matrix);
275   writeMatrix(output_file, "raw_depth_size", size_matrix);
276   output_file.release();
277 }
```

```
279  void writeROSMatrix()
280  {
281    // Reload nestk calibration file.
282    RGBDCalibration calib;
283    calib.loadFromFile(global::opt_output_file());
284
285    cv::Mat1f depth_proj = calib.depth_pose->cvProjectionMatrix();
286    cv::Mat1f rgb_proj = calib.rgb_pose->cvProjectionMatrix();
287    cv::Mat1f identity(3,3); setIdentity(identity);
288
289    FileStorage ros_depth_file ("calibration_depth.yaml",
290                                CV_STORAGE_WRITE);
291    writeMatrix(ros_depth_file, "camera_matrix", global::depth_intrinsics);
292    writeMatrix(ros_depth_file, "distortion_coefficients", global::depth_distortion);
293    writeMatrix(ros_depth_file, "rectification_matrix", identity);
294    writeMatrix(ros_depth_file, "projection_matrix", depth_proj);
295    ros_depth_file.release();
296
297    FileStorage ros_rgb_file ("calibration_rgb.yaml",
298                              CV_STORAGE_WRITE);
299    writeMatrix(ros_rgb_file, "camera_matrix", global::rgb_intrinsics);
300    writeMatrix(ros_rgb_file, "distortion_coefficients", global::rgb_distortion);
301    writeMatrix(ros_rgb_file, "rectification_matrix", identity);
302    writeMatrix(ros_rgb_file, "projection_matrix", rgb_proj);
303    ros_rgb_file.release();
304  }
305
306  int main(int argc, char** argv)
307  {
308    arg_base::set_help_option("--help");
309    arg_parse(argc, argv);
310    ntk::ntk_debug_level = 1;
311
312    namedWindow("corners");
313
314    global::images_dir = QDir(global::opt_image_directory());
315    ntk_ensure(global::images_dir.exists(), (global::images_dir.absolutePath() + " is n
316    global::images_list = global::images_dir.entryList(QStringList("view????"), QDir::D
317
318    {
319      std::ifstream f (global::opt_ref_points_file());
320      ntk_ensure(f.good(), "Could not read ref points file.");
321      int nb_points = 0;
322      f >> nb_points;
323      ntk_ensure(nb_points >= 4, "Reference pattern must have at least 4 points");
324      global::pattern_ref.resize(nb_points);
325      foreach_idx(i, global::pattern_ref)
326      {
327        f >>  global::pattern_ref[i].x >>  global::pattern_ref[i].y >>  global::pattern
328      }
329    }
330
331    std::vector< std::vector<Point2f> > rgb_stereo_corners;
332    std::vector< std::vector<Point2f> > depth_stereo_corners;
333
334    calibrate_kinect_rgb(rgb_stereo_corners);
335    calibrate_kinect_depth(depth_stereo_corners);
336    calibrate_kinect_stereo(rgb_stereo_corners, depth_stereo_corners);
337
338    writeNestkMatrix();
339    writeROSMatrix();
340
341    return 0;
342  }
```

reconstructor.cpp

```
#include <ntk/ntk.h>
#include <ntk/camera/calibration.h>
#ifdef NESTK_USE_OPENNI
# include <ntk/camera/openni_grabber.h>
#endif

#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <cstdlib>
#include <sstream>
#include <iomanip>

#include <ntk/image/sift_gpu.h>
#include <ntk/camera/opencv_grabber.h>
#include <ntk/camera/file_grabber.h>
#include <ntk/camera/rgbd_frame_recorder.h>
#include <ntk/geometry/incremental_pose_estimator_from_rgb_features.h>

#ifdef NESTK_USE_FREENECT
# include <ntk/camera/freenect_grabber.h>
#endif

#include <ntk/mesh/mesh_generator.h>
#include <ntk/mesh/surfels_rgbd_modeler.h>
#include "GuiController.h"
#include "ModelAcquisitionController.h"

#include <QApplication>
#include <QMetaType>

using namespace ntk;
using namespace cv;

namespace opt
{
ntk::arg<int> debug_level("--debug", "Debug level", 1);
ntk::arg<const char*> dir_prefix("--prefix", "Directory prefix for output", "grab1");
ntk::arg<const char*> calibration_file("--calibration", "Calibration file (yml)", 0);
ntk::arg<const char*> image("--image", "Fake mode, use given still image", 0);
ntk::arg<const char*> directory("--directory", "Fake mode, use all view???? images in
ntk::arg<int> camera_id("--camera-id", "Camera id for opencv", 0);
ntk::arg<bool> sync("--sync", "Synchronization mode", 0);
ntk::arg<bool> freenect("--freenect", "Force freenect driver", 0);
ntk::arg<bool> high_resolution("--highres", "High resolution color image.", 0);
ntk::arg<bool> use_icp("--icp", "Use ICP to refine pose estimation", 0);
}

int main (int argc, char** argv)
{
    arg_base::set_help_option("-h");
    arg_parse(argc, argv);
    ntk_debug_level = opt::debug_level();
    cv::setBreakOnError(true);

    QApplication::setGraphicsSystem("raster");
    QApplication app (argc, argv);

    const char* fake_dir = opt::image();
    bool is_directory = opt::directory() != 0;
```

```
80      if (opt::directory())
81          fake_dir = opt::directory();
82
83      ntk::RGBDProcessor* processor = 0;
84      RGBDGrabber* grabber = 0;
85
86  #ifdef NESTK_USE_OPENNI
87      OpenniDriver* ni_driver = 0;
88  #endif
89
90      bool use_openni = !opt::freenect();
91  #ifndef NESTK_USE_OPENNI
92      use_openni = false;
93  #endif
94
95      if (opt::image() || opt::directory())
96      {
97          std::string path = opt::image() ? opt::image() : opt::directory();
98          FileGrabber* file_grabber = new FileGrabber(path, opt::directory() != 0);
99          grabber = file_grabber;
100     }
101 #ifdef NESTK_USE_OPENNI
102     else if (use_openni)
103     {
104         // Config dir is supposed to be next to the binaries.
105         QDir prev = QDir::current();
106         QDir::setCurrent(QApplication::applicationDirPath());
107         if (!ni_driver) ni_driver = new OpenniDriver();
108         OpenniGrabber* k_grabber = new OpenniGrabber(*ni_driver);
109         k_grabber->setTrackUsers(false);
110         if (opt::high_resolution())
111             k_grabber->setHighRgbResolution(true);
112         k_grabber->initialize();
113         QDir::setCurrent(prev.absolutePath());
114         grabber = k_grabber;
115     }
116 #endif
117 #ifdef NESTK_USE_FREENECT
118     else
119     {
120         FreenectGrabber* k_grabber = new FreenectGrabber();
121         k_grabber->initialize();
122         k_grabber->setIRMode(false);
123         grabber = k_grabber;
124     }
125 #endif
126
127     ntk_ensure(grabber, "Could not create any grabber. Kinect support built?");
128
129     if (use_openni)
130     {
131         processor = new ntk::OpenniRGBDProcessor();
132     }
133     else
134     {
135         processor = new ntk::FreenectRGBDProcessor();
136     }
137
138     if (opt::sync())
139         grabber->setSynchronous(true);
140
141     RGBDFrameRecorder frame_recorder (opt::dir_prefix());
142     frame_recorder.setSaveOnlyRaw(false);
```

```
144     ntk::RGBDCalibrationPtr calib_data = 0;
145     if (opt::calibration_file())
146     {
147         calib_data = toPtr(new RGBDCalibration());
148         calib_data->loadFromFile(opt::calibration_file());
149     }
150     else if (use_openni)
151     {
152         calib_data = grabber->calibrationData();
153     }
154     else if (QDir::current().exists("kinect_calibration.yml"))
155     {
156         {
157             ntk_dbg(0) << "[WARNING] Using kinect_calibration.yml in current director
158             ntk_dbg(0) << "[WARNING] use --calibration to specify a different file.";
159         }
160         calib_data = toPtr(new RGBDCalibration());
161         calib_data->loadFromFile("kinect_calibration.yml");
162     }
163
164     ntk_ensure(calib_data, "You must specify a calibration file (--calibration)");
165     grabber->setCalibrationData(calib_data);
166
167     GuiController gui_controller (*grabber, *processor);
168     grabber->addEventListener(&gui_controller);
169     gui_controller.setFrameRecorder(frame_recorder);
170
171     if (opt::sync())
172         gui_controller.setPaused(true);
173
174     SurfelsRGBDModeler modeler;
175     modeler.setMinViewsPerSurfel(1);
176     processor->setFilterFlag(RGBDProcessorFlags::ComputeNormals, 1);
177     processor->setMaxNormalAngle(90);
178     processor->setFilterFlag(RGBDProcessorFlags::ComputeMapping, true);
179
180     ModelAcquisitionController* acq_controller = 0;
181     acq_controller = new ModelAcquisitionController (gui_controller, modeler);
182
183     IncrementalPoseEstimatorFromImage* pose_estimator = 0;
184     // FeatureSetParams params ("FAST", "BRIEF64", true);
185     FeatureSetParams params ("SURF", "SURF64", true);
186     pose_estimator = new IncrementalPoseEstimatorFromRgbFeatures(params, opt::use_icp
187
188     acq_controller->setPoseEstimator(pose_estimator);
189     gui_controller.setModelAcquisitionController(*acq_controller);
190
191     grabber->start();
192
193     app.exec();
194     delete acq_controller;
195 }
```

viewer.cpp

```
include <iostream>
include <stdio.h>
include <stdlib.h>
include <cstdlib>
include <sstream>
include <iomanip>

include <ntk/camera/opencv_grabber.h>
include <ntk/camera/file_grabber.h>
include <ntk/camera/rgbd_frame_recorder.h>
include <ntk/camera/rgbd_processor.h>

include <ntk/camera/rgbd_grabber_factory.h>

include <ntk/mesh/mesh_generator.h>
include <ntk/mesh/surfels_rgbd_modeler.h>
include "GuiController.h"
include "ObjectDetector.h"

include <QApplication>
include <QMetaType>
include <QMessageBox>

sing namespace ntk;
sing namespace cv;

amespace opt

tk::arg<int> debug_level("--debug", "Debug level", 1);
tk::arg<const char*> dir_prefix("--prefix", "Directory prefix for output", "grab1");
tk::arg<int> first_index("--istart", "First image index", 0);
tk::arg<const char*> calibration_file("--calibration", "Calibration file (yml)", 0);
tk::arg<const char*> image("--image", "Fake mode, use given still image", 0);
tk::arg<const char*> directory("--directory", "Fake mode, use all view???? images in
tk::arg<int> camera_id("--camera-id", "Camera id for opencv", 0);
tk::arg<bool> openni("--openni", "Force OpenNI driver", 0);
tk::arg<bool> freenect("--freenect", "Force freenect driver", 0);
tk::arg<bool> kin4win("--kin4win", "Force kin4win driver", 0);
tk::arg<bool> softkinetic("--softkinetic", "Force softkinetic driver", 0);
tk::arg<bool> pmd("--pmd", "Force pmd nano driver", 0);
tk::arg<bool> sync("--sync", "Synchronization mode", 0);
tk::arg<bool> high_resolution("--highres", "High resolution color image.", 0);
tk::arg<int> subsampling_factor("--subsampling", "Depth subsampling factor", 1);
tk::arg<bool> savePCD("--savepcd", "Include PCL point clouds in recorded images", 0);


nt main (int argc, char** argv)

    arg_base::set_help_option("-h");
    arg_parse(argc, argv);
    ntk_debug_level = opt::debug_level();
    cv::setBreakOnError(true);

    QApplication::setGraphicsSystem("raster");
    QApplication app (argc, argv);

    RGBDGrabberFactory& grabber_factory = RGBDGrabberFactory::instance();
    RGBDGrabberFactory::Params params;

    if (opt::directory())
        params.directory = opt::directory();
```

```
85      if (opt::openni())
86          params.default_type = RGBDGrabberFactory::OPENNI;
87
88      if (opt::freenect())
89          params.default_type = RGBDGrabberFactory::FREENECT;
90
91      if (opt::kin4win())
92          params.default_type = RGBDGrabberFactory::KIN4WIN;
93
94      if (opt::softkinetic())
95          params.default_type = RGBDGrabberFactory::SOFTKINETIC;
96
97      if (opt::pmd())
98          params.default_type = RGBDGrabberFactory::PMD;
99
100     if (opt::calibration_file())
101         params.calibration_file = opt::calibration_file();
102
103     if (opt::high_resolution())
104         params.high_resolution = true;
105
106     if (opt::sync())
107         params.synchronous = true;
108
109     std::vector<RGBDGrabberFactory::GrabberData> grabbers;
110     grabbers = grabber_factory.createGrabbers(params);
111
112     if (grabbers.size() < 1)
113     {
114         QMessageBox::critical(0, "Fatal error",
115                               "Cannot connect to any RGBD device.\n\nPlease check
116         return 1;
117     }
118
119     RGBDGrabber* grabber = grabbers[0].grabber;
120     RGBDProcessor* processor = grabbers[0].processor;
121
122     bool ok = grabber->connectToDevice();
123     if (!ok)
124     {
125         ntk_dbg(0) << "WARNING: connectToDevice failed.";
126         return 1;
127     }
128
129     RGBDFrameRecorder frame_recorder (opt::dir_prefix());
130     frame_recorder.setFrameIndex(opt::first_index());
131     frame_recorder.setSaveOnlyRaw(true);
132     frame_recorder.setUseBinaryRaw(true);
133     frame_recorder.setSavePCLPointCloud(opt::savePCD());
134
135     ObjectDetector detector;
136
137     MeshGenerator* mesh_generator = 0;
138
139     ntk::RGBDCalibrationPtr calib_data = grabber->calibrationData();
140
141     if (calib_data)
142     {
143         mesh_generator = new MeshGenerator();
144     }
```

```
145
146     GuiController gui_controller (*grabber, *processor);
147     grabber->addEventListener(&gui_controller);
148     gui_controller.setFrameRecorder(frame_recorder);
149     gui_controller.setObjectDetector(detector);
150
151     if (opt::sync())
152         gui_controller.setPaused(true);
153
154     if (mesh_generator)
155     {
156         mesh_generator->setUseColor(true);
157         gui_controller.setMeshGenerator(*mesh_generator);
158     }
159
160     grabber->start();
161
162     app.exec();
163     delete mesh_generator;
```

View3dWindow.cpp

```
20  #include "View3dWindow.h"
21  #include "ui_View3dWindow.h"
22
23  #include "GuiController.h"
24
25  #include <ntk/mesh/mesh_generator.h>
26
27  View3DWindow::View3DWindow(GuiController& controller, QWidget *parent) :
28      QMainWindow(parent),
29      ui(new Ui::View3DWindow),
30      m_controller(controller)
31  {
32    ui->setupUi(this);
33    ui->mesh_view->enableLighting();
34  }
35
36  View3DWindow::~View3DWindow()
37  {
38    delete ui;
39  }
40
41  void View3DWindow::on_resetCamera_clicked()
42  {
43    ui->mesh_view->resetCamera();
44    ui->mesh_view->updateGL();
45  }
46
47  void View3DWindow::on_colorMappingCheckBox_toggled(bool checked)
48  {
49    m_controller.meshGenerator()->setUseColor(checked);
50  }
51
52  void View3DWindow::on_pointCloudPushButton_clicked()
53  {
54    m_controller.meshGenerator()->setMeshType(ntk::MeshGenerator::PointCloudMesh);
55  }
```

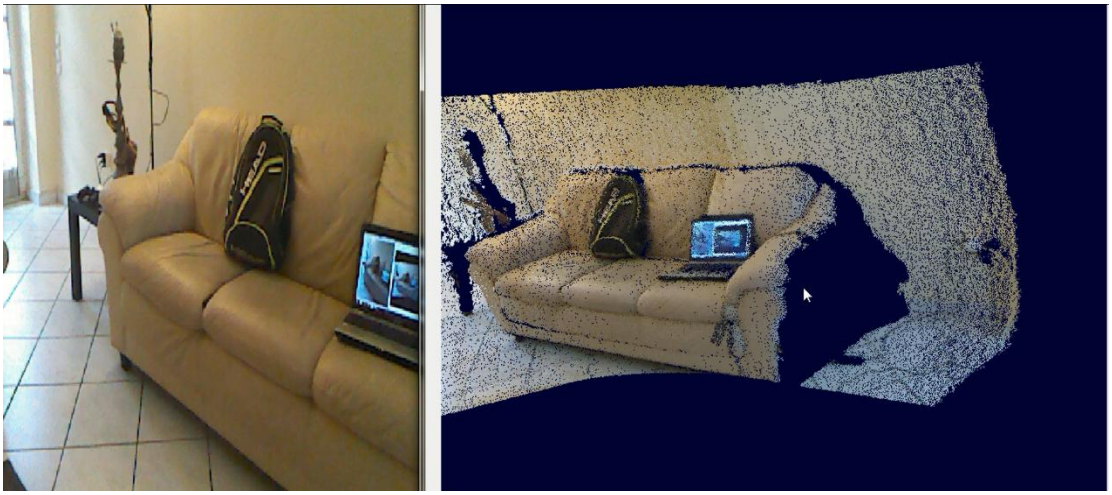
```
57  void View3DWindow::on_surfelsPushButton_clicked()
58  {
59    m_controller.meshGenerator()->setMeshType(ntk::MeshGenerator::SurfelsMesh);
60  }
61
62  void View3DWindow::on_trianglePushButton_clicked()
63  {
64    m_controller.meshGenerator()->setMeshType(ntk::MeshGenerator::TriangleMesh);
65  }
66
67  void View3DWindow::on_saveMeshPushButton_clicked()
68  {
69    m_controller.meshGenerator()->mesh().saveToPlyFile("current_mesh.ply");
70  }
71
```

Εικόνες

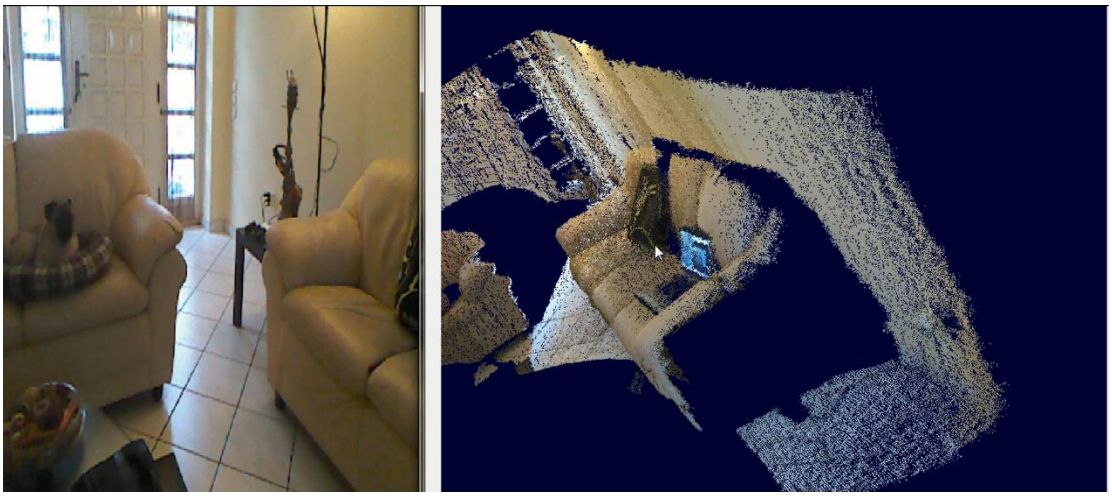
Παρατίθεται σειρά εικόνων που απεικονίζουν τον τρόπο λειτουργίας της εφαρμογής:

Εικόνα 1: Το πρώτο καρέ
Πηγή: Ίδια εικόνα

Εικόνα 2: Βελτίωση νέφους σημείων προϊόντος του χρόνου
Πηγή: Ίδια εικόνα



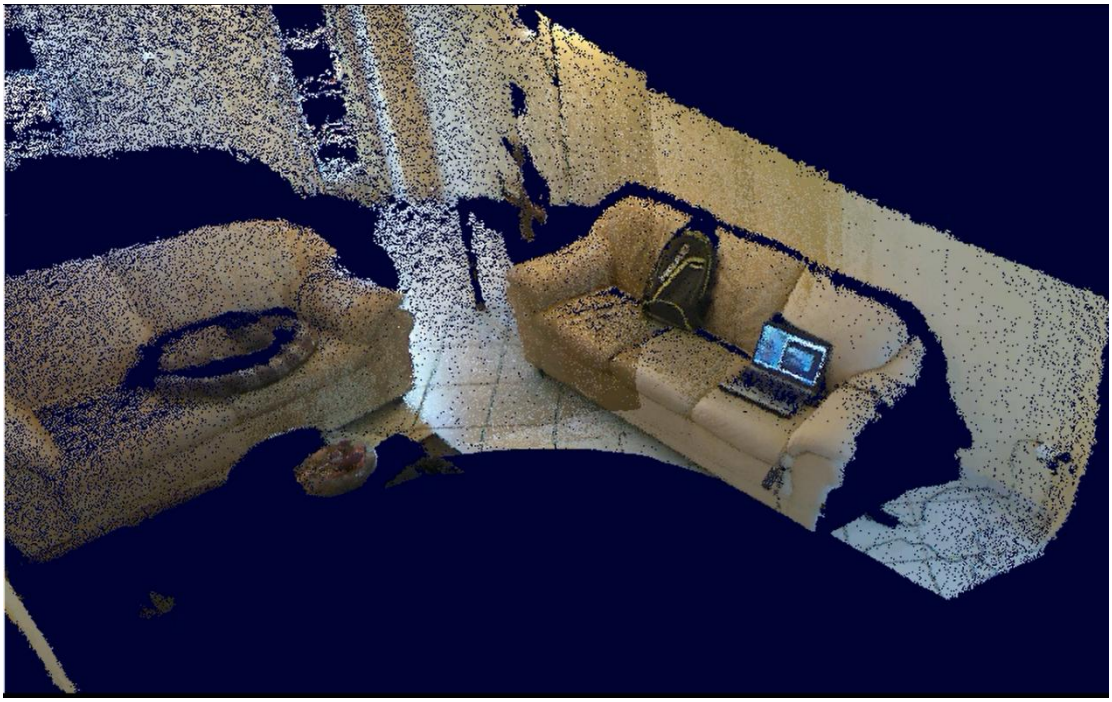
Εικόνα 3: Επόμενες λήψεις
Πηγή: Ίδια εικόνα



Εικόνα 4: Το μοντέλο προϊόντος του χρόνου
Πηγή: Ίδια εικόνα

Εικόνα 5: Επόμενες λήψεις
Πηγή: Ίδια εικόνα

Εικόνα 6: Εναλλακτική όψη
Πηγή: Ίδια εικόνα



Εικόνα 7: Εναλλακτική όψη
Πηγή: Ίδια εικόνα