Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Ηλεκτρονικής και Υπολογιστών

Υπολογισμός της Ευκλείδειας Απόστασης δύο Τριγωνικών Πλεγμάτων

Διπλωματική Εργασία
Καρελής Παναγιώτης

Επιβλέπων: Πιτσιάνης Νικόλαος
Αναπληρωτής Καθηγητής Α.Π.Θ.

1 Νοεμβρίου 2022

Περίληψη

Σε μια πληθώρα εφαρμογών της Υπολογιστικής Γεωμετρίας (Μηχανική με τη Βοήθεια Υπολογιστών (CAE), Προσομοιώσεις με Υπολογιστές, Ρομποτική, Γραφική με Υπολογιστές κ.α.) τα αντικείμενα του χώρου αναπαρίστανται συνήθως από πολυγωνικά πλέγματα. Κοινό πρόβλημα για όλους τους παραπάνω τομείς αποτελεί η εύρεση της απόστασης που διαχωρίζει δύο αντικείμενα και η ανίχνευση σύγκρουσης μεταξύ τους. Στην παρούσα εργασία, προτείνουμε αποδοτικούς αλγορίθμους που υπολογίζουν την Ευκλείδεια απόσταση δύο αντικειμένων του τρισδιάστατου χώρου και τους υλοποιούμε για την περίπτωση των τριγωνικών πλεγμάτων. Για τους αλγορίθμους αυτούς σχεδιάζουμε μια δενδρική δομή δεδομένων που ανήκει στην κατηγορία των Ιεραρχιών Οριοθετικών Όγκων (BVH). Η διαδικασία κατασκευής της παραπάνω δομής είναι παρόμοια με αυτή του KD-Tree, με τη διαφορά ότι η δομή μας διαχειρίζεται χωρικά δεδομένα και κάνει χρήση Οριοθετικών Πλαισίων Ευθυγραμμισμένων με τους Άξονες (AABB). Επιπλέον, περιγράφουμε έναν τρόπο διάσχισης της δομής ώστε να υποστηρίζει ερωτήματα κοντινότερου γείτονα για χωρικά δεδομένα. Η διάσχιση της δενδρικής δομής σχεδιάζεται ως μια κατευθυνόμενη αναζήτηση κατά βάθος (DFS), που στοχεύει στη σμίκρυνση του χώρου αναζήτησης μέσω κλαδέματος του δένδρου κατά την οπισθοδρόμηση. Ακόμη, στην υλοποίηση μας παραλληλοποιούμε τη διαδικασία κατασκευής του δένδρου όπως και τη διαδικασία αναζήτησης της ελάχιστης απόστασης κάνοντας χρήση πολλαπλών νημάτων επεξεργασίας. Τέλος, μετράμε και αναλύουμε την επίδοση των αλγορίθμων μας σε μια σειρά από περιπτώσεις ελέγχου που κατασκευάσαμε.

Λέξεις-κλειδιά:
Υπολογιστική Γεωμετρία, Πολυγωνικά Πλέγματα, Ευκλείδεια Απόσταση, Ιεραρχίες Οριοθετικών Όγκων

Abstract

In a plethora of fields in Computational Geometry (Computer Aided Engineering, Computer Simulations, Robotics, Computer Graphics etc.) objects in space are represented as polygonal meshes. A common problem, for all the above, is the computation of separation distance and collision detection of two objects. In this thesis, we propose efficient algorithms that compute the Euclidean distance of two objects in 3D space and we implement them for the case of triangle meshes. For these algorithms we design a tree data structure that belongs to the family of Bounding Volume Hierarchies (BVH). The construction procedure of this data structure is similar to the one used by the KD-Tree, but it differs, as our data structure manages spatial data and also uses Axis-Aligned Bounding Boxes (AABB). In addition, we describe a traversal scheme of the data structure in order to answer nearest neighbor queries for spatial data. The traversal of the tree structure is implemented as a directed depth first search (DFS), aiming to reduce the searching space by pruning the tree during backtracking. Furthermore, we parallelize the construction of the data structure as well as the procedure of finding the Euclidean distance, using multithreading. Finally, we measure and analyze the efficiency of our algorithms on a series of test cases we created.

Keywords:
Computational Geometry, Polygonal Meshes, Euclidean Distance, Bounding Volumes Hierarchies

Ευχαριστίες

Άδειο

Υπολογισμός της Ευκλείδειας Απόστασης δύο Τριγωνικών Πλεγμάτων

Παναγιώτης Καρελής
karelisp@ece.auth.gr

1 Νοεμβρίου 2022

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο

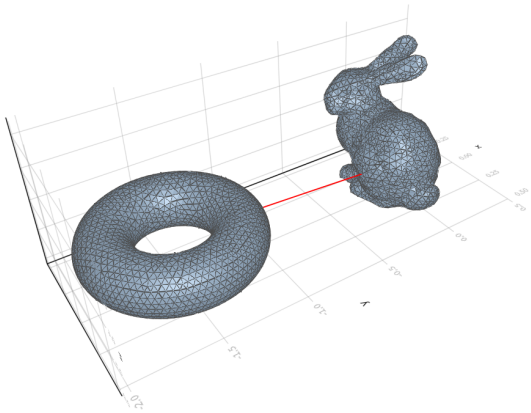
Ο υπολογισμός της απόστασης που διαχωρίζει δύο αντικείμενα στο χώρο αποτελεί θεμελιώδες πρόβλημα στον τομέα της Υπολογιστικής Γεωμετρίας. Στη ρομποτική, στη σχεδίαση και μηχανική με τη βοήθεια υπολογιστών (CAD και CAE), στις προσομοιώσεις με υπολογιστές και στη γραφική υπολογιστών είναι σημαντικό να γνωρίζουμε εάν δύο αντικείμενα, που περιγράφονται από μαθηματικά μοντέλα στον τρισδιάστατο χώρο, τέμνονται/συγκρούονται ή βρίσκονται σε κοντινή απόσταση. Για την παρούσα διπλωματική εργασία, εξετάζουμε την περίπτωση όπου τα αντικείμενα του χώρου περιγράφονται από τριγωνικά πλέγματα (βλ. 2.1).

Στη ρομποτική, για παράδειγμα, η επίλυση του παραπάνω προβλήματος είναι απαιτητή για τον σχεδιασμό διαδρομής με την παρουσία εμποδίων (path-planning problem) [9], [10], [11], [13]. Επιπλέον, σε εφαρμογές CAD-CAE στις οποίες σχεδιάζονται περίπλοκες κατασκευές που αποτελούνται από μεγάλο αριθμό εξαρτημάτων, ο εντοπισμός σύγκρουσης μεταξύ των εξαρτημάτων είναι απαραίτητος τόσο για τις αναλύσεις και δοκιμές των προϊόντων όσο και για την παραγωγή τους [8]. Στη γραφική με υπολογιστές και συγκεκριμένα κατά την κίνηση των αντικειμένων σε μια εικονική σκηνή, όπως στα βιντεοπαιχνίδια και στα κινούμενα σχέδια, είναι πιθανό τα αντικείμενα να διεισδύσουν το ένα στο άλλο. Αυτή η κατάσταση δεν είναι επιθυμητή όταν με τα γραφικά επιδιώκεται η αναπαράσταση ενός ρεαλιστικού κόσμου [35]. Τέλος, το πρόβλημα υπολογισμού της απόστασης που διαχωρίζει δύο αντικείμενα στο χώρο συναντάται και στον τομέα της υπολογιστικής φυσικής για εφαρμογές ανάλυσης πεπερασμένων στοιχείων (FEA) και προσομοιώσεων [28].

Η ραγδαία ανάπτυξη όλων των παραπάνω τομέων τα τελευταία χρόνια και η απαίτηση λεπτομερέστερης περιγραφής των αντικειμένων του τρισδιάστατου χώρου, κάνουν επιτακτική την ανάγκη για μελέτη και σχεδιασμό αλγορίθμων ικανών να διαχειριστούν μεγάλους όγκους δεδομένων εισόδου. Σε αυτή τη διπλωματική εργασία προτείνουμε αποδοτικούς αλγορίθμους για τον υπολογισμό της απόστασης δύο πλεγμάτων στον τρισδιάστατο χώρο.

1.2 Περιγραφή του Προβλήματος

Δοθέντων δύο αντικειμένων στον χώρο, τα οποία περιγράφονται από τριγωνικά πλέγματα, το πιο φυσικό μέγεθος για την περιγραφή της εγγύτητας μεταξύ τους είναι η Ευκλείδεια απόσταση. Δηλαδή, το μήκος του μικρότερου ευθυγράμμου τμήματος που ενώνει τα δύο αντικείμενα. Σε περίπτωση που τα αντικείμενα συγκρούονται η απόσταση τους είναι μηδέν.



Σχήμα 1.1: Ευκλείδεια Απόσταση δύο Τριγωνικών Πλεγμάτων - Το κόκκινο ευθύγραμμο τμήμα αναπαριστά την Ευκλείδεια απόσταση ενός Τόρου και του Stanford Bunny.

Με τον ίδιο ακριβώς τρόπο ορίζεται και η Ευκλείδεια απόσταση μεταξύ δύο τριγώνων στον τρισδιάστατο χώρο. Σχεδιάζοντας μια τέτοια ρουτίνα (βλ. 4.1.1) τότε μπορούμε να δώσουμε έναν εναλλακτικό ορισμό του προβλήματος και ταυτόχρονα έναν αλγόριθμο που το επιλύει:

Ορισμός 1. *Έστω δύο αντικείμενα του τρισδιάστατου χώρου που περιγράφονται από τριγωνικά πλέγματα. Επιπλέον, έστω τα σύνολα X, Y που αποτελούνται από τα τρίγωνα των δύο πλεγμάτων, αντίστοιχα, και $tria\_dist(x, y)$ η ρουτίνα που υπολογίζει την Ευκλείδεια απόσταση δύο τριγώνων x, y. Η Ευκλείδεια απόσταση d των δύο τριγωνικών πλεγμάτων είναι*

$$d = \min_{\forall x \in X, \forall y \in Y} tria\_dist(x, y)$$

Ο τετριμμένος αλγόριθμος, που βασίζεται στον ορισμό, υπολογίζει την απόσταση κάθε πιθανού ζεύγους τριγώνων και επιλέγει την ελάχιστη. Τέτοιοι αλγόριθμοι εξαντλητικής αναζήτησης περιγράφονται στο 4.2, όμως είναι αδύνατο να χρησιμοποιηθούν σε πραγματικές εφαρμογές. Ο λόγος είναι η μεγάλη υπολογιστική τους πολυπλοκότητα της τάξης $O(N \cdot M)$ (όπου N, M το πλήθος των τριγώνων των συνόλων X, Y αντίστοιχα).

1.3 Στόχοι της Διπλωματικής Εργασίας

Η συνεισφορά της διπλωματικής εργασίας είναι η ακόλουθη:

- Σχεδιάζουμε μια δενδρική δομή δεδομένων που ανήκει στην οικογένεια των Ιεραρχιών Οριοθετικών Όγκων (BVH). Η δομή αυτή χρησιμοποιείται ως χωρικό ευρετήριο (spatial indexing).
- Προτείνουμε έναν τρόπο διάσχισης της παραπάνω δομής ώστε να υποστηρίζονται ερωτήματα κοντινότερου γείτονα για χωρικά δεδομένα. Στόχος είναι να ελεγχθεί μόνο ένα μικρό υποσύνολο του χώρου αναζήτησης. Αυτό επιτυγχάνεται με το κλάδεμα (pruning) του δένδρου κατά την αναζήτηση.
- Προτείνουμε δύο αλγορίθμους που επιλύουν το πρόβλημα υπολογισμού της Ευκλείδειας απόστασης δύο πλεγμάτων. Οι αλγόριθμοι είναι γενικοί και μπορούν να εφαρμοστούν σε πλέγματα που αποτελούνται από διαφόρων ειδών πολύτοπα (πολύγωνα και πολύεδρα) υπό προϋποθέσεις. Στην εργασία αυτή μελετώνται τα τριγωνικά πλέγματα.
- Υλοποιούμε τους αλγορίθμους για συστήματα υψηλής απόδοσης που υποστηρίζουν πολλαπλά νήματα (multithreading). Παραλληλοποιούμε τις διαδικασίες κατασκευής του δένδρου και αναζήτησης της ελάχιστης απόστασης.
- Μετράμε και αναλύουμε την επίδοση των αλγορίθμων μας σε μια σειρά από περιπτώσεις ελέγχου που κατασκευάσαμε.

1.4 Διάρθρωση της Διπλωματικής Εργασίας

Στο **Κεφάλαιο 1** έγινε μια εισαγωγή στο πρόβλημα υπολογισμού της απόστασης δύο πλεγμάτων και παρουσιάστηκαν τα κίνητρα που οδήγησαν στην υλοποίηση των αλγορίθμων που θα παρουσιαστούν.

Στο **Κεφάλαιο 2** παρουσιάζεται το θεωρητικό υπόβαθρο που απαιτείται από τον αναγνώστη ώστε να κατανοήσει πλήρως το πρόβλημα και την προτεινόμενη λύση.

Στο **Κεφάλαιο 3** αναφέρεται η σχετική βιβλιογραφία, δηλαδή πώς αντιμετώπισαν άλλοι ερευνητές το ίδιο ή παρόμοια προβλήματα. Παρουσιάζονται επίσης ομοιότητες και διαφορές των υπολοίπων προσεγγίσεων σε σχέση με τη δική μας.

Στο **Κεφάλαιο 4** αναλύεται η μεθοδολογία που προτείνουμε και παρουσιάζεται η υλοποίηση των αλγορίθμων μας.

Στο **Κεφάλαιο 5** παρατίθενται τα αποτελέσματα από τα πειράματα που εκτελέσαμε. Οι μετρήσεις των πειραμάτων περιλαμβάνουν την εκτίμηση της μετρικής κόστους που ορίζεται στην ενότητα 4.3 καθώς και τους χρόνους εκτέλεσης των αλγορίθμων.

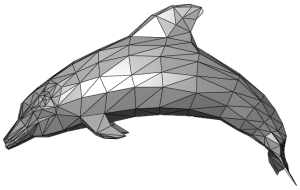
Στο **Κεφάλαιο 6** σχολιάζονται τα αποτελέσματα και παρουσιάζονται σκέψεις για μελλοντική επέκταση και βελτίωση των ιδεών της παρούσας διπλωματικής εργασίας.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 Πλέγματα

Στην Υπολογιστική Γεωμετρία τα *πλέγματα* αποτελούν την αναπαράσταση μιας μεγαλύτερης γεωμετρικής περιοχής από μικρότερα διακριτά στοιχεία. Τα πλέγματα χρησιμοποιούνται συνήθως για τον υπολογισμό λύσεων μερικών διαφορικών εξισώσεων, για την απόδοση γραφικών υπολογιστών, και για ανάλυση γεωγραφικών και χαρτογραφικών δεδομένων. Ένα πλέγμα χωρίζει τον χώρο σε μικρότερα στοιχεία (πολύγωνα ή πολύεδρα) όπου μπορούν να λυθούν οι εξισώσεις, το οποίο στη συνέχεια προσεγγίζει τη λύση στο ευρύτερο πεδίο. Τα πλέγματα που αποτελούνται από πολύεδρα αντιπροσωπεύουν ρητά τόσο την επιφάνεια όσο και τον όγκο ενός αντικειμένου, ενώ τα πολυγωνικά πλέγματα αντιπροσωπεύουν μόνο την επιφάνεια (ο όγκος υπονοείται). Για το πρόβλημα υπολογισμού της Ευκλείδειας απόστασης, ενδιαφερόμαστε μόνο για την εξωτερική επιφάνεια των αντικειμένων του τρισδιάστατου χώρου.



Σχήμα 2.1: Παράδειγμα τριγωνικού πλέγματος που αναπαριστά ένα δελφίνι.

Ένας τύπος πολυγωνικών πλεγμάτων είναι τα τριγωνικά πλέγματα (σχήμα 2.1). Αποτελούνται από ένα σύνολο τριγώνων στις τρεις διαστάσεις, τα οποία συνδέονται με τις κοινές τους ακμές ή κορυφές. Γεωμετρικά, ένα πλέγμα είναι μια τμηματικά επίπεδη επιφάνεια. Η τελευταία ιδιότητα ισχύει πάντοτε για τα τριγωνικά πλέγματα.

Υπάρχουν διάφοροι τρόποι για την αποθήκευση ενός τριγωνικού πλέγματος στη μνήμη του υπολογιστή. Υπάρχουν επίσης μέθοδοι μετατροπής του ενός τρόπου αποθήκευσης σε έναν άλλο. Ενδεικτικά αναφέρουμε την αποθήκευση με:

- **Σετ τριγώνων**: Το πλέγμα αναπαρίσταται απλά από τα τρίγωνα του. Δηλαδή αποθηκεύονται οι συντεταγμένες των κορυφών κάθε τριγώνου.
- **Σετ Τριγώνων με δείκτες**: Το πλέγμα αναπαρίσταται από ένα σετ κόμβων και ένα σετ από τριπλέτες με δείκτες στους κόμβους. Η κάθε τριπλέτα αναπαριστά ένα τρίγωνο.
- **Λωρίδες Τριγώνων**: Η αποθήκευση αυτή βασίζεται στο γεγονός ότι δύο γειτονικά τρίγωνα μοιράζονται τη μία τους πλευρά. Αυτός ο τρόπος χρησιμοποιείται για συμπίεση των πλεγμάτων.

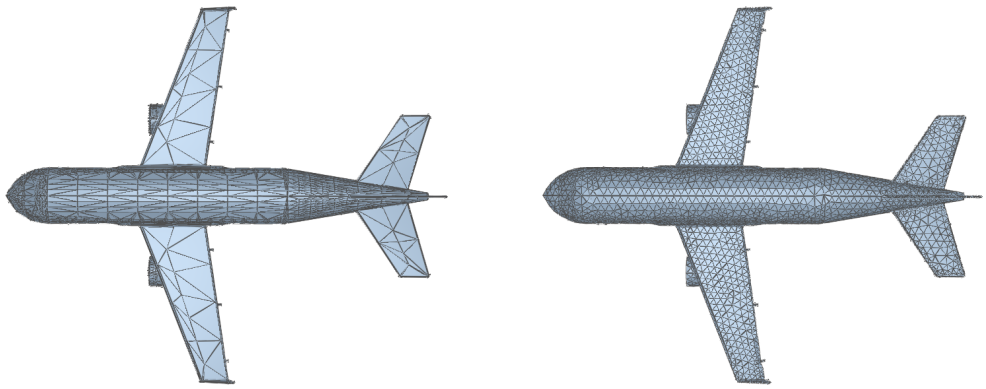
- **Δομή Τρίγωνου-Γείτονα**: Υποστηρίζει ερωτήματα γειτνίασης τριγώνων.
- **Δομή Winged-Edge**: Αποθηκεύει δεδομένα κόμβων, ακμών και όψεων. Επιτρέπει εύκολη διάβαση στο πλέγμα μεταξύ όψεων, ακμών και κορυφών.

Για τους αλγορίθμους που περιγράφουμε παρακάτω αρκεί ο πρώτος τρόπος αναπαράστασης.

Τέλος, για τις διάφορες εφαρμογές που χρησιμοποιούνται, τα πλέγματα χαρακτηρίζονται και από την ποιότητα τους. Οι πιο συνηθισμένες μετρικές ποιότητας είναι:

- **Λοξότητα (Skewness)**:
  Η λοξότητα είναι ο λόγος της απόκλισης μεταξύ του βέλτιστου μεγέθους στοιχείου προς στο υπάρχον μέγεθος στοιχείου. Το εύρος της λοξότητας είναι μεταξύ 0 (ιδανικό) έως 1 (χειρότερο). Τα πολύ λοξά στοιχεία δεν προτιμώνται λόγω της κακής ακρίβειας που προκαλούν στις παρεμβαλλόμενες περιοχές. Ανάλογα με το στοιχείο (τρίγωνο, τετράπλευρο, τετράεδρο, εξάεδρο κλπ) διαφοροποιείται μαθηματικός τύπος για τον υπολογισμό της λοξότητας.
- **Ομαλότητα (Smoothness)**:
  Η αλλαγή στο μέγεθος των στοιχείων πρέπει να είναι ομαλή. Συνήθως αποφεύγονται ξαφνικά άλματα στο μέγεθος των στοιχείων γιατί αυτό μπορεί να προκαλέσει λανθασμένα αποτελέσματα σε κοντινούς κόμβους.
- **Αναλογία Διαστάσεων (Aspect Ratio)**:
  Εν συντομία, ο λόγος διαστάσεων είναι ο λόγος του μεγαλύτερου μήκους ενός στοιχείου προς το μικρότερο μήκος. Ο ιδανικός λόγος διαστάσεων είναι 1. Όσο μικρότερος είναι, τόσο υψηλότερη είναι η ποιότητα ενός στοιχείου. Η μέθοδος υπολογισμού ποικίλλει ανάλογα με τον τύπο κελιού.

Σε πραγματικές εφαρμογές, τα πλέγματα συνήθως σχεδιάζονται έτσι ώστε να μην παραβιάζουν σε μεγάλο βαθμό τις παραπάνω μετρικές ποιότητας. Αυτή η παρατήρηση είναι χρήσιμη για τον σχεδιασμό της δομής δεδομένων που προτείνουμε.



Σχήμα 2.2: Παράδειγμα Ποιότητας Πλεγμάτων - Και τα δύο πλέγματα αναπαριστούν το ίδιο αεροπλάνο. Το δεξί πλέγμα είναι καλύτερης ποιότητας από το αριστερό που φαίνεται να παραβιάζει όλα τα κριτήρια ποιότητας που αναφέρθηκαν (skweness, smoothness, aspect ratio). Και τα δύο πλέγματα αποτελούνται από τον ίδιο αριθμό τριγώνων περίπου (γύρω στα 15000 τρίγωνα).

2.2 Οριοθετικοί Όγκοι

Οριοθετικός όγκος (**bounding volume**) ενός συνόλου από αντικείμενα του τρισδιάστατου χώρου ονομάζεται οποιοσδήποτε κλειστός όγκος που εξ' ολοκλήρου περικλείει τα αντικείμενα του συνόλου. Οι οριοθετικοί όγκοι χρησιμοποιούνται για να επιταχύνουν αλγορίθμους που εκτελούν γεωμετρικούς ελέγχους χρησιμοποιώντας απλούς όγκους που περικλείουν πολύπλοκα αντικείμενα. Οι έλεγχοι σε οριοθετικούς όγκους είναι τυπικά πολύ ταχύτεροι από ελέγχους στο ίδιο το στοιχείο ή αντικείμενο που περικλείουν Σε πολλές περιπτώσεις, ένας έλεγχος αρκεί για να απορριφθούν ή να επιβεβαιωθούν πολλαπλοί έλεγχοι που θα απαιτούνταν για κάθε ένα από τα στοιχεία ξεχωριστά.

Οι οριοθετικοί όγκοι βρίσκουν ευρεία εφαρμογή στους παρακάτω τομείς:

- **Ανίχνευση Ακτίνων (Ray Tracing)**:
  Οι οριοθετικοί όγκοι χρησιμοποιούνται σε ελέγχους τομής ακτίνων με αντικείμενα και σε πολλούς αλγόριθμους απόδοσης γραφικών. Για παράδειγμα, εάν η ακτίνα ή το οπτικό πεδίο της κάμερας δεν τέμνει τον οριοθετικό όγκο, τότε δεν μπορεί να τέμνει ούτε το αντικείμενο που περιέχεται μέσα. Έτσι αποφεύγονται οι αντίστοιχοι έλεγχοι, οι οποίοι κοστίζουν υπολογιστικά. Όμοια, εάν το οπτικό πεδίο της κάμερας περιέχει εξ ολοκλήρου τον οριοθετικό όγκο, το αντικείμενο, δηλαδή τα στοιχεία από τα οποία αποτελείται θα απεικονιστούν στην οθόνη χωρίς περισσότερους ελέγχους.
- **Ανίχνευση Σύγκρουσης (Collision Detection)**:
  Όμοια με πριν, όταν δύο οριοθετικοί όγκοι δε συγκρούονται/τέμνονται, τότε ούτε και τα αντικείμενα που περικλείουν δεν μπορούν να συγκρούονται.

Για τη δημιουργία οριοθετικών όγκων σύνθετων αντικειμένων, συνήθως χρησιμοποιούνται *Ιεραρχίες Οριοθετικών Όγκων* (βλ. 2.3). Δηλαδή, δενδρικές δομές δεδομένων όπου η βασική ιδέα κατασκευής τους είναι η ρίζα να περικλείει ολόκληρο το αντικείμενο ενώ τα φύλλα ένα μικρό υποσύνολο του.

Η επιλογή του τύπου οριοθετικού όγκου για μια δεδομένη εφαρμογή καθορίζεται από διάφορους παράγοντες. Τέτοιοι είναι το κόστος υπολογισμού ενός οριοθετικού όγκου για ένα αντικείμενο, το κόστος της ενημέρωσης του σε εφαρμογές στις οποίες τα αντικείμενα μπορούν να μετακινηθούν ή να αλλάξουν σχήμα, το κόστος ανίχνευσης σύγκρουσης ή υπολογισμού απόστασης και η επιθυμητή ακρίβεια για ελέγχους σύγκρουσης ή απόστασης. Η ακρίβεια αυτή σχετίζεται με τον όγκο του *κενού χώρου* που περικλείεται από τον οριοθετικό όγκο όμως δε σχετίζεται με το οριοθετημένο αντικείμενο. Τυπικά, ισχύει ο εξής συμβιβασμός: οι πιο εκλεπτυσμένοι οριοθετικοί όγκοι περικλείουν γενικά λιγότερο κενό χώρο, αλλά είναι πιο ακριβοί υπολογιστικά.

Οι πιο συνηθισμένοι τύποι οριοθετικών όγκων είναι:

- Η **Οριοθετική Σφαίρα (Bounding Sphere)**, η οποία είναι μια σφαίρα που περικλείει το αντικείμενο. Αναπαρίσταται από το κέντρο και την ακτίνα της και επιτρέπει πολύ γρήγορους ελέγχους σύγκρουσης και υπολογισμού απόστασης. Δύο σφαίρες τέμνονται όταν η απόσταση μεταξύ των κέντρων τους δεν υπερβαίνει το άθροισμα των ακτίνων τους.
- Ο **Οριοθετικός Κύλινδρος (Bounding Cylinder)**, είναι ένας κύλινδρος που περικλείει το αντικείμενο. Στις περισσότερες εφαρμογές ο άξονας του κυλίνδρου

είναι ευθυγραμμισμένος με την κατακόρυφη διεύθυνση της σκηνής. Οι κύλινδροι είναι κατάλληλοι για τρισδιάστατα αντικείμενα που μπορούν να περιστρέφονται μόνο γύρω από έναν κατακόρυφο άξονα αλλά όχι γύρω από άλλους άξονες, και κατά τα άλλα η κίνηση τους να είναι μόνο μεταφορική. Δύο κύλινδροι ευθυγραμμισμένοι με κατακόρυφο άξονα τέμνονται όταν, ταυτόχρονα, τέμνονται οι προβολές τους στον κατακόρυφο άξονα (που είναι ευθύγραμμα τμήματα), καθώς και οι προβολές τους στο οριζόντιο επίπεδο (που είναι κύκλοι). Οι δύο συνθήκες είναι εύκολο να ελεγχθούν. Στα βιντεοπαιχνίδια, οι οριοθετικοί κύλινδροι χρησιμοποιούνται συχνά ως οριοθετικοί όγκοι για χαρακτήρες που στέκονται όρθια.

- Το **Οριοθετικό Πλαίσιο (Bounding Box)**, είναι ένα ορθογώνιο παραλληλεπίπεδο που περικλείει το αντικείμενο. Στις προσομοιώσεις όπου η σκηνή αλλάζει δυναμικά, τα οριοθετικά πλαίσια προτιμώνται από άλλα σχήματα οριοθετικών όγκων (σφαίρες ή κυλίνδρους) για αντικείμενα που έχουν χονδρικά κυβοειδές σχήμα όταν ο έλεγχος σύγκρουσης πρέπει να είναι αρκετά ακριβής. Το όφελος είναι προφανές, για παράδειγμα, για αντικείμενα που ακουμπούν πάνω σε άλλα, όπως ένα αυτοκίνητο που ακουμπά στο έδαφος: μια οριοθετική σφαίρα θα έδειχνε πως το αυτοκίνητο πιθανώς να τέμνεται με το έδαφος, το οποίο στη συνέχεια θα έπρεπε να απορριφθεί από μια πιο ακριβή υπολογιστικά δοκιμή του πραγματικού μοντέλου του αυτοκινήτου. Ένα οριοθετικό πλαίσιο δείχνει αμέσως ότι το αυτοκίνητο δεν τέμνεται με το έδαφος, εξοικονομώντας έτσι την κοστοβόρα δοκιμή.

  Στη γενική περίπτωση, ένα αυθαίρετο οριοθετικό πλαίσιο ονομάζεται και **Προσανατολισμένο Οριοθετικό Πλαίσιο (Oriented Bounding Box) OBB** ή **OOBB** όταν χρησιμοποιείται το τοπικό σύστημα συντεταγμένων ενός αντικειμένου. Σε πολλές εφαρμογές το οριοθετικό πλαίσιο είναι ευθυγραμμισμένο με τους άξονες του συστήματος συντεταγμένων και ονομάζεται **Οριοθετικό Πλαίσιο Ευθυγραμμισμένο με τους Άξονες (Axis-Aligned Bounding Box)** ή **AABB**. Τα AABB είναι πιο απλά και αποδοτικά στην ανίχνευση σύγκρουσης μεταξύ τους από τα OBB, αλλά έχουν το μειονέκτημα ότι όταν το μοντέλο περιστρέφεται δεν μπορούν απλώς να περιστραφούν με αυτό, αλλά πρέπει να υπολογιστούν εκ νέου.

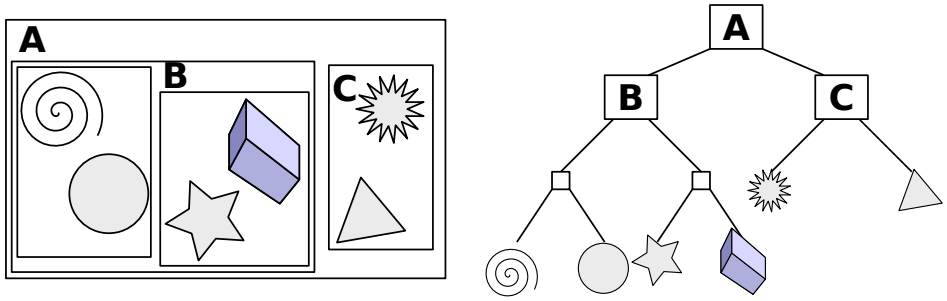
  Στην περίπτωση των δύο διαστάσεων, χρησιμοποιείται το **Ελάχιστο Οριοθετικό Παραλληλόγραμμο (Minimum Bounding Rectangle)** ή **MBR**. Το MBR είναι ειδική περίπτωση του AABB στο επίπεδο και χρησιμοποιείται συχνά για να περικλείει γεωγραφικά (ή γεωχωρικά) δεδομένα.

- Η **Οριοθετική Κάψουλα (Bounding Capsule)**, η οποία προκύπτει από τον συνολικό όγκο που περικλείει μια σφαίρα καθώς κινείται πάνω σε ένα ευθύγραμμο τμήμα (swept sphere). Ο όγκος που προκύπτει αποτελείται από έναν κύλινδρο και δύο ημισφαίρια στα άκρα του. Μπορούν να αναπαρασταθούν από το ευθύγραμμο τμήμα και το μήκος της ακτίνας της σφαίρας. Έχει χαρακτηριστικά παρόμοια με έναν κύλινδρο, αλλά είναι πιο εύκολη στη χρήση, επειδή ο έλεγχος σύγκρουσης είναι απλούστερος. Για παράδειγμα, δύο κάψουλες τέμνονται εάν η απόσταση μεταξύ των τμημάτων που τις ορίζουν είναι μικρότερη από το άθροισμα των ακτίνων τους. Αυτό ισχύει για τις αυθαίρετα προσανατολισμένες κάψουλες, γι' αυτό είναι πιο ελκυστικές από τους κυλίνδρους στην πράξη.

Στα πλαίσια αυτής της εργασίας, θα χρησιμοποιηθούν τα AABB.

2.3 Ιεραρχίες Οριοθετικών Όγκων

Η *Ιεραρχία Οριοθετικών Όγκων* (**Bounding Volume Hierarchy** ή **BVH**) είναι μια δενδρική δομή που αποθηκεύει και οργανώνει γεωμετρικά (ή χωρικά) στοιχεία (elements). Τέτοια είναι τα πολύτοπα, τα πολύεδρα, παραμετρικές καμπύλες ή επιφάνειες και άλλα. Κάθε κόμβος του δένδρου αναπαρίσταται από τον οριοθετικό όγκο του υποσυνόλου των στοιχείων που αποθηκεύει. Ο όγκος αυτός προκύπτει από τη συγχώνευση των οριοθετικών όγκων των παιδιών του κόμβου. Με αυτό τον αναδρομικό τρόπο, οργανώνεται το σύνολο των στοιχείων του δένδρου από τα φύλλα έως τη ρίζα. Συνεπώς, η ρίζα του δένδρου αναπαρίσταται από τον οριοθετικό όγκο που περικλείει όλα τα στοιχεία του δένδρου, ενώ τα φύλλα αναπαρίστανται από τους οριοθετικούς όγκους του κάθε στοιχείου ξεχωριστά (βλ. Σχήμα 2.3).



Σχήμα 2.3: Παράδειγμα μιας Ιεραρχίας Οριοθετικών Όγκων στις δύο διαστάσεις που χρησιμοποιεί ορθογώνια παραλληλόγραμμα για οριοθετικούς όγκους. Αριστερά απεικονίζονται τα αντικείμενα και τα οριοθετικά πλαίσια που περικλείουν τα αντικείμενα. Δεξιά απεικονίζεται η ιεραρχία σε μορφή δέντρου.

Γενικά, η ενθυλάκωση αντικειμένων μέσα σε οριοθετικούς όγκους επιταχύνει πολλές γεωμετρικές πράξεις πχ. collision detection, ray tracing, κλπ όπως έγινε φανερό στο 2.2, παρ' όλα αυτά δε μειώνει τον αριθμό των ελέγχων που απαιτούνται. Δηλαδή η υπολογιστική πολυπλοκότητα των αλγορίθμων παραμένει ίδια (βλ. 4.2 για ένα παράδειγμα τέτοιου αλγορίθμου). Με την οργάνωση των αντικειμένων σε ιεραρχίες οριοθετικών όγκων, ωστόσο, η υπολογιστική πολυπλοκότητα (δηλαδή το πλήθος των ελέγχων που εκτελούνται) μπορεί να μειωθεί σε λογαριθμική σε σχέση με το πλήθος των αντικειμένων. Σε μια τέτοια ιεραρχία τα αντικείμενα ενός κόμβου δε χρειάζεται να εξετασθούν αν το αποτέλεσμα του ελέγχου μπορεί να εξακριβωθεί από κάποιον πρόγονό του. Αυτή η τεχνική ονομάζεται κλάδεμα του δέντρου (tree pruning).

Για τη σχεδίαση ιεραρχιών οριοθετικών όγκων υπάρχουν πολλές επιλογές.

- Η *επιλογή του οριοθετικού όγκου* που θα χρησιμοποιείται από την ιεραρχία παίζει σημαντικό λόγο. Η επιλογή καθορίζεται από μια σύμβαση μεταξύ δύο στόχων: Από τη μία πλευρά, θα θέλαμε να χρησιμοποιήσουμε οριοθετικούς όγκους που έχουν πολύ απλό σχήμα γιατί η αποθήκευση τους στη μνήνη είναι μικρή και οι διάφοροι γεωμετρικοί έλεγχοι είναι γρήγοροι. Από την άλλη πλευρά, θα θέλαμε να έχουμε οριοθετικούς όγκους που να περικλείουν πολύ στενά τα αντικείμενα. Ο πιο κοινός τύπος οριοθετικού όγκου που χρησιμοποιείται είναι τα AABB.
- Η *τεχνική κατασκευής του δένδρου* αποτελεί επίσης σχεδιαστική επιλογή. Υπάρχουν τρεις κύριες κατηγορίες αλγορίθμων κατασκευής δένδρων:

- Οι *από πάνω προς τα κάτω (top-down)*, οι οποίοι λειτουργούν με διαμερισμό του συνόλου εισόδου σε δύο (ή περισσότερα) υποσύνολα, οριοθετώντας τα στον επιλεγμένο οριοθετικό όγκο και, στη συνέχεια, επαναλαμβάνουν την ίδια διαδικασία αναδρομικά έως ότου κάθε υποσύνολο να αποτελείται από ένα μόνο στοιχείο (δηλαδή η διαδικασία τερματίζει όταν φτάσει στα φύλλα του δέντρου).

- Οι *από κάτω προς τα πάνω (bottom-up)*, οι οποίοι ξεκινούν με το σύνολο των στοιχείων που αποθηκεύονται στα φύλλα του δέντρου και στη συνέχεια ομαδοποιούν δύο (ή περισσότερα) από αυτά για να σχηματίσουν έναν νέο εσωτερικό κόμβο. Η διαδικασία συνεχίζεται με τον ίδιο τρόπο έως ότου όλα τα στοιχεία ομαδοποιηθούν σε έναν μόνο κόμβο (τη ρίζα του δέντρου). Οι μέθοδοι από κάτω προς τα πάνω είναι πιο δύσκολοι στην υλοποίηση, αλλά είναι πιθανό να παράγουν καλύτερης ποιότητας δέντρα γενικά. Ένας ευρέως διαδεδομένος τρόπος για ομαδοποίηση των δεδομένων από κάτω προς τα πάνω είναι η χρήση *καμπυλών πλήρωσης χώρου (space filling curves)* [22] [27]. Τα στοιχεία ταξινομούνται αρχικά με βάση την καμπύλη και ομαδοποιούνται σύμφωνα με την ακολουθία που προκύπτει.

- Οι *μέθοδοι με εισαγωγή (insertion methods)*, οι οποίοι κατασκευάζουν το δέντρο εισάγοντας ένα στοιχείο τη φορά, ξεκινώντας από ένα κενό δέντρο. Η θέση που τοποθετείται ένα νέο στοιχείο επιλέγεται έτσι ώστε το δέντρο να μεγαλώσει όσο το δυνατόν λιγότερο, σύμφωνα με κάποια μετρική κόστους. Οι μέθοδοι με εισαγωγή θεωρούνται on-line μέθοδοι δεδομένου ότι δεν απαιτούν να είναι διαθέσιμα όλα τα στοιχεία πριν από την έναρξη της κατασκευής και επομένως επιτρέπουν την εκτέλεση ενημερώσεων (updates) κατά το χρόνο εκτέλεσης. Οι άλλες δύο κατηγορίες που αναφέρθηκαν θεωρούνται off-line μέθοδοι καθώς και οι δύο απαιτούν να είναι διαθέσιμα όλα τα στοιχεία πριν από την έναρξη της κατασκευής.

  Η εισαγωγή στοιχείων στο BVH συνήθως καταλήγει σε δέντρα με χειρότερη επίδοση στα ερωτήματα σε σχέση με μια πλήρη ανακατασκευή από την αρχή. Οι λύσεις που προτείνονται είναι είτε η ασύγχρονη ανακατασκευή του δέντρου, είτε η ανακατασκευή όταν ανιχνευθεί σημαντική αλλαγή (πχ υπάρχει μεγάλη επικάλυψη κοντά στα φύλλα, ή ο αριθμός των προσθαφαιρέσεων στοιχείων ξεπεράσει κάποιο κατώφλι, ή άλλες πιο εκλεπτυσμένες ευρετικές μεθόδους).

- Ο *βαθμός* (αριθμός των παιδιών των κόμβων) του δένδρου. Ένα δέντρο χαμηλού βαθμού θα έχει μεγαλύτερο ύψος. Αυτό αυξάνει τον χρόνο διάσχισης από τη ρίζα έως τα φύλλα. Από την άλλη πλευρά, λιγότεροι υπολογισμοί πρέπει να δαπανηθούν σε κάθε κόμβο που επισκέπτεται ο αλγόριθμος για να ελέγξει ποια παιδιά θα επισκεφθεί και με ποια σειρά. Το αντίθετο ισχύει για ένα δέντρο υψηλού βαθμού, δηλαδή αν και το δέντρο θα είναι μικρότερου ύψους, δαπανάται περισσότερη δουλειά σε κάθε κόμβο. Στην πράξη, τα δυαδικά δέντρα (βαθμός = 2) είναι μακράν τα πιο κοινά. Ένας από τους κύριους λόγους είναι ότι τα δυαδικά δέντρα είναι πιο εύκολο να κατασκευαστούν.

- *Η μέθοδος διάσχισης* του δένδρου για την απάντηση γεωμετρικών ερωτημάτων. Όταν θα πρέπει να γίνει επίσκεψη σε περισσότερα από ένα από τα παιδιά ενός κόμβου, η σειρά με την οποία αυτά θα ελεγχθούν δεν επηρεάζει την ορθότητα του

αλγορίθμου, όμως έχει αντίκτυπο στην επίδοση του.

Τέλος, αναφέρουμε κάποιες επιθυμητές ιδιότητες [16] μιας Ιεραρχίας Οριοθετικών Όγκων οι οποίες πρέπει να ληφθούν υπόψη κατά τον σχεδιασμό της για τις διάφορες εφαρμογές:

- Τα στοιχεία που περιλαμβάνει οποιοδήποτε υπο-δέντρο θα πρέπει να είναι κοντά το ένα στο άλλο (χωρικά). Όσο πιο χαμηλά στο δέντρο, τόσο πιο κοντά θα πρέπει να βρίσκονται τα αντικείμενα.
- Ο όγκος της επικάλυψης των αδελφικών κόμβων πρέπει να είναι ελάχιστος
- Κάθε κόμβος του δέντρου πρέπει να έχει τον ελάχιστο δυνατό όγκο.
- Το άθροισμα όλων των οριοθετικών όγκων πρέπει να είναι ελάχιστο.
- Θα πρέπει να δοθεί μεγαλύτερη προσοχή στους κόμβους κοντά στη ρίζα του BVH. Το κλάδεμα ενός κόμβου κοντά στη ρίζα του δέντρου αφαιρεί περισσότερα αντικείμενα από περαιτέρω εξέταση.
- Το BVH θα πρέπει να είναι όσο πιο ισορροπημένο γίνεται. Η εξισορρόπηση επιτρέπει να κλαδευτεί όσο το δυνατόν μεγαλύτερο μέρος του BVH όποτε δε διασχίζεται ένα κλαδί.

Κεφάλαιο 3

Σχετική Βιβλιογραφία

Τα ερωτήματα εγγύτητας και η ανίχνευση σύγκρουσης διερευνώνται εκτενώς για δεκαετίες από ερευνητές στα γραφικά υπολογιστών, στη ρομποτική, στις προσομοίωσες, την υπολογιστική γεωμετρία και τα κινούμενα σχέδια υπολογιστών. Εκτός από την ανίχνευση σύγκρουσης και τον υπολογισμό απόστασης, υπάρχει σημαντική βιβλιογραφία σχετική με τις Ιεραρχίες Οριοθετικών Όγκων (BVH) και τις διάφορες εφαρμογές τους. Σε αυτό το κεφάλαιο κάνουμε αναφορά σε άρθρα της βιβλιογραφίας που μελετούν το ίδιο ή παρόμοιο πρόβλημα με το δικό μας και επισημαίνουμε τις τεχνικές που χρησιμοποιούνται και στη δική μας μεθοδολογία.

3.1 Υπολογισμός Αποστάσεων προς Αντικείμενα

3.1.1 Κοντινότερος Κόμβος Πλέγματος από Σημείο

Αρχικά μελετάμε το παρακάτω πρόβλημα: *¨Δοθέντος ενός πλέγματος και ενός σημείου (query-point), ζητείται να βρεθεί ο κοντινότερος προς το σημείο* ***κόμβος*** *του πλέγματος¨*. Το πρόβλημα αυτό ανάγεται στο κλασικό πρόβλημα εύρεσης των k *κοντινότερων γειτόνων* από ένα σύνολο σημείων (kNN, όπου $k = 1$).

Για το πρόβλημα αυτό έχουν προταθεί διάφορες λύσεις. Για παράδειγμα η χρήση διαγραμμάτων Voronoi στην περίπτωση δεδομένων με μικρό αριθμό διαστάσεων [4]. Οι σημαντικότερες όμως συνεισφορές για το συγκεκριμένο πρόβλημα είναι αυτές του [5] με την εισαγωγή του *kD-Tree*, και του [40] με την εισαγωγή του *VP-Tree* (βλ. για παράδειγμα το [39] όπου γίνεται χρήση του *kD-Tree* για τον υπολογισμό του κοντινότερου κόμβου ενός τριγωνικού πλέγματος από σημείο).

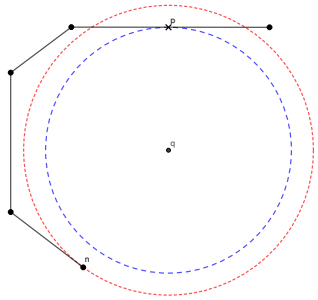
Τα δύο παραπάνω δυαδικά δέντρα αναζήτησης απαιτούν $O(n * logn)$ χρόνο προεπεξεργασίας των δεδομένων (για την κατασκευή τους από το σύνολο των n σημείων του τρισδιάστατου χώρου) και έπειτα $O(logn)$ για την απάντηση ερωτημάτων κοντινότερου γείτονα, στη μέση περίπτωση. Στο ίδιο πλαίσιο εργασίας αναπτύσσεται ο πρώτος εκ των δύο αλγορίθμων που προτείνουμε, για το δικό μας πρόβλημα, όπου κατασκευάζουμε μια δομή δεδομένων και στη συνέχεια κάνουμε ερωτήματα κοντινότερου γείτονα για να υπολογίσουμε την απόσταση.

Πιο συγκεκριμένα, για την κατασκευή του *kD-Tree* το σύνολο των σημείων ενός κόμβου διχοτομείται σε δύο υποσύνολα που αποτελούν τα δύο παιδιά του κόμβου. Η διαδικασία

αυτή επαναλαμβάνεται αναδρομικά:

- Η διχοτόμηση γίνεται με βάση κάποιον άξονα ο οποίος εναλλάσσεται κυκλικά (round-robin) για τα διαδοχικά επίπεδα του δέντρου. Για παράδειγμα, στις τρεις διαστάσεις, ο διαχωρισμός γίνεται πρώτα με βάση τον άξονα x, έπειτα με βάση τον άξονα y, έπειτα με τον z, ξανά με βάση τον x και ούτω καθεξής.
- Το κριτήριο για τον διαχωρισμό του συνόλου των σημείων V, ενός κόμβου, σε δύο υποσύνολα L, R είναι το εξής: Χωρίς βλάβη της γενικότητας θεωρούμε πως ο άξονας διαχωρισμού είναι ο x. Αν m είναι η διάμεσος των x-συντεταγμένων των σημείων, τότε $L = \{p \in V \mid p_x \leq m\}$ και $R = \{p \in V \mid p_x >= m\}$. Τα σύνολα L, R επιλέγονται ώστε να έχουν το ίδιο πλήθος στοιχείων ή να διαφέρουν κατά ένα. Όμοια προκύπτει το κριτήριο και για τους άλλους άξονες. Η διάμεσος m είναι πληροφορία που αποθηκεύεται για κάθε κόμβο του δέντρου και χρησιμοποιείται στα ερωτήματα εύρεσης του κοντινότερου γείτονα.

Έτσι προκύπτει ένα σχεδόν πλήρες, ισορροπημένο, δυαδικό δέντρο. Οι ιδέες 1) της διχοτόμησης του συνόλου των δεδομένων με κυκλική εναλλαγή του άξονα διαχωρισμού και 2) η χρήση της διαμέσου των αντίστοιχων συντεταγμένων για τον διαχωρισμό χρησιμοποιούνται *αυτούσιες* και στη δική μας μεθοδολογία για την κατασκευή μιας Ιεραρχίας Οριοθετικών Όγκων. Η διαφορά είναι πως χειριζόμαστε χωρικά δεδομένα, και όχι σημεία, επομένως επιλέγεται ένα *αντιπροσωπευτικό σημείο* για κάθε χωρικό αντικείμενο και εφαρμόζεται η ίδια μέθοδος κατασκευής της ιεραρχίας.



Σχήμα 3.1: Παράδειγμα όπου ο κοντινότερος *κόμβος n* ενός πλέγματος διαφέρει από το κοντινότερο *σημείο p*, δοθέντος του query-point q.

Για την κατασκευή του *VP-Tree*, ξανά, το σύνολο των σημείων ενός κόμβου διχοτομείται σε δύο υποσύνολα αναδρομικά:

- Αρχικά, επιλέγεται ένα τυχαίο σημείο από το σύνολο V των σημείων του κόμβου. Το σημείο αυτό ονομάζεται *vantage-point (vp)* και αποτελεί πληροφορία που αποθηκεύεται στον κόμβο.
- Έπειτα υπολογίζονται οι αποστάσεις όλων των υπολοίπων σημείων προς το vp και η διάμεσος m, που επίσης αποθηκεύεται στον κόμβο.
- Αν m είναι η διάμεσος των αποστάσεων που υπολογίστηκαν τότε το σύνολο του αριστερού παιδιού είναι το $L = \{p \in V \mid distance(p, vp) \leq m\}$ και του δεξιού παιδιού $R = \{p \in V \mid distance(p, vp) \geq m\}$

Ο τυπικός τρόπος για την απάντηση ερωτημάτων κοντινότερου κόμβου είναι, αρχικά, ο υπολογισμός μιας εκτίμησης αυτού διασχίζοντας το δέντρο από τη ρίζα έως κάποιο φύλλο. Το φύλλο αποτελεί την πρώτη εκτίμηση κοντινότερου κόμβου, ενώ η επιλογή του μονοπατιού από τη ρίζα μέχρι το φύλλο αποτελεί σχεδιαστική επιλογή και χρησιμοποιούνται ευρετικοί μέθοδοι (heuristics). Έπειτα μια σφαίρα εκτείνεται από το query-point έως το φύλλο και, με τη βοήθεια του δέντρου, από τα υπόλοιπα σημεία ελέγχονται μόνο αυτά που βρίσκονται εντός της σφαίρας. Η ακτίνα της σφαίρας μικραίνει όσο ανακαλύπτονται όλο και κοντινότεροι κόμβοι. Αυτή η διαδικασία λέγεται *κλάδεμα (pruning)* του δέντρου, και μειώνει τον χώρο αναζήτησης με

το να απορρίπτει περιοχές που δεν μπορούν να περιέχουν τον κοντινότερο γείτονα. Με αυτόν τον τρόπο, ο αριθμός των αναμενόμενων συγκρίσεων είναι της τάξης $O(logn)$ για τη μέση περίπτωση. Με ίδιο μοτίβο λογικής σχεδιάζεται και η διαδικασία αναζήτησης κοντινότερου γείτονα για χωρικά δεδομένα που προτείνουμε.

Το παραπάνω πρόβλημα δεν πρέπει να συγχέεται με το πρόβλημα εύρεσης του κοντινότερου *σημείου* ενός πολυγωνικού πλέγματος από κάποιο query-point. Γενικά ο κοντινότερος *κόμβος* και το κοντινότερο *σημείο* ενός πλέγματος μπορούν να απέχουν πολύ μεταξύ τους όπως φαίνεται στο Σχήμα 3.1: Η πολυγωνική γραμμή αντιπροσωπεύει το σύνορο ενός υποθετικού πλέγματος στις δύο διαστάσεις και η ελάχιστη απόσταση του πλέγματος από το σημείο q (ακτίνα του κόκκινου κύκλου) διαφέρει από την απόσταση του κοντινότερου κόμβου (ακτίνα του μπλε κύκλου).

3.1.2 Απόσταση Σημείου από Πολυγωνικό Πλέγμα

Η περιγραφή του προβλήματος είναι η ακόλουθη: *¨Δοθέντος ενός πλέγματος και ενός σημείου του χώρου (query-point), ζητείται να βρεθεί το κοντινότερο **σημείο** που ανήκει στο πλέγμα¨*. Τα δύο παραπάνω σημεία ορίζουν την Ευκλείδεια απόσταση πλέγματος και σημείου, η οποία ονομάζεται και απόσταση διαχωρισμού (πλέγματος και σημείου).

Αν και το πρόβλημα υπολογισμού της Ευκλείδειας απόστασης δύο αντικειμένων στο χώρο αποτελεί γενίκευση του παραπάνω προβλήματος, στο [23] υποστηρίζεται ότι αξίζει να μελετηθεί ξεχωριστά γιατί μπορούν να αναπτυχθούν πιο αποδοτικοί αλγόριθμοι. Στην προαναφερθείσα δημοσίευση, χρησιμοποιείται μια ιεραρχία οριοθετικών όγκων πολλαπλών αναλύσεων που δημιουργείται από γεωμετρική απλοποίηση του πλέγματος. Καθώς η ανάλυση του πλέγματος βελτιώνεται, ολοένα και πιο ακριβής γίνεται η εκτίμηση του κοντινότερου σημείου, ενώ ταυτόχρονα ο αλγόριθμος σε κάθε βήμα μπορεί να παρέχει τα άνω και κάτω φράγματα του σφάλματος της εκτίμησης. Οι οριοθετικοί όγκοι που χρησιμοποιούνται στην ιεραρχία είναι τρίγωνα που σαρώνονται από μια σφαίρα (triangles swept by sphere) και αποτελούν γενίκευση της οριοθετικής κάψουλας.

3.1.3 Απόσταση Δύο Αντικειμένων

Η Ευκλείδεια απόσταση δύο αντικειμένων ορίζεται ως *¨το μήκος του μικρότερου ευθυγράμμου τμήματος που ενώνει τα δύο αντικείμενα¨*.

Στη βιβλιογραφία αλγόριθμοι που υπολογίζουν απευθείας την απόσταση δύο αντικειμένων επικεντρώνονται σε κυρτά αντικείμενα. Έστω $M = M_1 + M_2$ είναι το σύνολο των κορυφών και των δύο αντικειμένων, για την περίπτωση των δύο διαστάσεων στο [38] δίνεται ένας αλγόριθμος $O(log^2 M)$, ενώ στα [15] και [12] δίνονται αλγόριθμοι $O(logM)$ για τον υπολογισμό της απόστασης κυρτών πολυγώνων. Για την περίπτωση των τριών διαστάσεων η απόσταση δύο κυρτών πολυέδρων υπολογίζεται στο [14] σε $O(M)$. Στο [7] ο ίδιος υπολογισμός γίνεται με την αναγωγή του προβλήματος σε πρόβλημα ελαχιστοποίησης μη γραμμικής συνάρτησης υπό περιορισμούς. Στις προηγούμενες δημοσιεύσεις δίνεται περισσότερη έμφαση στην ασυμπτωτική ανάλυση των αλγορίθμων και δεν είναι ξεκάθαρο αν οι αλγόριθμοι είναι αποδοτικοί για πρακτικά προβλήματα όπου το M δεν είναι υπερβολικά μεγάλο (για παράδειγμα για την απόσταση μεταξύ τριγώνων

είναι $M = 6$). Όλοι οι παραπάνω θεωρητικοί αλγόριθμοι κρύβουν άγνωστες, και πιθανόν, μεγάλες σταθερές πολυπλοκότητας χρόνου. Δυστυχώς, ελάχιστα, έως καθόλου, υπολογιστικά πειράματα έχουν γίνει. Τέλος, στο [18] περιγράφεται ένας αποδοτικός αλγόριθμος για το ίδιο πρόβλημα, γνωστός ως GJK (Gilbert-Johnson-Keerthi). Ο αλγόριθμος αυτός είναι ο πλέον διαδεδομένος σε εφαρμογές υπολογισμού αποστάσεων και ανίχνευσης σύγκρουσης, καθώς είναι αξιόπιστος και αποδοτικός ειδικά για τις τρεις διαστάσεις. Το υπολογιστικό του κόστος είναι γραμμικό σε σχέση με το πλήθος των κορυφών, $O(M)$, με μικρό συντελεστή πολυπλοκότητας, σύμφωνα με τα πειράματα που έχουν γίνει.

Σε πραγματικές εφαρμογές, τα αντικείμενα συνήθως δεν είναι κυρτά. Για τον υπολογισμό της απόστασης μεταξύ μη-κυρτών αντικειμένων, μπορούμε να διαχωρίσουμε τα αντικείμενα σε κυρτά μέρη και στη συνέχεια να χρησιμοποιήσουμε οποιονδήποτε από τους παραπάνω αλγορίθμους. Τότε, η απόσταση μεταξύ των δύο αντικειμένων είναι η μικρότερη απόσταση μεταξύ οποιουδήποτε ζεύγους κυρτών περιοχών. Ωστόσο, μια απλοϊκή υλοποίηση που εξετάζει όλα αυτά τα ζεύγη έχει πολυπλοκότητα $O(n*m)$, όπου n και m είναι οι αριθμοί των κυρτών περιοχών του κάθε αντικειμένου.

Για να επιταχυνθεί η αναζήτηση έχουν χρησιμοποιηθεί ιεραρχίες οριοθετικών όγκων

- με οριοθετικές σφαίρες [36],
- με προσανατολισμένα οριοθετικά πλαίσια [26],
- με οριοθετικές κάψουλες [31].

Ξεκινώντας με δύο ιεραρχίες οριοθετικών όγκων, μια για κάθε αντικείμενο αντίστοιχα, οι ιεραρχίες διασχίζονται αναδρομικά. Κατά τη διάσχιση, ζεύγη οριοθετικών όγκων ελέγχονται για να καθοριστεί εάν η τρέχουσα εκτίμηση της ελάχιστης απόστασης μπορεί να βελτιωθεί. Στην περίπτωση που δεν μπορεί να βελτιωθεί, τότε τα αντίστοιχα υποδέντρα κλαδεύονται (pruning) και δεν απαιτούνται περαιτέρω έλεγχοι. Στην αντίθετη περίπτωση η διάσχιση των δέντρων πρέπει να συνεχιστεί. Όταν η αναζήτηση φτάσει στα φύλλα των δέντρων, τότε γίνονται οι έλεγχοι στοιχείο - προς - στοιχείο οι οποίοι συνήθως είναι πιο ακριβοί υπολογιστικά από τους ελέγχους μεταξύ οριοθετικών όγκων.

Το ίδιο πλαίσιο εργασίας ακολουθεί και η μεθοδολογία που προτείνουμε, δηλαδή κατασκευάζουμε μια ιεραρχία οριοθετικών όγκων και υλοποιούμε αλγορίθμους που προσπαθούν να μειώσουν τον χώρο αναζήτησης.

3.2 Οργάνωση Χωρικών Δεδομένων σε Ιεραρχίες

Η ιεραρχική οργάνωση των αντικειμένων του χώρου σε δενδρικές δομές δεδομένων αποτελεί κοινή μεθοδολογία για πολλά προβλήματα της Υπολογιστικής Γεωμετρίας. Τέτοιες δομές χρησιμοποιούνται σε διάφορες εφαρμογές για να επιταχύνουν τη διαδικασία επίλυσης του εκάστοτε προβλήματος, όπως ακριβώς συμβαίνει και στη μεθοδολογία που προτείνουμε. Για τον λόγο αυτό πολλές φορές αναφέρονται και ως *δομές δεδομένων επιτάχυνσης (acceleration data structures)*.

Χαρακτηριστικά παραδείγματα αποτελούν:

- Οι δομές *VP-Tree*, *kD-Tree* που περιγράφονται στο 3.1.1 και χρησιμοποιούνται για το πρόβλημα εύρεσης των k κοντινότερων γειτόνων (*kNN*).

- Το *R-Tree* που περιγράφεται στο [24], και χρησιμοποιείται ως ευρετήριο σε βάσεις δεδομένων που αποθηκεύουν χωρικά δεδομένα/αντικείμενα. Το R-Tree χρησιμοποιεί *ελάχιστα οριοθετικά παραλληλόγραμμα (MBR)* για την οργάνωση των χωρικών αντικειμένων, από τα φύλλα έως τη ρίζα και ανήκει στην οικογένεια των Ιεραρχιών Οριοθετικών Όγκων (BVH).

  Υποστηρίζει εισαγωγή, διαγραφή και ενημέρωση δεδομένων καθώς επίσης και ερωτήματα εύρεσης των αντικειμένων που βρίσκονται εντός μιας δοθείσας περιοχής. Γενικά, το R-Tree δεν είναι δυαδικό δέντρο και ο τρόπος διάσχισης του δεν είναι τετριμμένος. Στο [37] περιγράφεται ένας αλγόριθμος εύρεσης των k κοντινότερων αντικειμένων από κάποιο σημείο p, για το R-Tree. Στον αλγόριθμο αυτόν εισάγωνται οι μετρικές απόστασης *MINDIST* και *MINMAXDIST* βάση των οποίων ορίζεται η σειρά της αναζήτησης με κάποια προτεραιότητα. Οι μετρικές αποτελούν το κάτω και άνω φράγμα, αντίστοιχα, της απόστασης του αντικειμένου από το σημείο p.

- Το *BSP-Tree*, που περιγράφεται στο [17], χρησιμοποιείται για την απόδοση γραφικών καθώς μπορεί να δώσει χωρικές πληροφορίες για τα αντικείμενα μιας σκηνής. Συνήθως τα αντικείμενα είναι πολύγωνα στον τρισδιάστατο χώρο. Η δομή του επιτρέπει την εύκολη διάσχιση των αντικειμένων μιας σκηνής από πίσω προς τα εμπρός για δοσμένη θέση θέασης.

  Συγκεκριμένα, η κατασκευή του γίνεται μια μόνο φορά για κάποια στατική σκηνή και το ίδιο δέντρο μπορεί να χρησιμοποιηθεί για οποιαδήποτε γωνία θέασης. Για την κατασκευή του, ο χώρος διαχωρίζεται αναδρομικά σε δύο ημι-χώρους (half-spaces) με βάση κάποιο επίπεδο. Τα αντικείμενα (ολόκληρα ή κάποιο μέρος τους) τοποθετούνται στο αριστερό ή δεξί παιδί ανάλογα με τον ημι-χώρο στον οποίο ανήκουν. Παρόλο που αποθηκεύει χωρικά δεδομένα (και όχι σημεία), το BSP-Tree θεωρείται γενίκευση του kD-Tree διότι τα επίπεδα που χωρίζουν τον χώρο μπορεί να έχουν οποιονδήποτε προσανατολισμό σε αντίθεση με τα ευθυγραμμισμένα με τους άξονες επίπεδα του kD-Tree.

- Το *OBBTree*, που περιγράφεται στο [21] ανήκει, επίσης, στην οικογένεια των Ιεραρχιών Οριοθετικών Όγκων και χρησιμοποιεί *προσανατολισμένα οριοθετικά πλαίσια (OBB)*. Χρησιμοποιείται για την ανίχνευση σύγκρουσης αντικειμένων που αναπαρίστανται από πολυγωνικά πλέγματα. Για την κατασκευή του δέντρου είναι απαραίτητα:

  1. Μια διαδικασία που για δοσμένο σύνολο πολυγώνων υπολογίζει κάποιο OBB με παρόμοιες διαστάσεις και προσανατολισμό ώστε να περικλείει τα πολύγωνα.

  2. Ένας τρόπος οργάνωσης των OBB σε ιεραρχία.

  Στην παραπάνω δημοσίευση, για τον υπολογισμό του OBB αρχικά κάθε πολύγωνο διασπάται σε τρίγωνα και έπειτα χρησιμοποιούνται στατιστικά πρώτης και δεύτερης τάξης που περιγράφουν τις συντεταγμένες των κόμβων (μέσος όρος και πίνακας συνδιακύμανσης). Έτσι υπολογίζεται ένα ορθοκανονικό σύστημα συντεταγμένων για τον προσανατολισμό του OBB και το μέγεθος του ορίζεται από τα ακραία σημεία κατά μήκος κάθε άξονα.

  Για την οργάνωση των OBB σε ιεραρχία προτείνεται μια μέθοδος από πάνω προς

τα κάτω: ο μεγαλύτερος άξονας του OBB χωρίζεται με βάση ένα επίπεδο κάθετο προς τον άξονα και τα πολύγωνα διαχωρίζονται ανάλογα τη μεριά του επιπέδου που αντιστοιχίζεται το κέντρο τους. Η συντεταγμένη του επιπέδου διαχωρισμού κατά μήκος αυτού του άξονα επιλέγεται να είναι η μέση τιμή των συντεταγμένων των κόμβων. Αντί για τη μέση τιμή μπορεί να χρησιμοποιηθεί η διάμεσος, οπότε το δέντρο που θα προκύψει είναι ισορροπημένο. Ο χρόνος κατασκευής του δέντρου με τον παραπάνω αλγόριθμο είναι της τάξης $O(nlogn)$, για ένα σετ με n τρίγωνα.

Έχοντας τα OBBTree για δύο αντικείμενα ο αλγόριθμος για τον έλεγχο σύγκρουσης, τυπικά, ξοδεύει τον περισσότερο χρόνο ελέγχοντας ζευγάρια από OBB για επικάλυψη. Για τον έλεγχο επικάλυψης δύο OBB σχεδιάζεται ειδική ρουτίνα όπου γίνεται χρήση του *θεωρήματος του διαχωριστικού άξονα (separating axis theorem - **SAT**)* [20].

- Το *Octree*, που περιγράφεται στο [34] χρησιμοποιείται σε πολλές εφαρμογές για απόδοση γραφικών, ανίχνευση σύγκρουσης, ανίχνευση ακτίνων, ως χωρικό ευρετήριο, ως γεννήτρια πλεγμάτων κ.α. Ένας κόμβος του δέντρου μπορεί να έχει έως και οκτώ παιδιά που κάθε ένα αντιπροσωπεύει ένα ογδοημόριο του τρισδιάστατου χώρου που καταλαμβάνει ο πατρικός κόμβος. Τα ογδοημόρια είναι οριοθετικά πλαίσια ευθυγραμμισμένα με τους άξονες.

  Συνήθως για την κατασκευή του, αρχικά ορίζεται το ελάχιστο μέγεθος που μπορεί να έχει ένα ογδοημόριο και ο χώρος που καταλαμβάνει η ρίζα του δέντρου. Έπειτα κάθε κόμβος του δέντρου διαιρείται σε ογδοημόρια μέχρι είτε να μην περιέχεται κανένα αντικείμενο εντός του ογδοημορίου, είτε το ογδοημόριο να πάρει το ελάχιστο δυνατό μέγεθος. Οι κόμβοι του δέντρου που δεν περιέχουν τίποτα στο εσωτερικό τους μπορούν να αγνοηθούν. Έτσι, οποιοδήποτε τρισδιάστατο αντικείμενο μπορεί να αναπαρασταθεί με οσοδήποτε μεγάλη ανάλυση (resolution).

  Στην παραπάνω δημοσίευση περιγράφονται επίσης αλγόριθμοι για τις διάφορες πράξεις μεταξύ αντικειμένων που αναπαρίστανται από Octrees. Τέτοιες είναι οι πράξεις της ένωσης, τομής, διαφοράς, οι γεωμετρικές πράξεις μεταφοράς, περιστροφής, κλιμάκωσης και η ανίχνευση σύγκρουσης και απόδοσης γραφικών από οποιαδήποτε γωνία θέασης.

Πολλές από τις τεχνικές που αναφέρονται παραπάνω ακολουθούνται και στη δική μας μεθοδολογία. Παραδείγματα αποτελούν η χρήση οριοθετικών όγκων και η αναδρομική οργάνωση των αντικειμένων σε μια ιεραρχία, η μοντελοποίηση του προβλήματος ως πρόβλημα κοντινότερων γειτόνων για χωρικά δεδομένα και ο ορισμός της σειράς επίσκεψης των κόμβων της ιεραρχίας, κατά την αναζήτηση, με βάση μια μετρική απόστασης.

Τέλος, αξίζει να σημειωθεί ότι στη βιβλιογραφία γίνεται λόγος για την ποιότητα των Ιεραρχιών Οριοθετικών Όγκων. Ορίζονται μετρικές ποιότητας που χρησιμεύουν τόσο στην αξιολόγηση όσο και στην κατασκευή των ιεραρχικών δομών. Επί του παρόντος, δεν υπάρχει πρακτικός τρόπος για τον προσδιορισμό του βέλτιστου διαχωρισμού των αντικειμένων ενός κόμβου σε δύο υποσύνολα. Έτσι, δεν είναι γνωστό ποια αντικείμενα πρέπει να ομαδοποιηθούν μαζί με άλλα για να επιτευχθεί η καλύτερη δυνατή επίδοση των αλγορίθμων. Η πιο διαδεδομένη μετρική κόστους είναι η *SAH (Surface Area Heuristic)* [19] [33] η οποία, περιληπτικά, ¨επιβραβεύει¨ μικρούς χωρικά κόμβους με πολλά αντικείμενα και ¨τιμωρεί¨ μεγάλους κόμβους με λίγα τρίγωνα. Επίσης, στο [3]

εισάγωνται οι μετρικές *EPO (End-point Overlap)* και *LCV (Leaf Count Variability)* ως συμπληρωματικές της SAH για να εξηγήσουν την παρατηρούμενη επίδοση των δέντρων σε περιπτώσεις όπου η SAH αδυνατεί να δικαιολογήσει την επίδοση κάποιων δέντρων συγκριτικά με άλλα.

Κεφάλαιο 4

Μεθοδολογία

Σε αυτή την εργασία κάνουμε την υπόθεση ότι τα αντικείμενα του τρισδιάστατου χώρου αναπαρίστανται από την εξωτερική τους επιφάνεια η οποία περιγράφεται από τριγωνικά πλέγματα. Αυτή η υπόθεση δεν είναι περιοριστική, καθώς η προσαρμογή της μεθοδολογίας που ακολουθούμε σε περιγραφές άλλου τύπου δεν είναι δύσκολη. Για την εφαρμογή της μεθοδολογίας θα πρέπει:

1. Είτε οι αναπαραστάσεις διαφορετικού τύπου να μετατραπούν σε αναπαράσταση με τριγωνικό πλέγμα (αυτή η μετατροπή είναι πάντοτε εφικτή, για παράδειγμα, από πολυγωνικά πλέγματα).
2. Είτε να δοθούν δύο ρουτίνες για τα στοιχεία από τα οποία αποτελείται η αναπαράσταση. Η μία να υπολογίζει το AABB του στοιχείου ενώ η άλλη να υπολογίζει την Ευκλείδεια απόσταση δύο τέτοιων στοιχείων.

Επιπλέον, σημειώνουμε ότι η χρήση επιφανειακών αναπαραστάσεων για τα αντικείμενα υπονοεί πως δε θα γίνει ανίχνευση σύγκρουσης εάν το ένα αντικείμενο περικλείει εντελώς το άλλο, αλλά τέτοιες καταστάσεις αποφεύγονται σε πολλές εφαρμογές.

4.1 Στοιχειώδεις Γεωμετρικές Πράξεις

4.1.1 Ευκλείδεια Απόσταση δύο Τριγώνων

Η Ευκλείδεια απόσταση $d(P, Q)$ μεταξύ δύο τριγώνων P και Q ορίζεται ως:

$$d(P, Q) = \min_{p \in P, q \in Q} \|p - q\|$$

όπου με $\|p - q\|$ συμβολίζεται η Ευκλείδεια απόσταση των σημείων p και q.

Για τον υπολογισμό της απόστασης δύο τριγώνων του τρισδιάστατου χώρου χρησιμοποιούμε τη ρουτίνα `TriDist()` από το πακέτο *PQP* [30] [31].

Μια σύντομη περιγραφή της ρουτίνας είναι η ακόλουθη: Αρχικά υπολογίζονται οι αποστάσεις κάθε ζεύγους πλευρών, δηλαδή 9 συνδυασμοί. Ο υπολογισμός της απόστασης ευθυγράμμων τμημάτων περιγράφεται στο [32]. Έστω p, q είναι τα σημεία με τη μικρότερη απόσταση για κάποιο ζεύγος πλευρών. Το διάνυσμα $\vec{pq}$ ορίζει μια πλάκα (ένα επίπεδο κάθετο στο $\vec{pq}$ που διέρχεται από το p και ένα επίπεδο κάθετο στο $\vec{pq}$ που

διέρχεται από το q). Αν η τρίτη κορυφή του κάθε τριγώνου (αυτή που δεν ανήκει στην πλευρά που εξετάζεται) είναι εκτός της πλάκας, τότε τα p, q είναι τα κοντινότερα σημεία των δύο τριγώνων και η ρουτίνα τερματίζει. Ταυτόχρονα με αυτούς τους ελέγχους, η ρουτίνα εξετάζει αν τα τρίγωνα είναι μη-επικαλυπτόμενα. Ακόμη και αν αποτύχουν οι παραπάνω έλεγχοι, η ρουτίνα αποθηκεύει τη μικρότερη απόσταση που υπολογίστηκε. Στην περίπτωση που δε βρέθηκε ζεύγος πλευρών με τις τρίτες κορυφές εκτός της πλάκας, θα πρέπει να ισχύει ένα από τα παρακάτω:

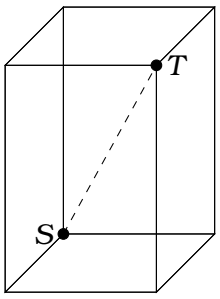
1. Το ένα από τα κοντινότερα σημεία είναι κορυφή του ενός τριγώνου και το άλλο βρίσκεται σε εσωτερικό σημείο του άλλου τριγώνου.
2. Τα τρίγωνα συγκρούονται, οπότε η απόσταση είναι 0.
3. Μια πλευρά του ενός τριγώνου είναι παράλληλη στο επίπεδο του άλλου τριγώνου. Τότε, οι παραπάνω 9 έλεγχοι πλευρών περιέχουν τα κοντινότερα σημεία.
4. Ένα ή και τα δύο τρίγωνα είναι εκφυλισμένα. Τότε, πάλι, οι παραπάνω 9 έλεγχοι περιέχουν τα κοντινότερα σημεία.

Πρώτα εξετάζεται η περίπτωση 1: Προβάλλονται οι κορυφές του ενός τριγώνου στο επίπεδο του άλλου. Αν οι τρεις κορυφές βρίσκονται από την ίδια μεριά του επιπέδου η ρουτίνα βρίσκει την κορυφή με τη μικρότερη προβολή και επίσης σημειώνει πως τα τρίγωνα είναι μη-επικαλυπτόμενα. Για την κορυφή με τη μικρότερη προβολή, εξετάζεται αν το προβαλλόμενο σημείο είναι εντός του άλλου τριγώνου. Αν ισχύει αυτό, τότε αυτή είναι η απόσταση των τριγώνων και η ρουτίνα τερματίζει. Η ίδια διαδικασία επαναλαμβάνεται και για τις κορυφές του άλλου τριγώνου. Αν δεν ισχύει η πρώτη περίπτωση, τότε αν τα τρίγωνα βρέθηκαν να είναι μη-επικαλυπτόμενα ισχύουν οι περιπτώσεις 3,4 αλλιώς η περίπτωση 2.

4.1.2 Γεωμετρικές Πράξεις για ΑΑΒΒ

Ένα οριοθετικό πλαίσιο ευθυγραμμισμένο με τους άξονες (ΑΑΒΒ) είναι, απλώς, ένα ορθογώνιο παραλληλεπίπεδο του οποίου οι έδρες είναι κάθετες σε ένα από τα διανύσματα βάσης.

Ένα ΑΑΒΒ αναπαριστάται στη μνήμη από τα άκρα της κύριας διαγωνίου του. Δηλαδή το σημείο S με τις μικρότερες x, y, z συντεταγμένες και το σημείο T με τις μεγαλύτερες x, y, z συντεταγμένες.



Σχήμα 4.1: Αναπαράσταση ενός ΑΑΒΒ

Υπολογισμός Απόστασης AABB-AABB

Έστω τα ευθυγραμμισμένα με τους άξονες πλαίσια P και Q στον $\mathbb{R}^3$. Η Ευκλείδεια απόσταση $d(P, Q)$ μεταξύ των πλαισίων P και Q ορίζεται ως:

$$d(P, Q) = \min_{p \in P, q \in Q} \|p - q\|$$

όπου με $\|p - q\|$ συμβολίζεται η Ευκλείδεια απόσταση των σημείων p και q.

Για τον υπολογισμό της απόστασης μεταξύ δύο τέτοιων πλαισίων, εκμεταλλευόμαστε το γεγονός ότι οι πλευρές τους είναι ευθυγραμμισμένες με του άξονες. Έτσι μπορούμε να υπολογίσουμε την απόσταση χωρίζοντας τη στις συνιστώσες της. Τελικά, προκύπτει ο αλγόριθμος 1, όμοιος με αυτόν που περιγράφεται στο [29]. Στον παραπάνω αλγόριθμο ο τελεστής ¨−¨ στις γραμμές 1,2 είναι διανυσματική αφαίρεση των συντεταγμένων των σημείων. Ο τελεστής `max(0, v)` είναι επίσης διανυσματικός και αντικαθιστά όλες τις αρνητικές τιμές ενός διανύσματος v με 0. Ο τελεστής $\cdot$ είναι το εσωτερικό γινόμενο διανυσμάτων.

Algorithm 1: Υπολογισμός της Ευκλείδειας Απόστασης δύο AABB

Input: Two AABBs P, Q
Output: Euclidean distance of P and Q

`AABB_distance` (P, Q)
1 $w \leftarrow \max(0, P.S - Q.T)$
2 $v \leftarrow \max(0, Q.S - P.T)$
3 **return** $\sqrt{v \cdot v + w \cdot w}$

Κατασκευή AABB Από Τρίγωνο

Με τον αλγόριθμο 2 κατασκευάζουμε ένα AABB που εξ' ολοκλήρου περικλείει ένα τρίγωνο. Η ρουτίνα `vertices(T)` επιστρέφει τις τρεις κορυφές του τριγώνου T. Οι τελεστές `min(), max()` είναι διανυσματικοί, δέχονται ως όρισμα μια λίστα από διανύσματα και επιστρέφουν ένα διάνυσμα με τις ελάχιστες/μέγιστες συντεταγμένες ανά άξονα.

Algorithm 2: Κατασκευή AABB από Τρίγωνο

Input: A triangle T
Output: An AABB enclosing T

`AABB` (T)
1 $S \leftarrow \min(\text{vertices}(T))$
2 $T \leftarrow \max(\text{vertices}(T))$
3 **return** $\{S, T\}$

Συνένωση Δύο AABB

Ο αλγόριθμος 3 κατασκευάζει ένα AABB περικλείει εξ' ολοκλήρου δύο άλλα AABB που δέχεται ως ορίσματα. Αυτή η πράξη είναι χρήσιμη γιατί αν έχουμε ήδη βρει τα AABB για δύο ομάδες αντικειμένων ξεχωριστά, μπορούμε σε σταθερό χρόνο $O(1)$ να κατασκευάσουμε το AABB που περικλείει όλα τα αντικείμενα. Οι τελεστές `min(),`

`max()` είναι ίδιοι με πριν.

Algorithm 3: Κατασκευή AABB από δύο άλλα AABB

Input: Two AABBs P, Q
Output: An AABB enclosing P and Q

```
combine_AABB (P, Q)
1 S ← min(P.S, Q.S)
2 T ← max(P.T, Q.T)
3 return {S, T}
```

4.2 Αλγόριθμοι Εξαντλητικής Αναζήτησης

Έστω δύο αντικείμενα του τρισδιάστατου χώρου που αναπαρίστανται από τριγωνικά πλέγματα. Από τον ορισμό της Ευκλείδειας απόστασης δύο τριγωνικών πλεγμάτων που δόθηκε στην ενότητα 1.2 και τη χρήση μιας ρουτίνας `triangle_distance(p, q)`, προκύπτει ο τετριμμένος αλγόριθμος 4. Η πολυπλοκότητα του αλγορίθμου είναι $O(N * M)$, όπου N και M είναι το πλήθος των τριγώνων των δύο αντικειμένων.

Algorithm 4: Απόσταση Τριγωνικών Πλεγμάτων με Πλήρη Αναζήτηση

Input: Two triangle meshes P, Q
Output: Euclidean distance of P and Q

```
triangle_mesh_distance (P, Q)
1 T_P ← triangles (P)
2 T_Q ← triangles (Q)
3 distance ← inf
4 foreach p ∈ T_P do
5 |   foreach q ∈ T_Q do
6 |   |   distance ← min(distance, triangle_distance(p, q))
7 |   end
8 end
9 return distance
```

Σύμφωνα με τη μελέτη που έγινε στην ενότητα 2.2 μπορούμε να επιταχύνουμε τον χρόνο εκτέλεσης του αλγορίθμου αν χρησιμοποιήσουμε οριοθετικούς όγκους. Συγκεκριμένα, θα χρησιμοποιήσουμε Οριοθετικά Πλαίσια Ευθυγραμμισμένα με τους Άξονες (AABB). Όπως φαίνεται από τον πίνακα της ενότητας 5.1, ο υπολογισμός της ελάχιστης απόστασης *AABB - AABB* είναι πολύ πιο ¨οικονομικός¨ υπολογιστικά σε σχέση με τον υπολογισμό της απόστασης *τρίγωνο - τρίγωνο*. Μπορούμε να εκμεταλλευτούμε αυτό το γεγονός και σε συνδυασμό με το παρακάτω λήμμα να σχεδιάσουμε έναν ταχύτερο αλγόριθμο.

Λήμμα 1. *Έστω δύο σύνολα S_A και S_B που αποτελούνται από χωρικά αντικείμενα. Έστω οι οριοθετικοί όγκοι BV_A και BV_B που εξ' ολοκλήρου περικλείουν τα αντικείμενα των σύνολων S_A και S_B, αντίστοιχα. Τότε:*

$$distance(BV_A, BV_B) \leq distance(a, b), \forall a \in S_A, \forall b \in S_B$$

Δηλαδή, η Ευκλείδεια απόσταση οποιουδήποτε ζεύγους αντικειμένων από τα δύο σύνολα θα είναι μεγαλύτερη ή ίση με την Ευκλείδεια απόσταση των οριοθετικών όγκων των συνόλων.

Απόδειξη. Συμβολίζουμε με BV_A το σύνολο των σημείων του οριοθετικού όγκου, BV_A. Όμοια και για το σύνολο BV_B.

Επίσης, έχουμε $S_A = \bigcup_{a \in S_A} a$ και $S_B = \bigcup_{b \in S_B} b$, όπου με a, b συμβολίζουμε το σύνολο των σημείων των χωρικών αντικειμένων a και b.

Αφού οι οριοθετικοί όγκοι περικλείουν τα σύνολα των αντικειμένων S_A και S_B ισχύει επίσης ότι $S_A \subseteq BV_A$ και $S_B \subseteq BV_B$.

Έστω ότι υπάρχουν αντικείμενα $a \in S_A$, $b \in S_B$ με απόσταση μικρότερη από αυτή των οριοθετικών όγκων. Τότε, υπάρχουν σημεία $p \in a \subseteq S_A \subseteq BV_A$ και $q \in b \subseteq S_B \subseteq BV_A$ ώστε

$$\|p - q\| < distance(BV_A, BV_B)$$

Από τον ορισμό της Ευκλείδειας απόστασης δύο οριοθετικών όγκων καταλήγουμε σε άτοπο. □

Προκύπτει, λοιπόν, ο αλγόριθμος 5. Αρχικά, κάθε τρίγωνο περικλείεται από ένα AABB. Σε κάθε βήμα του αλγορίθμου είναι γνωστή η μικρότερη απόσταση που έχει υπολογιστεί μέχρι εκείνη τη στιγμή. Έτσι, για κάθε ζεύγος τριγώνων, πρώτα γίνεται ο γρήγορος υπολογισμός απόστασης AABB-AABB και μόνο όταν αυτή η απόσταση είναι μικρότερη από την τρέχουσα απόσταση γίνεται ο έλεγχος τρίγωνο-τρίγωνο. Η υπολογιστική πολυπλοκότητα παραμένει $O(N * M)$, όπως και πριν. Σημειώνεται, ότι ο χρόνος εκτέλεσης επηρεάζεται από τη σειρά που θα γίνουν οι πράξεις. Δηλαδή αν η μικρότερη απόσταση βρεθεί από πολύ νωρίς, τότε θα γίνουν πολύ λίγοι υπολογισμοί τρίγωνο-τρίγωνο, ενώ το αντίθετο θα συμβεί αν η μικρότερη απόσταση βρεθεί πιο αργά.

Algorithm 5: Απόσταση Τριγωνικών Πλεγμάτων με Πλήρη Αναζήτηση και AABB

Input: Two triangle meshes P, Q
Output: Euclidean distance of P and Q

```
triangle_mesh_distance (P, Q)
 1 T_P ← triangles (P)
 2 T_Q ← triangles (Q)
 3 precalculate AABBs of T_P
 4 precalculate AABBs of T_Q
 5 distance ← inf
 6 foreach p ∈ T_P do
 7     foreach q ∈ T_Q do
 8         if AABB_distance (AABB (p), AABB (q)) < distance then
 9             distance ← min(distance, triangle_distance(p, q))
10         end
11     end
12 end
13 return distance
```

Παραπάνω, θα μπορούσαμε να χρησιμοποιήσουμε οποιονδήποτε τύπο οριοθετικού όγκου. Οι λόγοι για τους οποίους γίνεται η επιλογή των AABB είναι:

1. Η Ιεραρχία Οριοθετικών Όγκων που σχεδιάζουμε και προτείνουμε στην ενότητα 4.4 χρησιμοποιεί επίσης AABB. Επομένως, μπορεί να υπάρξει ένα μέτρο σύγκρισης μεταξύ των αλγορίθμων.
2. Η κατασκευή ενός AABB που περικλείει εξ' ολοκλήρου ένα τρίγωνο και έχει ελάχιστο όγκο είναι εύκολη (αλγόριθμος 2).
3. Ο υπολογισμός της Ευκλείδειας απόστασης δύο AABB (αλγόριθμος 1) είναι εύκολος (αλγόριθμος 1).
4. Η συνένωση δύο AABB για την κατασκευή ενός μεγαλύτερου που περικλείει και τα δύο είναι επίσης εύκολη (αλγόριθμος 3). Η συνένωση είναι χρήσιμη πράξη για την κατασκευή της ιεραρχίας.

4.3 Ορισμός Μετρικής Κόστους Αναζήτησης

Σε αυτήν την ενότητα, ορίζουμε μια μετρική κόστους αναζήτησης ώστε να μπορούμε να συγκρίνουμε την επίδοση των αλγορίθμων που μελετώνται. Η ίδια μετρική χρησιμοποιείται και στα [21], [31] για τη σύγκριση διάφορων ιεραρχικών δομών δεδομένων. Δοθέντος δύο αντικειμένων, το συνολικό κόστος για τον υπολογισμό της μεταξύ τους Ευκλείδειας απόστασης μπορεί να δοθεί από την ακόλουθη εξίσωση κόστους:

$$T = N_v \times C_v + N_p \times C_p$$

Όπου

T το συνολικό κόστος για τον υπολογισμό της απόστασης,

N_v το πλήθος των υπολογισμών απόστασης μεταξύ οριοθετικών όγκων,

C_v το κόστος υπολογισμού απόστασης μεταξύ οριοθετικών όγκων,

N_p το πλήθος των υπολογισμών απόστασης μεταξύ των primitives [1]

C_p το κόστος υπολογισμού απόστασης μεταξύ των primitives

Η επιλογή του τύπου οριοθετικού όγκου που θα χρησιμοποιηθεί, κατά τη σχεδίαση μιας ιεραρχίας, διέπεται από δύο αντικρουόμενους περιορισμούς[2]:

1. Θα πρέπει να ακολουθεί το μοντέλο όσο πιο στενά γίνεται (tight-fit) για να μειωθούν οι τιμές N_v, N_p.
2. Ο υπολογισμός της απόστασης μεταξύ οριοθετικών όγκων να είναι όσο το δυνατόν ταχύτερος (για να μειωθεί το κόστος C_v)

Για αυτή την εργασία οι οριοθετικοί όγκοι είναι AABB, ενώ τα primitives είναι τρίγωνα. Τα κόστη C_v, C_p δίνονται στους πίνακες της ενότητας 5.1. Οι τιμές των N_v, N_p μετρώνται

[1] Geometric primitives είναι κάποια απλά γεωμετρικά σχήματα που μπορεί να χειριστεί ένα σύστημα. Συνήθως είναι σημεία, πολύγωνα, πολύεδρα κλπ.

[2] Στις ενότητες 2.2, 2.3, γίνεται λεπτομερέστερη περιγραφή για αυτό το trade-off

πειραματικά ανάλογα με τα δεδομένα εισόδου (γεωμετρία, μέγεθος, προσανατολισμό, ποιότητα των πλεγμάτων).

Σημείωση: στην παραπάνω μετρική κόστους δεν προσμετράται το κόστος προεπεξεργασίας των δεδομένων (πχ για την κατασκευή μιας ιεραρχίας). Το συνολικό κόστος (προεπεξεργασία και αναζήτηση) ¨συμπεριλαμβάνεται¨ στη μέτρηση των χρόνων εκτέλεσης των πειραμάτων.

4.4 Σχεδιασμός μιας Δομής BVH, το sKD-Tree

Για τον υπολογισμό της απόστασης των δύο πλεγμάτων αποδοτικά, αρχικά κάνουμε μια προ-επεξεργασία των δεδομένων οργανώνοντας τα σε μια Ιεραρχία Οριοθετικών Όγκων (BVH). Έπειτα σχεδιάζουμε αλγορίθμους, που εκμεταλλευόμενοι αυτή την ιεραρχική οργάνωση, παρουσιάζουν καλύτερη υπολογιστική πολυπλοκότητα, μέσης περίπτωσης, σε σχέση με τους αλγορίθμους εξαντλητικής αναζήτησης. Επειδή ο τρόπος οργάνωσης των χωρικών δεδομένων είναι παρόμοιος με τον τρόπο κατασκευής του KD-Tree, η δομή που προτείνουμε ονομάζεται spatial KD-Tree ή sKD-Tree.

4.4.1 Κατασκευή του sKD-Tree

Το sKD-Tree είναι ένα σχεδόν πλήρες, ισορροπημένο, δυαδικό δέντρο που χρησιμοποιεί AABB για την οργάνωση των αντικειμένων. Η ιεραρχία κατασκευάζεται με βάση τα στοιχεία (elements/primitives) ενός αντικειμένου. Στην περίπτωση μας τα στοιχεία είναι τρίγωνα. Κάθε κόμβος του δέντρου είναι ένα AABB, ενώ το δέντρο έχει τις παρακάτω ιδιότητες:

- Η ένωση των AABB των φύλλων του δέντρου περικλείει εξ' ολοκλήρου την επιφάνεια του αντικειμένου.
- Το AABB ενός κόμβου περικλείει εξ' ολοκλήρου τα AABB όλων των απογόνων του.

Η ιδέα για την κατασκευή της ιεραρχίας είναι η ακόλουθη: Αρχικά, τα φύλλα του δέντρου σχεδόν προσεγγίζουν την επιφάνεια του αντικειμένου και οι εσωτερικοί κόμβοι αναπαριστούν προσεγγίσεις των απογόνων φύλλων τους. Έτσι, όσο πιο κοντά στη ρίζα βρίσκεται ένας κόμβος, τόσο πιο αδρή είναι η προσέγγιση της επιφάνειας. Η οργάνωση αυτή είναι χρήσιμη, γιατί με βάση το AABB κάποιου εσωτερικού κόμβου, μπορεί να βρεθεί το κάτω φράγμα της απόστασης προς οποιοδήποτε στοιχείο της επιφάνειας.

Το πρώτο βήμα για την κατασκευή του sKD-Tree είναι η επικάλυψη των τριγώνων του πλέγματος με AABB. Αυτά θα αποτελούν τα φύλλα του δέντρου. Έπειτα ακολουθούμε μια στρατηγική από πάνω προς τα κάτω ξεκινώντας από τη ρίζα του δέντρου. Σε κάθε βήμα, τα n τρίγωνα του κόμβου χωρίζονται σε δύο υποσύνολα μεγέθους $\lceil n/2 \rceil$ και $\lfloor n/2 \rfloor$ αντίστοιχα. Για κάθε ένα από τα σύνολα, καλείται η αναδρομική ρουτίνα που δημιουργεί τα δύο υποδέντρα. Στη συνέχεια, τα AABB των δύο παιδιών του κόμβου ενώνονται και δημιουργείται το AABB του τρέχοντος κόμβου. Η αναδρομή σταματά όταν το σύνολο αποτελείται από ένα μόνο τρίγωνο, του οποίου το AABB είχαμε υπολογίσει στην αρχή.

Για τον διαχωρισμό των τριγώνων σε δύο υποσύνολα, αρχικά, επιλέγεται ένα *αντιπροσωπευτικό σημείο* για κάθε τρίγωνο. Στην υλοποίηση μας, το αντιπροσωπευτικό σημείο για

το κάθε τρίγωνο επιλέγεται να είναι το κέντρου του AABB του[3]. Ο διαχωρισμός γίνεται είτε με βάση τον άξονα x, είτε με τον y, είτε με τον z. Ο άξονας αυτός εναλλάσσεται κυκλικά σε κάθε επίπεδο του δέντρου. Έστω ότι ο άξονας διαχωρισμού είναι ο $k \in \{x, y, z\}$. Συμβολίζουμε με T_i το i-στο τρίγωνο (από τα n) κάποιου κόμβου, ενώ με RP_i συμβολίζουμε το αντιπροσωπευτικό του σημείο. Αν m είναι η διάμεσος των τιμών RP_{ik} τότε τα δύο σύνολα που προκύπτουν είναι τα $L = \{T_i \mid RP_{ik} \leq m\}$ και $R = \{T_i \mid RP_{ik} \geq m\}$ τέτοια ώστε $|L| = \lceil n/2 \rceil$ και $|R| = \lfloor n/2 \rfloor$. Για τον διαχωρισμό των τριγώνων χρησιμοποιείται ο αλγόριθμος `QUICKSELECT` [25].

Τα παραπάνω συνοψίζονται στους αλγορίθμους 6 και 7. Για να υπολογίσουμε την υπολογιστική πολυπλοκότητα του αλγορίθμου 7 θα χρησιμοποιήσουμε το *Θεώρημα του Κυρίαρχου Όρου (Master Theorem) [6]*: Για κάθε αναδρομική κλήση ο αριθμός των πράξεων είναι $T(n) = 2 * T(n/2) + O(n) + O(1)$. Επομένως, η πολυπλοκότητα του αλγορίθμου αυτού είναι $O(n*logn)$, όπου n είναι ο αριθμός των τριγώνων του πλέγματος. Συνεπώς, και ο αλγόριθμος 6 έχει ακριβώς την ίδια υπολογιστική πολυπλοκότητα.

Algorithm 6: Κατασκευή του sKD-Tree

Input: A triangle mesh M
Output: The root of the sKD-Tree

```
tree_build (M)
1  T ← triangles(M)
2  precalculate AABBs of T
3  precalculate RPs of T
4  root is the root of the tree
5  tree_build_recursive(root, T, x)
6  return root
```

Algorithm 7: Αναδρομική Κατασκευή του sKD-Tree

Input: A tree node v, a set of triangles T, a splitting axis *axis*
Output: Fills the node's entries, thus building the tree

```
tree_build_recursive (v, T, axis)
1  if |T| = 1 then
2  |  v.aabb ← AABB(T_1)
3  |  v.triangle ← T_1
4  |  return
5  end
6  L, R ← split_triangles(T, axis)
7  axis ← next_axis(axis)
8  tree_build_recursive(v.left, L, axis)
9  tree_build_recursive(v.right, R, axis)
10 v.aabb ← combine_AABB(v.left.aabb, v.right.aabb)
```

Στην εικόνα 4.2 φαίνεται ο τρόπος με τον οποίο κατασκευάζεται ένα sKD-Tree.

[3]Επιλέγουμε σημείο με βάση το AABB και όχι με βάση το τρίγωνο για να είναι εύκολη η επέκταση του αλγορίθμου και σε στοιχεία άλλου τύπου

4.4.2 Ερωτήματα Κοντινότερου Γείτονα στο sKD-Tree

Δοθέντος ενός συνόλου T που αποτελείται από τρίγωνα και ενός τυχαίου τριγώνου q (*query-triangle*) αναζητούμε το τρίγωνο $t \in T$ που απέχει τη μικρότερη Ευκλείδεια απόσταση από το q. Το πρόβλημα αυτό αποτελεί γενίκευση του kNN για χωρικά αντικείμενα, που στην περίπτωση μας είναι τρίγωνα που σχηματίζουν κάποιο πλέγμα. Προφανώς, όπως αναφέρθηκε παραπάνω, η μεθοδολογία που προτείνουμε μπορεί να γενικευτεί και για άλλα στοιχεία πέραν των τριγώνων.

Η διαδικασία διάσχισης του sKD-Tree που προτείνουμε για την απάντηση ερωτημάτων κοντινότερου γείτονα είναι μια κατευθυνόμενη αναζήτηση κατά βάθος (DFS). Αρχικά, επιλέγεται ένα μονοπάτι από τη ρίζα έως κάποιο φύλλο, το οποίο είναι η πρώτη εκτίμηση κοντινότερου γείτονα. Για τον γείτονα αυτόν υπολογίζεται μια απόσταση. Αυτή η απόσταση ενημερώνεται κάθε φορά που εντοπίζεται όλο και κοντινότερος γείτονας. Κατά την οπισθοδρόμηση (backtracking) η διαδικασία χρησιμοποιώντας την τρέχουσα εκτίμηση απόστασης κοντινότερου γείτονα απορρίπτει μονοπάτια του δέντρου βάση του λήμματος 1.

Πιο αναλυτικά, έστω το τυχαίο τρίγωνο q του ερωτήματος. Αρχικά κατασκευάζεται το AABB του q και ορίζεται η απόσταση *dist* του κοντινότερου γείτονα (έως τώρα) να είναι άπειρη. Έπειτα καλείται η αναδρομική διαδικασία αναζήτησης. Έστω ότι η διαδικασία επισκέπτεται κάποιον κόμβο v του δέντρου. Αν ο v είναι εσωτερικός κόμβος, ο επόμενος κόμβος που θα επισκεφθεί η διαδικασία επιλέγεται να είναι αυτός που απέχει τη μικρότερη απόσταση από το AABB του q. Δηλαδή, υπολογίζονται οι αποστάσεις AABB-AABB των δύο παιδιών του κόμβου ως προς το AABB του q και επιλέγεται ο κοντινότερος. Αν ο κόμβος v είναι φύλλο, τότε υπολογίζεται η απόσταση τρίγωνο-τρίγωνο και ενημερώνεται η μεταβλητή *dist* όταν η απόσταση που υπολογίστηκε είναι μικρότερη. Κατά την οπισθοδρόμηση της αναδρομής, δηλαδή στη δεύτερη επίσκεψη του κόμβου, θα πρέπει να γίνει έλεγχος αν μπορεί να απορριφθεί το υποδέντρο του κόμβου που δεν επιλέχθηκε προηγουμένως.Τα παραπάνω περιγράφονται στους αλγορίθμους 8 και 9.

Algorithm 8: Υπολογισμός Απόστασης Κοντινότερου Γείτονα

Input: An sKD-Tree *tree*, a query-triangle q, a search radius r (optional)
Output: The distance of the nearest neighbor

```
nearest_search (tree, q, r = ∞)
1 precalculate the AABB of q
2 distance ← r
3 nearest_search_recursive (tree.root, q, distance)
4 return distance
```

Επεξήγηση του αλγορίθμου 9:

γραμμές 1-4 Είναι η περίπτωση όπου η αναζήτηση έχει φτάσει σε φύλλο του δέντρου, οπότε γίνεται έλεγχος απόστασης του q από το τρίγωνο του φύλλου.

γραμμές 5-6 Γίνεται υπολογισμός της απόστασης του AABB του q με τα AABB των παιδιών του κόμβου. Αυτές οι αποστάσεις χρησιμοποιούνται για να καθοδηγήσουν την αναζήτηση.

γραμμές 7,15 Γίνεται καθοδήγηση της σειράς με την οποία θα γίνει η αναζήτηση.

Algorithm 9: Αναδρομικός Υπολογισμός Απόστασης Κοντινότερου Γείτονα

Input: An sKD-Tree node v, a query-triangle q, NN distance so far *distance*

Output: Changes *distance* to be the distance of the nearest neighbor

```
nearest_search_recursive (v, q, distance)
 1 if v is leaf then
 2 |   distance ← min(distance, triangle_distance(q, v.triangle))
 3 |   return
 4 end
 5 aabb_dist_l ← AABB_distance(AABB(q), v.left.aabb)
 6 aabb_dist_r ← AABB_distance(AABB(q), v.right.aabb)
 7 if aabb_dist_l < aabb_dist_r then
 8 |   if aabb_dist_l < distance then
 9 |   |   nearest_search_recursive (v.left, q, distance)
10 |   end
11 |   if aabb_dist_r < distance then
12 |   |   nearest_search_recursive (v.right, q, distance)
13 |   end
14 end
15 else
16 |   if aabb_dist_r < distance then
17 |   |   nearest_search_recursive (v.right, q, distance)
18 |   end
19 |   if aabb_dist_l < distance then
20 |   |   nearest_search_recursive (v.left, q, distance)
21 |   end
22 end
23 return
```

γραμμές 8-13 Πρώτα γίνεται αναζήτηση στο αριστερό παιδί και έπειτα στο δεξί.

γραμμές 16-21 Πρώτα γίνεται αναζήτηση στο δεξί παιδί και έπειτα στο αριστερό.

γραμμές 11,19 Έλεγχος για το αν είναι απαραίτητη η επίσκεψη στο παιδί που δεν επιλέχθηκε προηγουμένως.

γραμμές 8,16 Πρόωρο τερματισμό της αναζήτησης (early exit). Αυτό είναι χρήσιμο όταν ένας κόμβος δεν μπορεί να απορριφθεί κατά την αναζήτηση, όμως μπορούν να απορριφθούν τα παιδιά του. Έτσι η αναζήτηση δε χρειάζεται να φτάσει μέχρι το φύλλο του υποδέντρου.

Σημειώνουμε ότι δεν είναι εφικτό να υπολογίσουμε την πολυπλοκότητα του αλγορίθμου αναζήτησης ούτε να εγγυηθούμε καλή επίδοση χειρότερης περίπτωσης. Αυτό οφείλεται στο ότι κατά την αναζήτηση μπορεί να γίνει επίσκεψη και στα δύο υποδέντρα ενός κόμβου (ή μέρος τους).

4.5 Αλγόριθμοι Αναζήτησης με sKD-Tree

Έχοντας πλέον οργανώσει τα δεδομένα σε μια ιεραρχία όπως το sKD-Tree μπορούμε να σχεδιάσουμε αλγορίθμους που βασίζονται στις ρουτίνες που περιγράψαμε προηγουμένως.

Ο πιο απλός αλγόριθμος κατασκευάζει ένα sKD-Tree για τα τρίγωνα του ενός αντικειμένου και έπειτα για κάθε τρίγωνο του άλλου αντικειμένου εκτελεί αναζήτηση κοντινότερου γείτονα. Από τις αποστάσεις που θα υπολογιστούν, επιλέγεται η μικρότερη. Μπορούμε να επιταχύνουμε τον χρόνο εκτέλεσης αυτού του αλγορίθμου εάν σε κάθε αναζήτηση κοντινότερου γείτονα ορίσουμε την ακτίνα αναζήτησης ίση με την τρέχουσα μικρότερη απόσταση που έχει υπολογιστεί. Με αυτό τον τρόπο, μπορεί να γίνει κλάδεμα του δέντρου από πολύ νωρίς, ακόμη και πριν η αναζήτηση φτάσει σε κάποιο φύλλο του δέντρου. Η διαδικασία συνοψίζεται στον αλγόριθμο 10.

Algorithm 10: Απόσταση Τριγωνικών Πλεγμάτων με NN Queries σε sKD-Tree

Input: Two triangle meshes P, Q
Output: Euclidean distance of P and Q

```
triangle_mesh_distance (P, Q)
 1 T_P ← triangles (P)
 2 tree ← tree_build(Q)
 3 distance ← inf
 4 foreach p ∈ T_Q do
 5 |   result ← nearest_search(tree, p, distance)
 6 |   if result < distance then
 7 |   |   distance ← result
 8 |   end
 9 end
10 return distance
```

Ο χρόνος εκτέλεσης του αλγορίθμου 10 εξαρτάται από τη σειρά που θα γίνουν οι πράξεις. Δηλαδή, αν βρεθεί από νωρίς ο κοντινότερος γείτονας, η απάντηση συγκλίνει πολύ γρήγορα και τα υπόλοιπα ερωτήματα στο δέντρο απαιτούν λιγότερο χρόνο, λόγω κλαδέματος. Συνεπώς, ο παραπάνω αλγόριθμος περιλαμβάνει έναν παράγοντα τύχης.

Ο δεύτερος αλγόριθμος που προτείνουμε, χρησιμοποιεί δύο ιεραρχίες sKD-Tree, μία για κάθε τριγωνικό πλέγμα. Με αυτό τον τρόπο καθορίζουμε τη σειρά των πράξεων έτσι ώστε να επιτευχθεί η σύγκλιση γρηγορότερα. Το επιπλέον κόστος αυτής της προσέγγισης είναι η κατασκευή ενός δεύτερου δέντρου. Πρώτα διασχίζεται η μια ιεραρχία από τη ρίζα έως κάποιο φύλλο, και έπειτα εκτελείται ένα ερώτημα κοντινότερου γείτονα στην άλλη ιεραρχία (για το τρίγωνο του φύλλου). Κατά την οπισθοδρόμηση γίνεται εκ νέου χρήση του λήμματος 1 για το κλάδεμα της πρώτης ιεραρχίας. Τέλος, ο τρόπος διάσχισης της πρώτης ιεραρχίας μοιάζει με τον τρόπο διάσχισης που περιγράφηκε στον αλγόριθμο 9. Έστω ότι η διαδικασία επισκέπτεται κάποιον κόμβο v του πρώτου δέντρου. Αν ο v είναι εσωτερικός κόμβος, ο επόμενος κόμβος που θα επισκεφθεί η διαδικασία επιλέγεται να είναι αυτός που απέχει τη μικρότερη απόσταση από το AABB της ρίζας του άλλου δέντρου. Ενώ αν ο v είναι φύλλο, τότε εκτελείται ένα ερώτημα κοντινότερου γείτονα στην άλλη ιεραρχία ορίζοντας ως ακτίνα αναζήτησης την τρέχουσα μικρότερη

απόσταση. Όλα τα παραπάνω περιγράφονται με ψευδοκώδικα στους αλγορίθμους 11 και 12.

Επεξήγηση του αλγορίθμου 12:

γραμμές 1-5 Η αναζήτηση έχει επισκεφθεί ένα φύλλο του πρώτου δέντρου, οπότε εκτελείται αναζήτηση κοντινότερου γείτονα στο δεύτερο δέντρο, με ακτίνα αναζήτησης την τρέχουσα μικρότερη.

γραμμές 6-7 Υπολογίζονται οι αποστάσεις των AABB των παιδιών του v με το AABB της ρίζας του άλλου δέντρου.

γραμμές 8-23 Δίνεται συγκεκριμένη κατεύθυνση στην αναζήτηση και γίνεται χρήση του λήμματος 1 για το κλάδεμα του πρώτου δέντρου.

Algorithm 11: Απόσταση Τριγωνικών Πλεγμάτων με Δύο sKD-Tree

Input: Two triangle meshes P, Q
Output: Euclidean distance of P and Q

```
triangle_mesh_distance (P, Q)
1 tree1 ← tree_build (P)
2 tree2 ← tree_build (Q)
3 distance ← inf
4 nearest_search_two_trees (tree1.root, tree2, distance)
5 return distance
```

4.6 Λεπτομέρειες Υλοποίησης των Αλγορίθμων

Για την υλοποίηση των αλγορίθμων αναζήτησης που περιγράφηκαν παραπάνω χρησιμοποιήθηκαν οι ακόλουθες παρατηρήσεις και τεχνικές με στόχο την επίτευξη υψηλής επίδοσης:

Μνήμη. Το sKD-Tree είναι ένα στατικό, σχεδόν πλήρες και ισορροπημένο δυαδικό δέντρο. Επομένως, μπορεί να αποθηκευτεί σε συνεχόμενες θέσεις μνήμης (πίνακα). Αυτό είναι χρήσιμο, καθώς αποφεύγονται πολλαπλές δεσμεύσεις μνήμης (allocations) κατά την κατασκευή του δέντρου. Το μόνο που απαιτείται είναι μία μόνο δέσμευση μνήμης. Επιπλέον, ένα sKD-Tree που κατασκευάζεται από n στοιχεία/τρίγωνα απαιτεί $2n - 1$ κόμβους. Η ίδια σχέση ισχύει, προφανώς, και για οποιοδήποτε υποδέντρο του sKD-Tree και θα χρησιμοποιηθεί παρακάτω για τον προσδιορισμό των δεικτών των παιδιών ενός κόμβου.

Απαριθμούμε τους κόμβους του δέντρου σύμφωνα με τη διάσχιση Euler (pre-order traversal) και τους τοποθετούμε στις αντίστοιχες θέσεις του πίνακα. Έστω ότι εξετάζουμε τον κόμβο με δείκτη v ο οποίος περικλείει n_v τρίγωνα. Τότε το αριστερό παιδί του v θα έχει δείκτη $v + 1$. Το αριστερό παιδί, επίσης, περικλείει $\lceil n_v/2 \rceil$ τρίγωνα άρα συνολικά θα υπάρχουν $2 * \lceil n_v/2 \rceil - 1$ κόμβοι στο αριστερό υποδέντρο. Επομένως, ο δείκτης του δεξιού παιδιού του v θα έχει δείκτη $v + 2 * \lceil n_v/2 \rceil$.

Στην υλοποίηση μας χρησιμοποιούμε έναν πίνακα *indices* ο οποίος δεικτοδοτεί κάθε τρίγωνο του sKD-Tree. Αρχικά, ο πίνακας αυτός περιέχει τις τιμές από 1 έως n στη

Algorithm 12: Απόσταση Τριγωνικών Πλεγμάτων με Δύο sKD-Tree

Input: A node v of the first tree, the second tree *other_tree*, NN *distance* so far
Output: Changes *distance* to be the distance of the nearest triangles

```
nearest_search_two_trees (v, other_tree, distance)
 1 if v is leaf then
 2 |   result ← nearest_search (other_tree, v.triangle, distance)
 3 |   distance ← min(distance, result)
 4 |   return
 5 end
 6 aabb_dist_l ← AABB_distance(other_tree.root.aabb, v.left.aabb)
 7 aabb_dist_r ← AABB_distance(other_tree.root.aabb, v.right.aabb)
 8 if aabb_dist_l < aabb_dist_r then
 9 |   if aabb_dist_l < distance then
10 |   |   nearest_search_recursive (v.left, other_tree, distance)
11 |   end
12 |   if aabb_dist_r < distance then
13 |   |   nearest_search_recursive (v.right, other_tree, distance)
14 |   end
15 end
16 else
17 |   if aabb_dist_r < distance then
18 |   |   nearest_search_recursive (v.right, other_tree, distance)
19 |   end
20 |   if aabb_dist_l < distance then
21 |   |   nearest_search_recursive (v.left, other_tree, distance)
22 |   end
23 end
24 return
```

σειρά. Για κάθε κόμβο του δέντρου αναθέτουμε μια συνεχόμενη περιοχή του πίνακα *indices*, με τη ρίζα του δέντρου να είναι υπεύθυνη για το εύρος $[1..n]$. Έστω ότι ένας κόμβος είναι υπεύθυνος για το τμήμα $[start...end]$ τότε υπολογίζεται το $mid = \lfloor (start + end)/2 \rfloor$ οπότε το αριστερό παιδί είναι υπεύθυνο για το τμήμα $[start...mid]$, ενώ το δεξί παιδί για το τμήμα $[mid + 1...end]$. Ο πίνακας 4.1 συνοψίζει όλα τα παραπάνω.

Αυτό που μένει είναι να αναθέσουμε τους σωστούς δείκτες τριγώνων στις αντίστοιχες περιοχές του αριστερού και του δεξιού παιδιού. Η ανάθεση επιτυγχάνεται με τον αλγόριθμο `QUICKSELECT` που αναδιατάσσει τους δείκτες μιας περιοχής σύμφωνα με κάποιο κριτήριο. Το κριτήριο είναι η διάμεσος των συντεταγμένων, όπως περιγράφηκε

Κόμβος	Εύρος Τριγώνων	Δείκτης
Τυχαίος κόμβος	$[start...end]$	v
Αριστερό παιδί	$[start...mid]$	$v + 1$
Δεξί παιδί	$[mid + 1...end]$	$v + 2 * (mid - start + 1)$

Πίνακας 4.1

στο 4.4. Έτσι καταφέρνουμε να χωρίσουμε ένα σύνολο τριγώνων σε δύο υποσύνολα χωρίς μετακίνηση, δέσμευση μνήμης ή αντιγραφή δεδομένων.

Τέλος, για την αποθήκευση του sKD-Tree απαιτείται:

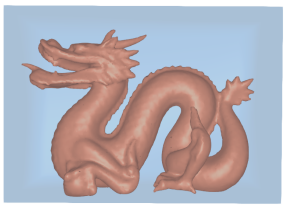
- Ένας πίνακας ακεραίων *indices* μεγέθους n.
- Ένας πίνακας *data* με τα τρίγωνα, μεγέθους n. Ο πίνακας αυτός, ανάλογα με την υλοποίηση, μπορεί είτε να περιέχει δείκτες προς τα τρίγωνα, είτε να περιέχει τα ίδια τα τρίγωνα.
- Ένας πίνακας *data_aabb* με τα AABB των τριγώνων
- Ένας πίνακας *nodes* μεγέθους $2 * n - 1$ για τα δεδομένα των κόμβων του δέντρου. Στην υλοποίηση μας τα μόνα δεδομένα που αποθηκεύονται ανά κόμβο είναι ένα AABB, αυτό του κόμβου.

Χρήση Πολλαπλών Νημάτων. Στις υλοποιήσεις μας χρησιμοποιούμε πολλαπλά νήματα (multithreading) για την παράλληλη κατασκευή του sKD-Tree και την παράλληλη εκτέλεση ερωτημάτων.

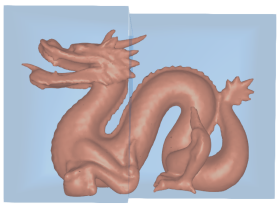
Στον αλγόριθμο 7 η κατασκευή του αριστερού και του δεξιού παιδιού ενός κόμβου είναι δύο ανεξάρτητες εργασίες. Επομένως, καλούμε ένα thread για την κατασκευή του αριστερού παιδιού ενώ η κατασκευή του δεξιού παιδιού συνεχίζεται στο τρέχουν thread. Πριν τη συνένωση των AABB των δύο παιδιών θα πρέπει το thread που δημιουργήθηκε να ολοκληρώσει την εργασία του. Επομένως, η ροή του τρέχοντος thread σταματά μέχρι την ολοκλήρωση της εργασίας του άλλου thread. Επιπλέον, προσθέσαμε μια συνθήκη για την αποφυγή της κλήσης πολλών εργασιών ανεξέλεγκτα. Έστω *nthreads* είναι το πλήθος των διαθέσιμων threads του συστήματος, η συνθήκη για παράλληλη κατασκευή είναι $nthreads \geq 2^{depth-1}$, όπου *depth* είναι το βάθος του κόμβου που κατασκευάζεται. Σε διαφορετική περίπτωση, η κατασκευή γίνεται σειριακά.

Για τη μετατροπή του αλγορίθμου 10 σε παράλληλο, χωρίζουμε το πλήθος των ερωτημάτων ισόποσα σε κάθε νήμα (thread). Όμως επειδή η μεταβλητή `distance` συνεχώς αλλάζει όταν εντοπίζονται κοντινότεροι γείτονες, για να αποφύγουμε τη χρήση *lock*, το κάθε νήμα εκτελεί τις γραμμές 3-9 του αλγορίθμου με μια τοπική μεταβλητή `local_distance`. Στο τέλος, επιστρέφεται η μικρότερη από αυτές.

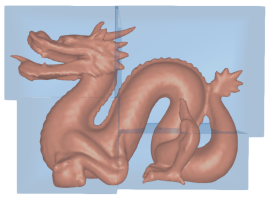
Απλοποίηση Πράξεων Σε όλους τους παραπάνω αλγορίθμους μπορούμε να αντικαταστήσουμε τις αποστάσεις από το τετράγωνο τους. Αυτή η αλλαγή μπορεί να εφαρμοστεί καθώς η μόνη πράξη που εκτελείται μεταξύ αποστάσεων είναι η σύγκριση και, επίσης, η τετραγωνική συνάρτηση είναι γνησίως αύξουσα. Έτσι μπορούμε να αποφύγουμε τον υπολογισμό της τετραγωνικής ρίζας σε όλους τους υπολογισμούς αποστάσεων. Η μοναδική φορά που απαιτείται να υπολογιστεί η τετραγωνική ρίζα είναι στο τέλος, όταν η διαδικασία επιστρέψει την απόσταση των δύο αντικειμένων.



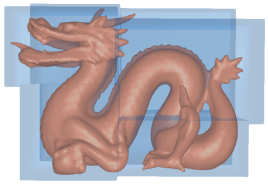
(α΄) Επίπεδο 1 (ρίζα)



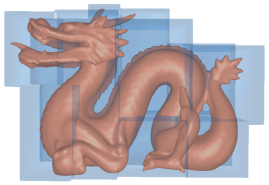
(β΄) Επίπεδο 2



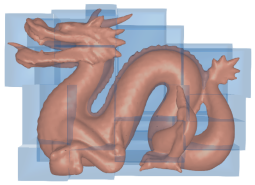
(γ΄) Επίπεδο 3



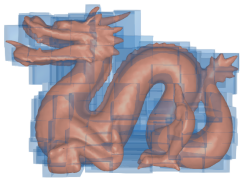
(δ΄) Επίπεδο 4



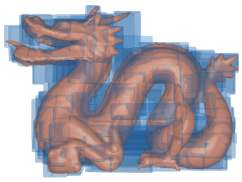
(ε΄) Επίπεδο 5



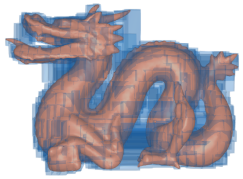
(ϛ΄) Επίπεδο 6



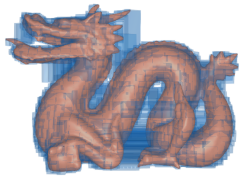
(ζ) Επίπεδο 9



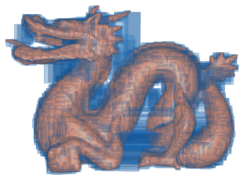
(η΄) Επίπεδο 10



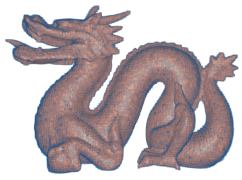
(θ΄) Επίπεδο 11



(ι΄) Επίπεδο 12



(ια΄) Επίπεδο 13



(ιβ΄) Φύλλα του δέντρου

Σχήμα 4.2: Κατασκευή sKD-Tree για το Stanford Dragon - Οι διάφορες εικόνες αναπαριστούν κάποια επίπεδα του δέντρου. Η τελευταία εικόνα απεικονίζει τα φύλλα του δέντρου. [4]

Κεφάλαιο 5

Πειράματα και Αποτελέσματα

Η εκτέλεση όλων των πειραμάτων που παρουσιάζονται παρακάτω πραγματοποιήθηκε σε πόρους της Ιδρυματικής συστοιχίας του ΑΠΘ ¨Αριστοτέλης¨. Το μοντέλο επεξεργαστή που χρησιμοποιήθηκε για τα πειράματα είναι Intel Xeon E5-2630 v4. Η υλοποίηση των αλγορίθμων έγινε στη γλώσσα "Julia Version 1.7.2".

5.1 Κόστος Υπολογισμού Απόστασης Στοιχείων

Για τον προσδιορισμό του κόστους C_v και C_p της μετρικής κόστους της ενότητας 4.3 θεωρήσαμε πως το κόστος υπολογισμού της τετραγωνική απόστασης δύο σημείων είναι μονάδα. Όλα τα υπόλοιπα κόστη, λοιπόν, υπολογίζονται σε σχέση με το κόστος σημείο-σημείο. Μετρήσαμε τον χρόνο υπολογισμού των τετραγωνικών αποστάσεων περίπου 210×10^6 σημείων, AABB, ευθυγράμμων τμημάτων και τριγώνων και προέκυψε ο πίνακας 5.1 με τα σχετικά κόστη. Επομένως, $C_v = 23.5$, $C_p = 385.7$.

Υπολογισμός Απόστασης	Κόστος
Σημείο-Σημείο	1
AABB-AABB	23.5
Ευθ. Τμήμα - Ευθ. Τμήμα	129.8
Τρίγωνο-Τρίγωνο	385.7

Πίνακας 5.1: Σχετικό κόστος υπολογισμού της απόστασης διαφόρων στοιχείων

5.2 Κατασκευή Δεδομένων Ελέγχου

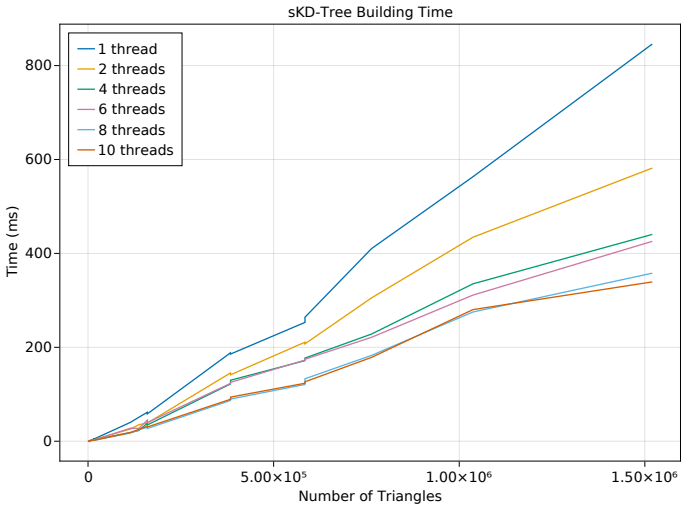
Για την εκτέλεση των πειραμάτων κατασκευάσαμε διάφορα σενάρια. Κάθε σενάριο αποτελείται από δύο αντικείμενα και για το κάθε αντικείμενο κατασκευάσαμε τριγωνικά πλέγματα με τρεις διαφορετικές αναλύσεις (διαφορετικό πλήθος τριγώνων). Έτσι, για κάθε σενάριο εκτελούμε ερωτήματα εύρεσης της απόστασης των αντικειμένων για κάθε συνδυασμό ανάλυσης (σύνολο 9 μετρήσεις). Έγινε προσπάθεια συμπερίληψης διάφορων περιπτώσεων, με αντικείμενα διαφόρων σχημάτων, διαστάσεων και προσανατολισμού.

Η λήψη των αντικειμένων που χρησιμοποιούνται σε αυτή την εργασία έγινε από το GrabCAD [2] και το 3D Content Central [1]. Τα αντικείμενα, αρχικά, ήταν σε μορφή

IGES, ενώ για τη δημιουργία των πλεγμάτων χρησιμοποιήθηκε το πρόγραμμα BETA CAE SYSTEMS - Ansa v21.0.0.

5.3 Χρόνος Κατασκευής του sKD-Tree

Στο σχήμα 5.1 παρουσιάζονται οι μετρήσεις του χρόνου κατασκευής του sKD-Tree συναρτήσει του μεγέθους ενός τριγωνικού πλέγματος και του πλήθους των νημάτων (threads) που χρησιμοποιήθηκαν.



Σχήμα 5.1: Χρόνος κατασκευής του sKD-Tree συναρτήσει του πλήθους των τριγώνων ενός πλέγματος.

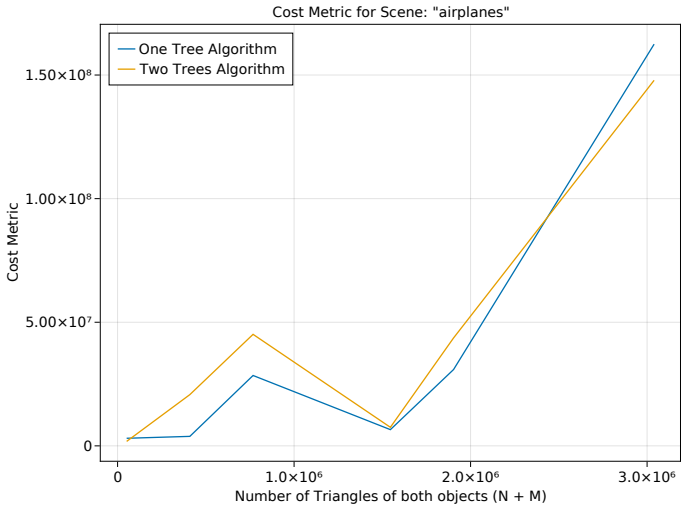
5.4 Πειράματα ανά Σενάριο

Για κάθε ένα από τα παρακάτω σενάρια, παρουσιάζονται οι μετρήσεις για το κόστος αναζήτησης και το συνολικό χρόνο εκτέλεσης για την απάντηση ερωτημάτων απόστασης. Στα διαγράμματα παρακάτω, γίνεται σύγκριση της επίδοσης των αλγορίθμων 10 και 11.

Το κόστος αναζήτησης υπολογίστηκε από τη μετρική κόστους που περιγράφεται στην ενότητα 4.3 μετρώντας τον αριθμό κλήσεων των ρουτινών `triangle_distance()` και `AABB_distance()`.

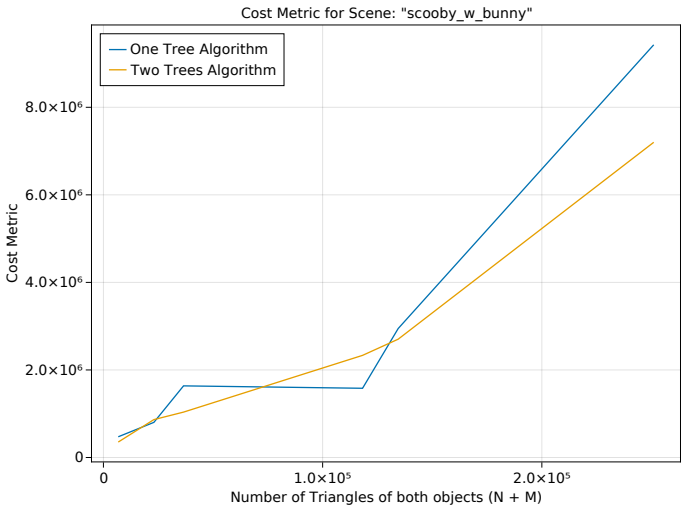
Ο συνολικός χρόνος εκτέλεσης ερωτημάτων απόστασης περιλαμβάνει το χρόνο προεπεξεργασίας των δεδομένων και τον χρόνο αναζήτησης. Η προεπεξεργασία των δεδομένων είναι η κατασκευή ενός ή δύο δέντρων ανάλογα με τον αλγόριθμο που χρησιμοποιείται. Οι χρόνοι που παρουσιάζονται αντιστοιχούν σε πειράματα με χρήση ενός, τεσσάρων και οκτώ νημάτων επεξεργασίας.

5.4.1 Αεροπλάνα



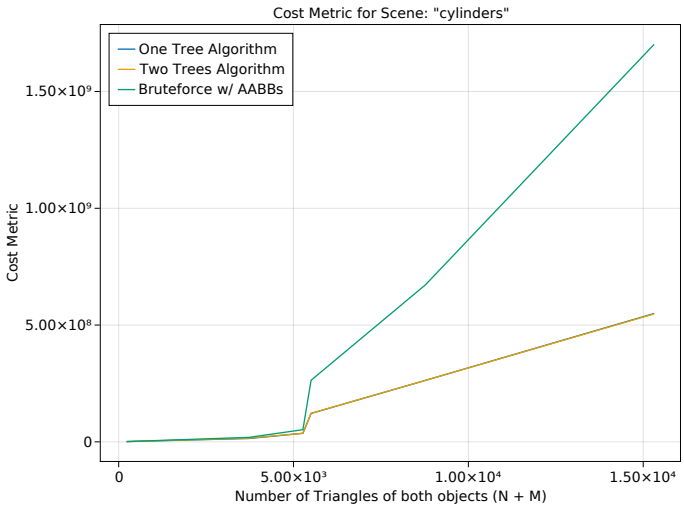
Σχήμα 5.2: Κόστος αναζήτησης για τη σκηνή ¨αεροπλάνα¨ συναρτήσει του μεγέθους των δύο μοντέλων.

5.4.2 Scooby με Stanford Bunny



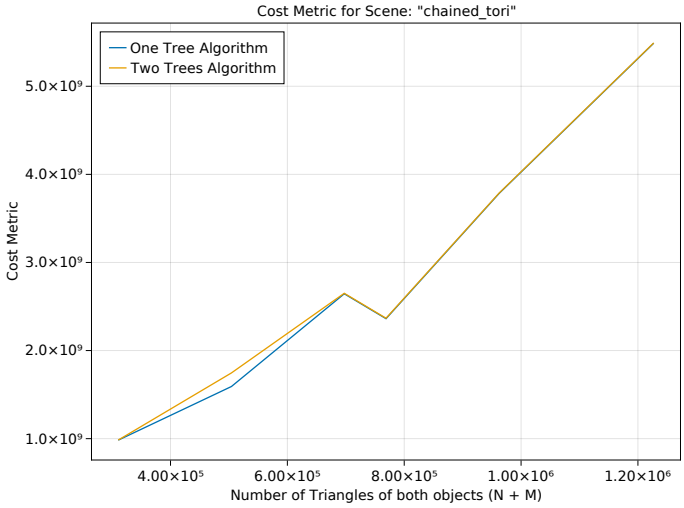
Σχήμα 5.3: Κόστος αναζήτησης για τη σκηνή ¨Scooby με Stanford Bunny¨ συναρτήσει του μεγέθους των δύο μοντέλων.

5.4.3 Ομοαξονικοί Κύλινδροι



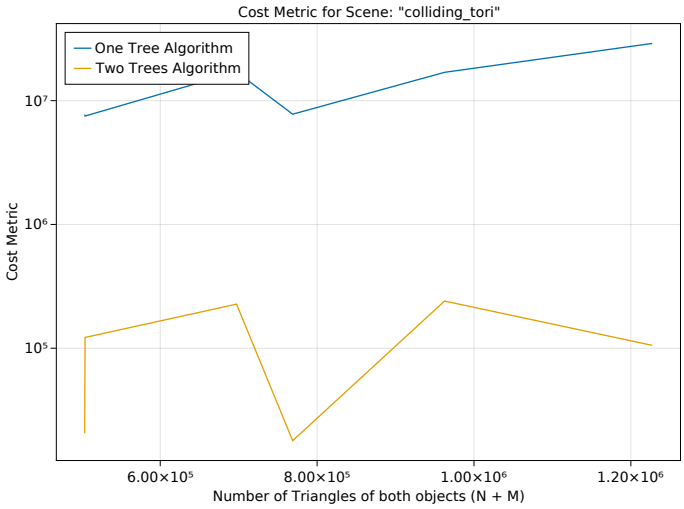
Σχήμα 5.4: Κόστος αναζήτησης για τη σκηνή ¨ομοαξονικοί κύλινδροι¨ συναρτήσει του μεγέθους των δύο μοντέλων.

5.4.4 Αλυσιδωτοί Τόροι



Σχήμα 5.5: Κόστος αναζήτησης για τη σκηνή ¨αλυσιδωτοί τόροι¨ συναρτήσει του μεγέθους των δύο μοντέλων.

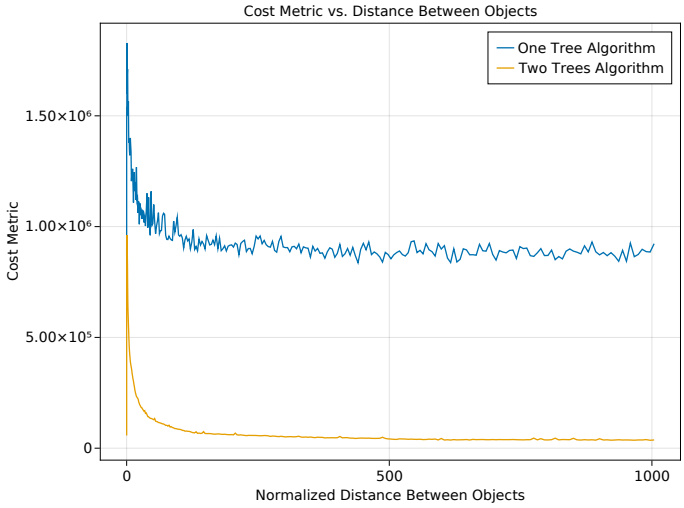
5.4.5 Συγκρουόμενοι Τόροι



Σχήμα 5.6: Κόστος αναζήτησης για τη σκηνή ¨συγκρουόμενοι τόροι¨ συναρτήσει του μεγέθους των δύο μοντέλων.

5.5 Σχέση Απόστασης - Κόστους Αναζήτησης

Για το πείραμα αυτό επιλέχθηκαν τα αντικείμενα της σκηνής ¨Scooby με Stanford Bunny¨ με περίπου 20000 και 16600 τρίγωνα αντίστοιχα. Αρχικά τα αντικείμενα τοποθετούνται έτσι ώστε να υπάρχει σύγκρουση μεταξύ τους. Έπειτα μετακινείται το αντικείμενο Stanford Bunny προς μια συγκεκριμένη κατεύθυνση ώστε να απομακρύνεται από το αντικείμενο Scooby.



Σχήμα 5.7: Κόστος αναζήτησης συναρτήσει της απόστασης. Η απόσταση είναι κανονικοποιημένη και ίση με $\frac{dist}{diagonal}$, όπου *dist* η πραγματική απόσταση των αντικειμένων και *diagonal* το μήκος της διαγωνίου του AABB του Stanford Bunny.

Κεφάλαιο 6

Συμπεράσματα και Μελλοντική Εργασία

6.1 Σχολιασμός των Αποτελεσμάτων από τα Πειράματα

6.2 Μελλοντική Εργασία

Παράρτημα Α΄

Ακρωνύμια και συντομογραφίες

AABB Axis-Aligned Bounding Box

BSP Binary Space Partitioning

BVH Bounding Volume Hierarchy

CAE Computer Aided Engineering

DFS Depth First Search

FEA Finite Element Analysis

MBR Minimum Bounding Rectangle

OBB Oriented Bounding Box

SAH Surface Area Heuristic

SAT Separating Axis Theorem

Bibliography

[1] 3D Content Central. https://www.3dcontentcentral.com/.

[2] GrabCAD. https://grabcad.com/library.

[3] T. Aila, T. Karras, and S. Laine. On quality metrics of bounding volume hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference*, pages 101–107, 2013.

[4] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[5] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[6] J. L. Bentley, D. Haken, and J. B. Saxe. A general method for solving divide-and-conquer recurrences. *ACM SIGACT News*, 12(3):36–44, 1980.

[7] J. E. Bobrow. A direct minimization approach for obtaining the distance between convex polyhedra. *The International Journal of Robotics Research*, 8(3):65–76, 1989.

[8] J. W. Boyse. Interference detection among solids and surfaces. *Communications of the ACM*, 22(1):3–9, 1979.

[9] R. A. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, (2):224–233, 1985.

[10] S. Cameron. A study of the clash detection problem in robotics. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 488–493. IEEE, 1985.

[11] J. Canny. Collision detection for moving polyhedra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):200–209, 1986.

[12] F. Chin and C. A. Wang. Optimal algorithms for the intersection and the minimum distance problems between planar polygons. *IEEE Transactions on Computers*, 32(12):1203–1207, 1983.

[13] R. Culley and K. Kempf. A collision detection algorithm based on velocity and distance bounds. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1064–1069. IEEE, 1986.

[14] D. P. Dobkin and D. G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *Journal of algorithms*, 6(3):381–392, 1985.

[15] H. Edelsbrunner. Computing the extreme distances between two convex polygons. *Journal of Algorithms*, 6(2):213–224, 1985.

[16] C. Ericson. *Real-Time Collision Detection*. Morgan Kaufmann series in interactive 3D technology. Taylor & Francis, 2004.

[17] H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 124–133, 1980.

[18] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.

[19] J. Goldsmith and J. Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, 1987.

[20] S. Gottschalk. Separating axis theorem. 1996.

[21] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180, 1996.

[22] Y. Gu, Y. He, K. Fatahalian, and G. Blelloch. Efficient bvh construction via approximate agglomerative clustering. In *Proceedings of the 5th High-Performance Graphics Conference*, pages 81–88, 2013.

[23] A. Guezlec. " meshsweeper": dynamic point-to-polygonal mesh distance and applications. *IEEE Transactions on visualization and computer graphics*, 7(1):47–61, 2001.

[24] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, 1984.

[25] C. A. Hoare. Algorithm 65: find. *Communications of the ACM*, 4(7):321–322, 1961.

[26] D. E. Johnson and E. Cohen. A framework for efficient minimum distance computations. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 4, pages 3678–3684. IEEE, 1998.

[27] T. Karras. Thinking parallel, part iii: Tree construction on the gpu. `https://developer.nvidia.com/blog/thinking-parallel-part-iii-tree-construction-gpu/`, December 2012.

[28] A. Khamayseh and G. Hansen. Use of the spatial kd-tree in computational physics applications. *Commun Comput Phys*, 2(3):545–576, 2007.

[29] A. Krishnamurthy, S. McMains, and K. Haller. Gpu-accelerated minimum distance and clearance queries. *IEEE transactions on visualization and computer graphics*, 17(6):729–742, 2011.

[30] E. Larsen, S. Gottchalk, M. Lin, and D. Manocha. Pqp-a proximity query package, 2014.

[31] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical report, Technical Report TR99-018, Department of Computer Science, University of . . . , 1999.

[32] V. J. Lumelsky. On fast computation of distance between line segments. *Information Processing Letters*, 21(2):55–61, 1985.

[33] J. D. MacDonald and K. S. Booth. Heuristics for ray tracing using space subdivision. *The Visual Computer*, 6(3):153–166, 1990.

[34] D. Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147, 1982.

[35] M. Moore and J. Wilhelms. Collision detection and response for computer animation. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 289–298, 1988.

[36] S. Quinlan. Efficient distance computation between non-convex objects. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3324–3329. IEEE, 1994.

[37] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 71–79, 1995.

[38] J. T. Schwartz. Finding the minimum distance between two convex polygons. *Information Processing Letters*, 13(4-5):168–170, 1981.

[39] D. A. Simon. *Fast and accurate shape-based registration*. Carnegie Mellon University, 1996.

[40] P. N. Yianilos. Data structures and algorithms for nearest neighbor. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, volume 66, page 311. SIAM, 1993.