Pointclouds Project

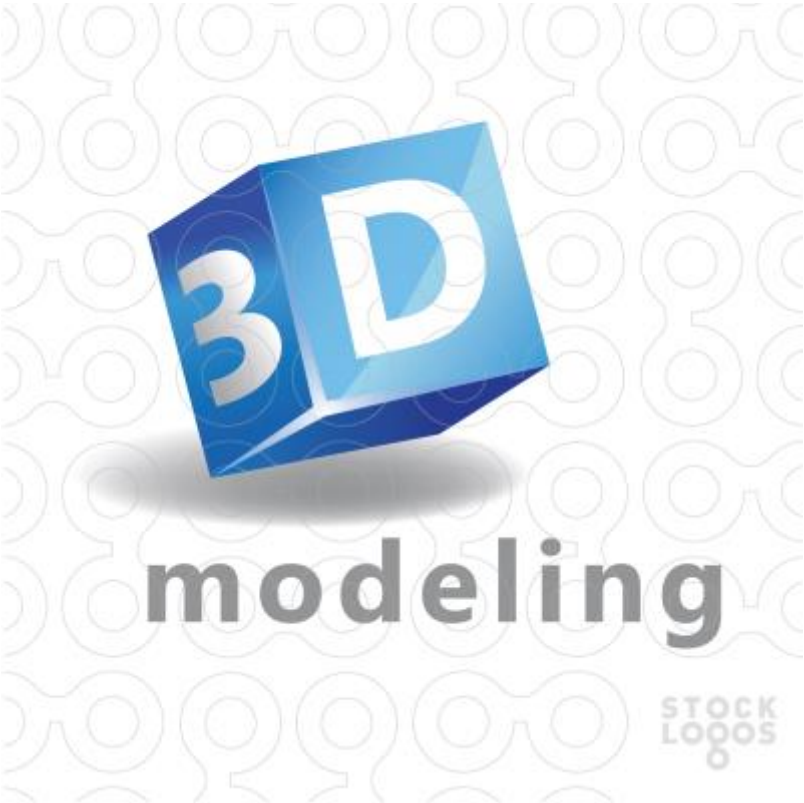
Υπολογιστική Γεωμετρία&

Εφαρμογές 3Δ Μοντελοποίησης

Αθανασόπουλος Νικόλαος Α.Μ.:228263

Απαλλακτική Εργασία 4

Ανακατασκευή επιφάνειας από αλληλουχία εικόνων βάθους



➢ Μέρος Α:

●Ερώτημα 1ο

Φορτώστε την αλληλουχία εικόνων βάθους σε ένα διάνυσμα F_i={F_1,F_2,...F_N} και δημιουργήστε τα αντίστοιχα 3Δ νέφη σημείων C_i={C_1,C_2,... C_N} (*Θεωρήστε κατακόρυφο/οριζόντιο οπτικό πεδίο (fov) της κάμερας 48.6ο & 62ο αντίστοιχα.*). Δώστε δυνατότητα στον χρήση να απεικονίσει ένα συγκεκριμένο νέφος σημείων.

Στην περίπτωσή μας εργαζόμαστε με το σύνολο των εικόνων που υπάρχουν στον φάκελο meeting_small_1 που είναι στο πλήθος 179 , και πιο συγκεκριμένα η αλληλουχία των φακέλων που διατρέχουμε για να φτάσουμε στον φάκελο meeting_small_1 είναι η εξής:

resources/rgbd-scenes/meeting_small/meeting_small_1/

Αρχικά φορτώνουμε τις εικόνες βάθους σε ένα διάνυσμα $FI_i = \{F_1 , F_2 , \dots , F_N\}$ όπου N=179 είναι το πλήθος των εικόνων, βρίσκοντας το μέγιστο βάθος από όλες τις εικόνες maxDepth , το οποίο θα χρησιμοποιήσουμε για την κανονικοποίηση των εικόνων βάθους. Στη συνέχεια, φορτώνουμε τις εικόνες χρώματος rgb σε ένα διάνυσμα $I_i = \{I_1 , I_2 , \dots , I_N\}$ και προχωράμε στην κανονικοποίηση (normalization) των εικόνων βάθους .

Ακολούθως ,χρησιμοποιώντας την συνάρτηση PC(const cv::Mat& depth, std::vector<vec> &vertices) , η οποία δέχεται σαν όρισμα την εκάστοτε εικόνα βάθους depth και μας επιστρέφει τα vertices της εικόνας αυτής , κατασκευάζουμε τα 3Δ νέφη σημείων $C_i = \{C_1 , C_2 , \dots , C_N\}$. Για να το πετύχουμε αυτό σχηματίζουμε 3Δ σημεία (Point3D) χρησιμοποιώντας τις συντεταγμένες των κόμβων που μας επιστρέφει η συνάρτηση PC.

▫ Αναλυτικότερα στην συνάρτηση PC υπολογίζουμε τις συντεταγμένες X,Y και Z για κάθε σημείο του 3δ νέφους και η μεθοδολογία είναι η εξής:

Ισχύει η σχέση:

$$FieldofView = 2 * \arctan(\frac{d}{2 * Focal\ Length})$$

Όπου $d = film\ width = 1$

Από την παραπάνω σχέση προκύπτει:

$$Focal\ Length = \frac{d}{(2 * \tan\left(\frac{FieldofView}{2}\right))} = \frac{1}{(2 * \tan\left(\frac{FieldofView}{2}\right))}$$

Από την εκφώνηση της άσκησης μας δίνονται: $FieldofView_x = 62^o$ και $FieldofView_y = 48.6^o$.Αντικαθιστώντας τις τιμές αυτές βρίσκουμε αντίστοιχα τα $Focal\ Length_x$ και $Focal\ Length_y$. Στην συνέχεια, υπολογίζουμε :

$f_x = ResX * Focal\ Length_x$ και $f_y = ResY * Focal\ Length_y$

,όπου ResX είναι η ανάλυση της εικόνας ως προς τον οριζόντιο άξονα και ισούται με τις στήλες της εικόνας βάθους depthimage , ενώ ResY είναι η ανάλυση της εικόνας ως προς τον κατακόρυφο άξονα και ισούται με τις γραμμές της εικόνας βάθους depthimage.

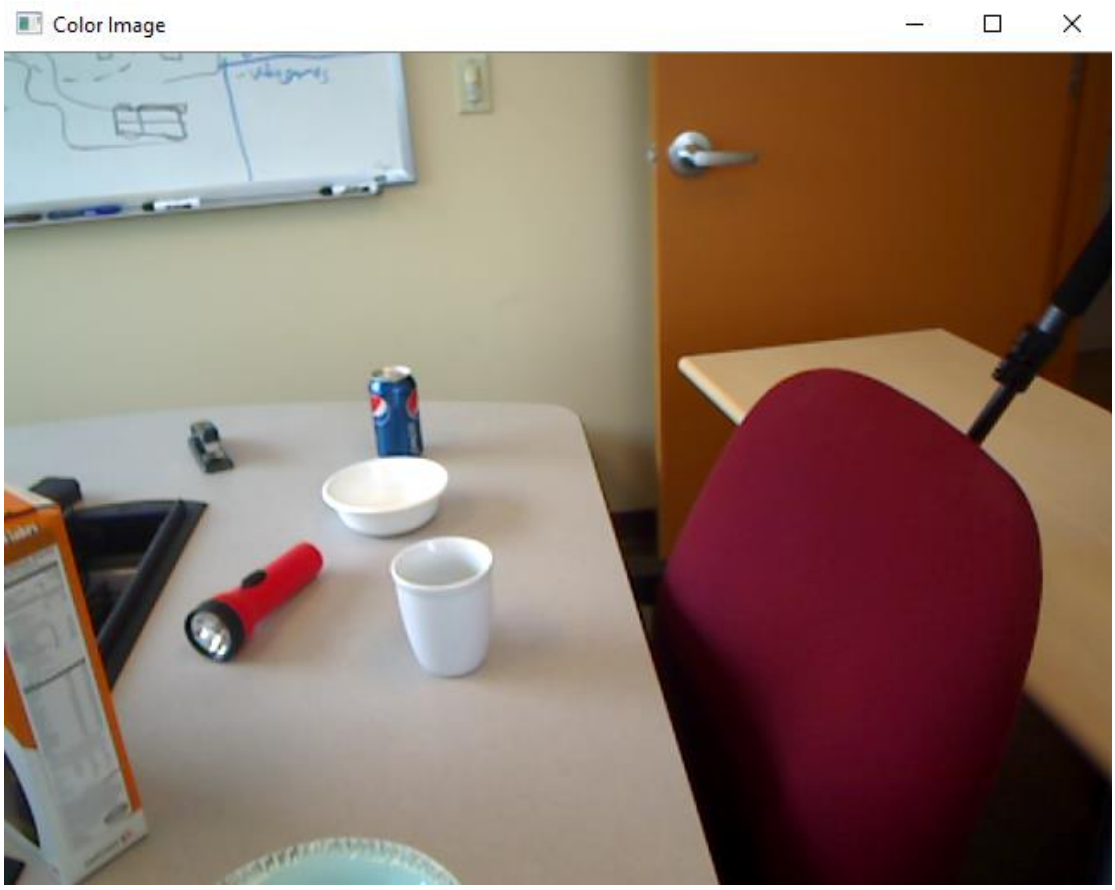
Τελικά , αν (V,U) είναι οι συντεταγμένες του κάθε pixel της εικόνας βάθους depthimage , οι συντεταγμένες X,Y και Z του κάθε σημείου υπολογίζονται από τις σχέσεις:

$$Z = depth(V, U)$$
$$X = (V - V_o) * \frac{Z}{f_x}$$
$$Y = (U - U_o) * \frac{Z}{f_y}$$

<u>Σημείωση Νο1:</u> Ούτως ώστε να εκτελείται το πρόγραμμά μας γρηγορότερα δειγματοληπτούμε εντός της συνάρτησης PC τα σημεία που μας επιστρέφει με ένα βήμα δειγματοληψίας D=5 και έτσι ο συνολικός αριθμός των σημείων που απεικονίζονται είναι μικρότερος από τον πραγματικό. Εάν θελήσουμε να δούμε όλα τα σημεία που υπάρχουν αρκεί να θέσουμε μηδενικό βήμα δειγματοληψίας D=0. Σε αυτή την περίπτωση όμως ο χρόνος εκτέλεσης του προγράμματος θα είναι αρκετά μεγαλύτερος.

◦ Παράλληλα ,δίνεται η δυνατότητα στον χρήστη μέσω της πρώτης μπάρας να επιλέγει ποιο νέφος σημείων επιθυμεί να απεικονίσει . Τον αριθμό του νέφους που επέλεξε να απεικονίσει ο χρήστης μας δείχνει η μεταβλητή N. Τα διάφορα νέφη σημείων που απεικονίζονται με πράσινο χρώμα , καθώς και οι αντίστοιχες εικόνες χρώματος παρουσιάζονται παρακάτω για διαφορετικές τιμές της μεταβλητής N:

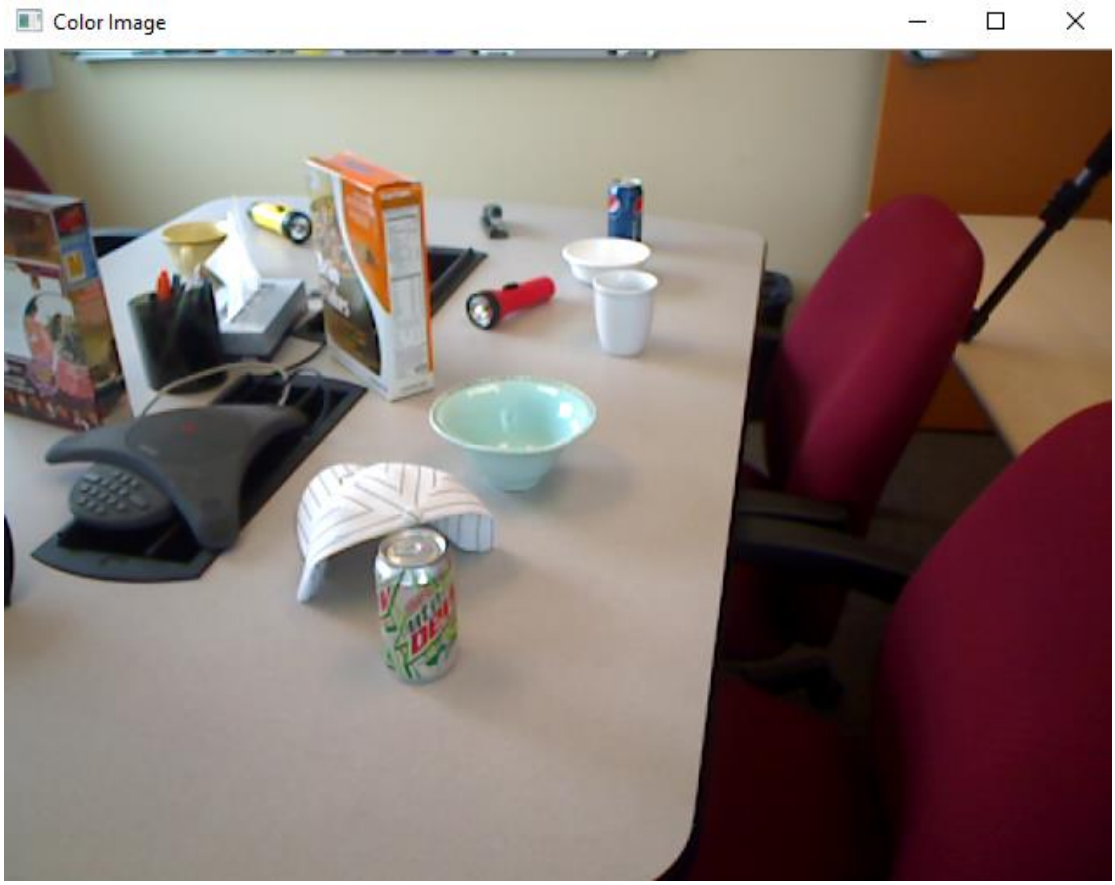
◦ N=7



Εικόνα No1

Εικόνα No2

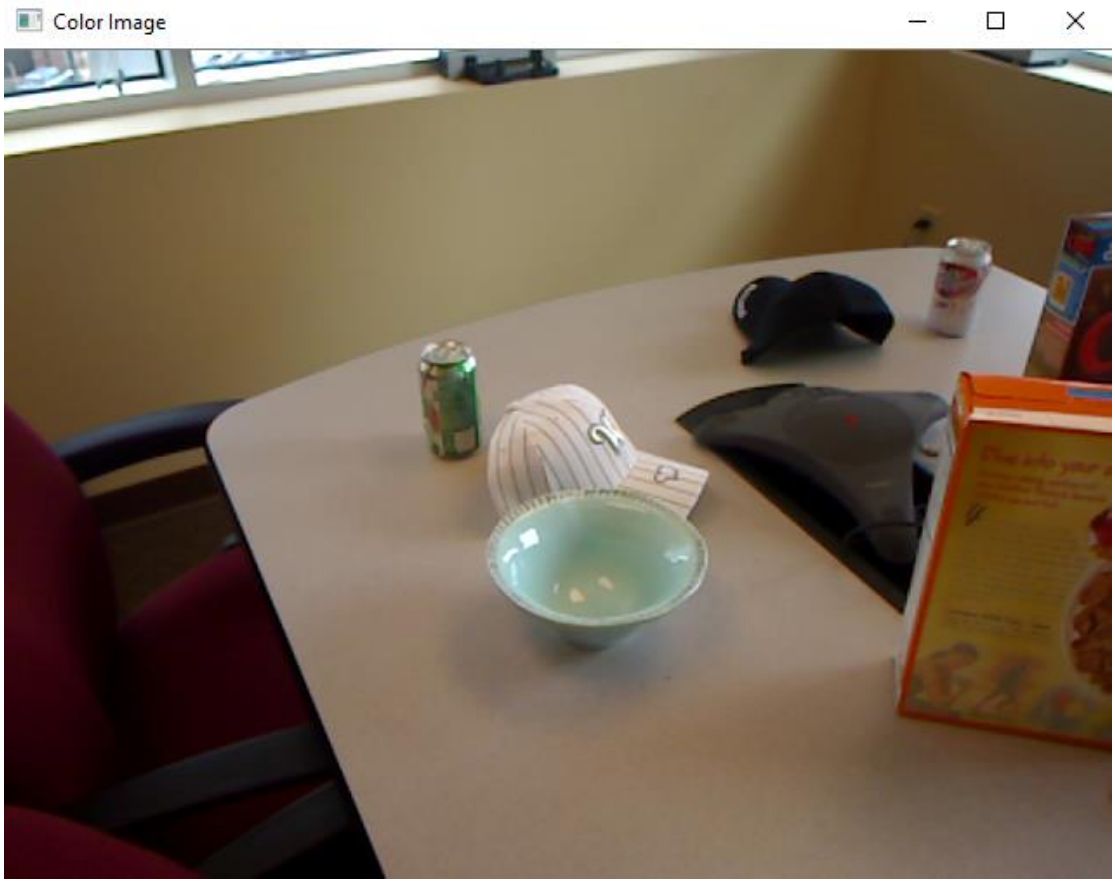
◦ N=34



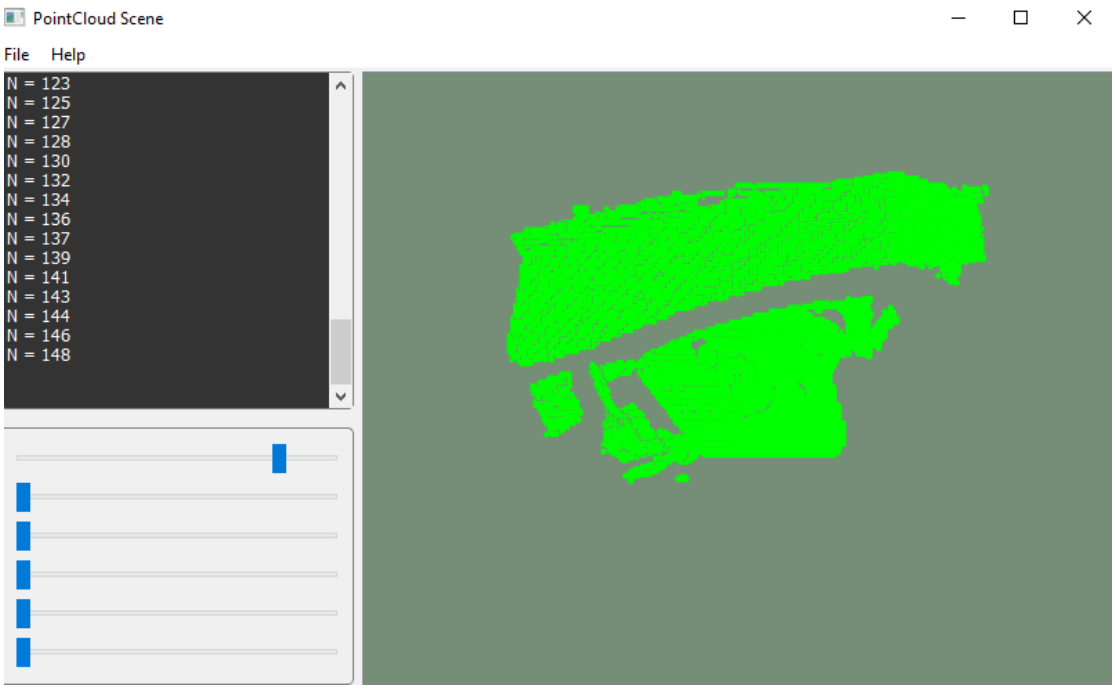
Εικόνα No3

Εικόνα No4

◦ N=148



Εικόνα No5



Εικόνα No6

●Ερώτημα 2ο

Δεδομένου νέφους σημείων C_i = {p_1,p_2,...p_n}, υπολογίστε την αντιστοίχιση σημείων με το νέφος C_{i-1}= {q_1,q_2,...q_n}, βρίσκοντας τον πλησιέστερο γείτονα για κάθε σημείο $p_i \in C_i$. Απεικονίστε τα δύο (2) νέφη σημείων με διαφορετικά χρώματα καθώς και μια γραμμή για κάθε σημείο που δείχνει την αντιστοίχιση. Τυπώστε την συνολική απόσταση $e = \Sigma \|p_i - q_i\|_{ni=12}$, την μέση απόσταση, καθώς και τον χρόνο υπολογισμού της αντιστοίχισης σημείων.

Για να μεταβούμε από το πρώτο ερώτημα στο δεύτερο αρκεί να πατήσουμε το πλήκτρο 'd' από το πληκτρολόγιο. Το νέφος $C_i = \{p_1 , p_2 , \dots , p_n\}$ επιλέγεται από τον χρήστη και πάλι με την χρήση της πρώτης μπάρας και ο αριθμός του νέφους σημείων Ci δείχνεται και πάλι από την μεταβλητή N. Ως νέφος $C_{i-1} = \{q_1 , q_2 , \dots , q_n\}$ θεωρείται (σε περίπτωση που υπάρχει προηγούμενο νέφος από το N) το νέφος N-1 , όπου N είναι ο αριθμός που επιλέγει ο χρήστης. Σε περίπτωση που δεν υπάρχει προηγούμενο νέφος από το N , δηλαδή N=0 , τότε στην οθόνη μας φαίνεται μόνο το νέφος N=0 και δεν εκτελείται η διαδικασία του ερωτήματος 2.

- Εφόσον επιλέχθηκε το νέφος σημείων C_i , και κατά συνέπεια το νέφος σημείων C_{i-1} , εξάγουμε με την βοήθεια της συνάρτησης PC τα vertices των αντίστοιχων νεφών σημείων . Στη συνέχεια , υπολογίζουμε για κάθε σημείο του νέφους C_i τον πλησιέστερο γείτονα του νέφους C_{i-1} . Για να το πετύχουμε αυτό υπολογίζουμε για κάθε σημείο p_i του νέφους σημείων N τις Ευκλείδειες αποστάσεις από κάθε σημείο q_i του προηγούμενου νέφους σημείων N-1 . Ως πλησιέστερος γείτονας ορίζεται εκείνο το σημείο q του νέφους C_{i-1} που απέχει την μικρότερη απόσταση από το σημείο του νέφους C_i που ελέγξαμε. Το q αποτελεί το αντίστοιχο σημείο του p_i και συνεπώς σχεδιάζουμε ένα ευθύγραμμο τμήμα μεταξύ των σημείων αυτών που μας δείχνει την αντιστοίχιση. Το εν λόγω ευθύγραμμο τμήμα απεικονίζεται με μπλε χρώμα.

- Κάθε φορά που υπολογίζεται ο πλησιέστερος γείτονας q ενός σημείου p_i , αθροίζουμε την μεταξύ τους απόσταση και την κρατάμε στην μεταβλητή SinApostasi . Όταν υπολογίσουμε για κάθε σημείο p_i του νέφους C_i τον αντίστοιχο πλησιέστερο γείτονα η μεταβλητή SinApostasi θα περιέχει την συνολική απόσταση του κάθε σημείου p_i από το αντίστοιχό του , δηλαδή:

$$SinApostasi = \sum_{i=1}^{n} ||p_i - q_i||^2$$

Για να βρούμε την μέση απόσταση εκτελούμε την πράξη:

$$MesiApostasi = \frac{SinApostasi}{N1}$$

, όπου N1:πλήθος των vertices του νέφους σημείων C_i.

<u>Σημείωση Νο2:</u> Το ερώτημα 2 εκτελείται μία φορά για δεδομένη τιμή της μεταβλητής N, δηλαδή για δεδομένα νέφη σημείων C_N και C_{N-1} . Συνεχίζουν όμως να γίνονται draw η ευθυγράμμιση και τα δύο Pointclouds έως ότου ο χρήστης αλλάξει ερώτημα πατώντας πλήκτρο που αντιστοιχεί σε άλλο ερώτημα είτε αλλάξει την τιμή της μεταβλητής N από την πρώτη μπάρα . Στην τελευταία περίπτωση , δηλαδή αν αλλάξει το N και γίνει N' , το ερώτημα 2 εκτελείται μία φορά για τα καινούργια νέφη σημείων $C_{N'}$ και $C_{N',-1}$ και συνεχίζουν να γίνονται draw η νέα ευθυγράμμιση και τα δύο νέα Pointclouds μέχρι ο χρήστης και πάλι να προβεί σε κάποια από τις προαναφερθείσες κινήσεις.

▫ Στη συνέχεια , παρατίθενται μία σειρά εικόνων για διαφορετικές τιμές της μεταβλητής N στις οποίες εμπεριέχονται:

-Η απεικόνιση του νέφους σημείων C_N γίνεται με κόκκινο χρώμα.

- Η απεικόνιση του νέφους σημείων C_{N-1} γίνεται με κίτρινο χρώμα.

-Η απεικόνιση της αντιστοίχισης των σημείων p_i του νέφους σημείων C_N με τα αντίστοιχα (πλησιέστερα) σημεία q του νέφους σημείων C_{N-1} γίνεται με μπλε χρώμα.

-Η εκτύπωση του χρόνου υπολογισμού της αντιστοίχισης σημείων.

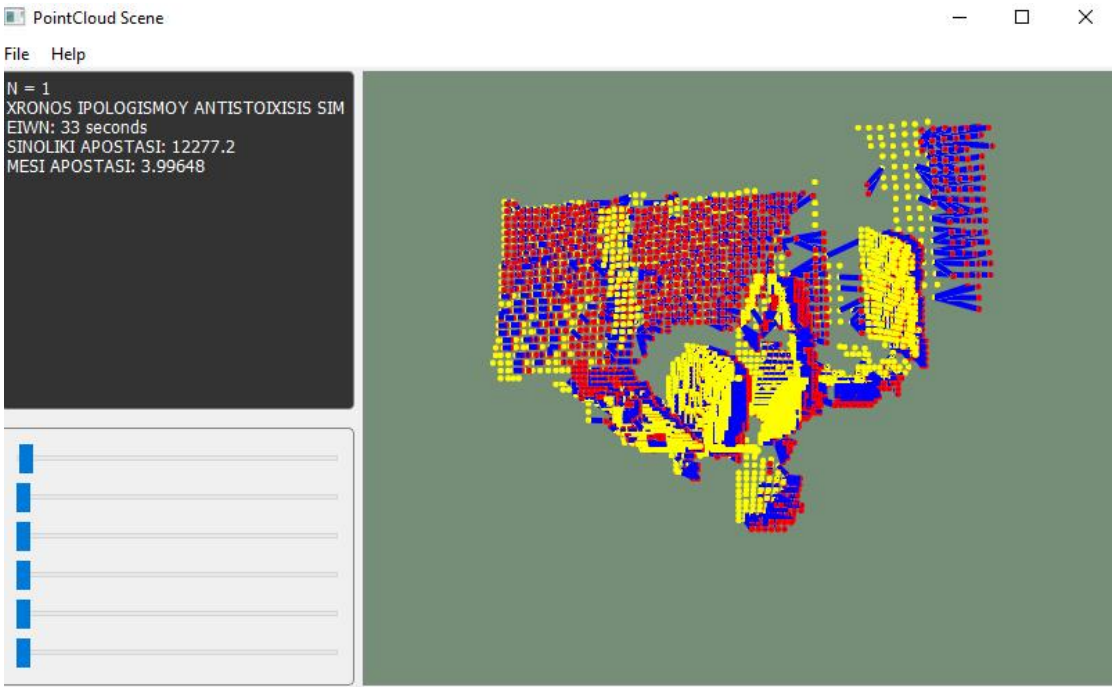
-Η εκτύπωση της συνολικής απόστασης.

-Η εκτύπωση της μέσης απόστασης.

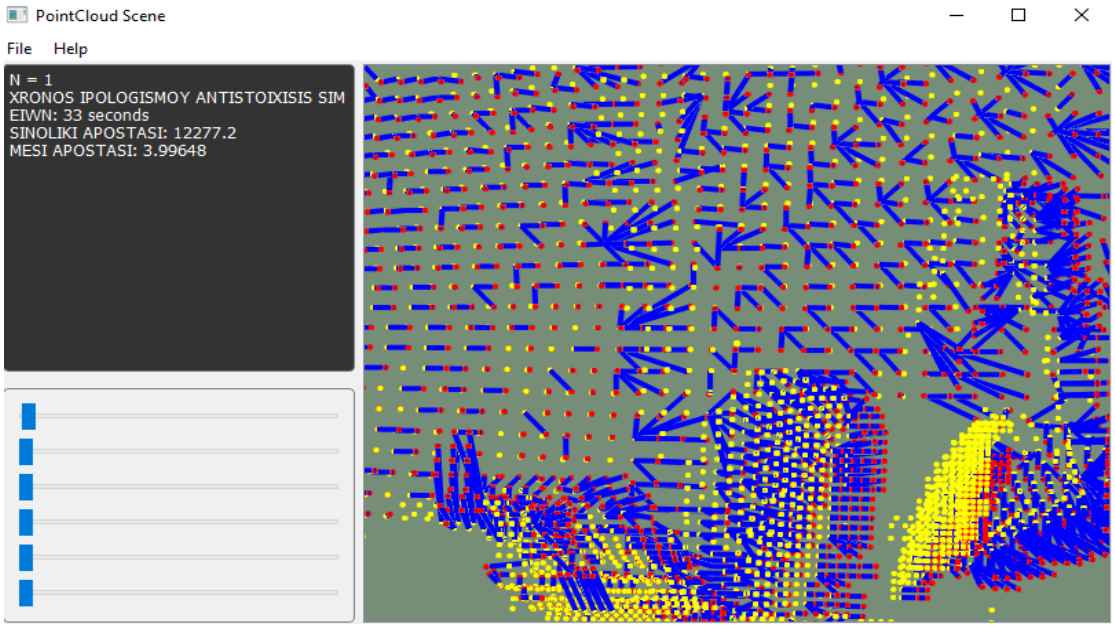
<u>Σημείωση Νο3:</u>

Ως βήμα δειγματοληψίας έχει θεωρηθεί D=10 στο συγκεκριμένο ερώτημα.

◦ N=1
Χρόνος υπολογισμού αντιστοίχισης σημείων: 33seconds
Συνολική απόσταση: 12277.2
Μέση απόσταση: 3.99648



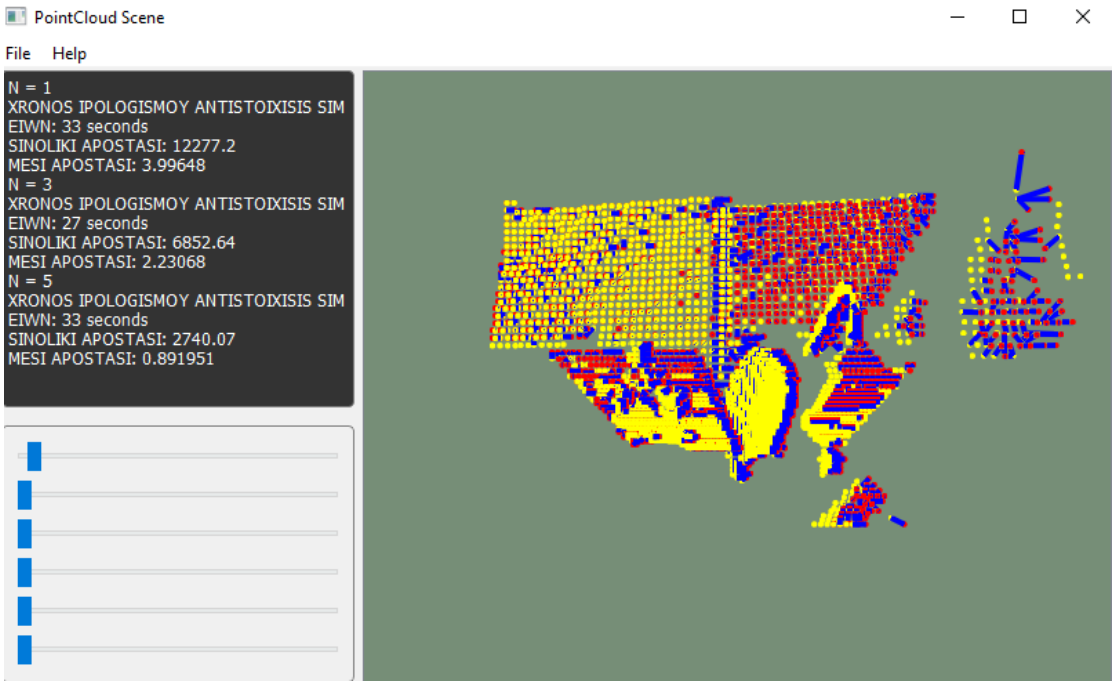
Εικόνα Νο7



Εικόνα Νο8

◦ N=5

Χρόνος υπολογισμού αντιστοίχισης σημείων: 33seconds
Συνολική απόσταση: 2740.07
Μέση απόσταση: 0.891951



Εικόνα No9

Εικόνα No10

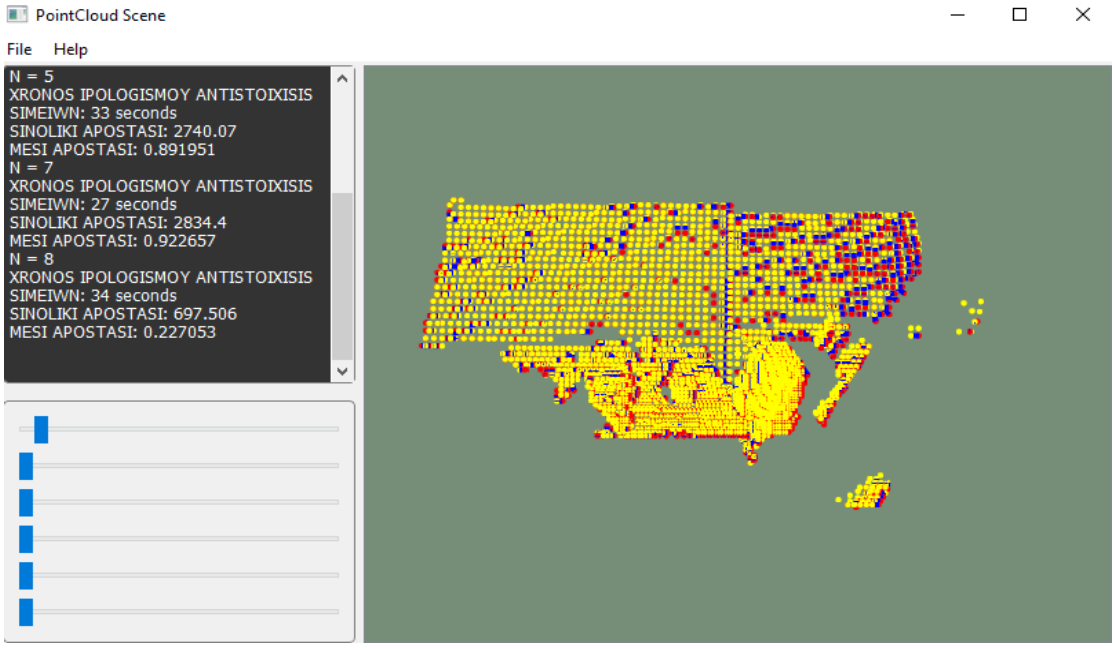
◦ N=7

Χρόνος υπολογισμού αντιστοίχισης σημείων: 27seconds
Συνολική απόσταση: 2834.4
Μέση απόσταση: 0.922657

Εικόνα No11

◦ N=8

Χρόνος υπολογισμού αντιστοίχισης σημείων: 34seconds
Συνολική απόσταση: 697.506
Μέση απόσταση: 0.227053



Εικόνα No12

➢ Μέρος Β:

●Ερώτημα 3ο

Δημιουργήστε μια συνάρτηση η οποία υπολογίζει την περιστροφή $\boldsymbol{R} \in \mathbb{R}^{3x3}$ και την μετατόπιση $\boldsymbol{t}$ ώστε να ευθυγραμμίζει τα νέφη σημείων του προηγούμενου ερωτήματος, δηλαδή ελαχιστοποιεί την συνάρτηση :
$\boldsymbol{e = \sum_{i=1}^{n} ||(R * p_i + t) - q_i||^2}$.

Ούτως ώστε να δημιουργήσουμε μια συνάρτηση ,η οποία να υπολογίζει την περιστροφή R και την μετατόπιση t για την ευθυγράμμιση δύο νεφών σημείων , κατασκευάζουμε αρχικά μια δομή struct με το όνομα RotTran. Η δομή αυτή έχει ένα μέλος R για την περιστροφή και ένα μέλος t για την μετατόπιση. Ακολούθως, κατασκευάζουμε την συνάρτηση rotran της δομής RotTran στην οποία πραγματοποιούνται οι απαραίτητοι υπολογισμοί και έχει την εξής δομή:

```
RotTran rotran(vvr::PointCloud pointc1, vvr::PointCloud pointc2)
```

Όπως είναι εμφανές η συνάρτηση rotran δέχεται σαν ορίσματα δύο νέφη σημείων, όπου p_i είναι τα σημεία του Pointcloud1 και q_i είναι τα σημεία του Pointcloud2 (Pointclouds). Η μεθοδολογία που ακολουθήθηκε για τον υπολογισμό της περιστροφής R και της μετατόπισης t , ούτως ώστε να πετύχουμε την ευθυγράμμιση των δύο νεφών σημείων που δέχεται η συνάρτηση ως ορίσματα, αποτελείται από 5 βήματα τα οποία παρουσιάζονται αναλυτικά στη συνέχεια.

Σημείωση Νο4:

Για την μεθοδολογία που ακολουθεί θεωρήθηκε βάρος w=0.1 .

❖ ΒΗΜΑ 1ο

Υπολογισμός των κέντρων μάζας με χρήση βάρους w για κάθε ένα από τα νέφη σημείων

Τα κέντρα βάρους υπολογίζονται από τις εξής σχέσεις:

▫Για το Pointcloud pointc1:

$$\bar{p} = \frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i}$$

▫Για το Pointcloud pointc2:

$$\bar{q} = \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i}$$

❖ ΒΗΜΑ 2ο

Μετατόπιση των σημείων ως προς το κέντρο μάζας

▫Για το Pointcloud pointc1:

$$x_i = p_i - \bar{p}$$

▫Για το Pointcloud pointc2:

$$y_i = q_i - \bar{q}$$

❖ ΒΗΜΑ 3ο

Υπολογισμός του 3x3 πίνακα S

Ο πίνακας S ισούται με: $S = XWY^T$, όπου:

X : 3xn πίνακας που έχει στις στήλες τα στοιχεία x_i

Y : 3xn πίνακας που έχει στις στήλες τα στοιχεία y_i

W: ο διαγώνιος πίνακας W=diag(w_1,w_2 ,… ,w_N)

με $w_1 = w_2 = \cdots = w_N = w = 0.1$

<u>Σημείωση Νο5:</u>

Στην περίπτωσή μας το Βήμα 2 εμπεριέχεται στο Βήμα 3 καθώς υπολογίζεται κατ'ευθείαν ο πίνακας XW ως εξής:

$$XW = x_i * w = (p_i - \bar{p}) * w$$

❖ ΒΗΜΑ 4ο

Υπολογισμός της περιστροφής R

Σε πρώτο στάδιο αποσυνθέτουμε τον 3x3 πίνακα S δημιουργώντας τους πίνακες U,Σ και V χρησιμοποιώντας την εντολή svd , όπου $S = U\Sigma V^T$. Οι πίνακες U,V και Σ έχουν διαστάσεις 3x3 και ο πίνακας Σ έχει στην διαγώνιο άσσους , ενώ τα υπόλοιπα στοιχεία είναι μηδενικά. Ούτως ώστε να αποφύγουμε το φαινόμενο mirroring σχηματίζουμε εκ νέου τον πίνακα Σ ως εξής:

$$\Sigma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{bmatrix}$$

Τελικά , ο πίνακας R προκύπτει ως εξής:

$$R = V\Sigma U^T$$

❖ ΒΗΜΑ 5ο

Υπολογισμός της Μεταφοράς t

Η μεταφορά t , εφόσον έχει υπολογιστεί η περιστροφή R υπολογίζεται με τον εξής τρόπο:

$$t = \bar{q} - R * \bar{p}$$

- Μετά το πέρας αυτής της διαδικασίας θα έχει προκύψει μια τιμή R για την περιστροφή και μια τιμή t για την μετατόπιση του Pointcloud1 , έτσι ώστε να ευθυγραμμιστεί με το Pointcloud2. Με άλλα λόγια , εάν εφαρμόσουμε την μετατόπιση και την περιστροφή στα σημεία του πρώτου νέφους σημείων ως εξής:

$p_i' = R * \mathrm{p_i} + \mathrm{t}$, όπου p_i' τα ευθυγραμμισμένα σημεία του Pointcloud1

, τότε θα ελαχιστοποιηθεί η συνάρτηση :

$e = \sum_{\mathrm{i}=1}^{\mathrm{n}} \left|\left|(R * \mathrm{p_i} + \mathrm{t}) - \mathrm{q_i}\right|\right|^2 = \sum_{\mathrm{i}=1}^{\mathrm{n}} \left|\left|\mathrm{p}_i' - \mathrm{q_i}\right|\right|^2$, δηλαδή τα δύο νέφη σημείων θα έχουν ευθυγραμμιστεί.

●Ερώτημα 4ο

Χρησιμοποιήστε την συνάρτηση του προηγούμενου ερωτήματος για να ευθυγραμμίσετε τα νέφη C_i έως C_j για i,j είσοδο από τον χρήστη (π.χ. C_2 έως C_{24}). Δημιουργήστε το ολικό νέφος M ως σύνολο των ευθυγραμμισμένων νεφών αφαιρώντας τα διπλά σημεία μετά την ευθυγράμμιση, δηλαδή τα σημεία που έχουν απόσταση από το αντίστοιχο τους $d < \varepsilon$, όπου ε κάποια αυθαίρετη ανοχή. Απεικονίστε το ολικό νέφος M και τυπώστε τον χρόνο υπολογισμού του για διάφορες εισόδους. Πως μπορούμε να αποφασίσουμε μία «καλή» ανοχή ε αυτόματα?

Για να μεταβούμε από το δεύτερο ερώτημα στο τέταρτο αρκεί να πατήσουμε το πλήκτρο 't' από το πληκτρολόγιο. Το νέφος C_i επιλέγεται από τον χρήστη με την χρήση της δεύτερης μπάρας και ο αριθμός του νέφους σημείων C_i δείχνεται από την μεταβλητή Ni , ενώ το νέφος C_j επιλέγεται από τον χρήστη με την χρήση της τρίτης μπάρας και ο αριθμός του νέφους σημείων C_j δείχνεται από την μεταβλητή Nj .

Μεθοδολογία για τον υπολογισμό του ολικού νέφους M

-Πρώτη φορά εκτέλεσης του προγράμματος για το ερώτημα 4 – First Pass

Την πρώτη φορά που εκτελείται το πρόγραμμα κρατάμε σε ένα διάνυσμα vtr τα σημεία του pointcloud N_i ,τα οποία απεικονίζονται στην οθόνη μας και βρίσκουμε τα πλησιέστερα σημεία του pointcloud N_i ως προς το pointcloud N_{i+1} , δηλαδή τα σημεία του νέφους σημείων N_i που βρίσκονται πιο κοντά στα σημεία του νέφους σημείων N_{i+1}. Στην συνέχεια , χρησιμοποιώντας την συνάρτηση rotran που κατασκευάστηκε στο προηγούμενο ερώτημα ευθυγραμμίζουμε τα σημεία του νέφους C_{Ni+1} στα πλησιέστερα σημεία που υπολογίσαμε. Στην συνέχεια, και μόνο την πρώτη φορά εκτέλεσης του ερωτήματος 4, υπολογίζουμε την ανοχή ε ως την μέση απόσταση των σημείων του νέφους C_{Ni+1} από τα πλησιέστερά τους ως προς το νέφος C_{Ni}. Έπειτα, αυτό που μας ζητείται από την εκφώνηση της άσκησης είναι να αφαιρέσουμε τα διπλά σημεία μετά την ευθυγράμμιση, δηλαδή τα σημεία που έχουν απόσταση από το αντίστοιχο τους: $d < \varepsilon$. Για να το πετύχουμε αυτό υπολογίζουμε για τα ευθυγραμμισμένα σημεία που προέκυψαν τα πλησιέστερά τους ως προς τα σημεία του νέφους C_{Ni}. Για κάθε ευθυγραμμισμένο σημείο μετράμε την απόστασή του από το πλησιέστερό του που υπολογίστηκε και την συγκρίνουμε με την ανοχή ε. Ο τρόπος που εργαζόμαστε είναι αντί να αφαιρέσουμε τα διπλά σημεία, να προσθέσουμε στο διάνυσμα vtr τα ευθυγραμμισμένα εκείνα σημεία που δεν είναι διπλά , δηλαδή τα σημεία για τα οποία ισχύει $d > \varepsilon$. Τελικά, το διάνυσμα vtr, το οποίο περιέχει τα σημεία του ολικού νέφους M, θα περιέχει τα σημεία του νέφους σημείων C_{Ni}, καθώς και τα σημεία ευθυγραμμισμένα σημεία του νέφους C_{Ni+1} ως προς το νέφος C_{Ni} , για τα οποία το αντίστοιχό τους απέχει απόσταση d>ε.

-Επόμενες φορές εκτέλεσης του προγράμματος για το ερώτημα 4

Μετά την πρώτη φορά εκτέλεσης του ερωτήματος 4 η διαδικασία που ακολουθείται είναι παρόμοια με την πρώτη φορά, ωστόσο είναι ελαφρώς διαφοροποιημένη. Αρχικά, υπολογίζουμε τα πλησιέστερα σημεία του νέφους C_{i+1} με τα ευθυγραμμισμένα σημεία του νέφους C_i που υπολογίστηκαν στον προηγούμενο κύκλο επανάληψης του προγράμματος. Ακολούθως, ευθυγραμμίζουμε τα σημεία του νέφους C_{i+1} ως προς τα πλησιέστερα ευθυγραμμισμένα σημεία του νέφους C_i. Στο τελευταίο στάδιο της διαδικασίας βρίσκουμε για κάθε ευθυγραμμισμένο σημείο που προέκυψε το αντίστοιχο πλησιέστερό του από όλα τα προηγούμενα σημεία που περιέχει το vtr, δηλαδή τα σημεία του νέφους N_i καθώς και τα σημεία που αναγνωρίστηκαν από την διαδικασία ως μη διπλά στις προηγούμενες επαναλήψεις. Τέλος, για κάθε ευθυγραμμισμένο σημείο μετράμε την απόστασή του από το πλησιέστερό του που υπολογίστηκε και αν η απόστασή τους d είναι μεγαλύτερη από την ανοχή ε, δηλαδή $d > \varepsilon$, τότε το εισάγουμε το ευθυγραμμισμένο σημείο στο διάνυσμα vtr , το οποίο περιέχει όλα τα σημεία του ολικού νέφους M.

Σημείωση Νο5:

Ως ανοχή ε όπως αναφέρθηκε και παραπάνω έχει οριστεί η μέση απόσταση των σημείων του νέφους C_{Ni+1} από τα πλησιέστερά τους ως προς το νέφος C_{Ni}. Να τονιστεί ότι η ανοχή υπολογίζεται μόνο την πρώτη φορά εκτέλεσης του προγράμματος και για όλες τις επόμενες είναι ίδια.

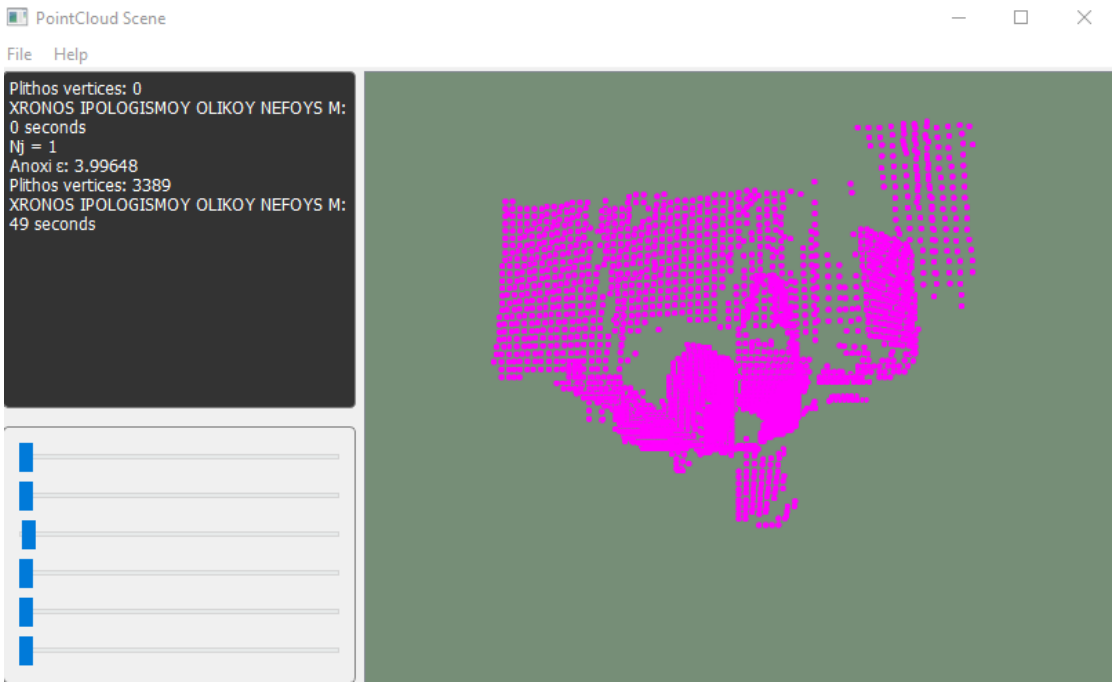
Σημείωση Νο6:

Με αυτό τον τρόπο υπολογισμού του ολικού νέφους M που ακολουθήθηκε προσαρμόζουμε το ολικό νέφος στην καινούργια πληροφορία που έχουμε κάθε φορά με την εισαγωγή επιπρόσθετων νεφών σημείων με την αύξηση της μεταβλητής N_j από την τρίτη μπάρα .

- Ακολούθως παρουσιάζονται τα διαφορετικά ολικά νέφη M που προέκυψαν για διαφορετικά νέφη σημείων C_i και C_j. Ταυτόχρονα παρατίθενται ο συνολικός χρόνος υπολογισμού του ολικού νέφους M για διαφορετικές εισόδους i και j , το πλήθος των σημείων του ολικού νέφους, καθώς και η τιμή της ανοχής ε. Το βήμα δειγματοληψίας που χρησιμοποιήθηκε είναι D=7.

◦ $N_i=0$

$N_j=1$



Εικόνα No13

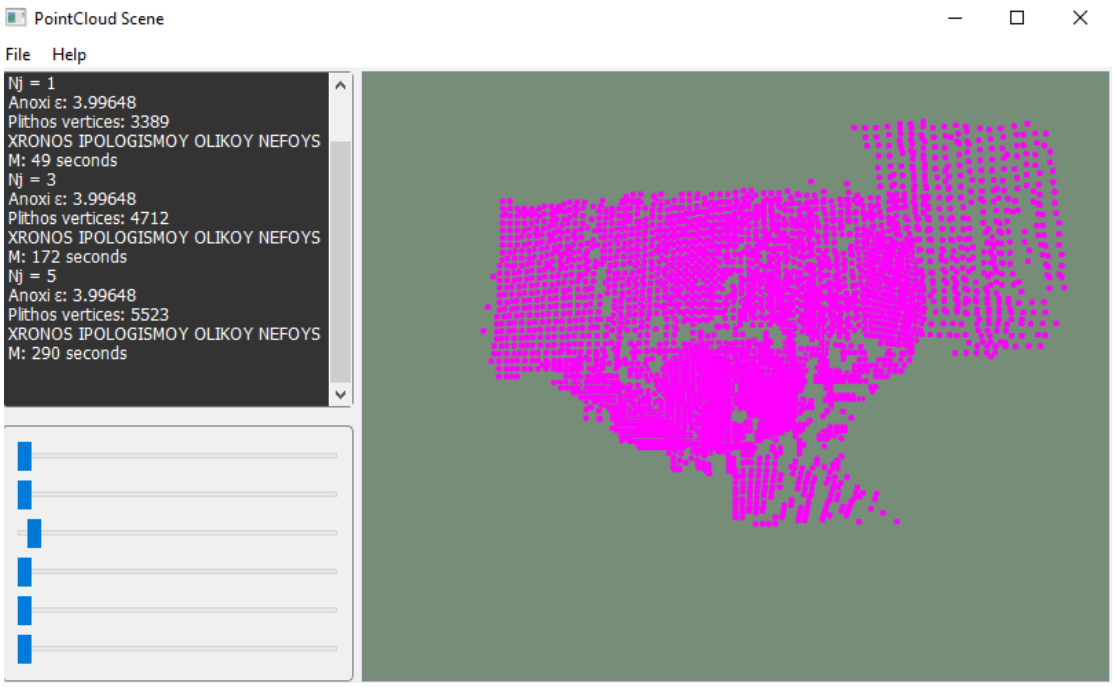
Χρόνος υπολογισμού ολικού νέφους M: 49seconds
Πλήθος σημείων: 3389
Ανοχή ε: 3.99648

$N_j=3$

Εικόνα No14

Χρόνος υπολογισμού ολικού νέφους M: 172seconds
Πλήθος σημείων: 4712
Ανοχή ε: 3.99648

N_j=5

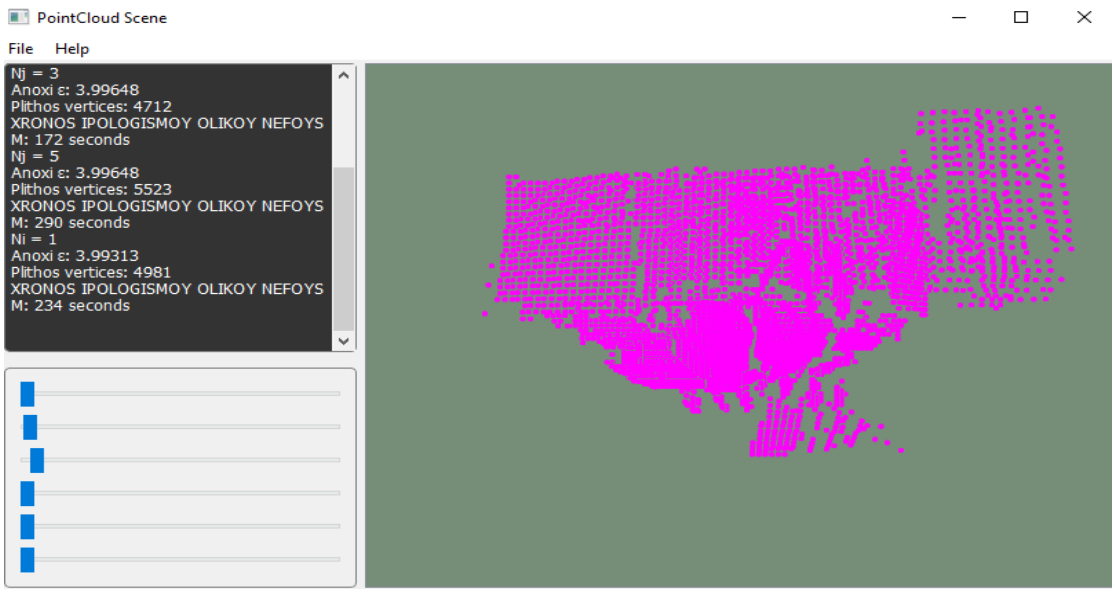


Εικόνα No15

Χρόνος υπολογισμού ολικού νέφους M: 290seconds
Πλήθος σημείων: 5523
Ανοχή ε: 3.99648

- N_i=1

N_j=5



Εικόνα No16

Χρόνος υπολογισμού ολικού νέφους M: 234seconds
Πλήθος σημείων: 4981
Ανοχή ε: 3.99313

●Ερώτημα 5ο

Χρησιμοποιήστε μια δομή δεδομένων διάτμησης χώρου (octree, kd-tree κτλ.) για να επιταχύνετε την αναζήτηση του πλησιέστερου γείτονα και επαναλάβετε το προηγούμενο ερώτημα. Δείξτε τον βαθμό επιτάχυνσης του αλγορίθμου. Τι περιορισμούς μας δημιουργεί?

Στο ερώτημα αυτό μας ζητείται να εκτελέσουμε εκ νέου τα ερωτήματα 2,4 χρησιμοποιώντας μια δομή δεδομένων διάτμησης χώρου , που στην περίπτωσή μας είναι kd-trees. Για την μετάβαση από το τέταρτο στο πέμπτο ερώτημα αρκεί να πατήσουμε το πλήκτρο ‘ f ’ από το πληκτρολόγιο. Επίσης, για να προχωρήσουμε στην εκτέλεση του ερωτήματος 5-α , δηλαδή στην εκτέλεση του ερωτήματος 2 με KD-trees, αρκεί να πατήσουμε το πλήκτρο ‘ l ’ από το πληκτρολόγιο. Αντίστοιχα, για να προχωρήσουμε στην εκτέλεση του ερωτήματος 5-β , δηλαδή στην εκτέλεση του ερωτήματος Β-4 με KD-trees, αρκεί να πατήσουμε το πλήκτρο ‘ q ’ από το πληκτρολόγιο.

❖ Ερώτημα Α 2 με KD-trees

Το νέφος $C_i = \{p_1 , p_2 , \dots , p_n\}$ επιλέγεται από τον χρήστη αυτή την φορά με την χρήση της τέταρτης μπάρας και ο αριθμός του νέφους σημείων Ci δείχνεται από την μεταβλητή N5. Ως νέφος $C_{i-1} = \{q_1 , q_2 , \dots , q_n\}$ θεωρείται (σε περίπτωση που υπάρχει προηγούμενο νέφος από το N5) το νέφος N5-1 , όπου N5 είναι ο αριθμός που επιλέγει ο χρήστης. Σε περίπτωση που δεν υπάρχει προηγούμενο νέφος από το N5 , δηλαδή N5=0 , τότε στην οθόνη μας φαίνεται μόνο το νέφος N5=0 και δεν εκτελείται η διαδικασία του ερωτήματος 5-α.

Το ερώτημα αυτό είναι πανομοιότυπο με το ερώτημα Α2 , μόνο που αυτή την φορά η εύρεση των πλησιέστερων σημείων γίνεται με χρήση KD-trees. Πιο συγκεκριμένα, κατασκευάζουμε ένα δένδρο που περιέχει τα σημεία της εικόνας N5-1 και για κάθε σημείο της εικόνας N5 βρίσκουμε τον πλησιέστερο γείτονα διατρέχοντας το δένδρο που δημιουργήσαμε. Αφού βρούμε τον πλησιέστερο γείτονα σχεδιάζουμε το ευθύγραμμο τμήμα που ορίζεται από το σημείο της εικόνας N5 που εξετάζουμε και τον πλησιέστερο γείτονα που βρέθηκε.

▫ Η εύρεση του πλησιέστερου γείτονα πραγματοποιείται με την βοήθεια της συνάρτησης:

```
NearestNeighboorKDtree(const vec& test_pt, const KDNode* root, const KDNode* &nn,
float& best_dist)
```

Η συνάρτηση αυτή δέχεται ως ορίσματα ένα test_pt , που είναι το σημείο του οποίου θέλουμε να βρούμε τον πλησιέστερο γείτονα , καθώς και την ρίζα του δένδρου root και μας επιστρέφει τον πλησιέστερο κόμβο nn και την απόσταση που αυτός απέχει από το test_pt. Η συνάρτηση αυτή λειτουργεί ως εξής:

Αρχικά θέτουμε ως καλύτερη απόσταση μια αρκετά μεγάλη τιμή και στη συνέχεια υπολογίζουμε την Ευκλείδεια απόσταση του κόμβου αναφοράς test_pt από την ρίζα του δένδρου. Αν η απόσταση αυτή είναι μικρότερη από την καλύτερη απόσταση τότε η καλύτερη απόσταση παίρνει την τιμή της Ευκλείδειας αυτής απόστασης. Στη συνέχεια, ελέγχουμε αν υπάρχει δεξί παιδί. Σε περίπτωση που υπάρχει δεξί παιδί καλούμε αναδρομικά την συνάρτηση NearestNeighboorKDtree δίνοντας ως ρίζα το δεξί παιδί. Έπειτα ελέγχουμε αν υπάρχει αριστερό παιδί. Σε περίπτωση που υπάρχει αριστερό παιδί καλούμε αναδρομικά την συνάρτηση NearestNeighboorKDtree δίνοντας ως ρίζα το αριστερό παιδί.

Στην συνάρτηση πλησιέστερου γείτονα με KD-tree το δένδρο διατρέχεται μια φορά από πάνω προς τα κάτω ούτως ώστε να βρεθεί ο πλησιέστερος κόμβος που βρίσκεται στο ίδιο κελί με τον κόμβο αναφοράς tset_pt .Στη συνέχεια το δένδρο διατρέχεται ακόμα μια φορά από κάτω προς τα πάνω ώστε να βρεθεί , σε περίπτωση που υπάρχει , κόμβος που βρίσκεται σε κάποιο γειτονικό κελί από το κελί του κόμβου αναφοράς και είναι πλησιέστερος στον κόμβο αναφοράς σε σχέση με τον κόμβο που βρέθηκε στην προσπέλαση από πάνω προς τα κάτω .

▫ Στη συνέχεια , παρατίθενται μία σειρά εικόνων για διαφορετικές τιμές της μεταβλητής N5 στις οποίες εμπεριέχονται:

-Η απεικόνιση του νέφους σημείων C_{N5} γίνεται με μαύρο χρώμα.

- Η απεικόνιση του νέφους σημείων C_{N5-1} γίνεται με μωβ χρώμα.

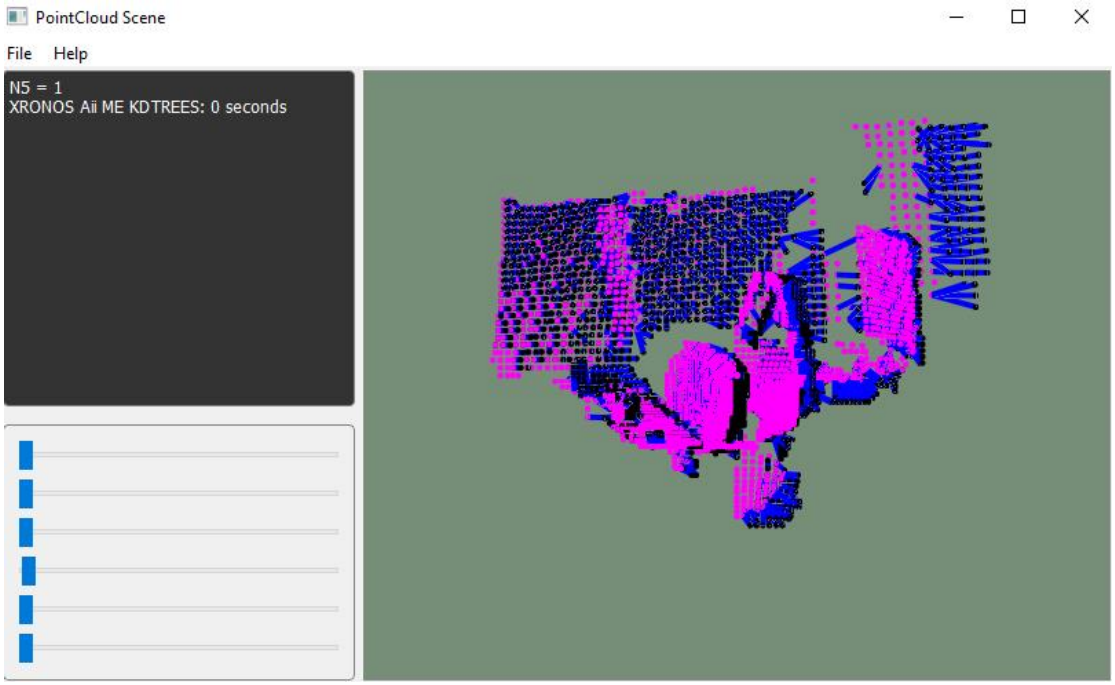
-Η απεικόνιση της αντιστοίχισης των σημείων p_i του νέφους σημείων C_{N5} με τα αντίστοιχα (πλησιέστερα) σημεία q του νέφους σημείων C_{N5-1} γίνεται με μπλε χρώμα.

-Η εκτύπωση του χρόνου υπολογισμού της αντιστοίχισης σημείων.

Ως βήμα δειγματοληψίας έχει θεωρηθεί D=10.

◦ N5=1

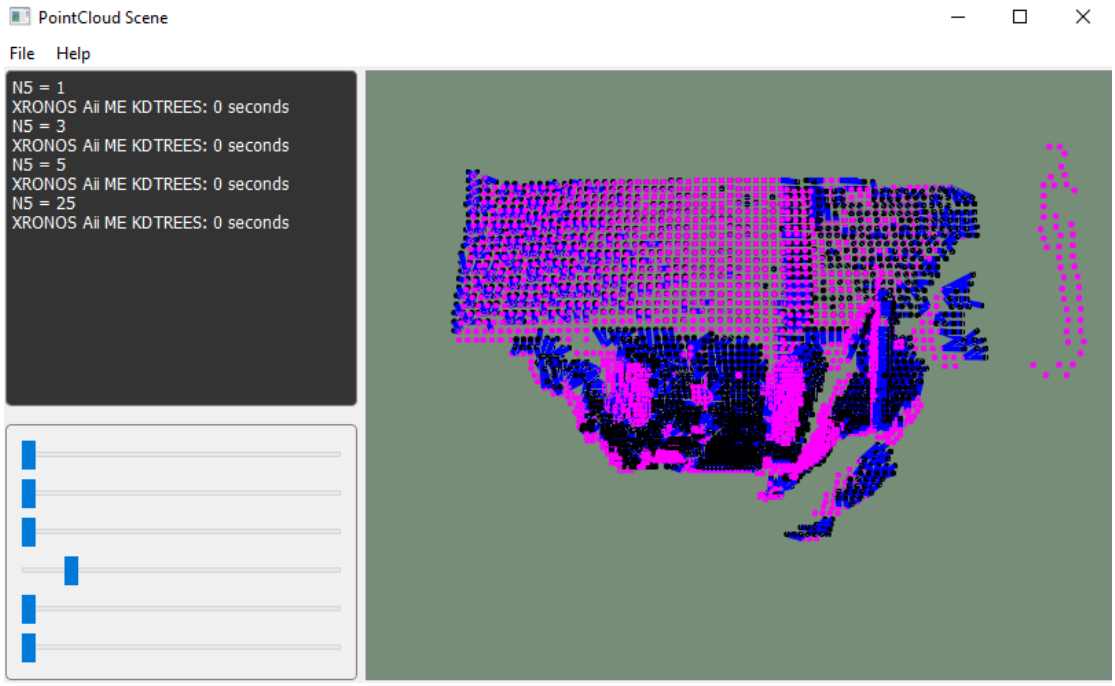
Χρόνος υπολογισμού αντιστοίχισης σημείων: 0seconds



Εικόνα No17

◦ N5=25

Χρόνος υπολογισμού αντιστοίχισης σημείων: 0seconds



Εικόνα No18

Σχόλιο:

Όπως γίνεται σαφές οι χρόνοι εκτέλεσης του ερωτήματος 2 με KD-trees είναι κατά πολύ μικρότεροι από τους αντίστοιχους του ερωτήματος 2 χωρίς δένδρο αναζήτησης.

Π.χ. για N5=1 : KD-tree 0seconds

για N=1 : Χωρίς KD-tree 33seconds

❖ Ερώτημα Β 4 με KD-trees

Το νέφος C_i επιλέγεται από τον χρήστη με την χρήση της δεύτερης μπάρας και ο αριθμός του νέφους σημείων C_i δείχνεται από την μεταβλητή Ni , ενώ το νέφος C_j επιλέγεται από τον χρήστη με την χρήση της τρίτης μπάρας και ο αριθμός του νέφους σημείων C_j δείχνεται από την μεταβλητή Nj .

Στο ερώτημα αυτό ακολουθούμε ακριβώς την ίδια διαδικασία με το ερώτημα 4 μόνο που χρησιμοποιούνται KD-trees για την εύρεση του πλησιέστερου γείτονα. Πιο συγκεκριμένα, για την εύρεση των πλησιέστερων σημείων του νέφους pointcloud C_i ως προς το poincloud C_{i+1} , δηλαδή τα σημεία του νέφους σημείων C_i που βρίσκονται πιο κοντά στα σημεία του νέφους σημείων C_{i+1} χρησιμοποιούμε την συνάρτηση:

```
PlisiesterosKDTREE(std::vector<vec> vertices1, std::vector<vec> vertices2,
std::vector<vec> &P)
```

Η συνάρτηση αυτή δέχεται ως ορίσματα τα vertices του pointcloud i+1 και του pointcloud i και μας επιστρέφει το διάνυσμα P , το οποίο περιέχει τα σημεία του pointcloud i που είναι πλησιέστερα στα σημεία του pointcloud i+1. Για την εύρεση των πλησιέστερων σημείων με KD-trees χρησιμοποιείται η συνάρτηση NearestNeighboorKDtree που εξηγήθηκε παραπάνω.

Επίσης, δένδρο αναζήτησης χρησιμοποιείται και για την εισαγωγή των ευθυγραμμισμένων σημείων στο διάνυσμα vtr , που περιέχει όλα τα σημεία του ολικού νέφους M. Η διαδικασία αυτή πραγματοποιείται με την βοήθεια της εξής συνάρτησης:

```
KDTREEProsthesimidiplwn(std::vector<vec> vertices1, std::vector<vec> &vertices2,
float e)
```

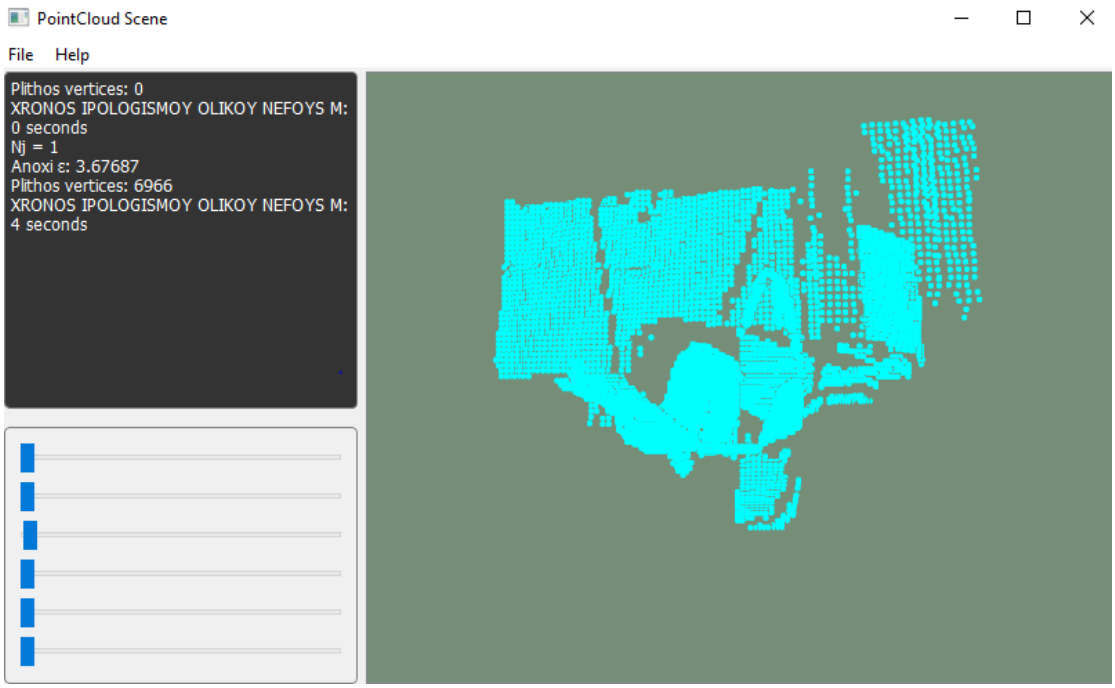
,η οποία δέχεται σαν ορίσματα τα ευθυγραμμισμένα vertices1 και όλα τα σημεία του μέχρι εκείνη την στιγμή ολικού νέφους M vertices2 , καθώς και την ανοχή e. Η συνάρτηση αυτή κατασκευάζει ένα KD-tree με τα σημεία του vertices2 (τα οποία είναι περισσότερα ή ίσα με τα vertices1) ώστε να επιταχύνεται η

διαδικασία της αναζήτησης του πλησιέστερου γείτονα και ψάχνει για κάθε ένα από τα ευθυγραμμισμένα vertices1τον πλησιέστερο γείτονα του μέχρι τότε ολικού νέφους που ανήκει στο vertices2. Έπειτα, υπολογίζει την Ευκλείδεια απόσταση του κάθε σημείου του vertices1 με το πλησιέστερο σημείο που βρέθηκε και αν αυτή η απόσταση είναι μεγαλύτερη από την ανοχή e , δηλαδή $d > e$, τότε προσθέτει το ευθυγραμμισμένο σημείο του vertices1 στο ολικό νέφος σημείων.

▫ Ακολούθως παρουσιάζονται τα διαφορετικά ολικά νέφη M που προέκυψαν για διαφορετικά νέφη σημείων C_i και C_j. Ταυτόχρονα παρατίθενται ο συνολικός χρόνος υπολογισμού του ολικού νέφους M για διαφορετικές εισόδους i και j , το πλήθος των σημείων του ολικού νέφους, καθώς και η τιμή της ανοχής ε. Το βήμα δειγματοληψίας που χρησιμοποιήθηκε είναι D=7.

◦ N_i=0

N_j=1



Εικόνα No19

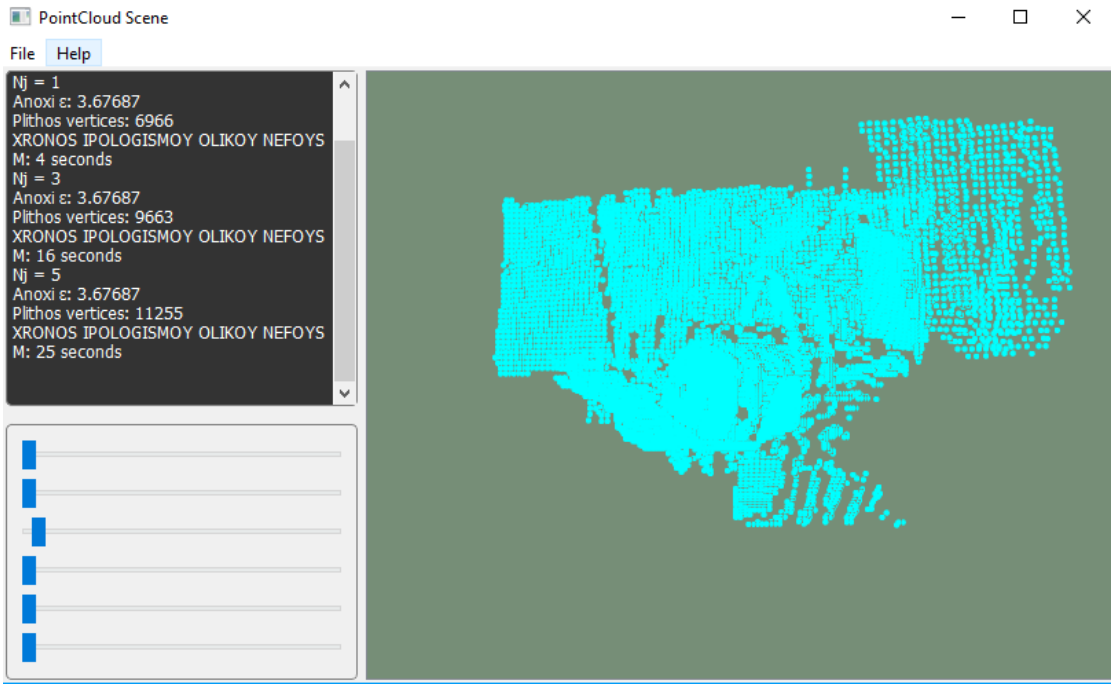
Χρόνος υπολογισμού ολικού νέφους M: 4seconds
Πλήθος σημείων: 6966
Ανοχή ε: 3.67687

N_j=3

Εικόνα No20

Χρόνος υπολογισμού ολικού νέφους M: 16seconds
Πλήθος σημείων: 9663
Ανοχή ε: 3.67687

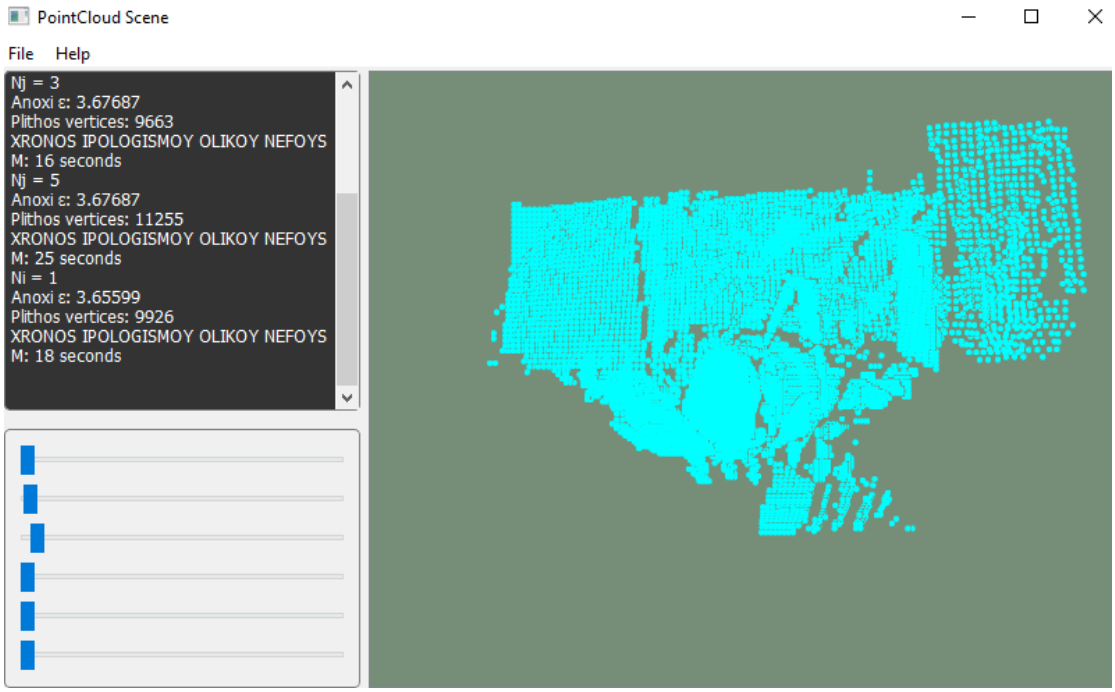
N_j=5



Εικόνα No21

Χρόνος υπολογισμού ολικού νέφους M: 25seconds
Πλήθος σημείων: 11255
Ανοχή ε: 3.67687

◦ $N_i=1$
$N_j=5$



Εικόνα No22

Χρόνος υπολογισμού ολικού νέφους M: 18seconds
Πλήθος σημείων: 9926
Ανοχή ε: 3.6599

<u>Σχόλιο1:</u>

Οι περιορισμοί που μας δημιουργεί μια δομή δεδομένων διάτμησης χώρου αφορά κυρίως τον χώρο της μνήμης που καταλαμβάνεται, καθώς θα είναι μεγαλύτερος εφόσον δημιουργούνται δένδρα.

<u>Σχόλιο2:</u>

Όπως γίνεται σαφές οι χρόνοι εκτέλεσης του ερωτήματος 4 με KD-trees είναι κατά πολύ μικρότεροι από τους αντίστοιχους του ερωτήματος 4 χωρίς δένδρο αναζήτησης.

Π.χ. για $N_i=0$, $N_j=1$: KD-tree 4seconds
Χωρίς KD-tree 49seconds

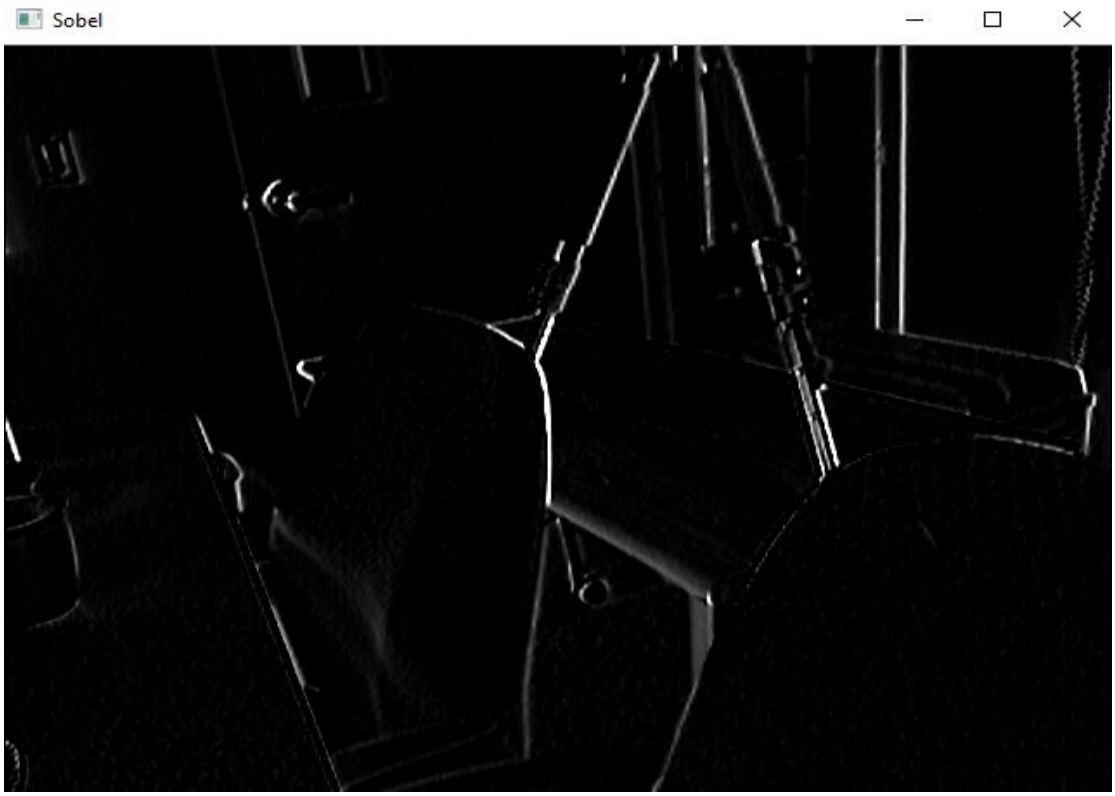
Σχόλιο3:

Η ανακατασκευή της επιφάνειας για μεγάλες τιμές της μεταβλητής Nj δεν φαίνεται καλά , καθώς δεν είναι σωστή η ευθυγράμμιση αφού κατά την εύρεση των πλησιέστερων γειτόνων η αντιστοιχία που υπάρχει δεν είναι απαραίτητα ορθή.

●Ερώτημα 6ο

Εφαρμόστε φίλτρο Sobel στις εικόνες χρώματος, και αφού απορρίψετε χαμηλές τιμές με χρήση κατωφλιού (thresholding), χρησιμοποιήστε μόνο τα αντίστοιχα σημεία για την βελτιστοποίηση της συνάρτησης του ερωτήματος (iii). Δείξτε το αποτέλεσμα και τον βαθμό επιτάχυνσης του αλγορίθμου

Για να μεταβούμε από το πέμπτο ερώτημα στο έκτο αρκεί να πατήσουμε το πλήκτρο ‘ e ’ από το πληκτρολόγιο. Το νέφος C_i επιλέγεται από τον χρήστη με την χρήση της δεύτερης μπάρας και ο αριθμός του νέφους σημείων C_i δείχνεται από την μεταβλητή Ni , ενώ το νέφος C_j επιλέγεται από τον χρήστη με την χρήση της τρίτης μπάρας και ο αριθμός του νέφους σημείων C_j δείχνεται από την μεταβλητή Nj .
Με την εκτέλεση του εν λόγω ερωτήματος μετατρέπονται αρχικά οι εικόνες χρώματος σε εικόνες grey scale , ούτως ώστε να εφαρμοστεί σε αυτές στην συνέχεια φίλτρο Sobel. Για παράδειγμα η Sobel εικόνα για $N_i = 0$ και $N_j = 1$ είναι η παρακάτω:



Εικόνα No23

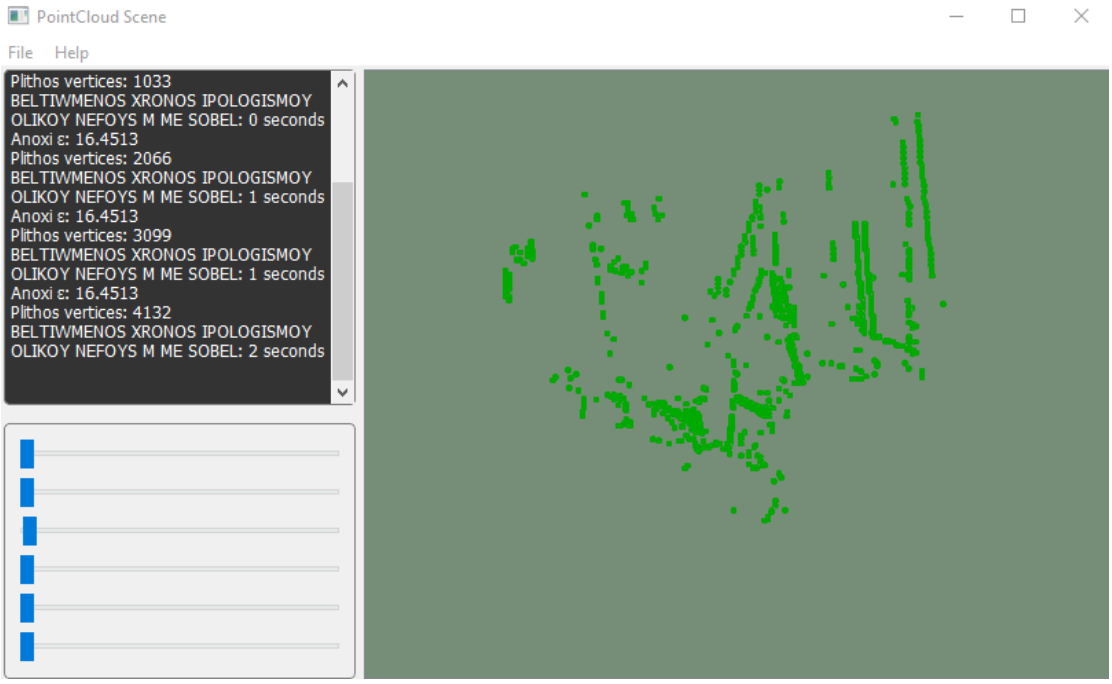
Ακολούθως με χρήση κατωφλιού (thresholding) απορρίπτουμε τις χαμηλές τιμές , κρατώντας από την εικόνα που προέκυψε από το Sobel φιλτράρισμα , τιμές μεγαλύτερες του 10. Η εικόνα που προέκυψε από το threshold για $N_i = 0$ και $N_j = 1$ είναι η παρακάτω:

Εικόνα No23

Στη συνέχεια, δημιουργούμε το 3Δ νέφος σημείων της εκάστοτε εικόνας βάθους, κρατώντας από αυτήν τα σημεία τα οποία στην Threshold εικόνα είναι άσπρα , δηλαδή το κρατώντας το επίπεδο 255. Η διαδικασία αυτή πραγματοποιείται χρησιμοποιώντας την συνάρτηση:

```
PCSobel(const cv::Mat& depth, std::vector<vec> &vertices, const cv::Mat& thresh)
```

,η οποία δέχεται σαν ορίσματα την εικόνα βάθους depth , την εικόνα που προέκυψε από το thresholding thresh και μας επιστρέφει τα vertices που στην εικόνα threshold είναι άσπρα. Έπειτα, ακολουθούμε ακριβώς την ίδια διαδικασία με το ερώτημα 5-Β , δηλαδή υλοποιούμε το ερώτημα 4 με KD-trees για τα σημεία που προέκυψαν από την συνάρτηση PCSobel για την εικόνα βάθους i , καθώς και την εικόνα βάθους i+1. Το αποτέλεσμα που προέκυψε για $N_i = 0$, $N_j = 1$ και βήμα δειγματοληψίας D=5 είναι το ακόλουθο:

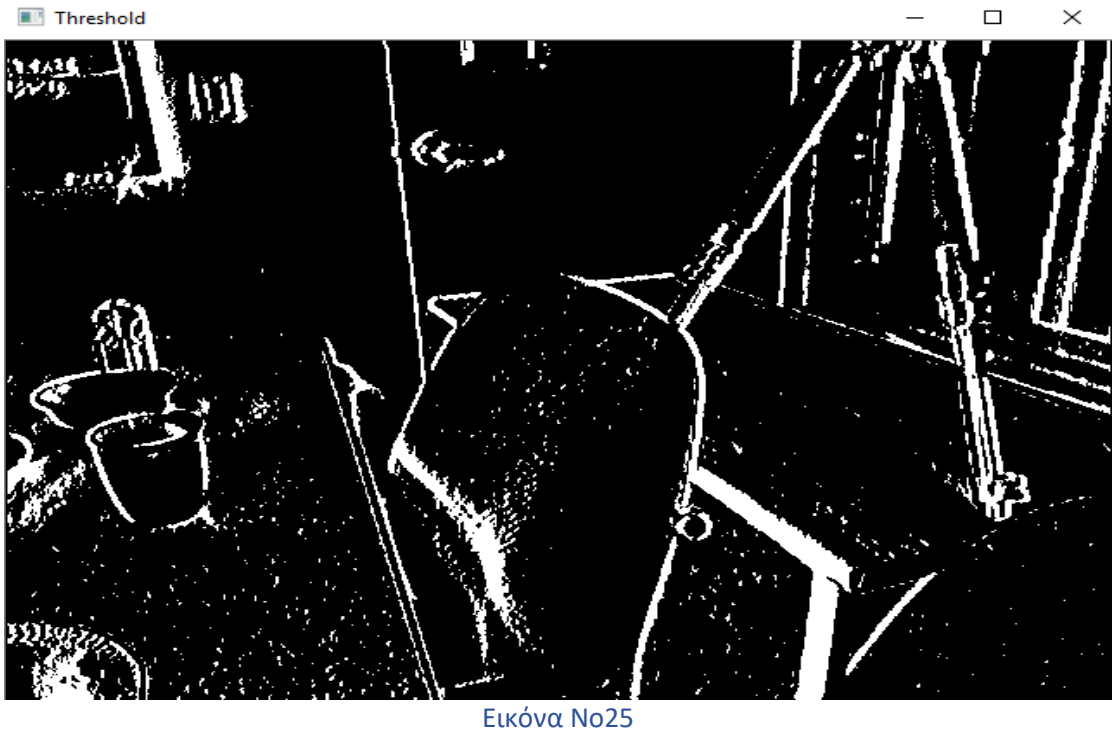


Εικόνα No24

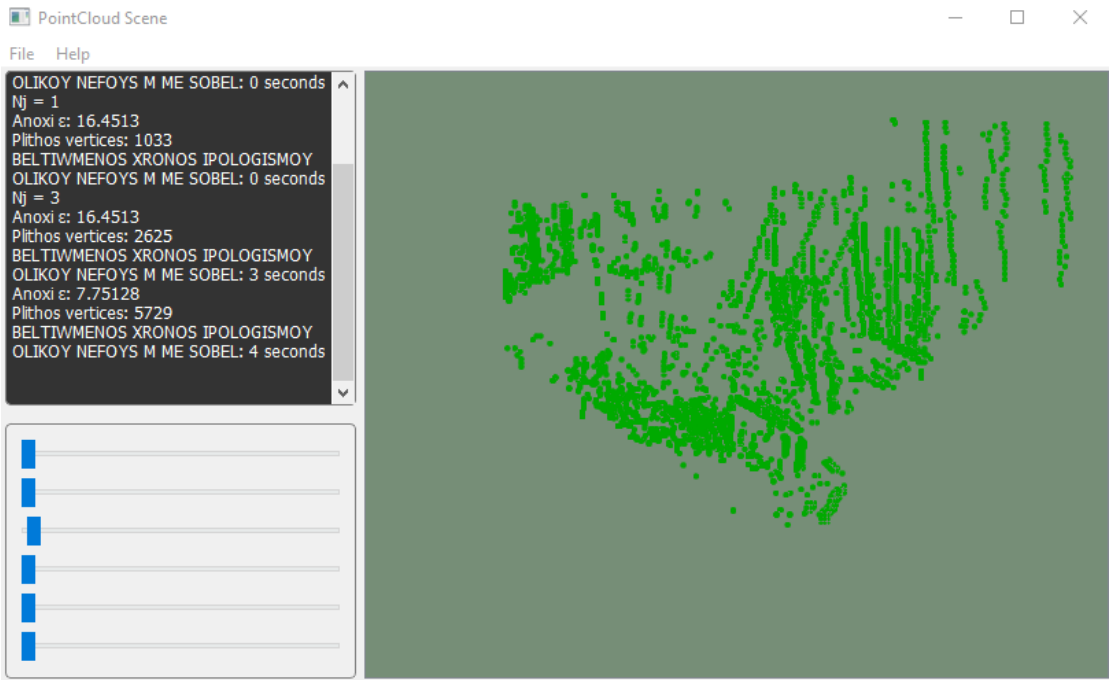
- Ακολούθως παρουσιάζονται τα διαφορετικά ολικά νέφη M που προέκυψαν για διαφορετικά νέφη σημείων C_i και C_j , δηλαδή για διαφορετικές εισόδους i και j . Το βήμα δειγματοληψίας που χρησιμοποιήθηκε είναι D=7.

- N_i=0

N_j=3



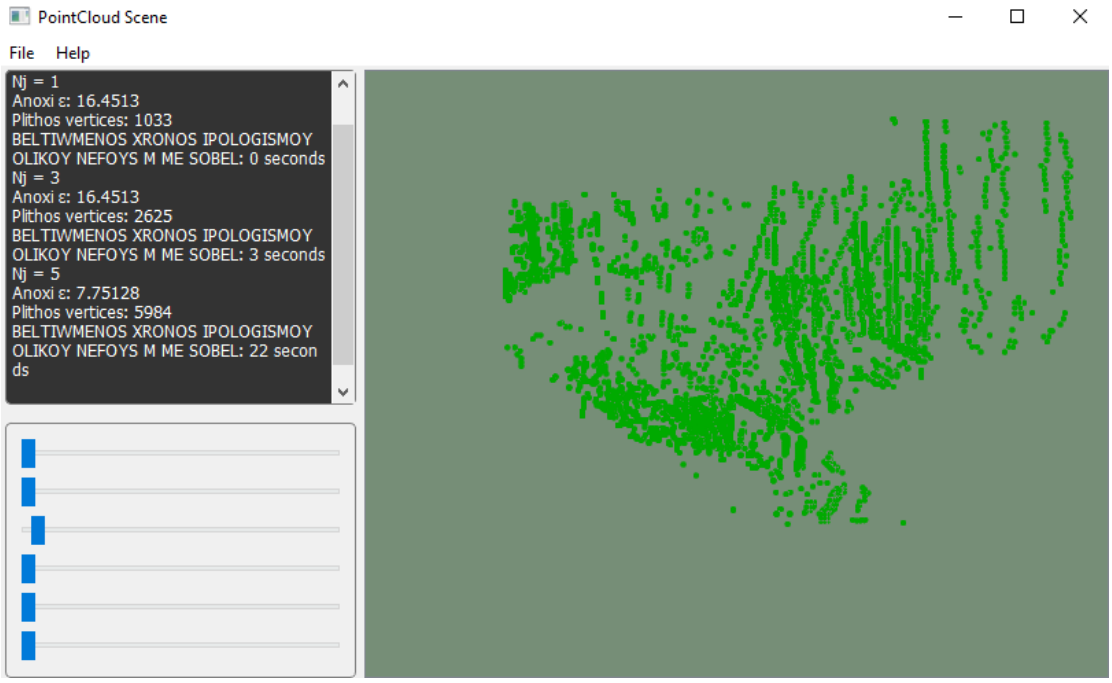
Εικόνα No25



Εικόνα No26

◦ $N_i=1$
$N_j=5$

Εικόνα No27



Εικόνα No28

Σχόλιο:

Οι χρόνοι εκτέλεσης του ερωτήματος 6 είναι ακόμα μικρότεροι από τους αντίστοιχους του ερωτήματος 5. Αυτό οφείλεται στο γεγονός ότι τα δύο νέφη σημείων που προέκυψαν από το threshold έχουν ακόμα λιγότερα σημεία από τα αντίστοιχα του ερωτήματος 5.

Π.χ. για N_i=1 , N_j=5: Sobel: 10seconds
Ερώτημα 5: 18seconds

◦ N_i=1 & N_j=5 & D=7