ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

Αποκοπή κυρτών περιβλημάτων ως εφαρμογή εξακρίβωσης παραμέτρων διμεταβλήτων μορφοκλασματικών επιφανειών παρεμβολής και ως επέκταση γλωσσών οπτικού προγραμματισμού

Δημήτριος Ματθές

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Επιβλέπων

Βασίλειος Δρακόπουλος

Λαμία, Μάρτιος 2025

UNIVERSITY OF THESSALY
SCHOOL OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE AND BIOMEDICAL INFORMATICS

Convex hull clipping as an application for parameter identification of bivariate fractal interpolation surfaces and as an extension of visual programming languages

Dimitrios Matthes

PHD THESIS

Supervisor

Vasileios Drakopoulos

Lamia, March 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

Αποκοπή κυρτών περιβλημάτων ως εφαρμογή εξακρίβωσης παραμέτρων διμεταβλήτων μορφοκλασματικών επιφανειών παρεμβολής και ως επέκταση γλωσσών οπτικού προγραμματισμού

Δημήτριος Ματθές

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Επιβλέπων

Βασίλειος Δρακόπουλος

Λαμία, Μάρτιος 2025

«Υπεύθυνη Δήλωση μη λογοκλοπής και ανάληψης προσωπικής ευθύνης»

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων και γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπευθύνως και ενυπογράφως ότι η παρούσα διατριβή με τίτλο «Αποκοπή κυρτών περιβλημάτων ως εφαρμογή εξακρίβωσης παραμέτρων διμεταβλήτων μορφοκλασματικών επιφανειών παρεμβολής και ως επέκταση γλωσσών οπτικού προγραμματισμού» αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές από τις οποίες χρησιμοποίησα δεδομένα, ιδέες, φράσεις, προτάσεις ή λέξεις, είτε επακριβώς (όπως υπάρχουν στο πρωτότυπο ή μεταφρασμένες) είτε με παράφραση, έχουν δηλωθεί κατάλληλα και ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει, διότι είναι προϊόν λογοκλοπής.

Ο ΔΗΛΩΝ

27 Μαρτίου 2025

Υπογραφή

Αποκοπή κυρτών περιβλημάτων ως εφαρμογή εξακρίβωσης παραμέτρων διμεταβλήτων μορφοκλασματικών επιφανειών παρεμβολής και ως επέκταση γλωσσών οπτικού προγραμματισμού

Δημήτριος Ματθές

Τριμελής Συμβουλευτική Επιτροπή

1. Δρακόπουλος Βασίλειος, Αναπληρωτής Καθηγητής (επιβλέπων), Πανεπιστήμιο Θεσσαλίας
2. Θεοχάρης Θεοχάρης, Καθηγητής, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
3. Πλατής Νικόλαος, Επίκουρος Καθηγητής, Πανεπιστήμιο Πελοποννήσου

Επταμελής Εξεταστική Επιτροπή

1. Δελήμπασης Κωνσταντίνος, Καθηγητής, Πανεπιστήμιο Θεσσαλίας
2. Δρακόπουλος Βασίλειος, Αναπληρωτής Καθηγητής (επιβλέπων), Πανεπιστήμιο Θεσσαλίας
3. Θεοχάρης Θεοχάρης, Καθηγητής, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
4. Παπαϊωάννου Γεώργιος, Αναπληρωτής Καθηγητής, Οικονομικό Πανεπιστήμιο Αθηνών
5. Πλαγιανάκος Βασίλειος, Καθηγητής - Κοσμήτορας Σχολής Θετικών Επιστημών, Πανεπιστήμιο Θεσσαλίας
6. Πλατής Νικόλαος, Επίκουρος Καθηγητής, Πανεπιστήμιο Πελοποννήσου
7. Σαβελώνας Μιχαήλ, Αναπληρωτής Καθηγητής, Πανεπιστήμιο Θεσσαλίας

"It's very satisfying to take a problem we thought difficult and find a simple solution. The best solutions are always simple."

Ivan Sutherland

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διδακτορική διατριβή αποτέλεσε ένα μακρύ και δύσκολο ταξίδι που διήρκησε περίπου έξι έτη. Στην πορεία του ταξιδιού αυτού μοιράστηκα τους προβληματισμούς μου, τις ανησυχίες, τα διλήμματα, τις γνώσεις και την εμπειρία μου με κάποιους ανθρώπους στους οποίους οφείλω να εκφράσω την ευγνωμοσύνη μου. Αν και φαινομενικά η εκπόνηση μιας διδακτορικής διατριβής δείχνει να είναι μία μοναχική πορεία, στην πράξη βασίζεται σε κοινές εποικοδομητικές σχέσεις και συνεργασίες μέσα από τις οποίες ο υποψήφιος οδηγείται σε ένα τελικό αποτέλεσμα.

Για τον λόγο αυτό, θα ήθελα πρωτίστως να ευχαριστήσω θερμά τον επιβλέποντά μου, Βασίλειο Δρακόπουλο, Αναπληρωτή Καθηγητή Πανεπιστημίου Θεσσαλίας, για το ενδιαφέρον και τις συμβουλές του από την αρχή μέχρι και την ολοκλήρωση της πορείας αυτής. Με την καθοδήγησή του, τις εύστοχες παρατηρήσεις του, τις ενδιαφέρουσες τοποθετήσεις του, την ανεξάντλητη υπομονή και κατανόησή του καθώς και τη γενναιόδωρη διάθεση του χρόνου του, συνέβαλε καθοριστικά στην ολοκλήρωση αυτού του εγχειρήματος.

Θερμά ευχαριστώ και τον Θεοχάρη Θεοχάρη, Καθηγητή Ε.Κ.Π.Α, για τις εύστοχες υποδείξεις του σε αρκετά σημεία της εργασίας μου. Ομοίως, τον Νικόλαο Πλατή, Επίκουρο Καθηγητή Πανεπιστημίου Πελοποννήσου, για τον έλεγχο τόσο σε σημεία του κώδικα που αφορούσαν την τομή των κυρτών πολυέδρων όσο και για τις γενικότερες παρατηρήσεις του επί της διατριβής. Εκτιμώ ιδιαιτέρως και ευχαριστώ για τις συμβουλές του τον Κωνσταντίνο Δελήμπαση, Καθηγητή Πανεπιστημίου Θεσσαλίας, σχετικά με τα αποτελέσματα της αξιολόγησης των αλγορίθμων αποκοπής όπως και για την διάταξη ορισμένων κεφαλαίων. Θετική ενίσχυση έλαβα και από τον Μιχαήλ Σαβελώνα, Αναπληρωτή Καθηγητή Πανεπιστημίου Θεσσαλίας, ο οποίος μέσω των παραινέσεών του με βοήθησε σημαντικά σε διάφορες στιγμές της όλης μου πορείας. Τέλος, ήταν μεγάλη μου τιμή να συμμετέχουν στην επιτροπή εξέτασης της διδακτορικής μου διατριβής, διαθέτοντας μέρος από τον πολύτιμο χρόνο τους, ο Βασίλειος Πλαγιανάκος, Καθηγητής και Κοσμήτορας της Σχολής Θετικών Επιστημών του Πανεπιστημίου Θεσσαλίας, καθώς και ο Γεώργιος Παπαϊωάννου, Αναπληρωτής Καθηγητής του Οικονομικού Πανεπιστημίου Αθηνών.

Ιδιαίτερες ευχαριστίες θα ήθελα επίσης να απευθύνω στον Ιωάννη Σ. Τριανταφύλλου, Αναπληρωτή Καθηγητή

Πανεπιστημίου Πειραιώς, για την πολύτιμη βοήθεια και τις συμβουλές του στο κομμάτι της στατιστικής ανάλυσης όπως επίσης και στους (συν)υποψήφιους διδάκτορες: Νικόλαο Νταούλα, Παναγιώτη Σιούλα και Ναυσικά Τεγούση, με τους οποίους συνεργάστηκα αρκετές φορές, εντός και εκτός πλαισίων του διδακτορικού μου έργου.

Τέλος, θα ήθελα να αναφερθώ και στους αγαπημένους μου ανθρώπους, που στάθηκαν δίπλα μου με υπομονή σε όλη αυτή την προσπάθειά μου. Ευχαριστώ τους γονείς μου, Γιάννη και Μαρία, καθώς και την αδελφή μου, Παναγιώτα, για την ηθική τους υποστήριξη αλλά πάνω από όλα την αγάπη τους. Επίσης, νιώθω χρέος να ευχαριστήσω την σύζυγό μου, Άλκηστη, για την αμέριστη συμπαράσταση, υποστήριξη, υπομονή και κατανόησή της, όπως και για την γενναιοδωρία της ψυχής της που καθημερινά ενέπνεε και έδινε δύναμη στον αγώνα μου από την αρχή μέχρι το τέλος.

ΠΕΡΙΛΗΨΗ

Σε αυτή τη διατριβή μελετάται η αποκοπή ευθυγράμμων τμημάτων δια μέσου κυρτών περιβλημάτων με εφαρμογές στον οπτικό προγραμματισμό καθώς και η εξακρίβωση παραμέτρων συγκεκριμένων Μορφοκλασματικών Επιφανειών Παρεμβολής (Μ.Ε.Π.). Αποτελείται από δύο μέρη: Το πρώτο μέρος ασχολείται με την αποκοπή τόσο σε κυρτό παράθυρο στις δύο διαστάσεις, όσο και σε κυρτό όγκο στις τρεις διαστάσεις ενώ το δεύτερο μέρος σχετίζεται με τις Μ.Ε.Π.

Η λογικότερη απορία που μπορεί να δημιουργηθεί σε κάποιον διαβάζοντας το πρώτο μέρος και γνωρίζοντας, έστω και ελάχιστα, τι είναι ένα μορφόκλασμα, είναι το πως σχετίζονται μεταξύ τους αυτές οι δύο φαινομενικά ασύνδετες έννοιες. Η απάντηση είναι απλή: Οι αλγόριθμοι αποκοπής που δημιουργήθηκαν στο πρώτο μέρος από τον συγγραφέα και τον επιβλέποντα έχουν ένα κοινό χαρακτηριστικό στον τρόπο λειτουργίας τους: Σε όλες τις περιπτώσεις αποκοπής ευθυγράμμων τμημάτων, είτε στις δύο είτε στις τρεις διαστάσεις, εφ' όσον το παράθυρο ή ο όγκος αποκοπής είναι κυρτός, ο εκάστοτε αλγόριθμος εξετάζει κάθε σύνορό τους «εναγκαλιζόμενος» το ευθύγραμμο τμήμα και αποκόπτοντάς το σταδιακά. Η διαπίστωση αυτή οδήγησε την έρευνα στα μορφοκλάσματα όπου για την κατασκευή Μ.Ε.Π. και την εύρεση της βέλτιστης τιμής της ελεύθερης παραμέτρου κατακόρυφης κλιμάκωσης απαιτείται ο υπολογισμός της τομής κυρτών πολυέδρων. Η λογική του εναγκαλισμού βρίσκει απόλυτη εφαρμογή και στην περίπτωση αυτή.

SUMMARY

In this thesis, the clipping of straight-line segments through convex hulls is studied, with applications in visual programming as well as the parameter identification of specific Fractal Interpolation Surfaces, and it consists of two parts. The first part addresses line clipping against a convex region in two dimensions and a convex volume in three dimensions, while the second part focuses on Fractal Interpolation Surfaces.

After reading the first part and understanding the concept of fractals, one may wonder: How these two seemingly unrelated concepts are connected? The answer is simple. All clipping-related algorithms developed in the first part by the author and the supervisor share a common approach in their functionality. In every case of line segment clipping—whether in two or three dimensions—when the clipping region or volume is convex, the algorithm examines each clipping boundary by "embracing" the segment and incrementally clipping it.

This observation naturally led the research toward fractals. Specifically, when determining the optimal value of the free vertical scaling parameter in the creation of fractal interpolation surfaces, the intersection of convex polyhedra is required. The concept of embracing applies seamlessly in this context as well.

ΕΙΣΑΓΩΓΗ

Η διατριβή αυτή εστιάζει στις μεθόδους αποκοπής ευθυγράμμων τμημάτων στις δύο και στις τρεις διαστάσεις, καθώς και στην εξακρίβωση παραμέτρων συγκεκριμένων Μορφοκλασματικών Επιφανειών Παρεμβολής (Μ.Ε.Π.). Είναι χωρισμένη σε δύο μέρη. Το πρώτο επικεντρώνεται στην αποκοπή ευθυγράμμων τμημάτων. Πιο συγκεκριμένα, το Κεφάλαιο 1 αναφέρεται γενικά στην έννοια της αποκοπής. Το Κεφάλαιο 2 σχετίζεται με την αποκοπή ευθυγράμμων τμημάτων σε ορθογώνιο παράθυρο αποκοπής στις δύο διαστάσεις. Σε αυτό αναλύονται και αξιολογούνται σημαντικοί αλγόριθμοι όπως οι Cohen-Sutherland, Liang-Barsky, Nicholl-Lee-Nicholl, Midpoint Subdivision, Skala 2005, S-Clip E2, Kodituwakku-Wijeweera-Chamikara, ενώ προτείνεται και ένας αλγόριθμος ο οποίος δημιουργήθηκε από τον συγγραφέα και τον επιβλέποντα. Στο Κεφάλαιο 3 εξετάζονται οι αλγόριθμοι που σχετίζονται με την αποκοπή στις δύο διαστάσεις αλλά με τη διαφορά ότι το παράθυρο είναι κυρτό πολύγωνο. Στην περίπτωση αυτή προτείνεται παρομοίως ένας αλγόριθμος ο οποίος δημιουργήθηκε από τον συγγραφέα και τον επιβλέποντα. Στο Κεφάλαιο 4, παρουσιάζονται και αξιολογούνται αλγόριθμοι αποκοπής ευθυγράμμων τμημάτων σε κυρτό όγκο θέασης στις τρεις διαστάσεις, ενώ προτείνεται και ένας αλγόριθμος του συγγραφέα και του επιβλέποντα για την περίπτωση αυτή.

Το δεύτερο μέρος αφορά την μορφοκλασματική παρεμβολή και την εξακρίβωση παραμέτρων συγκεκριμένων Μ.Ε.Π. Αποτελείται από το Κεφάλαιο 5 στο οποίο γίνεται παρουσίαση των Μ.Ε.Π. και αναφέρονται οι προαπαιτούμενες μαθηματικές έννοιες για την καλύτερη κατανόησή τους. Στο Κεφάλαιο 6 αναλύονται τα Επαναλαμβανόμενα Συστήματα Συναρτήσεων (Ε.Σ.Σ.) τα οποία σχετίζονται με την κατασκευή των Μ.Ε.Π. Στο Κεφάλαιο 7 παρουσιάζεται ένας τρόπος δημιουργίας Μ.Ε.Π. και ακολουθεί μία μεθοδολογία για την εύρεση της βέλτιστης τιμής των παραγόντων κατακόρυφης κλιμάκωσης. Η μεθοδολογία του Κεφαλαίου 7 απαιτεί τις μεθόδους οι οποίες αναπτύσσονται στο Κεφάλαιο 8, το οποίο αναφέρεται στην έννοια του κυρτού περιβλήματος, στο Κεφάλαιο 9, το οποίο αναφέρεται στον υπολογισμό του όγκου των κυρτών πολυέδρων, και στο Κεφάλαιο 10, το οποίο αναλύει την έννοια της τομής μεταξύ κυρτών πολυέδρων. Στο Κεφάλαιο 11, παρουσιάζεται η εφαρμογή «Μορφοκλασματικές Επιφάνειες Παρεμβολής» που δημιουργήθηκε για την απεικόνιση των Μ.Ε.Π. μέσω Η/Υ, ενώ στο τελευταίο

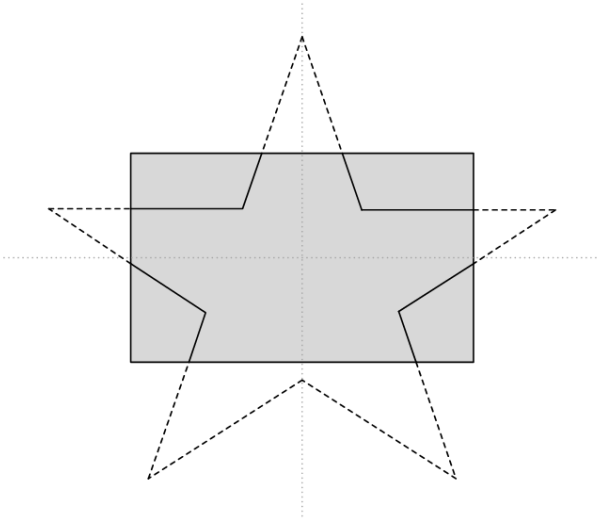
Κεφάλαιο 12, συνοψίζουμε την παρούσα διατριβή μαζί με τα κύρια αποτελέσματα της έρευνας, τα συμπεράσματα, τις προοπτικές και το μελλοντικό έργο.

Μέρος Ι

ΑΠΟΚΟΠΗ ΕΥΘΥΓΡΑΜΜΩΝ ΤΜΗΜΑΤΩΝ ΣΕ ΚΥΡΤΟ ΠΑΡΑΘΥΡΟ ΑΠΟΚΟΠΗΣ ΚΑΙ ΣΕ ΚΥΡΤΟ ΟΓΚΟ ΘΕΑΣΗΣ

1. ΓΕΝΙΚΑ ΠΕΡΙ ΑΠΟΚΟΠΗΣ

Κατά τη δημιουργία γραφικών στους υπολογιστές εμφανίζεται πολλές φορές η ανάγκη της αποκοπής ορισμένων αντικειμένων τα οποία βρίσκονται εκτός ενός συγκεκριμένου πλαισίου. Τα αντικείμενα αυτά ενδέχεται να είναι σημεία, ευθύγραμμα τμήματα, τρίγωνα, πολύγωνα ή ακόμα και πολύπλοκα γεωμετρικά σχήματα. Στις δύο διαστάσεις, το πλαίσιο στο οποίο ο δημιουργός θέλει να περιορίσει το γραφικό αποτέλεσμα είναι, συνήθως, ένα ορθογώνιο παραλληλόγραμμο ευθυγραμμισμένο με τον οριζόντιο και τον κατακόρυφο άξονα του Καρτεσιανού συστήματος συντεταγμένων. Φυσικά, υπάρχουν και περιπτώσεις όπου αυτό είναι άλλου σχήματος, όπως κυκλικό, κυρτό ή μη κυρτό πολύγωνο κ.ο.κ. [Com06]. Ως επί το πλείστον, η περίπτωση του πλαισίου με σχήμα ορθογωνίου παραλληλογράμμου ταυτίζεται με τα όρια της οθόνης του υπολογιστή. Το πλαίσιο στο οποίο περιορίζεται εντός του το αποτέλεσμα της δημιουργίας γραφικών, ενώ αποκόπτεται οτιδήποτε άλλο υπάρχει εκτός αυτού, ονομάζεται «παράθυρο αποκοπής». Κατά αντιστοιχία, η διαδικασία με την οποία εξαλείφονται τα τμήματα των γραφικών τα οποία βρίσκονται εκτός του παραθύρου αποκοπής ονομάζεται «αλγόριθμος αποκοπής» ή απλώς αποκοπή (Σχ. 1.1). Στις τρεις διαστάσεις, το αντίστοιχο παράθυρο ονομάζεται «όγκος θέασης».



Σχήμα 1.1: Αστέρι το οποίο απαρτίζεται από ευθύγραμμα τμήματα και αποκόπτεται

Ο σχεδιασμός ενός αρχεγόνου, δηλαδή ενός στοιχειώδους αντικείμενου, όπως ένα σημείο ή ένα ευθύγραμμο τμήμα, μπορεί να έχει τις εξής πιθανές σχέσεις με το παράθυρο αποκοπής:

1. Να βρίσκεται εξ ολοκλήρου εντός αυτού, οπότε σχεδιάζεται κανονικά.
2. Να βρίσκεται εξ ολοκλήρου εκτός αυτού, οπότε δεν σχεδιάζεται καθόλου.
3. Να τέμνει το παράθυρο, οπότε πρέπει να πραγματοποιηθεί αποκοπή και να σχεδιαστεί μόνο το τμήμα που βρίσκεται εντός του.

Η αποκοπή εστιάζει κυρίως στην τρίτη περίπτωση και περιλαμβάνει αλγόριθμους σχετικούς με:

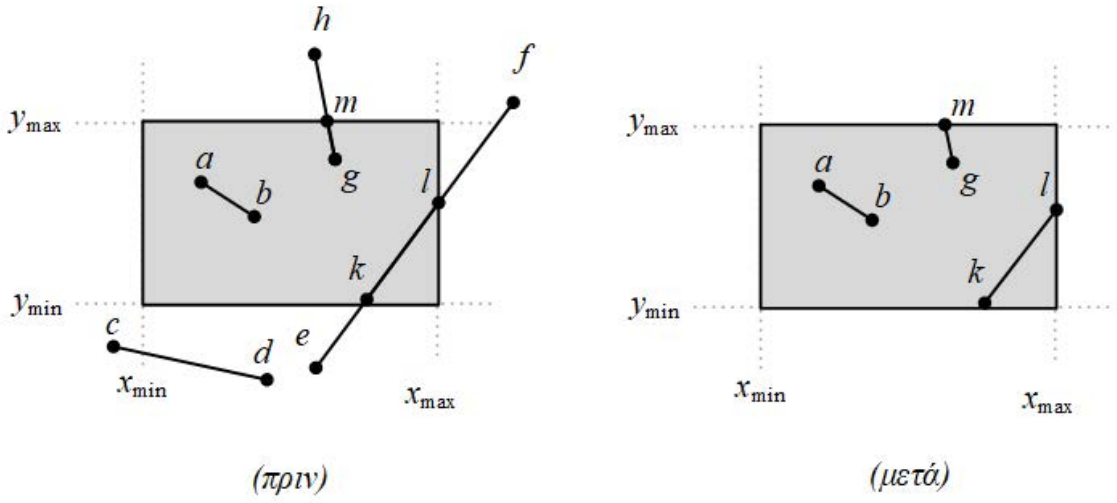
- Αποκοπή σημείου
- Αποκοπή ευθυγράμμου τμήματος
- Αποκοπή περιοχής γεμίσματος (πολύγωνα)
- Αποκοπή καμπύλης
- Αποκοπή κειμένου

Οι αλγόριθμοι αποκοπής σημείων, ευθυγράμμων τμημάτων και πολυγώνων θεωρούνται πολύ σημαντικοί όσον αφορά την γραφική υπολογιστών. Όμως, παρόμοιες μέθοδοι μπορούν να εφαρμοστούν και σε άλλα αντικείμενα, ιδιαίτερα σε εκείνα που έχουν κωνικό σχήμα, όπως κύκλους, ελλείψεις και σφαίρες. Ωστόσο, τα αντικείμενα με μη γραμμικά συνήθως σύνορα προσεγγίζονται με ευθύγραμμα τμήματα ή επιφάνειες πολυγώνου για τη μείωση των υπολογισμών [HBC14].

Ένας αποτελεσματικός τρόπος για αποκοπή στην σωλήνωση των γραφικών είναι η εφαρμογή των αλγορίθμων στα κανονικοποιημένα σύνορα του παραθύρου αποκοπής, συνήθως από -1 έως 1. Αυτό μειώνει τους υπολογισμούς, επειδή όλα τα μητρώα μετασχηματισμών και προβολής μπορούν να ενωθούν και να εφαρμοστούν σε μία περιγραφή της σκηνής πριν από την πραγματοποίηση της αποκοπής. Η αποκομμένη σκηνή μπορεί στη συνέχεια να μεταφερθεί στις συντεταγμένες της οθόνης για την τελική επεξεργασία.

2. ΑΠΟΚΟΠΗ ΕΥΘΥΓΡΑΜΜΟΥ ΤΜΗΜΑΤΟΣ ΑΠΟ ΟΡΘΟΓΩΝΙΟ ΠΑΡΑΘΥΡΟ ΑΠΟΚΟΠΗΣ ΣΤΙΣ ΔΥΟ ΔΙΑΣΤΑΣΕΙΣ

Ένας αλγόριθμος αποκοπής ευθυγράμμου τμήματος ελέγχει ένα προς ένα το κάθε ευθύγραμμο τμήμα μίας σκηνής και προσπαθεί, μέσα από μία σειρά συγκρίσεων και υπολογισμών, να εντοπίσει, εάν αυτό ευρίσκεται εξ ολοκλήρου εντός ή εκτός αυτής ή απλώς τέμνει τα σύνορά της σε ένα ή δύο σημεία (βλ. Σχ. 2.1). Στην τελευταία περίπτωση, ο αλγόριθμος πρέπει να υπολογίσει, επιπροσθέτως, και τα σημεία τομής του ευθυγράμμου τμήματος με τα σύνορα αυτά. Ο προσδιορισμός των σημείων τομής είναι υπολογιστικά δαπανηρός οπότε βασικός στόχος για οποιονδήποτε αλγόριθμο είναι να ελαχιστοποιήσει τέτοιου είδους υπολογισμούς.



Σχήμα 2.1: Ευθύγραμμα τμήματα πριν και μετά την αποκοπή

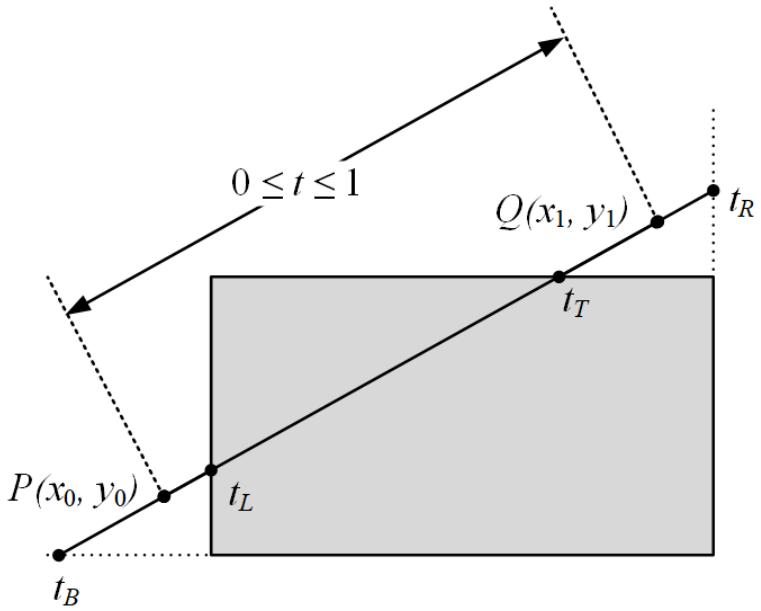
Για να γίνει κάτι τέτοιο πρέπει αρχικά να γίνουν συγκρίσεις προκειμένου να προσδιοριστεί, εάν το ευθύγραμμο τμήμα είναι εξ ολοκλήρου εντός ή εξ ολοκλήρου εκτός του παραθύρου αποκοπής. Εάν δεν προσδιοριστεί ότι ανήκει σε μία από τις δύο αυτές περιπτώσεις, τότε πρέπει να ακολουθήσουν οι υπολογισμοί για την εύρεση των σημείων τομής μεταξύ του ευθυγράμμου τμήματος και του παραθύρου.

Ένας τρόπος προσδιορισμού των σημείων τομής είναι η χρήση της παραμετρικής εξίσωσης του ευθυγράμμου

τμήματος, όπου τα σημεία με συντεταγμένες (x_0, y_0) και (x_1, y_1) ανήκουν σε αυτό:

$$x = x_0 + t \cdot (x_1 - x_0)$$
$$y = y_0 + t \cdot (y_1 - y_0) \qquad 0 \leq t \leq 1$$

Μέσω αυτής, μπορεί να υπολογιστεί σε ποιο σημείο το ευθύγραμμο τμήμα τέμνει τα σύνορα του παραθύρου αποκοπής εκχωρώντας την αντίστοιχη τιμή συντεταγμένων του συνόρου, είτε στο x είτε στο y, και λύνοντας ως προς την παράμετρο t. Για παράδειγμα, εάν προσδιοριστεί ότι το ένα εκ των δύο σημείων του ευθυγράμμου τμήματος βρίσκεται αριστερά του παραθύρου αποκοπής, τότε το αριστερό σύνορο x_{min} χρησιμοποιείται στην παραμετρική εξίσωση $(x = x_{\text{min}})$ προκειμένου να προσδιοριστεί η τιμή της παραμέτρου t και, εφ' όσον αυτή είναι μεταξύ $0 \leq t \leq 1$, να υπολογιστεί η νέα τεταγμένη y (βλ. Σχ. 2.2). Ωστόσο, εάν η τιμή της t βρίσκεται εκτός του εύρους $[0, 1]$ τότε το ευθύγραμμο τμήμα δεν τέμνει το συγκεκριμένο σύνορο.



Σχήμα 2.2: Υπολογισμός του σημείου τομής με χρήση της παραμετρικής εξίσωσης του ευθυγράμμου τμήματος

Εναλλακτικός τρόπος εύρεσης των σημείων τομής έναντι της παραμετρικής εξίσωσης αποτελεί και η κλασική εξίσωση ευθείας:

$$y - y_0 = m \cdot (x - x_0), \qquad m = \frac{y_1 - y_0}{x_1 - x_0}.$$

Η επεξεργασία των ευθυγράμμων τμημάτων μίας σκηνής χρησιμοποιώντας την προσέγγιση της αποκοπής που περιγράφηκε παραπάνω είναι απλή αλλά όχι πάντοτε αποτελεσματική. Οι αλγόριθμοι αποκοπής στοχεύουν στο να μειώσουν τον χρόνο των ελέγχων και της επεξεργασίας ώστε να επιτύχουν όσο το δυνατόν βέλτιστη απόδοση αποκοπής. Ανάλογα με τον σχεδιασμό τους, μερικοί από αυτούς αποκόπτουν αποκλειστικά στις δύο διαστάσεις ενώ άλλοι τροποποιούνται εύκολα αποκόπτοντας και στις τρεις διαστάσεις.

Βασικοί αλγόριθμοι αποκοπής ευθυγράμμων τμημάτων θεωρούνται οι ακόλουθοι:

1. Cohen–Sutherland
2. Cyrus–Beck [CB78]
3. Liang–Barsky [LB84]
4. Nicholl–Lee–Nicholl [NLN87]

Με την πάροδο των ετών έχουν δημιουργηθεί πολλοί άλλοι, όπως οι Midpoint Subdivision, Fast Clipping [SPY87], Skala '93 [Ska93], Skala '94 [Ska94], Skala 2005 [Ska05], S-Clip E^2 [Ska12], Ray [Ray12], Andreev and Sofianska [AS91], Day [Day92], Rappoport [Rap91], Dimri [Dim15], αλλά πολλοί από αυτούς αποτελούν παραλλαγές των τεσσάρων βασικών.

Οι αλγόριθμοι αποκοπής ευθυγράμμων τμημάτων μπορούν να χωριστούν σε τρεις κατηγορίες:

1. Σε εκείνους που πραγματοποιούν κωδικοποίηση των σημείων που προσδιορίζουν το ευθύγραμμο τμήμα. Χαρακτηριστικός στην κατηγορία αυτή είναι ο Cohen–Sutherland.
2. Σε εκείνους που κάνουν χρήση της παραμετρικής εξίσωσης του ευθυγράμμου τμήματος. Χαρακτηριστικοί στην κατηγορία αυτή είναι οι Cyrus–Beck και Liang–Barsky.
3. Σε όσους εφαρμόζουν υποδιαίρεση του μεσαίου σημείου, χωρίζουν δηλαδή το ευθύγραμμο τμήμα σε δύο ίσα μικρότερα τμήματα και πραγματοποιούν ελέγχους στο κάθε ένα από αυτά. Χαρακτηριστικός στην κατηγορία αυτή είναι ο Midpoint Subdivision.

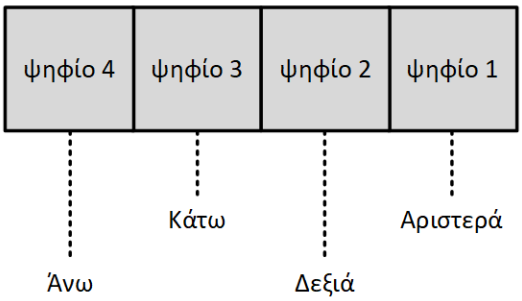
Πέρα από τις τρεις προαναφερθείσες κατηγορίες, υπάρχουν και άλλες προσεγγίσεις, όπως ο αλγόριθμος Wenjun Huang [HW09; Hua10] ο οποίος εφαρμόζει κηδεστικούς μετασχηματισμούς στα ευθύγραμμα τμήματα που δεν μπορεί να κατατάξει ως εξ ολοκλήρου εντός ή εξ ολοκλήρου εκτός της περιοχής αποκοπής μεταβάλλοντας την

κλίση τους ενώ ταυτόχρονα μεταβάλλει και την γεωμετρία της ίδιας της περιοχής αποκοπής. Ενδιαφέρουσα επίσης προσέγγιση αποτελεί και ο αλγόριθμος [ERA19] με τον οποίο παρουσιάζεται ένα μαθηματικό πρότυπο για την αξιολόγηση των σημείων τομής και συνεπώς την αποκοπή στην συνέχεια των ευθυγράμμων τμημάτων. Παρουσίαση και σύγκριση των αλγορίθμων αυτών μπορεί κανείς να συναντήσει στα [PJ13; SA16].

2.1 Αλγόριθμος Cohen-Sutherland

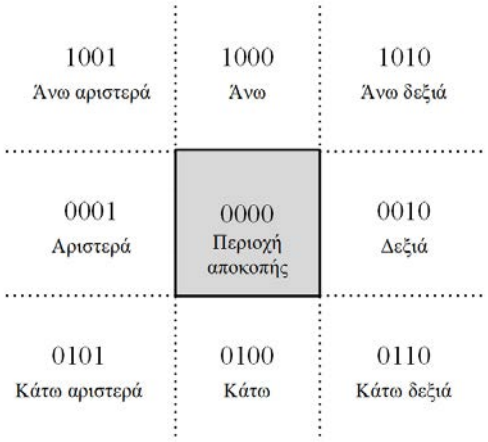
Ο αλγόριθμος των Danny Cohen και Ivan Sutherland (CS) αναπτύχθηκε το 1969 κατά τη διάρκεια δημιουργίας ενός προσομοιωτή πτήσεων και είναι ένας από τους πρώτους που εφαρμόζει ταχεία αποκοπή. Παραλλαγές αυτού του αλγορίθμου χρησιμοποιούνται ευρέως ακόμη και σήμερα. Ο χρόνος επεξεργασίας μειώνεται σημαντικά, διότι οι περισσότεροι έλεγχοι εκτελούνται πριν τον υπολογισμό των σημείων τομής.

Αρχικά, ο χώρος διαμερίζεται σε εννέα «περιοχές»: στην «περιοχή αποκοπής», η οποία βρίσκεται στο κέντρο, και σε οκτώ ακόμα οι οποίες βρίσκονται πέριξ αυτής. Σε κάθε μία αποδίδεται μία τετραψήφια δυαδική τιμή που ονομάζεται κωδικός περιοχής, βλ. Σχ. 2.3



Σχήμα 2.3: Οι κωδικοί περιοχών στον αλγόριθμο CS

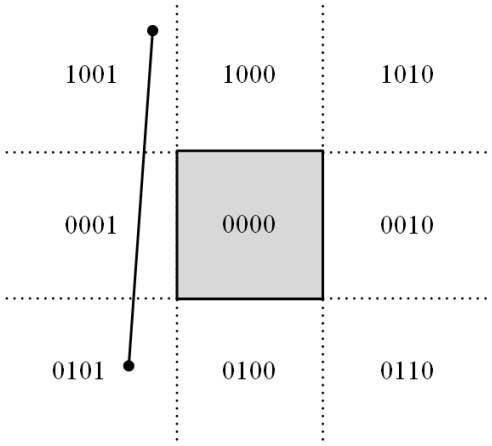
Το κάθε δυαδικό ψηφίο της τετραψήφιας δυαδικής τιμής αντιπροσωπεύει την θέση μίας περιοχής σε σχέση με την κεντρική περιοχή αποκοπής δηλαδή περιγράφει εάν είναι άνωθεν, κάτωθεν, αριστερά ή δεξιά αυτής. Το Σχήμα 2.4 απεικονίζει μία πιθανή σειρά για τις θέσεις των δυαδικών ψηφίων. Για αυτήν την σειρά, το Λιγότερο Σημαντικό Ψηφίο (δυαδικό ψηφίο 1) αναφέρεται στο αριστερό σύνορο της περιοχής αποκοπής, η αμέσως επόμενη θέση (δυαδικό ψηφίο 2) αναφέρεται στο δεξιό σύνορο, η επόμενη θέση (δυαδικό ψηφίο 3) αναφέρεται στο κάτω σύνορο και το Περισσότερο Σημαντικό Ψηφίο (δυαδικό ψηφίο 4) αναφέρεται στο άνω σύνορο.



Σχήμα 2.4: Το κάθε ψηφίο στον κωδικό περιοχής υποδεικνύει την θέση που βρίσκεται το σημείο σε σχέση με τα σύνορα της περιοχής αποκοπής

Οι κωδικοί των περιοχών βοηθούν τον αλγόριθμο να κατατάξει τα τελικά σημεία που προσδιορίζουν το προς αποκοπή ευθύγραμμο τμήμα σε ποια από τις εννέα περιοχές ανήκουν και να αποφασίσει γρηγορότερα, εάν θα χρειαστεί να εφαρμόσει αποκοπή ή όχι. Συνεπώς, ο αλγόριθμος αποδίδει στην αρχή, σε κάθε ένα εκ των δύο τελικών σημείων, τον κωδικό της περιοχής εντός της οποίας βρίσκεται. Αν υπάρχει η τιμή "1" (Αληθής) σε οποιοδήποτε ψηφίο του κωδικού, τότε αυτή υποδεικνύει ότι το σημείο βρίσκεται εκτός του αντίστοιχου συνόρου της περιοχής αποκοπής. Η τιμή "0" (Ψευδής) σε οποιοδήποτε ψηφίο του κωδικού υποδεικνύει ότι το σημείο βρίσκεται εντός ή επί του αντίστοιχου συνόρου της περιοχής αποκοπής. Για τον λόγο αυτό, ο κωδικός περιοχής αναφέρεται πολλές φορές και ως εξωκωδικός επειδή το "1" σε οποιοδήποτε ψηφίο υποδεικνύει ότι το σημείο βρίσκεται εκτός του αντίστοιχου συνόρου.

Το επόμενο βήμα του αλγορίθμου είναι να προσδιορίσει άμεσα ποια ευθύγραμμα τμήματα βρίσκονται εξ ολοκλήρου εντός της περιοχή αποκοπής και ποια εξ ολοκλήρου εκτός. Εκείνα που βρίσκονται εξ ολοκλήρου εντός έχουν ως κωδικό περιοχής τον 0000 και στα δύο τελικά τους σημεία οπότε σχεδιάζονται κανονικά. Εκείνα που βρίσκονται εξ ολοκλήρου εκτός έχουν το ψηφίο "1" στην ίδια θέση στα δύο τελικά τους σημεία οπότε απορρίπτονται. Για παράδειγμα, εάν ένα ευθύγραμμο τμήμα έχει κωδικό 1001 στο ένα τελικό σημείο και κωδικό 0101 στο άλλο, τότε σημαίνει ότι βρίσκεται αριστερά της περιοχής αποκοπής, όπως υποδεικνύεται από την τιμή "1" στη θέση του Λιγότερου Σημαντικού Ψηφίου κάθε κωδικού περιοχής (βλ. Σχ. 2.5).

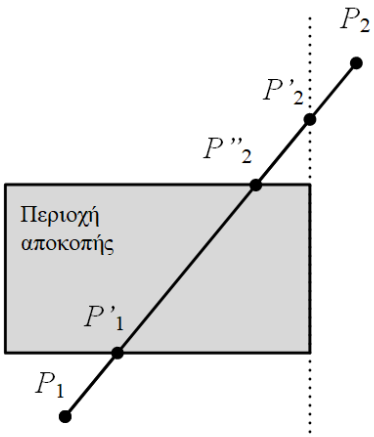


Σχήμα 2.5: Εξ αρχής απόρριψη του ευθυγράμμου τμήματος του οποίου τα τελικά σημεία έχουν κωδικούς περιοχής 1001 και 0101

Η χρήση λογικών τελεστών βοηθάει σημαντικά στην επιλογή αυτή. Όταν η λογική διάζευξη μεταξύ των δύο κωδικών περιοχής είναι Ψευδής (0000), τότε το ευθύγραμμο τμήμα βρίσκεται εντός της περιοχής αποκοπής, σχεδιάζεται εξ ολοκλήρου και ο αλγόριθμος συνεχίζει με τον έλεγχο του επόμενου ευθυγράμμου τμήματος της σκηνής. Όταν η λογική σύζευξη μεταξύ των δύο κωδικών περιοχής είναι Αληθής (όχι 0000), τότε το ευθύγραμμο τμήμα βρίσκεται εξ ολοκλήρου εκτός της περιοχής αποκοπής και εξαλείφεται από την σκηνή. Συνοπτικά, οι έλεγχοι αυτοί είναι:

- (Κωδικός περιοχής 1) Ή (Κωδικός περιοχής 2) = 0000: Το ευθύγραμμο τμήμα βρίσκεται εξ ολοκλήρου εντός.
- (Κωδικός περιοχής 1) ΚΑΙ (Κωδικός περιοχής 2) $\neq$ 0000: Το ευθύγραμμο τμήμα βρίσκεται εξ ολοκλήρου εκτός.

Τα ευθύγραμμα τμήματα που δεν μπορούν να προσδιοριστούν ότι βρίσκονται εξ ολοκλήρου εντός ή εξ ολοκλήρου εκτός, ελέγχονται στη συνέχεια προκειμένου να βρεθούν τα σημεία τομής τους με την περιοχή αποκοπής. Όπως φαίνεται στο Σχ. 2.6, μπορεί να τέμνουν τα σύνορά της χωρίς να εισέρχονται στο εσωτερικό της και, κατά συνέπεια, να χρειαστούν αρκετοί υπολογισμοί για να βρεθούν τα σημεία τομής αναλόγως της σειράς με την οποία τα επεξεργαζόμαστε.



Σχήμα 2.6: Ορισμένες φορές χρειάζεται να πραγματοποιηθούν διαδοχικοί υπολογισμοί για να βρεθούν τα σημεία τομής του ευθυγράμμου τμήματος με την περιοχή αποκοπής

Κατά την επεξεργασία με κάποιο από τα σύνορα αποκοπής, ένα μέρος του ευθυγράμμου τμήματος αποκόπτεται και το υπόλοιπο ελέγχεται με τα υπόλοιπα σύνορα. Η εξάλειψη των τμημάτων συνεχίζεται έως ότου είτε το ευθύγραμμο τμήμα αποκοπεί πλήρως ή το υπόλοιπο μέρος του να βρίσκεται εξ ολοκλήρου εντός της περιοχής αποκοπής. Αν υποθέσουμε ότι τα σύνορα της περιοχής αποκοπής επεξεργάζονται με την ακόλουθη σειρά: αριστερά, δεξιά, κάτω, άνω, τότε για να προσδιορίσουμε εάν το ευθύγραμμο τμήμα διασχίζει ένα συγκεκριμένο σύνορο μπορούμε να ελέγξουμε τις αντίστοιχες τιμές ψηφίων στους δύο κωδικούς περιοχής των τελικών σημείων του. Εάν μία από αυτές τις τιμές ψηφίου είναι "1" και η άλλη είναι "0" τότε το ευθύγραμμο τμήμα διασχίζει το σύνορο αυτό.

Για το ευθύγραμμο τμήμα $P_1 - P_2$ του Σχ. 2.6, ο αλγόριθμος δεν μπορεί να αναγνωρίσει άμεσα εάν αυτό βρίσκεται εξ ολοκλήρου εντός η εξ ολοκλήρου εκτός, αφού οι κωδικοί περιοχής των τελικών σημείων που το προσδιορίζουν είναι 0100 και 1010 αντίστοιχα. Η διαδικασία εύρεσης των σημείων τομής του ευθυγράμμου τμήματος με την περιοχή αποκοπής που θα ακολουθήσει θα ξεκινήσει με το αριστερό σύνορο. Και τα δύο σημεία βρίσκονται εντός του συγκεκριμένου συνόρου οπότε καμία αλλαγή δεν γίνεται στο ευθύγραμμο τμήμα ως προς αυτό. Στη συνέχεια, ελέγχεται το δεξί σύνορο όπου το σημείο P_1 είναι εντός αυτού αλλά το σημείο P_2 είναι εκτός άρα πρέπει να υπολογισθεί το σημείο τομής τους. Αν υποθέσουμε ότι το σημείο τομής είναι το P'_2, τότε το τμήμα μεταξύ των σημείων P_2 και P'_2 αποκόπτεται και το προκύπτον ευθύγραμμο τμήμα $P_1 - P'_2$ χρησιμοποιείται στη συνέχεια. Στο επόμενο βήμα, εξετάζεται το κάτω σύνορο και διαπιστώνεται ότι το σημείο P'_2 είναι εντός αυτού αλλά το σημείο P_1 είναι εκτός οπότε θα χρειαστεί και σε αυτή τη φάση να υπολογιστεί το νέο σημείο τομής

με αυτό, έστω P_1'. Το τμήμα μεταξύ των σημείων P_1 και P_1' αποκόπτεται και το προκύπτον ευθύγραμμο τμήμα $P_1' - P_2'$ χρησιμοποιείται στη συνέχεια. Το τελευταίο σύνορο που ελέγχεται είναι το επάνω, όπου διαπιστώνεται ότι το P_1' είναι εντός αλλά το P_2' είναι εκτός και πρέπει να υπολογιστεί το νέο σημείο τομής, έστω P_2''. Το τμήμα μεταξύ των σημείων P_2'' και P_2' αποκόπτεται. Εφ' όσον οι έλεγχοι και των τεσσάρων συνόρων έχουν ολοκληρωθεί, σχεδιάζεται τελικώς το ευθύγραμμο τμήμα που προσδιορίζεται από τα σημεία P_1' και P_2''.

Ο προσδιορισμός των σημείων τομής του ευθυγράμμου τμήματος με την περιοχή αποκοπής γίνεται μέσω της εξίσωσης της ευθείας. Αν (x_0, y_0) και (x_1, y_1) είναι δύο δεδομένα σημεία αυτής, η συντεταγμένη y σε σχέση με την συντεταγμένη x ενός τυχαίου σημείου, που ανήκει κι αυτό στην ευθεία, υπολογίζεται από την εξίσωση:

$$y = y_0 + m \cdot (x - x_0)$$

όπου

$$m = \frac{y_1 - y_0}{x_1 - x_0}$$

είναι η κλίση της ευθείας. Η εξίσωση μπορεί να γραφεί και ως:

$$x = x_0 + \frac{(y - y_0)}{m}.$$

Για τον υπολογισμό των συντεταγμένων των σημείων τομής, στην πρώτη μορφή, το x λαμβάνει είτε την τιμή x_{min} είτε την τιμή x_{max} ενώ στη δεύτερη, το y λαμβάνει είτε την τιμή y_{min} είτε την τιμή y_{max}.

Τα βήματα του αλγορίθμου συνοπτικά είναι:

1. Για κάθε τελικό σημείο του ευθυγράμμου τμήματος υπολογίζεται ο κωδικός περιοχής.
2. Εφαρμόζεται λογική διάζευξη μεταξύ των δύο κωδικών περιοχής. Αν το αποτέλεσμα είναι 0000, τότε το ευθύγραμμο τμήμα βρίσκεται εξ ολοκλήρου εντός της περιοχής αποκοπής, οπότε σχεδιάζεται ως έχει και ο αλγόριθμος τερματίζει διαφορετικά συνεχίζει στο επόμενο βήμα.
3. Εφαρμόζεται λογική σύζευξη μεταξύ των δύο κωδικών περιοχής. Αν το αποτέλεσμα δεν είναι 0000, τότε το ευθύγραμμο τμήμα βρίσκεται εξ ολοκλήρου εκτός της περιοχής αποκοπής οπότε απορρίπτεται και ο αλγόριθμος τερματίζει διαφορετικά συνεχίζει στο επόμενο βήμα.

4. Οι κωδικοί περιοχής χρησιμοποιούνται για να υπολογιστούν τα σημεία τομής του ευθυγράμμου τμήματος με την περιοχή αποκοπής.

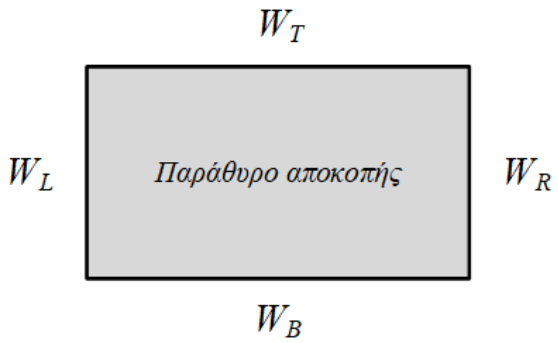
Η υλοποίηση του αλγορίθμου βρίσκεται στο Παράρτημα A - Κώδικας A.1.

2.2 Αλγόριθμος Cyrus-Beck

Ο αλγόριθμος των Mike Cyrus και Jay Beck (CB) [CB78] δημιουργήθηκε το 1978 και βασίζεται στην παραμετρική εξίσωση της ευθείας:

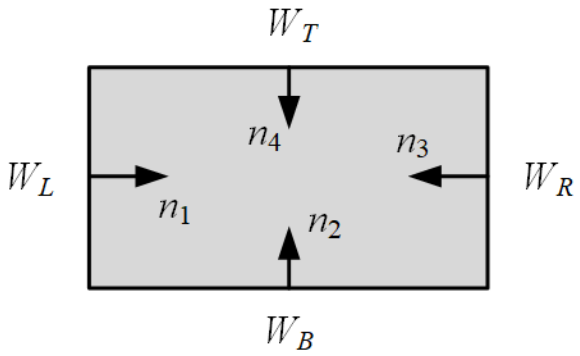
$$P(t) = P_0 + t \cdot (P_1 - P_0)$$

Ο αλγόριθμος μπορεί να εφαρμοστεί σε ένα τυπικό ορθογώνιο παράθυρο αποκοπής (βλ. Σχ. 2.7) αλλά και σε οποιαδήποτε άλλο με μορφή κυρτού πολυγώνου, σε αντίθεση με τον Cohen-Sutherland ο οποίος εφαρμόζεται μόνο σε παράθυρα ορθογωνίου σχήματος. Το πλήθος των ακμών (πλευρών) του παραθύρου δεν αλλάζει τον τρόπο λειτουργίας του αλγορίθμου αλλά επηρεάζει την απόδοσή του.



Σχήμα 2.7: Τυπικό ορθογώνιο παράθυρο αποκοπής που χρησιμοποιεί ο αλγόριθμος CB

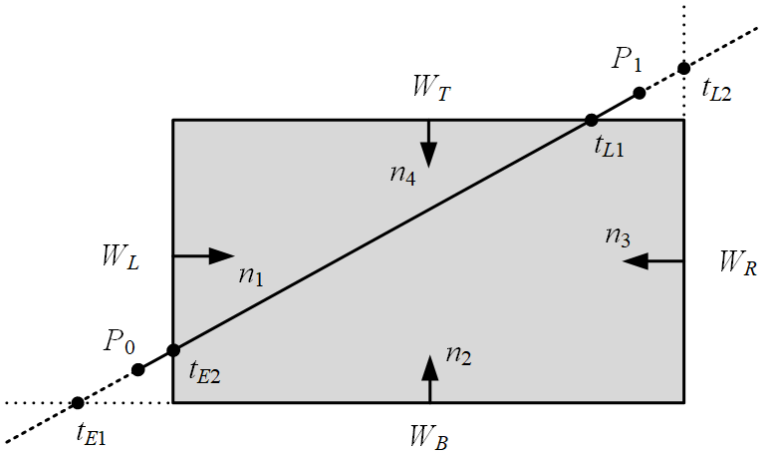
Για οποιοδήποτε παράθυρο αποκοπής κυρτού σχήματος, ο αλγόριθμος υπολογίζει τα εισερχόμενα φυσιολογικά διανύσματα των ακμών του. Τα διανύσματα αυτά είναι κάθετα ως προς τις ακμές του παραθύρου και έχουν φορά προς το κέντρο του. Σε ένα τυπικό ορθογώνιο παράθυρο αποκοπής υπάρχουν τέσσερα τέτοια διανύσματα και οποιαδήποτε άλλα φυσιολογικά διανύσματα είναι ισοδύναμά τους (βλ. Σχ. 2.8). Με τη βοήθεια των φυσιολογικών διανυσμάτων υπολογίζονται τα σημεία τομής του προς αποκοπή ευθυγράμμου τμήματος με το παράθυρο αποκοπής. Το πλήθος των σημείων τομής είναι ίσο με το πλήθος των ακμών του παραθύρου.



Σχήμα 2.8: Τα εισερχόμενα φυσιολογικά διανύσματα των ακμών του παραθύρου αποκοπής

Ο αλγόριθμος υπολογίζει σταδιακά όλα τα σημεία τομής και τα κατηγοριοποιεί είτε ως Δυνητική Εισχώρηση (P_E) είτε ως Δυνητική Αποχώρηση (P_L) σε σχέση με το παράθυρο αποκοπής. Όταν ένα σημείο τομής έχει χαρακτηριστεί ως P_E σημαίνει ότι, αν υποθέσουμε ότι κινούμαστε επί του ευθυγράμμου τμήματος, το παράθυρο είναι «έμπροσθεν» και πρόκειται να «εισχωρήσουμε» σε αυτό. Ομοίως, όταν ένα σημείο τομής έχει χαρακτηριστεί ως P_L σημαίνει ότι, αν θεωρήσουμε ότι κινούμαστε επί του ευθυγράμμου τμήματος, το παράθυρο βρίσκεται «όπισθεν» και έχουμε «αποχωρήσει» από αυτό.

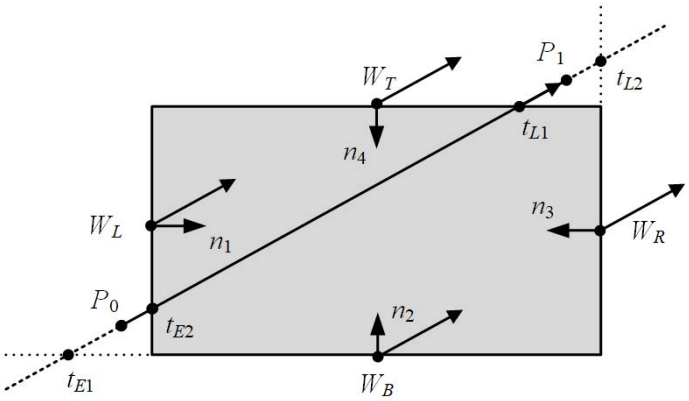
Επιπλέον των χαρακτηρισμών Δυνητική Εισχώρηση και Δυνητική Αποχώρηση, ο αλγόριθμος υπολογίζει τις τιμές της παραμέτρου t ανά περίπτωση (t_E ή t_L) από την εξίσωση της ευθείας και σχηματίζει δύο ομάδες: μία ομάδα με όλες τις t_E τιμές και μία ομάδα με όλες τις t_L (βλ. Σχ. 2.9). Οι τιμές του t που είναι είτε μικρότερες από το μηδέν είτε μεγαλύτερες από την μονάδα, απορρίπτονται.



Σχήμα 2.9: Ο αλγόριθμος σχηματίζει μία ομάδα με όλες τις τιμές t_E και μία ομάδα με όλες τις τιμές t_L

Η κατηγοριοποίηση του σημείου τομής και της αντίστοιχης παραμέτρου t σε Δυνητική Εισχώρηση ή Δυνητική

Αποχώρηση επιτυγχάνεται με τη βοήθεια του διανύσματος το οποίο σχηματίζεται από τα σημεία P_0 και P_1 τα οποία περιγράφουν το ευθύγραμμο τμήμα. Κάθε εισερχόμενο φυσιολογικό διάνυσμα συγκρίνεται με το διάνυσμα $\overrightarrow{P_0P_1}$ και αν η γωνία τους είναι μικρότερη από 90°, τότε το σημείο τομής είναι Δυνητική Εισχώρηση (P_E) ενώ αν είναι μεγαλύτερη από 90°, τότε το σημείο τομής είναι Δυνητική Αποχώρηση (P_L) (βλ. Σχ. 2.10).



Σχήμα 2.10: Η κατηγοριοποίηση του σημείου τομής και της αντίστοιχης παραμέτρου του σε Δυνητική Εισχώρηση ή Δυνητική Αποχώρηση επιτυγχάνεται με τη βοήθεια του διανύσματος $\overrightarrow{P_0P_1}$

Εφ' όσον το ευθύγραμμο τμήμα τέμνει το παράθυρο αποκοπής το πολύ σε δύο σημεία, ο αλγόριθμος επιλέγει μόνο μία τιμή t_E από όλες που έχει υπολογίσει και συγκεκριμένα την μέγιστη η οποία είναι η πλησιέστερη στα σύνορά του. Ομοίως, επιλέγει μόνο μία από όλες τις τιμές t_L που έχει υπολογίσει και συγκεκριμένα την ελάχιστη η οποία είναι επίσης η πλησιέστερη στα σύνορα.

Ο Cyrus–Beck είναι ένας γενικευμένος αλγόριθμος αποκοπής ευθυγράμμων τμημάτων που σχεδιάστηκε να είναι αποτελεσματικότερος από άλλους αλγόριθμους αποκοπής, όπως ο Cohen–Sutherland. Χρησιμοποιεί επαναλαμβανόμενη αποκοπή, είναι πολύ σταθερός και η απόδοσή του είναι σχεδόν ανεξάρτητη από παράγοντες όπως η γεωμετρική κατανομή των αποκομμένων τμημάτων [Ska05]. Εφ' όσον ο αλγόριθμος υπολογίζει το σημείο τομής για κάθε ακμή, η πολυπλοκότητά του είναι $O(N)$, όπου N το πλήθος των ακμών. Στις περισσότερες περιπτώσεις, αποκόπτει μόνο μία ή δύο φορές σε αντίθεση με το Cohen–Sutherland όπου τα ευθύγραμμα τμήματα αποκόπτονται περίπου τέσσερις φορές μέχρι το τελικό αποτέλεσμα. Ο αλγόριθμος μπορεί εύκολα να τροποποιηθεί και να εφαρμοστεί στις τρεις διαστάσεις.

Τα βήματα του αλγορίθμου συνοπτικά είναι:

1. Θεώρησε το παράθυρο αποκοπής ως ένα κυρτό πολύγωνο. Κάθε ακμή του παραθύρου αντιπροσωπεύεται από ένα ζεύγος κορυφών.

2. Όρισε τα δύο σημεία που προσδιορίζουν το προς αποκοπή ευθύγραμμο τμήμα.

3. Υπολόγισε το διάνυσμα κατεύθυνσης του ευθυγράμμου τμήματος. Το διάνυσμα αυτό προκύπτει αφαιρώντας τις συντεταγμένες του σημείου εκκίνησης από τις συντεταγμένες του σημείου τερματισμού.

4. Υπολόγισε τις παραμετρικές τιμές (τιμές t) της εξίσωσης του ευθυγράμμου τμήματος για κάθε σημείο τομής του με το παράθυρο αποκοπής.

$$t_E = -\frac{\overrightarrow{N} \cdot (\overrightarrow{v_1} - \overrightarrow{v_0})}{\overrightarrow{N} \cdot \overrightarrow{D}} \qquad t_L = -\frac{\overrightarrow{N} \cdot (\overrightarrow{v_0} - \overrightarrow{v_1})}{\overrightarrow{N} \cdot \overrightarrow{D}}$$

   όπου $\overrightarrow{N}$ είναι το εισερχόμενο φυσιολογικό διάνυσμα της ακμής παραθύρου, v_0 και v_1 είναι δύο κορυφές της ακμής του παραθύρου, $\overrightarrow{D}$ είναι το διάνυσμα κατεύθυνσης του ευθυγράμμου τμήματος.

5. Έλεγξε τις παραμετρικές τιμές t που υπολογίστηκαν στο προηγούμενο βήμα προκειμένου να προσδιοριστεί, εάν το προς αποκοπή ευθύγραμμο τμήμα βρίσκεται εντός ή εκτός του παραθύρου αποκοπής. Εάν $t_E > t_L$, τότε το τμήμα αυτό είναι εξ ολοκλήρου εκτός και ο αλγόριθμος τερματίζεται.

6. Απόκοψε το ευθύγραμμο τμήμα χρησιμοποιώντας τις παραμετρικές τιμές που υπολογίστηκαν στο βήμα 4. Το αποκομμένο τμήμα προκύπτει λαμβάνοντας υπόψη τα σημεία τομής εντός του παραμετρικού εύρους.

$$P'_0 = \overrightarrow{P_0} + t_E \cdot \overrightarrow{D} \qquad P'_1 = \overrightarrow{P_0} + t_L \cdot \overrightarrow{D}$$

   Εάν και οι δύο τιμές, t_E και t_L, βρίσκονται στο εύρος $[0, 1]$, τότε το ευθύγραμμο τμήμα βρίσκεται μερικώς εντός του παραθύρου αποκοπής.

Η υλοποίηση του αλγορίθμου βρίσκεται στο Παράρτημα A - Κώδικας A.2.

2.3 Αλγόριθμος Liang-Barsky

Ο You-Dong Liang και ο Brian Barsky βασίστηκαν στον Cyrus-Beck προκειμένου να αναπτύξουν έναν ταχύτερο αλγόριθμο για την αποκοπή ενός ευθυγράμμου τμήματος. Οι βασικοί του στόχοι είναι η ταχύτητα και η αποφυγή περιττών ελέγχων διασφαλίζοντας έτσι ότι γίνονται μόνο οι απαραίτητοι υπολογισμοί [LB84].

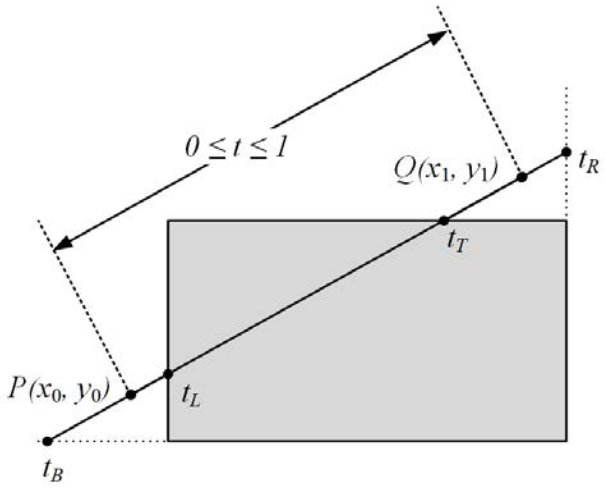
Όπως έχουμε ήδη αναφέρει, για ένα ευθύγραμμο τμήμα το οποίο περιγράφεται από τα σημεία $P(x_0, y_0)$ και

$Q(x_1, y_1)$, οι συντεταγμένες ενός τυχόντος σημείου (x, y) βάσει της παραμετρικής του μορφής είναι:

$$x = x_0 + t \cdot (x_1 - x_0)$$
$$y = y_0 + t \cdot (y_1 - y_0) \quad , \quad 0 \leq t \leq 1$$

Αν θέσουμε $\Delta x = (x_1 - x_0)$ και $\Delta y = (y_1 - y_0)$, τότε οι προηγούμενες εξισώσεις παίρνουν την ακόλουθη μορφή (βλ. Σχ. 2.11):

$$x = x_0 + t \cdot \Delta x$$
$$y = y_0 + t \cdot \Delta y$$



Σχήμα 2.11: Η παράμετρος t του ευθυγράμμου τμήματος λαμβάνει τιμές μεταξύ 0 και 1. Για κάθε σημείο τομής υπολογίζεται και η αντίστοιχη τιμή του t (t_T, t_B, t_L, t_R)

Όταν το ευθύγραμμο τμήμα αποκοπεί, τότε οι νέες συντεταγμένες των σημείων που το προσδιορίζουν βρίσκονται εντός του παραθύρου αποκοπής. Αυτό σημαίνει ότι οι προηγούμενες εξισώσεις μπορούν να γραφούν και ως εξής:

$$x_{\text{min}} \leq x_0 + t \cdot \Delta x \leq x_{\text{max}}$$
$$y_{\text{min}} \leq y_0 + t \cdot \Delta y \leq y_{\text{max}}$$

Οι ανισότητες αυτές δύναται να αναλυθούν ακόμα περισσότερο:

$$\begin{aligned} -t \cdot \Delta x &\leq x_0 - x_{\text{min}} \\ t \cdot \Delta x &\leq x_{\text{max}} - x_0 \\ -t \cdot \Delta y &\leq y_0 - y_{\text{min}} \\ t \cdot \Delta y &\leq y_{\text{max}} - y_0 \end{aligned}$$

η απλούστερα:

$$t \cdot p_i \leq q_i \quad , \quad i : 0, 1, 2, 3$$

όπου $i : 0, 1, 2, 3$ αντίστοιχα τα σύνορα αριστερά, δεξιά, κάτω και άνω. Οι παράμετροι p_i και q_i ισοδυναμούν με:

$$\begin{aligned} p_0 &= -\Delta x, & q_0 &= x_0 - x_{\text{min}} \\ p_1 &= \Delta x, & q_1 &= x_{\text{max}} - x_0 \\ p_2 &= -\Delta y, & q_2 &= y_0 - y_{\text{min}} \\ p_3 &= \Delta y, & q_3 &= y_{\text{min}} - y_0 \end{aligned}$$

Από τα παραπάνω μπορούμε να βγάλουμε τα εξής συμπεράσματα:

1. Εάν $p_i = 0$, τότε το ευθύγραμμο τμήμα είναι παράλληλο με το αντίστοιχο σύνορο του παραθύρου αποκοπής.
2. Εάν $p_i = 0$ και $q_i < 0$, τότε το ευθύγραμμο τμήμα είναι εξ ολοκλήρου εκτός του παραθύρου αποκοπής και απορρίπτεται.
3. Εάν $p_i \neq 0$, τότε ο λόγος $r_i = \frac{q_i}{p_i}$ δίνει μία τιμή της παραμέτρου t για το σημείο τομής του ευθυγράμμου τμήματος και του συνόρου i του παραθύρου αποκοπής.
4. Εάν $p_i < 0$, τότε το ευθύγραμμο τμήμα εισέρχεται στο σύνορο i του παραθύρου αποκοπής.
5. Εάν $p_i > 0$, τότε το ευθύγραμμο τμήμα εξέρχεται από το σύνορο i του παραθύρου αποκοπής.

Υπάρχουν δύο (από τα τέσσερα) σημεία τομής με το παράθυρο αποκοπής και έχουν τιμές t_0 και t_1. Για το t_0, ο αλγόριθμος υπολογίζει όλες τις τιμές της παραμέτρου t για τις οποίες ισχύει ότι $p_i < 0$ (δηλαδή το ευθύγραμμο

τμήμα εισέρχεται στο σύνορο i του παραθύρου αποκοπής) και του αποδίδει την μέγιστη τιμή. Για το t_1, ο αλγόριθμος υπολογίζει όλες τις τιμές της παραμέτρου t για τις οποίες ισχύει ότι $p_i > 0$ (δηλαδή το ευθύγραμμο τμήμα εξέρχεται από το σύνορο i του παραθύρου αποκοπής) και του αποδίδει την ελάχιστη τιμή. Εάν $t_0 > t_1$, τότε το ευθύγραμμο τμήμα είναι εξ ολοκλήρου εκτός του παραθύρου αποκοπής και απορρίπτεται. Σε αντίθετη περίπτωση τα νέα σημεία υπολογίζονται από τις τιμές t_0 και t_1 και μεταξύ αυτών σχεδιάζεται το αποκομμένο ευθύγραμμο τμήμα. Ο υπολογισμός των t_0 και t_1 περιγράφεται από τις επόμενες μαθηματικές εκφράσεις:

$$t_0 = \max\left(\frac{q_i}{p_i} \quad \middle| \quad p_i < 0, \quad i : 0,1,2,3 \cup \{0\}\right),$$
$$t_1 = \min\left(\frac{q_i}{p_i} \quad \middle| \quad p_i > 0, \quad i : 0,1,2,3 \cup \{1\}\right).$$

Τα σύνολα $\{0\}$ και $\{1\}$ προστίθενται έτσι ώστε η αρχική και η τελική τιμή της παραμέτρου t να περιοριστούν στα άκρα του ευθυγράμμου τμήματος.

Γενικά, ο αλγόριθμος Liang–Barsky είναι αποτελεσματικότερος σε σχέση με τους Cohen–Sutherland και Cyrus–Beck. Χρησιμοποιεί υπολογισμούς κινητής υποδιαστολής για την εύρεση των κατάλληλων τελικών σημείων με το πολύ τέσσερις υπολογισμούς [Nis17]. Αντίθετα, ο Cohen-Sutherland υπολογίζει τα σημεία τομής κατ' επανάληψη, ακόμα και αν το ευθύγραμμο τμήμα ευρίσκεται εξ ολοκλήρου εκτός του παραθύρου αποκοπής. Επιπλέον, ο υπολογισμός των σημείων τομής στον αλγόριθμο Cohen–Sutherland απαιτεί υπολογισμούς που εμπεριέχουν διαιρέσεις και πολλαπλασιασμούς, κάτι που τον καθιστά πιο αργό σε σχέση με τον Liang-Barsky.

Τα βήματα του αλγορίθμου συνοπτικά είναι:

1. Όρισε το παράθυρο αποκοπής χρησιμοποιώντας τις συντεταγμένες $(x_{\text{min}}, y_{\text{min}})$ και $(x_{\text{max}}, y_{\text{max}})$.
2. Έστω ότι το προς αποκοπή ευθύγραμμο τμήμα αναπαριστάται με τα σημεία (x_0, y_0) και (x_1, y_1). Υπολόγισε τις παραμέτρους p_0, p_1, p_2, p_3 της εξίσωσης της γραμμής στην οποία αυτό ανήκει.
3. Ομοίως, υπολόγισε τις παραμέτρους q_0, q_1, q_2, q_3.
4. Αρχικοποίησε τις παραμέτρους t_0 και t_1 με τις τιμές 0 και 1 αντίστοιχα.
5. Αν $p_i = 0$ και $q_i < 0$ όπου $i : 0,1,2,3$, τότε το ευθύγραμμο τμήμα απορρίπτεται.
6. Αν $p_i \neq 0$, υπολόγισε τους λόγους $\frac{q_i}{p_i}$ όπου $i : 0,1,2,3$.

7. Υπολόγισε τις δύο παραμέτρους t_0 και t_1:

$$t_0 = \max\left(\frac{q_i}{p_i} \quad | \quad p_i < 0, \quad i : 0,1,2,3 \cup \{0\}\right),$$
$$t_1 = \min\left(\frac{q_i}{p_i} \quad | \quad p_i > 0, \quad i : 0,1,2,3 \cup \{1\}\right)$$

8. Υπολόγισε τις συντεταγμένες (x_a, y_a) και (x_b, y_b) του αποκομμένου ευθυγράμμου τμήματος από τις εξισώσεις:

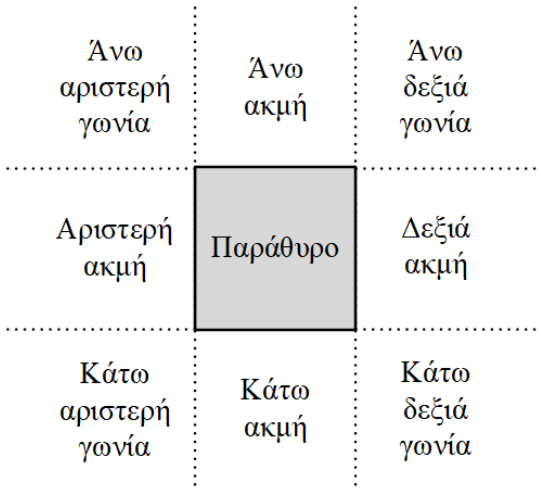
$$x_a = x_0 + t_0 \cdot (x_1 - x_0) \qquad y_a = y_0 + t_0 \cdot (y_1 - y_0)$$
$$x_b = x_0 + t_1 \cdot (x_1 - x_0) \qquad y_b = y_0 + t_1 \cdot (y_1 - y_0)$$

Η υλοποίηση του αλγορίθμου βρίσκεται στο Παράρτημα A - Κώδικας A.3.

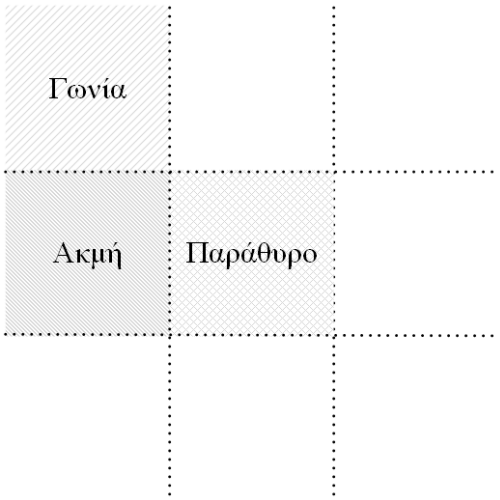
2.4 Αλγόριθμος Nicholl-Lee-Nicholl

Ο αλγόριθμος Nicholl–Lee–Nicholl (NLN) δημιουργήθηκε το 1987 από τους Tina M. Nicholl, D.T. Lee και Robin A. Nicholl. Το κύριο χαρακτηριστικό του είναι ότι προσπαθεί να περιορίσει το πλήθος των συνόρων του παραθύρου αποκοπής που θα ελεγχθούν σε σχέση με το ευθύγραμμο τμήμα έτσι ώστε να πραγματοποιήσει λιγότερους υπολογισμούς για την εύρεση των σημείων τομής. Μάλιστα, οι δημιουργοί του ισχυρίζονται πως η απόδοσή του είναι καλύτερη από άλλους αλγόριθμους, όπως ο Cohen–Sutherland και ο Liang–Barsky, καθώς εκτελεί λιγότερες συγκρίσεις και διαιρέσεις. Σε σχέση όμως με τους συγκεκριμένους, η αποκοπή με τον NLN περιορίζεται μόνο στις δύο διαστάσεις.

Όπως έχει ήδη προαναφερθεί, στον αλγόριθμο Cohen–Sutherland ο διδιάστατος χώρος διαμερίζεται σε 9 περιοχές: στην κεντρική περιοχή (παράθυρο αποκοπής) και σε οκτώ ακόμα οι οποίες βρίσκονται πέριξ αυτής. Για την αποφυγή περιττών ελέγχων και υπολογισμών, ο NLN υιοθετεί μία παρόμοια λογική με τον CS (βλ. Σχ. 2.12) αλλά χρησιμοποιεί μόνο τις τρεις από τις εννέα περιοχές: την άνω αριστερή γωνία την οποία ονομάζει «Γωνία», την αριστερή ακμή την οποία ονομάζει «Ακμή» και το παράθυρο αποκοπής το οποίο ονομάζει «Παράθυρο» (βλ. Σχ. 2.13).

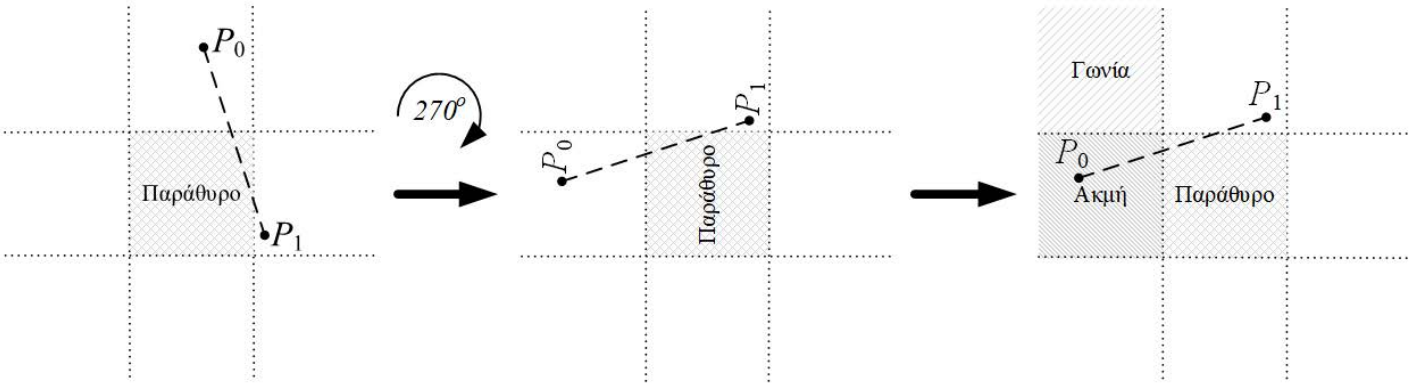


Σχήμα 2.12: Ο NLN υιοθετεί παρόμοια λογική με τον CS όσον αφορά τις περιοχές του διδιάστατου χώρου



Σχήμα 2.13: Ο NLN χρησιμοποιεί μόνο τρεις από τις εννέα περιοχές τις οποίες ονομάζει «Γωνία», «Ακμή» και «Παράθυρο»

Ο αλγόριθμος θεωρεί ότι το πρώτο σημείο του ευθυγράμμου τμήματος πρέπει να βρίσκεται σε μία από τις τρεις βασικές περιοχές. Εάν δεν συμβαίνει κάτι τέτοιο, τότε πρέπει να μετατοπιστεί σε μία από αυτές χρησιμοποιώντας γεωμετρικούς μετασχηματισμούς. Το δεύτερο σημείο λαμβάνεται υπ' όψη αργότερα. Για παράδειγμα, εάν ένα ευθύγραμμο τμήμα προσδιορίζεται από τα σημεία P_0, P_1 και το P_0 βρίσκεται άνωθεν της περιοχής του Παραθύρου, τότε το τελευταίο μπορεί να μεταφερθεί στην περιοχή της Ακμής χρησιμοποιώντας μία περιστροφή 270°, σύμφωνα με την φορά των δεικτών του ωρολογίου (βλ. Σχ. 2.14).



Σχήμα 2.14: Παράδειγμα μετατόπισης του σημείου P_0 στην Ακμή μετά από περιστροφή 270°

Όλοι οι διαθέσιμοι γεωμετρικοί μετασχηματισμοί που χρησιμοποιούνται από τον αλγόριθμο είναι οι ακόλουθοι:

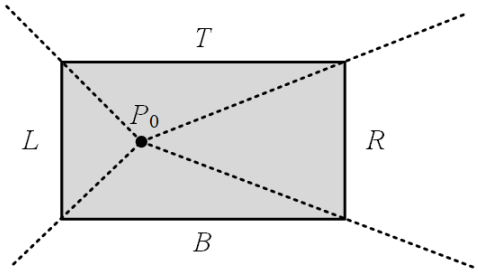
1. Περιστροφή 90° από την αρχή των αξόνων, σύμφωνα με την φορά των δεικτών του ωρολογίου.
2. Περιστροφή 180° από την αρχή των αξόνων, σύμφωνα με την φορά των δεικτών του ωρολογίου.
3. Περιστροφή 270° από την αρχή των αξόνων, σύμφωνα με την φορά των δεικτών του ωρολογίου.
4. Ανάκλαση της ευθείας $x = -y$.
5. Ανάκλαση γύρω από τον οριζόντιο άξονα των X.

Προφανώς, εκτός από τα σημεία που προσδιορίζουν το ευθύγραμμο τμήμα, οι γεωμετρικοί μετασχηματισμοί πρέπει να εφαρμόζονται και στα σύνορα του Παραθύρου.

Υποθέτοντας ότι τα P_0 και P_1 δεν βρίσκονται ταυτόχρονα εντός του Παραθύρου μετά τους μετασχηματισμούς, ο αλγόριθμος διαμερίζει ξανά τον χώρο σε περιοχές. Οι νέες αυτές περιοχές σχετίζονται με τη θέση του πρώτου σημείου του ευθυγράμμου τμήματος (P_0). Τα σύνορα όμως των νέων περιοχών είναι ευθύγραμμα τμήματα που ξεκινούν από την θέση P_0 και διέρχονται από κάθε γωνία του Παραθύρου. Βάσει αυτού, διακρίνονται τρεις κύριες περιπτώσεις:

1. *Το P_0 βρίσκεται εντός του Παραθύρου και το P_1 εκτός:*

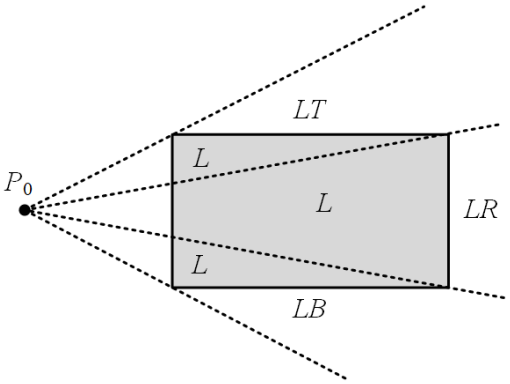
   Ο αλγόριθμος ορίζει τέσσερις περιοχές με ονόματα L, T, R, B, όπως στο Σχήμα 2.15. Στη συνέχεια, ανάλογα με το ποια από τις τέσσερις περιοχές περιέχει το P_1, υπολογίζει το σημείο τομής του ευθυγράμμου τμήματος με το αντίστοιχο σύνορο του Παραθύρου.



Σχήμα 2.15: Οι τέσσερις νέες περιοχές, με ονόματα L, T, R, B, που δημιουργούνται εάν το σημείο P_0 είναι εντός και το P_1 εκτός του Παραθύρου

2. *Το P_0 βρίσκεται στην περιοχή της Ακμής:*

   Ο αλγόριθμος δημιουργεί τέσσερις περιοχές με ονόματα L, LT, LR, LB, όπως στο Σχήμα 2.16. Αυτές οι τέσσερις περιοχές καθορίζουν ένα μοναδικό σύνορο του Παραθύρου σε σχέση με το ευθύγραμμο τμήμα και την θέση του P_1. Για παράδειγμα, εάν το P_1 βρίσκεται σε οποιαδήποτε από τις τρεις περιοχές με την ένδειξη L, ο αλγόριθμος αποκόπτει χρησιμοποιώντας μόνο το αριστερό σύνορο του Παραθύρου και σχεδιάζει το ευθύγραμμο τμήμα από το σημείο τομής έως το P_1. Εάν το P_1 βρίσκεται στην περιοχή LT, σχεδιάζει το ευθύγραμμο τμήμα από το αριστερό σύνορο του Παραθύρου έως το άνω σύνορο. Ομοίως, η ίδια λογική ισχύει για τις περιοχές LR και LB. Ωστόσο, εάν το P_1 δεν βρίσκεται σε καμία από αυτές τις τέσσερις περιοχές, το ευθύγραμμο τμήμα απορρίπτεται.

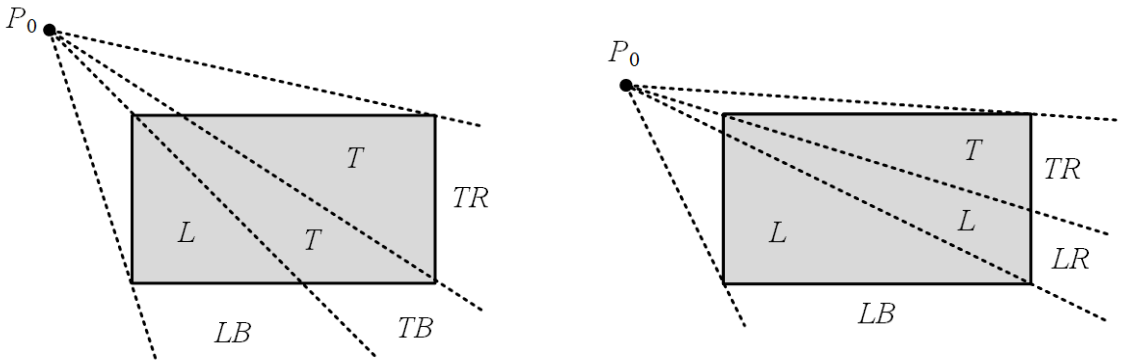


Σχήμα 2.16: Οι τέσσερις περιοχές αποκοπής, όταν το σημείο P_0 είναι στην Ακμή

3. *Το P_0 βρίσκεται στη περιοχή της Γωνίας:*

   Όταν το P_0 βρίσκεται στην περιοχή της Γωνίας ο αλγόριθμος χρησιμοποιεί ένα από τα δύο σύνολα όπως φαίνεται στο Σχήμα 2.17. Η επιλογή του (α) ή του (β) συνόλου εξαρτάται από τη θέση του P_0 εντός της Γωνίας. Όταν το P_0 είναι πιο κοντά στο αριστερό σύνορο του Παραθύρου, ο αλγόριθμος χρησιμοποιεί το

σύνολο των περιοχών της περίπτωσης (α) αλλά, όταν το P_0 είναι πιο κοντά στο άνω σύνορο του Παραθύρου, τότε χρησιμοποιεί τις περιοχές της περίπτωσης (β). Εάν το P_1 βρίσκεται σε μία από τις περιοχές T, L, TR, TB, LR ή LB, αυτό καθορίζει ένα μοναδικό σύνορο αποκοπής για τον υπολογισμό των σημείων τομής διαφορετικά ολόκληρο το ευθύγραμμο τμήμα απορρίπτεται.



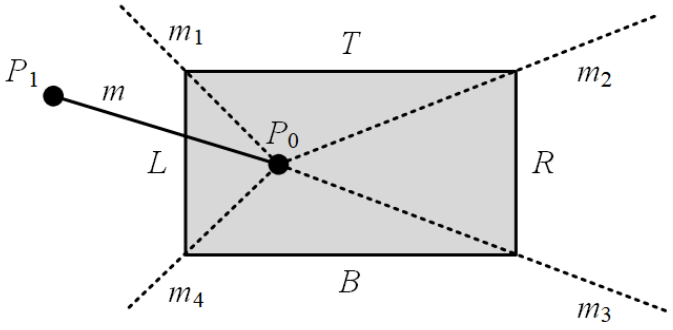
(α) Το P_0 κοντά στο αριστερό σύνορο

(β) Το P_0 κοντά στο άνω σύνορο

Σχήμα 2.17: Τα δύο σύνολα των περιοχών αποκοπής, όταν το σημείο P_0 είναι στην Γωνία

Για να προσδιορίσει την περιοχή στην οποία βρίσκεται το P_1, ο αλγόριθμος συγκρίνει την κλίση του ευθυγράμμου τμήματος με τις κλίσεις των νέων συνόρων που προκύπτουν από τους μετασχηματισμούς. Για παράδειγμα, εάν το P_0 είναι εντός του Παραθύρου, το P_1 είναι εκτός, το m είναι η κλίση του ευθυγράμμου τμήματος $P_0 - P_1$ και m_1, m_2, m_3, m_4 είναι οι κλίσεις των συνόρων L, T, R, B αντίστοιχα (βλ. Σχ. 2.18), τότε σύμφωνα με τις ακόλουθες συνθήκες το P_1 είναι:

- Αν $m_1 < m < m_2$, τότε το P_1 ευρίσκεται άνωθεν του Παραθύρου.
- Αν $m_2 < m < m_3$, τότε το P_1 ευρίσκεται δεξιά από το Παράθυρο.
- Αν $m_2 < m < m_3$, τότε το P_1 ευρίσκεται κάτωθεν του Παραθύρου.
- Αν $m_4 < m < m_1$, τότε το P_1 ευρίσκεται αριστερά από το Παράθυρο.



Σχήμα 2.18: Σύγκριση των κλίσεων για τον προσδιορισμό του P_1

Στο ίδιο παράδειγμα του Σχ. 2.18, εάν το P_1 ευρίσκεται αριστερά του Παραθύρου, τότε ο υπολογισμός της συντεταγμένης y του σημείου τομής του ευθυγράμμου τμήματος με το αριστερό σύνορο γίνεται ως εξής:

$$x = x_L$$
$$y = y_0 + m \cdot (x - x_0)$$

όπου

$$m = \frac{y_1 - y_0}{x_1 - x_0}$$

Ομοίως, η συντεταγμένη x:

$$y = y_L$$
$$x = x_0 + \frac{y - y_0}{m}.$$

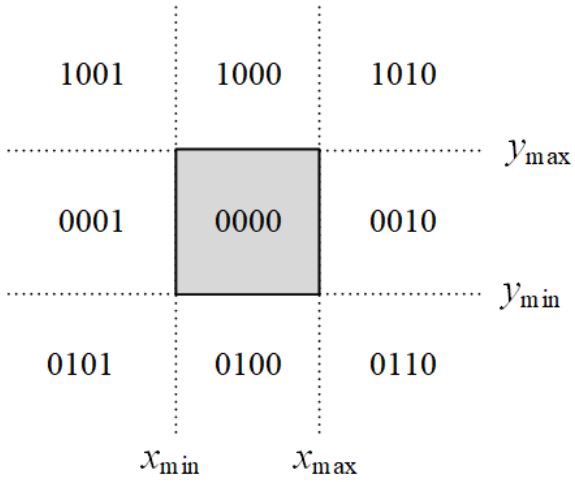
Η υλοποίηση του αλγορίθμου βρίσκεται στο Παράρτημα A - Κώδικας A.4.

2.5 Αλγόριθμος Υποδιαίρεσης μεσαίου σημείου

Ο αλγόριθμος της Υποδιαίρεσης μεσαίου σημείου (MS) είναι μία επέκταση του αλγορίθμου Cohen-Sutherland και ακολουθεί τη στρατηγική «Διαίρει και Βασίλευε». Χρησιμοποιείται κυρίως για τον υπολογισμό των ορατών μερών των ευθυγράμμων τμημάτων που υπάρχουν στο παράθυρο αποκοπής και ακολουθεί την αρχή της διχοτό-

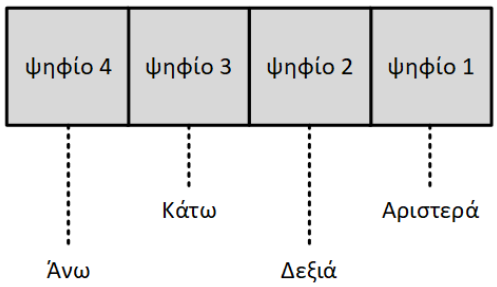
μησης του καθενός από αυτά σε ίσα μισά, πολλές φορές. Ο αλγόριθμος δεν είναι αποτελεσματικός εκτός και αν υλοποιηθεί μέσω ειδικού υπολογιστικού υλικού. Ενώ ο Cohen–Sutherland απαιτεί τον υπολογισμό των σημείων τομής του ευθυγράμμου τμήματος με το παράθυρο αποκοπής, στον Midpoint Subdivision οι υπολογισμοί αυτοί μπορούν να αποφευχθούν με την επαναλαμβανόμενη υποδιαίρεση του ευθυγράμμου τμήματος στο μισό του.

Αρχικά, ο αλγόριθμος κατηγοριοποιεί τα σημεία που προσδιορίζουν το ευθύγραμμο τμήμα και εκχωρεί στο καθένα έναν κωδικό περιοχής τεσσάρων δυαδικών ψηφίων, όπως κάνει ο Cohen-Sutherland. Ο κωδικός αυτός (αλλιώς εξωκωδικός) καθορίζεται με βάση την περιοχή που βρίσκεται το εκάστοτε τελικό σημείο (βλ. Σχ. 2.19).



Σχήμα 2.19: Οι περιοχές που χωρίζει ο αλγόριθμος τον χώρο και οι αντίστοιχοι κωδικοί τους

Ξεκινώντας από το Λιγότερο Σημαντικό Ψηφίο (ΛΣΨ) και μεταβαίνοντας προς το Περισσότερο Σημαντικό Ψηφίο (ΠΣΨ), κάθε δυαδικό ψηφίο αντιπροσωπεύει μία περιοχή: αριστερά, δεξιά, κάτω, άνω (βλ. Σχ. 2.20). Εάν ένα τελικό σημείο βρίσκεται εντός της συγκεκριμένης περιοχής, τότε το αντίστοιχο ψηφίο λαμβάνει την τιμή “1” (Αληθής) διαφορετικά την τιμή “0” (Ψευδής).



Σχήμα 2.20: Αρχίζοντας από το ΛΣΨ, τα δυαδικά ψηφία του εξωκωδικού αντιστοιχούν στην δεξιά, αριστερά, κάτω και άνω περιοχή

Υπάρχουν τρεις πιθανές περιπτώσεις για κάθε ευθύγραμμο τμήμα. Να είναι:

1. *Εξ ολοκλήρου ορατό*: Εάν οι κωδικοί περιοχής των δύο σημείων που προσδιορίζουν το ευθύγραμμο τμήμα είναι 0000, τότε αυτό βρίσκεται εντός της περιοχής αποκοπής, είναι πλήρως ορατό, οπότε σχεδιάζεται εξ ολοκλήρου.

2. *Εξ ολοκλήρου αόρατο*: Εάν το αποτέλεσμα της λογικής σύζευξης μεταξύ των κωδικών περιοχής των δύο σημείων που προσδιορίζουν το ευθύγραμμο τμήμα δεν είναι 0000, τότε αυτό βρίσκεται εξ ολοκλήρου εκτός της περιοχής αποκοπής οπότε απορρίπτεται χωρίς να σχεδιαστεί.

3. *Υποψήφιο προς αποκοπή*: Εάν το ευθύγραμμο τμήμα δεν ανήκει ούτε στην πρώτη, ούτε στην δεύτερη περίπτωση, τότε είναι μερικώς ορατό και πρέπει να υποδιαιρεθεί σε δύο ίσα μέρη. Κατόπιν, οι έλεγχοι ορατότητας εφαρμόζονται σε κάθε ένα από αυτά τα μισά με τον τρόπο που προαναφέρθηκε. Η διαδικασία της υποδιαίρεσης επαναλαμβάνεται συνεχώς έως ότου λάβουμε ένα εξ ολοκλήρου ορατό και ένα εξ ολοκλήρου μη ορατό ευθύγραμμο τμήμα.

Τα βήματα του αλγορίθμου συνοπτικά είναι:

1. Υπολόγισε τους εξωκωδικούς για κάθε ένα από τα δύο τελικά σημεία του ευθυγράμμου τμήματος.

2. Εφάρμοσε λογική διάζευξη μεταξύ των δύο εξωκωδικών. Αν το αποτέλεσμα είναι 0000, τότε το ευθύγραμμο τμήμα είναι εξ ολοκλήρου ορατό, βρίσκεται εντός της περιοχής αποκοπής, σχεδίασέ το ολόκληρο και τερμάτισε τον αλγόριθμο.

3. Εφάρμοσε λογική σύζευξη μεταξύ των δύο εξωκωδικών. Αν το αποτέλεσμα δεν είναι 0000, τότε το ευθύγραμμο τμήμα είναι εξ ολοκλήρου αόρατο, βρίσκεται εκτός της περιοχής αποκοπής, απόρριψέ το και τερμάτισε τον αλγόριθμο.

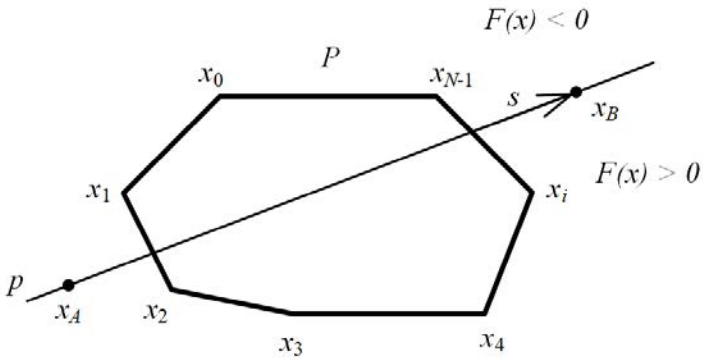
4. Διαίρεσε το ευθύγραμμο τμήμα σε δύο ίσα μέρη και επανέλαβε την ίδια διαδικασία για κάθε ένα από αυτά ξεκινώντας από το Βήμα 1.

Η υλοποίηση του αλγορίθμου βρίσκεται στο Παράρτημα A - Κώδικας A.5.

2.6 Αλγόριθμος Skala 2005

Ο αλγόριθμος Skala (SKA05) δημιουργήθηκε από τον καθηγητή Václav Skala το 2005 και πραγματοποιεί αποκοπή ευθυγράμμου τμήματος σε ορθογώνιο παράθυρο αποκοπής αλλά και σε οποιοδήποτε άλλο κυρτό πολύγωνο. Στους υπολογισμούς του, η διαίρεση, η οποία είναι υπολογιστικά δαπανηρή πράξη, εφαρμόζεται ελάχιστα ενώ παράλληλα κάνει χρήση ομογενών συντεταγμένων για την αναπαράσταση των σημείων που προσδιορίζουν, τόσο το αρχικό, όσο και το αποκομμένο ευθύγραμμο τμήμα. Σύμφωνα με τον δημιουργό του, εκμεταλλεύεται επιπροσθέτως ορισμένες από τις λειτουργίες επιτάχυνσης που διαθέτει το υλικό, οι οποίες σχετίζονται με τις πράξεις μεταξύ διανυσμάτων [Ska05].

Ο αλγόριθμος θεωρεί ένα κυρτό πολύγωνο P και μία ευθεία p η οποία περιγράφεται από την εξίσωση $F(x) = ax + by + c = 0$. Η ευθεία p διαιρεί το επίπεδο σε δύο ημιεπίπεδα ενώ η συνάρτηση $F(x)$ κατηγοριοποιεί τις κορυφές του κυρτού πολυγώνου ανάλογα με το αν βρίσκονται αριστερά ή δεξιά της ευθείας. Αριστερά σημαίνει ότι διασχίζοντας την ευθεία από το άπειρο με τη φορά του διανύσματος $\overrightarrow{x_A x_B}$, τότε για κάθε σημείο που υπάρχει στο αριστερό ημιεπίπεδο ισχύει ότι $F(x) < 0$ (βλ. Σχ. 2.21).



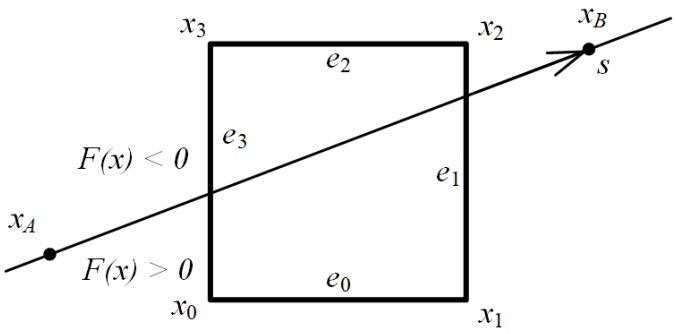
Σχήμα 2.21: Κατηγοριοποίηση των κορυφών ενός παραθύρου αποκοπής σχήματος κυρτού πολύγωνου

Ποιο συγκεκριμένα, κάθε κορυφή του πολυγώνου κατηγοριοποιείται ως εξής:

$$c_i = \begin{cases} 1 & \text{αν } F(x_i) \geq 0 \\ 0 & \text{σε οποιοδήποτε άλλη περίπτωση} \end{cases} \qquad i : 0, 1, \ldots, N-1$$

Στην περίπτωση που το κυρτό πολύγωνο είναι ορθογώνιο παραλληλόγραμμο όπως αυτό του Σχ. 2.22, οι κορυφές του χαρακτηρίζονται από ένα σύνολο c τεσσάρων δυαδικών ψηφίων:

$$c = [c_3, c_2, c_1, c_0]^T$$



Σχήμα 2.22: Κατηγοριοποίηση των κορυφών ενός ορθογωνίου παραθύρου αποκοπής

Ανάλογα με την θέση της ευθείας, η κατηγοριοποίηση των κορυφών του ορθογωνίου παραθύρου αποκοπής αντικατοπτρίζεται στον Πίνακα 2.1:

c	c_3	c_2	c_1	c_0	**TAB1**	**TAB2**	MASK
0	0	0	0	0	None	None	None
1	0	0	0	1	0	3	0100
2	0	0	1	0	0	1	0100
3	0	0	1	1	1	3	0010
4	0	1	0	0	1	2	0010
5	0	1	0	1	N/A	N/A	N/A
6	0	1	1	0	0	2	0100
7	0	1	1	1	2	3	1000
8	1	0	0	0	2	3	1000
9	1	0	0	1	0	2	0100
10	1	0	1	0	N/A	N/A	N/A
11	1	0	1	1	1	2	0010
12	1	1	0	0	1	3	0010
13	1	1	0	1	0	1	0100
14	1	1	1	0	0	3	0100
15	1	1	1	1	None	None	None

Πίνακας 2.1: Κατηγοριοποίηση των κορυφών ενός ορθογωνίου παραθύρου αποκοπής

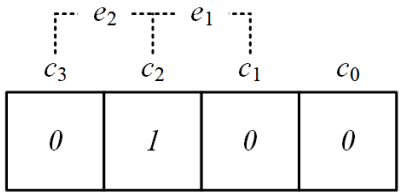
Προφανώς υπάρχουν και ορισμένες περιπτώσεις για τις οποίες δεν μπορεί να γίνει κατηγοριοποίηση, βλ. $c = 5$ και $c = 10$.

Από τις τιμές των c_i, ο αλγόριθμος μπορεί να εντοπίσει ποια από τα δύο σύνορα (ακμές) του παραθύρου τέμνει το ευθύγραμμο τμήμα. Οι τιμές TAB1 και TAB2 υποδεικνύουν ακριβώς αυτό. Για παράδειγμα, αν θεωρήσουμε το ευθύγραμμο τμήμα του Σχήματος 2.22, διαπιστώνουμε ότι οι τιμές των c_0 και c_1 είναι 1 ενώ οι τιμές των c_2 και c_3 είναι 0. Από τον Πίνακα 2.1 φαίνεται ότι πρόκειται για την περίπτωση $c = 3$ όπου οι τιμές των TAB1 και TAB2 είναι 1 και 3 αντίστοιχα και, κατά συνέπεια, η ευθεία τέμνει τις ακμές $e1$ και $e3$. Η στήλη MASK βοηθάει σε ορισμένες περιπτώσεις τον αλγόριθμο να προσδιορίσει ποια από τις δύο ακμές που υποδεικνύουν οι τιμές των TAB1 και TAB2 θα χρησιμοποιηθεί στη διαδικασία της αποκοπής.

Οι τιμές TAB1 και TAB2 δεν σχετίζονται με την φορά που διατρέχουμε την ευθεία και κατά συνέπεια του ευθυγράμμου τμήματος που πρόκειται να αποκοπεί. Στο παράδειγμα του Σχήματος 2.22, θα περίμενε κανείς ότι αν διατρέξουμε την ευθεία ξεκινώντας από το σημείο x_A και καταλήγοντας στο σημείο x_B, τότε θα συναντήσουμε πρώτα την ακμή e_3 και μετά την e_1 οπότε και οι τιμές των TAB1 και TAB2 θα έπρεπε, θεωρητικά, να ήταν 3 και 1 αντίστοιχα (περίπτωση $c = 3$). Ομοίως, διαπιστώνουμε ότι αν διατρέξουμε την ευθεία από το σημείο x_B προς το σημείο x_A, τότε συναντούμε πρώτα την ακμή e_1 και μετά την e_3 οπότε οι τιμές των TAB1 και TAB2 φαίνεται ότι ορθώς είναι 1 και 3 αντίστοιχα (περίπτωση $c = 12$). Στην πράξη, η σειρά στις τιμές των TAB1 και TAB2 δεν επηρεάζει την λειτουργία του αλγορίθμου. Αυτό εξηγείται ως εξής: Αν υποθέσουμε ότι και τα δύο σημεία του ευθυγράμμου τμήματος βρίσκονται εκτός του παραθύρου αποκοπής αλλά το ευθύγραμμο τμήμα τέμνει το παράθυρο, τότε εφ' όσον ο αλγόριθμος προσδιορίσει τα σημεία τομής με τις ακμές που υποδεικνύουν οι τιμές των TAB1 και TAB2, απλώς σχεδιάζει το αποκομμένο νέο ευθύγραμμο τμήμα μεταξύ αυτών. Αν όμως το ένα από τα δύο σημεία βρίσκεται εντός του παραθύρου αποκοπής και το άλλο βρίσκεται εκτός, τότε ο αλγόριθμος προσδιορίζει, με την βοήθεια της τιμής MASK, την σωστή ακμή η οποία είναι είτε η τιμή του TAB1 είτε η τιμή του TAB2.

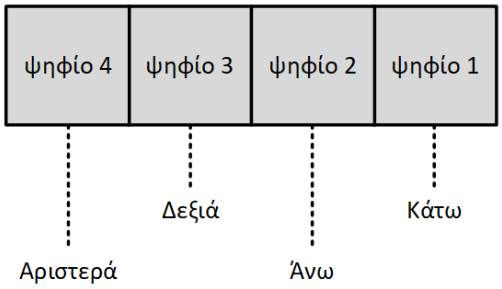
Ο Πίνακας 2.1 μπορεί να χρησιμοποιηθεί ως έχει εάν πρόκειται για ορθογώνιο παράθυρο αποκοπής ή να προϋπολογιστεί πριν ξεκινήσει η διαδικασία της αποκοπής. Αυτό είναι ιδιαίτερα χρήσιμο, όταν το παράθυρο αποκοπής περιλαμβάνει περισσότερες από τέσσερις ακμές επειδή ο πίνακας είναι πολυπλοκότερος. Ο προϋπολογισμός των τιμών γίνεται με τον ακόλουθο τρόπο: Εφ'όσον οι τιμές c_i χαρακτηρίζουν την θέση των κορυφών x_i του παραθύρου αποκοπής σε σχέση με το ευθύγραμμο τμήμα, οποιαδήποτε εναλλαγή από 0 σε 1 ή από 1 σε 0 υποδηλώνει ότι η ακμή στην οποία βρίσκονται οι κορυφές αυτές τέμνει το ευθύγραμμο τμήμα. Για παράδειγμα, αν εξετάσουμε την περίπτωση $c = 4$ του Πίνακα 2.1, διαπιστώνουμε ότι οι τιμές c_3, c_2, c_1, c_0 είναι $0, 1, 0, 0$ αντίστοιχα άρα το ευθύγραμμο τμήμα τέμνει τις ακμές στις οποίες βρίσκονται οι κορυφές x_1x_2 και x_2x_3, δηλαδή τις e_1 και e_2 (βλ.

Σχ. 2.23).

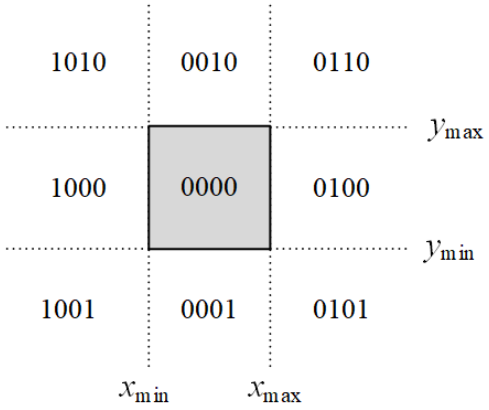


Σχήμα 2.23: Παράδειγμα εντοπισμού των εναλλαγών από 0 σε 1 και αντιστρόφως

Εκτός από την κατηγοριοποίηση των κορυφών, ο αλγόριθμος κατηγοριοποιεί και τα σημεία που προσδιορίζουν το ευθύγραμμο τμήμα με την λογική του αλγορίθμου Cohen-Sutherland με σκοπό την γρηγορότερη αποδοχή ή απόρριψή του. Έτσι, ο χώρος διαμερίζεται και εδώ σε εννέα περιοχές όπου κάθε μία χαρακτηρίζεται από έναν τετραψήφιο κωδικό περιοχής (εξωκωδικός) με την μόνη διαφορά ότι τα δυαδικά του ψηφία δείχνουν τις περιοχές (ξεκινώντας από το ΛΣΨ και καταλήγοντας στο ΠΣΨ) με την σειρά: κάτω, άνω, δεξιά, αριστερά (βλ. Σχ. 2.24 και Σχ. 2.25).



Σχήμα 2.24: Τα ψηφία του κωδικού περιοχής στον αλγόριθμο Skala 2005



Σχήμα 2.25: Οι κωδικοί περιοχών του αλγορίθμου Skala 2005

Η συνάρτηση outcode του Κώδικα A.6 στο Παράρτημα A υπολογίζει τον κωδικό περιοχής κάθε τελικού σημείου του ευθυγράμμου τμήματος.

Μεταξύ των δύο κωδικών περιοχής ο αλγόριθμος:

1. Εφαρμόζει λογική διάζευξη και αν το αποτέλεσμα είναι 0000, τότε το ευθύγραμμο τμήμα είναι εξ ολοκλήρου εντός της περιοχής αποκοπής και σχεδιάζεται ως έχει.

2. Εφαρμόζει λογική σύζευξη και αν το αποτέλεσμα δεν είναι 0000, τότε το ευθύγραμμο τμήμα είναι εξ ολοκλήρου εκτός της περιοχής αποκοπής οπότε απορρίπτεται.

Ο χαρακτηρισμός της θέσης των κορυφών με 0 ή 1 (αριστερό ή δεξιό ημιεπίπεδο) και ο προσδιορισμός του δείκτη c μπορεί να γίνει με την συνάρτηση calculate_c_index του Κώδικα A.7 του Παραρτήματος A. Για τις τιμές 0 ή 15 του δείκτη c ο αλγόριθμος σταματά ενώ για οποιαδήποτε άλλη τιμή που βρίσκεται στο διάστημα $[1..14]$ υπολογίζονται οι τιμές των TAB1 και TAB2.

Ο έλεγχος συνεχίζεται με τους κωδικούς των περιοχών στις οποίες βρίσκονται τα σημεία του ευθυγράμμου τμήματος. Αν οι δύο κωδικοί είναι 0000, τότε και τα δύο σημεία τέμνουν τις ακμές που υποδεικνύουν τα TAB1 και TAB2 οπότε ο αλγόριθμος υπολογίζει τα σημεία τομής τους με την χρήση του εξωτερικού γινομένου και σχεδιάζει μεταξύ τους το αποκομμένο ευθύγραμμο τμήμα που προκύπτει.

Εάν μόνο ένα από τα δύο σημεία βρίσκονται εντός της κεντρικής περιοχής (παράθυρο αποκοπής), δηλαδή έχει κωδικό περιοχής 0000, τότε ο αλγόριθμος ελέγχει το άλλο σημείο για το ποια από τις ακμές τέμνει. Σε αυτό βοηθάει η αντίστοιχη τιμή MASK του Πίνακα 2.1. Έτσι, εφαρμόζεται λογική σύζευξη μεταξύ του κωδικού περιοχής του σημείου και της τιμής MASK και ο αλγόριθμος σχεδιάζει το νέο αποκομμένο ευθύγραμμο τμήμα ξεκινώντας από το σημείο που βρίσκεται εντός της κεντρικής περιοχής έως το σημείο τομής που προέκυψε από την λογική σύζευξη.

Τα βήματα του αλγορίθμου συνοπτικά είναι:

1. Υπολόγισε τους κωδικούς περιοχών για κάθε ένα από τα δύο σημεία που προσδιορίζουν το ευθύγραμμο τμήμα.

2. Εφάρμοσε λογική διάζευξη μεταξύ των δύο κωδικών περιοχής. Αν το αποτέλεσμα είναι 0000, τότε το ευθύγραμμο τμήμα είναι εξ ολοκλήρου ορατό οπότε σχεδίασέ το και σταμάτησε.

3. Εφάρμοσε λογική σύζευξη μεταξύ των δύο κωδικών περιοχής. Αν το αποτέλεσμα δεν είναι 0000, τότε το ευθύγραμμο τμήμα είναι εξ ολοκλήρου αόρατο οπότε απόρριψέ το και σταμάτησε.

4. Χαρακτήρισε με 0 ή 1 τις κορυφές της κεντρικής περιοχής ανάλογα με το αν βρίσκονται στο αριστερό ή δεξιό ημιεπίπεδο που δημιουργεί η ευθεία στην οποία βρίσκεται το ευθύγραμμο τμήμα.

5. Υπολόγισε τον δείκτη c. Αν έχει είτε την τιμή 0 είτε την τιμή 15, τότε σταμάτησε.

6. Υπολόγισε τις τιμές των TAB1 και TAB2 με την βοήθεια του δείκτη c.

7. Αν η τιμή και των δύο κωδικών περιοχής είναι 0000, τότε το ευθύγραμμο τμήμα τέμνει και τις δύο ακμές της κεντρικής περιοχής που υποδεικνύουν οι τιμές των TAB1 και TAB2. Υπολόγισε τα σημεία τομής τους και σχεδίασε το μεταξύ τους ευθύγραμμο τμήμα.

8. Αν μόνο η τιμή του πρώτου κωδικού περιοχής είναι 0000 (άρα εντός της κεντρικής περιοχής), τότε έλεγξε τον κωδικό περιοχής του δεύτερου σημείου με την τιμή της μάσκας (MASK) που υποδεικνύει ο δείκτης c μέσω λογικής σύζευξης. Αν το αποτέλεσμα δεν είναι 0000, τότε σχεδίασε το ευθύγραμμο τμήμα μεταξύ του πρώτου σημείου και του ορίου που υποδεικνύει η τιμή TAB1 αλλιώς με το σύνορο που υποδεικνύει η τιμή TAB2.

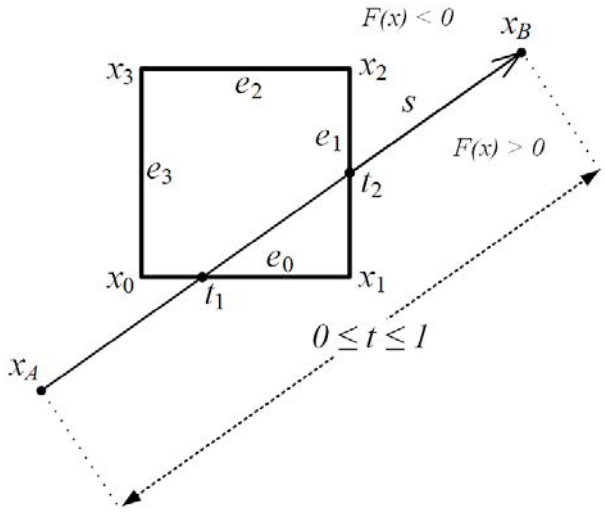
9. Αν μόνο η τιμή του δεύτερου κωδικού περιοχής είναι 0000 (άρα εντός της κεντρικής περιοχής), τότε έλεγξε τον κωδικό περιοχής του πρώτου σημείου με την τιμή της μάσκας (MASK) που υποδεικνύει ο δείκτης c μέσω λογικής σύζευξης. Αν το αποτέλεσμα δεν είναι 0000, τότε σχεδίασε το ευθύγραμμο τμήμα μεταξύ του δεύτερου σημείου και του ορίου που υποδεικνύει η τιμή TAB1 αλλιώς με το σύνορο που υποδεικνύει η τιμή TAB2.

Η υλοποίηση του αλγορίθμου βρίσκεται στο Παράρτημα A - Κώδικας A.8.

2.7 Αλγόριθμος S-Clip E^2

Ο S-Clip E^2 είναι ένας ακόμη αλγόριθμος ο οποίος δημιουργήθηκε από τον καθηγητή Václav Skala το 2012. Στην πράξη, πρόκειται για μία βελτιωμένη έκδοση του Skala 2005 η οποία βασίζεται στην λογική «πρώτα έλεγξε και μετά υπολόγισε» ενώ παράλληλα απλουστεύει ορισμένες ενέργειες του [Ska12]. Η βασικότερη διαφορά του S-Clip E^2 είναι ότι δεν χρειάζεται να αποδοθούν κωδικοί περιοχής στα σημεία που χαρακτηρίζουν το ευθύγραμμο τμήμα. Η αξιολόγηση της θέσης των σημείων γίνεται μέσω της τιμής t της παραμετρικής εξίσωσης $P(t) = P_0 + t \cdot (P_1 - P_0)$.

Ας θεωρήσουμε το ορθογώνιο παράθυρο αποκοπής του Σχ. 2.26. Το παράθυρο ορίζεται από τις τέσσερις κορυφές x_i, όπου $i: 0, 1, 2, 3$. Από αυτές σχηματίζονται οι τέσσερις ακμές e_i ενώ υπάρχει και το ευθύγραμμο τμήμα s το οποίο προσδιορίζεται από τα σημεία x_A και x_B το οποίο τέμνει τις ακμές e_3 και e_2. Υπάρχουν δύο τιμές, έστω t_1 και t_2, οι οποίες επαληθεύουν την παραμετρική εξίσωση στα σημεία τομής.



Σχήμα 2.26: Η τιμή t στην παραμετρική εξίσωση του ευθυγράμμου τμήματος κυμαίνεται από 0 έως 1

Στην αρχή, ο αλγόριθμος χαρακτηρίζει τις κορυφές του παραθύρου αποκοπής με τον ίδιο τρόπο που κάνει ο Skala 2005, αποδίδοντας σε αυτές τιμές 0 ή 1, ανάλογα με το αν βρίσκονται στο αριστερό ή δεξιό ημιεπίπεδο που δημιουργεί η ευθεία επάνω στην οποία βρίσκεται το ευθύγραμμο τμήμα. Με το σύνολο των δυαδικών ψηφίων τα οποία χαρακτηρίζουν την θέση των κορυφών, σχηματίζεται ο *κωδικός* c και με τη βοήθεια του Πίνακα 2.2 υπολογίζονται οι τιμές των TAB1 και TAB2, οπότε ο αλγόριθμος γνωρίζει ποιες ακμές τέμνει το ευθύγραμμο τμήμα. Για τις περιπτώσεις που ο κωδικός είναι 0000 ή 1111, η διαδικασία της αποκοπής σταματά. Οι τιμές TAB1 και TAB2 στο παράδειγμα του Σχ. 2.26 είναι 0 και 1 αντίστοιχα (κωδικός 0010).

Το επόμενο βήμα του αλγορίθμου είναι να υπολογίσει τα δύο σημεία τομής κάνοντας χρήση του εξωτερικού γινομένου και στη συνέχεια να προσδιορίσει τις τιμές των παραμέτρων t_1 και t_2. Συγκρίνοντας τις τιμές t_1 και t_2, ο αλγόριθμος ενεργεί ως ακολούθως:

1. Αν $t_1 < 0$ και $t_2 > 1$, τότε το ευθύγραμμο τμήμα είναι εντός του παραθύρου αποκοπής και σχεδιάζεται εξ ολοκλήρου, περίπτωση (α) στο Σχ. 2.27.

2. Αν $0 \leq t_1 \leq 1$ και $0 \leq t_2 \leq 1$, τότε το ευθύγραμμο τμήμα διέρχεται και από τις δύο ακμές του παραθύρου

code	TAB1	TAB2	code	TAB1	TAB2
0000	-1	-1	1111	-1	-1
0001	3	0	1110	0	3
0010	0	1	1110	1	0
0011	3	1	1101	1	3
0100	1	2	1100	2	1
0101	N/A	N/A	1011	N/A	N/A
0110	0	2	1001	2	0
0111	3	2	1000	2	3

Πίνακας 2.2: Οι κωδικοί και οι τιμές των TAB1 και TAB2, όταν το παράθυρο αποκοπής είναι ορθογώνιο

αποκοπής, αποκόπτεται και σχεδιάζεται ξεκινώντας από το σημείο τομής με παράμετρο t_1 έως το σημείο τομής με παράμετρο t_2, περίπτωση (β) στο Σχ. 2.27.

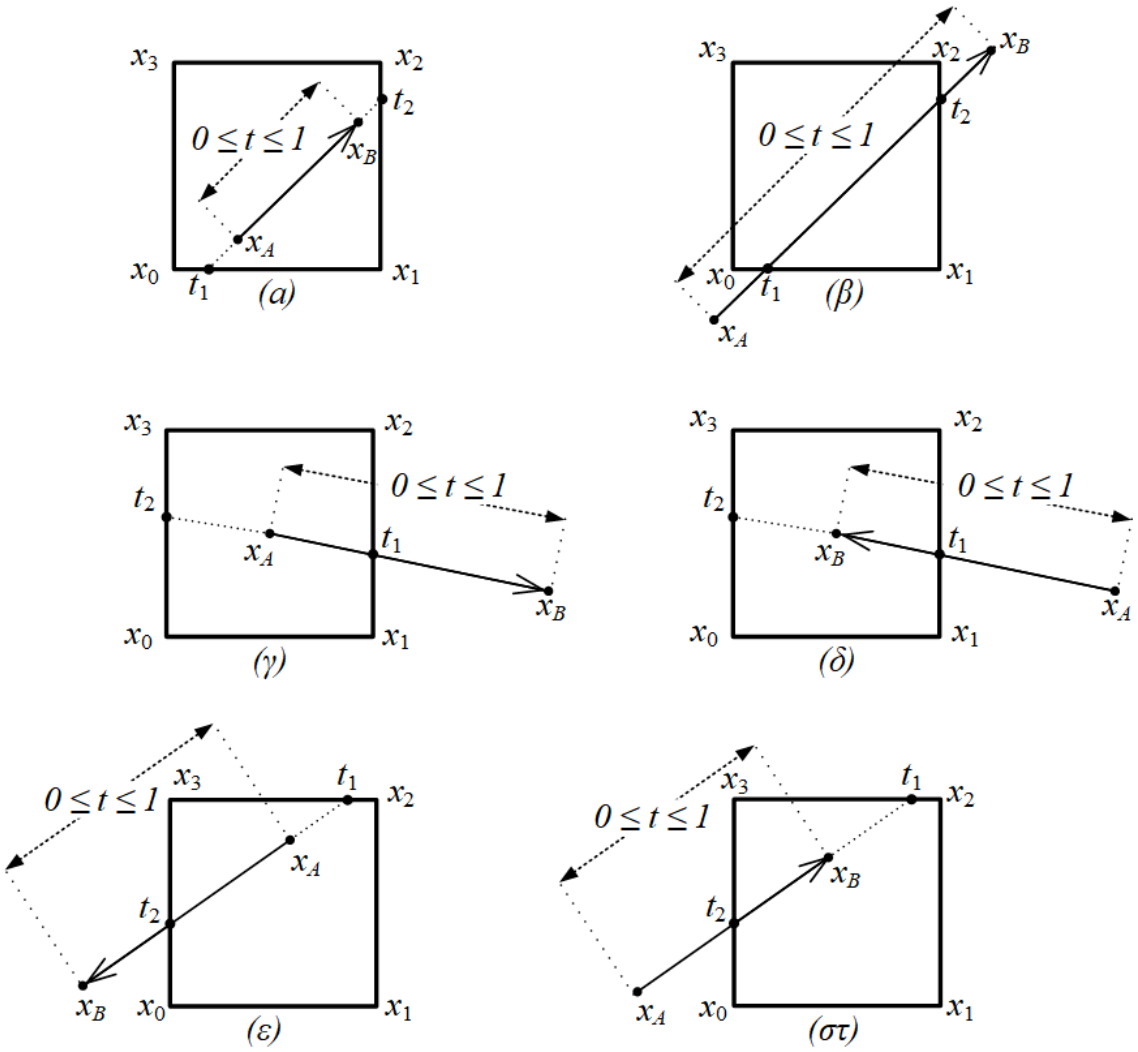
3. Αν $0 \leq t_1 \leq 1$ και $t_2 < 0$, τότε ο αλγόριθμος αποκόπτει και σχεδιάζει το ευθύγραμμα τμήμα ξεκινώντας από το σημείο που ορίζει την αρχή του ευθυγράμμου τμήματος έως το σημείο τομής με παράμετρο t_1, περίπτωση (γ) στο Σχ. 2.27.

4. Αν $0 \leq t_1 \leq 1$ και $t_2 > 1$, τότε ο αλγόριθμος αποκόπτει και σχεδιάζει το ευθύγραμμα τμήμα ξεκινώντας από το σημείο τομής με παράμετρο t_1 έως το σημείο που ορίζει το τέλος του ευθυγράμμου τμήματος, περίπτωση (δ) στο Σχ. 2.27.

5. Αν $0 \leq t_2 \leq 1$ και $t_1 < 0$, τότε ο αλγόριθμος αποκόπτει και σχεδιάζει το ευθύγραμμα τμήμα ξεκινώντας από το σημείο που ορίζει την αρχή του ευθυγράμμου τμήματος έως το σημείο τομής με παράμετρο t_2, περίπτωση (ε) στο Σχ. 2.27.

6. Αν $0 \leq t_2 \leq 1$ και $t_1 > 1$, τότε ο αλγόριθμος αποκόπτει και σχεδιάζει το ευθύγραμμα τμήμα ξεκινώντας από το σημείο τομής με παράμετρο t_2 έως το σημείο που ορίζει το τέλος του ευθυγράμμου τμήματος, περίπτωση (στ) στο Σχ. 2.27.

7. Σε οποιαδήποτε άλλη περίπτωση, το ευθύγραμμο τμήμα βρίσκεται εξ ολοκλήρου εκτός και δεν σχεδιάζεται.

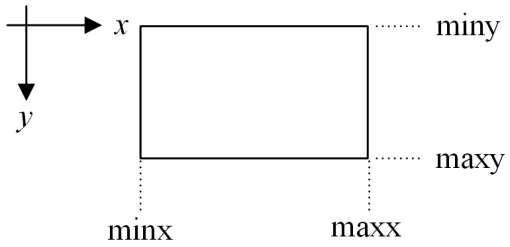


Σχήμα 2.27: Περιπτώσεις αποκοπής του αλγορίθμου S-Clip E^2 ανάλογα με τις τιμές των παραμέτρων t_1 και t_2

Ο Κώδικας A.9 του Παραρτήματος A αποτελεί την υλοποίηση του αλγορίθμου S-Clip E^2 και κάνει χρήση της συνάρτησης *calculate_c_ index* που χρησιμοποιήθηκε στον Skala 2005 (βλ. Κωδ. A.7).

2.8 Αλγόριθμος Kodituwakku–Wijeweere–Chamikara

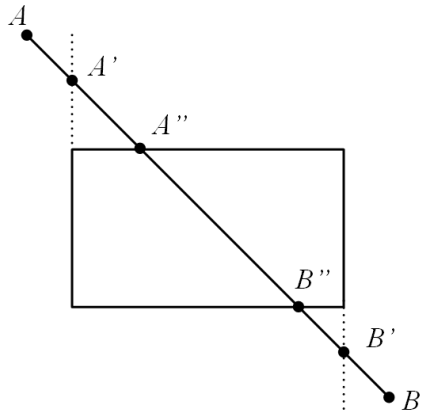
Το 2013 δημιουργήθηκε ο αλγόριθμος KWC από τους Kodituwakku S.R., Wijeweere K.R. και Chamikara M.A.P ο οποίος έχει παρόμοια προσέγγιση με εκείνη του Cohen-Sutherland[KWC13]. Το παράθυρο αποκοπής ορίζεται από τα σημεία $(\text{minx}, \text{miny})$ και $(\text{maxx}, \text{maxy})$ (βλ. Σχ. 2.28).



Σχήμα 2.28: Τα σημεία που ορίζουν το παράθυρο αποκοπής του αλγορίθμου KWC

Αν το προς αποκοπή ευθύγραμμο τμήμα περιγράφεται από τα σημεία $A(x_0, y_0)$ και $B(x_1, y_1)$, τότε οι συντεταγμένες του ελέγχονται σε σχέση με τα σύνορα του ορθογωνίου παραθύρου αποκοπής. Σε περίπτωση που μία συντεταγμένη υπερβαίνει κάποιο από τα σύνορα, τότε χρησιμοποιείται η συντεταγμένη του συγκεκριμένου συνόρου ενώ η άλλη συντεταγμένη υπολογίζεται με την εξίσωση της ευθείας στις δύο διαστάσεις $y = m \cdot x + c$, όπου m η κλίση της η οποία ισούται με $m = (y_1 - y_0)/(x_1 - x_0)$.

Ας θεωρήσουμε το ευθύγραμμο τμήμα του Σχ. 2.29 το οποίο δεν είναι παράλληλο ούτε με τον οριζόντιο άξονα ούτε με τον κατακόρυφο.



Σχήμα 2.29: Η διαδικασία ελέγχου και σταδιακής αποκοπής του ευθυγράμμου τμήματος

Αρχικά, ο αλγόριθμος θα ελέγξει τη θέση του σημείου A με κάθε σύνορο του παραθύρου αποκοπής. Αν υποθέσουμε οι έλεγχοι γίνονται διαδοχικά με τα σύνορα αριστερά, άνω, δεξιά και κάτω, τότε οι συντεταγμένες του σημείου A θα μεταβληθούν αντίστοιχα ως εξής:

- Έλεγχος με το αριστερό σύνορο ($A_x < x_{\text{min}}$): Οι συντεταγμένες του A μεταβάλλονται και γίνονται A'.
- Έλεγχος με το άνω σύνορο ($A'_y < y_{\text{min}}$): Οι συντεταγμένες του A' μεταβάλλονται και γίνονται A''.

- Έλεγχος με το δεξί σύνορο ($A''_x > x_{\text{max}}$): Οι συντεταγμένες του A'' δεν μεταβάλλονται.
- Έλεγχος με το κάτω σύνορο ($A''_y > y_{\text{max}}$): Οι συντεταγμένες του A'' δεν μεταβάλλονται.

Στη συνέχεια, με τον ίδια τρόπο ελέγχεται το σημείο B:

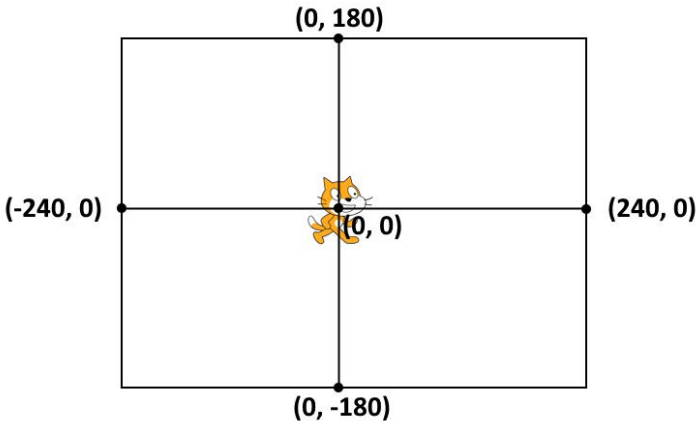
- Έλεγχος με το αριστερό σύνορο ($B_x < x_{\text{min}}$): Οι συντεταγμένες του B δεν μεταβάλλονται.
- Έλεγχος με το άνω σύνορο ($B'_y < y_{\text{min}}$): Οι συντεταγμένες του B δεν μεταβάλλονται.
- Έλεγχος με το δεξί σύνορο ($B''_x > x_{\text{max}}$): Οι συντεταγμένες του B μεταβάλλονται και γίνονται B'.
- Έλεγχος με το κάτω σύνορο ($B''_y > y_{\text{max}}$): Οι συντεταγμένες του B'' μεταβάλλονται και γίνονται B''.

Το προκύπτον ευθύγραμμο τμήμα μεταξύ των σημείων A'' και B'' είναι το επιθυμητό αποτέλεσμα. Προφανώς, αν ένα από τα δύο σημεία A και B ή και τα δύο μαζί είναι εντός του παραθύρου αποκοπής, τότε οι έλεγχοι πραγματοποιούνται σε αυτά χωρίς να επηρεάζουν τις συντεταγμένες τους.

Η υλοποίηση του αλγορίθμου φαίνεται στον Κώδικα A.10 του Παραρτήματος A ο οποίος, μάλιστα, λαμβάνει υπόψη τα κάθετα και τα οριζόντια στους άξονες ευθύγραμμα τμήματα.

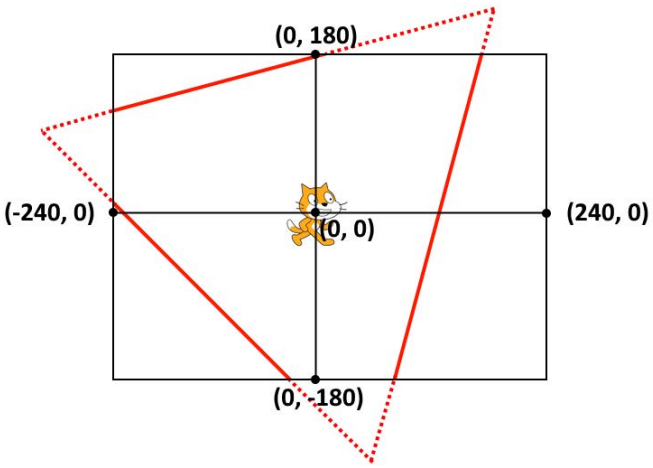
2.9 Προτεινόμενος αλγόριθμος

Ο ακόλουθος αλγόριθμος προτάθηκε από τον συγγραφέα και τον επιβλέποντά του το 2019 [MD19b] στα πλαίσια μίας ερευνητικής εργασίας της Δευτεροβάθμιας Εκπαίδευσης βασισμένη στο προγραμματιστικό περιβάλλον Scratch. Το Scratch αποτελεί μία δωρεάν διερμηνευόμενη δυναμική οπτική γλώσσα προγραμματισμού η οποία χρησιμοποιεί γραφικά στοιχεία ως εντολές (πλακίδια). Χρησιμοποιείται ευρέως στην εκπαίδευση ως εισαγωγικό εργαλείο στην διδασκαλία του προγραμματισμού. Διαθέτει μία οθόνη σταθερών διαστάσεων (480x360 εινονοστοιχεία) για την απεικόνιση αντικειμένων, η οποία αποκαλείται Σκηνή (βλ. Σχ. 2.30).



Σχήμα 2.30: Η Σκηνή του Scratch

Το Scratch δεν επιτρέπει στα αντικείμενά του να ξεπεράσουν τα σύνορα της Σκηνής. Λόγω αυτού του περιορισμού, η σχεδίαση των γραφικών περιορίζεται, επίσης, εντός της Σκηνής διότι πραγματοποιείται με την χρήση αντικειμένων. Οι μαθητές, οι οποίοι έλαβαν μέρος στην ερευνητική εργασία, θέλοντας να σχεδιάσουν ένα απλό γεωμετρικό σχήμα σε μέγεθος μεγαλύτερο της Σκηνής, διαπίστωσαν ότι αυτό δεν ήταν εφικτό με τους περιορισμούς και τους υφιστάμενους τρόπους σχεδίασης που διαθέτει το Scratch. Υπήρξε, λοιπόν, η ανάγκη για την χρήση ενός αλγορίθμου αποκοπής (βλ. Σχ. 2.31).

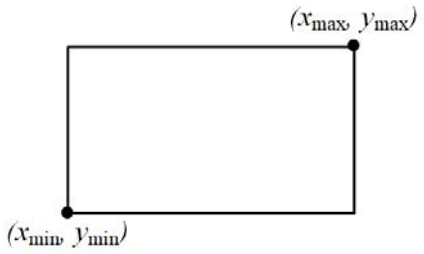


Σχήμα 2.31: Ανάγκη για την χρήση αλγορίθμου αποκοπής κατά την σχεδίαση στο Scratch όταν το μέγεθος του σχήματος ξεπερνάει τα όρια της Σκηνής

Οι παραδοσιακοί αλγόριθμοι δυσκόλευαν τους μαθητές κυρίως επειδή χρησιμοποιούν προχωρημένες μαθηματικές έννοιες ενώ, ορισμένοι από αυτούς, είναι εκτενείς σε μέγεθος. Για την επίλυση του προβλήματος αυτού,

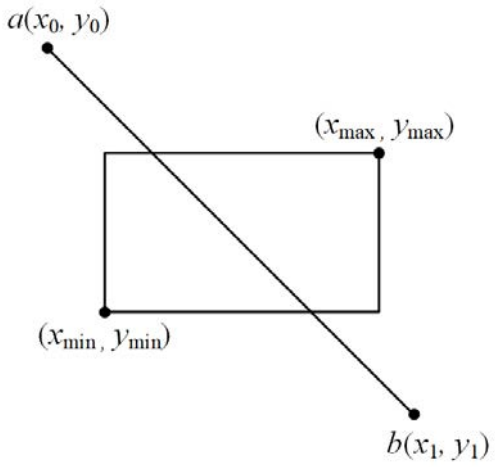
δημιουργήθηκε ο προτεινόμενος αλγόριθμος ο οποίος πραγματοποιεί αποκοπή ευθυγράμμων τμημάτων, είτε σε ορθογώνιο παράθυρο αποκοπής είτε σε κυρτό πολύγωνο, ενώ μπορεί εύκολα να επεκταθεί στις τρεις διαστάσεις. Οι βασικοί στόχοι κατά τη υλοποίησή του ήταν η απλότητα του κώδικα, ούτως ώστε να μπορεί να γίνει εύκολα κατανοητός, καθώς και η ταχύτητα [MD19a].

Ας θεωρήσουμε το ορθογώνιο παράθυρο αποκοπής το οποίο ορίζεται από τα σημεία $(x_{\text{min}}, y_{\text{min}})$ και $(x_{\text{max}}, y_{\text{max}})$ (βλ. Σχ. 2.32).



Σχήμα 2.32: Το παράθυρο αποκοπής του προτεινόμενου αλγορίθμου

Αν δύο σημεία $a(x_0, y_0)$ και $b(x_1, y_1)$ ορίζουν το προς αποκοπή ευθύγραμμο τμήμα (βλ. Σχ. 2.33)



Σχήμα 2.33: Το προς αποκοπή ευθύγραμμο τμήμα και τα σημεία a και b που το ορίζουν

τότε η κλίση του m είναι σταθερή και εκφράζεται από τον λόγο:

$$m = \frac{y_1 - y_0}{x_1 - x_0} \tag{2.1}$$

Οποιοδήποτε σημείο ανήκει στην ευθεία και έχει συντεταγμένες (x, y) επαληθεύει τον παραπάνω λόγο ο οποίος

γίνεται:

$$m = \frac{y - y_0}{x - x_0}$$

Λύνοντας ως προς y, η εξίσωση παίρνει την μορφή:

$$y - y_0 = m \cdot (x - x_0) \Rightarrow$$
$$y = y_0 + m \cdot (x - x_0)$$

Και αν αντικαταστήσουμε το m χρησιμοποιώντας την εξίσωση 2.1:

$$y = y_0 + \frac{y_1 - y_0}{x_1 - x_0} \cdot (x - x_0) \tag{2.2}$$

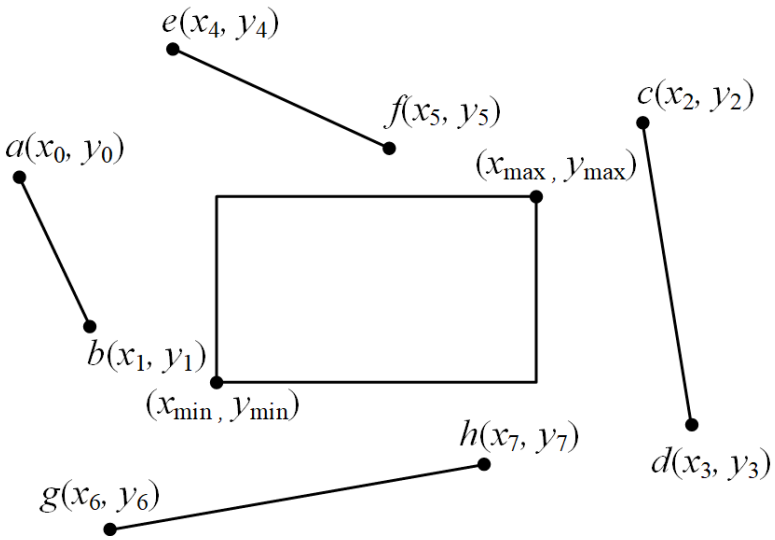
Λύνοντας ως προς x, η εξίσωση γίνεται:

$$x = x_0 + \frac{x_1 - x_0}{y_1 - y_0} \cdot (y - y_0) \tag{2.3}$$

Το πρώτο βήμα του αλγορίθμου είναι να ελέγξει εάν και τα δύο σημεία a, b του ευθυγράμμου τμήματος βρίσκονται ταυτόχρονα εκτός κάποιου συνόρου αποκοπής, δηλαδή αριστερά ή δεξιά ή κάτω ή άνω αυτού. Οι έλεγχοι που γίνονται είναι:

$x_0 < x_{\text{min}}$ ΚΑΙ $x_1 < x_{\text{min}}$	(το ευθύγραμμο τμήμα βρίσκεται αριστερά του παραθύρου)
$x_0 > x_{\text{max}}$ ΚΑΙ $x_1 > x_{\text{max}}$	(το ευθύγραμμο τμήμα βρίσκεται δεξιά του παραθύρου)
$y_0 < y_{\text{min}}$ ΚΑΙ $y_1 < y_{\text{min}}$	(το ευθύγραμμο τμήμα βρίσκεται κάτωθεν του παραθύρου)
$y_0 > y_{\text{max}}$ ΚΑΙ $y_1 > y_{\text{max}}$	(το ευθύγραμμο τμήμα βρίσκεται άνωθεν του παραθύρου)

Αν μία από τις παραπάνω περιπτώσεις είναι Αληθής, τότε η διαδικασία σταματάει, το ευθύγραμμο τμήμα απορρίπτεται και ο αλγόριθμος δεν σχεδιάζει (βλ. Σχ. 2.34).



Σχήμα 2.34: Τα ευθύγραμμα τμήματα ab, cd, ef και gh απορρίπτονται διότι βρίσκονται εκτός του παραθύρου αποκοπής

Στο δεύτερο βήμα, ο αλγόριθμος συγκρίνει διαδοχικά κάθε ένα από τα δύο τελικά σημεία του ευθυγράμμου τμήματος με τα σύνορα του παραθύρου αποκοπής. Η σειρά είναι αυθαίρετη, παραδείγματος χάριν αριστερά, δεξιά, κάτω και άνωθεν. Αν κάποια από τις συντεταγμένες του ελεγχόμενου σημείου ευρίσκεται εκτός των συνόρων αυτών, τότε η συγκεκριμένη συντεταγμένη παίρνει την τιμή της συντεταγμένης του συνόρου ενώ η άλλη μεταβάλλεται αντίστοιχα χρησιμοποιώντας τις εξισώσεις 2.2 και 2.3.

Οι έλεγχοι και οι μεταβολές των συντεταγμένων των σημείων του ευθυγράμμου τμήματος ανά περίπτωση είναι οι ακόλουθοι:

- Αν $x_i < x_{\text{min}}$, τότε

$$x_i = x_{\text{min}}$$

$$y_i = y_0 + \frac{y_1 - y_0}{x_1 - x_0} \cdot (x_{\text{min}} - x_0)$$

- Αν $x_i > x_{\text{max}}$, τότε

$$x_i = x_{\text{max}}$$

$$y_i = y_0 + \frac{y_1 - y_0}{x_1 - x_0} \cdot (x_{\text{max}} - x_0)$$

- Αν $y_i < y_{\text{min}}$, τότε

$$y_i = y_{\text{min}}$$

$$x_i = x_0 + \frac{x_1 - x_0}{y_1 - y_0} \cdot (y_{\text{min}} - x_0)$$

- Αν $y_i > y_{\text{max}}$, τότε

$$y_i = y_{\text{max}}$$

$$x_i = x_0 + \frac{x_1 - x_0}{y_1 - y_0} \cdot (y_{\text{max}} - x_0)$$

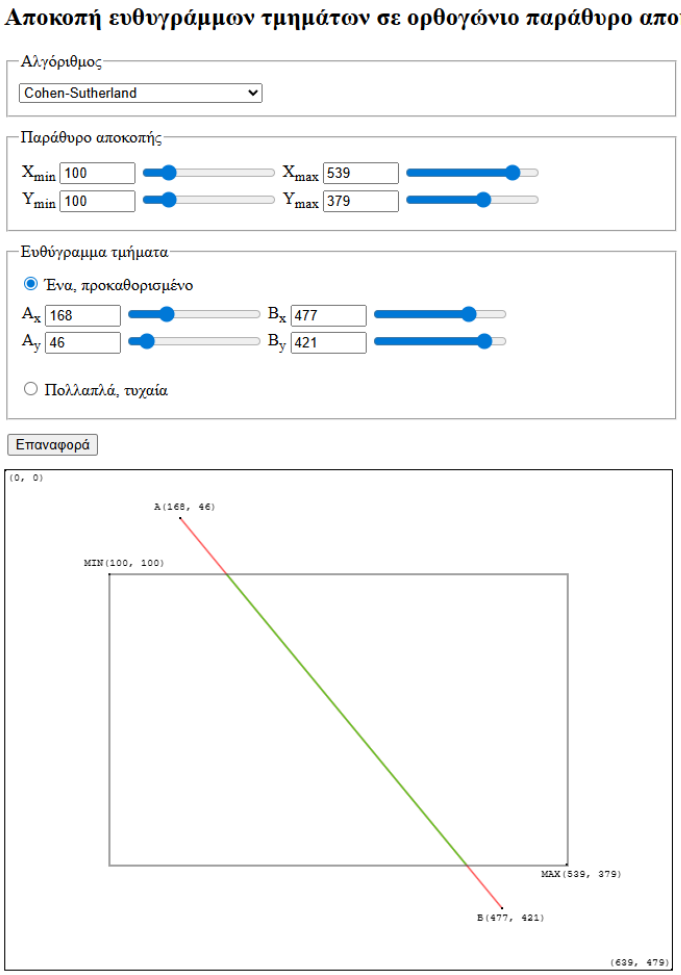
όπου $i = 0, 1$.

Το τρίτο και τελευταίο βήμα του αλγορίθμου σχεδιάζει μεταξύ των σημείων το αποκομμένο ευθύγραμμο τμήμα. Παρατηρώντας τις εξισώσεις 2.2 και 2.3 αναρωτιέται κανείς αν υπάρχουν περιπτώσεις όπου οι συντεταγμένες των σημείων του ευθυγράμμου τμήματος δημιουργούν διαίρεση με το μηδέν. Πράγματι, για την εξίσωση 2.2, όταν $x_0 = x_1$, και για την εξίσωση 2.3, όταν $y_0 = y_1$, κάτι τέτοιο δείχνει πιθανό. Ας εξετάσουμε την περίπτωση όπου $y_0 = y_1$, δηλαδή το ευθύγραμμο τμήμα είναι παράλληλο με τον οριζόντιο άξονα. Οι πιθανές θέσεις του τμήματος αυτού είναι τρεις: α) άνωθεν του παραθύρου αποκοπής, β) κάτωθεν του παραθύρου αποκοπής και γ) ενδιάμεσα του παραθύρου αποκοπής. Το πρώτο βήμα του αλγορίθμου απορρίπτει τις περιπτώσεις (α) και (β) εξ αρχής. Στην περίπτωση (γ), ο αλγόριθμος ενδέχεται να πραγματοποιήσει αποκοπή αλλά θα είναι μόνο εφ'όσον $x_i < x_{\text{min}}$ ή $x_i > x_{\text{max}}$, με $i : 0, 1$. Ο παρονομαστής όμως δεν επηρεάζεται από τα y_0 και y_1 άρα δεν πραγματοποιείται ποτέ διαίρεση με το μηδέν. Ομοίως, στην περίπτωση μίας κάθετης γραμμής όπου $x_0 = x_1$, δεν θα συνέβαινε ποτέ διαίρεση με το μηδέν εξαιτίας των αντίστοιχων ελέγχων που πραγματοποιεί ο αλγόριθμος.

Ο Κώδικας A.11 του Παραρτήματος A υλοποιεί τον προτεινόμενο αλγόριθμο.

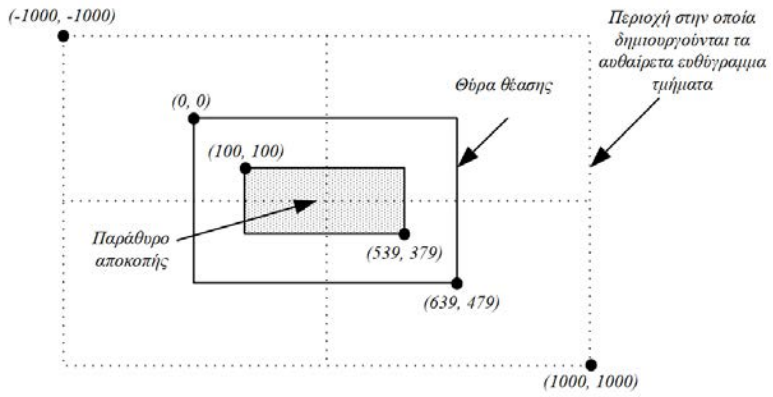
2.10 Αξιολόγηση όλων των αλγορίθμων

Για την πρακτική εφαρμογή και αξιολόγηση των προαναφερθέντων αλγορίθμων, όταν το παράθυρο αποκοπής είναι ορθογώνιο παραλληλόγραμμο, δημιουργήθηκε μία διαδικτυακή εφαρμογή βασισμένη σε HTML και Javascript η οποία είναι προσβάσιμη από τη διεύθυνση: https://matthes.sites.sch.gr/clipping (βλ. Σχ. 2.35).



Σχήμα 2.35: Η διαδικτυακή εφαρμογή για την αξιολόγηση των αλγορίθμων αποκοπής

Η διαδικασία αξιολόγησης ήταν η εξής: Κατά την εκτέλεση κάθε αλγορίθμου αποκόπηκε ένα μεγάλο πλήθος αυθαιρέτων ευθύγραμμων τμημάτων στις δύο διαστάσεις τα οποία δημιουργούνταν σε μία περιοχή που προσδιοριζόταν από τα σημεία $(-1000, -1000)$ και $(1000, 1000)$. Η θύρα θέασης είχε διάσταση 640×480 εικονοστοιχεία ενώ το παράθυρο αποκοπής βρισκόταν εντός αυτού και οριζόταν από τα σημεία $(100, 100)$ και $(539, 479)$, δηλαδή έχει διαστάσεις 440×380 (βλ. Σχ. 2.36).



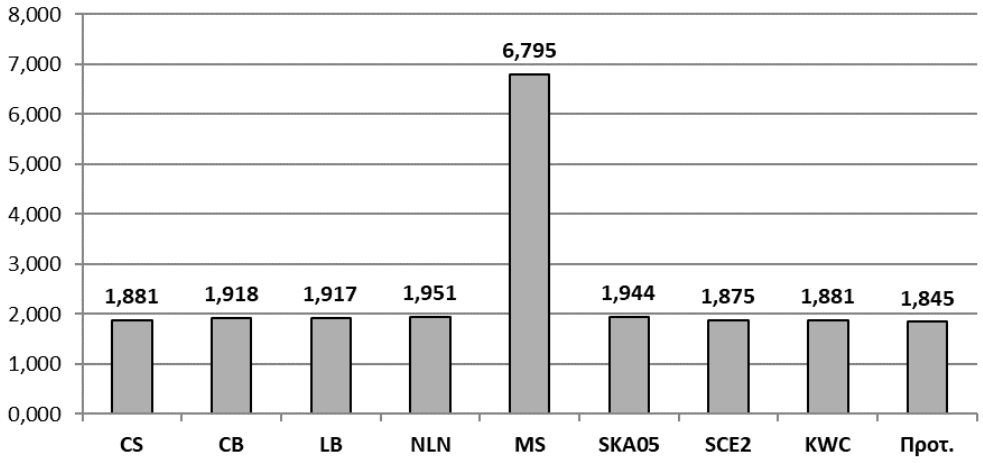
Σχήμα 2.36: Η περιοχή δημιουργίας αυθαίρετων τμημάτων και το παράθυρο αποκοπής για την αξιολόγηση των αλγορίθμων

Κάθε αλγόριθμος εκτελέστηκε δέκα φορές και ο χρόνος που χρειάστηκε για να αποκόψει και να σχεδιάσει τα ευθύγραμμα τμήματα σε κάθε εκτέλεση καταγράφηκε. Στο τέλος, ο μέσος χρόνος των δέκα εκτελέσεων υπολογίστηκε και αποτέλεσε το κριτήριο αξιολόγησης του αλγορίθμου ως προς την ταχύτητα. Όλες οι μετρήσεις έγιναν με την χρήση του ίδιου υπολογιστικού συστήματος του οποίου οι προδιαγραφές ήταν οι ακόλουθες: α) επεξεργαστής Intel Core i7-9750H @ 2.60 GHz CPU, β) μνήμη RAM 16 GB, γ) κάρτα γραφικών NVIDIA GeForce RTX 2070/8 GB GPU, δ) λειτουργικό σύστημα Windows 10 Pro 64 bit. Η ταχύτητα σύνδεσης στο διαδίκτυο δεν επηρέασε την ταχύτητα εκτέλεσης της εφαρμογής αφού η δεύτερη λειτουργεί σε τοπικό επίπεδο στον υπολογιστή μετά την έναρξή της.

Στον Πίνακα 2.3 φαίνονται οι χρόνοι για την αποκοπή 1.000.000 ευθυγράμμων τμημάτων ενώ στο Σχ. 2.37 απεικονίζονται με τη μορφή ραβδογράμματος οι μέσοι χρόνοι.

Exec.	**CS (sec)**	**CB (sec)**	**LB (sec)**	**NLN (sec)**	**MS (sec)**	**SKA05 (sec)**	**SCE2 (sec)**	**KWC (sec)**	**Προτ. (sec)**
1	1.910	1.956	1.981	1.925	6.869	1.956	1.925	1.900	1.849
2	1.960	1.979	1.933	1.924	6.651	1.969	1.897	1.908	1.860
3	1.892	1.906	1.891	1.957	6.788	1.931	1.945	1.909	1.808
4	1.903	1.931	1.941	1.970	6.789	1.980	1.896	1.873	1.869
5	1.860	1.798	1.901	1.958	6.907	1.958	1.885	1.863	1.811
6	1.891	1.897	1.858	1.956	6.794	1.950	1.731	1.802	1.853
7	1.855	1.894	1.962	1.940	6.701	1.944	1.845	1.899	1.861
8	1.875	1.937	1.879	1.980	6.931	1.875	1.855	1.912	1.819
9	1.842	1.966	1.915	1.986	6.866	1.918	1.931	1.884	1.876
10	1.820	1.916	1.907	1.916	6.649	1.963	1.838	1.863	1.847
Μ.Ο.	**1.881**	**1.918**	**1.917**	**1.951**	**6.795**	**1.944**	**1.875**	**1.881**	**1.845**

Πίνακας 2.3: Οι χρόνοι εκτέλεσης των αλγορίθμων για την αποκοπή 1.000.000 ευθυγράμμων τμημάτων



Σχήμα 2.37: Γραφική αναπαράσταση με τον μέσο χρόνο αποκοπής κάθε αλγορίθμου

Κάθε αλγόριθμος παρουσιάζει πλεονεκτήματα και μειονεκτήματα. Ο Cohen-Sutherland έχει πάρα πολύ καλή απόδοση, μόλον ότι είναι ο παλαιότερος. Ο Cyrus-Beck έχει μέτρια σχετικά απόδοση, χρησιμοποιεί περισσότερο πολύπλοκες μαθηματικές έννοιες αλλά είναι πιο ευέλικτος αφού μπορεί να εφαρμοστεί και σε οποιαδήποτε άλλο παράθυρο αποκοπής με πέντε ή περισσότερες κορυφές καθώς και να επεκταθεί στις τρεις διαστάσεις. Ο Liang-Barsky μοιάζει με τον Cyrus-Beck, χρησιμοποιεί επίσης πολύπλοκες μαθηματικές έννοιες, είναι ευέλικτος, μπορεί να επεκταθεί εύκολα στις τρεις διαστάσεις, είναι ταχύτερος από τον CB αλλά δεν μπορεί να εφαρμοστεί κυρτό παράθυρο αποκοπής με πέντε ή περισσότερες κορυφές. Ο Midpoint Subdivision έχει την χειρότερη απόδοση έναντι των υπολοίπων και αυτό οφείλεται στις διαδοχικές διαιρέσεις που πραγματοποιεί για να προσεγγίσει τα σημεία τομής. Ο Nicholl-Lee-Nicholl έχει σχετικά καλή απόδοση, όμως ο κώδικάς του είναι ιδιαίτερα μεγάλος σε έκταση αφού χρησιμοποιεί έναν μεγάλο αριθμό υποπεριπτώσεων και συναρτήσεων για τους μετασχηματισμούς των σημείων του ευθυγράμμου τμήματος. Ο Skala 2005 χρησιμοποιεί προχωρημένες μαθηματικές έννοιες και είναι πιο αργός. Ο S-Clip E^2 έχει πολύ καλύτερη απόδοση σε σχέση με τον Skala 2005 ενώ κάποιες διαδικασίες του έχουν απλοποιηθεί. Ο Kodituwakku–Wijeweere–Chamikara χρησιμοποιεί παρόμοια προσέγγιση με τον Cohen-Sutherland, είναι αρκετά γρήγορος, αλλά ο κώδικάς του περιλαμβάνει πολλές υποπεριπτώσεις και συγκρίσεις σχετικά με το αν το προς αποκοπή ευθύγραμμο τμήμα είναι οριζόντιο ή κάθετο με τους άξονες. Τέλος, ο προτεινόμενος είναι ο ταχύτερος και ο μικρότερος ως προς την έκταση του κώδικα.

2.11 Στατιστικός έλεγχος των αποτελεσμάτων

Για να διαπιστωθεί αν τα αριθμητικά αποτελέσματα και τα συμπεράσματα της προηγούμενης παραγράφου είναι έγκυρα και στατιστικώς ορθά, χρησιμοποιήθηκε η δοκιμασία αθροίσματος κατάταξης Wilcoxon. Πρόκειται για μία μη παραμετρική έκδοση της δοκιμασίας t δύο δειγμάτων. Η δοκιμασία t απαιτεί τα δείγματα:

1. Να είναι ανεξάρτητα το ένα από το άλλο.
2. Να έχουν ίση διακύμανση ή εξάπλωση.
3. Να είναι κανονικά κατανεμημένα.

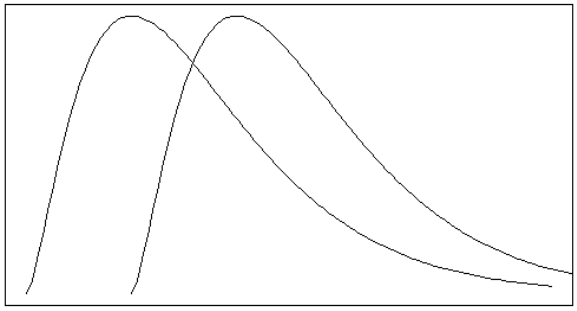
Το (1) πρέπει να ικανοποιείται όπως και να έχει, προκειμένου να πραγματοποιηθεί η δοκιμασία t δύο δειγμάτων. Όταν οι προϋποθέσεις (2) και (3) (ίση διακύμανση και κανονικότητα) δεν ικανοποιούνται αλλά τα δείγματα είναι μεγάλα, για παράδειγμα μεγαλύτερα από 30, τα αποτελέσματα είναι περίπου σωστά. Αλλά, όταν τα δείγματά μας είναι μικρά και τα δεδομένα μας δεν δείχνουν απολύτως φυσιολογικά, τότε η δοκιμασία t δύο δειγμάτων δεν θεωρείται αξιόπιστη.

Τη λύση στο πρόβλημα αυτό έρχεται να δώσει η δοκιμασία αθροίσματος κατάταξης Wilcoxon η οποία προϋποθέτει μόνο το (1) και το (2), δηλαδή την ανεξαρτησία και την ίση διακύμανση. Δεν προϋποθέτει ότι τα δεδομένα μας έχουν Κανονική Κατανομή[1], οι γνωστές κατανομές περιγράφονται με μαθηματικούς τύπους. Αυτοί οι τύποι έχουν παραμέτρους που υπαγορεύουν το σχήμα ή/και τη θέση της κατανομής. Για παράδειγμα, η διακύμανση και η μέση τιμή είναι οι δύο παράμετροι της Κανονικής Κατανομής που υπαγορεύουν το σχήμα και τη θέση της, αντίστοιχα. Επειδή λοιπόν η δοκιμασία αθροίσματος κατάταξης Wilcoxon δεν προϋποθέτει Κανονικές Κατανομές και δεν ασχολείται με παραμέτρους, ονομάζεται και μη παραμετρική δοκιμή.

Ενώ η μηδενική υπόθεση[2] της δοκιμασίας t δύο δειγμάτων είναι οι ίσοι μέσοι όροι, η μηδενική υπόθεση της δοκιμασίας αθροίσματος κατάταξης Wilcoxon είναι συνήθως οι ίσοι διάμεσοι. Εάν απορρίψουμε την μηδενική υπόθεση, τότε η μία εκ των δύο κατανομών μετατοπίζεται είτε προς τα αριστερά είτε προς τα δεξιά της άλλης. Εφ' όσον υποθέτουμε ότι οι κατανομές μας είναι ίσες, η απόρριψη της μηδενικής υπόθεσης σημαίνει ότι οι διάμεσοι των δύο δειγμάτων διαφέρουν, κάτι που είναι γνωστό και ως «μετατόπιση τοποθεσίας» (βλ. Σχ. 2.38).

[1] Η Κανονική Κατανομή ή αλλιώς η Κατανομή Gauss είναι μία συνεχής κατανομή πιθανότητας, συμμετρική και στις δυο πλευρές του μέσου όρου. Εξαρτάται αποκλειστικά από δυο παραμέτρους του συνόλου δεδομένων: την μέση τιμή και την τυπική απόκλιση.

[2] Η μηδενική υπόθεση είναι ο ισχυρισμός ότι το αποτέλεσμα που μελετάται δεν υπάρχει. Μπορεί επίσης να περιγραφεί ως η υπόθεση στην οποία δεν υπάρχει σχέση μεταξύ δύο συνόλων δεδομένων ή μεταβλητών που αναλύονται.



Σχήμα 2.38: Όμοιες κατανομές με διαφορετικές διάμεσους.

Το λογισμικό που χρησιμοποιήθηκε για την εφαρμογή της δοκιμασίας Wilcoxon είναι η "R" η οποία είναι γλώσσα προγραμματισμού ανοικτού κώδικα που παρέχει στον χρήστη τη δυνατότητα να πραγματοποιεί υπολογιστική στατιστική και γραφήματα.

Μετά την εισαγωγή των δεδομένων, συγκρίνοντας τον προτεινόμενο αλγόριθμο με τους υπόλοιπους, τα αποτελέσματα που ελήφθησαν απεικονίζονται στον Πίνακα 2.4:

Προτεινόμενος					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CS	77.5	0.0411700	0.001012839	0.063947746	0.03398584
CB	90	0.0015050	0.045	0.110	0.081
LB	96	0.0001299	0.039	0.104	0.0695
NLN	100	1.083e-05	0.079	0.131	0.1065
MS	100	1.083e-05	4.843	5.046	4.946
SKA05	99	2.165e-05	0.075	0.125	0.1015
SCE^2	72	0.1051	-0.008	0.077	0.0365
KWC	85	0.009082	0.01202550	0.06092114	0.0399986

Πίνακας 2.4: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon

Αν εξετάσουμε την πρώτη περίπτωση, δηλαδή την σύγκριση του προτεινόμενου αλγορίθμου με τον CS, και δεδομένου ότι p-value = 0.04117, άρα μικρότερη του 0.05, διαπιστώνουμε ότι απορρίπτεται σε επίπεδο σημαντικότητας 5%[3] η μηδενική υπόθεση, δηλαδή συμπεραίνουμε ότι ο προτεινόμενος έχει στατιστικά σημαντική διαφορά ως προς τον χρόνο υλοποίησης έναντι του CS. Το διάστημα εμπιστοσύνης 95% για τη διαφορά μεταξύ του χρόνου υλοποίησης του CS έναντι του προτεινόμενου δίνεται ως (0.001012839, 0.063947746). Δεδομένου ότι η διαφορά έχει σταθερά θετικό πρόσημο, συμπεραίνουμε ότι η στατιστικά σημαντική διαφορά που διαπιστώθηκε νωρίτερα,

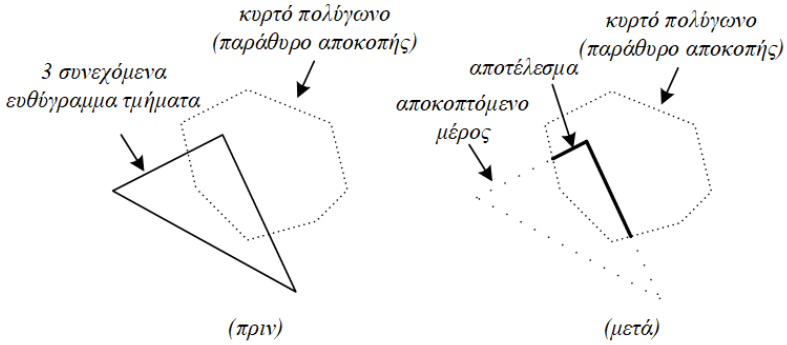
[3] Στις κοινωνικές επιστήμες έχει καθοριστεί ότι για να έχουμε στατιστικώς σημαντικά αποτελέσματα η πιθανότητα σφάλματος πρέπει να είναι το πολύ 0.05 (5%).

οδηγεί στο συμπέρασμα ότι ο χρόνος του αλγορίθμου CS είναι στατιστικώς σημαντικά μεγαλύτερος από τον χρόνο του προτεινόμενου αλγορίθμου. Ομοίως, εξάγουμε τα ίδια συμπεράσματα και για τις υπόλοιπες συγκρίσεις του προτεινόμενου αλγορίθμου με τους υπόλοιπους πλην της περίπτωσης με τον SCE^2.

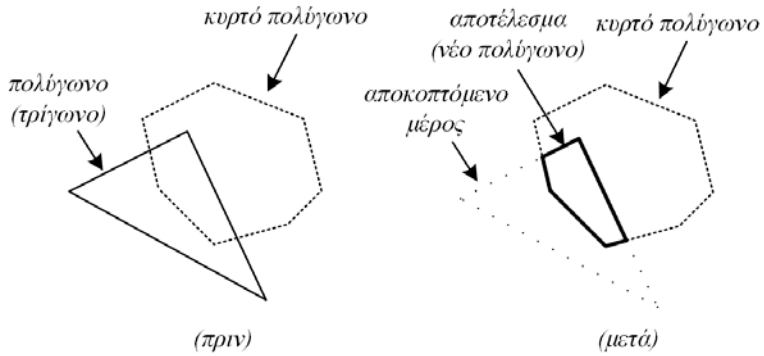
Στο ζεύγος SCE^2-προτεινόμενος παρατηρείται στατιστικώς μη σημαντική διαφορά. Δεδομένου ότι p-value = 0.1051, άρα μεγαλύτερη του 0.05, δεν απορρίπτεται σε επίπεδο σημαντικότητας 5% η μηδενική υπόθεση, δηλαδή συμπεραίνουμε ότι ο προτεινόμενος αλγόριθμος δεν έχει στατιστικά σημαντική διαφορά ως προς τον χρόνο υλοποίησης έναντι του SCE^2. Όμως, το διάστημα εμπιστοσύνης 95% για τη διαφορά μεταξύ του χρόνου υλοποίησης του SCE^2 έναντι του προτεινόμενου δίνεται ως (-0.008, 0.077). Δεδομένου ότι η διαφορά δεν έχει σταθερό πρόσημο, συμπεραίνουμε ότι ο χρόνος του SCE^2 δεν είναι στατιστικώς σημαντικά μεγαλύτερος από τον χρόνο του προτεινόμενος αλγορίθμου. Βεβαίως, επειδή το μεγαλύτερο μέρος του διαστήματος βρίσκεται στον θετικό ημιάξονα, υπάρχουν ενδείξεις ότι ο χρόνος του προτεινόμενου τείνει να είναι μικρότερος (αλλά όχι στατιστικώς σημαντικά μικρότερος). Το συμπέρασμα αυτό ενισχύεται από το γεγονός ότι η σημειακή εκτίμηση της διαφοράς μεταξύ του χρόνου του SCE^2 έναντι του χρόνου του προτεινόμενου είναι θετική και ίση με 0.0365.

3. ΑΠΟΚΟΠΗ ΕΥΘΥΓΡΑΜΜΟΥ ΤΜΗΜΑΤΟΣ ΑΠΟ ΚΥΡΤΟ ΠΟΛΥΓΩΝΟ ΣΤΙΣ ΔΥΟ ΔΙΑΣΤΑΣΕΙΣ

Εδώ, το προς αποκοπή αντικείμενο είναι ένα ευθύγραμμο τμήμα ενώ το παράθυρο αποκοπής έχει σχήμα κυρτού πολύγωνου. Ο όρος αυτός δεν πρέπει να συγχέεται με την έννοια της «αποκοπής πολυγώνων», όπου τόσο το αποκόπτον αντικείμενο όσο και το αποκοπτόμενο, είναι πολύγωνα. Στην πρώτη περίπτωση, ένα ή περισσότερα ευθύγραμμα τμήματα αποκόπτονται διαδοχικά σε ένα κυρτό παράθυρο αποκοπής και το αποτέλεσμα είναι ένα ή περισσότερα ευθύγραμμα τμήματα (βλ. Σχ. 3.1) [Rog97]. Στη δεύτερη περίπτωση, ένα πολύγωνο (κυρτό ή μη) αποκόπτεται σε ένα κυρτό παράθυρο αποκοπής που το σχήμα του είναι πολύγωνο και το αποτέλεσμα είναι ένα νέο πολύγωνο (βλ. Σχ. 3.2).



Σχήμα 3.1: Αποκοπή τριών συνεχόμενων ευθύγραμμων τμημάτων τα οποία σχηματίζουν ένα τρίγωνο. Το αποτέλεσμα είναι ευθύγραμμα τμήματα



Σχήμα 3.2: Αποκοπή ενός πολυγώνου (τρίγωνο) σε ένα παράθυρο αποκοπής σχήματος κυρτού πολυγώνου. Το αποτέλεσμα είναι ένα νέο πολύγωνο

Είναι προφανές ότι οι αλγόριθμοι αποκοπής ευθυγράμμων τμημάτων δεν λειτουργούν στην περίπτωση της αποκοπής μεταξύ πολυγώνων και, κατά συνέπεια, χρειάζονται ειδικοί αλγόριθμοι για τον σκοπό αυτό όπως οι Weiler - Arlington [WA77], Sutherland–Hodgman [SH74], Greiner–Hormann [GH98], Vatti [Vat92]. Το πρόβλημα με τους συγκεκριμένους αλγόριθμους εστιάζεται στο ότι χειρίζονται το αποκοπτόμμενο πολύγωνο ως ένα σύνολο από διαδοχικά ευθύγραμμα τμήματα. Ένας αλγόριθμος αποκοπής πολυγώνων πρέπει να υπολογίζει την τομή της επιφάνειας του αποκοπτόμενου πολυγώνου με την επιφάνεια του πολυγώνου αποκοπής [Θεο+15].

3.1 Αλγόριθμος Cyrus-Beck

Ο αλγόριθμος Cyrus-Beck δεν αλλάζει τρόπο λειτουργίας, όταν το παράθυρο αποκοπής είναι κυρτό πολύγωνο και ακολουθεί την λογική που περιγράφηκε στο προηγούμενο κεφάλαιο. Ο Κώδικας A.2 του Παραρτήματος A μπορεί να χρησιμοποιηθεί ως έχει.

3.2 Αλγόριθμος Skala 2005

Ο αλγόριθμος Skala 2005, στην εκδοχή του για αποκοπή ευθυγράμμου τμήματος σε κυρτό πολύγωνο, διαφέρει κυρίως ως προς τον υπολογισμό των τιμών TAB1 και TAB2. Όταν το παράθυρο αποκοπής έχει πέντε ή περισσότερες κορυφές, τότε οι τιμές αυτές δεν μπορούν να αναζητηθούν στον προϋπολογισμένο πίνακα και υπολογίζονται σε πραγματικό χρόνο πριν ξεκινήσει η αποκοπή.

Ο Κώδικας A.12 του Παραρτήματος A υλοποιεί τον αλγόριθμο Skala 2005 για την αποκοπή ευθυγράμμου τμήματος από κυρτό πολύγωνο.

3.3 Αλγόριθμος S-Clip E^2

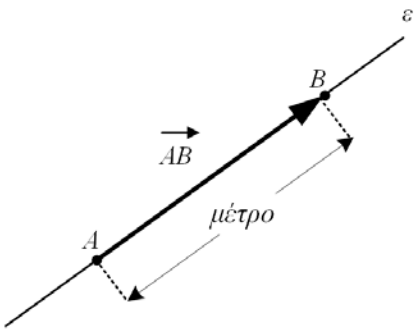
Ομοίως με τον Skala 2005, ο αλγόριθμος S-Clip E^2, στην εκδοχή του για αποκοπή ευθυγράμμου τμήματος σε κυρτό πολύγωνο, διαφέρει κυρίως ως προς τον υπολογισμό των τιμών TAB1 και TAB2. Όταν το παράθυρο αποκοπής έχει πέντε ή περισσότερες κορυφές, τότε οι τιμές αυτές δεν μπορούν να αναζητηθούν από τον προϋπολογισμένο πίνακα και υπολογίζονται σε πραγματικό χρόνο πριν ξεκινήσει η αποκοπή.

Ο Κώδικας A.13 του Παραρτήματος A υλοποιεί τον αλγόριθμο S-Clip E^2 για την αποκοπή ευθυγράμμου τμήματος από κυρτό πολύγωνο.

3.4 Προτεινόμενος αλγόριθμος

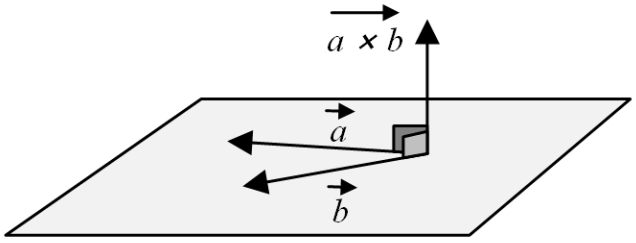
Η βασική λογική του προτεινόμενου αλγορίθμου του προηγούμενου κεφαλαίου για την αποκοπή ενός ευθυγράμμου τμήματος σε ορθογώνιο παράθυρο αποκοπής εφαρμόζεται και όταν το παράθυρο έχει σχήμα κυρτού πολυγώνου. Έτσι, ο [MD22] εκμεταλλεύεται τις ιδιότητες του εξωτερικού γινομένου δύο διανυσμάτων. Βασικό χαρακτηριστικό του είναι ότι οι υπολογισμοί των σημείων τομής μεταξύ του ευθυγράμμου τμήματος και των ακμών γίνεται μόνο στις ακμές που αυτοί απαιτούνται.

Γνωρίζουμε ότι ένα διάνυσμα προσδιορίζεται από δύο σημεία και έχει ως βασικά χαρακτηριστικά το μέτρο (μήκος) και τη φορά (κατεύθυνση) (βλ. Σχ. 3.3).



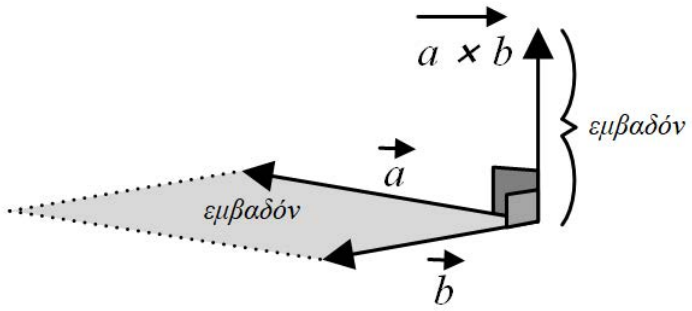
Σχήμα 3.3: Δύο από τα χαρακτηριστικά του διανύσματος $\overrightarrow{AB}$: η φορά και το μέτρο

Μεταξύ δύο διανυσμάτων $\vec{a}$ και $\vec{b}$ τα οποία βρίσκονται στον τριδιάστατο χώρο μπορεί να εφαρμοστεί η πράξη που ονομάζεται *εξωτερικό γινόμενο* (ή αλλιώς *διανυσματικό γινόμενο*) και συμβολίζεται με το σύμβολο $\times$. Το εξωτερικό γινόμενο $\vec{a} \times \vec{b}$ είναι ένα τρίτο διάνυσμα το οποίο είναι κάθετο προς αυτά τα δύο διανύσματα $\vec{a}$ και $\vec{b}$ και κατά συνέπεια κάθετο στο επίπεδο που τα περιέχει (βλ. Σχ. 3.4).



Σχήμα 3.4: Το εξωτερικό γινόμενο δύο διανυσμάτων είναι ένα τρίτο διάνυσμα το οποίο είναι κάθετο προς αυτά τα δύο

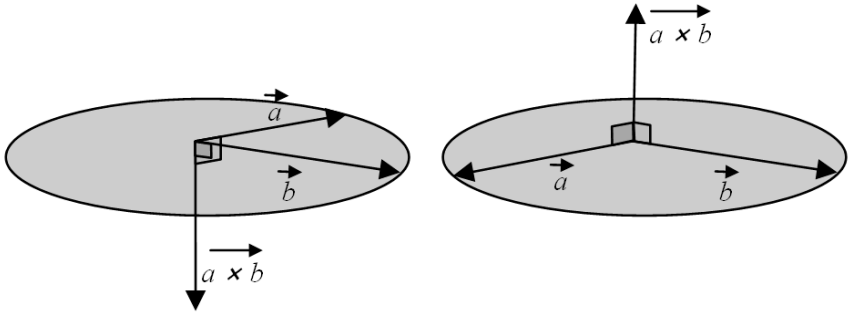
Το μέτρο του εξωτερικού γινομένου ισοδυναμεί με το εμβαδόν του παραλληλογράμμου που σχηματίζουν τα διανύσματα $\vec{a}$ και $\vec{b}$ (βλ. Σχ. 3.5).



Σχήμα 3.5: Το εξωτερικό γινόμενο δύο διανυσμάτων είναι ένα τρίτο διάνυσμα το οποίο είναι κάθετο προς αυτά τα δύο

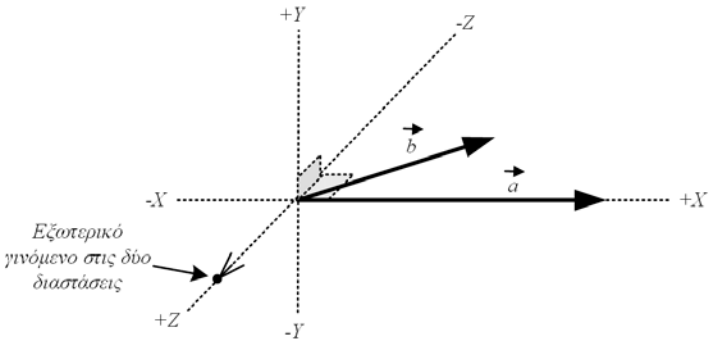
Το εξωτερικό γινόμενο έχει τα ακόλουθα χαρακτηριστικά:

- Έχει μηδενικό μέτρο (μήκος), όταν τα διανύσματα $\vec{a}$ και $\vec{b}$ έχουν ίδια ή αντίθετη φορά.
- Η μέγιστη τιμή του μέτρου του είναι όταν τα διανύσματα $\vec{a}$ και $\vec{b}$ σχηματίζουν ορθή γωνία μεταξύ τους.
- Η φορά του εξαρτάται από την γωνία που σχηματίζουν τα διανύσματα $\vec{a}$ και $\vec{b}$ (βλ. Σχ. 3.6).



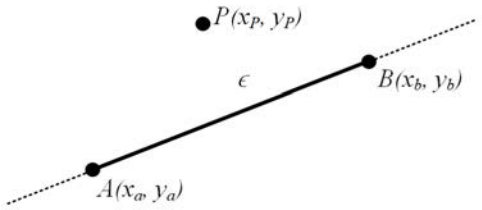
Σχήμα 3.6: Η φορά του εξωτερικού γινομένου εξαρτάται από την γωνία που σχηματίζουν τα διανύσματα $\vec{a}$ και $\vec{b}$

Από τα παραπάνω γίνεται αντιληπτό ότι το εξωτερικό γινόμενο δεν υφίσταται στις δύο διαστάσεις. Όμως, σε ορισμένες περιπτώσεις, είναι βολικό να υποθέσουμε ότι υπάρχει στον διδιάστατο χώρο με την παραδοχή ότι η συντεταγμένη z των διανυσμάτων είναι 0. Το αποτέλεσμα του εξωτερικού γινομένου είναι ένα βαθμωτό, δηλαδή ένα μονόμετρο μέγεθος, το οποίο μπορεί να θεωρηθεί ως ένα σημείο που βρίσκεται κάθετα στο επίπεδο $X - Y$. Από το πρόσημο του βαθμωτού μπορούμε να καταλάβουμε την θέση του ενός διανύσματος σε σχέση το άλλο, δηλαδή αν βρίσκεται «αριστερά» ή «δεξιά» του (βλ. Σχ. 3.7).



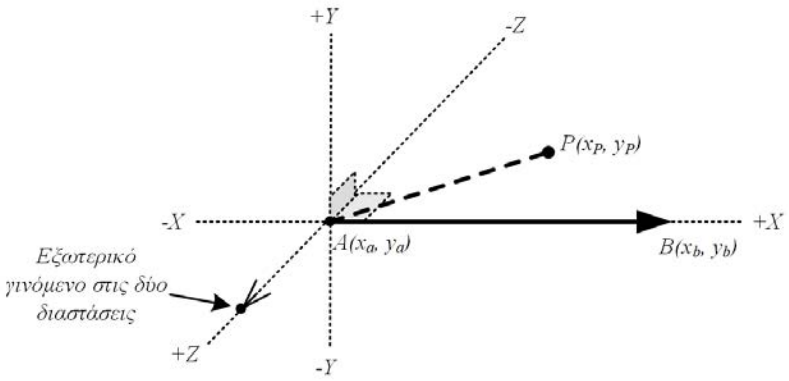
Σχήμα 3.7: Από το πρόσημο του βαθμωτού μπορούμε να καταλάβουμε την θέση του ενός διανύσματος σε σχέση το άλλο

Με βάση τα παραπάνω, μπορούμε να προσδιορίσουμε εάν ένα σημείο $P(x_p, y_p)$ βρίσκεται αριστερά, δεξιά ή επί ενός ευθυγράμμου τμήματος ϵ το οποίο προσδιορίζεται από τα σημεία $A(x_a, y_a)$ και $B(x_b, y_b)$ (βλ. Σχ. 3.8).



Σχήμα 3.8: Με το εξωτερικό γινόμενο μπορούμε να προσδιορίσουμε εάν το σημείο P βρίσκεται αριστερά, δεξιά ή επάνω στο ευθύγραμμο τμήμα ϵ

Θεωρώντας ότι όλα τα σημεία A, B και P σχηματίζουν δύο διανύσματα, το διάνυσμα $\overrightarrow{AP}$ και το διάνυσμα $\overrightarrow{AB}$, με κοινό σημείο την αρχή των αξόνων, ελέγχουμε το πρόσημο του εξωτερικού γινομένου. Θετική ή αρνητική τιμή στο πρόσημο σημαίνει ότι το σημείο P βρίσκεται δεξιά ή αριστερά του ευθυγράμμου τμήματος ϵ και αντίστοιχα μηδενική τιμή σημαίνει ότι το σημείο βρίσκεται επί του ϵ (βλ. Σχ. 3.9).



Σχήμα 3.9: Θετική ή αρνητική τιμή στο πρόσημο σημαίνει ότι το σημείο P βρίσκεται αριστερά ή δεξιά του διανύσματος $\overrightarrow{AB}$ αντίστοιχα ενώ μηδενική τιμή σημαίνει ότι το σημείο βρίσκεται επί αυτού

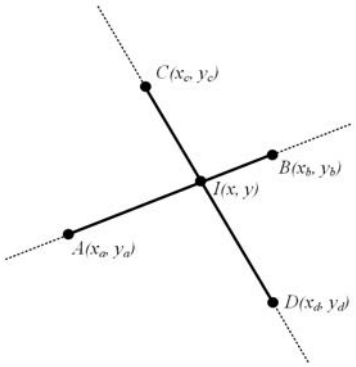
Βασιζόμενοι στο Σχ. 3.9, το εξωτερικό γινόμενο του διανύσματος $\overrightarrow{AB}$ με το διάνυσμα $\overrightarrow{AP}$ είναι:

$$\overrightarrow{AB} \times \overrightarrow{AP} = \begin{vmatrix} (x_b - x_a) & (y_b - y_a) \\ (x_p - x_a) & (y_p - y_a) \end{vmatrix} \Rightarrow$$
$$\overrightarrow{AB} \times \overrightarrow{AP} = (x_b - x_a) \cdot (y_p - y_a) - (y_b - y_a) \cdot (x_p - x_a)$$

Χρησιμοποιώντας τον κανόνα του δεξιού χεριού, εάν το εξωτερικό γινόμενο έχει θετικό πρόσημο, τότε το σημείο P είναι αριστερά από το διάνυσμα $\overrightarrow{AB}$ και κατά συνέπεια αριστερά του ϵ. Ομοίως, εάν έχει αρνητικό πρόσημο τότε το σημείο P βρίσκεται στα δεξιά του, ενώ εάν το εξωτερικό γινόμενο είναι μηδέν τότε το σημείο βρίσκεται επί του ευθυγράμμου τμήματος.

Ο Κωδ. A.14 του Παραρτήματος A παρουσιάζει μία συνάρτηση η οποία υπολογίζει την θέση ενός σημείου σε σχέση με ένα ευθύγραμμο τμήμα κάνοντας χρήση του εξωτερικού γινομένου.

Το εξωτερικό γινόμενο μπορεί επιπλέον να χρησιμοποιηθεί για τον προσδιορισμό του σημείου τομής μεταξύ δύο ευθυγράμμων τμημάτων. Ας υποθέσουμε ότι δύο σημεία $A(x_a, y_a)$ και $B(x_b, y_b)$ προσδιορίζουν το ευθύγραμμο τμήμα AB και δύο σημεία $C(x_c, y_c)$ και $D(x_d, y_d)$ προσδιορίζουν το ευθύγραμμο τμήμα CD στον διδιάστατο χώρο. Τα ευθύγραμμα τμήματα αυτά έχουν σημείο τομής το $I(x, y)$ (βλ. Σχ. 3.10).

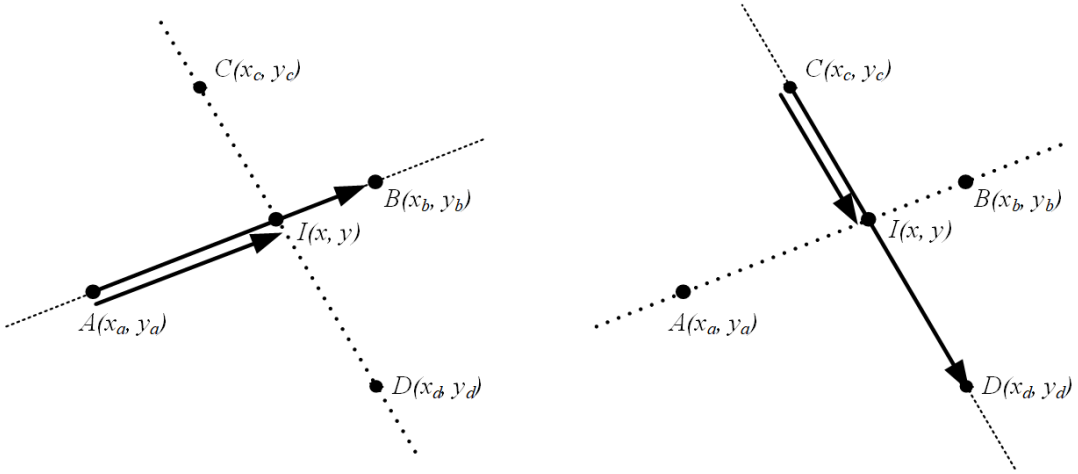


Σχήμα 3.10: Το σημείο τομής I των ευθυγράμμων τμημάτων AB και CD.

Όσον αφορά το σημείο τομής I:

- Το εξωτερικό γινόμενο του διανύσματος $\overrightarrow{AI}$ με το διάνυσμα $\overrightarrow{AB}$ είναι μηδέν.
- Το εξωτερικό γινόμενο του διανύσματος $\overrightarrow{CI}$ με το διάνυσμα $\overrightarrow{CD}$ είναι μηδέν.

Αυτό απεικονίζεται καλύτερα στα Σχ. 3.11.



Σχήμα 3.11: Το εξωτερικό γινόμενο των διανυσμάτων $\overrightarrow{AI}$ και $\overrightarrow{AB}$ και το εξωτερικό γινόμενο των διανυσμάτων $\overrightarrow{CI}$ και $\overrightarrow{CD}$ είναι μηδέν

Για τα διανύσματα $\overrightarrow{AI}$ και $\overrightarrow{AB}$ του Σχ. 3.11, το εξωτερικό γινόμενο είναι μηδέν οπότε:

$$\overrightarrow{AI} \times \overrightarrow{AB} = 0 \Rightarrow$$

$$\begin{vmatrix} (x - x_a) & (y - y_a) \\ (x_b - x_a) & (y_b - y_a) \end{vmatrix} = 0 \Rightarrow$$

$$(x - x_a) \cdot (y_b - y_a) - (x_b - x_a) \cdot (y - y_a) = 0 \Rightarrow$$

$$x \cdot (y_b - y_a) - x_a \cdot (y_b - y_a) - y \cdot (x_b - x_a) + y_a \cdot (x_b - x_a) = 0 \Rightarrow$$

$$(y_b - y_a) \cdot x - (x_b - x_a) \cdot y = x_a \cdot (y_b - y_a) - y_a \cdot (x_b - x_a)$$

Για λόγους απλότητας, συμβολίζουμε τις διαφορές $(x_b - x_a)$ και $(y_b - y_a)$ ως

$$dx_1 = (x_b - x_a) \quad \text{and} \quad dy_1 = (y_b - y_a),$$

οπότε η προηγούμενη εξίσωση γίνεται

$$dy_1 \cdot x - dx_1 \cdot y = x_a \cdot dy_1 - y_a \cdot dx_1. \tag{3.1}$$

Ομοίως, για τα διανύσματα $\overrightarrow{CI}$ και $\overrightarrow{CD}$ του Σχ. 3.11, το εξωτερικό γινόμενο είναι μηδέν οπότε:

$$x \cdot (y_d - y_c) - y \cdot (x_d - x_c) = x_c \cdot (y_d - y_c) - y_c \cdot (x_d - x_c)$$

Για λόγους απλότητας, συμβολίζουμε τις διαφορές $(x_d - x_c)$ και $(y_d - y_c)$ ως:

$$dx_2 = (x_d - x_c) \quad \text{and} \quad dy_2 = (y_d - y_c)$$

Από τα παραπάνω έχουμε:

$$dy_2 \cdot x - dx_2 \cdot y = x_c \cdot dy_2 - y_c \cdot dx_2. \tag{3.2}$$

Στις εξισώσεις (3.1), (3.2) υπάρχουν δύο άγνωστοι οπότε με χρήση οριζουσών:

$$D = \begin{vmatrix} dy_1 & -dx_1 \\ dy_2 & -dx_2 \end{vmatrix} = dy_2 \cdot dx_1 - dy_1 \cdot dx_2$$

$$\begin{aligned} DX &= \begin{vmatrix} (x_a \cdot dy_1 - y_a \cdot dx_1) & -dx_1 \\ (x_c \cdot dy_2 - y_c \cdot dx_2) & -dx_2 \end{vmatrix} \\ &= (x_c \cdot dy_2 - y_c \cdot dx_2) \cdot dx_1 - (x_a \cdot dy_1 - y_a \cdot dx_1) \cdot dx_2 \end{aligned}$$

$$\begin{aligned} DY &= \begin{vmatrix} dy_1 & (x_a \cdot dy_1 - y_a \cdot dx_1) \\ dy_2 & (x_c \cdot dy_2 - y_c \cdot dx_2) \end{vmatrix} \\ &= (x_c \cdot dy_2 - y_c \cdot dx_2) \cdot dy_1 - (x_a \cdot dy_1 - y_a \cdot dx_1) \cdot dy_2 \end{aligned}$$

Επιλύοντας ως προς x και y:

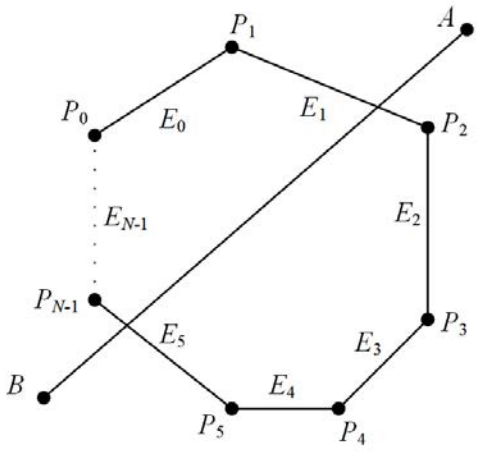
$$x = \frac{DX}{D} = \frac{(x_c \cdot dy_2 - y_c \cdot dx_2) \cdot dx_1 - (x_a \cdot dy_1 - y_a \cdot dx_1) \cdot dx_2}{dy_2 \cdot dx_1 - dy_1 \cdot dx_2} \tag{3.3}$$

$$y = \frac{DY}{D} = \frac{(x_c \cdot dy_2 - y_c \cdot dx_2) \cdot dy_1 - (x_a \cdot dy_1 - y_a \cdot dx_1) \cdot dy_2}{dy_2 \cdot dx_1 - dy_1 \cdot dx_2} \tag{3.4}$$

Ο Κωδ. A.15 του Παραρτήματος A παρουσιάζει μία συνάρτηση η οποία υπολογίζει το σημείο τομής μεταξύ δύο ευθυγράμμων τμημάτων υλοποιώντας τις παραπάνω μαθηματικές εξισώσεις.

Ας θεωρήσουμε ένα παράθυρο αποκοπής σε σχήμα κυρτού πολυγώνου με N κορυφές οι οποίες δίνονται σύμφωνα με την φορά των δεικτών του ωρολογίου. Το πρώτο σημείο του πολυγώνου είναι το $P_0(x_0, y_0)$ και το τελευταίο το $P_{N-1}(x_{N-1}, y_{N-1})$. Επίσης, υπάρχουν N ακμές με ονόματα E_0 έως E_{N-1}. Το προς αποκοπή ευθύγραμμο τμήμα προσδιορίζεται από τα σημεία $A(x_a, y_a)$ και $B(x_b, y_b)$ (βλ. Σχ. 3.12). Επειδή η διάταξη των κορυφών είναι σύμφωνα με τη φορά των δεικτών του ωρολογίου, εάν διατρέξουμε τις ακμές από E_0 έως E_{N-1}, τότε τα σημεία

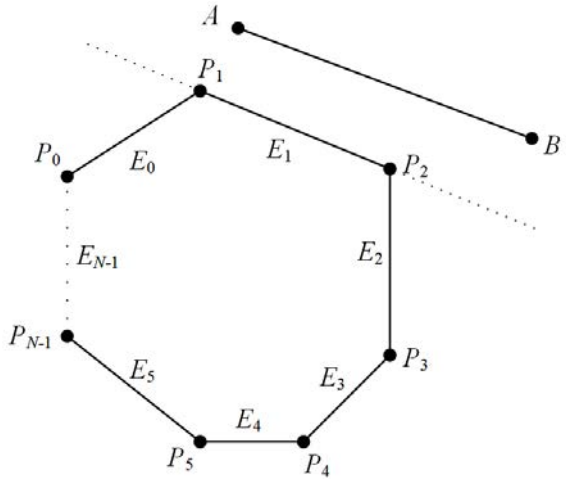
που βρίσκονται αριστερά αυτών έχουν θετικό πρόσημο όσον αφορά το εξωτερικό γινόμενο, τα σημεία που βρίσκονται δεξιά έχουν αρνητικό, ενώ τα σημεία που βρίσκονται επί της αντίστοιχης ακμή έχουν εξωτερικό γινόμενο ίσο με το μηδέν.



Σχήμα 3.12: Παράθυρο αποκοπής σχήματος κυρτού πολυγώνου με N κορυφές και N ακμές μαζί με ένα προς αποκοπή ευθύγραμμο τμήμα

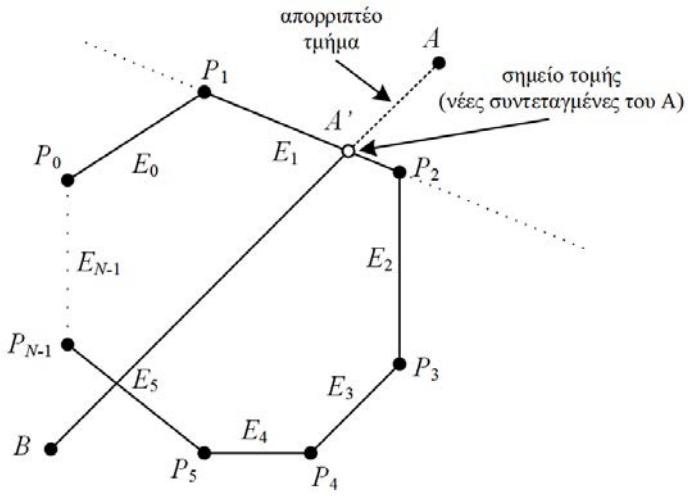
Για την αποκοπή του ευθυγράμμου τμήματος AB, ο αλγόριθμος πρέπει να ελέξει την θέση των σημείων A και B σε σχέση με την κάθε ακμή. Σε κάθε έλεγχο, ο αλγόριθμος πρέπει να προσδιορίσει εάν κάθε σημείο του ευθυγράμμου τμήματος είναι αριστερά ή δεξιά της ακμής ή εάν βρίσκεται επί αυτής.

Στην περίπτωση που και τα δύο σημεία A και B είναι αριστερά της ελεγχόμενης ακμής, τότε το ευθύγραμμο τμήμα απορρίπτεται, διότι βρίσκεται εξ ολοκλήρου εκτός της περιοχής αποκοπής (βλ. Σχ. 3.13).



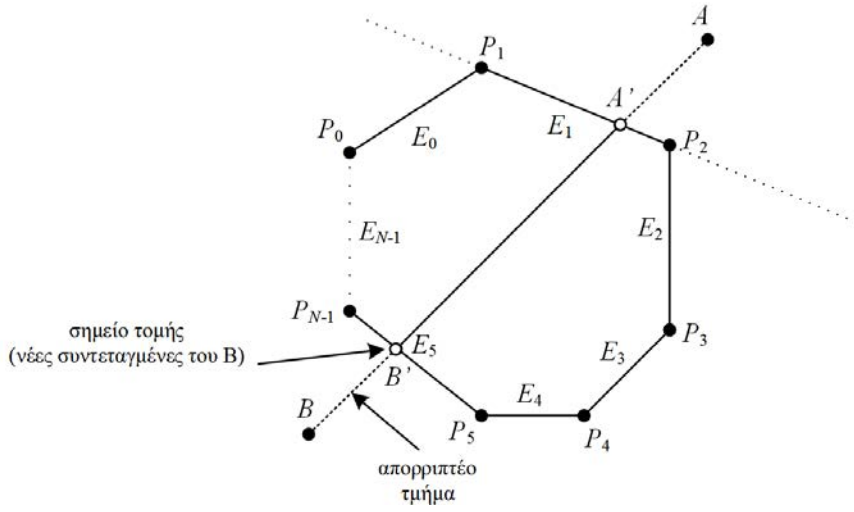
Σχήμα 3.13: Αν και τα δύο σημεία A και B είναι αριστερά της ελεγχόμενης ακμής, τότε το ευθύγραμμο τμήμα απορρίπτεται διότι, ευρίσκεται εξ ολοκλήρου εκτός

Αν το σημείο A είναι αριστερά της ελεγχόμενης ακμής και το σημείο B είναι δεξιά, τότε ο αλγόριθμος υπολογίζει το σημείο τομής μεταξύ της ευθείας και της συγκεκριμένης ακμής. Στη συνέχεια, αντικαθιστά τις συντεταγμένες του σημείου A με εκείνες του σημείου τομής (βλ. Σχ. 3.14).



Σχήμα 3.14: Οι συντεταγμένες του σημείου τομής αντικαθιστούν τις συντεταγμένες του σημείου A που βρίσκεται εκτός του κυρτού πολυγώνου

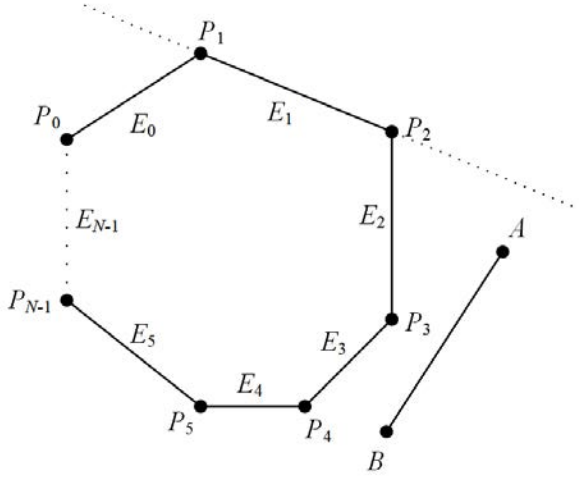
Ομοίως, εάν το σημείο B είναι αριστερά της ελεγχόμενης ακμής και το σημείο Α είναι δεξιά, τότε ο αλγόριθμος υπολογίζει το σημείο τομής μεταξύ του ευθυγράμμου τμήματος και της συγκεκριμένης ακμής. Στη συνέχεια, αντικαθιστά τις συντεταγμένες του σημείου B με εκείνες του σημείου τομής (βλ. Σχ. 3.15).



Σχήμα 3.15: Οι συντεταγμένες του σημείου τομής αντικαθιστούν τις συντεταγμένες του σημείου B που βρίσκεται εκτός του κυρτού πολυγώνου

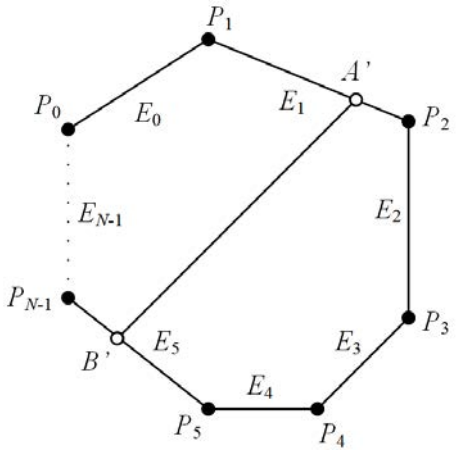
Αν και τα δύο σημεία A και B είναι δεξιά της ελεγχόμενης ακμής ή εάν ένα από τα δύο είναι επί της ακμής και

το άλλο δεξιά αυτής, τότε ο αλγόριθμος συνεχίζει τον έλεγχο στην επόμενη ακμή (βλ. Σχ. 3.16).



Σχήμα 3.16: Αν και τα δύο σημεία A και B είναι δεξιά της ελεγχόμενης ακμής, τότε ο αλγόριθμος συνεχίζει τον έλεγχο στην επόμενη ακμή

Μετά το πέρας των ελέγχων σε όλες τις ακμές, ο αλγόριθμος σχεδιάζει το αποκομμένο ευθύγραμμο τμήμα μεταξύ των νέων συντεταγμένων των σημείων A και B (βλ. Σχ. 3.17).



Σχήμα 3.17: Ο αλγόριθμος σχεδιάζει το αποκομμένο ευθύγραμμο τμήμα μεταξύ των νέων σημείων A και B

Συνοπτικά, τα βήματα του αλγορίθμου είναι:

1. Σε σχέση με την ακμή του πολυγώνου η οποία προσδιορίζεται από τις κορυφές $P_i(x_i, y_i)$ και $P_{i+1}(x_{i+1}, y_{i+1})$ με $i: 0, 1, \ldots, N-1$, έλεγξε τη σχετική θέση των σημείων $A(x_a, y_a)$ και $B(x_b, y_b)$ τα οποία προσδιορίζουν το ευθύγραμμο τμήμα. Τα σημεία A και B μπορεί να βρίσκονται αριστερά, δεξιά ή επί της ακμής.

2. Αν και τα δύο σημεία είναι αριστερά της ελεγχόμενης ακμής, τότε σταμάτησε, το ευθύγραμμο τμήμα ευρίσκεται εξ ολοκλήρου εκτός του κυρτού πολυγώνου.

3. Αν μόνο το σημείο A είναι αριστερά της ελεγχόμενης ακμής, τότε υπολόγισε το σημείο τομής μεταξύ του ευθυγράμμου τμήματος και της συγκεκριμένης ακμής. Αντικατέστησε τις συντεταγμένες του σημείου A με τις συντεταγμένες του σημείου τομής και επανέλαβε από το Βήμα 1 με την επόμενη ακμή.

4. Αν μόνο το σημείο B είναι αριστερά της ελεγχόμενης ακμής, τότε υπολόγισε το σημείο τομής μεταξύ του ευθυγράμμου τμήματος και της συγκεκριμένης ακμής. Αντικατέστησε τις συντεταγμένες του σημείου B με τις συντεταγμένες του σημείου τομής και επανέλαβε από το Βήμα 1 με την επόμενη ακμή.

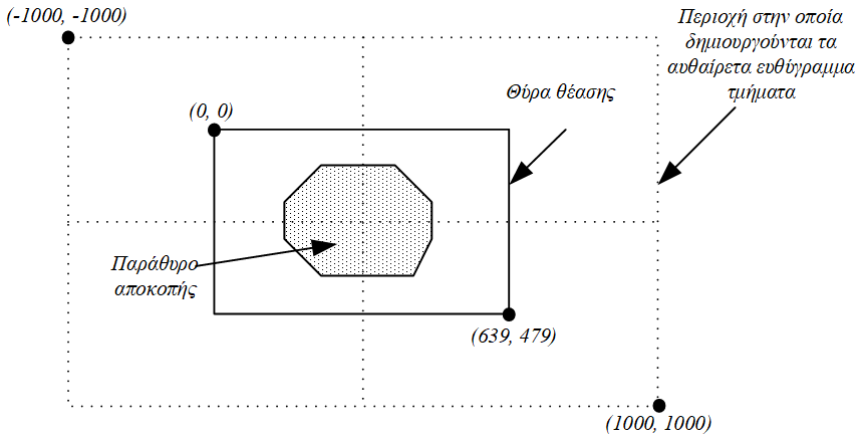
5. Αν και τα δύο σημεία A και B είναι δεξιά της ελεγχόμενης ακμής ή αν ένα από τα δύο είναι επάνω στην ακμή και το άλλο δεξιά αυτής, τότε επανέλαβε από το Βήμα 1 με την επόμενη ακμή ή σταμάτησε εάν έχεις ελέγξει και τις N ακμές.

6. Σχεδίασε το αποκομμένο ευθύγραμμο τμήμα μεταξύ των νέων σημείων A και B.

Ο Κωδ. A.16 του Παραρτήματος A αποτελεί την υλοποίηση του αλγορίθμου για αποκοπή ευθυγράμμου τμήματος σε κυρτό πολύγωνο.

3.5 Αξιολόγηση όλων των αλγορίθμων

Προκειμένου να προσδιορίσουμε την αποτελεσματικότητα των παραπάνω αλγορίθμων οι οποίοι σχετίζονται με την αποκοπή ευθυγράμμων τμημάτων σε παράθυρο αποκοπής σχήματος κυρτού πολυγώνου, χρησιμοποιήθηκε η εφαρμογή η οποία βρίσκεται στη διεύθυνση: https://matthes.sites.sch.gr/clipping. Η διαδικασία που ακολουθήθηκε για την αξιολόγησή τους είναι παρόμοια με εκείνη της αξιολόγησης των αλγορίθμων του προηγούμενου κεφαλαίου: Κατά την εκτέλεση κάθε αλγορίθμου αποκόπηκε ένα μεγάλο πλήθος αυθαιρέτων ευθυγράμμων τμημάτων στις δύο διαστάσεις τα οποία δημιουργήθηκαν σε μία περιοχή που προσδιοριζόταν από τα σημεία $(-1000, -1000)$ και $(1000, 1000)$. Η θύρα θέασης είχε διάσταση 640×480 εικονοστοιχεία, ενώ το κυρτό πολύγωνο (δηλ. το παράθυρο αποκοπής) βρισκόταν εντός αυτού. Το πλήθος των ακμών του παραθύρου αποκοπής καθοριζόταν κάθε φορά από τον χρήστη (βλ. Σχ. 3.18).



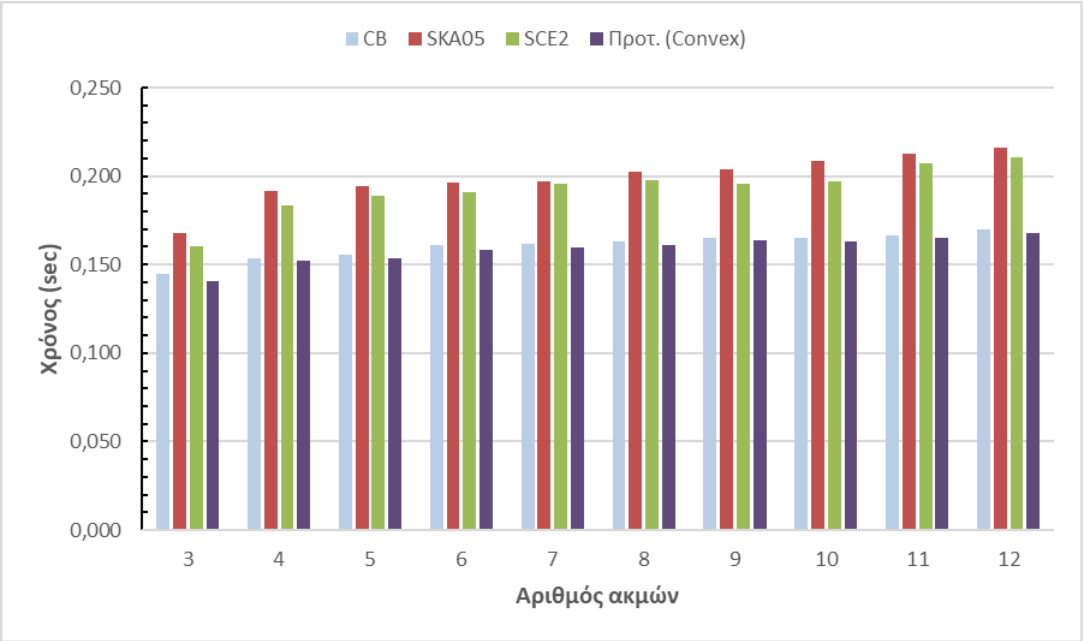
Σχήμα 3.18: Η περιοχή δημιουργίας αυθαίρετων τμημάτων και το παράθυρο αποκοπής σχήματος κυρτού πολυγώνου για την αξιολόγηση των αλγορίθμων

Κάθε αλγόριθμος εκτελέστηκε δέκα φορές και ο χρόνος που χρειάστηκε για να αποκόψει και για να σχεδιάσει τα ευθύγραμμα τμήματα σε κάθε εκτέλεση καταγράφηκε. Στο τέλος, ο μέσος χρόνος των δέκα εκτελέσεων υπολογίστηκε και αποτέλεσε το κριτήριο αξιολόγησης του αλγορίθμου ως προς την ταχύτητα. Όλες οι μετρήσεις έγιναν με την χρήση του ίδιου υπολογιστικού συστήματος του οποίου οι προδιαγραφές ήταν οι ακόλουθες: α) επεξεργαστής Intel Core i7-9750H @ 2.60 GHz CPU, β) μνήμη RAM 16 GB, γ) κάρτα γραφικών NVIDIA GeForce RTX 2070/8 GB GPU, δ) λειτουργικό σύστημα Windows 10 Pro 64 bit. Η ταχύτητα σύνδεσης στο διαδίκτυο δεν επηρέασε την ταχύτητα εκτέλεσης της εφαρμογής αφού η δεύτερη λειτουργεί σε τοπικό επίπεδο στον υπολογιστή μετά την έναρξή της.

Στον Πίνακα 3.1 φαίνονται οι χρόνοι για την αποκοπή 100.000 ευθυγράμμων τμημάτων, ενώ στο Σχ. 3.19 απεικονίζονται οι χρόνοι αυτοί με τη μορφή ραβδογράμματος.

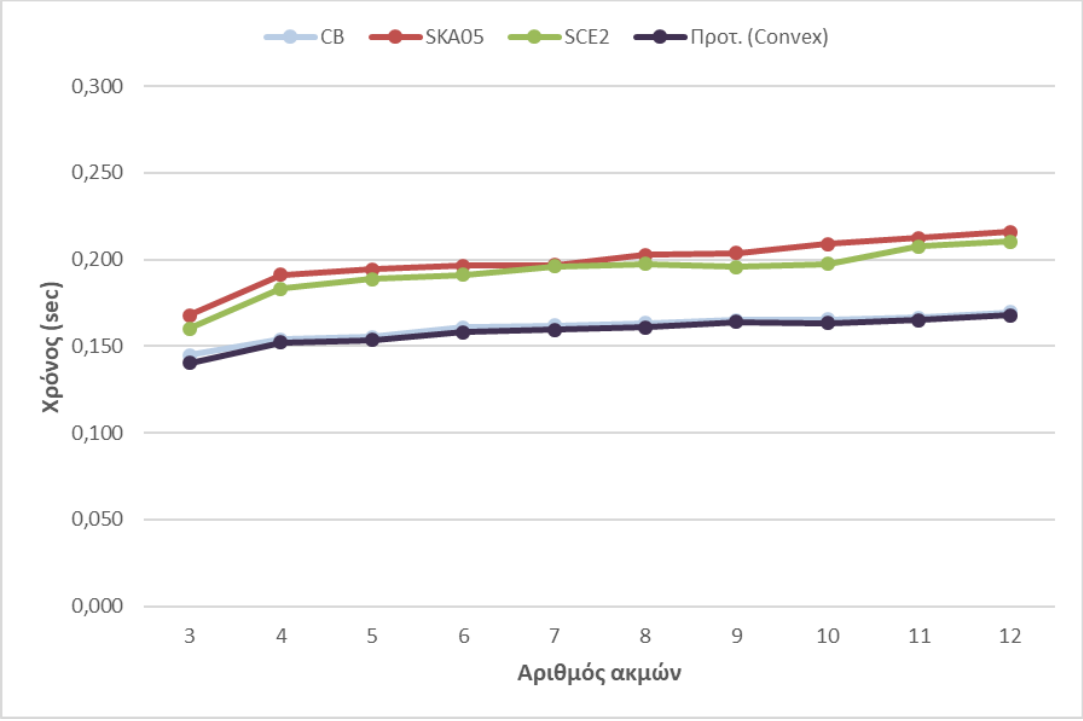
Αρ. ακμών	**Cyrus–Beck (sec)**	**Skala 2005 (sec)**	**S-Clip E2 (sec)**	**Προτ. (Convex) (sec)**
3	0.145	0.168	0.160	0.140
4	0.154	0.191	0.183	0.152
5	0.155	0.194	0.189	0.154
6	0.161	0.196	0.191	0.158
7	0.162	0.197	0.196	0.160
8	0.163	0.203	0.197	0.161
9	0.165	0.204	0.196	0.164
10	0.165	0.209	0.197	0.163
11	0.166	0.212	0.207	0.165
12	0.170	0.216	0.212	0.168

Πίνακας 3.1: Οι μέσοι χρόνοι εκτέλεσης των αλγορίθμων για την αποκοπή ευθυγράμμων τμημάτων σε κυρτά πολύγωνα με διαφορετικό αριθμό ακμών



Σχήμα 3.19: Ραβδογράμματα μέσων χρόνων κάθε αλγορίθμου ανά πλήθος ακμών

Ομοίως, η γραφική παράσταση των μέσων χρόνων κάθε αλγορίθμου φαίνεται στο Σχ. 3.20.



Σχήμα 3.20: Γραφική παράσταση μέσων χρόνων κάθε αλγορίθμου ανά πλήθος ακμών

3.6 Στατιστικός έλεγχος των αποτελεσμάτων

Για να διαπιστώσουμε αν τα αριθμητικά αποτελέσματα και τα συμπεράσματα της προηγούμενης παραγράφου είναι έγκυρα και στατιστικώς ορθά, χρησιμοποιήθηκε η δοκιμασία Wilcoxon μέσω της γλώσσας "R". Μετά την εισαγωγή των δεδομένων και συγκρίνοντας τον προτεινόμενο αλγόριθμο με τους υπόλοιπους, τα αποτελέσματα που ελήφθησαν είναι τα ακόλουθα:

12 κορυφές

Προτεινόμενος (Convex) - 12 κορυφές					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	64	0.306	-0.00294649	0.00602455	0.002007371
SKA05	100	0.0001796	0.04504122	0.05397925	0.04903547
SCE^2	99	0.0002409	0.008009259	0.015016706	0.01208666

Πίνακας 3.2: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon (12 κορυφές)

11 κορυφές

Προτεινόμενος (Convex) - 11 κορυφές					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	66	0.2361	-0.0009448591	0.0030428553	0.001018277
SKA05	100	0.0001727	0.04506609	0.04992173	0.04702674
SCE^2	100	0.0001766	0.03901580	0.04500733	0.04294539

Πίνακας 3.3: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon (11 κορυφές)

10 κορυφές

Προτεινόμενος (Convex) - 10 κορυφές					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	70	0.1375	-0.0009686124	0.0049401083	0.002017233
SKA05	100	0.0001688	0.04397609	0.04799281	0.04597437
SCE^2	110	0.000118	0.02996120	0.03792214	0.03291655

Πίνακας 3.4: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon (10 κορυφές)

9 κορυφές

Προτεινόμενος (Convex) - 9 κορυφές					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	66.5	0.2217	-0.0009609866	0.0030035422	0.001052661
SKA05	100	0.0001756	0.03607649	0.04303255	0.03997236
SCE^2	100	0.0001766	0.02795105	0.03696412	0.03304464

Πίνακας 3.5: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon (9 κορυφές)

8 κορυφές

Προτεινόμενος (Convex) - 8 κορυφές					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	70	0.1381	-0.0009821401	0.0059998287	0.00207872
SKA05	100	0.0001746	0.03895589	0.04404335	0.04194916
SCE^2	100	0.0001756	0.03293739	0.03999074	0.0369831

Πίνακας 3.6: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon (8 κορυφές)

7 κορυφές

Προτεινόμενος (Convex) - 7 κορυφές					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	75	0.06232	-4.229997e-05	4.999007e-03	0.002983573
SKA05	100	0.0001756	0.03699426	0.04296030	0.04003881
SCE^2	100	0.0001786	0.03102325	0.03896839	0.03598384

Πίνακας 3.7: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon (7 κορυφές)

6 κορυφές

Προτεινόμενος (Convex) - 6 κορυφές					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	70	0.1379	-0.001017164	0.005001824	0.001984609
SKA05	100	0.0001756	0.03497923	0.04095593	0.03798463
SCE^2	100	0.0001746	0.03001346	0.03502661	0.032959

Πίνακας 3.8: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon (6 κορυφές)

5 κορυφές

Προτεινόμενος (Convex) - 5 κορυφές					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	69	0.1586	-0.0009489694	0.0040272957	0.002
SKA05	100	0.0001776	0.03793242	0.04397598	0.04096813
SCE^2	100	0.0001776	0.03196378	0.03893769	0.03501081

Πίνακας 3.9: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon (5 κορυφές)

4 κορυφές

Προτεινόμενος (Convex) - 4 κορυφές					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	64.5	0.2846	-0.001051749	0.004990262	0.001049062
SKA05	100	0.0001786	0.03594742	0.04296326	0.03892615
SCE^2	100	0.0001796	0.02704258	0.03497424	0.03001586

Πίνακας 3.10: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon (4 κορυφές)

3 κορυφές

Προτεινόμενος (Convex) - 3 κορυφές					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	87	0.005487	0.001036648	0.007049168	0.004040042
SKA05	100	0.0001776	0.02398091	0.03095759	0.0269881
SCE^2	100	0.0001776	0.01601800	0.02293261	0.01903653

Πίνακας 3.11: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon (3 κορυφές)

Συνολικά

Προτεινόμενος (Convex) - Συνολικά					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CB	60.5	0.4485	-0.004978998	0.008980651	0.001971768
SKA05	99.5	0.000211	0.03196571	0.05103664	0.04161793
SCE^2	94.5	0.0008689	0.02292998	0.04198141	0.03203044

Πίνακας 3.12: Στατιστικός έλεγχος των συνολικών αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon

Εξετάζοντας την πρώτη περίπτωση του Πίνακα 3.2, δηλαδή την σύγκριση του προτεινόμενου αλγορίθμου με τον CB, και δεδομένου ότι p-value = 0.306, άρα μεγαλύτερη του 0.05, δεν απορρίπτεται σε επίπεδο σημαντικότητας 5% η μηδενική υπόθεση, δηλαδή συμπεραίνουμε ότι ο προτεινόμενος αλγόριθμος δεν έχει στατιστικά σημαντική διαφορά ως προς τον χρόνο υλοποίησης έναντι του CB.

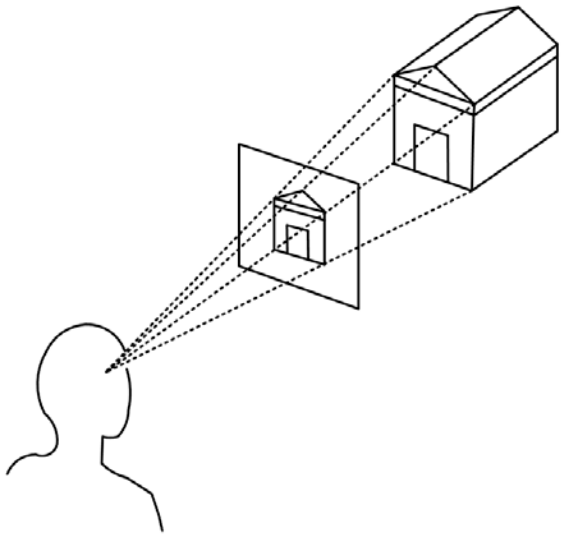
Το διάστημα εμπιστοσύνης 95% για τη διαφορά μεταξύ του χρόνου υλοποίησης του CB έναντι του προτεινόμενου αλγορίθμου δίνεται ως (-0.00294649, 0.00602455). Δεδομένου ότι η διαφορά δεν έχει σταθερό πρόσημο, συμπεραίνουμε ότι ο χρόνος του CB δεν είναι στατιστικώς σημαντικά μεγαλύτερος από τον χρόνο του προτεινόμενου αλγορίθμου. Βεβαίως, επειδή το μεγαλύτερο μέρος του διαστήματος βρίσκεται στον θετικό ημιάξονα, υπάρχουν ενδείξεις ότι ο χρόνος του προτεινόμενου τείνει να είναι μικρότερος (αλλά όχι στατιστικώς σημαντικά μικρότερος). Το συμπέρασμα αυτό ενισχύεται και από το γεγονός ότι η σημειακή εκτίμηση της διαφοράς μεταξύ του χρόνου του CB έναντι του χρόνου του προτεινόμενου είναι θετική και ίση με 0.002007371.

Για τις υπόλοιπες περιπτώσεις συμπεραίνουμε ότι ο προτεινόμενος αλγόριθμος έχει στατιστικά σημαντική διαφορά ως προς τον χρόνο υλοποίησης. Τα διαστήματα εμπιστοσύνης 95% έχουν σταθερά θετικό πρόσημο οπότε οδηγούμαστε στο συμπέρασμα ότι οι χρόνοι των υπολοίπων αλγορίθμων είναι στατιστικώς σημαντικά μεγαλύτεροι από τον χρόνο του προτεινόμενου.

Ανάλογα συμπεράσματα μπορεί κανείς να εξάγει, αν εξετάσει τους Πίνακες 3.3-3.12.

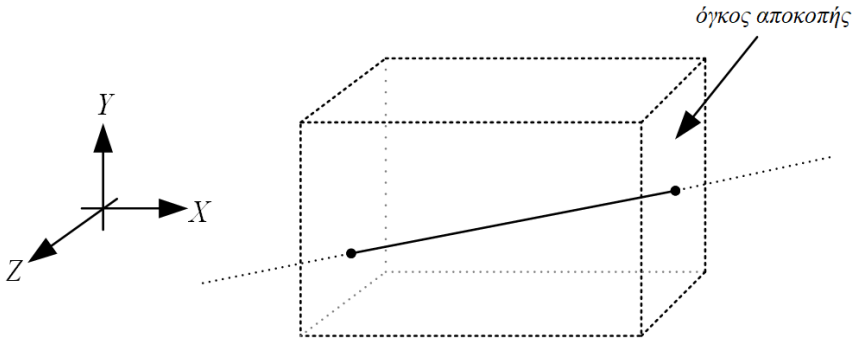
4. ΑΠΟΚΟΠΗ ΕΥΘΥΓΡΑΜΜΟΥ ΤΜΗΜΑΤΟΣ ΑΠΟ ΟΡΘΟΓΩΝΙΟ ΟΓΚΟ ΘΕΑΣΗΣ ΣΤΙΣ ΤΡΕΙΣ ΔΙΑΣΤΑΣΕΙΣ

Συγκρίνοντας τις δύο με τις τρεις διαστάσεις, η διαδικασία απεικόνισης των τριδιάστατων γραφικών είναι πολυπλοκότερη, αφενός λόγω της πρόσθετης διάστασης και αφετέρου επειδή οι συσκευές προβολής είναι, ως επί το πλείστον, διδιάστατες [Fol+90]. Αν υποθέσουμε ότι μία κάμερα εστιάζει σε έναν τριδιάστατο χώρο, τότε αυτός θα πρέπει να μετατραπεί και να απεικονιστεί σε μία περιοχή ορισμένη στις δύο διαστάσεις, όπως για παράδειγμα η οθόνη ενός υπολογιστή (βλ. Σχ. 4.1).



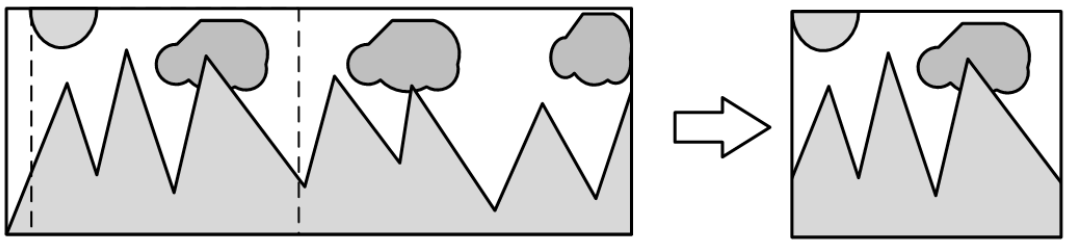
Σχήμα 4.1: Προβολή τριδιάστατου χώρου σε διδιάστατο επίπεδο

Στα διανυσματικά γραφικά, μία εικόνα ή ένα αντικείμενο αποτελείται κυρίως από σημεία και ευθύγραμμα τμήματα. Εφ' όσον πρέπει να αποκοπεί μεγάλο πλήθος σημείων ή ευθυγράμμων τμημάτων για την απόδοση μίας τυπικής σκηνής ή εικόνας, η αποτελεσματικότητα των αλγορίθμων αποκοπής είναι πολύ σημαντική. Σε πολλές περιπτώσεις η πλειονότητα των σημείων ή των ευθύγραμμων τμημάτων ευρίσκεται είτε εξ ολοκλήρου εντός είτε εξ ολοκλήρου εκτός του όγκου θέασης. Επομένως, είναι σημαντικό ένα σημείο ή ένα ευθύγραμμο τμήμα να μπορεί άμεσα να γίνει αποδεκτό ή όχι [Rog97] (βλ. Σχ. 4.2).



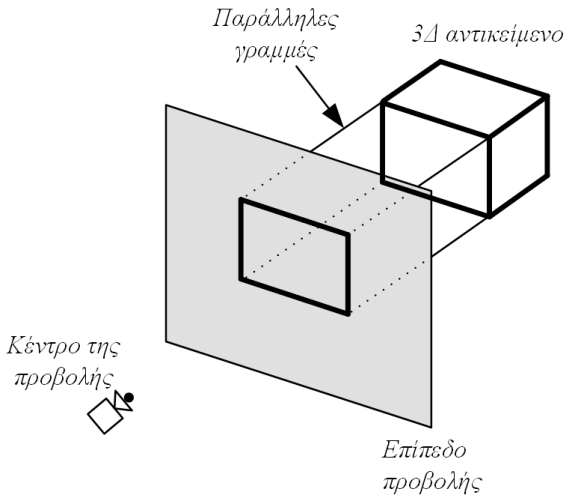
Σχήμα 4.2: Η αποτελεσματικότητα των αλγορίθμων αποκοπής ευθυγράμμων τμημάτων στις τρεις διαστάσεις είναι πολύ σημαντική

Στις δύο διαστάσεις, η προβολή μίας σκηνής σε μία οθόνη ή σε οποιαδήποτε άλλη συσκευή εξόδου είναι σχετικά απλή. Χρησιμοποιώντας ορθογώνια σύνορα, μία διαδικασία αποκοπής αποκόπτει τη διδιάστατη σκηνή και αντιστοιχίζει την πληροφορία που υπάρχει εντός των συνόρων αυτών στις συντεταγμένες της συσκευής (βλ. Σχ. 4.3).



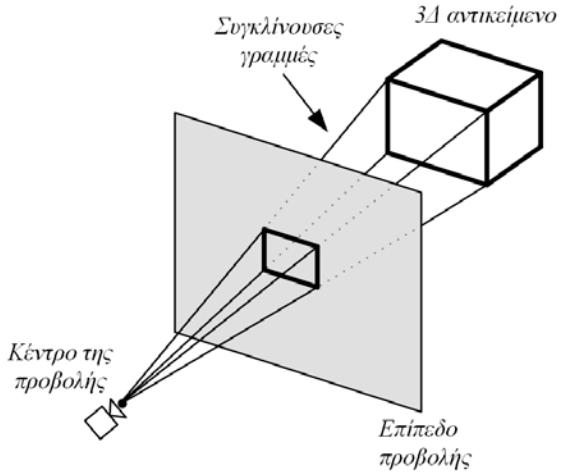
Σχήμα 4.3: Χρησιμοποιώντας ορθογώνια σύνορα, μία διαδικασία αποκοπής αποκόπτει τη διδιάστατη σκηνή και την αντιστοιχίζει στις συντεταγμένες της συσκευής

Οι λειτουργίες όμως της μετατροπής της τριδιάστατης προβολής σε διδιάσταστη είναι πιο περίπλοκες, διότι υπάρχουν επιπλέον διαδικασίες που δεν υφίστανται στην διδιάσταστη. Για να πραγματοποιηθεί η προβολή της τριδιάστατης σκηνής, θα πρέπει να δημιουργηθεί ένα επίπεδο προβολής όπου επάνω σε αυτό αποτυπώνεται η τριδιάστατη σκηνή αφού καθοριστεί ο τύπος και ο όγκος προβολής. Θα μπορούσε κανείς να παρομοιάσει το επίπεδο αυτό με το φιλμ μίας φωτογραφικής μηχανής ή μίας κινηματογραφικής κάμερας και οι κορυφές που συνθέτουν τα αντικείμενα της τριδιάστατης σκηνής να προβάλλονται στο επίπεδο αυτό. Αντίθετα με το φιλμ της κάμερας, υπάρχουν διάφορες μέθοδοι για την προβολή των κορυφών στο επίπεδο προβολής. Για παράδειγμα, μία τέτοια μέθοδος μπορεί να χρησιμοποιεί παράλληλες γραμμές οι οποίες ξεκινούν από τις κορυφές και καταλήγουν στο επίπεδο προβολής. Η τεχνική αυτή ονομάζεται παράλληλη προβολή και χρησιμοποιείται κυρίως στα μηχανολογικά ή αρχιτεκτονικά σχέδια τα οποία απαιτούν ακρίβεια κατά την προβολή των αντικειμένων (βλ. Σχ. 4.4).



Σχήμα 4.4: Στην παράλληλη προβολή γίνεται χρήση παράλληλων γραμμών οι οποίες ξεκινούν από τις κορυφές του αντικειμένου και καταλήγουν στο επίπεδο προβολής

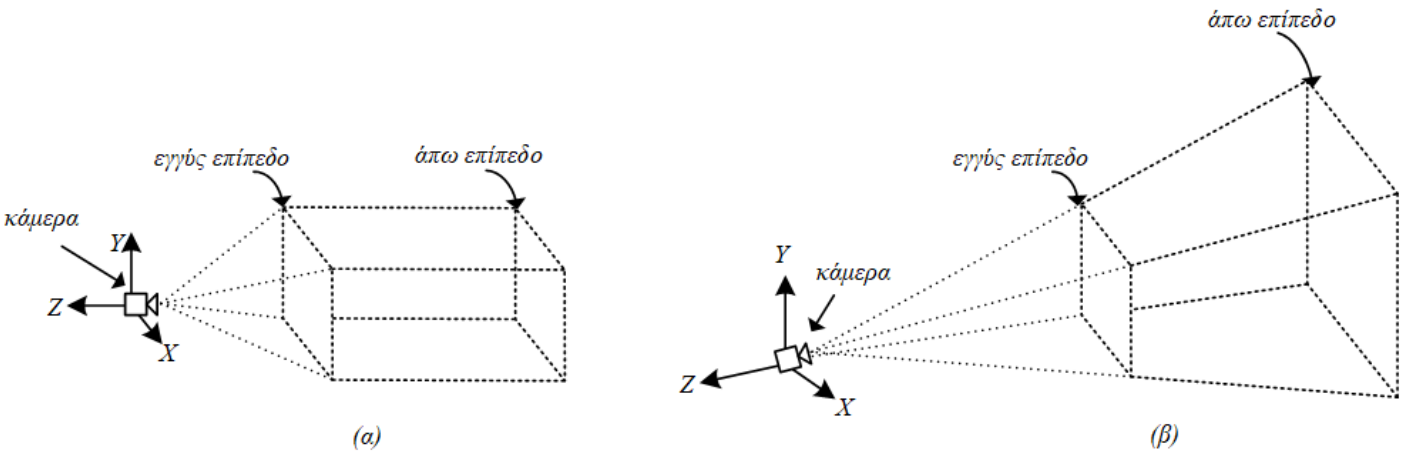
Εναλλακτικά, μία μέθοδος μπορεί να χρησιμοποιεί συγκλίνουσες γραμμές όπου η κάθε μία ξεκινά από την εκάστοτε κορυφή και κατευθύνεται προς το κέντρο της προβολής, δηλαδή την κάμερα (βλ. Σχ. 4.5). Η τεχνική αυτή ονομάζεται προοπτική προβολή. Το πλεονέκτημα της προοπτικής προβολής είναι ότι όσο πιο πολύ απομακρύνονται τα αντικείμενα από το κέντρο της προβολής τόσο πιο μικρά απεικονίζονται εξομοιώνοντας έτσι την λειτουργία του ανθρώπινου οφθαλμού.



Σχήμα 4.5: Στην προοπτική προβολή γίνεται χρήση συγκλινουσών γραμμών όπου κάθε μία ξεκινά από την εκάστοτε κορυφή και κατευθύνεται προς το κέντρο της προβολής

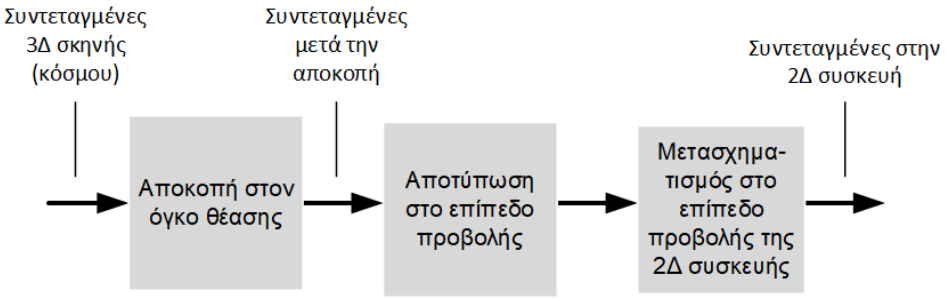
Με βάση τα παραπάνω, και ανάλογα με τον τύπο της προβολής που χρησιμοποιείται, ο όγκος θέασης έχει συνήθως τη μορφή ενός ορθογωνίου παραλληλεπίπεδου ή μίας κόλουρης πυραμίδας (βλ. Σχ. 4.6). Εντός του

όγκου θέασης πραγματοποιείται αποκοπή γι' αυτό, εναλλακτικά, ονομάζεται και όγκος αποκοπής.



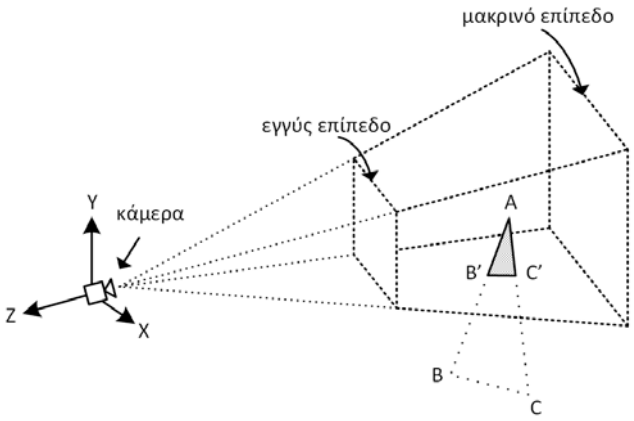
Σχήμα 4.6: (α) Ορθογώνιος παραλληλεπίπεδος όγκος θέασης και (β) Όγκος θέασης σε σχήμα κόλουρης πυραμίδας

Το Σχ. 4.7 παρουσιάζει τα στάδια της διαδικασίας προβολής μίας τριδιάστατης σκηνής σε μία διδιάστατη συσκευή. Ο όγκος θέασης χρησιμοποιείται για την απόρριψη των περιττών και εκτός συνόρων στοιχείων της τριδιάστατης σκηνής ενώ, στη συνέχεια, τα υπόλοιπα στοιχεία προβάλλονται στο επίπεδο προβολής. Κατόπιν, το επίπεδο προβολής μετατρέπεται σε διδιάστατες συντεταγμένες συσκευής.



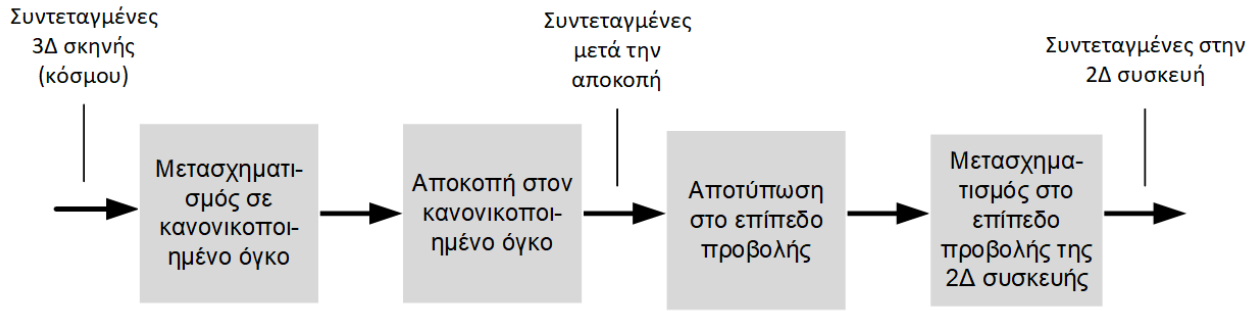
Σχήμα 4.7: Τα στάδια της διαδικασίας προβολής μίας τριδιάστατης σκηνής σε μία διδιάστατη συσκευή

Δυστυχώς, εάν η προβολή είναι προοπτική και ο όγκος θέασης είναι κόλουρη πυραμίδα, μπορεί να είναι υπολογιστικά δαπανηρό να ελεγχθεί εάν υπάρχουν αντικείμενα εντός των συνόρων και να πραγματοποιηθεί αποκοπή (βλ. Σχ. 4.8).

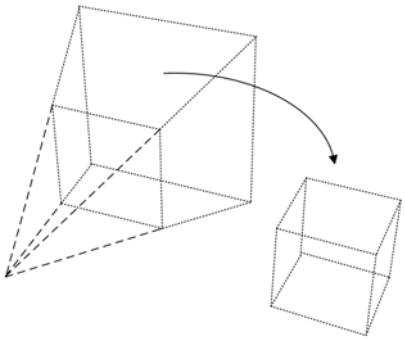


Σχήμα 4.8: Εάν ο όγκος θέασης είναι κόλουρη πυραμίδα, τότε η αποκοπή είναι υπολογιστικά δαπανηρή

Προκειμένου να απλοποιηθεί και να επιταχυνθεί η διαδικασία της αποκοπής, προστίθεται ένα επιπλέον στάδιο στο παραπάνω μοντέλο (βλ. Σχ. 4.9). Πριν από τα στάδια αποκοπής και προβολής, η κόλουρη πυραμίδα μετατρέπεται σε κανονικοποιημένο όγκο προβολής , δηλαδή έναν κύβο που ορίζεται από τα σημεία $(-1,-1,-1)$ και $(1,1,1)$, και στη συνέχεια εφαρμόζονται οι υπολογισμοί αποκοπής (βλ. Σχ. 4.10).



Σχήμα 4.9: Η μετατροπή της κόλουρης πυραμίδας σε κανονικοποιημένο όγκο θέασης προστίθεται στη διαδικασία προβολής της τριδιάστατης σκηνής

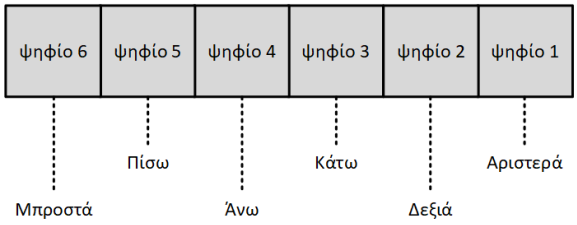


Σχήμα 4.10: Η αποκοπή στον κανονικοποιημένο όγκο θέασης είναι ταχύτερη σε σχέση με την κόλουρη πυραμίδα

Στο προηγούμενο κεφάλαιο μελετήσαμε πολλούς αλγόριθμους αποκοπής ευθυγράμμων τμημάτων σε δύο διαστάσεις αλλά μόνο ορισμένοι από αυτούς μπορούν να εφαρμόσουν αποκοπή σε έναν τριδιάστατο χώρο. Αυτοί είναι οι Cohen–Sutherland, Liang–Barsky και Cyrus–Beck. Φυσικά, με την πάροδο των χρόνων, έχουν δημιουργηθεί και άλλοι σχετικοί με την τριδιάστατη αποκοπή, όπως οι [KWC12] και [Kol94].

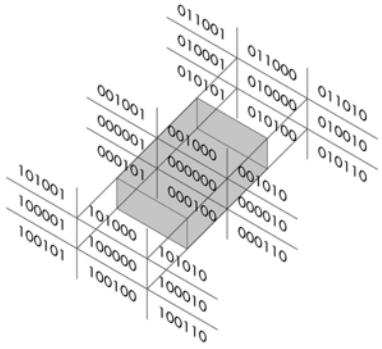
4.1 Αλγόριθμος Cohen-Sutherland 3D

Ο Cohen–Sutherland θεωρείται ένας κλασικός αλγόριθμος αποκοπής ευθυγράμμων τμημάτων στις δύο διαστάσεις ενώ η τριδιάστατη εκδοχή του (CS 3D) προκύπτει κατόπιν μικρών τροποποιήσεων. Για έναν ορθογώνιο όγκο θέασης, ο χώρος διαιρείται εκ νέου σε περιοχές αλλά, αυτή τη φορά, ο κωδικός περιοχής περιλαμβάνει έξι ψηφία, αντί τεσσάρων, και χρησιμοποιείται για την ταξινόμηση των σημείων που προσδιορίζουν το τριδιάστατο προς αποκοπή ευθύγραμμο τμήμα. Αυτά τα έξι ψηφία απεικονίζονται στο Σχ. 4.11.



Σχήμα 4.11: Ο κωδικός περιοχής του αλγορίθμου Cohen–Sutherland 3D περιλαμβάνει έξι ψηφία

Τα δύο επιπρόσθετα ψηφία (5 και 6) έχουν να κάνουν με τον χώρο που βρίσκεται όπισθεν και έμπροσθεν, αντίστοιχα, του όγκου θέασης. Όλοι οι πιθανοί συνδυασμοί των ψηφίων και οι αντίστοιχοι κωδικοί περιοχής φαίνονται στο Σχ. 4.12.



Σχήμα 4.12: Οι κωδικοί περιοχής στην τριδιάστατη έκδοση του αλγορίθμου Cohen–Sutherland

Ο έλεγχος των σημείων του ευθυγράμμου τμήματος είναι σχεδόν ίδιος με εκείνον της διδιάστατης έκδοσης.

Τα ψηφία του κωδικού περιοχής λαμβάνουν την τιμή “1” (Αληθής) ή “0” (Ψευδής) ανάλογα με το αν ισχύουν οι παρακάτω συνθήκες:

Ψηφίο 1: Το σημείο βρίσκεται έμπροσθεν του όγκου θέασης

Ψηφίο 2: Το σημείο βρίσκεται όπισθεν του όγκου θέασης

Ψηφίο 3: Το σημείο βρίσκεται άνωθεν του όγκου θέασης

Ψηφίο 4: Το σημείο βρίσκεται κάτωθεν του όγκου θέασης

Ψηφίο 5: Το σημείο βρίσκεται δεξιά από τον όγκο θέασης

Ψηφίο 6: Το σημείο βρίσκεται αριστερά από τον όγκο θέασης

Εφ’ όσον πραγματοποιηθεί η κατηγοριοποίηση των σημείων μετά την απόδοση των κωδικών περιοχής σε αυτά, το ευθύγραμμο τμήμα είναι:

Εξ ολοκλήρου ορατό:	Αν και οι δύο κωδικοί περιοχής είναι 000000
Εξ ολοκλήρου αόρατο:	Αν το αποτέλεσμα της λογικής σύζευξης δεν είναι 000000
Υποψήφιο για αποκοπή:	Αν το αποτέλεσμα της λογικής σύζευξης είναι 000000

Αυτό σημαίνει ότι όλα τα ευθύγραμμα τμήματα των οποίων τα σημεία έχουν κωδικό περιοχής 000000 είναι αποδεκτά. Όλα τα ευθύγραμμα τμήματα των οποίων οι κωδικοί των σημείων τους μοιράζονται ένα κοινό ψηφίο σε οποιαδήποτε θέση, απορρίπτονται αυτόματα. Το αποκομμένο ευθύγραμμο τμήμα προκύπτει μέσα από την εξίσωση της ευθείας σε τριδιάστατο χώρο χρησιμοποιώντας είτε ομογενείς συντεταγμένες είτε τις συντεταγμένες των συνόρων του όγκου θέασης, όποτε αυτό είναι απαραίτητο.

Τα βήματα του αλγορίθμου συνοπτικά είναι:

1. Για κάθε τελικό σημείο του ευθυγράμμου τμήματος υπολογίζεται ο κωδικός περιοχής.
2. Εφαρμόζεται λογική διάζευξη μεταξύ των δύο κωδικών περιοχής. Αν το αποτέλεσμα είναι 000000, τότε το ευθύγραμμο τμήμα είναι εξ ολοκλήρου εντός του όγκου θέασης οπότε σχεδιάζεται ως έχει και ο αλγόριθμος τερματίζει διαφορετικά συνεχίζει στο επόμενο βήμα.
3. Εφαρμόζεται λογική σύζευξη μεταξύ των δύο κωδικών περιοχής. Αν το αποτέλεσμα δεν είναι 000000, τότε το ευθύγραμμο τμήμα είναι εξ ολοκλήρου εκτός του όγκου θέασης οπότε απορρίπτεται και ο αλγόριθμος τερματίζει διαφορετικά συνεχίζει στο επόμενο βήμα.
4. Οι κωδικοί περιοχής χρησιμοποιούνται για να υπολογιστούν τα σημεία τομής του ευθυγράμμου τμήματος με τον όγκο θέασης.

Ο Κώδικας A.17 του Παραρτήματος A υλοποιεί τον αλγόριθμο Cohen-Sutherland 3D.

4.2 Αλγόριθμος Cyrus-Beck 3D

Ο Cyrus–Beck, ο δεύτερος παραδοσιακός αλγόριθμος αποκοπής ευθυγράμμων τμημάτων που μελετήθηκε σε προηγούμενο κεφάλαιο, μπορεί σχετικά εύκολα να τροποποιηθεί για να πραγματοποιήσει αποκοπή και στον τριδιάστατο χώρο. Σύμφωνα με τον αλγόριθμο, ένα ευθύγραμμο τμήμα μπορεί να τέμνει ένα κλειστό και οριοθετημένο κυρτό σχήμα το πολύ σε δύο σημεία [CB78]. Κάθε πλευρά του όγκου θέασης αποτελεί ένα κυρτό πολύγωνο σε τριδιάστατο χώρο και χρησιμοποιείται ως επίπεδο αποκοπής. Αυτό σημαίνει ότι ο αλγόριθμος εξετάζει και τα έξι επίπεδα του όγκου θέασης και πραγματοποιεί αποκοπή όπου κρίνεται απαραίτητο. Για κάθε ένα επίπεδο εφαρμόζονται τα ακόλουθα:

1. Υπολογίζεται το εισερχόμενο φυσιολογικό διάνυσμα του επιπέδου (κάθετο προς το επίπεδο) χρησιμοποιώντας τρεις από τις τέσσερις γνωστές κορυφές που σχηματίζουν το επίπεδο.
2. Υπολογίζεται το διάνυσμα το οποίο περιγράφεται από τα τελικά σημεία του προς αποκοπή ευθυγράμμου τμήματος (διάνυσμα ευθυγράμμου τμήματος).
3. Υπολογίζεται το εσωτερικό γινόμενο μεταξύ του φυσιολογικού διανύσματος και του διανύσματος που προκύπτει από την διαφορά μίας κορυφής του επιπέδου και ενός επιλεγμένου τελικού σημείου του προς αποκοπή ευθυγράμμου τμήματος (Εσωτερικό γινόμενο 1).
4. Υπολογίζεται το εσωτερικό γινόμενο μεταξύ του διανύσματος ευθυγράμμου τμήματος και του φυσιολογικού διανύσματος (Εσωτερικό γινόμενο 2).
5. Το Εσωτερικό γινόμενο 1 διαιρείται με το Εσωτερικό γινόμενο 2 και πολλαπλασιάζεται με -1. Με τον τρόπο αυτό λαμβάνονται οι τιμές του t.
6. Οι τιμές του t ταξινομούνται σε Δυνητικές Εισχωρήσεις και Δυνητικές Αποχωρήσεις παρατηρώντας τους παρονομαστές τους (Εσωτερικό γινόμενο 2).
7. Από κάθε ομάδα Δυνητικών Εισχωρήσεων και Αποχωρήσεων επιλέγεται μία τιμή του t η οποία χρησιμοποιείται στην παραμετρική μορφή της εξίσωσης της ευθείας για τον υπολογισμό των συντεταγμένων.
8. Εάν η τιμή t μίας Δυνητικής Εισχώρησης είναι μεγαλύτερη από την τιμή t μίας Δυνητικής Αποχώρησης, τότε το προς αποκοπή ευθύγραμμο τμήμα απορρίπτεται.

Ο Κώδικας A.18 του Παραρτήματος A υλοποιεί τον αλγόριθμο Cyrus-Beck 3D.

4.3 Αλγόριθμος Liang-Barsky 3D

Ο αλγόριθμος των Liang–Barsky επεκτείνεται πολύ εύκολα στις τρεις διαστάσεις (LB 3D) και θεωρείται αρκετά πιο αποτελεσματικός από τον Cohen–Sutherland 3D. Η τριδιάστατη έκδοσή του χρησιμοποιεί τις παρακάτω έννοιες:

1. Την παραμετρική εξίσωση της ευθείας.
2. Τις ανισότητες που περιγράφουν τα σύνορα του όγκου θέασης για τον προσδιορισμό των τομών μεταξύ αυτών και του προς αποκοπή ευθυγράμμου τμήματος.

Οι παραμετρικές εξισώσεις που περιγράφουν μία γραμμή στις τρεις διαστάσεις είναι:

$$\begin{aligned} x &= x_0 + t \cdot (x_1 - x_0) \\ y &= y_0 + t \cdot (y_1 - y_0) \\ z &= z_0 + t \cdot (z_1 - z_0) \quad , \quad 0 \leq t \leq 1 \end{aligned}$$

Αν θεωρήσουμε ένα τριδιάστατο σημείο με συντεταγμένες (x, y, z) και έναν όγκο θέασης ο οποίος ορίζεται από τα σημεία $(x_{\text{min}}, y_{\text{min}}, z_{\text{min}})$ και $(x_{\text{max}}, y_{\text{max}}, z_{\text{max}})$ τότε οι παραπάνω εξισώσεις μπορούν να μετατραπούν στις ανισώσεις:

$$x_{\text{min}} \leq x_0 + (x_1 - x_0) \cdot t \leq x_{\text{max}}$$

$$y_{\text{min}} \leq y = y_0 + (y_1 - y_0) \cdot t \leq y_{\text{max}}$$

$$z_{\text{min}} \leq z = z_0 + (z_1 - z_0) \cdot t \leq z_{\text{max}}$$

και οι ανισώσεις μπορούν να γραφούν διαφορετικά ως:

$$p_k \cdot \ t \leq q_k,$$

όπου οι τιμές $k = 1, 2, 3, 4, 5, 6$ αντιστοιχούν στο αριστερό (x_{min}), δεξί (x_{max}), κάτω (y_{min}), άνω (y_{max}), μακρινό

(z_{min}) και κοντινό (z_{max}) σύνορο. Τα p και q προσδιορίζονται ως ακολούθως:

$$
\begin{aligned}
p_1 &= -(x_1 - x_0) & q_1 &= (x_0 - x_{\text{min}}) & &\text{(αριστερό σύνορο)}\\
p_2 &= (x_1 - x_0) & q_2 &= (x_{\text{max}} - x_0) & &\text{(δεξί σύνορο)}\\
p_3 &= -(y_1 - y_0) & q_3 &= (y_0 - y_{\text{min}}) & &\text{(κάτω σύνορο)}\\
p_4 &= (y_1 - y_0) & q_4 &= (y_{\text{max}} - y_0) & &\text{(άνω σύνορο)}\\
p_5 &= -(z_1 - z_0) & q_5 &= (z_0 - z_{\text{min}}) & &\text{(μακρινό σύνορο)}\\
p_6 &= (z_1 - z_0) & q_6 &= (z_{\text{max}} - z_0) & &\text{(εγγύς σύνορο)}
\end{aligned}
$$

Από τα παραπάνω μπορεί κανείς να συμπεράνει ότι:

- Όταν το ευθύγραμμο τμήμα είναι παράλληλο με τον όγκο θέασης, η τιμή του p είναι μηδέν.
- Όταν $p_k < 0$, όσο το t αυξάνει το ευθύγραμμο τμήμα εισέρχεται στον όγκο θέασης.
- Όταν $p_k > 0$, το ευθύγραμμο τμήμα εξέρχεται του όγκου θέασης.
- Όταν $p_k = 0$ και $q_k < 0$, το ευθύγραμμο τμήμα είναι εκτός του όγκου θέασης.
- Όταν $p_k = 0$ και $q_k > 0$, το ευθύγραμμο τμήμα είναι εντός του όγκου θέασης.

Οι παράμετροι t_1 και t_2 μπορούν να υπολογιστούν προκειμένου να προσδιοριστεί το ευθύγραμμο τμήμα που βρίσκεται εντός του όγκου θέασης. Πιο συγκεκριμένα:

- Όταν $p_k < 0$, χρησιμοποιείται η τιμή $\max(0, q_k/p_k)$.
- Όταν $p_k > 0$, χρησιμοποιείται η τιμή $\min(1, q_k/p_k)$.

Εάν $t_1 > t_2$, το ευθύγραμμο τμήμα είναι εξ ολοκλήρου εκτός του όγκου θέασης και απορρίπτεται. Σε διαφορετική περίπτωση τα σημεία του αποκοπτόμενου τμήματος υπολογίζονται από τις δύο παραμέτρους t.

Ο Κώδικας A.19 του Παραρτήματος A παρουσιάζει την υλοποίηση του αλγορίθμου Liang-Barsky 3D.

4.4 Αλγόριθμος Kolingerová 3D

Το 1994, η Ivana Kolingerová παρουσίασε μία συγκριτική μελέτη για τριδιάστατους αλγόριθμους αποκοπής ευθυγράμμων τμημάτων. Βασιζόμενη στον αλγόριθμο Cyrus-Beck, πρότεινε βελτιώσεις για την αύξηση της ταχύτητάς του είτε μέσω του πρόωρου τερματισμού των υπολογισμών όταν βρεθούν τα δύο σημεία τομής του προς

αποκοπή ευθυγράμμου τμήματος έναντι του όγκου θέασης είτε μέσω αποδοτικότερων υπολογισμών των σημείων τομής. Στο άρθρο της [Kol94] χρησιμοποιούνται οι ακόλουθοι συμβολισμοί:

Η ευθεία L: $x = X_A + s^T t$, όπου x_A είναι ένα σημείο της L, και s είναι ένα διάνυσμα κατεύθυνσης

Το κυρτό πολύεδρο P: ακμές $\{x_i, i\colon 1, 2, \dots, M\}$ όψεις $\{f_j, j\colon 1, 2, \dots, N\}$

Ακολουθούν μερικές από τις προτάσεις της για βελτίωση:

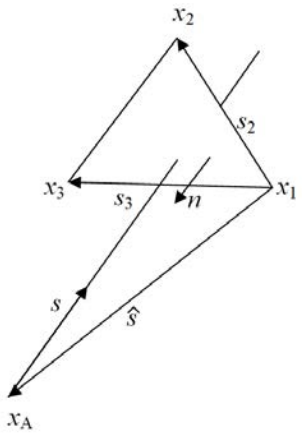
1. **Υπολογισμός μέσω παραμετρικών εξισώσεων:** Αυτή η πρόταση είναι ίσως η περισσότερο χρησιμοποιούμενη. Στην αρχή υπολογίζεται το σημείο τομής του ευθυγράμμου τμήματος με το επίπεδο στο οποίο βρίσκεται η εξεταζόμενη όψη του πολυέδρου και στη συνέχεια ελέγχεται για να διαπιστωθεί εάν αυτό βρίσκεται μέσα στα όρια της όψης αυτής. Για παράδειγμα, η τομή ενός ευθυγράμμου τμήματος και ενός τριγώνου υπολογίζεται από τις ακόλουθες παραμετρικές εξισώσεις:

$$x(p, q) = x_1 + s_2 p + s_3 q, \quad 0 \le p, q \le 1,$$
$$p + q \le 1 \quad \text{(ένα τρίγωνο)}$$
$$x(t) = x_A + st \quad -\infty \le t \le +\infty$$

Αν για το σημείο τομής x_k το $p \in [0, 1]$ και το $q \in [0, 1-p]$, τότε το x_k βρίσκεται εντός του τριγώνου. Ο αλγόριθμος απεικονίζεται πλησίον του Σχήματος 4.13.

$$s_2 = x_2 - x_1$$
$$s_3 = x_3 - x_1$$
$$\hat{s} = x_A - x_1$$
$$n = s_2 \times s_3$$
$$\hat{n} = s \times \hat{s}$$

$$p = -{s_3}^T \hat{n} / s^T n$$
$$q = -{s_2}^T \hat{n} / s^T n$$
$$t = -\hat{s}^T n / s^T n$$



Σχήμα 4.13: Υπολογισμοί μέσω παραμετρικών εξισώσεων της Kolingerová

Αυτή η μέθοδος χρειάζεται περισσότερους υπολογισμούς από τον αλγόριθμο Cyrus-Beck αλλά τα πλεονε-

κτήματά της είναι:

α) Όταν βρεθούν δύο τομές, ο αλγόριθμος τερματίζει.

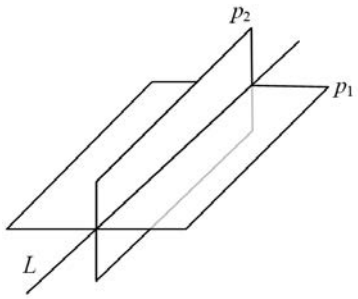
β) Η μέθοδος μπορεί να χρησιμοποιηθεί και σε μη κυρτά πολύεδρα.

2. **Δοκιμασίες με επίπεδα:** Επειδή ο αλγόριθμος της προηγούμενης περίπτωσης είναι χρονοβόρος, προτείνεται η βελτίωσή του μέσω δοκιμασιών με επίπεδα. Η δεδομένη ευθεία L μπορεί να περιγραφεί ως η τομή δύο ορθογώνιων επιπέδων p_1 και p_2 τα οποία είναι συγγραμμικά με έναν εκ των αξόνων συντεταγμένων (βλ. Σχ. 4.14). Εάν κάποιο από αυτά τα επίπεδα δεν τέμνει καμία όψη του πολύεδρου, τότε η γραμμή δεν μπορεί να τέμνει ούτε το πολύεδρο. Εάν και τα δύο επίπεδα τέμνουν μία όψη, τότε πρέπει να χρησιμοποιηθεί η δοκιμή για το αν ένα σημείο είναι εντός ενός πολυγώνου της προηγούμενης περίπτωσης.
   Η τομή μεταξύ του επιπέδου και της όψης μπορεί να ελεγχθεί με έναν απλό τρόπο: ένα τρίγωνο τέμνεται από ένα επίπεδο $p_i, i = 1, 2$ εάν και μόνο εάν υπάρχουν δύο κορυφές του τριγώνου x_l και x_2 τέτοιες ώστε:

$$\text{sign}(F_i(x_1)) \neq \text{sign}(F_i(x_2))$$

   όπου $F_i(x) = a_i x + b_i y + c_i z + d_i$ είναι η εξίσωση του επιπέδου $p_i, i = 1, 2$. Αν τα p_i ληφθούν παράλληλα με έναν άξονα συντεταγμένων, τότε θα ισχύει:

$$F_1(x) = a_1 x + c_1 z + d_1$$
$$F_2(x) = b_2 y + c_2 z + d_2$$



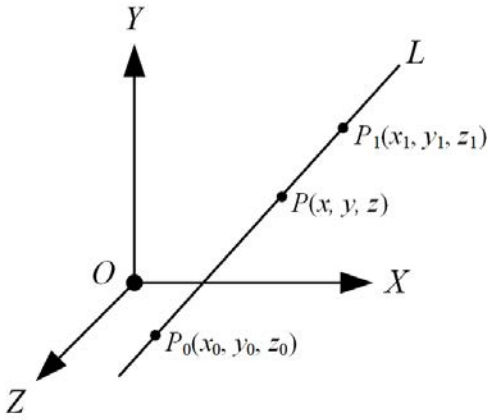
Σχήμα 4.14: Η γραμμή L περιγράφεται από την τομή των επιπέδων p_1 και p_2

Ο Κώδικας A.20 του Παραρτήματος A υλοποιεί την 1η πρόταση της Kolingerová.

4.5 Προτεινόμενος αλγόριθμος

Ο προτεινόμενος αλγόριθμος που χρησιμοποιήθηκε για την αποκοπή ευθυγράμμων τμημάτων στις δύο διαστάσεις μπορεί εύκολα να τροποποιηθεί και να χρησιμοποιηθεί στις τρεις διαστάσεις [MD23].

Ας θεωρήσουμε ένα ευθύγραμμο τμήμα στις τρεις διαστάσεις το οποίο ορίζεται από τα σημεία $P_0(x_0, y_0, z_0)$ και $P_1(x_1, y_1, z_1)$ (βλ. Σχ. 4.15).



Σχήμα 4.15: Ευθύγραμμο τμήμα στις τρεις διαστάσεις που ορίζεται από τα σημεία P_0 και P_1

Έστω, επίσης, ένα αυθαίρετο σημείο $P(x, y, z)$ το οποίο βρίσκεται επί της ευθείας L στην οποία ανήκει το ευθύγραμμο τμήμα. Η παραμετρική εξίσωση της ευθείας είναι:

$$x = x_0 + (x_1 - x_0) \cdot t$$
$$y = y_0 + (y_1 - y_0) \cdot t$$
$$z = z_0 + (z_1 - z_0) \cdot t$$

Εφ'όσον η παράμετρος t είναι κοινή και για τις τρεις παραπάνω εξισώσεις, αυτές μπορούν να γραφούν και ως εξής:

$$\frac{x - x_0}{x_1 - x_0} = \frac{y - y_0}{y_1 - y_0} = \frac{z - z_0}{z_1 - z_0}$$

Θέτοντας

$$dx = x_1 - x_0$$

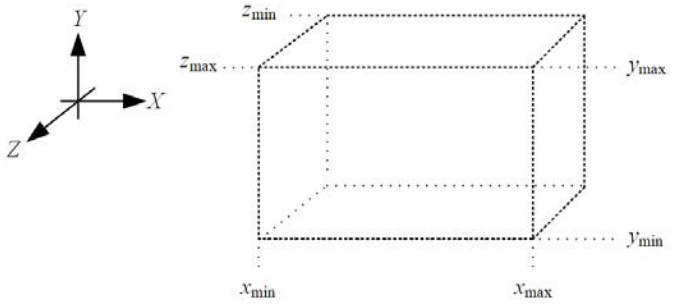
$$dy = y_1 - y_0$$

$$dz = z_1 - z_0$$

η εξίσωση παίρνει τη μορφή:

$$\frac{x - x_0}{dx} = \frac{y - y_0}{dy} = \frac{z - z_0}{dz} \tag{4.1}$$

Τώρα θέλουμε να χρησιμοποιήσουμε τον όγκο θέασης του Σχ. 4.16 για να αποκόψουμε το υπό εξέταση ευθύγραμμο τμήμα. Οι διαστάσεις του όγκου κυμαίνονται από x_{min} έως x_{max} για τον άξονα X (πλάτος), από y_{min} έως y_{max} για τον άξονα Y (ύψος) και από z_{min} έως z_{max} για τον άξονα Z (βάθος).



Σχήμα 4.16: Ο όγκος θέασης για την αποκοπή του υπό εξέταση ευθύγραμμου τμήματος

Όπως και στην περίπτωση της διδιάστατης έκδοσης, ο αλγόριθμος ελέγχει πρώτα εάν και τα δύο σημεία του ευθυγράμμου τμήματος είναι εκτός του όγκου θέασης και ταυτόχρονα στην ίδια περιοχή (αριστερά, δεξιά, κάτωθεν, άνωθεν, έμπροσθεν, όπισθεν). Αν ένα από τα ακόλουθα είναι Αληθές, τότε το ευθύγραμμο τμήμα απορρίπτεται και ο αλγόριθμος σταματά (βλ. Σχ. 4.17):

$x_0 < x_{\min}$ ΚΑΙ $x_1 < x_{\min}$ (το ευθύγραμμο τμήμα βρίσκεται αριστερά του όγκου θέασης)

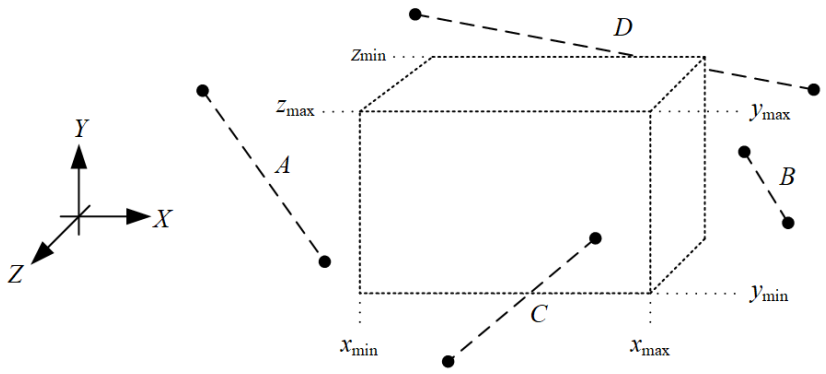
$x_0 > x_{\max}$ ΚΑΙ $x_1 > x_{\max}$ (το ευθύγραμμο τμήμα βρίσκεται δεξιά του όγκου θέασης)

$y_0 < y_{\min}$ ΚΑΙ $y_1 < y_{\min}$ (το ευθύγραμμο τμήμα βρίσκεται κάτωθεν του όγκου θέασης)

$y_0 > y_{\max}$ ΚΑΙ $y_1 > y_{\max}$ (το ευθύγραμμο τμήμα βρίσκεται άνωθεν του όγκου θέασης)

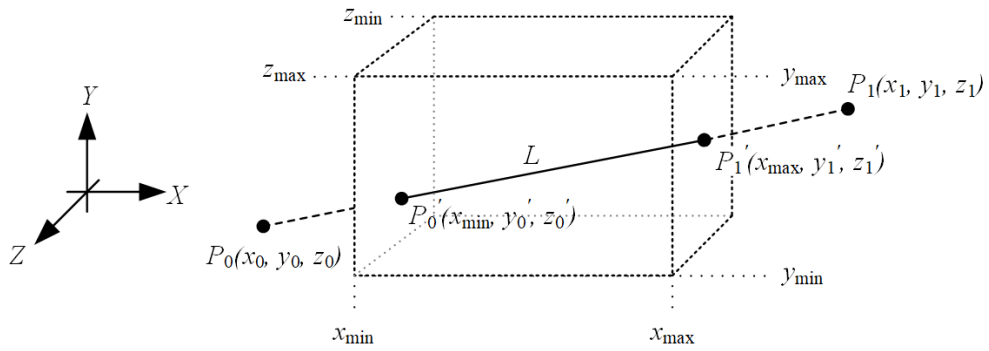
$z_0 < z_{\min}$ ΚΑΙ $z_1 < z_{\min}$ (το ευθύγραμμο τμήμα βρίσκεται όπισθεν του όγκου θέασης)

$z_0 > z_{\max}$ ΚΑΙ $z_1 > z_{\max}$ (το ευθύγραμμο τμήμα βρίσκεται έμπροσθεν του όγκου θέασης)



Σχήμα 4.17: Απορριπτέα ευθύγραμμα τμήματα: το A βρίσκεται αριστερά του όγκου θέασης, το B δεξιά, το C έμπροσθεν και το D όπισθεν

Στο δεύτερο βήμα, ο αλγόριθμος συγκρίνει τις συντεταγμένες των δύο σημείων του ευθυγράμμου τμήματος με τα σύνορα του όγκου θέασης. Πιο συγκεκριμένα, συγκρίνει κάθε μία από τις συντεταγμένες x_0 και x_1 με τα σύνορα $x_{\min}$ και $x_{\max}$, κάθε μία από τις συντεταγμένες y_0 και y_1 με τα σύνορα $y_{\min}$ και $y_{\max}$ και, τέλος, κάθε μία από τις συντεταγμένες z_0 και z_1 με τα σύνορα $z_{\min}$ και $z_{\max}$. Αν κάποια από τις συγκρινόμενες συντεταγμένες είναι εκτός των συνόρων του όγκου θέασης, τότε οι συντεταγμένες των συνόρων αντικαθιστούν τις συντεταγμένες αυτές (βλ. Σχ. 4.18).



Σχήμα 4.18: Χρησιμοποιώντας τις συντεταγμένες των συνόρων του όγκου θέασης στα σημεία του ευθυγράμμου τμήματος

Για κάθε σημείο του ευθυγράμμου τμήματος και με βάση την εξίσωση 4.1, οι συγκρίσεις και οι νέες συντεταγμένες που προκύπτουν είναι:

- Αν $x_i < x_{\text{min}}$, τότε

$$\frac{x_{\text{min}} - x_0}{dx} = \frac{y - y_0}{dy} \Rightarrow y = \frac{dy}{dx} \cdot (x_{\text{min}} - x_0) + y_0$$
$$\frac{x_{\text{min}} - x_0}{dx} = \frac{z - z_0}{dz} \Rightarrow z = \frac{dz}{dx} \cdot (x_{\text{min}} - x_0) + z_0$$
$$x = x_{\text{min}}$$

- Αν $x_i > x_{\text{max}}$, τότε

$$\frac{x_{\text{max}} - x_0}{dx} = \frac{y - y_0}{dy} \Rightarrow y = \frac{dy}{dx} \cdot (x_{\text{max}} - x_0) + y_0$$
$$\frac{x_{\text{max}} - x_0}{dx} = \frac{z - z_0}{dz} \Rightarrow z = \frac{dz}{dx} \cdot (x_{\text{max}} - x_0) + z_0$$
$$x = x_{\text{max}}$$

- Αν $y_i < y_{\text{min}}$, τότε

$$\frac{y_{\text{min}} - y_0}{dx} = \frac{x - x_0}{dy} \Rightarrow x = \frac{dx}{dy} \cdot (y_{\text{min}} - y_0) + x_0$$
$$\frac{y_{\text{min}} - y_0}{dx} = \frac{z - z_0}{dz} \Rightarrow z = \frac{dz}{dy} \cdot (y_{\text{min}} - y_0) + z_0$$
$$y = y_{\text{min}}$$

- Αν $y_i > y_{\text{max}}$, τότε

$$\frac{y_{\text{max}} - y_0}{dx} = \frac{x - x_0}{dy} \Rightarrow x = \frac{dx}{dy} \cdot (y_{\text{max}} - y_0) + x_0$$
$$\frac{y_{\text{max}} - y_0}{dx} = \frac{z - z_0}{dz} \Rightarrow z = \frac{dz}{dy} \cdot (y_{\text{max}} - y_0) + z_0$$
$$y = y_{\text{max}}$$

- Αν $z_i < z_{\text{min}}$, τότε

$$\frac{z_{\text{min}} - z_0}{dx} = \frac{x - x_0}{dy} \Rightarrow x = \frac{dx}{dz} \cdot (z_{\text{min}} - z_0) + x_0$$
$$\frac{z_{\text{min}} - z_0}{dx} = \frac{y - y_0}{dz} \Rightarrow y = \frac{dy}{dz} \cdot (z_{\text{min}} - z_0) + y_0$$
$$z = z_{\text{min}}$$

- Αν $z_i > z_{\text{max}}$, τότε

$$\frac{z_{\text{max}} - z_0}{dx} = \frac{x - x_0}{dy} \Rightarrow x = \frac{dx}{dz} \cdot (z_{\text{max}} - z_0) + x_0$$
$$\frac{z_{\text{max}} - z_0}{dx} = \frac{y - y_0}{dz} \Rightarrow y = \frac{dy}{dz} \cdot (z_{\text{max}} - z_0) + y_0$$
$$z = z_{\text{max}}$$

όπου $i : 0, 1$.

Στο τρίτο και τελικό στάδιο, ο αλγόριθμος σχεδιάζει το ευθύγραμμο τμήμα μεταξύ των σημείων P_0 και P_1 των οποίων οι συντεταγμένες ενδεχομένως έχουν μεταβληθεί από τη διαδικασία της αποκοπής.

Όπως και στην διδιάστατη έκδοση, έτσι και εδώ, ο αλγόριθμος ποτέ δεν πραγματοποιεί διαίρεση με το μηδέν στις εξισώσεις που χρησιμοποιεί εξαιτίας των ελέγχων του πρώτου βήματος.

Τα βήματα του αλγορίθμου συνοπτικά είναι:

1. Έλεγξε αν και τα δύο σημεία που προσδιορίζουν το ευθύγραμμο τμήμα βρίσκονται στην ίδια περιοχή του ορθογωνίου όγκου θέασης. Αν αυτό συμβαίνει, τότε σταμάτησε αλλιώς προχώρησε στο Βήμα 2.

2. Σύγκρινε κάθε μία από τις συντεταγμένες των σημείων με τα σύνορα του όγκου θέασης (αριστερά, δεξιά, άνω, κάτω, εμπρός, πίσω). Αν κάποια από τις συντεταγμένες είναι εκτός των συνόρων, τότε αντικατέστησε

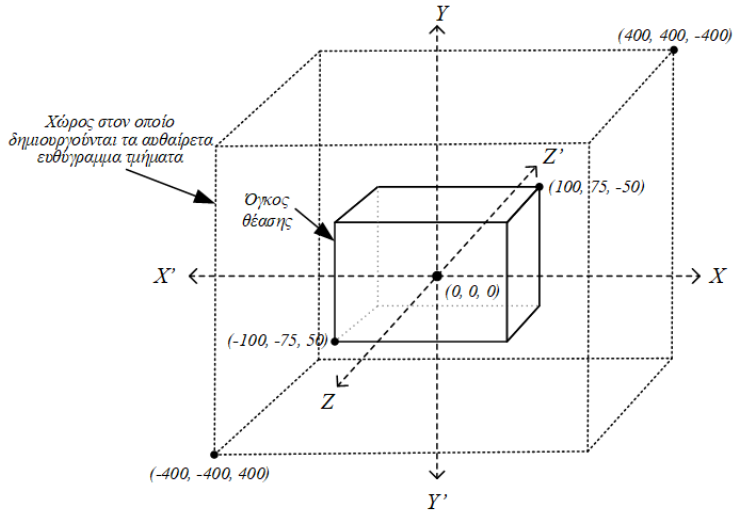
την με την συντεταγμένη του συνόρου και υπολόγισε τις υπόλοιπες μέσω της εξίσωσης της γραμμής στις τρεις διαστάσεις.

3. Σχεδίασε το ευθύγραμμο τμήμα μεταξύ των σημείων του.

Ο Κώδικας A.21 του Παραρτήματος A υλοποιεί τον προτεινόμενο αλγόριθμο.

4.6 Αξιολόγηση όλων των αλγορίθμων

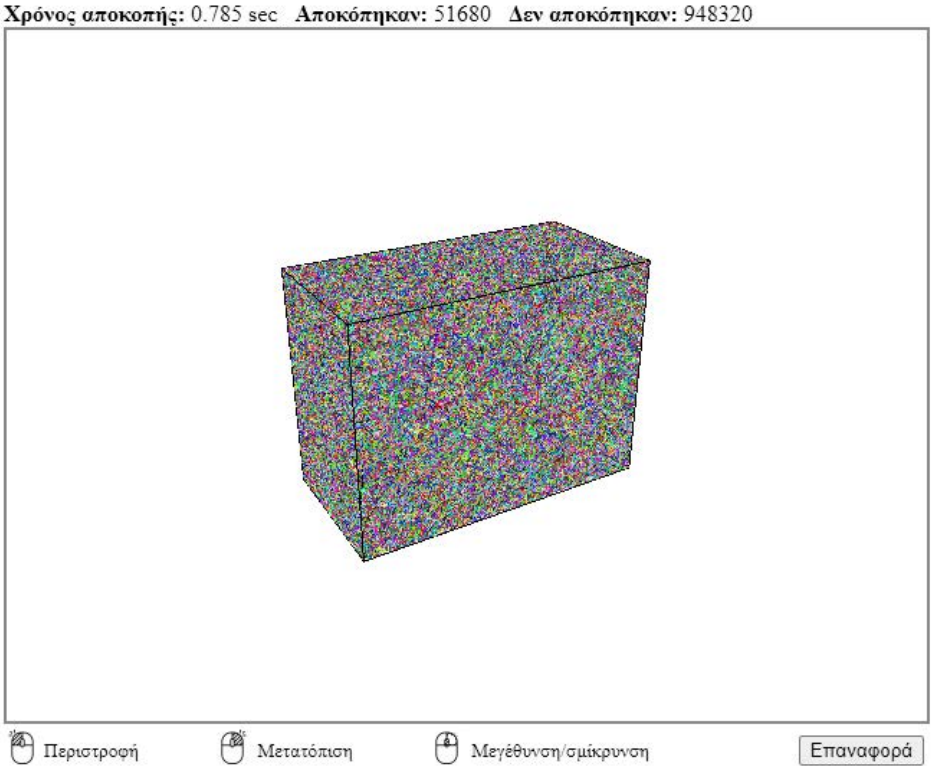
Προκειμένου να προσδιορίσουμε την αποτελεσματικότητα των παραπάνω αλγορίθμων οι οποίοι σχετίζονται με την αποκοπή ευθυγράμμων τμημάτων σε όγκο θέασης στις τρεις διαστάσεις, χρησιμοποιήσαμε την διαδικτυακή εφαρμογή που βρίσκεται στη διεύθυνση: https://matthes.sites.sch.gr/clipping. Η διαδικασία της αξιολόγησης είναι παρόμοια με εκείνη που χρησιμοποιήθηκε στα προηγούμενα κεφάλαια: Κατά την εκτέλεση κάθε αλγορίθμου αποκόπηκε ένας μεγάλος πλήθος αυθαιρέτων ευθυγράμμων τμημάτων στις τρεις διαστάσεις τα οποία δημιουργήθηκαν σε έναν ευρύτερο χώρο που προσδιορίζεται από τα σημεία $(-400, -400, -400)$ και $(400, 400, 400)$. Ο αρχικός όγκος θέασης είχε διάσταση 200 εικονοστοιχεία πλάτος, 150 εικονοστοιχεία ύψος και 100 εικονοστοιχεία βάθος. Η αρχή των αξόνων βρίσκεται στο κέντρο της περιοχής σχεδίασης (βλ. Σχ. 4.19).



Σχήμα 4.19: Οι χώροι στους οποίους δημιουργούνται και αποκόπτονται τα αυθαίρετα ευθύγραμμα τμήματα

Κάθε αλγόριθμος εκτελέστηκε δέκα φορές, ενώ καταγράφηκε ο χρόνος που χρειάστηκε για να αποκόψει και για να σχεδιάσει τα ευθύγραμμα τμήματα σε κάθε εκτέλεση. Στο τέλος, ο μέσος χρόνος των δέκα εκτελέσεων

υπολογίστηκε και αποτέλεσε το κριτήριο αξιολόγησης του αλγορίθμου ως προς την ταχύτητα. Όλες οι μετρήσεις έγιναν με τη χρήση του ίδιου υπολογιστικού συστήματος του οποίου οι προδιαγραφές είναι οι ακόλουθες: α) επεξεργαστής Intel Core i7-9750H @ 2.60 GHz CPU, β) μνήμη RAM 16 GB, γ) κάρτα γραφικών NVIDIA GeForce RTX 2070/8 GB, δ) λειτουργικό σύστημα Windows 10 Pro 64 bit. Η ταχύτητα σύνδεσης στο διαδίκτυο δεν επηρέασε την ταχύτητα εκτέλεσης της εφαρμογής αξιολόγησης αφού η δεύτερη λειτουργεί σε τοπικό επίπεδο στον υπολογιστή μετά την έναρξή της (βλ. Σχ. 4.20).

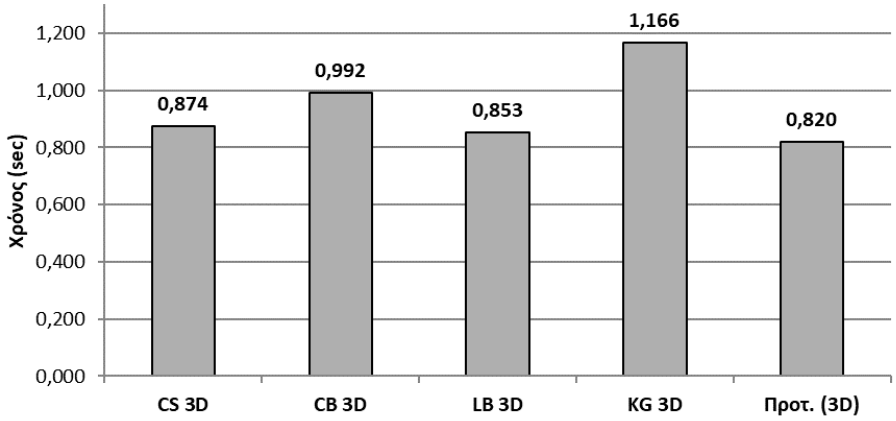


Σχήμα 4.20: Το αποτέλεσμα της εφαρμογής μετά την αποκοπή 1.000.000 ευθυγράμμων τμημάτων

Στον Πίνακα 4.1 φαίνονται οι χρόνοι για την αποκοπή 1.000.000 ευθυγράμμων τμημάτων ενώ στο Σχ. 4.21 απεικονίζεται η γραφική παράσταση των χρόνων αυτών.

Αρ. εκτέλεσης	CS 3D (sec)	LB 3D (sec)	CB 3D (sec)	KG 3D (sec)	Προτ. (3D) (sec)
1	0.898	1.028	0.969	1.128	0.793
2	0.894	0.998	0.826	1.111	0.820
3	0.843	1.005	0.836	1.121	0.806
4	0.886	1.030	0.829	1.221	0.810
5	0.958	0.993	0.894	1.218	0.804
6	0.841	0.970	0.834	1.130	0.858
7	0.833	0.968	0.835	1.224	0.890
8	0.871	0.973	0.819	1.235	0.773
9	0.867	0.977	0.822	1.156	0.823
10	0.847	0.976	0.867	1.118	0.818
Μέσος χρόνος:	**0.874**	**0.992**	**0.853**	**1.166**	**0.820**

Πίνακας 4.1: Οι μέσοι χρόνοι εκτέλεσης για την αποκοπή 1.000.000 ευθυγράμμων τμημάτων



Σχήμα 4.21: Μέσος χρόνος κάθε αλγορίθμου για την αποκοπή 1.000.000 ευθυγράμμων τμημάτων

Από τα αποτελέσματα διαπιστώνουμε ότι ο προτεινόμενος αλγόριθμος είναι ταχύτερος των Cohen-Sutherland κατά 6,63%, Cyrus-Beck κατά 21,03%, Liang-Barsky κατά 4,10% και τέλος κατά 42,31% από τον Kolingerová.

Η αξιολόγηση πραγματοποιήθηκε, επίσης, χρησιμοποιώντας διαφορετικά πλήθη ευθυγράμμων τμημάτων καθώς και όγκους θέασης διαφορετικών διαστάσεων. Επιπλέον, ελέγχθηκε η αποτελεσματικότητά τους και στην περίπτωση που το πλήθος των ευθυγράμμων τμημάτων βρίσκεται εξ ολοκλήρου εντός του όγκου θέασης ή εξ ολοκλήρου εκτός. Τα αποτελέσματα σε όλες τις περιπτώσεις ήταν σε πλήρη αναλογία με τα αποτελέσματα που παρουσιάστηκαν πιο πάνω.

4.7 Στατιστικός έλεγχος των αποτελεσμάτων

Για να διαπιστώσουμε αν τα αριθμητικά αποτελέσματα και τα συμπεράσματα της προηγούμενης παραγράφου είναι έγκυρα και στατιστικώς ορθά, χρησιμοποιήθηκε ξανά η δοκιμασία Wilcoxon. Μετά την εισαγωγή των δεδομένων, συγκρίνοντας τον προτεινόμενο αλγόριθμο με τους υπόλοιπους, τα αποτελέσματα που ελήφθησαν απεικονίζονται στον Πίνακα 4.2:

Προτεινόμενος αλγόριθμος (3D)					
Σύγκριση με	**W**	**p-value**	**Διάστημα εμπιστοσύνης 95%**		**Διαφορά**
			Από	**Μέχρι**	
CS 3D	89	0.002089	0.023	0.082	0.052
CB 3D	100	1.083e-05	0.150	0.201	0.1725
LB 3D	82	0.01469	0.006	0.061	0.0245
KG 3D	100	1.083e-05	0.305	0.406	0.3355

Πίνακας 4.2: Στατιστικός έλεγχος των αποτελεσμάτων από την σύγκριση του προτεινόμενου αλγορίθμου με τους υπόλοιπους με χρήση της δοκιμασίας Wilcoxon

Αν εξετάσουμε την πρώτη περίπτωση, δηλαδή την σύγκριση του προτεινόμενου αλγορίθμου με τον CS 3D, και δεδομένου ότι p-value = 0.002089, άρα μικρότερη του 0.05, συμπεραίνουμε ότι ο προτεινόμενος έχει στατιστικά σημαντική διαφορά ως προς τον χρόνο υλοποίησης έναντι του CS 3D. Το διάστημα εμπιστοσύνης 95% για τη διαφορά μεταξύ του χρόνου υλοποίησης του CS 3D έναντι προτεινόμενου δίνεται ως (0.023, 0.082). Δεδομένου ότι η διαφορά έχει σταθερά θετικό πρόσημο, η στατιστικά σημαντική διαφορά που διαπιστώθηκε νωρίτερα οδηγεί στο συμπέρασμα ότι ο χρόνος του αλγορίθμου CS 3D είναι στατιστικώς σημαντικά μεγαλύτερος από τον χρόνο του προτεινόμενου αλγορίθμου. Ομοίως, εξάγουμε τα ίδια συμπεράσματα και για τις υπόλοιπες συγκρίσεις.

Μέρος ΙΙ

ΕΞΑΚΡΙΒΩΣΗ ΠΑΡΑΜΕΤΡΩΝ ΔΙΔΙΑΣΤΑΤΗΣ ΜΟΡΦΟΚΛΑΣΜΑΤΙΚΗΣ ΠΑΡΕΜΒΟΛΗΣ

5. ΜΟΡΦΟΚΛΑΣΜΑΤΙΚΕΣ ΕΠΙΦΑΝΕΙΕΣ ΠΑΡΕΜΒΟΛΗΣ

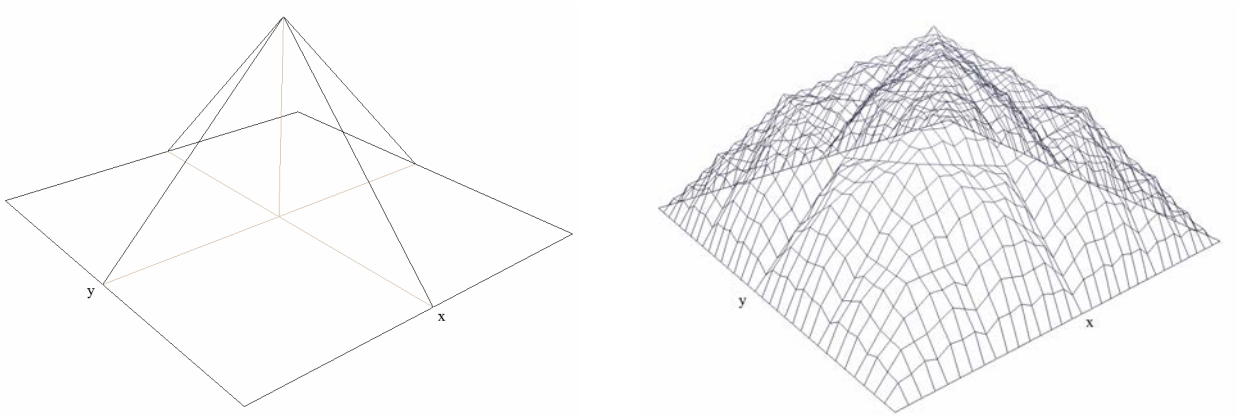
Οι παρουσιασθέντες στα προηγούμενα κεφάλαια αλγόριθμοι αποκοπής που δημιουργήθηκαν από τον συγγραφέα και τον επιβλέποντα έχουν ένα κοινό χαρακτηριστικό στον τρόπο λειτουργίας τους. Σε όλες τις περιπτώσεις αποκοπής ευθυγράμμων τμημάτων, είτε στις δύο είτε στις τρεις διαστάσεις, εφ' όσον το παράθυρο αποκοπής ή ο όγκος θέασης είναι κυρτά σύνολα, ο εκάστοτε αλγόριθμος εξετάζει κάθε σύνορο αποκοπής «εναγκαλιζόμενος» το ευθύγραμμο τμήμα και αποκόπτοντάς το σταδιακά. Η διαπίστωση αυτή οδήγησε την έρευνα στα μορφοκλάσματα, όπου για την κατασκευή Μορφοκλασματικών Επιφανειών Παρεμβολής και την εύρεση της βέλτιστης τιμής της ελεύθερης παραμέτρου κατακόρυφης κλιμάκωσης απαιτείται ο υπολογισμός της τομής κυρτών πολυέδρων. Η λογική του εναγκαλισμού βρίσκει απόλυτη εφαρμογή στην περίπτωση αυτή.

Ως μορφόκλασμα ή αλλιώς μορφοκλασματικό σύνολο θεωρούμε ένα «απείρως περίπλοκο» γεωμετρικό αντικείμενο το οποίο επαναλαμβάνεται σε άπειρο βαθμό μεγέθυνσης. Τα επαναλαμβανόμενα υποδείγματα των μορφοκλασμάτων έχουν εξάψει την φαντασία μαθηματικών, επιστημόνων, καλλιτεχνών και πολλών άλλων, με εφαρμογές που κυμαίνονται από την μοντελοποίηση φυσικών φαινομένων έως και την γραφική υπολογιστών. Μία από αυτές είναι η δημιουργία επιφανειών γνωστές ως «Μορφοκλασματικές Επιφάνειες Παρεμβολής», ή απλώς Μ.Ε.Π., οι οποίες δημιουργούνται μέσω παρεμβολής από ένα δεδομένο σύνολο σημείων κάνοντας χρήση της μορφοκλασματικής γεωμετρίας. Η μορφοκλασματική παρεμβολή αξιοποιεί τις ιδιότητες των μορφοκλασμάτων για να δημιουργήσει τις Μ.Ε.Π. που παρεμβάλλουν ένα σύνολο σημείων (βλ. Σχ. 5.1).

Σε αντίθεση με τις παραδοσιακές μεθόδους παρεμβολής που χρησιμοποιούν συναρτήσεις πολυωνύμων ή καμπύλες spline, η μορφοκλασματική παρεμβολή χρησιμοποιεί Επαναλαμβανόμενα Συστήματα Συναρτήσεων, ή εν συντομία Ε.Σ.Σ., για τη δημιουργία αυτοόμοιων δομών.

5.1 Επιστημονική έρευνα

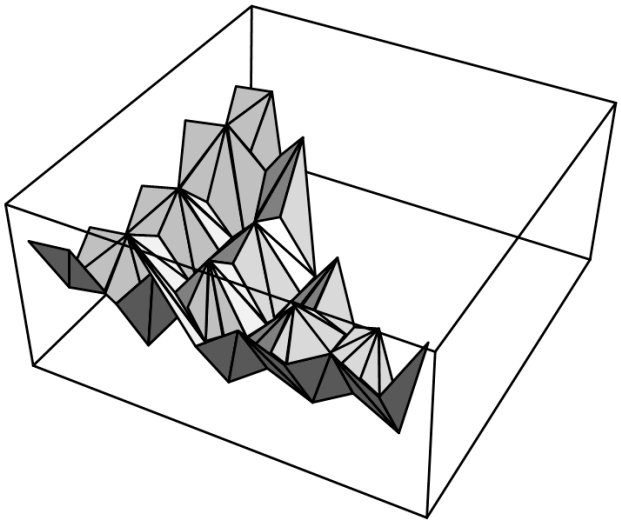
Η κατασκευή μορφοκλασματικών επιφανειών παρεμβολής χρησιμοποιώντας ένα καταλλήλως επιλεγμένο επαναλαμβανόμενο σύστημα συναρτήσεων προτάθηκε αρχικά από τον Peter R. Massopust στο [Mas90], όπου εξε-



(α) Αρχικά σημεία *(β)* Μετά από μορφοκλασματική παρεμβολή

Σχήμα 5.1: Μορφοκλασματική Επιφάνεια Παρεμβολής

τάζεται η περίπτωση των τριγωνικών χωρίων με συνεπίπεδα συνοριακά δεδομένα. Συγκεκριμένα, ο Massopust χρησιμοποίησε δεδομένα τοποθετημένα σε ένα τριγωνικό χωρίο, συνευθειακά σε κάθε πλευρά του χωρίου αυτού, και κηδεστικές απεικονίσεις ορισμένες στον $\mathbb{R}^3$ (βλ. Σχ. 5.2), διαμερίζοντας το χωρίο σε μικρότερα τριγωνικά χωρία (τομείς). Για να προκύψει, όμως, συνεχής επιφάνεια, πρέπει τα υπόλοιπα σημεία παρεμβολής, που βρίσκονται σε κάθε πλευρά του τριγωνικού χωρίου, να είναι συνευθειακά έτσι ώστε οι επιφάνειες που προκύπτουν από δύο απεικονίσεις που αντιστοιχούν σε γειτονικούς τομείς να μοιράζονται κοινές ακμές.



Σχήμα 5.2: Παράδειγμα Μ.Ε.Π. σύμφωνα με την κατασκευή του Massopust

Η κατασκευή αυτή στερείται της απαραίτητης προσαρμοστικότητας για να κατασκευαστούν πολύπλοκες επι-

φάνειες που θα προσεγγίζουν φυσικά πρότυπα. Μία ελαφρώς γενικότερη κατασκευή προτάθηκε από τους Jeffrey S. Geronimo και Douglas Hardin στο [GH93], όπου εξετάζονται πολυγωνικές περιοχές με αυθαίρετα σημεία παρεμβολής αλλά ίσους παράγοντες κατακόρυφης κλιμάκωσης. Στο [Mas93], παρουσιάζονται μορφοκλασματικές συναρτήσεις με τιμές στον $\mathbb{R}^m$. Οι δύο τελευταίες κατασκευές χρησιμοποιούν αναδρομικά Ε.Σ.Σ. Η μεθοδολογία του Craig M. Wittenbrink στο [Wit95], είτε παράγει ασυνεχείς συναρτήσεις είτε ανάγεται στην περίπτωση των σταθερών παραγόντων κατακόρυφης κλιμάκωσης. Ο Nailiang Zhao στο [Zha96] χειρίζεται τους παράγοντες ως μία συνεχή «συνάρτηση συστολής» και χρησιμοποιεί συνεπή τριγωνισμό για να εξασφαλίσει τη συνέχεια, αποφεύγοντας έτσι τη χρήση συνευθειακών σημείων παρεμβολής στα σύνορα του χωρίου. Το κοινό σύνορο δύο γειτονικών τριγωνικών τομέων αποτελεί την εικόνα του ίδιου συνόρου του μεγάλου χωρίου μέσω οποιασδήποτε από τις απεικονίσεις που αντιστοιχούν στους τομείς αυτούς.

Όλες οι προηγούμενες μέθοδοι χρησιμοποιούν τριγωνικά υποχωρία. Καθώς είναι δυνατή η κατασκευή Μ.Ε.Π. ως τανυστικό γινόμενο συνεχών μονομεταβλητών συναρτήσεων, ο Peter R. Massopust στο [Mas95] πρότεινε την κατασκευή τους ως τανυστικό γινόμενο δύο Μ.Ε.Π. Η προκύπτουσα επιφάνεια ορίζεται μονοσήμαντα από τον υπολογισμό της σε ζεύγη γειτονικών πλευρών του ορθογώνιου χωρίου. Οι Heping Xie και Hongquan Sun στο [XS98] πρότειναν την κατασκευή ενός έλκτη ο οποίος περιέχει τα σημεία παρεμβολής ενός συνόλου δεδομένων σημείων ορθογώνιας μορφής, αλλά δεν είναι εν γένει συνεχής. Η μεθοδολογία αυτή ακολουθήθηκε από τους ίδιους συγγραφείς και στο [XS11].

Η επίλυση του προταθέντος από τον Nailang Zhao προβλήματος επέτρεψε την κατασκευή των Jeffery R. Price και Monson H.Hayes III στο [PH98], όπου χρησιμοποιείται μεθοδολογία βασισμένη σε περιστροφές και ανακλάσεις για την κατασκευή Μ.Ε.Π. επί ορθογώνιων δικτυωμάτων. Στο [XF99], ορίζονται από τους Heping Xie και Zhigang Feng μορφοκλασματικές επιφάνειες αστρικού γινομένου. Οι Heping Xie και Hongquan Sun μαζί με τους Yang Ju και Zhigang Feng δημοσίευσαν μία βελτιωμένη εκδοχή του προηγούμενου άρθρου τους, χωρίς όμως να αποδεικνύουν τα βασικά αποτελέσματα (βλ. [Xie+01]). Η Λεώνη Δάλλα, στο [Dal02], έλυσε ένα παρόμοιο πρόβλημα για την ειδική περίπτωση όπου τα σημεία παρεμβολής σε κάθε ακμή του ορθογώνιου χωρίου είναι συγγραμμικά, προτείνοντας επίσης μία γενική κατασκευή παρόμοια με το [PH98]. Μη γραμμικές γενικεύσεις της μορφοκλασματικής παρεμβολής, μίας και δύο μεταβλητών, για εφαρμογές στην επεξεργασία εικόνων προτάθηκαν στο [KP05].

Διμεταβλητές, μη τανυστικού γινομένου Μ.Ε.Π., ορισμένες επί ορθογώνιων χωρίων κατασκευάστηκαν πλήρως από τον Xiao-yuan Qian στο [Qia02]. Οι Hongquan Sun και Heping Xie επανέλαβαν στο [SX04] τη μεθοδολογία

του [Xie+01]. Ο Hong-Yong Wang στο [Wan06], βασισμένος σε εργασίες όπως του Xiao-yuan Qian αλλά και της Λεώνης Δάλλα, χρησιμοποίησε μία ευρεία τάξη τριδιάστατων Ε.Σ.Σ. και έδειξε ότι οι ελκυστές τους αποτελούν μία κατηγορία Μ.Ε.Π. Ο Robert Malysz παρουσίασε στο [Mał06] μία κατασκευή διμεταβλητών Μ.Ε.Π. Οι παράγοντες συσταλτικότητας είναι σταθεροί και το Ε.Σ.Σ. αποτελείται από γραμμικές οριζόντιες συστολές και κατακόρυφες συσταλτικές απεικονίσεις που είναι τετραγωνικά πολυώνυμα. Ο Zhigang Feng στο [Fen08] διερευνά εργασίες όπως του Xiao-yuan Qian αλλά και του Hong-Yong Wang για Μ.Ε.Π. επί ορθογωνίων χωρίων. Τέλος, η κατασκευή των Wolfgang Metzler και Cholhui Yun στο [MY08] βασίζεται στα [PH98] και [Zha96].

5.2 Προαπαιτούμενες μαθηματικές έννοιες

Ακολουθεί η παρουσίαση ορισμένων μαθηματικών εννοιών οι οποίες κρίνονται απαραίτητες για τον ορισμό της μορφοκλασματικής παρεμβολής και την κατασκευή Μ.Ε.Π. Οι έννοιες αυτές προέρχονται κυρίως από τους κλάδους της γενικής τοπολογίας και της πραγματικής ανάλυσης. Αρκετές από αυτές αναφέρονται στην [Δρα19], στην [Σγο12] καθώς και στην [Μπο06].

5.2.1 Μετρικοί χώροι

Μία πολύ βασική έννοια των μαθηματικών είναι ο *μετρικός χώρος*. Αποτελείται από ένα μη κενό σύνολο X όπου μπορούμε να μετρήσουμε την απόσταση ρ μεταξύ δύο σημείων του. Η απόσταση αυτή ορίζεται από μία συνάρτηση που ονομάζεται *μετρική* και το σύνολο σε συνδυασμό με την μετρική συνήθως συμβολίζεται με (X, ρ).

Ορισμός 5.2.1. *Εάν X είναι ένα μη κενό σύνολο, τότε μετρική στο X είναι μία απεικόνιση $\rho\colon X \times X \to \mathbb{R}$ η οποία ικανοποιεί τις ακόλουθες ιδιότητες:*

1. Μη αρνητικότητα*: Η απόσταση μεταξύ δύο σημείων είναι μεγαλύτερη ή ίση του μηδενός $(\rho(x, y) \geq 0)$. Η απόσταση μεταξύ δύο σημείων είναι μηδέν, τότε και μόνο, αν τα σημεία συμπίπτουν $(\rho(x, y) = 0$ αν και μόνο αν $x = y)$.*

2. Συμμετρία*: Η απόσταση από το x προς το y είναι ίση με την απόσταση από το y προς το x $(\rho(x, y) = \rho(y, x))$.*

3. Τριγωνική ανισότητα*: Η απόσταση από το x προς το z είναι μικρότερη ή ίση με το άθροισμα των αποστάσεων από το x προς το y και από το y προς το z $(\rho(x, z) \leq \rho(x, y) + \rho(y, z))$.*

Ένα παράδειγμα μετρικού χώρου είναι ο Ευκλείδειος χώρος διάστασης n ή, διαφορετικά, το σύνολο

$$\mathbb{R}^n = \{(x_1, x_2, \dots, x_n)\colon x_i \in \mathbb{R}, \quad i = 1, 2, \dots, n\}$$

εφοδιασμένο με τη μετρική

$$\rho(x,y) = \|x-y\| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \quad \text{όπου} \quad x = (x_1, x_2, \dots, x_n), \quad y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^n$$

Για ένα δεδομένο σύνολο X, είναι δυνατό να ορισθούν διαφορετικές μετρικές, ορίζουσες η κάθε μία διαφορετικούς μετρικούς χώρους.

Ορισμός 5.2.2. *Έστωσαν (X,ρ) μετρικός χώρος και $k > 0$ τέτοιο, ώστε $\rho(x,y) \leq k$ για κάθε $x, y \in X$. Τότε ο (X,ρ) ονομάζεται* φραγμένος.

Ένας πεπερασμένος μετρικός χώρος $(X = \{x_1, x_2, \dots, x_n\}, \rho)$ είναι φραγμένος με φράγμα το $\max\{\rho(x_i, y_j) : 1 \leq i, j \leq n\}$, εν αντιθέσει με τον μετρικό χώρο $(\mathbb{R}, \rho)$ υπό τη συνήθη μετρική ο οποίος είναι μη φραγμένος.

Ορισμός 5.2.3. *Ένα υποσύνολο $A \subset X$ του μετρικού χώρου (X,ρ) είναι* φραγμένο*, αν υπάρχουν $x \in X$ και $r > 0$ τέτοια, ώστε για κάθε $a \in X$ ισχύει $\rho(x,a) < r$.*

Σημαντικοί είναι και οι παρακάτω ορισμοί ισοδυναμίας κατά Lipschitz, που ακολουθούν.

Ορισμός 5.2.4. *Δύο μετρικές ρ και d ορισμένες επί ενός συνόλου* X *ονομάζονται* ισοδύναμες κατά Lipschitz *ή* φραγμένως ισοδύναμες*, αν υπάρχουν πραγματικές σταθερές $0 < c_1 < c_2 < \infty$ τέτοιες ώστε*

$$c_1\rho(x,y) \leq d(x,y) \leq c_2\rho(x,y),$$

για κάθε ζεύγος σημείων $x, y \in X$.

Ορισμός 5.2.5. *Δύο μετρικοί χώροι (X,ρ_X) και (Y,ρ_Y) ονομάζονται* ισοδύναμοι κατά Lipschitz *, αν υπάρχει μία αμφιμονοσήμαντη και επί συνάρτηση $h : X \to Y$ τέτοια ώστε η μετρική ρ επί του X ορισμένη ως*

$$\rho(x_1, x_2) = \rho_Y(h(x_1), h(x_2))$$

να είναι ισοδύναμη κατά Lipschitz προς την ρ_X.

5.2.2 Ανοικτά και κλειστά σύνολα

Ορισμός 5.2.6. *Έστω (X,ρ) μετρικός χώρος.*
- Το σύνολο $D(x,r) = \{y \in X : \rho(x,y) < r\}$ όπου $x \in X$ και $r > 0$, ονομάζεται ανοικτός δίσκος κέντρου x και

ακτίνας r *(ως προς τη μετρική ρ).*

- Το σύνολο $\Delta(x,r) = \{y \in X : \rho(x,y) \leq r\}$ *όπου* $x \in X$ *και* $r > 0$*, ονομάζεται* κλειστός δίσκος κέντρου x και ακτίνας r *(ως προς τη μετρική ρ).*

- Το σύνολο $\mathrm{K}(x,r) = \{y \in X : \rho(x,y) = r\}$ *όπου* $x \in X$ *και* $r > 0$ *ονομάζεται* κύκλος κέντρου x και ακτίνας r *(ως προς τη μετρική ρ).*

Ορισμός 5.2.7. *Έστω (X,ρ) μετρικός χώρος και $A \subset X$.*

α) Ένα σημείο $a \in A$ ονομάζεται εσωτερικό σημείο *του A, εάν υπάρχει $\varepsilon > 0$ τέτοιο ώστε $D(a,\varepsilon) \subset A$.*

β) Το σύνολο όλων των εσωτερικών σημείων του A καλείται (ανοικτός) πυρήνας *ή* εσωτερικό *του A και συμβολίζεται με A^0 ή με $intA$, δηλαδή $A^0 = \{x \in A :$ υπάρχει $\varepsilon > 0$ ώστε $D(x,\varepsilon) \subset A\}$*

γ) Έμφραξη *του A, συμβολίζεται με $\overline{A}$, είναι το μικρότερο κλειστό υποσύνολο του X που περιέχει το A.*

δ) Το A ονομάζεται ανοικτό *(ως προς τη μετρική ρ), εάν όλα τα σημεία του είναι εσωτερικά, δηλαδή αν $A = A^0$.*

ε) Το A ονομάζεται κλειστό *(ως προς τη μετρική ρ), αν το $X \setminus A$ είναι ανοικτό.*

Θεώρημα 5.2.1. *Έστω (X,ρ) μετρικός χώρος. Ισχύουν τα παρακάτω:*

α) Το $\emptyset$ και το X είναι ταυτοχρόνως ανοικτά και κλειστά υποσύνολα του X.

β) Η ένωση ανοικτών υποσυνόλων του X είναι ανοικτό υποσύνολο του X.

γ) Η ένωση κλειστών υποσυνόλων του X δεν είναι πάντα κλειστό υποσύνολο του X, εκτός αν τα σύνολα είναι πεπερασμένα.

δ) Η τομή πεπερασμένου πλήθους ανοικτών υποσυνόλων του X είναι πάντα ανοικτό υποσύνολο του X.

ε) Η τομή πεπερασμένου πλήθους κλειστών υποσυνόλων του X είναι πάντα κλειστό υποσύνολο του X.

Από τους ορισμούς, είναι φανερό ότι για κάθε $a, b \in \mathbb{R}$ τα διαστήματα $(a,b), (-\infty, a)$ καθώς επίσης και $(a, +\infty)$ είναι ανοικτά υποσύνολα του $\mathbb{R}$ και επομένως τα $[a,b], a < b, (-\infty, a]$ και $[a, +\infty)$ και είναι κλειστά υποσύνολα του $\mathbb{R}$.

Ορισμός 5.2.8. *Έστωσαν σύνολο X και ρ, d δύο μετρικές στο X. Οι ρ, d ονομάζονται* ισοδύναμες *, αν ισχύει ότι οι μετρικοί χώροι (X,ρ) και (X,d) έχουν τα ίδια ανοικτά σύνολα, δηλαδή ένα υποσύνολο $U \subset X$ είναι ανοικτό ως προς την μετρική ρ τότε και μόνο, αν είναι ανοικτό ως προς την d.*

5.2.3 Πλήρεις μετρικοί χώροι

Για τον ορισμό της πληρότητας ενός μετρικού χώρου, απαραίτητοι είναι οι ορισμοί της σύγκλισης και του ορίου μίας ακολουθίας, καθώς επίσης της φραγμένης και της Cauchy (ή βασικής) ακολουθίας.

Ορισμός 5.2.9. *Μία ακολουθία $(x_n)_{n\in\mathbb{N}}$ εντός ενός μετρικού χώρου (X,ρ) συγκλίνει προς ένα $x\in X$ (ως προς τη μετρική ρ), αν για κάθε $\varepsilon>0$, υπάρχει $n_0=n_0(\varepsilon)\in\mathbb{N}$ τέτοιο, ώστε $\rho(x_n,x)<\varepsilon$ για κάθε $n\geq n_0$. Το σημείο x καλείται το* όριο της ακολουθίας *και γράφουμε*

$$\lim_{n\to\infty}\rho(x_n,x)=0 \qquad \acute{\eta} \qquad \lim_{n\to\infty}x_n=x \qquad \acute{\eta} \qquad x_n\to x$$

Ορισμός 5.2.10. *Μία ακολουθία $(x_n)_{n\in\mathbb{N}}$ ονομάζεται* φραγμένη *τότε και μόνο, εάν υπάρχει κάποιο $k\geq 0$ τέτοιο, ώστε $|x_n|\leq k$ για κάθε $n\in\mathbb{N}$.*

Ορισμός 5.2.11. *Μία ακολουθία $(x_n)_{n\in\mathbb{N}}$ σ' έναν μετρικό χώρο (X,ρ) ονομάζεται* ακολουθία Cauchy *ή* βασική ακολουθία*, εάν για κάθε $\varepsilon>0$ υπάρχει $n_0\in\mathbb{N}$ ώστε $\rho(x_n,x_m)<\varepsilon$ για κάθε $n,m\geq n_0$.*

Με βάση τα παραπάνω προκύπτει ο ακόλουθος ορισμός για την πληρότητα μετρικών χώρων.

Ορισμός 5.2.12. *Ένας μετρικός χώρος (X,ρ) ονομάζεται* πλήρης*, αν κάθε ακολουθία Cauchy στον (X,ρ) είναι συγκλίνουσα.*

5.2.4 Συμπάγεια

Έστω (X,ρ) ένας μετρικός χώρος.

Ορισμός 5.2.13. *Έστω $A\subset X$. Το A καλείται* ολικώς φραγμένο*, αν για κάθε $\varepsilon>0$ υπάρχουν $x_1,x_2,\dots,x_n\in X$, με $n\in\mathbb{N}$, ώστε $A\subset\bigcup_{i=1}^{n}D(x_i,\varepsilon)$.*

Ορισμός 5.2.14. *Έστω $B\subset X$. Μία οικογένεια $\{A_i\colon i\in I\}$, ανοικτών υποσυνόλων του X, λέγεται ότι είναι ένα* ανοικτό κάλυμμα *του B, εάν $B\subset\bigcup_{i\in I}A_i$.*

Ορισμός 5.2.15. *Έστω $B\subset X$. Το B καλείται* συμπαγές*, εάν κάθε ανοικτό κάλυμμα του B έχει ένα πεπερασμένο υποκάλυμμα, δηλαδή εάν για κάθε οικογένεια $\{A_i\colon i\in I\}$ ανοικτών συνόλων με $B\subset\bigcup_{i\in I}A_i$, υπάρχουν $n\in\mathbb{N}$ και $i_1,i_2,\dots,i_n\in I$ τέτοια, ώστε $B\subset\bigcup_{k=1}^{n}A_{i_k}$.*

Αποδεικνύεται ότι το καρτεσιανό γινόμενο συμπαγών συνόλων είναι συμπαγές σύνολο.

Θεώρημα 5.2.2. *Για τον μετρικό χώρο (X, ρ) ισχύει:*

α) Κάθε συμπαγές υποσύνολο του X είναι κλειστό.

β) Κάθε κλειστό υποσύνολο ενός συμπαγούς υποσυνόλου είναι συμπαγές.

Θεώρημα 5.2.3. *Αν (X, ρ) είναι συμπαγής μετρικός χώρος και $f : (X, \rho) \to (Y, d)$ είναι συνεχής, τότε το σύνολο $f(X)$ είναι συμπαγές υποσύνολο του Y.*

Ορισμός 5.2.16. *Μία οικογένεια υποσυνόλων $(F_i)_{i\in I}$ του X έχει την ιδιότητα της* πεπερασμένης τομής, *εάν για κάθε πεπερασμένο υποσύνολο G του I ισχύει ότι $\bigcap_{i\in G} f_i \neq \emptyset$.*

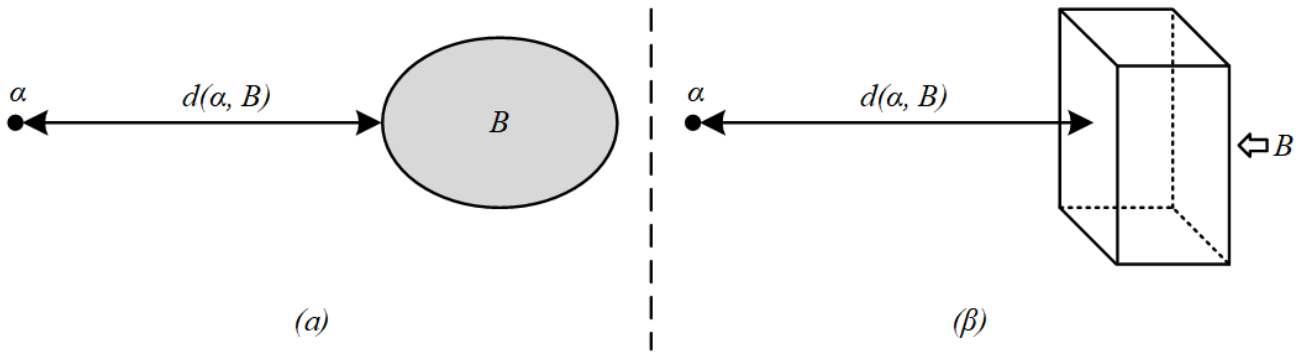
Πολύ σημαντικά για τη συμπάγεια ενός μετρικού χώρου είναι τα ακόλουθα δύο θεωρήματα.

Θεώρημα 5.2.4. *Ο μετρικός χώρος (X, ρ) είναι συμπαγής τότε και μόνον, εάν είναι πλήρης και ολικώς φραγμένος.*

Θεώρημα 5.2.5. *(ή Θεώρημα Heine–Borel): Ένα υποσύνολο K του $\mathbb{R}^n (n \geq 1)$ είναι συμπαγές τότε και μόνο, εάν είναι κλειστό και φραγμένο.*

5.2.5 Αποστάσεις μεταξύ συνόλων

Ορισμός 5.2.17. *Έστωσαν (X, ρ) μετρικός χώρος και $B \subset X$. Ένα σημείο $x \in X$ ονομάζεται συνοριακό σημείο του B αν $x \in \overline{B} \cap (\overline{X \setminus B})$. Τα συνοριακά σημεία του B τα συμβολίζουμε με ∂B*



c ncgf

6d8c

Σχήμα 5.3: Η απόσταση του σημείου α από το σύνολο B: (α) στον $\mathbb{R}^2$ και (β) στον $\mathbb{R}^3$

Ορισμός 5.2.18. *Έστωσαν (X, ρ) μετρικός χώρος, $a \in A$ και μη κενά υποσύνολα $A, B \subset X$. Απόσταση του σημείου a από το υποσύνολο B, συμβολίζεται με $d(a, B)$, ορίζεται να είναι*

$$d(a, B) = \inf\{\rho(a, b) \colon b \in B\}.$$

Η απόσταση

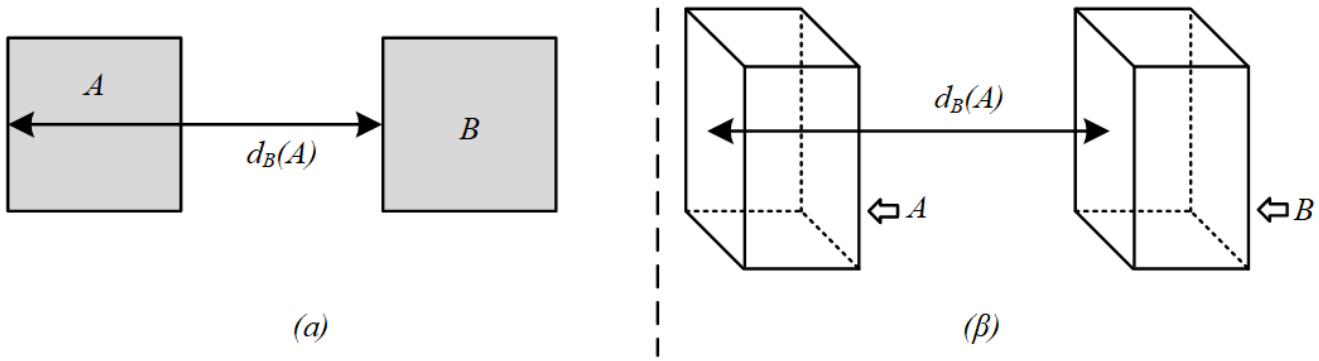
$$d_B(A) = \sup\{d(a, B) \colon a \in A\}$$

καλείται κατευθυνόμενη απόσταση Hausdorff από το A προς το B.

Αν τα σύνολα A, B στον παραπάνω ορισμό είναι συμπαγή, τότε ισχύει

$$d_B(A) = \max\{d(a, B) \colon a \in A\} = \max\{\min\{\rho(a, b) \colon b \in B\} \colon a \in A\}.$$

Ένα παράδειγμα απόστασης σημείου από σύνολο απεικονίζεται στο Σχήμα 5.3, ενώ ένα παράδειγμα κατευθυνόμενης απόστασης Hausdorff δύο συνόλων στο Σχήμα 5.4.



Σχήμα 5.4: Η κατευθυνόμενη απόσταση Hausdorff από το σύνολο Α προς το σύνολο Β. (α) στον $\mathbb{R}^2$ (β) στον $\mathbb{R}^3$

5.2.6 Η μετρική Hausdorff

Έστω (X, ρ) μετρικός χώρος. Ο χώρος του οποίου τα σημεία είναι τα συμπαγή, διάφορα του κενού, υποσύνολα του X, συμβολίζεται ως $\mathcal{H}(X)$, δηλαδή $\mathcal{H}(X) = \{\emptyset \neq A \subset X \colon A \text{ συμπαγές}\}$.

Ορισμός 5.2.19. *Η μετρική Hausdorff μεταξύ των συνόλων $A, B \in \mathcal{H}(X)$ ορίζεται ως*

$$h(A, B) = \max\{d_B(A), d_A(B)\}$$

Θεώρημα 5.2.6. *Έστω μετρικός χώρος (X, ρ). Η συνάρτηση h είναι μία μετρική επί του συνόλου $\mathcal{H}(X)$.*

Ο μετρικός χώρος $(\mathcal{H}(X), h)$ αναφέρεται ενίοτε ως ο «χώρος των μορφοκλασμάτων στο X» .

5.2.7 Η πληρότητα του $(\mathcal{H}(X), h)$

Ο μετρικός χώρος $(\mathcal{H}(\mathbb{R}^n), h)$ είναι ο χώρος εντός του οποίου κατασκευάζονται μορφοκλασματικά σύνολα με την βοήθεια ενός θεωρήματος σταθερού σημείου.

Θεώρημα 5.2.7. *Ο χώρος $(\mathcal{H}(\mathbb{R}^n), h)$ είναι πλήρης μετρικός χώρος.*

Το βασικό αυτό θεώρημα ισχύει και για τον $(\mathcal{H}(X), h)$, αν ο (X, ρ) είναι πλήρης μετρικός χώρος.

Θεώρημα 5.2.8. *(ή θεώρημα επιλογής του Blaschke): Έστω $(\mathcal{H}(B), h)$ το σύνολο των συμπαγών υποσυνόλων του συμπαγούς συνόλου $B \subset \mathbb{R}^n$. Τότε ο $(\mathcal{H}(B), h)$ είναι συμπαγής μετρικός χώρος.*

Περισσότερες λεπτομέρειες και αποδείξεις των Θεωρημάτων 5.2.6, 5.2.7, 5.2.8 μπορούν να βρεθούν στο [Δρα19].

5.2.8 Η τροποποιημένη απόσταση Hausdorff

Με βάση τα παραπάνω, στον ακόλουθο ορισμό ορίζεται η τροποποιημένη απόσταση Hausdorff.

Ορισμός 5.2.20. *Έστω (X, ρ) μετρικός χώρος και A, B μη κενά, πεπερασμένα υποσύνολα του X. Η τροποποιημένη απόσταση Hausdorff μεταξύ των συνόλων A και B ορίζεται ως*

$$\mathcal{H}_{MHD}(A, B) = \max\{d_{MHD}(A, B), d_{MHD}(B, A)\}$$

όπου

$$d_{MHD}(A, B) = \frac{1}{N_a} \sum_{a \in A} \rho(a, B)$$

και N_a συμβολίζει το πλήθος των στοιχείων του συνόλου A [DJ94].

5.2.9 *Η μετρική της συμμετρικής διαφοράς*

Η παρακάτω ανάλυση αναφέρεται στον Ευκλείδειο χώρο $\mathbb{R}^3$ με την συνήθη μετρική.

Ορισμός 5.2.21. *Ένα σύνολο $A \subset \mathbb{R}^3$ λέγεται ότι είναι κυρτό , αν για κάθε ζεύγος σημείων $a_1, a_2 \in A$ το ευθύγραμμο τμήμα που τα ενώνει περιέχεται εξ ολοκλήρου στο A.*

Το σύνολο των κυρτών, συμπαγών μη κενών υποσυνόλων του $\mathbb{R}^3$ συμβολίζεται ως K_0^3.

Ορισμός 5.2.22. *Η συμμετρική διαφορά δύο συνόλων $A, B \in K_0^3$ ορίζεται ως*

$$A \triangle B = (A \setminus B) \cup (B \setminus A).$$

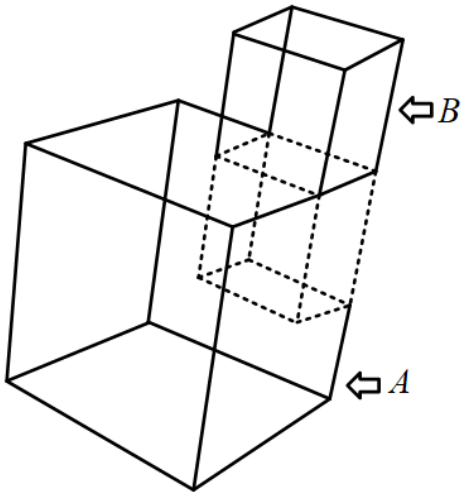
Ένα παράδειγμα συμμετρικής διαφοράς συνόλων στον $\mathbb{R}^3$ φαίνεται στο Σχήμα 5.5.

Ορισμός 5.2.23. *Η μετρική της συμμετρικής διαφοράς στο K_0^3 ορίζεται ως*

$$\delta_s(A, B) = \mathcal{H}^3(A \triangle B) = \mathcal{H}^3((A \setminus B) \cup (B \setminus A)), A, B \in K_0^3 \tag{5.1}$$

όπου $\mathcal{H}^3$ είναι το μέτρο Hausdorff στον $\mathbb{R}^3$. Η (5.1) σαν συνάρτηση των όγκων των συνόλων διατυπώνεται ισοδύναμα ως

$$\begin{aligned}\delta_s(A, B) &= Volume(A \setminus B) + Volume(B \setminus A) = \\ &= Volume(A \cup B) – Volume(A \cap B) = \\ &= Volume(A) + Volume(B) – 2 \times Volume(A \cap B)\end{aligned}$$



Σχήμα 5.5: Η συμμετρική διαφορά των συνόλων A, B είναι ο χώρος που καταλαμβάνουν τα σύνολα A, B μείον τον χώρο που καταλαμβάνει ο κύβος με τις διακεκομμένες γραμμές, ο οποίος αποτελεί την τομή των δύο αρχικών συνόλων A και B

Θεώρημα 5.2.9. *Η συνάρτηση* $\delta_s(A, B)$ *είναι μία μετρική επί του συνόλου* K_0^3.

6. ΕΠΑΝΑΛΑΜΒΑΝΟΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΣΥΝΑΡΤΗΣΕΩΝ

Ένα Επαναλαμβανόμενο Σύστημα Συναρτήσεων, ή απλώς Ε.Σ.Σ., ορίζεται ένα σύνολο συσταλτικών συναρτήσεων που εφαρμόζονται επαναληπτικά σε σημεία, οδηγώντας στη δημιουργία μορφοκλασματικών μοτίβων. Αυτά τα συστήματα είναι συνήθως πεπερασμένα, δηλαδή αποτελούνται από μία αλληλουχία πεπερασμένων καταστάσεων ή δυνατών τιμών, και η εξέλιξή τους στο χρόνο περιγράφεται επίσης από μία αλληλουχία συναρτήσεων κατάστασης.

Κάθε συνάρτηση στο επαναλαμβανόμενο σύστημα περιγράφει το πώς οι καταστάσεις του συστήματος εξελίσσονται από ένα χρονικό βήμα στο επόμενο, βασιζόμενες σε κάποιους κανόνες ή σχέσεις. Καθώς η εξέλιξη του συστήματος προχωρά, η συνάρτηση εφαρμόζεται επανειλημμένα για κάθε χρονικό βήμα, δημιουργώντας ένα σύνολο επαναλήψεων.

Τα επαναλαμβανόμενα συστήματα συναρτήσεων έχουν ευρεία εφαρμογή σε πολλούς τομείς, όπως η φυσική, η βιολογία, η οικονομία, η ρομποτική και άλλοι. Χρησιμοποιούνται για τη μοντελοποίηση και την ανάλυση δυναμικών συστημάτων που εμφανίζουν πολύπλοκη συμπεριφορά και αλληλεπιδράσεις μεταξύ των στοιχείων τους. Ως εκ τούτου, τα επαναλαμβανόμενα συστήματα συναρτήσεων κατατάσσονται ως ένας από τους βασικότερους μηχανισμούς δημιουργίας μορφοκλασματικών συνόλων.

6.1 Μετασχηματισμοί και απεικονίσεις συστολής

Οι μετασχηματισμοί στα Επαναλαμβανόμενα Συστήματα Συναρτήσεων αναφέρονται σε μεταβολές ή αλλαγές που εφαρμόζονται στο σύστημα με σκοπό να διευκολυνθεί η ανάλυσή του ή να εξεταστεί η συμπεριφορά του. Στην δημιουργία μορφοκλασμάτων χρησιμοποιούνται μετασχηματισμοί που επηρεάζουν την διαδικασία επανάληψης και τη δημιουργία των δομικών στοιχείων του μορφοκλάσματος. Οι μετασχηματισμοί αυτοί ονομάζονται κηδεστικοί.

Ορισμός 6.1.1. *Έστω (X, ρ) μετρικός χώρος. Ένας μετασχηματισμός $w\colon X \to X$ καλείται κηδεστικός, αν είναι δυνατόν να αναπαρασταθεί από ένα μητρώο* $\mathbf{A}$ *και μία μετατόπιση* $\mathbf{t}$ *ως $w(x) = \mathbf{A}x + \mathbf{t}$.*

Σαν παραδείγματα κηδεστικών μετασχηματισμών μπορούμε να θεωρήσουμε τον μετασχηματισμό:

$$w \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d \\ e \end{bmatrix}$$

όπου $X = \mathbb{R}^2$, καθώς επίσης και τον μετασχηματισμό:

$$w \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & g \\ h & k & s \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} l \\ m \\ r \end{bmatrix}$$

όπου $X = \mathbb{R}^3$.

Οι κηδεστικοί μετασχηματισμοί εφαρμόζονται επανειλημμένα σε αρχικά σημεία ή σύνολα σημείων στον χώρο, παράγοντας πλούσια δομική ποικιλομορφία και πολυπλοκότητα στο τελικό μορφόκλασμα. Ενδεικτικοί τέτοιοι μετασχηματισμοί είναι οι ακόλουθοι:

1. Μετατόπιση: Μετακινεί κάθε σημείο του μορφοκλάσματος κατά σταθερές αποστάσεις.
2. Κλιμάκωση: Αλλάζει την κλίμακα κάθε σημείου του μορφοκλάσματος, επηρεάζοντας το μέγεθος του κατά μία σταθερά.
3. Περιστροφή: Περιστρέφει κάθε σημείο του μορφοκλάσματος γύρω από ένα σταθερό κέντρο κατά ένα γωνιακό ποσοστό.
4. Ανάκλαση: Αντιστρέφει τη θέση των σημείων του μορφοκλάσματος κατά μήκος ενός καθορισμένου άξονα.
5. Αναδίπλωση: Αναδιπλώνει και συμπιέζει το μορφοκλάσμα, δημιουργώντας πιο σύνθετες δομές και λεπτομέρειες.

Ένας μετασχηματισμός που εφαρμόζεται σε ένα μορφόκλασμα με σκοπό να μειωθεί η κλίμακά του σε κάθε επανάληψη αποτελεί μία απεικόνιση συστολής, δημιουργώντας έτσι πιο λεπτομερείς δομές και πιο αναλυτικά σχήματα. Αυτό συνήθως επιτυγχάνεται με τον μετασχηματισμό κλιμάκωσης, όπου το μορφόκλασμα μειώνεται κατά μία σταθερή συντελεστική παράμετρο σε κάθε επανάληψη. Κατά τη διαδικασία της απεικόνισης συστολής, το μορφόκλασμα μειώνεται στην κλίμακα κατά τρόπο που οι δομικές λεπτομέρειες εμφανίζονται σε κάθε επίπεδο επανάληψης. Αυτό δημιουργεί μία ιεραρχία δομικών στοιχείων, με τα πιο λεπτομερή σχήματα να εμφανίζονται σε

μικρότερες κλίμακες και τα πιο γενικευμένα σχήματα να διατηρούνται σε μεγαλύτερες κλίμακες. Η απεικόνιση συστολής είναι ένας βασικός μηχανισμός στη δημιουργία μορφοκλασμάτων, καθώς συμβάλλει στη δημιουργία πλούσιας δομικής ποικιλομορφίας και λεπτομερειών σε όλα τα επίπεδα του μορφοκλάσματος.

Ένα απλό παράδειγμα απεικόνισης συστολής σε ένα μορφόκλασμα είναι το τρίγωνο του Sierpiński. Για τη δημιουργία του τριγώνου Sierpiński ακολουθείται μία σειρά απλών βημάτων προκειμένου να δημιουργηθεί ένα σχέδιο που αποτελείται από τρίγωνα, με κάθε τρίγωνο να χωρίζεται σε τρία μικρότερα τρίγωνα σε κάθε επαναληπτική διαδικασία. Πιο συγκεκριμένα, σε κάθε επανάληψη το μορφόκλασμα μειώνεται στο μισό μέγεθος και εφαρμόζονται τα παραπάνω βήματα για να δημιουργηθούν νέα τρίγωνα στο επίπεδο. Κάθε φορά, οι λεπτομέρειες του σχήματος αυξάνονται και παράγονται πιο αναλυτικά δομικά στοιχεία. Το τελικό αποτέλεσμα είναι ένα μορφόκλασμα που αποτελείται από πολλά μικρά τρίγωνα, με κάθε τρίγωνο να αναδύεται από τον εαυτό του σε μία ιεραρχία σχημάτων.

Σχήμα 6.1: Το τρίγωνο του Sierpiński

Προκειμένου να περιγραφεί η συμπεριφορά των συναρτήσεων μεταξύ δύο σημείων χρησιμοποιείται μία συνάρτηση Hölder, η οποία αποτελεί έναν τύπο συνάρτησης στη μαθηματική ανάλυση. Οι συναρτήσεις Hölder είναι αντιστοιχίες που ελέγχουν πόσο γρήγορα μία συνάρτηση μπορεί να αλλάξει τιμή καθώς μετακινείται από ένα σημείο στο άλλο.

Ορισμός 6.1.2. *Έστωσαν $(X, \rho), (Y, \sigma)$ μετρικοί χώροι και συνάρτηση $f : X \to Y$. Η συνάρτηση f ονομάζεται συνάρτηση Hölder εκθέτη a, αν υπάρχει πραγματικός αριθμός $k \geq 0$ τέτοιος, ώστε για κάθε $x, y \in X$ ισχύει*

$$\sigma(f(x), f(y)) \leq k\,[\rho(x, y)]^a, a \geq 0. \tag{6.1}$$

Αν επιπλέον $a = 1$, τότε η συνάρτηση f ονομάζεται Lipschitz και ισχύει

$$\sigma(f(x), f(y)) \leq k\,\rho(x, y). \tag{6.2}$$

Αν $k = 1$, η f ονομάζεται αδιάσταλτη , ενώ αν $k < 1$, ονομάζεται συστολή με παράγοντα συσταλτικότητας k.

Ορισμός 6.1.3. *Έστωσαν $(X, \rho), (Y, \sigma)$ μετρικοί χώροι και συνάρτηση $f : X \to Y$. Η συνάρτηση f ονομάζεται συσταλτική, αν για κάθε $x, y \in X$ ισχύει*

$$\sigma(f(x), f(y)) < \rho(x, y). \tag{6.3}$$

Συνοψίζοντας, σχηματικά έχουμε

$$\text{Συστολή} \Rightarrow \text{συσταλτική} \Rightarrow \text{αδιάσταλτη} \Rightarrow \text{Lipschitz}$$

6.2 Σταθερά σημεία μετασχηματισμών

Όπως έχει προαναφερθεί, τα επαναλαμβανόμενα συστήματα συναρτήσεων βασίζουν τη λειτουργία τους στην επαναληπτική εφαρμογή καταλλήλως ορισμένων συναρτήσεων. Για το σκοπό αυτό είναι απαραίτητος ο ορισμός της σύνθεσης ενός μετασχηματισμού με τον εαυτό του.

Ορισμός 6.2.1. *Έστωσαν μετρικός χώρος (X, ρ) και μετασχηματισμός $f : X \to X$. Η σύνθεση του f με τον εαυτό του n φορές, όπου $n \in \mathbb{N}$, συμβολίζεται ως f^n και είναι ίση με $f^n = \underbrace{f \cdot f \cdot \cdots \cdot f}_{n \text{παράγοντες}}$, όπου f^0 είναι ο ταυτοτικός μετασχηματισμός.*

Η σύγκλιση της μεθοδολογίας προς ένα επιθυμητό όριο γίνεται άμεσα κατανοητή με τον ορισμό του σταθερού σημείου συνάρτησης που ακολουθεί.

Ορισμός 6.2.2. *Σταθερό σημείο μιας συνάρτησης f είναι ένα σημείο x_0 του πεδίου ορισμού της, για το οποίο ισχύει ότι $f(x_0) = x_0$.*

Στην ίδια λογική κινείται και ο ορισμός του σταθερού συνόλου.

Ορισμός 6.2.3. *Έστωσαν μετρικός χώρος (X, ρ) και συνάρτηση $f : X \to X$. Ένα μη κενό υποσύνολο A του X καλείται σταθερό σύνολο για την συνάρτηση f, αν $f(A) = A$.*

Η μοναδικότητα του σταθερού σημείου εξασφαλίζεται από το παρακάτω θεώρημα του Σταθερού Σημείου του Banach.

Θεώρημα 6.2.1. *(Σταθερού Σημείου του Banach): Έστω f μετασχηματισμός συστολής σε έναν πλήρη μετρικό χώρο (X, ρ). Τότε υπάρχει ένα μοναδικό x_0 στον X το οποίο ικανοποιεί τις συνθήκες*

i. $f(x_0) = x_0$

ii. $\lim_{n\to\infty} f^n(x) = x_0$, *για κάθε* $x \in X$.

Για τον ορισμό των Επαναλαμβανόμενων Συστημάτων Συναρτήσεων απαραίτητοι είναι, επίσης, το λήμμα και οι ορισμοί που ακολουθούν.

Λήμμα 6.2.1. *Έστω* $w\colon X \to X$ *απεικόνιση συστολής στον μετρικό χώρο* (X, ρ) *με παράγοντα συσταλτικότητας* s. *Τότε η* $W\colon \mathcal{H}(X) \to \mathcal{H}(X)$ *με* $W(A) = \{w(a)\colon a \in A\}$ *για κάθε* $A \in \mathcal{H}(X)$ *είναι απεικόνιση συστολής με παράγοντα συσταλτικότητας* s.

Ορισμός 6.2.4. *Έστωσαν μετρικός χώρος* (X, ρ) *και μετασχηματισμοί* $w_m\colon X \to X, m = 1, 2, \dots, M$. *Η απεικόνιση αλληλοεπικάλυψης* $W\colon \mathcal{H}(X) \to \mathcal{H}(X)$ *ορίζεται ως*

$$W(E) = \bigcup_{m=1}^{M} w_m(E), \quad \textit{όπου} \quad E \in \mathcal{H}(X)$$

.

Ορισμός 6.2.5. *Έστωσαν* (X, ρ) *μετρικός χώρος και* $\{w_m\colon m = 1, 2, \dots, M\}$ *συναρτήσεις συστολής στον* $(\mathcal{H}(X), h)$ *με αντίστοιχους παράγοντες συσταλτικότητας* $\{s_m\colon m = 1, 2, \dots, M\}$. *Η συνάρτηση* $W\colon \mathcal{H}(X) \to \mathcal{H}(X)$ *με*

$$W(B) = w_1(B) \cup w_2(B) \cup \cdots \cup w_M(B)$$

όπου $B \in \mathcal{H}(X)$, *είναι συνάρτηση συστολής με παράγοντα συσταλτικότητας* $s = \max\{s_m\colon m = 1, 2, \dots, M\}$.

6.3 Μορφοκλασματικά σύνολα ως σταθερά σημεία

Με βάση τους ορισμούς των προηγούμενων ενοτήτων του κεφαλαίου, προκύπτει ο ορισμός των Επαναλαμβανόμενων Συστημάτων Συναρτήσεων.

Ορισμός 6.3.1. *Ένα Επαναλαμβανόμενο Σύστημα Συναρτήσεων, ή Ε.Σ.Σ., είναι μία συλλογή ενός πλήρους μετρικού χώρου* (X, ρ) *μαζί με ένα πεπερασμένο σύνολο συνεχών απεικονίσεων* $w_m\colon X \to X, m = 1, 2, \dots, M$, *όπου* ρ *είναι η συνάρτηση απόστασης μεταξύ των στοιχείων του* X. *Είναι σύνηθες ένα Ε.Σ.Σ. να συμβολίζεται ως*

$$\{X; w_1, w_2, \dots, w_M\} \quad \textit{ή} \quad \{X; w_{1-M}\}$$

Αν οι μετασχηματισμοί w_m είναι συστολές με αντίστοιχους παράγοντες συσταλτικότητας s_m για κάθε $m = 1, 2, \dots, M$, τότε το Ε.Σ.Σ. ονομάζεται υπερβολικό.

Με βάση τον ορισμό ενός Ε.Σ.Σ. προκύπτει το παρακάτω θεώρημα, καθώς επίσης και ο ορισμός του έλκτη που ακολουθεί.

Θεώρημα 6.3.1. *Έστω $\{X; w_{1-M}\}$ υπερβολικό Ε.Σ.Σ. με παράγοντα συσταλτικότητας s. Τότε ο μετασχηματισμός $W : \mathcal{H}(X) \to \mathcal{H}(X)$, με*

$$W(B) = \bigcup_{m=1}^{M} w_m(B)$$

για κάθε $B \in \mathcal{H}(X)$, είναι μία απεικόνιση συστολής επί του πλήρους μετρικού χώρου $(\mathcal{H}(X), h(\rho))$ με παράγοντα συσταλτικότητας s. Το μοναδικό σταθερό σημείο της $\mathcal{A} \in \mathcal{H}(X)$, ικανοποιεί την

$$\mathcal{A} = W(\mathcal{A}) = \bigcup_{m=1}^{M} w_m(\mathcal{A})$$

και δίνεται από την

$$\mathcal{A} = \lim_{k \to \infty} W^k(B)$$

για κάθε $B \in \mathcal{H}(X)$.

Απόδειξη. Επειδή ο $(\mathcal{H}(X), h)$ είναι πλήρης μετρικός χώρος και η W είναι μετασχηματισμός συστολής, το συμπέρασμα έπεται από το Θεώρημα Σταθερού Σημείου του Banach (Θεώρημα 6.2.1). □

Ορισμός 6.3.2. *Έλκτης ενός υπερβολικού Ε.Σ.Σ. με παράγοντα συσταλτικότητας s, καλείται το μοναδικό σύνολο $\mathcal{A}$ για το οποίο ισχύει*

$$\lim_{k \to \infty} W^k(A_0) = \mathcal{A},$$

για κάθε σύνολο εκκίνησης A_0.

Ο όρος έλκτης υποδεικνύει την μετακίνηση του A_0 προς το $\mathcal{A}$ ύστερα από διαδοχικές εφαρμογές του W. Το σύνολο $\mathcal{A}$ καλείται αναλλοίωτο σύνολο του Ε.Σ.Σ. επειδή είναι το μοναδικό σύνολο στον $\mathcal{H}(X)$ το οποίο δεν μεταβάλλεται από τον W, καθώς ισχύει $W(\mathcal{A}) = \mathcal{A}$.

Οι ελκυστές των Ε.Σ.Σ. εμφανίζουν μορφοκλασματικά χαρακτηριστικά και χρησιμοποιούνται για την κατασκευή μορφοκλασματικών συνόλων στον μετρικό χώρο $(\mathcal{H}(\mathbb{R}^n), h)$ με Ε.Σ.Σ. των οποίων οι μετασχηματισμοί

είναι κηδεστικοί. Όταν το Ε.Σ.Σ. ορίζεται με στόχο την προσέγγιση δεδομένου συνόλου από τον έλκτη του τότε το ακόλουθο θεώρημα παρέχει ένα φράγμα για την ακρίβεια της προσέγγισης αυτής.

Θεώρημα 6.3.2. *(Θεώρημα Αλληλοεπικάλυψης) Έστω (X, ρ) πλήρης μετρικός χώρος, $L \in \mathcal{H}(X)$ και $\varepsilon \geq 0$. Επιλέξτε ένα Ε.Σ.Σ. $\{X; w_m : m = 1, 2, \ldots, M\}$ με παράγοντα συσταλτικότητας $0 \leq s < 1$ ούτως ώστε*

$$h\left(L, \bigcup_{m=1}^{M} w_m(L)\right) \leq \varepsilon$$

Τότε

$$h(L, \mathcal{A}) \leq \frac{\varepsilon}{1-s}$$

όπου $\mathcal{A}$ είναι ο έλκτης του Ε.Σ.Σ. Ισοδύναμα,

$$h(L, \mathcal{A}) \leq (1-s)^{-1} h\left(L, \bigcup_{m=1}^{M} w_m(L)\right)$$

για κάθε $L \in \mathcal{H}(X)$.

7. Μ.Ε.Π. ΚΑΙ ΕΥΡΕΣΗ ΤΩΝ ΠΑΡΑΓΟΝΤΩΝ ΚΑΤΑΚΟΡΥΦΗΣ ΚΛΙΜΑΚΩΣΗΣ

7.1 Μέθοδος δημιουργίας Μ.Ε.Π.

Οι Μ.Ε.Π. αποτελούν συνεχείς συναρτήσεις στον τριδιάστατο χώρο των οποίων το διάγραμμα είναι έλκτης καταλλήλως επιλεγμένου Ε.Σ.Σ. για σύνολα της μορφής $\{(x_i, y_j,\, z_{ij} = z(x_i, y_j) \in \mathbb{R}^3 \colon i = 0, 1, \dots, M; j = 0, 1, \dots, N\}$ και έχουν μορφοκλασματική διάσταση μεταξύ δύο και τρία.

Ένας τρόπος δημιουργίας Μ.Ε.Π. δημοσιεύθηκε στο [DM20] από τους Β. Δρακόπουλο και Π. Μανουσόπουλο. Η μεθοδολογία που προτείνουν αποτελεί μία βελτίωση της μεθοδολογίας της [XS11]. Επιπλέον, βασίζεται στο Κεφάλαιο 5, σελ. 123 – 128, 134-135, της [Μαν09]. Ακόμη, χρησιμοποιήθηκε στο [Dra+23] καθώς και στο Κεφάλαιο αυτό.

Έστωσαν $\Delta_{x,1}, \Delta_{x,2}$ διαμερισμοί του πραγματικού συμπαγούς διαστήματος $I_x = [a, b]$ τέτοιοι ώστε $\Delta_{x,1} = \{u_0, u_1, \dots, u_{M'}\}$, με $a = u_0 < u_1 < \cdots < u_{M'} = b$, και $\Delta_{x,2} = \{x_0, x_1, \dots, x_M\}$, με $a = x_0 < x_1 < \cdots < x_M = b$, με τον $\Delta_{x,1}$ να αποτελεί εκλέπτυνση του $\Delta_{x,2}$.

Παρομοίως, έστωσαν $\Delta_{y,1}, \Delta_{y,2}$ διαμερισμοί του πραγματικού συμπαγούς διαστήματος $I_y = [c, d]$ τέτοιοι ώστε $\Delta_{y,1} = \{\upsilon_0, \upsilon_1, \dots, \upsilon_{N'}\}$, με $c = \upsilon_0 < \upsilon_1 < \dots < \upsilon_{N'} = d$, και $\Delta_{y,2} = \{y_0, y_1, \cdots, y_N\}$, με $c = y_0 < y_1 < \dots < y_N = d$, με τον $\Delta_{y,1}$ να αποτελεί εκλέπτυνση του $\Delta_{y,2}$.

Η ανάλυση που ακολουθεί αναφέρεται στον πλήρη μετρικό χώρο $K = D \times \mathbb{R}$, με $D = I_x \times I_y$, ως προς την Ευκλείδεια ή άλλη ισοδύναμη μετρική. Το σύνολο των δεδομένων σημείων συμβολίζεται ως

$$P = \{(u_k, \upsilon_l, \hat{z}_{k,l} = \hat{z}(u_k, \upsilon_l)) \in K \colon k = 0, 1, \dots, M'; l = 0, 1, \dots, N'\}$$

ενώ το σύνολο των σημείων παρεμβολής ως

$$Q = \{(x_i, y_j, z_{i,j} = z(x_i, y_j)) \in K \colon i = 0, 1, \dots, M; j = 0, 1, \dots, N\}$$

όπου $Q \subset P$.

Επίσης τα $I_{x_m} = [x_{m-1}, x_m]$, $I_{y_n} = [y_{n-1}, y_n]$, $D_{m,n} = I_{x_m} \times I_{y_n}$ αντιστοιχούν στα οριζόμενα υποδιαστήματα παρεμβολής , ενώ τα $P_{m,n} = \{(u, v, \hat{z}) \in P : (u, v) \in D_{m,n}\}$ στα υποσύνολα των δεδομένων σημείων εντός της $m \times n$ περιοχής παρεμβολής, για κάθε $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$.

Έστωσαν συσταλτικοί ομοιομορφισμοί $L_{x_m} : I_x \to I_{x_m}$, $L_{y_n} : I_y \to I_{y_n}$, $L_{m,n} = (L_{x_m}, L_{y_n})$ για κάθε $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$, τέτοιοι ώστε

$$L_{x_m}(x_0) = x_{m-1}, \qquad L_{x_m}(x_M) = x_m, \qquad L_{y_n}(y_0) = y_{n-1}, \qquad L_{y_n}(y_N) = y_n \tag{7.1}$$

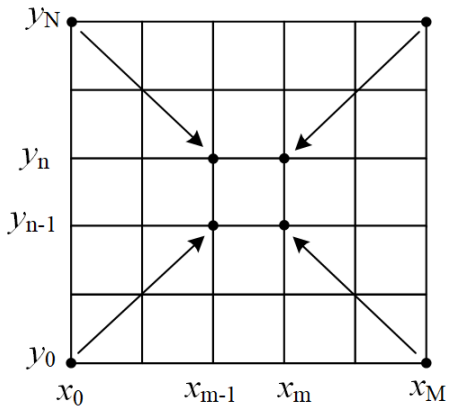
$$|L_{x_m}(x) - L_{x_m}(x')| \le k_1 |x - x'|, \qquad |L_{y_n}(y) - L_{y_n}(y')| \le k_2 |y - y'| \tag{7.2}$$

όπου $x, x' \in I_x$ και $y, y' \in I_y$ για κάποια $k_1, k_2 \in [0, 1)$.

Στο Σχήμα 7.1 απεικονίζεται ένα παράδειγμα εφαρμογής των μετασχηματισμών L_{x_m} και L_{y_n}. Το Ε.Σ.Σ. $\{D; L_{1-M,1-N}\}$ είναι υπεβολικό και έχει μοναδικό έλκτη

$$D = \bigcup_{m=1}^{M} \bigcup_{n=1}^{N} L_{m,n}(D) = \bigcup_{m=1}^{M} \bigcup_{n=1}^{N} D_{m,n}$$

όπου $\dot{D}_{m_1,n_1} \cap \dot{D}_{m_2,n_2} = \emptyset$, όταν $(m_1, n_1) \neq (m_2, n_2)$, για $m_1, m_2 = 1, 2, \dots, M$ και $n_1, n_2 = 1, 2, \dots, N$.



Σχήμα 7.1: Παράδειγμα εφαρμογής των μετασχηματισμών L_{x_m} και L_{y_n} στις τέσσερις κορυφές του πλέγματος

Επίσης, έστω συνεχείς απεικονίσεις $F_{m,n}\colon K \to \mathbb{R}$, με $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$ τέτοιες ώστε

$$\begin{aligned} F_{m,n}(x_0, y_0, z_{0,0}) &= z_{m-1,n-1} \quad & F_{m,n}(x_M, y_0, z_{M,0}) &= z_{m,n-1} \\ F_{m,n}(x_0, y_N, z_{0,N}) &= z_{m-1,n} & F_{m,n}(x_M, y_N, z_{M,N}) &= z_{m,n} \end{aligned} \tag{7.3}$$

και

$$|F_{m,n}(x, y, z) - F_{m,n}(x, y, z')| \leq k_3 |z - z'| \tag{7.4}$$

για κάθε $(x, y) \in D, z, z' \in \mathbb{R}$ και κάποιο $k_3 \in [0, 1)$. Η 7.4 υποδηλώνει ότι οι $F_{m,n}$ είναι συσταλτικές ως προς την τρίτη μεταβλητή, για $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$.

Ορίζονται συναρτήσεις $w_{m,n}\colon K \to K$ τέτοιες ώστε

$$w_{m,n}(x, y, z) = (L_{x_m}(x), L_{y_n}(y), F_{m,n}(x, y, z)) = (L_{m,n}(x, y), F_{m,n}(x, y, z)) \tag{7.5}$$

για κάθε $(x, y, z) \in K, m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$.

Το Ε.Σ.Σ. $\{K; w_{1-M,1-N}\}$ ενδεχομένως να μην είναι υπερβολικό. Απαραίτητη προϋπόθεση για την κατασκευή υπερβολικού Ε.Σ.Σ. με έλκτη διάγραμμα συνάρτησης, είναι οι απεικονίσεις $F_{m,n}, m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$ να ικανοποιούν, εκτός της συνθήκης 7.4, και την παρακάτω

$$|F_{m,n}(x, y, z) - F_{m,n}(x', y', z)| \leq \lambda_1 |x - x'| + \lambda_2 |y - y'| \tag{7.6}$$

για κάθε $x, x' \in I_x, y, y' \in I_y, z \in \mathbb{R}, m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$ και κάποια $\lambda_1, \lambda_2 > 0$. Η τελευταία συνθήκη υποδηλώνει ότι οι $F_{m,n}$ είναι ομοιομόρφως Lipschitz ως προς τις δύο πρώτες μεταβλητές, για $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$.

Θεώρημα 7.1.1. *Υπάρχει μετρική ρ_θ στο K, ισοδύναμη κατά Lipschitz προς την Ευκλείδεια μετρική, τέτοια ώστε το Ε.Σ.Σ. $\{K; w_{1-M,1-N}\}$ είναι υπερβολικό ως προς τη ρ_θ. Επιπλέον, το Ε.Σ.Σ. $\{K; w_{1-M,1-N}\}$ έχει μοναδικό έλκτη $G \in H(K)$.*

Απόδειξη. Βλ. [Μαν09], σελ. 125-126. □

Ορίζονται οι ομοιομορφισμοί $L_{x_m} : I_x \to I_{x_m}$ και $L_{y_n} : I_y \to I_{y_m}$, τέτοιοι ώστε

$$L_{x_m}(x) = a_m x + b_m,$$
$$L_{y_n}(y) = c_n y + d_n$$

όπου οι πραγματικοί αριθμοί a_m, b_m, c_n και d_n για $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$, επιλέγονται έτσι ώστε να ισχύει η συνθήκη 7.1, ενώ M, N θετικοί ακέραιοι μεγαλύτεροι της μονάδας. Οπότε ισχύει:

$$L_{x_m}(I_x) = I_{x_m},$$
$$L_{y_n}(I_y) = I_{y_n}$$

Επομένως, για $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$ με $M, N > 1$, ισχύει

$$a_m = \frac{x_m - x_{m-1}}{x_M - x_0}, \qquad b_m = \frac{x_M x_{m-1} - x_0 x_m}{x_M - x_0}$$

$$c_n = \frac{y_n - y_{n-1}}{y_N - y_0}, \qquad d_n = \frac{y_N y_{n-1} - y_0 y_n}{y_N - y_0}$$

Εφ' όσον $M, N \geq 2, |a_m| < 1$ και $|c_n| < 1$, οι L_{x_m} και L_{y_n} πληρούν τη συνθήκη 7.2 με $k_1 = \max\{|a_m| : m = 1, 2, \dots, M\}$ και $k_2 = \max\{|c_n| : n = 1, 2, \dots, N\}$, οπότε αποτελούν συσταλτικούς ομοιομορφισμούς.

Οι μετασχηματισμοί αυτοί μπορούν να γραφούν και ως

$$L_{x_m}(x) \equiv x' = \frac{x - x_0}{x_M - x_0} x_m + \frac{x_M - x}{x_M - x_0} x_{m-1}$$

$$L_{y_n}(y) \equiv y' = \frac{y - y_0}{y_N - y_0} y_n + \frac{y_N - y}{y_N - y_0} y_{n-1}$$

ή ως

$$\begin{aligned} L_{x_m}(x) \equiv x' &= \frac{1}{\Delta_x}[(x - x_0)x_m + (x_M - x)x_{m-1}] \\ L_{y_n}(y) \equiv y' &= \frac{1}{\Delta_y}[(y - y_0)y_n + (y_N - y)y_{n-1}] \end{aligned} \tag{7.7}$$

για $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$, όπου Δ_x η διαφορά μεταξύ x_M και x_0, και Δ_y η διαφορά μεταξύ y_N και y_0.

Ορίζεται η $F_{m,n} : K \to \mathbb{R}$ ως

$$F_{m,n}(x, y, z) = s_{m,n} z + h_{m,n}(x, y)$$

όπου

$$h_{m,n}(x, y) = e_{m,n} x + f_{m,n} y + g_{m,n} xy + k_{m,n}$$

για $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$. Οι πραγματικές σταθερές $e_{m,n}$, $f_{m,n}$, $g_{m,n}$ και $k_{m,n}$ εξαρτώνται από την ρυθμιζόμενη παράμετρο $s_{m,n}$, και επιλέγονται έτσι ώστε να ισχύει η συνθήκη (7.3).

Δηλαδή επιλέγονται τα $s_{m,n} \in (-1, 1)$ και κατόπιν υπολογίζονται οι πραγματικές σταθερές $e_{m,n}, f_{m,n}, g_{m,n}$ και $k_{m,n}$ από τις σχέσεις

$$g_{m,n} = \frac{z_{m,n} + z_{m-1,n-1} - z_{m-1,n} - z_{m,n-1} - s_{m,n}(z_{0,0} + z_{M,N} - z_{0,N} - z_{M,0})}{\Delta_x \Delta_y},$$

$$e_{m,n} = \frac{z_{m,n-1} - z_{m-1,n-1} + s_{m,n}(z_{0,0} - z_{M,0})) - g_{m,n}\Delta_x y_0}{\Delta_x},$$

$$f_{m,n} = \frac{z_{m-1,n} - z_{m-1,n-1} + s_{m,n}(z_{0,0} - z_{0,N}) - g_{m,n} x_0 \Delta_y}{\Delta_y},$$

$$k_{m,n} = z_{m,n} - e_{m,n} x_M - f_{m,n} y_N - s_{m,n} z_{M,N} - g_{m,n} x_M y_N,$$

για $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$.

Οι απεικονίσεις $F_{m,n}, m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$, πληρούν τη συνθήκη 7.4 με $k_3 = \max\{|s_{m,n}| : m = 1, 2, \dots, M \text{ και } n = 1, 2, \dots, N\}$, καθώς επίσης και τη συνθήκη 7.6 με $\lambda_1 = \max\{|e_{m,n}| + |g_{m,n} y_N| : m = 1, 2, \dots, M \text{ και } n = 1, 2, \dots, N\}$ και $\lambda_2 = \max\{|f_{m,n}| + |g_{m,n} x_M| : m = 1, 2, \dots, M \text{ και } n = 1, 2, \dots, N\}$.

Ορίζεται το Ε.Σ.Σ. της μορφής $\{K; w_{1-M,1-N}\}$, του οποίου οι απεικονίσεις έχουν τη μορφή

$$w_{m,n} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_m & 0 & 0 \\ 0 & c_n & 0 \\ e_{m,n} & f_{m,n} & s_{m,n} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} b_m \\ d_n \\ g_{m,n} xy + k_{m,n} \end{pmatrix} \tag{7.8}$$

όπου $a_m, b_m, c_n, d_n, e_{m,n}, f_{m,n}, g_{m,n}, k_{m,n}$ είναι πραγματικοί αριθμοί για $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$. Οι μετασχηματισμοί $w_{m,n}$ είναι διμεταβλητοί και έχουν παράγοντες κατακόρυφης κλιμάκωσης τους $s_{m,n}$, ενώ περιοριζόμενοι από τις συνθήκες 7.1 και 7.3 έχουν ως αποτέλεσμα

$$w_{m,n}\begin{pmatrix} x_0 \\ y_0 \\ z_{0,0} \end{pmatrix} = \begin{pmatrix} x_{m-1} \\ y_{n-1} \\ z_{m-1,n-1} \end{pmatrix}, \quad w_{m,n}\begin{pmatrix} x_M \\ y_0 \\ z_{M,0} \end{pmatrix} = \begin{pmatrix} x_m \\ y_{n-1} \\ z_{m,n-1} \end{pmatrix}$$

$$w_{m,n}\begin{pmatrix} x_0 \\ y_N \\ z_{0,N} \end{pmatrix} = \begin{pmatrix} x_{m-1} \\ y_n \\ z_{m-1,n} \end{pmatrix}, \quad w_{m,n}\begin{pmatrix} x_M \\ y_N \\ z_{M,N} \end{pmatrix} = \begin{pmatrix} x_m \\ y_n \\ z_{m,n} \end{pmatrix}$$

για $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$.

Οπότε ισχύει:

$$w_{m,n}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

όπου τα x'και y' δίνονται από την 7.7 και

$$\begin{aligned} z' = {} & \frac{(x'-x_{m-1})(y'-y_{n-1})}{(x_m-x_{m-1})(y_n-y_{n-1})} z_{m,n} + \frac{(x'-x_m)(y'-y_{n-1})}{(x_{m-1}-x_m)(y_n-y_{n-1})} z_{m-1,n} \\ & + \frac{(x'-x_{m-1})(y'-y_n)}{(x_m-x_{m-1})(y_{n-1}-y_n)} z_{m,n-1} + \frac{(x'-x_m)(y'-y_n)}{(x_{m-1}-x_m)(y_{n-1}-y_n)} z_{m-1,n-1} \\ & + s_{m,n}[z - \frac{(x'-x_{m-1})(y'-y_{n-1})}{(x_m-x_{m-1})(y_n-y_{n-1})} z_{M,N} - \frac{(x'-x_m)(y'-y_{n-1})}{(x_{m-1}-x_m)(y_n-y_{n-1})} z_{0,N} \\ & - \frac{(x'-x_{m-1})(y'-y_n)}{(x_m-x_{m-1})(y_{n-1}-y_n)} z_{M,0} - \frac{(x'-x_m)(y'-y_n)}{(x_{m-1}-x_m)(y_{n-1}-y_n)} z_{0,0} \end{aligned} \tag{7.9}$$

Στην περίπτωση που $s_{m,n} = 0$ καταλήγουμε στην εξίσωση 8 της [KP05].

7.3 Μεθοδολογίες υπολογισμού των παραγόντων κατακόρυφης κλιμάκωσης

Οι παράγοντες κατακόρυφης κλιμάκωσης $s_{m,n}$, για $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$, των προτεινόμενων διμεταβλητών Μ.Ε.Π. είναι ελεύθερες παράμετροι των μετασχηματισμών περιοριζόμενες στο διάστημα $(-1, 1)$. Ο υπολογισμός τους μπορεί να πραγματοποιηθεί στοχεύοντας στην ελαχιστοποίηση κάποιου μέτρου σφάλματος μεταξύ των αρχικών και των ανακατασκευασμένων σημείων.

Στην [BDD06] παρουσιάζεται μία μεθοδολογία όπου ο υπολογισμός των παραγόντων πραγματοποιείται με

βάση το λόγο των μηκών καταλλήλως επιλεγμένων ευθυγράμμων τμημάτων. Συγκεκριμένα, χρησιμοποιείται το μέσο μήκος ευθυγράμμων τμημάτων παραλλήλων προς τον άξονα των z τα οποία ενώνουν σημεία της επιφάνειας με ευθύγραμμα τμήματα οριζόμενα από αντίστοιχα ακραία σημεία του πλέγματος.

Στις [SX04] και [Xie+01] προτείνεται ένας παρόμοιος υπολογισμός των παραγόντων. Συγκεκριμένα, τα δεδομένα σημεία προσεγγίζονται από μία «επιφάνεια τάσης» πρώτου βαθμού της μορφής $\mathcal{Z}(x,y) = b_0 + b_1 x + b_2 y$. Ο υπολογισμός των συντελεστών b_0, b_1 και b_2 γίνεται με την μέθοδο των ελαχίστων τετραγώνων ως προς τις κατηγμένες. Κατόπιν, κάθε παράγοντας υπολογίζεται ως $s_{m,n} = e_{m,n}/e$, για κάθε $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$, όπου $e_{m,n} = z_{mn} - Z(x_m, y_n)$ και $e = \max_{m\in\{0,1,\dots,M\}, n\in\{0,1,\dots,N\}} |e_{m,n}|$.

Στην [Μαν09] προτείνεται η επέκταση των μεθόδων περιβαλλόντων όγκων στις διμεταβλητές Μ.Ε.Π. Πιο συγκεκριμένα, έστω $B \in K_0^3$ κυρτός περιβάλλων όγκος του P και $B_{m,n} \in K_0^3$ κυρτοί περιβάλλοντες όγκοι των $P_{m,n}$, για κάθε $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$. Στόχος είναι η ελαχιστοποίηση του όγκου της συμμετρικής διαφοράς $B_{m,n} \triangle w_{m,n}(B)$, μέσω της επιλογής των καταλλήλων $s_{m,n}$, για κάθε $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$. Ως περιβάλλοντες όγκοι είναι δυνατόν να χρησιμοποιηθούν κυρτά περιβλήματα στον $\mathbb{R}^3$ ή περιβάλλοντα παραλληλεπίπεδα. Ο υπολογισμός των βέλτιστων παραγόντων δύναται να γίνει με αλγοριθμικό τρόπο, όπως στην Μέθοδο Κυρτών Περιβλημάτων (Μ.Κ.Π.) της Παραγράφου 3.1.5 της [Μαν09].

7.4 Μέθοδος Κυρτών Περιβλημάτων

Η Μέθοδος Κυρτών Περιβλημάτων (Μ.Κ.Π.) βελτιώθηκε, υλοποιήθηκε και παρουσιάστηκε το 2023 από τους Δρακόπουλο Β., Ματθέ Δ., Σγούρδο Δ. και Vijender N [Dra+23]. Με βάση τον ορισμό των Μ.Ε.Π., ο υπολογισμός των παραγόντων $s_{m,n}$ μπορεί να πραγματοποιηθεί έχοντας σαν στόχο την ελαχιστοποίηση της απόστασης Hausdorff ανάμεσα στα αρχικά και τα ανακατασκευασμένα δεδομένα σημεία.

Κάθε κηδεστικός μετασχηματισμός $w_{m,n}$ μετασχηματίζει τον χώρο $[x_0, x_M] \times [y_0, y_N] \times \mathbb{R}(\supset P)$ στον αντίστοιχο $[x_{m-1}, x_m] \times [y_{n-1}, y_n] \times \mathbb{R}(\supset P_{m,n})$, ενώ ισχύει:

$$h(P, \bigcup_{m=1}^{M}\bigcup_{n=1}^{N} w_{m,n}(P)) = h(\bigcup_{m=1}^{M}\bigcup_{n=1}^{N} P_{m,n}, \bigcup_{m=1}^{M}\bigcup_{n=1}^{N} w_{m,n}(P)) \leq \max_{1\leq m\leq M, 1\leq n\leq N}\{h(P_{m,n}, w_{m,n}(P))\}$$

Παρ' όλα αυτά, ο άμεσος υπολογισμός του αντίστοιχου βέλτιστου παράγοντα $s_{m,n}$, ο οποίος ελαχιστοποιεί την απόσταση $h(P_{m,n}, w_{m,n}(P))$ για κάθε $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$, δεν είναι εφικτός. Είναι δυνατόν όμως να πραγματοποιηθεί με τη χρήση περιβαλλόντων όγκων των P και $P_{m,n}$, έτσι ώστε το μετασχηματισμένο

σύνολο $w_{m,n}(P)$ να αποτελεί μία όσο το δυνατόν καλύτερη προσέγγιση του αρχικού συνόλου $P_{m,n}$.

Έστω $B \in K_0^3$ κυρτός περιβάλλων όγκος του P και $B_{m,n} \in K_0^3$ κυρτοί περιβάλλοντες όγκοι των $P_{m,n}$, για κάθε $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$. Δηλαδή, ισχυει $P \subset B$ και $P_{m,n} \subset B_{m,n}$ για κάθε $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$. Όπως έχει προαναφερθεί, η επιλογή του $s_{m,n}$ που ελαχιστοποιεί την απόσταση $h(B_{m,n}, w_{m,n}(B))$ είναι ανέφικτη. Όταν οι όγκοι B και $B_{m,n}$ αποτελούν τα κυρτά περιβλήματα των P και $P_{m,n}$ αντιστοίχως, ισχύει $h(B_{m,n}, w_{m,n}(B)) \leq h(P_{m,n}, w_{m,n}(P))$ (βλ. [Web94]), οπότε η ελαχιστοποίηση δεν παρέχει κάποιο άνω φράγμα της απόστασης $h(P_{m,n}, w_{m,n}(P))$.

Με τη χρήση όμως της μετρικής της συμμετρικής διαφοράς οδηγούμαστε στην ελαχιστοποίηση του όγκου της συμμετρικής διαφοράς $B_{m,n} \triangle w_{m,n}(B)$, για κάθε $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$. Ως αποτέλεσμα, με την επιλογή τιμών για τους παράγοντες $s_{m,n}$ τέτοιων ώστε να προκύπτει μέγιστη επικάλυψη των αντιστοίχων περιβαλλόντων όγκων, είναι δυνατόν να επιτευχθεί καλύτερη αναπαράσταση των δεδομένων σημείων. Η προτεινόμενη μεθοδολογία περιορίζεται σε κυρτούς περιβάλλοντες όγκους, αφού οι μη κυρτοί είναι αβέβαιο ότι μπορούν να συνδυαστούν με αποδοτικούς αλγορίθμους ή να βελτιώσουν τα αποτελέσματα.

Στόχος είναι η ελαχιστοποίηση του όγκου των μη επικαλυπτόμενων τμημάτων του κυρτού περιβλήματος του συνόλου $P_{m,n}$ και του μετασχηματισμένου από τον $w_{m,n}$ κυρτού περιβλήματος του συνόλου P για κάθε $m = 1, 2, \dots, M$ και $n = 1, 2, \dots, N$. Ισχύει:

$$\begin{aligned} &\delta_S(CH(P_{m,n}), w_{m,n}(CH(P))) = \\ &= Volume\{CH(P_{m,n})\} + Volume\{w_{m,n}(CH(P))\} - 2 \cdot Volume\{CH(P_{m,n}) \cap w_{m,n}(CH(P))\} \\ &= Volume\{CH(P_{m,n})\} + Volume\{w_{m,n}(CH(P))\} - 2 \cdot Volume\{CH(P_{m,n} \cap w_{m,n}(P))\} \end{aligned} \tag{7.10}$$

όπου $CH(\cdot)$ συμβολίζει το κυρτό περίβλημα ενός συνόλου σημείων. Πρέπει να σημειωθεί ότι ισχύει

$$w_{m,n}(CH(\cdot)) = CH(w_{m,n}(\cdot))$$

καθώς ο μετασχηματισμός $w_{m,n}$ είναι κηδεστικός. Όπως προκύπτει από τη σχέση 7.10, ο υπολογισμός του δ_S είναι αλγοριθμικός και περιλαμβάνει τον υπολογισμό κυρτών περιβλημάτων, τομών πολυέδρων καθώς επίσης και όγκων πολυέδρων. Για τον λόγο αυτό, είναι απαραίτητη η χρησιμοποίηση κάποιας μεθόδου μονοδιάστατης βελτιστοποίησης χωρίς παραγώγους, όπως είναι η μέθοδος του Brent (βλ. [Pre+92]), η οποία αποτελεί μία μέθοδο εγκιβωτισμού με παραβολική παρεμβολή.

Αρχικά, για την σύγκλιση της μεθόδου στο βέλτιστο $\hat{s}^{+}_{m,n}$, πρέπει να εγκιβωτιστεί, πρέπει δηλαδή να βρεθούν τρεις σταθερές $s^{a}_{m,n}, s^{b}_{m,n}, s^{c}_{m,n}$, τέτοιες ώστε

$$0 \leq s^{a}_{m,n} < s^{b}_{m,n} < s^{c}_{m,n} \leq 1$$
$$Vol(s^{b}_{m,n}) < Vol(s^{a}_{m,n}), Vol(s^{b}_{m,n}) < Vol(s^{c}_{m,n})$$
$$s^{a}_{m,n} < \hat{s}^{+}_{m,n} < s^{c}_{m,n}$$

Μία καλή προσέγγιση αποτελεί ο υπολογισμός του $s^{b}_{m,n}$ έτσι ώστε

$$Vol\{w_{m,n}(CH(P))\} \approx Vol\{CH(P_{m,n})\}$$

Στη συνέχεια τα $s^{a}_{m,n}, s^{c}_{m,n}$ προκύπτουν από το $s^{b}_{m,n}$, μετατρέποντάς το κατά ένα συντελεστή. Δηλαδή:

$$s^{b}_{m,n} = Vol\{CH(P_{m,n}\}/(a_{m,n} \cdot c_{m,n} \cdot Vol\{CH(P)\})$$
$$s^{a}_{m,n} = s^{b}_{m,n}/c$$
$$s^{c}_{m,n} = \min\{cs^{b}_{m,n}, 1\}$$

για κάποιο $c > 1$. Η σταθερά c είναι δυνατόν είτε να έχει προκαθορισμένη τιμή (για παράδειγμα $c = 2$), είτε ασφαλέστερα να ορίζεται επαναληπτικά με διαδοχικά αυξανόμενες τιμές μέχρι να βρεθεί κατάλληλη τριάδα εγκιβωτισμού.

Στην περίπτωση αρνητικών παραγόντων, ο βέλτιστος $\hat{s}^{-}_{m,n}$ υπολογίζεται με παρόμοιο τρόπο, με χρήση της παρακάτω τριάδας εγκιβωτισμού

$$s^{b'}_{m,n} = -s^{b}_{m,n}$$
$$s^{a'}_{m,n} = \max\{cs^{b'}_{m,n}, -1\}$$
$$s^{c'}_{m,n} = s^{b'}_{m,n}/c$$

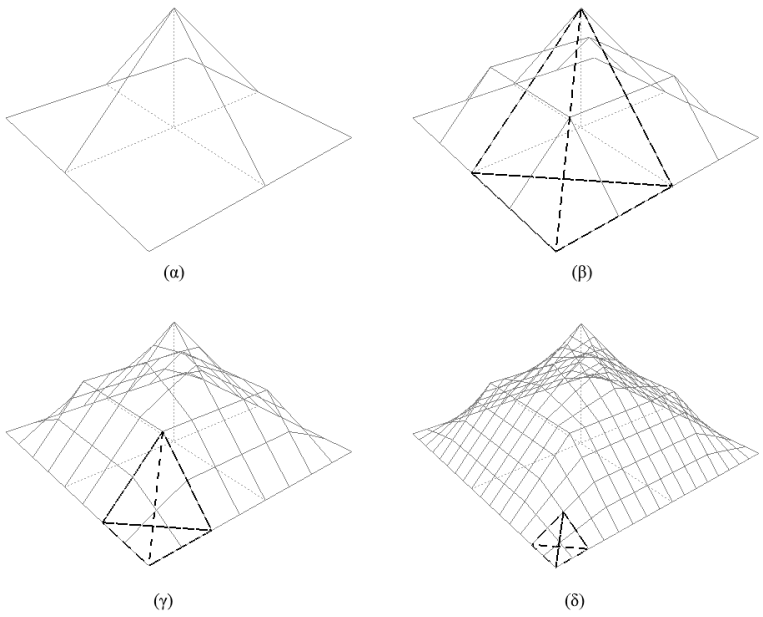
Ακολουθεί ο αλγόριθμος υπολογισμού βέλτιστων παραγόντων $s_{m,n}$ με τη μορφή αναλυτικών βημάτων ο οποίος υλοποιεί τον υπολογισμό του $\hat{s}_{m,n} \in (-1, 1)$ που ελαχιστοποιεί τον όγκο

$$Vol(s_{m,n}) \equiv \delta^{S}(CH(P_{m,n}), w_{m,n}(CH(P)))$$

για κάθε $m = 1, 2, \ldots, M$ και $n = 1, 2, \ldots, N$:

Αλγόριθμος υπολογισμού βέλτιστων παραγόντων $s_{m,n}$

1. Υπολόγισε το κυρτό περίβλημα του συνόλου των δεδομένων σημείων P
2. Για κάθε επίπεδο παρεμβολής $I_{m,n}$, $m = 1, 2, \ldots, M$ και $n = 1, 2, \ldots, N$
   2.1. Υπολόγισε τις γνωστές παραμέτρους του μετασχηματισμού $w_{m,n}$, δηλαδή τις $a_{m,n}$, $b_{m,n}$, $c_{m,n}$, $d_{m,n}$, $e_{m,n}$, $f_{m,n}$, $g_{m,n}$, $k_{m,n}$
   2.2. Υπολόγισε το κυρτό περίβλημα του συνόλου $P_{m,n}$ (βλ. Σχ. 7.2)
   2.3. Υπολόγισε τον βέλτιστο $\hat{s}^{+}_{m,n} \in [0, 1)$ που ελαχιστοποιεί τον όγκο των μη επικαλυπτόμενων τμημάτων των όγκων $CH(P_{m,n})$ και $w_{m,n}(CH(P))$, χρησιμοποιώντας την τριάδα εγκιβωτισμού $(s^{b}_{m,n}/c,\ s^{b}_{m,n},\ \min\{cs^{b}_{m,n}, 1\})$, όπου $s^{b}_{m,n} = Volume\{CH(P_{m,n})\}/(a_{m,n} \cdot c_{m,n} \cdot Volume\{CH(P)\})$ και $c > 1$
   2.4. Υπολόγισε τον βέλτιστο $\hat{s}^{-}_{m,n} \in (-1, 0]$ που ελαχιστοποιεί τον όγκο των μη επικαλυπτόμενων τμημάτων των όγκων $CH(P_{m,n})$ και $w_{m,n}(CH(P))$, χρησιμοποιώντας την τριάδα εγκιβωτισμού $(\max\{cs^{b'}_{m,n}, -1\},\ s^{b'}_{m,n},\ s^{b'}_{m,n}/c)$, όπου $s^{b'}_{m,n} = -s_b$ και $c > 1$.
   2.5. Εάν $Vol(\hat{s}^{+}_{m,n}) \leq Vol(\hat{s}^{-}_{m,n})$ τότε $\hat{s}_{m,n} = \hat{s}^{+}_{m,n}$ αλλιώς $\hat{s}_{m,n} = \hat{s}^{-}_{m,n}$



Σχήμα 7.2: (α) Αρχικά σημεία που προσδιορίζουν την επιφάνεια και υπολογισμός του όγκου του κυρτού περιβλήματος του συνόλου $P_{1,1}$ πριν από: (β) μία επανάληψη, (γ) δύο επαναλήψεις, (δ) τρεις επαναλήψεις

Ο υπολογισμός της τομής δύο κυρτών πολυέδρων, τα οποία έχουν m και n κορυφές αντίστοιχα, απαιτεί $O(n+m)$ χρόνο. Ο υπολογισμός του όγκου ενός κυρτού πολυέδρου με n κορυφές απαιτεί $O(n)$ χρόνο, ενώ ο υπολογισμός του κυρτού περιβλήματος n σημείων του χώρου χρησιμοποιώντας τον Αλγόριθμο Ταχέως Περιβλήματος απαιτεί γενικά στην χειρίστη περίπτωση $O(n \log r)$ χρόνο, όπου r το πλήθος των επεξεργασμένων σημείων. Ο αλγόριθμος βελτιστοποίησης του Brent απαιτεί σταθερό πλήθος βημάτων, το οποίο εξαρτάται μόνο από την απαιτούμενη ακρίβεια και όχι από το πλήθος των δεδομένων σημείων. Επομένως, ο υπολογισμός ενός παράγοντα $s_{m,n}$, ο οποίος εξαρτάται από το πλήθος των σημείων εντός του αντίστοιχου επιπέδου παρεμβολής, απαιτεί χρόνο στη χείριστη περίπτωση $O(Klog(K)/M \cdot N)$. Εφ' όσον υπάρχουν συνολικά $M \cdot N$ παράγοντες $s_{m,n}$, στην χειρότερη περίπτωση ο αλγόριθμος έχει πολυπλοκότητα $O(KlogK)$ ως προς το πλήθος των δεδομένων σημείων. Επιπρόσθετα, τα κυρτά περιβλήματα συνήθως έχουν λίγες κορυφές πρακτικά καθώς ένα επίπεδο παρεμβολής περιέχει σχετικά περιορισμένο πλήθος σημείων με αποτέλεσμα οι υπολογισμοί επί των πολυέδρων να είναι ταχείς.

7.5 Απαιτήσεις της Μεθόδου Κυρτών Περιβλημάτων

Βάσει των προηγουμένων, κατά την υλοποίηση της Μεθόδου Κυρτών Περιβλημάτων απαιτούνται:

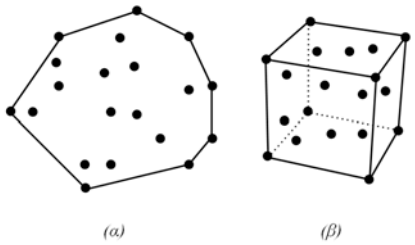
- Ο υπολογισμός του κυρτού περιβλήματος ενός συνόλου σημείων.
- Ο υπολογισμός του όγκου ενός κυρτού περιβλήματος.
- Ο υπολογισμός της τομής δύο κυρτών περιβλημάτων.

Ο υπολογισμός του κυρτού περιβλήματος ενός συνόλου σημείων μπορεί να υλοποιηθεί αποτελεσματικά με τον Αλγόριθμο Ταχέως Περιβλήματος. Στα επόμενα κεφάλαια προτείνονται μεθοδολογίες οι οποίες υπολογίζουν αποδοτικά:

1. Την εύρεση του όγκου ενός κυρτού πολυέδρου δια μέσω διαμέρισής του σε τετράεδρα.
2. Την τομή δύο κυρτών πολυέδρων υπό την προϋπόθεση ότι το ένα εκ των δύο είναι τριγωνισμένο.

8. ΚΥΡΤΟ ΠΕΡΙΒΛΗΜΑ

Κυρτό περίβλημα ενός συνόλου S που αποτελείται από n σημεία ονομάζεται το ελάχιστο κυρτό πολύγωνο στις δύο διαστάσεις ή το ελάχιστο κυρτό πολύεδρο στις τρεις διαστάσεις που μπορεί να περικλείει τα σημεία αυτά. Το κυρτό περίβλημα έχει σημαντικές εφαρμογές σε διάφορους τομείς όπως τα γραφικά υπολογιστών, η υπολογιστική γεωμετρία, τα Συστήματα Γεωγραφικών Πληροφοριών, η ρομποτική και άλλα. Αποτελεί θεμελιώδη έννοια για την επίλυση πολλών γεωμετρικών προβλημάτων και χρησιμοποιείται ευρέως σε αλγόριθμους όπως η ανάλυση ενός σχήματος ή η αναγνώριση προτύπων. Μάλιστα, ο Aurenhammer [Aur91] περιγράφει εφαρμογές των κυρτών περιβλημάτων στη δημιουργία πλεγμάτων, στην αναζήτηση αρχείων, στην κατηγοριοποίηση δεδομένων, στην ανίχνευση συγκρούσεων στα ηλεκτρονικά παίγνια, στην κρυσταλλογραφία, στη μεταλλουργία, στον πολεοδομικό σχεδιασμό, στην χαρτογραφία, στην επεξεργασία εικόνας, στην αριθμητική ολοκλήρωση, στη στατιστική, στη διευθέτηση σφαιρικών μη επικαλυπτόμενων όγκων καθώς επίσης και στον εντοπισμό της τοποθεσίας σημείου. Το κυρτό περίβλημα ενός συνόλου σημείων στις δύο διαστάσεις μπορεί να περιγραφεί και ως εξής: εάν θεωρήσουμε ότι τα σημεία είναι καρφιά καρφωμένα σε ένα επίπεδο κομμάτι ξύλου, τότε το κυρτό περίβλημά τους είναι το πολύγωνο που σχηματίζεται, όταν τεντώσουμε έναν ελαστικό ιμάντα γύρω από τα καρφιά αυτά. Αντίστοιχα το κυρτό περίβλημα ενός συνόλου σημείων στις τρεις διαστάσεις είναι το πολύεδρο που σχηματίζεται αν υποθέσουμε ότι περιτυλίγουμε όλα τα σημεία με μία καλά τεντωμένη ελαστική μεμβράνη. (βλ. Σχ. 8.1).



Σχήμα 8.1: Κυρτό περίβλημα: (α) Στις δύο διαστάσεις, (β) στις τρεις διαστάσεις

Ένα κυρτό περίβλημα περιγράφεται από ένα σύνολο όψεων και από ένα σύνολο λιστών που προσδιορίζει τις γειτονικές όψεις και τις κορυφές για κάθε όψη. Τα συνοριακά σημεία κάθε όψης ονομάζονται κορυφογραμμές.

Κάθε κορυφογραμμή σηματοδοτεί την τομή δύο όψεων. Στον $\mathbb{R}^3$, οι όψεις είναι τρίγωνα και οι κορυφογραμμές είναι ακμές.

8.1 Μαθηματικός ορισμός και ιδιότητες

Από μαθηματικής σκοπιάς, για ένα σύνολο σημείων $S = \{p_1, p_2, ..., p_n\} \subseteq \mathbb{R}^d, d \geq 2$, το κυρτό περίβλημά τους ορίζεται ως $CH(S) = \{\sum_{i=1}^{n} a_i \cdot p_i \mid a_i \geq 0$ για κάθε $i = 0, 1, ..., n,$ και $\sum_{i=1}^{n} a_i = 1\}$. Αυτό σημαίνει ότι οποιοδήποτε σημείο του κυρτού περιβλήματος μπορεί να εκφραστεί ως ένας κυρτός συνδυασμός των σημείων του S. Ένας κυρτός συνδυασμός είναι ένας γραμμικός συνδυασμός σημείων όπου όλοι οι συντελεστές είναι μη αρνητικοί και έχουν άθροισμα 1.

Οι ιδιότητες που διέπουν ένα κυρτό περίβλημα είναι:

- **Μοναδικότητα:** Το κυρτό περίβλημα ενός συνόλου σημείων είναι μοναδικό.
- **Ελαχιστοποίηση:** Το κυρτό περίβλημα είναι το ελάχιστο κυρτό σύνολο που περιέχει όλα τα σημεία.
- **Κυρτότητα:** Το κυρτό περίβλημα είναι ένα κυρτό σύνολο.
- **Αναπαράσταση ορίων:** Το σύνορο του κυρτού περιβλήματος σχηματίζεται από τις κορυφές που ορίζουν την περίμετρό του.

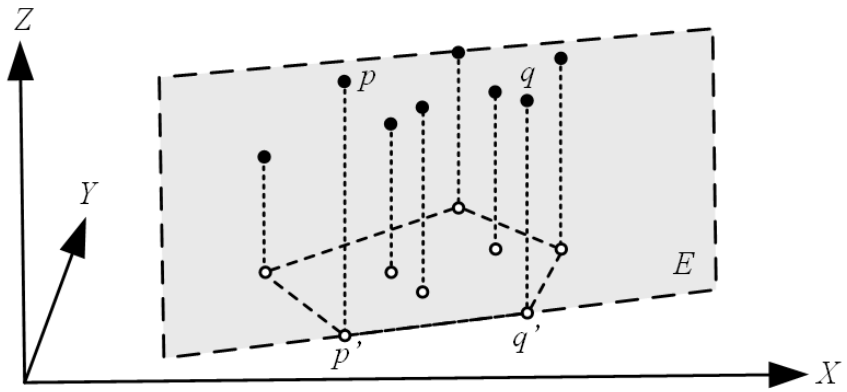
8.2 Αλγόριθμοι δημιουργίας κυρτών περιβλημάτων στις τρεις διαστάσεις

Για την εύρεση του βέλτιστου ελεύθερου συντελεστή s κατά τη δημιουργία Μορφοκλασματικών Επιφανειών Παρεμβολής, απαιτείται ο υπολογισμός του κυρτού περιβλήματος των σημείων παρεμβολής. Για τον υπολογισμό τριδιάστατων κυρτών περιβλημάτων έχουν επινοηθεί διάφοροι αλγόριθμοι. Ακολουθούν μερικοί από πιο σημαντικούς:

8.2.1 Αλγόριθμος Περιτυλίγματος

Ο Αλγόριθμος Περιτυλίγματος επινοήθηκε για κυρτά περιβλήματα σε οποιαδήποτε διάσταση [CK70]. Η τριδιάστατη έκδοσή του είναι μία άμεση γενίκευση του διδιάστατου αλγορίθμου η οποία αντί να χρησιμοποιήσει μία γραμμή την οποία περιτυλίγει γύρω από ένα σύνολο σημείων, χρησιμοποιεί ένα επίπεδο. Στο πρώτο βήμα του ο αλγόριθμος προβάλει το σύνολο των τριδιάστατων σημείων στις δύο διαστάσεις του προκειμένου να βρει μία ακμή εκκίνησης.

Μία ακμή του τριδιάστατου κυρτού περιβλήματος είναι ένα ευθύγραμμο τμήμα που συνδέει δύο σημεία του συνόλου αν και μόνο αν υπάρχει ένα επίπεδο που περνά από αυτά τα σημεία αφήνοντας όλα τα υπόλοιπα από τη μία μόνο πλευρά του. Με βάση αυτήν την ιδιότητα, μία ακμή εκκίνησης του τριδιάστατου Αλγορίθμου Περιτυλίγματος προσδιορίζεται ως ακολούθως: προβάλλουμε το δοσμένο σύνολο σημείων σε κάποιο επίπεδο, έστω το επίπεδο X-Y, και αν η ακμή αυτή αφήνει όλα τα υπόλοιπα σημεία από τη μία μόνο πλευρά της, τότε θεωρείται ότι είναι ακμή του κυρτού περιβλήματος του τριδιάστατου συνόλου και κατά συνέπεια η ακμή εκκίνησης. Όταν εντοπιστεί η ακμή εκκίνησης, ο τριδιάστατος Αλγόριθμος Περιτυλίγματος μπορεί πλέον να ξεκινήσει στρέφοντας το κάθετο επίπεδο στο οποίο αυτή βρίσκεται έως ότου το επίπεδο έλθει σε επαφή με κάποιο σημείο του γενικότερου συνόλου προσδιορίζοντας έτσι την πρώτη έδρα του κυρτού περιβλήματος (βλ. Σχ. 8.2).



Σχήμα 8.2: Προβολή των σημείων του συνόλου στο επίπεδο X-Y, εύρεση της ακμής εκκίνησης p-q και περιστροφή του επιπέδου E

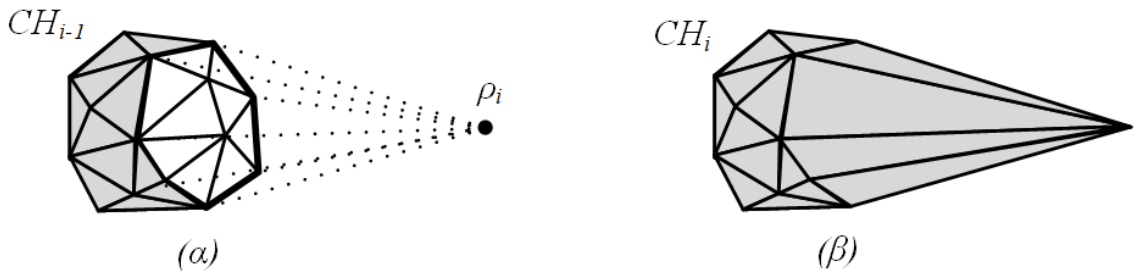
Η εύρεση της ακμής εκκίνησης απαιτεί χρόνο $O(n)$. Για κάθε σημείο ή ακμή του περιβλήματος, η εύρεση νέων σημείων απαιτεί χρόνο $O(n)$. Έτσι, ο συνολικός χρόνος εκτέλεσης του αλγορίθμου είναι ίδιος με την έκδοσή του στις δύο διαστάσεις:

$$\sum_{i=1}^{h} O(n) = O(nh)$$

8.2.2 Αυξητικός αλγόριθμος

Ο Αυξητικός αλγόριθμος, ο οποίος συχνά αναφέρεται και ως μέθοδος Beneath-Beyond, θεωρεί ότι στην i-οστή επανάληψη το τρέχον κυρτό περίβλημα CH_i είναι το κυρτό περίβλημα του CH_{i-1} και του τρέχοντος σημείου ρ_i. Υπάρχουν δύο ενδεχόμενα για τον υπολογισμό του CH_i: Αν το ρ_i ανήκει στο CH_{i-1}, τότε αγνοείται το ρ_i και ισχύει $CH_i = CH_{i-1}$. Εάν δεν συμβαίνει κάτι τέτοιο, τότε υπολογίζεται ο «κώνος» που έχει κορυφή το ρ_i και

εφάπτεται στο CH_{i-1} και με βάσει αυτή την γεωμετρία κατασκευάζεται το νέο περίβλημα (βλ. Σχ. 8.3).



Σχήμα 8.3: (α) Το κυρτό περίβλημα CH_{i-1} και το τρέχον σημείο ρ_i, (β) Το κυρτό περίβλημα CH_i

Ο έλεγχος, εάν το ρ_i είναι μέσα στο CH_{i-1}, γίνεται τότε και μόνο, αν το ρ_i βρίσκεται από την αριστερή πλευρά του επιπέδου κάθε έδρας του CH_{i-1}. Η κάθε έδρα θα πρέπει να έχει τον κατάλληλο προσανατολισμό, δηλαδή η σειρά των κορυφών κάθε έδρας να ακολουθεί αντίθετη φορά από τους δείκτες του ωρολογίου. Ο έλεγχος για το αν ένα σημείο είναι από την αριστερή πλευρά ενός επιπέδου που ορίζεται από τρία σημεία γίνεται χρησιμοποιώντας τον όγκο του τετραέδρου που ορίζουν τα τέσσερα σημεία. Ο έλεγχος αυτός μπορεί να εκτελεστεί σε χρόνο ανάλογο του πλήθους των εδρών του CH_{i-1} που είναι $O(n)$.

Όταν το σημείο ρ_i δεν ανήκει στο CH_{i-1} το πρόβλημα είναι πιο δύσκολο καθώς το περίβλημα θα πρέπει να τροποποιηθεί κατάλληλα. Η τροποποίηση απαιτεί την εύρεση δύο εφαπτόμενων επιπέδων τα οποία περικλείουν έναν «κώνο» από τριγωνικές έδρες όπου καθεμιά από αυτές ορίζεται από το ρ_i και μία ακμή του CH_{i-1}. Ας υποθέσουμε ότι βρισκόμαστε στο σημείο ρ_i και κοιτάζουμε προς το CH_{i-1}. Επίσης υποθέτουμε ότι το ρ_i δεν είναι συν επίπεδο με καμία έδρα του πολυέδρου: δηλαδή μία έδρα του CH_{i-1} είτε είναι εντελώς ορατή είτε δεν είναι καθόλου ορατή από το ρ_i. Δεν είναι δύσκολο να παρατηρήσει κάνεις ότι οι ορατές έδρες είναι ακριβώς οι έδρες που θα αγνοήσουμε κατά την μετάβαση από το CH_{i-1} στο CH_i. Επιπλέον οι ακμές του ορίζοντα, οι ακμές δηλαδή που περικλείουν την ορατή περιοχή, είναι ακριβώς οι ακμές που μαζί με το ρ_i ορίζουν τις έδρες του κώνου. Επειδή έχουμε ένα κυρτό πολύεδρο και ένα σημείο, ο ορίζοντας στην περίπτωση που αντιμετωπίζουμε είναι μία απλή κλειστή πολυγωνική γραμμή κατά μήκος των ακμών του CH_{i-1}.

Οι παραπάνω διαπιστώσεις συνεπάγονται ότι ο καθορισμός των ορατών και μη ορατών εδρών του CH_{i-1} μας επιτρέπει να βρούμε τις ακμές του ορίζοντα, και άρα να κατασκευάσουμε τον κώνο, αλλά και να εντοπίσουμε τις έδρες που θα αγνοήσουμε. Το εάν μία έδρα είναι ορατή ή όχι εξαρτάται από το εάν το ρ_i είναι δεξιά από το επίπεδο της έδρας ή όχι αντίστοιχα. Άρα, σύμφωνα με όσα αναφέραμε κατά τον έλεγχό του, εάν το σημείο ρ_i ανήκει στο

CH_{i-1} ή όχι, μία τριγωνική έδρα (a, b, c) είναι ορατή από το ρ_i αν και μόνο αν ο (προσημασμένος) όγκος του τετραέδρου που ορίζουν τα (a, b, c) και ρ_i είναι αρνητικός. Μάλιστα, ο έλεγχος για το εάν το σημείο ρ_i ανήκει στο CH_{i-1} και ο εντοπισμός των ορατών εδρών μπορούν να γίνουν παράλληλα.

Δεδομένου ότι $F = O(n)$ και $E = O(n)$, όπου n είναι το πλήθος των κορυφών ενός πολυέδρου, οι βρόχοι που διατρέχουν τις έδρες και τις ακμές απαιτούν γραμμικό χρόνο. Καθώς αυτοί οι βρόχοι εκτελούνται για την επανάληψη ενός άλλου εξωτερικού βρόχου που εκτελείται n φορές, η συνολική πολυπλοκότητα χρόνου είναι τετραγωνική δηλαδή $O(n^2)$.

Ακολουθεί ο Αυξητικός Αλγόριθμος με τη μορφή αναλυτικών βημάτων:

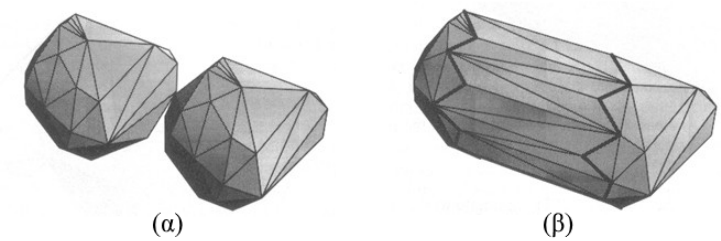
Αυξητικός αλγόριθμος

1. Σχημάτισε το τετράεδρο CH_4 με τα σημεία $\rho_1, \rho_2, \rho_3, \rho_4$.
2. Για i από 5 έως n επανάλαβε
   2.1. Θέσε το πλήθος των ορατών εδρών ίσο με 0.
   2.2. Για κάθε έδρα f του CH_{i-1} επανάλαβε
      2.2.1. Αν ο όγκος του τετραέδρου $(f, \rho_i) < 0$, τότε η f είναι ορατή οπότε αύξησε το πλήθος των ορατών εδρών κατά 1.
   2.3. Αν το πλήθος των ορατών εδρών είναι ίσο με 0, τότε θέσε $CH_i = CH_{i-1}$ διαφορετικά
      2.3.1. Για κάθε ακμή e του CH_{i-1} επανάλαβε
         2.3.1.1. Κατασκεύασε την έδρα του κώνου που ορίζεται από το e και το ρ_i.
      2.3.2. Για κάθε ορατή έδρα f επανάλαβε
         2.3.2.1. Διάγραψε την f.
      2.3.3. Ενημέρωσε το κυρτό περίβλημα CH_i

8.2.3 Αλγόριθμος «Διαιρεί και Βασίλευε»

Ο αλγόριθμος «Διαιρεί και Βασίλευε» των Preparata και Hong [PH77] επεκτείνεται σχετικά εύκολα από τις δύο στις τρεις διαστάσεις και επιτυγχάνει τη βέλτιστη πολυπλοκότητα χρόνου $O(n \log n)$. Ο αλγόριθμος είναι πολύ σημαντικός αλλά δύσκολος να υλοποιηθεί και έτσι δεν χρησιμοποιείται στην πράξη στο βαθμό που χρησιμοποιούνται άλλοι αλγόριθμοι όπως, για παράδειγμα, ο Αυξητικός Αλγόριθμος. Η λογική του αλγορίθμου στις

τρεις διαστάσεις είναι ίδια με την αντίστοιχη στις δύο: τα σημεία ταξινομούνται ως προς τη x-συντεταγμένη τους, χωρίζονται σε δύο σύνολα, κατασκευάζονται αναδρομικά τα κυρτά περιβλήματα των δύο συνόλων τα οποία και συγχωνεύονται κατόπιν σε ένα συνολικό κυρτό περίβλημα (βλ. Σχ. 8.4). Η συγχώνευση πρέπει να εκτελεστεί σε $O(n)$ χρόνο ώστε να επιτευχθεί η επιθυμητή συνολική πολυπλοκότητα χρόνου $O(n \log n)$.



Σχήμα 8.4: (α) Διαχωρισμός των σημείων και κατασκευή των κυρτών περιβλημάτων των δύο συνόλων, (β) Συγχώνευση σε ένα συνολικό κυρτό περίβλημα

8.2.4 Αλγόριθμος Ταχέως Περιβλήματος

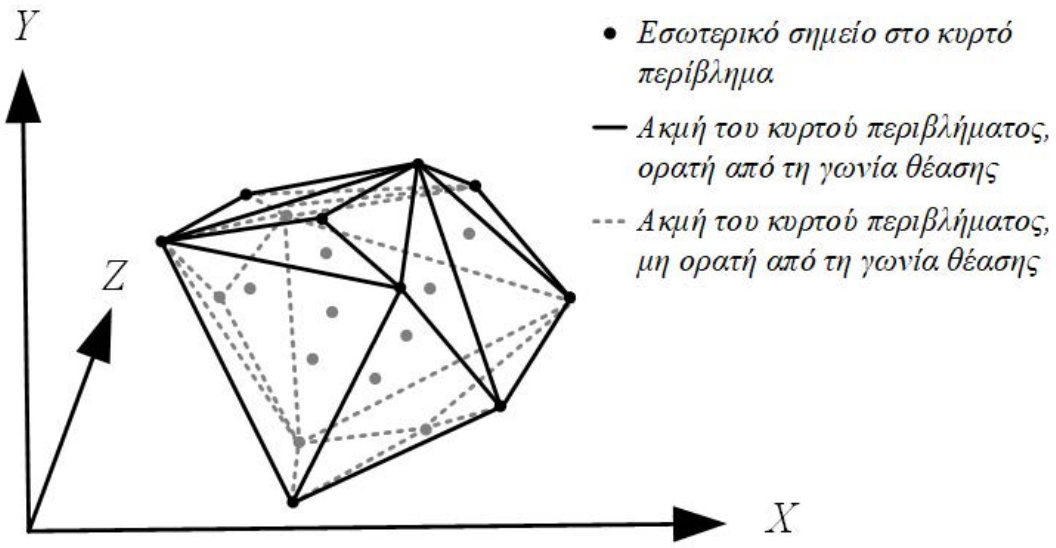
Ένας πολύ ευέλικτος και γρήγορος αλγόριθμος υπολογισμού κυρτού περιβλήματος δημιουργήθηκε το 1996 από τους C. Bradford Barber, David P. Dobkin και Hanuu HuhDanpaa [BDH96] ο οποίος ονομάστηκε Αλγόριθμος Ταχέως Περιβλήματος (QuickHull) ή εν συντομία Α.Τ.Π. Αποτελεί μία παραλλαγή του αλγόριθμου των Clarkson και Shor [CS89] με τη διαφορά ότι η επεξεργασία γίνεται στο χώρο των σημείων και των κυρτών περιβλημάτων και όχι στον δυικό χώρο των ημι-χώρων και των πολυτόπων. Αυτό έχει ως αποτέλεσμα το προκύπτον γράφημα να αντικαθίσταται από ένα εξωτερικό σύνολο για κάθε όψη. Ένα σημείο περιέχεται στο εξωτερικό σύνολο μίας όψης μόνο αν βρίσκεται άνωθεν της όψης. Όπως στον αλγόριθμο των Clarkson και Shor, ένα μη επεξεργασμένο σημείο βρίσκεται μόνο σε ένα εξωτερικό σύνολο αποκλειστικά. Η παραλλαγή του Α.Τ.Π. έγκειται στο ότι επεξεργάζεται το πιο μακρινό σημείο ενός εξωτερικού συνόλου σε αντίθεση με ένα τυχαίο σημείο.

Ο Α.Τ.Π. μπορεί να συγκριθεί με τους αυξητικούς αλγόριθμους με μία διαφοροποίηση στο βήμα επιλογής επόμενου σημείου. Αν ο Α.Τ.Π. επιλέγει ένα τυχαίο σημείο αντί για το πιο μακρινό, τότε μετατρέπεται σε έναν αυξητικό αλγόριθμο. Σε πρακτικές δοκιμές, ο Α.Τ.Π. εκτελείται ταχύτερα από τους παρόμοιας λογικής αλγόριθμους επειδή επεξεργάζεται λιγότερα εσωτερικά σημεία. Επιπλέον, ο Α.Τ.Π. επαναχρησιμοποιεί τη μνήμη που καταλαμβάνουν οι παλαιότερες όψεις.

Ορίζονται δύο συνθήκες οι οποίες εγγυούνται πανομοιότυπη συμπεριφορά με αυτή των τυχαιοποιημένων αλγορίθμων, όταν όλα τα σημεία έχουν ακραίες τιμές. Αν οι συνθήκες αυτές ικανοποιούνται, ο Α.Τ.Π. στον $\mathbb{R}^3$

εκτελείται σε χρόνο $O(n \log r)$, όπου n το σύνολο των σημείων και r τα επεξεργασμένα σημεία. Σε μικρότερες διαστάσεις ο χρόνος εκτέλεσης είναι επίσης $O(n \log r)$, ενώ σε μεγαλύτερες είναι $O(nf_r/r)$, όπου f_r είναι ο μέγιστος αριθμός όψεων για r κορυφές. Αν η ακρίβεια είναι περιορισμένη, τότε ο χρόνος εκτέλεσης γίνεται $O(\log n)$. Ο αριθμός των σημείων που επεξεργάζονται με τον Α.Τ.Π. είναι ανάλογος με τον αριθμό των κορυφών του κυρτού περιβλήματος κατά την έξοδο του αλγορίθμου.

Υποθέτουμε ότι τα σημεία είναι σε γενικευμένες θέσεις (και όχι ότι $d+1$ σημεία ορίζουν ένα επίπεδο διάστασης $d-1$), οπότε ορίζουν ένα σύμπλεγμα σημείων (Preparata και Shamos [PS84]). Ένα κυρτό περίβλημα d διάστασης περιγράφεται από τις κορυφές του και από τις $d-1$ διαστάσεων όψεις του. Ένα σημείο είναι ακραίο αν αποτελεί κορυφή του κυρτού περιβλήματος. Μία όψη εμπεριέχει ένα σύνολο κορυφών, ένα σύνολο γειτονικών όψεων και μία εξίσωση υπερεπιπέδων. Οι $d-2$ διαστάσεων όψεις είναι οι κορυφογραμμές του κυρτού περιβλήματος. Κάθε κορυφογραμμή είναι η τομή κορυφών από δύο γειτονικές όψεις.



Σχήμα 8.5: Κυρτό περίβλημα με εφαρμογή του Α.Τ.Π. σε ένα αυθαίρετο σύνολο σημείων του $\mathbb{R}^3$

Ο Α.Τ.Π. χρησιμοποιεί δύο βασικές διαδικασίες της υπολογιστικής γεωμετρίας: το προσανατολισμένο πολυεπίπεδο d σημείων και την προσημασμένη απόσταση από πολυεπίπεδο. Ένα πολυεπίπεδο περιγράφεται από την κανονικοποιημένη συνάρτηση των εξωτερικών του σημείων και από την απόστασή του από την αρχή των αξόνων. Η απόσταση ενός σημείου από ένα πολυεπίπεδο είναι το εσωτερικό γινόμενο του σημείου και της συνάρτησης περιγραφής του πολυεπιπέδου συν την απόσταση από την αρχή των αξόνων. Το πολυεπίπεδο ορίζει έναν ημιχώρο σημείων που έχουν αρνητικές αποστάσεις από το πολυεπίπεδο. Αν η απόσταση είναι θετική, το σημείο είναι

άνωθεν του πολυεπιπέδου.

Για την επεξεργασία ενός σημείου, χρησιμοποιείται μία απλοποίηση του θεωρήματος Beneath-Beyond του Grünbaum [Grü63]. Οι τυχαιοποιημένοι αυξητικοί αλγόριθμοι βασίζονται σε αυτό το θεώρημα.

Θεώρημα 8.2.1. *Απλοποιημένο Beneath-Beyond*
Έστω H, το κυρτό περίβλημα στον $\mathbb{R}^d$, και έστω p ένα σημείο στον $\mathbb{R}^d$–H. Τότε το F είναι μία όψη του κυρτού περιβλήματος $CH(p \cup H)$ τότε και μόνο, αν
(1) Το F είναι μία όψη του H, και το p είναι κάτωθι του F, ή
(2) Το F δεν είναι μία όψη του H, και οι κορυφές του είναι η p και οι κορυφές μίας κορυφογραμμής του H με μία προσπίπτουσα όψη κάτωθι του p και τις άλλες προσπίπτουσες όψεις άνωθεν του p.

Το βασικό πρόβλημα του Beneath-Beyond είναι η αποτελεσματική εξακρίβωση των ορατών όψεων. Από τη στιγμή που μία όψη συνδέεται με τις γειτονικές, η εξακρίβωση μίας ορατής όψης επιτρέπει την εξακρίβωση και των υπολοίπων γρήγορα. Οι περισσότεροι από τους τυχαιοποιημένους αυξητικούς αλγορίθμους χρησιμοποιούν τις προηγουμένως υπολογισθείσες όψεις για να εξακριβώσουν την πρώτη ορατή όψη από ένα σημείο. Η λύση που προτείνεται όμως με τον Α.Τ.Π. είναι απλούστερη. Μετά την αρχικοποίηση, ο αλγόριθμος εκχωρεί ένα μη επεξεργασμένο σημείο σε ένα εξωτερικό σύνολο. Εξ ορισμού, η αντίστοιχη όψη είναι ορατή από το σημείο.

Όταν ο Α.Τ.Π. δημιουργεί έναν κώνο από νέες όψεις, δημιουργεί νέα εξωτερικά σύνολα από τα εξωτερικά σύνολα των ορατών όψεων. Αυτή η διαδικασία ονομάζεται διαμέριση και εντοπίζει μία ορατή νέα όψη από ένα σημείο. Αν ένα σημείο ευρίσκεται άνωθεν πολλαπλών νέων όψεων, τότε μία από τις νέες όψεις επιλέγεται. Αν ένα σημείο ευρίσκεται κάτωθεν όλων των νέων όψεων, τότε το σημείο αυτό είναι εσωτερικό στο κυρτό περίβλημα οπότε απορρίπτεται. Η διαμέριση υπολογίζει επίσης το πιο μακρινό σημείο για κάθε εξωτερικό σύνολο.

Ο Α.Τ.Π. επιλέγει ένα μη εκφυλισμένο σύνολο σημείων για το αρχικό σύμπλεγμα σημείων. Αν είναι δυνατό, επιλέγει είτε σημεία με μέγιστες είτε με ελάχιστες συντεταγμένες. Ακολουθεί ο Αλγόριθμος Ταχέως Περιβλήματος με τη μορφή αναλυτικών βημάτων:

Ο Αλγόριθμος Ταχέως Περιβλήματος στον $\mathbb{R}^d$

1. Δημιούργησε ένα σύμπλεγμα $d + 1$ σημείων.
2. Για κάθε όψη F επανάλαβε
   2.1. Για κάθε μη προσημασμένο σημείο p, επανάλαβε

2.1.1. Αν το p είναι άνωθεν του F, συμπεριέλαβε το p στο εξωτερικό σύνολο του F

3. Για κάθε όψη F με μη κενό εξωτερικό σύνολο, επανάλαβε

3.1. Επίλεξε το μακρινότερο σημείο p από το εξωτερικό σύνολο του F

3.2. Αρχικοποίησε το σύνολο V των ορατών όψεων στο F

3.3. Για όλες τις μη επεξεργασμένες γειτονικές όψεις N του V,

3.3.1. Αν το p είναι άνωθεν του N, πρόσθεσε το N στο V

3.4. Το σύνορο του V είναι το σύνολο των οριζόμενων κορυφογραμμών H

3.5. Για κάθε κορυφογραμμή R του H

3.5.1. Δημιούργησε μία νέα όψη από το R και το p

3.5.2. Σύνδεσε την καινούρια όψη με τις γειτονικές της

3.6. Για κάθε νέα όψη F'

3.6.1. Για κάθε μη αντιστοιχισμένο σημείο q μέσα σε ένα εξωτερικό σύνολο από μία όψη του V,

3.6.1.1. Αν το q είναι άνωθεν του F', αντιστοίχησε το q στο εξωτερικό σύνολο του F'.

3.7. Διάγραψε τις όψεις στο V

Για να αποδειχθεί η ορθότητα του Α.Τ.Π., πρέπει πρώτα να αποδειχθεί ότι ένα σημείο μπορεί να διαμεριστεί σε ένα έγκυρο εξωτερικό σύνολο. Με αυτό τον τρόπο, ακραίες τιμές στην είσοδο θα αντιστοιχούν σε κορυφές στη έξοδο.

Λήμμα 8.2.1. *Αν ένα ακραίο αρχικό σημείο είναι άνωθεν δύο ή περισσοτέρων όψεων σε ένα βήμα διαμερισμού στον Α.Τ.Π., τότε αυτό προστίθεται στο περίβλημα ανεξαρτήτως σε ποιο εξωτερικό σύνολο είναι αντιστοιχισμένο.*

Απόδειξη. Έστω ότι ισχύει το αντίθετο και έστω ένα ακραίο σημείο p το οποίο δεν είναι αντιστοιχισμένο σε ένα εξωτερικό σύνολο και γι' αυτό δεν προστίθεται στο κυρτό περίβλημα. Από τη στιγμή που το p είναι ακραίο σημείο, πρέπει να είναι εξωτερικό σε τουλάχιστον μία όψη του αρχικού συμπλέγματος. Από την υπόθεση, υπάρχει ένα σημείο q με το p να ανήκει στα ορατά εξωτερικά σύνολά του αλλά όχι στα νέα εξωτερικά σύνολά του. Έτσι το p είναι άνωθεν μίας ορατής όψης και κάτωθι όλων των νέων ορατών όψεων από το q. Αυτό σημαίνει ότι το p είναι στο εσωτερικό του κυρτού περιβλήματος και γι' αυτό δεν αποτελεί ακραίο σημείο. □

Θεώρημα 8.2.2. *Ο Α.Τ.Π. παράγει το κυρτό περίβλημα ενός συνόλου σημείων στον* $\mathbb{R}^d$.

Απόδειξη. Ο Α.Τ.Π. εκκινεί με το κυρτό περίβλημα $d+1$ σημείων. Ο Α.Τ.Π. διαμερίζει τα σημεία σε εξωτερικά σύνολα και διαλέγει το πιο μακρινό για επεξεργασία. Από το λήμμα 8.2.1, ο διαμερισμός δεν μπορεί να αποτρέψει ένα ακραίο σημείο από το να επεξεργαστεί. Ο Α.Τ.Π. επεξεργάζεται ένα σημείο σύμφωνα με το Θεώρημα 8.2.1. Διατηρεί την όψη στο μέρος (1) του θεωρήματος και κατασκευάζει νέες όψεις όπως ορίζεται στο μέρος (2). Παράγει το κυρτό περίβλημα των επεξεργασμένων σημείων. Η επεξεργασία του τελευταίου σημείου παράγει το κυρτό περίβλημα του αρχικού συνόλου εισόδου. □

Οι περισσότεροι από τους τυχαιοποιημένους αυξητικούς αλγορίθμους εκτελούν τα ίδια βήματα με τον Α.Τ.Π., αλλά με διαφορετική σειρά. Χρησιμοποιούν τον Beneath-Beyond με τυχαία μετάθεση των σημείων. Βρίσκουν μία ορατή όψη με μία κατά-βάθος αναζήτηση από το προηγούμενο κυρτό περίβλημα. Θεωρούν την ακολουθία των όψεων που έχουν δοκιμαστεί για ένα σημείο. Ο Α.Τ.Π. είναι δυνατόν να δοκιμάσει την ίδια ακολουθία κατά τη διάρκεια διαδοχικών διαμερισμών σημείου σε εξωτερικά σύνολα.

Οι Edelsbrunner και Shah [ES92], ο Joe [Joe91] και οι Boissonnat και Teillaud [BT93] χρησιμοποιούν μία παρόμοια μέθοδο για τριγωνισμούς Delaunay και εκφράζουν τους αλγορίθμους τους ως προς τους τριγωνισμούς και τον έλεγχο εντός της σφαίρας. Με την αντιστοιχία μεταξύ του τριγωνισμού Delaunay και του κυρτού περιβλήματος, δηλαδή ότι κάθε τρίγωνο είναι μία όψη του κυρτού περιβλήματος, η εντός της σφαίρας δοκιμή καθορίζει τις ορατές όψεις για το ανυψωμένο σημείο. [Bro79].

9. ΟΓΚΟΣ ΚΥΡΤΟΥ ΠΟΛΥΕΔΡΟΥ

Όγκος ονομάζεται η ποσότητα του χώρου που καταλαμβάνει ένα αντικείμενο. Έχει, δηλαδή, αριθμητική αξία και δίνεται για να περιγράψει την τριδιάστατη αντίληψη για το πόσο χώρο κατέχει ένα αντικείμενο. Συμβολίζεται συνήθως με το αγγλικό γράμμα V (Volume). Μίας διάστασης αντικείμενα (όπως γραμμές) και δύο διαστάσεων αντικείμενα (όπως τετράγωνα) έχουν μηδενικό όγκο στον τριδιάστατο χώρο.

Η διεθνής μονάδα μέτρησης είναι το κυβικό μέτρο (m^3) που αντιστοιχεί στον όγκο ενός κύβου με πλευρά ένα μέτρο. Στο αγγλικό σύστημα αρίθμησης είναι το κυβικό πόδι (ft^3), και γενικά σε κάθε διαφορετικό σύστημα μέτρησης αντιστοιχεί η «κυβική» μονάδα μέτρησης της απόστασης.

9.1 Μέθοδοι υπολογισμού του όγκου ενός κυρτού πολυέδρου

Ο άμεσος υπολογισμός του όγκου είναι εφικτός για αντικείμενα με απλό σχήμα, όπως παραδείγματος χάριν κύβος, ορθογώνιο παραλληλεπίπεδο, πυραμίδα, κώνος, σφαίρα, κύλινδρος, πρίσμα κτλ. Ωστόσο για πολύεδρα με πιο πολύπλοκο σχηματισμό, η διαδικασία υπολογισμού του όγκου δυσχεραίνει. Συγκεκριμένα, το αντικείμενο διασπάται σε επιμέρους απλούστερα τμήματα, όπου είναι δυνατόν η απευθείας μέτρηση του όγκου τους. Έτσι ο όγκος του αρχικού αντικειμένου ισούται με το άθροισμα όλων των όγκων των τμημάτων του που έχει διαμεριστεί. Ο γενικός μαθηματικός τύπος υπολογισμού του όγκου οποιουδήποτε γεωμετρικού στερεού σώματος είναι

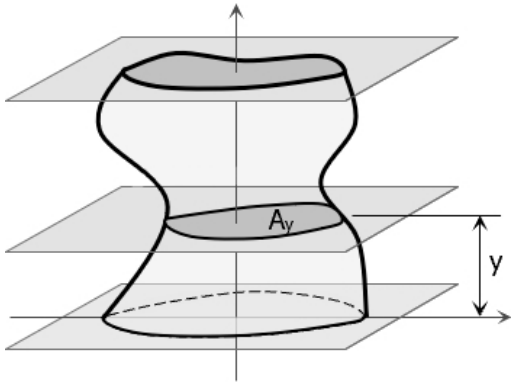
$$V = \int A(h)dh$$

όπου h μία διάσταση του γεωμετρικού στερεού σώματος και A η συνάρτηση του εμβαδού διατομής του ως προς τη διάσταση h.

9.1.1 Πρισματική εξίσωση

Ένα πρισματοειδές είναι ένα στερεό τέτοιο ώστε το εμβαδόν οποιουδήποτε τμήματος, ας το σημειώσουμε Ay, που βρίσκεται σε απόσταση y από ένα σταθερό επίπεδο και είναι παράλληλο προς αυτό, μπορεί να εκφραστεί ως

πολυώνυμο y βαθμού, όχι μεγαλύτερου από τρία (βλ. Σχ. 9.1).



Σχήμα 9.1: Πρισματοειδές στερεό

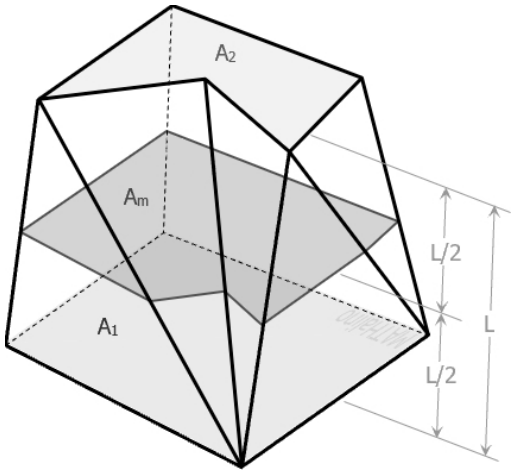
Ένα στερεό είναι πρισματοειδές εάν ισχύει

$$A_y = a + by + cy^2 + dy^3$$

όπου τα a, b, c και d είναι αυθαίρετες σταθερές που μπορεί να είναι θετικές, αρνητικές ή μηδενικές.

Τα πρίσματα και οι κύλινδροι γενικά είναι πρισματικά στα οποία το Ay είναι σταθερό, δηλαδή η τιμή των b, c και d είναι μηδέν. Οι κώνοι και οι πυραμίδες συμπεριλαμβανομένων των κόλουρων είναι πρισματοειδείς πρώτου βαθμού, που σημαίνει ότι η τιμή των c και d είναι μηδέν. Οι σφαίρες και το τμήμα των σφαιρών είναι παραδείγματα πρισματοειδών δευτέρου βαθμού.

Ένα πρισματοειδές πολύεδρο έχει βάσεις που περιέχουν όλες τις κορυφές του σε δύο παράλληλα επίπεδα. Οι πλευρικές όψεις είναι τρίγωνα ή τραπεζοειδή ή παραλληλόγραμμα με τη μία πλευρά να βρίσκεται σε μία από τις δύο βάσεις του (βλ. Σχ. 9.2).



Σχήμα 9.2: Πρισματοειδές πολύεδρο

Οι κύβοι, τα κυβοειδή, τα πρίσματα, οι πυραμίδες και οι κόλουρες πυραμίδες είναι παραδείγματα βασικών πρισματοειδών. Τα πρισματικά που έχουν κορυφές ίσες σε αριθμό και στις δύο βάσεις ονομάζονται πρισμοειδή. Ο όγκος του πρισματοειδούς δίνεται από αυτόν τον τύπο:

$$V = \frac{L}{6}[A_1 + 4A_m + A_2]$$

όπου A_1 και A_2 είναι τα εμβαδά παράλληλων βάσεων, A_m είναι το εμβαδόν του τμήματος στο μέσον μεταξύ A_1 και A_2 και L είναι η κάθετη απόσταση μεταξύ A_1 και A_2.

Προφανώς, η πρισματική εξίσωση είναι εφαρμόσιμη σε πρισματικά πολύεδρα με παράλληλες όψεις άνω και κάτω. Είναι λιγότερο γενική σε σχέση με άλλες μεθόδους οι οποίες θα εξεταστούν στη συνέχεια αλλά απλούστερη σε συγκεκριμένες περιπτώσεις.

9.1.2 Μέθοδος Monte Carlo

Η μέθοδος Monte Carlo είναι μία στατιστική τεχνική που αξιοποιεί την τυχαιότητα για την επίλυση προβλημάτων που μπορεί να είναι κατά βάση αιτιοκρατικά. Είναι ιδιαίτερα χρήσιμη για την εκτίμηση πολύπλοκων ολοκληρωμάτων και για την επίλυση πολυδιάστατων προβλημάτων όπου οι αναλυτικές λύσεις είναι δύσκολο να βρεθούν. Στο πλαίσιο της εύρεσης του όγκου ενός κυρτού πολυέδρου, η μέθοδος Monte Carlo προσφέρει μία πρακτική προσέγγιση, ειδικά όταν έχουμε να κάνουμε με περίπλοκα σχήματα ή μεγαλύτερες διαστάσεις. Η συγκεκριμένη μέθοδος περιλαμβάνει τον τυχαίο δειγματισμό σημείων εντός ενός γνωστού οριακού όγκου που περιέχει το πολύεδρο. Καθορίζοντας την αναλογία των σημείων που εμπίπτουν μέσα στο πολύεδρο σε σχέση με τον συνολικό αριθμό

των σημείων που ελήφθησαν κατά τη δειγματοληψία, μπορεί να εκτιμηθεί και ο όγκος του πολυέδρου [LZZ07].

Τα βήματα του αλγορίθμου ο οποίος προσδιορίζει τον όγκο ενός πολυέδρου χρησιμοποιώντας τη μέθοδο Monte Carlo είναι τα ακόλουθα:

1. Επίλεξε ένα ορθογώνιο παραλληλεπίπεδο το οποίο οριοθετεί το πολύεδρο και προσδιόρισε τον όγκο του μέσω της εξίσωσης $V_{bounding} = (x_{\max} - x_{\min}) \times (y_{\max} - y_{\min}) \times (z_{\max} - z_{\min})$.

2. Δημιούργησε N τυχαία σημεία $r_i = (x_i, y_i, z_i)$ εντός του οριοθετημένου όγκου. Αυτό μπορεί να γίνει χρησιμοποιώντας ομοιόμορφες τυχαίες κατανομές για τις συντεταγμένες x_i, y_i, z_i εντός των ορίων $|x_{\min}, x_{\max}|$, $|y_{\min}, y_{\max}|$ και $|z_{\min}, z_{\max}|$ αντίστοιχα.

3. Για κάθε τυχαίο σημείο r_i προσδιόρισε αν αυτό βρίσκεται εντός του πολυέδρου. Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας διάφορες γεωμετρικές τεχνικές, όπως για παράδειγμα η μέθοδος χύτευση ακτίνων.

4. Καταμέτρησε τα σημεία τα οποία βρίσκονται εντός του πολυέδρου, έστω ότι αυτά είναι N_{inside}.

5. Υπολόγισε κατά προσέγγιση τον όγκο χρησιμοποιώντας τον τύπο $V_{polyhedron} = V_{bounding} \times \frac{N_{inside}}{N}$.

Η συγκεκριμένη μέθοδος είναι αρκετά απλή στην εφαρμογή και την κατανόησή της, μπορεί να εφαρμοστεί σε οποιοδήποτε σχήμα, συμπεριλαμβανομένων και των εξαιρετικά ακανόνιστων, ενώ λειτουργεί και σε υψηλότερες διαστάσεις χωρίς σημαντικές τροποποιήσεις. Στον αντίποδα έχουμε το γεγονός ότι πρόκειται για μία υπολογιστικά ακριβή μέθοδο καθώς απαιτεί μεγάλο αριθμό τυχαίων δειγμάτων, ενώ η ακρίβειά της εξαρτάται από τον αριθμό των δειγμάτων αυτών. Όσο περισσότερα είναι τα τυχαία δείγματα τόσο αυξάνεται το υπολογιστικό κόστος.

9.1.3 Θεώρημα Αποκλίσεως

Ο όγκος ενός προσανατολιζόμενου πολυέδρου μπορεί να υπολογιστεί με τη βοήθεια του Θεωρήματος Αποκλίσεως, επίσης γνωστό ως θεώρημα του Gauss ή θεώρημα του Ostrogradsky [Kat79], το οποίο σχετίζει τη ροή ενός διανυσματικού πεδίου μέσω μίας κλειστής επιφάνειας προς την απόκλιση του πεδίου στον όγκο που περικλείεται. Πιο συγκεκριμένα, το Θεώρημα Αποκλίσεως δηλώνει ότι το επιφανειακό ολοκλήρωμα ενός διανυσματικού πεδίου σε μία κλειστή επιφάνεια, που ονομάζεται «ροή» μέσω της επιφάνειας, ισούται με το ολοκλήρωμα όγκου της απόκλισης στην περιοχή που περικλείεται από την επιφάνεια. Διαισθητικά, δηλώνει ότι «το άθροισμα όλων των πηγών του πεδίου σε μία περιοχή δίνει την καθαρή προς τα έξω ροή».

Το Θεώρημα Αποκλίσεως είναι ένα σημαντικό για τα μαθηματικά, τη φυσική και την μηχανική, ιδιαίτερα στην ηλεκτροστατική και τη δυναμική των ρευστών. Σε αυτά τα πεδία, εφαρμόζεται συνήθως στις τρεις διαστάσεις. Ωστόσο, γενικεύεται σε οποιονδήποτε αριθμό διαστάσεων. Σε μία διάσταση, είναι ισοδύναμο με το θεμελιώδες θεώρημα του λογισμού. Σε δύο διαστάσεις, είναι ισοδύναμο με το θεώρημα του Green.

Το Θεώρημα Αποκλίσεως περιγράφεται παρακάτω:

Θεώρημα 9.1.1. *(Θεώρημα Αποκλίσεως ή Gauss ή Ostrogradsky):*
Έστω $V \subset \mathbb{R}^n$ (στην περίπτωση όπου $n = 3$, το V αντιστοιχεί σε όγκο στον τριδιάστατο χώρο) ένα συμπαγές σύνολο με τμηματικά ομαλό όριο $S = \partial V$. Αν το $\mathbf{F}$ είναι ένα συνεχές διαφορίσιμο διανυσματικό πεδίο οριζόμενο σε μία ανοικτή περιοχή του V, τότε ισχύει

$$\iiint_V (\nabla \cdot \mathbf{F}) dV = \oiint_S (\mathbf{F} \cdot \hat{\mathbf{n}}) dS.$$

Το αριστερό μέρος της ισότητας αντιστοιχεί στο ολοκλήρωμα του όγκου V, ενώ το δεξιό μέρος της ισότητας στο ολοκλήρωμα για το όριο του όγκου V. Το κλειστό ∂V αποτελεί γενικά το όριο του όγκου V προσανατολιζόμενο από κατευθυνόμενα προς τα έξω φυσιολογικά ανύσματα, ενώ το n αποτελεί το κατευθυνόμενο προς τα έξω φυσιολογικό μοναδιαίο διανυσματικό πεδίο του ορίου ∂V. Το dS μπορεί να χρησιμοποιηθεί σαν μία συντόμευση του ndS. Με τον κυκλικό συμβολισμό ανάμεσα στα δύο ολοκληρώματα του δεξιού τμήματος της ισότητας τονίζεται ότι το ∂V αποτελεί μία κλειστή επιφάνεια.

Έστω διανυσματικό πεδίο $\overrightarrow{F}(\vec{x}) = \frac{1}{3}\vec{x}$, του οποίου η απόκλιση είναι ίση στη βέλτιστη περίπτωση με 1. Το Θεώρημα της Αποκλίσεως ορίζει ότι ο όγκος οποιασδήποτε περιοχής Ω είναι

$$Volume(\Omega) = \int_\Omega (\nabla \cdot \mathbf{F}) dV = \oint_S (\vec{\mathbf{F}} \cdot \hat{\mathbf{n}}) dS$$

Στην περίπτωση όπου Ω είναι η περιοχή που περικλείεται από ένα πολύεδρο, τότε η παραπάνω ισότητα απλοποιείται στην

$$Volume = \frac{1}{3} \sum_{face\, i} \overrightarrow{x_i} \cdot \widehat{n_i} A_i$$

όπου για την i-ιοστη, $\overrightarrow{x_i}$ είναι οποιοδήποτε σημείο της όψης,

$\widehat{n_i}$ είναι το φυσιολογικό άνυσμα,

A_i είναι το εμβαδόν της όψης.

9.1.4 Μέθοδος του Franklin

Μία διαφορετική προσέγγιση για τον υπολογισμό του όγκου ενός πολυέδρου παρουσιάστηκε στην [FK93]. Η προσέγγιση αυτή χρησιμοποιεί τοπολογικές πληροφορίες, έτσι ώστε ένα πολύεδρο αναπαρίσταται από ένα σύνολο ομαδοποιημένων πληροφοριών γειτνίασης, οι οποίες συσχετίζουν κορυφές, ακμές και έδρες. Τα γράμματα (V, T, N, B) αποτελούν ένα σύνολο τεσσάρων ομαδοποιημένων πληροφοριών για κάθε κορυφή-ακμή-έδρα γειτνίαση, όπου

V: Συντεταγμένες κορυφής,

T: Μοναδιαίο εφαπτόμενο διάνυσμα από την κορυφή κατά μήκος της ακμής προς την άλλη άκρη της ακμής,

N: Μοναδιαίο φυσιολογικό διάνυσμα στην ακμή, που βρίσκεται στο επίπεδο που ορίζει η έδρα, και με κατεύθυνση από την ακμή προς την έδρα, και

B: Μοναδιαίο φυσιολογικό διάνυσμα στο επίπεδο της έδρας, κατευθυνόμενο στο εσωτερικό του πολυέδρου.

Για παράδειγμα, για έναν κύβο υπάρχουν 6 γειτνιάσεις για κάθε κορυφή και συνολικά 48 γειτνιάσεις.

Ο όγκος ενός πολυέδρου, σύμφωνα με την μεθοδολογία του Franklin, είναι

$$Volume = -\frac{1}{6}\sum(V \cdot T \; V \cdot N \; V \cdot B),$$

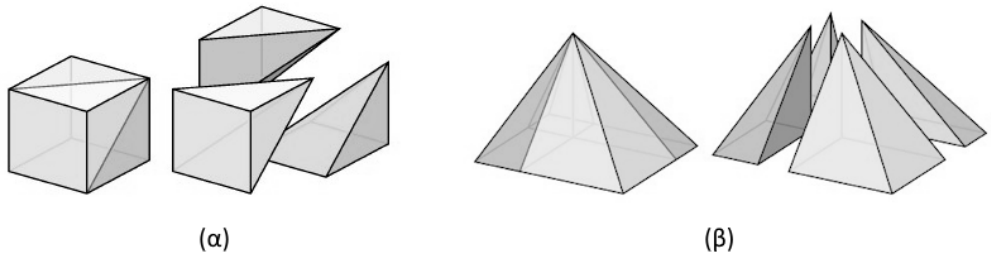
όπου με «·» συμβολίζεται το εσωτερικό γινόμενο δύο διανυσμάτων.

Εδώ πρέπει να τονισθεί ότι η μέθοδος δεν χρησιμοποιεί γενικές πληροφορίες σχετικά με το πολύεδρο. Παραδείγματος χάριν, δεν υπάρχουν δεδομένα για μία ακμή σαν ξεχωριστή οντότητα, παρά μόνο για τις κατευθύνσεις προς τις οποίες οι ακμές απομακρύνονται από τις κορυφές. Η μεθοδολογία αυτή είναι καλή σε περιπτώσεις που το πολύεδρο έχει πολύπλοκη τοπολογία με πολλά ένθετα περιβλήματα και επαναλαμβανόμενες ακμές και σε περιπτώσεις όπου η διάσπαση του πολυέδρου σε επιμέρους τμήματα είναι πολύπλοκη.

9.1.5 Διαμερισμός κυρτού πολυέδρου σε πυραμίδες

Μία αρκετά απλή μέθοδος υπολογισμού του όγκου ενός κυρτού πολυέδρου είναι ο διαμερισμός του σε πυραμίδες, Αυτός μπορεί να πραγματοποιηθεί με την επιλογή ενός σταθερού σημείου, που αντιστοιχεί είτε σε ένα εσωτερικό σημείο του πολυέδρου είτε σε ένα συνοριακό του σημείο, όπως σε μία κορυφή του, και εν συνεχεία στην ένωση του σημείου αυτού με κάθε έδρα του πολυέδρου. Με τον τρόπο αυτό οι πυραμίδες που δημιουργούνται δεν επικαλύπτονται μεταξύ τους, ενώ η ένωσή τους σχηματίζει το αρχικό κυρτό πολύεδρο. Έτσι ο όγκος του

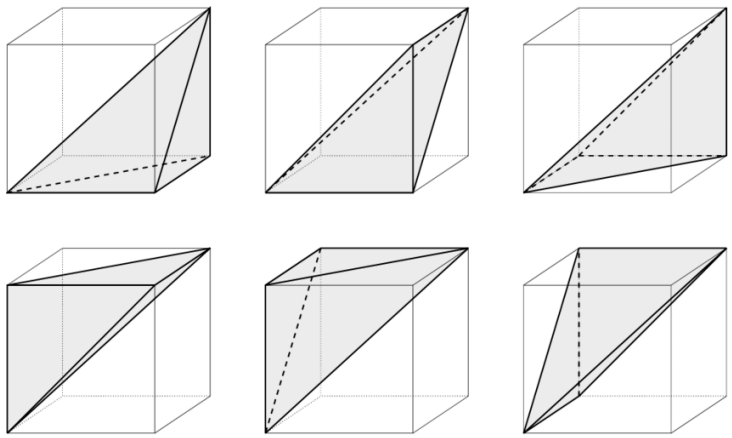
κυρτού πολυέδρου ισούται με το άθροισμα των όγκων όλων των πυραμίδων στις οποίες έχει διαμεριστεί (βλ. Σχ. 9.3). Ο όγκος κάθε πυραμίδας προκύπτει από το γινόμενο του εμβαδού της βάσης της επί το ύψος της.



Σχήμα 9.3: Παραδείγματα διαμερισμού κυρτού πολυέδρου σε πυραμίδες

9.1.6 Διαμερισμός κυρτού πολυέδρου σε τετράεδρα

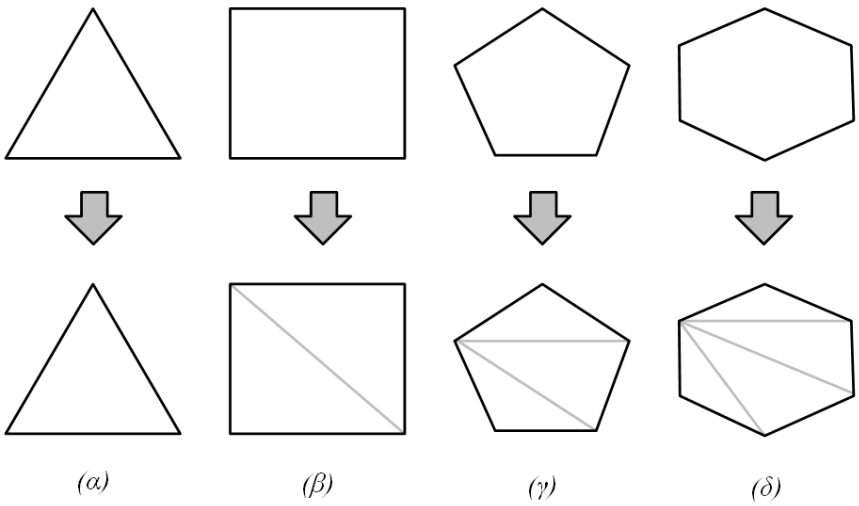
Μία παραλλαγή της παραπάνω μεθοδολογίας είναι ο διαμερισμός του κυρτού πολυέδρου σε τετράεδρα αντί για πυραμίδες. Η διαδικασία είναι παρόμοια με την μόνη διαφορά ότι οι έδρες του κυρτού πολυέδρου πρέπει να είναι τριγωνισμένες (σχήμα 5.2), ενώ επιλέγεται σαν αρχικό σημείο μία κορυφή του πολυέδρου. Έτσι η κορυφή, που έχει επιλεγεί σαν αρχικό σημείο, σχηματίζει τετράεδρα με τα τρίγωνα που αποτελούν την τριγωνισμένη μορφή του κυρτού πολυέδρου. Η παραλλαγή αυτή ταιριάζει απόλυτα στη θεματολογία που εξετάζει η παρούσα μελέτη καθώς τα πολύεδρα που προκύπτουν από τον υπολογισμό κυρτών περιβλημάτων έχουν τις έδρες τους τριγωνισμένες (βλ. Σχ. 9.4).



Σχήμα 9.4: Παράδειγμα διαμερισμού κυρτού πολυέδρου (κύβου) σε τετράεδρα.

Η βασική ιδέα της μεθοδολογίας αυτής είναι η απλοποίηση του προς υπολογισμού όγκου. Ο όγκος ενός κυρτού πολυέδρου δεν μπορεί να υπολογιστεί άμεσα ιδιαίτερα αν το πολύεδρο έχει πολύπλοκο σχηματισμό. Με τον διαμερισμό του όμως σε μικρότερα μη επικαλυπτόμενα τετράεδρα το πρόβλημα του υπολογισμού του όγκου του μετατοπίζεται στον υπολογισμό του όγκου των επιμέρους τετραέδρων. Ο όγκος του αρχικού κυρτού πολυέδρου ισούται με το άθροισμα των όγκων των επιμέρους στα οποία έχει διαμεριστεί, καθώς τα τετράεδρα αυτά δεν επικαλύπτονται και η ένωσή τους ισούται με το αρχικό πολύεδρο.

Για τον διαμερισμό του κυρτού πολυέδρου απαιτείται η επιλογή μίας κορυφής του, η οποία θα αποτελέσει το σταθερό σημείο για τον σχηματισμό των τετραέδρων, ενώ απαραίτητη προϋπόθεση είναι οι έδρες του τετραέδρου να είναι τριγωνισμένες. Παραδείγματα τριγωνισμένων επιφανειών παρουσιάζονται στο Σχ. 9.5, όπου η εκάστοτε επιφάνεια διαμερίζεται σε μικρότερα μη επικαλυπτόμενα τρίγωνα. Κάθε τετράεδρο προκύπτει με την ένωση της κορυφής, που έχει επιλεγεί σαν αρχικό σημείο, με κάθε ένα από τα τρίγωνα της τριγωνισμένης μορφής του πολυέδρου. Έτσι, κάθε τετράεδρο έχει κορυφές την αρχική κορυφή και τις κορυφές του τριγώνου από το οποίο προέρχεται. Από τη διαδικασία δημιουργίας τετραέδρων αποκλείονται τα τρίγωνα που ορίζουν επίπεδο στο οποίο βρίσκεται και η αρχικά επιλεγμένη κορυφή, καθώς τα τρίγωνα αυτά μαζί με την αρχική κορυφή θα δημιουργούσαν τετράεδρα με μηδενικό όγκο, οπότε μη χρήσιμα στην όλη διαδικασία.



Σχήμα 9.5: Αρχικές επιφάνειες και οι τριγωνισμένες τους μορφές

Ο όγκος V ενός τετραέδρου είναι άμεσα υπολογίσιμος και δίνεται από τον παρακάτω τύπο:

$$V = \frac{1}{3} A_0 h \tag{9.1}$$

όπου A_0 είναι το εμβαδόν της βάσης του και h το ύψος από τη βάση προς την κορυφή του.

Για ένα τετράεδρο με κορυφές $a = (a_1, a_2, a_3)$, $b = (b_1, b_2, b_3)$, $c = (c_1, c_2, c_3)$ και $d = (d_1, d_2, d_3)$, ο όγκος είναι $\frac{1}{6} \cdot |det(a-b, b-c, c-d)|$, ή οποιοσδήποτε άλλος συνδυασμός ζευγών κορυφών τέτοιος ώστε να δημιουργείται ένα απλός συνεκτικό γράφημα. Χρησιμοποιώντας το εσωτερικό και το εξωτερικό γινόμενο, παράγεται ο τύπος

$$V = \frac{|(a-d) \cdot ((b-d) \times (c-d))|}{6}.$$

Αν η αρχή των αξόνων του συστήματος συντεταγμένων συμπίπτει με την κορυφή d, τότε $d = 0$, οπότε

$$V = \frac{|a \cdot (b \times c)|}{6}$$

όπου τα a, b, και c αντιστοιχούν σε ακμές που τέμνονται σε μία κορυφή, ενώ το $a \cdot (b \times c)$ είναι το εσωτερικό τριπλό γινόμενο.

Το εσωτερικό τριπλό γινόμενο μπορεί να παρασταθεί με τις ακόλουθες ορίζουσες:

$$6 \cdot V = \begin{vmatrix} a & b & c \end{vmatrix}$$

ή

$$6 \cdot V = \begin{vmatrix} a \\ b \\ c \end{vmatrix}$$

όπου η $a = (a_1, a_2, a_3)$ εκφράζεται σαν διάνυσμα γραμμής ή στήλης αντίστοιχα.

Οπότε προκύπτει

$$36 \cdot V^2 = \begin{vmatrix} a^2 & a \cdot b & a \cdot c \\ a \cdot b & b^2 & b \cdot c \\ a \cdot c & b \cdot c & b^2 \end{vmatrix}$$

όπου $a \cdot b = \alpha\beta cos\gamma$ κτλ., και τελικά

$$V = \frac{abc}{6}\sqrt{1 + 2\cos\alpha\cos\beta\cos\gamma - \cos^2\alpha - \cos^2\beta - \cos^2\gamma}$$

όπου α, β, γ είναι οι γωνίες των επιπέδων σύμφωνα με την κορυφή d. Η γωνία α, είναι η γωνία μεταξύ των δύο ακμών που ενώνουν την κορυφή d με τις κορυφές b και c. Ομοίως η γωνία β, για τις κορυφές a και c, ενώ η γωνία γ ορίζεται από τις κορυφές a και b.

Δοθέντων των αποστάσεων μεταξύ των κορυφών ενός τετραέδρου, ο όγκος μπορεί να υπολογιστεί χρησιμοποιώντας την ορίζουσα Cayley – Menger:

$$288 \cdot V^2 = \begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & d_{12}^2 & d_{13}^2 & d_{14}^2 \\ 1 & d_{12}^2 & 0 & d_{23}^2 & d_{24}^2 \\ 1 & d_{13}^2 & d_{23}^2 & 0 & d_{34}^2 \\ 1 & d_{14}^2 & d_{24}^2 & d_{34}^2 & 0 \end{vmatrix}$$

όπου οι δείκτες $i, j \in \{1, 2, 3, 4\}$ αναπαριστούν τις κορυφές $\{a, b, c, d\}$ και d_{ij} είναι η απόσταση μεταξύ των κορυφών i και j ή διαφορετικά το μήκος της ακμής που ενώνει τις δύο κορυφές i και j. Αρνητική τιμή στην ορίζουσα σημαίνει ότι το τετράεδρο δεν μπορεί να κατασκευαστεί με τις δοθείσες αποστάσεις d_{ij} . Η μεθοδολογία αυτή είναι γνωστή και ως μεθοδολογία Tartaglia και βασίζεται ουσιαστικά στον Piero della Francesca (15ος αιώνας). Αποτελεί την επέκταση στον τριδιάστατο χώρο της μεθοδολογίας του Ήρωνα (1ος αιώνας) για το εμβαδόν ενός τριγώνου.

Σύμφωνα με την μεθοδολογία του Ήρωνα, αν U, V, W, u, v, w είναι τα μήκη των ακμών του τετραέδρου (τα πρώτα τρία μήκη για τις πλευρές του τριγώνου, ενώ u είναι το μήκος της ακμής απέναντι από την ακμή U κτλ.), τότε

$$Volume = \frac{\sqrt{(-a+b+c+d)(a-b+c+d)(a+b-c+d)(a+b+c-d)}}{192uvw}$$

όπου

$$a = \sqrt{xYZ}$$

$$b = \sqrt{yZX}$$

$$c = \sqrt{zXY}$$

$$d = \sqrt{xyz}$$

$$X = (w - U + v)(U + v + w)$$

$$x = (U - v + w)(v - w + U)$$

$$Y = (u - V + w)(V + w + u)$$

$$y = (V - w + u)(w - u + V)$$

$$Z = (v - W + u)(W + u + v)$$

$$z = (W - u + v)(u - v + W)$$

Παρακάτω δίνεται ο αλγόριθμος για την εύρεση του όγκου ενός κυρτού πολυέδρου σύμφωνα με τη μεθοδολογία που περιγράφηκε στην αρχή της ενότητας. Η μεθοδολογία απαιτεί το κυρτό πολύεδρο να είναι τριγωνισμένο:

Αλγόριθμος για την εύρεση του όγκου ενός κυρτού πολυέδρου

1. Επίλεξε μία κορυφή A του πολυέδρου P σαν αρχικό σημείο για το σχηματισμό των τετραέδρων που το διαμερίζουν.
2. Όρισε αρχικά τον όγκο V του P ίσο με 0.
3. Για κάθε τρίγωνο Tr_i της τριγωνισμένης μορφής του P
   3.1. Αν η κορυφή A δεν βρίσκεται στο επίπεδο που ορίζει το τρίγωνο Tr_i
      3.1.1. Δημιούργησε ένα τετράεδρο T_i με κορυφές την A και τις κορυφές του τριγώνου Tr_i.
      3.1.2. Υπολόγισε τον όγκο V_i του τετραέδρου T_i.
      3.1.3. Πρόσθεσε τον V_i στον αθροιστικά υπολογιζόμενο όγκο V του κυρτού πολυέδρου P.
4. Λάβε ως αποτέλεσμα τον όγκο V του κυρτού πολυέδρου P

Αν θεωρήσουμε ότι η μέτρηση του όγκου ενός τετραέδρου απαιτεί υπολογιστικό χρόνο ίσο με 1 μονάδα, τότε ο υπολογισμός του όγκου ενός κυρτού τριγωνισμένου πολυέδρου έχει πολυπλοκότητα ανάλογη με το πλήθος των τετραέδρων στα οποία έχει διαμεριστεί. Δηλαδή είναι ανάλογη με τον αριθμό των τριγώνων που έχουν προκύψει από τη διαδικασία τριγωνισμού. Έτσι, για τον υπολογισμό του όγκου ενός κυρτού περιβλήματος απαιτείται

πολυπλοκότητα $O(n)$, όπου n είναι ο αριθμός των τριγώνων του τριγωνισμένου κυρτού περιβλήματος.

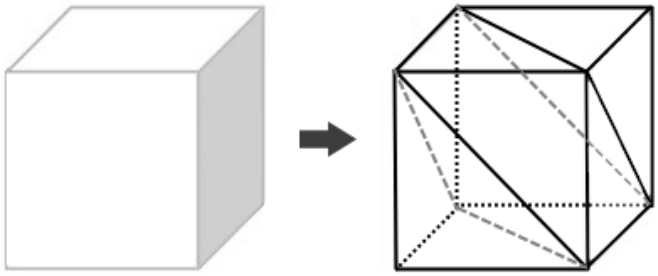
Η πολυπλοκότητα του αλγορίθμου μπορεί να εκφραστεί και σαν συνάρτηση των κορυφών, των ακμών ή των εδρών του, αν ληφθεί υπόψιν ότι οι κορυφές V, οι ακμές E και οι έδρες F ενός κυρτού πολυέδρου συνδέονται με τον τύπο

$$V + F - E = 2 \tag{9.2}$$

ενώ τα τρίγωνα του τριγωνισμένου κυρτού περιβλήματος προκύπτουν από τις έδρες του.

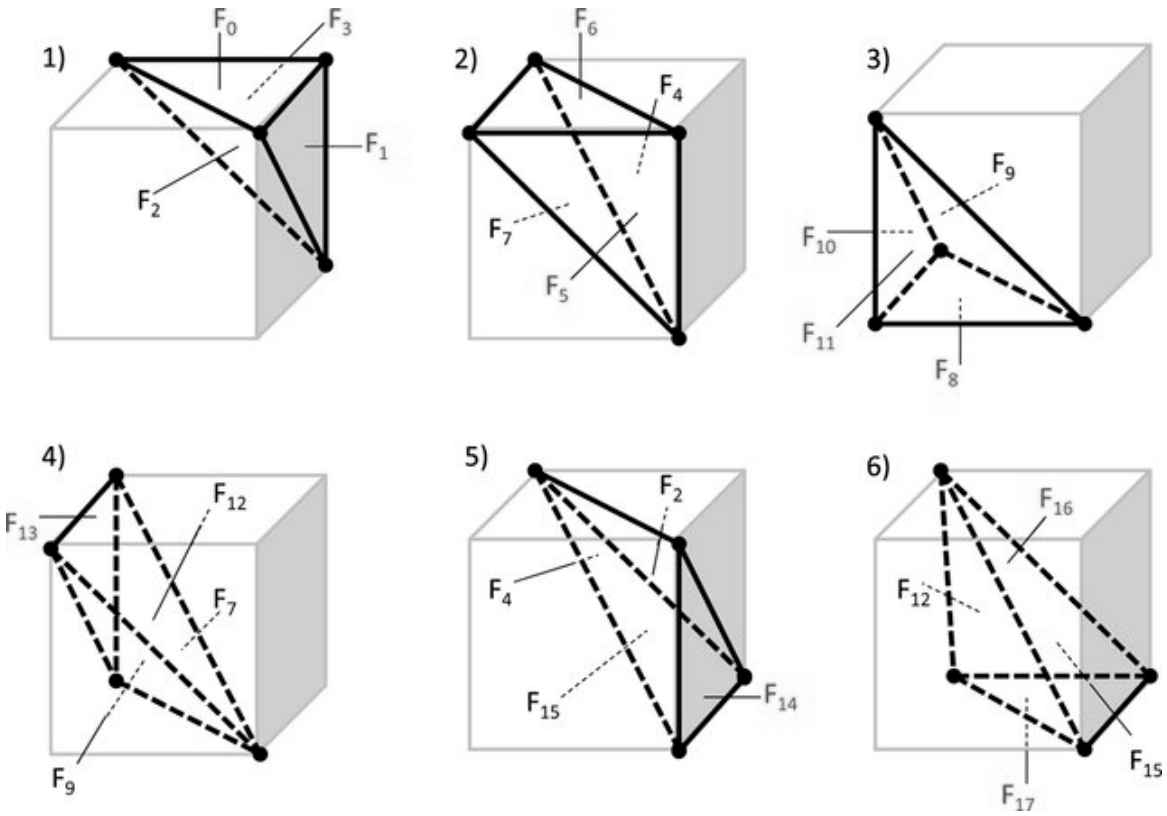
9.3 Παράδειγμα εφαρμογής του προτεινόμενου αλγόριθμου

Για την καλύτερη κατανόηση του προηγούμενου αλγορίθμου, παρουσιάζεται ένα παράδειγμα εφαρμογής του σε ένα απλό κυρτό πολύεδρο, τον κύβο. Ο αλγόριθμος εφαρμόζεται με τον ίδο τρόπο και σε πολυπλοκότερα κυρτά πολύεδρα, με περισσότερες κορυφές, περισσότερες έδρες ή πιο ιδιόμορφους σχηματισμούς. Έστω ο κύβος του Σχ. 9.6 πριν και μετά τον τριγωνισμό του.



Σχήμα 9.6: Τριγωνισμός του κυρτού πολυέδρου (κύβου)

Δημιουργούμε σταδιακά τα έξι τετράεδρα που φαίνονται στο Σχ. 9.7.



Σχήμα 9.7: Τα έξι τετράεδρα που προκύπτουν από τον διαμερισμό του κύβου

Τα τετράεδρα σχηματίζονται από τα ακόλουθα τρίγωνα και έχουν τους αντίστοιχους όγκους:

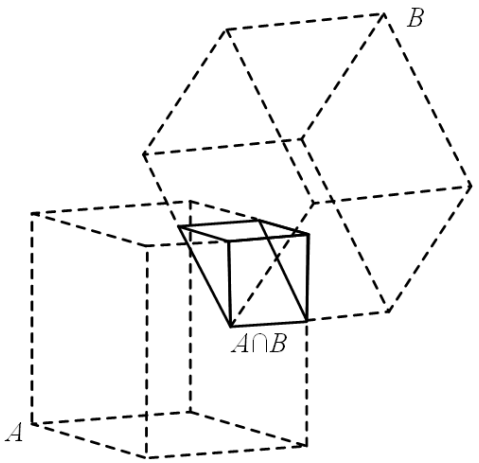
- Τετράεδρο 1, τρίγωνα F_0, F_1, F_2, F_3, όγκος V_1
- Τετράεδρο 2, τρίγωνα F_4, F_5, F_6, F_7, όγκος V_2
- Τετράεδρο 3, τρίγωνα F_8, F_9, F_{10}, F_{11}, όγκος V_3
- Τετράεδρο 4, τρίγωνα F_7, F_9, F_{12}, F_{13}, όγκος V_4
- Τετράεδρο 5, τρίγωνα F_2, F_4, F_{14}, F_{15}, όγκος V_5
- Τετράεδρο 6, τρίγωνα $F_{12}, F_{15}, F_{16}, F_{17}$, όγκος V_6

Υπολογίζοντας τον όγκο των τετραέδρων και αθροίζοντας τους υπολογιζόμενους όγκους, προκύπτει ο ζητούμενος όγκος, καθώς η ένωση αυτών των μη επικαλυπτόμενων τετραέδρων σχηματίζουν τον κύβο ο οποίος αποτελεί το κυρτό πολύεδρο για το συγκεκριμένο παράδειγμα.

$$V_{πολυεδρου} = V_1 + V_2 + V_3 + V_4 + V_5 + V_6$$

10. ΤΟΜΗ ΚΥΡΤΩΝ ΠΟΛΥΕΔΡΩΝ

Η κατανόηση της τομής δύο κυρτών πολύεδρων αποτελεί ένα θεμελιώδες πρόβλημα το οποίο συναντάται σε διάφορα πεδία όπως για παράδειγμα η γραφική υπολογιστών ή η ανίχνευση συγκρούσεων. Η τομή δύο κυρτών πολύεδρων αντιστοιχεί στο πεπερασμένο σχήμα του χώρου, το οποίο περικλείεται από ν επίπεδα πολυγωνικά σχήματα, εσωκλείεται ταυτόχρονα και στα δύο αρχικά κυρτά πολύεδρα. Με την προϋπόθεση ότι τα αρχικά πολύεδρα επικαλύπτονται, η τομή αποτελεί με τη σειρά της ένα κυρτό πολύεδρο (βλ. Σχ. 10.1).



Σχήμα 10.1: Τομή κυρτών πολυέδρων

Ενώ ο ορισμός της τομής δείχνει να είναι απλός, η εύρεση ενός αποδοτικού αλγορίθμου για τον υπολογισμό της αποτελεί ένα πολύπλοκο μαθηματικό πρόβλημα. Η δυσκολία έγκειται στην ιδιομορφία που πιθανώς να παρουσιάζει ο σχηματισμός των πολυέδρων καθώς και ο τρόπος με τον οποίο αυτά επικαλύπτονται. Σε ιδιάζουσες περιπτώσεις, η αποτελεσματικότητα των αλγορίθμων επιβραδύνεται, αυξάνοντας την πολυπλοκότητα και τον χρόνο εκτέλεσής τους.

Το γεγονός ότι η τομή δύο κυρτών πολύεδρων είναι επίσης ένα κυρτό πολύεδρο πηγάζει από το γεγονός ότι η τομή των κυρτών συνόλων είναι κυρτή. Έτσι, εάν τα P και Q είναι κυρτά πολύεδρα τότε η τομή τους $P \cap Q$

ορίζεται ως

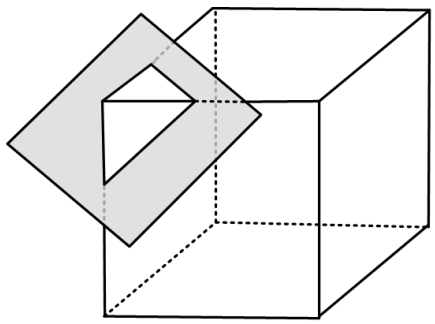
$$P \cap Q = x \mid x \in P \text{ και } x \in Q$$

Οπτικά, η τομή μπορεί να απεικονιστεί ως η επικαλυπτόμενη περιοχή όπου τα δύο πολύεδρα μοιράζονται κοινό χώρο. Αυτή η περιοχή διατηρεί τις ιδιότητες της κυρτότητας και ορίζεται από τα επίπεδα και τους ημιχώρους που συνέδεαν και τα δύο πολύεδρα.

10.1 Κατηγορίες αλγορίθμων υπολογισμού της τομής κυρτών πολυέδρων και προγενέστερες μελέτες

Υπάρχουν διάφορες κατηγορίες αλγορίθμων για την εύρεση της τομής των κυρτών πολυέδρων οι οποίες σχετίζονται τόσο με την υπολογιστική απόδοση όσο και με την γεωμετρική πολυπλοκότητα των πολυέδρων. Κάθε μία κατηγορία έχει πλεονεκτήματα και μειονεκτήματα ενώ η επιλογή του κατάλληλου αλγορίθμου εξαρτάται κυρίως από τα χαρακτηριστικά των πολυέδρων που εμπλέκονται, όπως το μέγεθος, την πολυπλοκότητά τους καθώς και τους απαιτούμενους υπολογιστικούς πόρους. Με την κατάλληλη επιλογή αλγορίθμου είναι δυνατός ο υπολογισμός της τομής των κυρτών πολύεδρων αποτελεσματικά και με ακρίβεια.

- **Σάρωση επιπέδων**: Αποτελεί μία τεχνική που περιλαμβάνει τον έλεγχο ενός ή περισσότερων επιπέδων και του τριδιάστατου χώρου που καταλαμβάνεται από τα πολύεδρα. Από τον έλεγχο αυτό προκύπτουν τομές μεταξύ των επιπέδων και του πολυέδρου οι οποίες στη συνέχεια μπορούν να χρησιμοποιηθούν συνδυαστικά για να σχηματίσουν το αποτέλεσμα δηλαδή το πολύεδρο που εκφράζει την τομή. Αυτή η μέθοδος θεωρείται ιδιαίτερα αποτελεσματική και είναι κατάλληλη για περιπτώσεις όπου τα πολύεδρα έχουν μεγάλο πλήθος όψεων (βλ. Σχ. 10.2).



Σχήμα 10.2: Επίπεδο το οποίο αποκόπτει τμήμα του κυρτού πολυέδρου

- **Σταδιακή δημιουργία**: Δημιουργεί το πολύεδρο τομής σταδιακά ξεκινώντας από μία μόνο όψη και προσθέτοντας σταδιακά νέες όψεις από ένα από τα πολύεδρα. Σε κάθε βήμα, η τομή με την τρέχουσα όψη υπολογίζεται και ενσωματώνεται στο υπάρχον πολύεδρο τομής. Αυτή η προσέγγιση επιτρέπει τη δυναμική ενημέρωση και είναι ιδιαίτερα αποτελεσματική, όταν τα πολύεδρα χαρακτηρίζονται ως μία ενιαία δομή κορυφών-ακμών-εδρών επιτρέποντας έτσι αποτελεσματικές ενημερώσεις και τροποποιήσεις.

- **Διαίρει και βασίλευε**: Διαιρεί το πρόβλημα σε πιο διαχειρίσιμα υποπροβλήματα. Κάθε πολύεδρο διαμερίζεται σε μικρότερα τμήματα και οι τομές υπολογίζονται ανεξάρτητα για κάθε τμήμα. Αυτά τα μερικά αποτελέσματα στη συνέχεια συνδυάζονται για να σχηματίσουν το τελικό τεμνόμενο πολύεδρο. Η συγκεκριμένη μέθοδος αξιοποιεί τη δύναμη της αναδρομικής επίλυσης προβλημάτων και της παράλληλης επεξεργασίας, καθιστώντας την κατάλληλη για πολύπλοκα πολύεδρα όπου ο άμεσος υπολογισμός θα ήταν υπολογιστικά απαιτητικός.

- **Γραμμικός προγραμματισμός**: Περιλαμβάνει τη διατύπωση του προβλήματος της τομής ως ένα σύνολο γραμμικών ανισοτήτων. Κάθε έδρα του πολύεδρου αναπαρίσταται ως γραμμικός περιορισμός και η λύση αυτών των περιορισμών δίνει το τεμνόμενο πολύεδρο. Αυτή η προσέγγιση είναι πολύ αποτελεσματική και μπορεί να χειριστεί ικανοποιητικά ζητήματα αριθμητικής σταθερότητας. Ο γραμμικός προγραμματισμός πλεονεκτεί, κυρίως, όταν τα πολύεδρα μπορούν να περιγραφούν χρησιμοποιώντας μεγάλο αριθμό γραμμικών περιορισμών παρέχοντας μία ακριβή και μαθηματικά αυστηρή λύση.

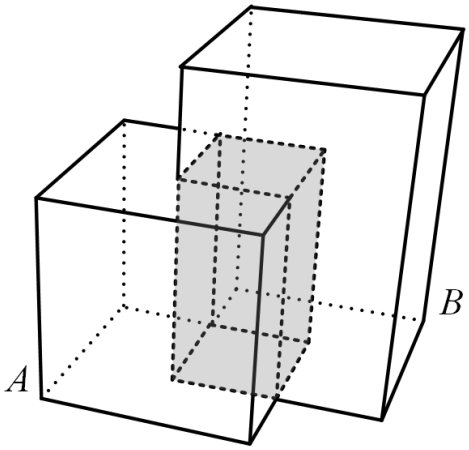
Αναπτύσσοντας το πρόβλημα σαν ένα γραμμικό πρόγραμμα τριών μεταβλητών, ο Megiddo [Meg82] και ο Dyer [Dye84] παρουσιάσαν αλγορίθμους γραμμικού χρόνου πολυπλοκότητας. Λοιπές βέλτιστες λύσεις για την εύρεση της τομής κυρτών πολυγώνων είχαν δοθεί από τους Shamos και Hoey [SH76] και από τον O'Rourke et al. [ORo+82]. Επιπλέον μελέτες σχετικές με το υπόβαθρο του προβλήματος της τομής πολυέδρων παρουσιάστηκαν από τους Edelsbrunner [Ede87], Mehlhorn [Meh84] και από τους Preparata και Shamos [PS84]. Το 1987 οι Chazelle και Dobkin [CD87] παρουσίασαν μία μέθοδο η οποία βασίζεται στον προϋπολογισμό των τριδιάστατων πολυέδρων προκειμένου να ελέγξουν εάν δύο πολύεδρα τέμνονται σε χρόνο $O(\log^3 n)$. Κάτι ανάλογο έγινε και από τους Dobkin και Kirkpatrick [DK83] οι οποίοι, χρησιμοποιώντας προϋπολογισμό, πέτυχαν την τομή δύο ανεξάρτητων πολυέδρων σε χρόνο $O(\log^2 n)$. Το 1990, οι Dobkin και Kirkpatrick [DK90] πρότειναν έναν αλγόριθμο ταχείας διερεύνησης ο οποίος χρησιμοποιεί την γραμμική ιεραρχική αναπαράσταση δύο πολυέδρων P και Q για να υπολογίσει την τομή τους σε χρόνο $O(\log |P| \, |Q|)$. Αυτό το επιτυγχάνουν διατηρώντας το πλησιέ-

στερο ζεύγος μεταξύ των υποσυνόλων των πολυέδρων P και Q καθώς οι αλγόριθμοι προχωρούν από άνω προς τα κάτω στην ιεραρχική αναπαράσταση. Λίγο αργότερα, το 1992, ο Bernard Chazelle [Cha92] παρουσίασε έναν αλγόριθμο για τη δημιουργία της τομής δύο κυρτών πολυέδρων. Ο συγκεκριμένος αλγόριθμος δεν χρησιμοποιεί πολύπλοκες δομές δεδομένων και φαίνεται να είναι ικανοποιητικός για πρακτικές εφαρμογές. Τέλος, μία μελέτη που περιλαμβάνει έναν αποδοτικό αλγόριθμο για την εύρεση της τομής δύο πολυέδρων, ένα εκ των οποίων είναι κυρτό, παρουσιάστηκε και από τους Mehlhorn και Simon [MS06] το 2006.

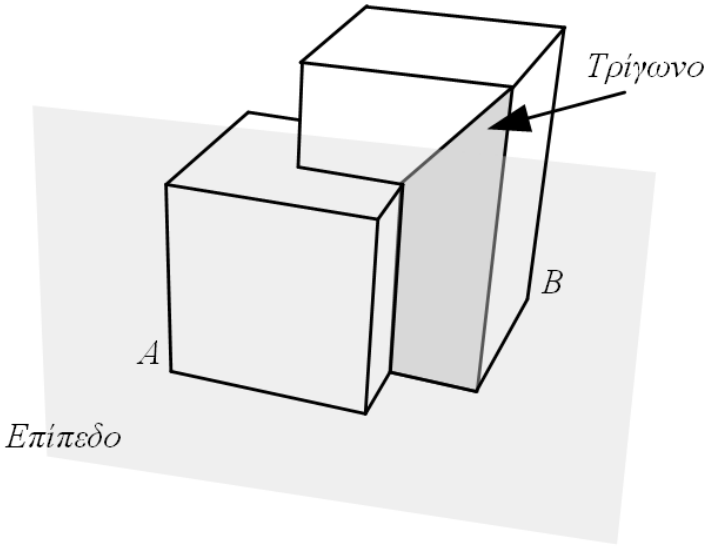
10.2 Προτεινόμενος αλγόριθμος υπολογισμού της τομής δύο κυρτών πολυέδρων

Οι Δρακόπουλος, Ματθές και Σγούρδος προτείνουν στην [DMS24] έναν αλγόριθμο υπολογισμού της τομής δύο κυρτών πολυέδρων. Η λογική του βασίζεται στην ίδια ιδέα που χρησιμοποιήθηκε και στους αλγορίθμους [MD19b], [MD22] και [MD23], όπου τα προς αποκοπή ευθύγραμμα τμήματα «εναγκαλίζονται» σταδιακά από την κυρτή περιοχή ή τον κυρτό όγκο προκειμένου να υποστούν αποκοπή. Σε κάθε βήμα του αλγόριθμου, μία μη προγενέστερα χρησιμοποιούμενη έδρα του πρώτου κυρτού πολυέδρου δημιουργεί ένα επίπεδο το οποίο διαχωρίζει τον τριδιάστατο χώρο σε δύο ημιχώρους. Το τμήμα του δεύτερου πολυέδρου που βρίσκεται στον συμπληρωματικό ημιχώρο σε σχέση με το πρώτο πολύεδρο απορρίπτεται κρατώντας μόνο την προβολή του στο επίπεδο ενώ το τμήμα που βρίσκεται στον ίδιο ημιχώρο διατηρείται και αποτελεί το δεύτερο πολύεδρο για το επόμενο βήμα του αλγορίθμου. Στο τέλος, το πολύεδρο που προκύπτει μετά την ολοκλήρωση όλων των βημάτων του αλγορίθμου, αποτελεί την τομή των δύο αρχικών πολυέδρων. Για να βρεθεί η τομή των κυρτών πολυέδρων απαιτείται το δεύτερο πολύεδρο να είναι τριγωνισμένο διότι οι έλεγχοι γίνονται μεταξύ του επιπέδου της κάθε έδρας του πρώτου και των τριγώνων του δεύτερου. Στην περίπτωση που υπάρχει τομή επιπέδου – τριγώνου και πρέπει να γίνει αποκοπή, το αποτέλεσμα πρέπει να είναι (αναλόγως την περίπτωση) ένα ή δύο νέα τρίγωνα

Έστωσαν κυρτά πολύεδρα A και B (βλ. Σχ. 10.3). Κάθε έδρα του πολυέδρου A, όπως έχει ήδη ειπωθεί, δημιουργεί ένα επίπεδο το οποίο διαμερίζει τον τριδιάστατο χώρο σε δύο ημιχώρους κάθε φορά. Για την εύρεση του τμήματος του B, που θα διατηρηθεί και θα αποτελέσει το προς επεξεργασία πολύεδρο στο επόμενο βήμα του αλγορίθμου, εξετάζονται ένα προς ένα τα τρίγωνα της τριγωνισμένης του μορφής (βλ. Σχ. 10.4).



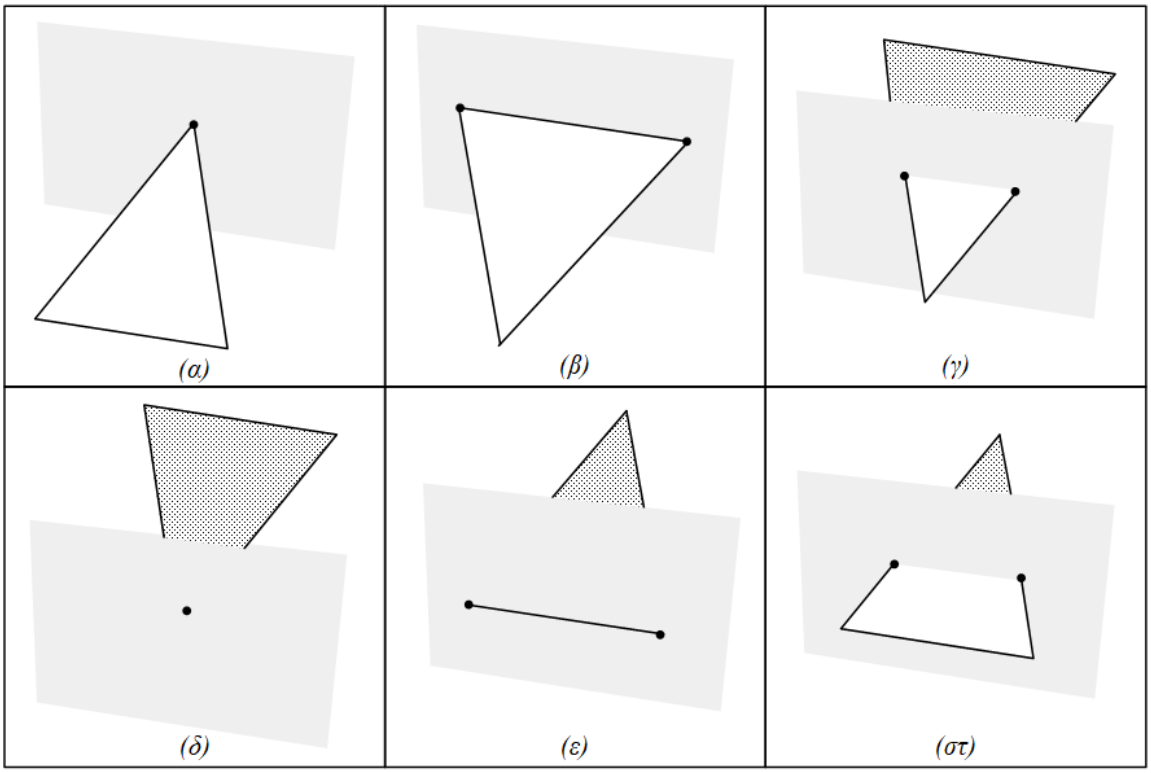
Σχήμα 10.3: Τα κυρτά πολύεδρα A και B και η τομή τους



Σχήμα 10.4: Το επίπεδο που βρίσκεται η κάθε έδρα του πολυέδρου A ελέγχεται σε σχέση με τα τρίγωνα του πολυέδρου B

Υπάρχουν δύο δυνατές περιπτώσεις για κάθε τρίγωνο του B: α) Είτε το τρίγωνο βρίσκεται αποκλειστικά σε έναν από τους δύο ημιχώρους, β) είτε το τρίγωνο τέμνει το τρέχον επίπεδο με κάποιο τρόπο. Στην περίπτωση που όλες οι κορυφές του τριγώνου βρίσκονται στον συμπληρωματικό ημιχώρο από αυτόν που βρίσκεται το πολύεδρο A τότε το τρίγωνο πρέπει να απορριφθεί οπότε ο αλγόριθμος διατηρεί απλώς την προβολή του στο επίπεδο E προκειμένου να μην χαλάσει η γεωμετρία της τομής που θα προκύψει στο τέλος. Αν όλες οι κορυφές του τριγώνου βρίσκονται στον ίδιο ημιχώρο με το A, τότε το τρίγωνο διατηρείται αυτούσιο και αποτελεί μέρος της τριγωνισμένης μορφής του τμήματος του πολυέδρου B.

Η περίπτωση ένα τρίγωνο να τέμνει το τρέχον επίπεδο διαχωρίζεται σε έξι υποπεριπτώσεις: Όταν η μία κορυφή του προς εξέταση τριγώνου βρίσκεται επάνω στο επίπεδο και οι άλλες δύο βρίσκονται στον ίδιο ημιχώρο με το πολύεδρο A, τότε το τρίγωνο αυτό διατηρείται ως έχει (βλ. Σχ. 10.5.α). Ομοίως, όταν δύο κορυφές του προς εξέταση τριγώνου βρίσκονται επάνω στο επίπεδο και η άλλη βρίσκεται στον ίδιο ημιχώρο με το πολύεδρο A, τότε το τρίγωνο επίσης διατηρείται (βλ. Σχ. 10.5.β). Όταν μία κορυφή του τρέχοντος τριγώνου βρίσκεται στο επίπεδο και οι άλλες δύο βρίσκονται στον συμπληρωματικό ημιχώρο με το πολύεδρο A, τότε το τρίγωνο απορρίπτεται (βλ. Σχ. 10.5.δ). Με τον ίδιο τρόπο, όταν δύο κορυφές του τρέχοντος τριγώνου βρίσκονται στο επίπεδο E και η άλλη βρίσκεται στον συμπληρωματικό ημιχώρο με το πολύεδρο A, τότε και αυτό το τρίγωνο απορρίπτεται (βλ. Σχ. 10.5.ε). Όταν μία κορυφή του τρέχοντος τριγώνου βρίσκεται στον ίδιο ημιχώρο με το πολύεδρο A και οι άλλες βρίσκονται στον συμπληρωματικό ημιχώρο με το πολύεδρο A, τότε διατηρείται το τρίγωνο που σχηματίζεται από την κορυφή αυτή και από τα δύο σημεία τομής των πλευρών του αρχικού τριγώνου με το επίπεδο (βλ. Σχ. 10.5.γ). Τέλος, όταν δύο κορυφές του τρέχοντος τριγώνου βρίσκονται στον ίδιο ημιχώρο με το πολύεδρο A και η άλλη βρίσκεται στον συμπληρωματικό ημιχώρο με το πολύεδρο A, τότε διατηρείται το τετράπλευρο που σχηματίζεται από τις κορυφές αυτές και από τα δύο σημεία τομής των πλευρών του αρχικού τριγώνου με το επίπεδο (βλ. Σχ. 10.5.στ).



Σχήμα 10.5: Όλες οι περιπτώσεις τομής του επιπέδου E του πολυέδρου A και του υπό εξέταση τριγώνου του πολυέδρου B

Η διαδικασία επαναλαμβάνεται μέχρις ότου έχουν εξεταστεί όλες οι έδρες (επίπεδα) του πολυέδρου A. Το τελικό πολύεδρο που προκύπτει μετά την ολοκλήρωση του αλγορίθμου αποτελεί την τομή των δύο αρχικών κυρτών επικαλυπτόμενων πολυέδρων το οποίο με τη σειρά του είναι κυρτό. Πρέπει να τονιστεί ότι ο αλγόριθμος εύρεσης της τομής δεν αστοχεί στην περίπτωση που τα δύο αρχικά πολύεδρα δεν επικαλύπτονται καθώς από το πρώτο κιόλας βήμα, έχουν απορριφθεί όλα τα τρίγωνα της τριγωνισμένης μορφής του B οπότε η εκτέλεση του αλγορίθμου τερματίζεται αναδεικνύοντας ότι τα δύο αρχικά πολύεδρα είναι μη επικαλυπτόμενα. Παρομοίως, στις ακραίες περιπτώσεις που η τομή των δύο αρχικών πολυέδρων είναι μία έδρα, μία ευθεία ή ένα σημείο, το αποτέλεσμα αναδεικνύεται μέσα από την ακολουθία της μεθοδολογίας εύρεσης της τομής.

Παρακάτω δίνεται ο αλγόριθμος υπολογισμού της τομής δύο κυρτών πολυέδρων με τη μορφή αναλυτικών βημάτων:

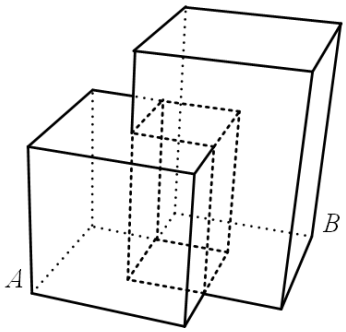
Αλγόριθμος υπολογισμού της τομής δύο κυρτών πολυέδρων

1. Λάβε ως είσοδο τα πολύεδρα A και B.

2. Θέσε το πολύεδρο επεξεργασίας T ίσο με το B.
3. Εφάρμοσε τριγωνισμό στις έδρες του T εφ'όσον χρειάζεται.
4. Για κάθε έδρα του πολυέδρου A
   4.1. Όρισε το επίπεδο E το οποίο διαμερίζει τον $\mathbb{R}^3$ σε δύο ημιχώρους
   4.2. Για κάθε τρίγωνο του T
      4.2.1. Αν οι κορυφές του τρέχοντος τριγώνου βρίσκονται όλες στον ίδιο ημιχώρο με το πολύεδρο A, τότε διατήρησε το τρίγωνο στο πολύεδρο T.
      4.2.2. Διαφορετικά, αν οι κορυφές του τρέχοντος τριγώνου βρίσκονται όλες στον συμπληρωματικό ημιχώρο με το πολύεδρο A, τότε απόρριψε το τρίγωνο και αντικατέστησέ το με την προβολή του στο επίπεδο E.
      4.2.3. Διαφορετικά, το επίπεδο E και το υπο εξέταση τρίγωνο τέμνονται με κάποιον τρόπο.
         4.2.3.1. Αν μία κορυφή του τρέχοντος τριγώνου βρίσκεται στο επίπεδο E και οι άλλες δύο βρίσκονται στον ίδιο ημιχώρο με το πολύεδρο A, τότε διατήρησε το τρίγωνο στο πολύεδρο T (βλ. Σχ. 10.5α).
         4.2.3.2. Αν δύο κορυφές του τρέχοντος τριγώνου βρίσκονται στο επίπεδο E και η άλλη βρίσκεται στον ίδιο ημιχώρο με το πολύεδρο A, τότε διατήρησε το τρίγωνο στο πολύεδρο T (βλ. Σχ. 10.5β).
         4.2.3.3. Αν μία κορυφή του τρέχοντος τριγώνου βρίσκεται στο επίπεδο E και οι άλλες δύο βρίσκονται στον συμπληρωματικό ημιχώρο με το πολύεδρο A, τότε απόρριψε το τρίγωνο (βλ. Σχ. 10.5δ).
         4.2.3.4. Αν δύο κορυφές του τρέχοντος τριγώνου βρίσκονται στο επίπεδο E και η άλλη βρίσκεται στον συμπληρωματικό ημιχώρο με το πολύεδρο A, τότε απόρριψε το τρίγωνο (βλ. Σχ. 10.5ε).
         4.2.3.5. Αν μία κορυφή του τρέχοντος τριγώνου βρίσκεται στον ίδιο ημιχώρο με το πολύεδρο A και οι άλλες βρίσκονται στον συμπληρωματικό ημιχώρο με το πολύεδρο A, τότε πρόσθεσε στο T το τρίγωνο που σχηματίζεται από την κορυφή αυτή και από τα δύο σημεία τομής των πλευρών του αρχικού τριγώνου με το επίπεδο E (βλ. Σχ. 10.5γ).
         4.2.3.6. Αν δύο κορυφές του τρέχοντος τριγώνου βρίσκονται στον ίδιο ημιχώρο με το πολύεδρο A και η άλλη βρίσκεται στον συμπληρωματικό ημιχώρο με το πολύεδρο A, τότε εφάρμοσε τριγωνισμό στο τετράπλευρο που σχηματίζεται από τις κορυφές αυτές και από τα δύο σημεία τομής των πλευρών του αρχικού τριγώνου με το επίπεδο E (βλ. Σχ. 10.5στ) και πρόσθεσε τα δύο τρίγωνα στο T.
5. Λάβε ως αποτέλεσμα το πολύεδρο T το οποίο αποτελεί την τομή των A και B.

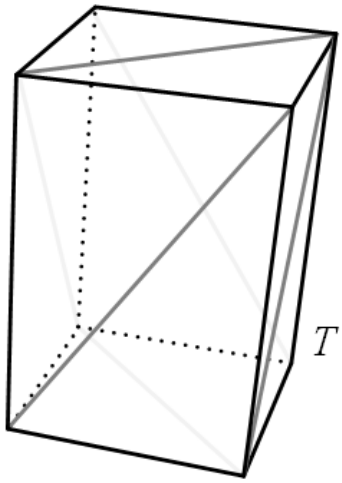
10.3 Παράδειγμα εφαρμογής του προτεινόμενου αλγορίθμου

Ας εξετάσουμε μία εφαρμογή του προτεινόμενου αλγορίθμου υπολογισμού της τομής δύο κυρτών πολυέδρων. Ενώ ο αλγόριθμος μπορεί να χρησιμοποιηθεί και σε περιπτώσεις όπου τα δύο κυρτά πολύεδρα έχουν πολύπλοκους σχηματισμούς ή ιδιόμορφες επικαλύψεις, εμείς, χάριν παραδείγματος, εξετάζομε την περίπτωση όπου τα πολύεδρα είναι κυβοειδή και αλληλοεπικαλυπτόμενα όπως στο Σχ. 10.6.



Σχήμα 10.6: Αλληλοεπικαλυπτόμενα κυβοειδή πολύεδρα A και B

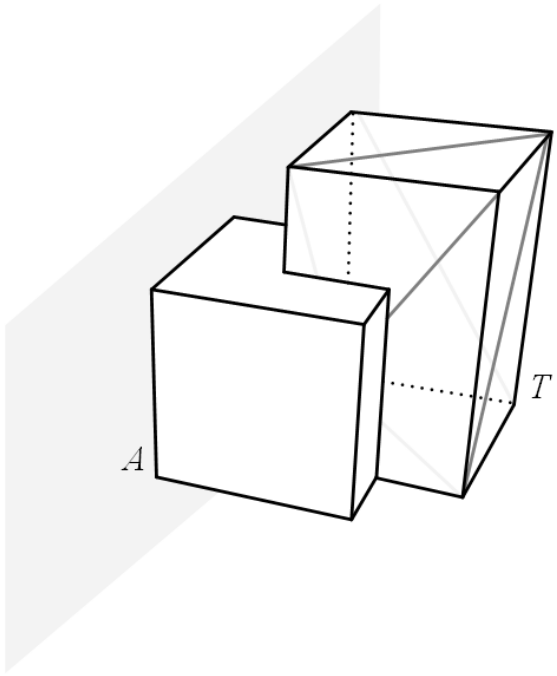
Θέτουμε το πολύεδρο επεξεργασίας T ίσο με το B και εφαρμόζουμε τριγωνισμό (βλ. Σχ. 10.7).



Σχήμα 10.7: Το τριγωνισμένο πολύεδρο επεξεργασίας T

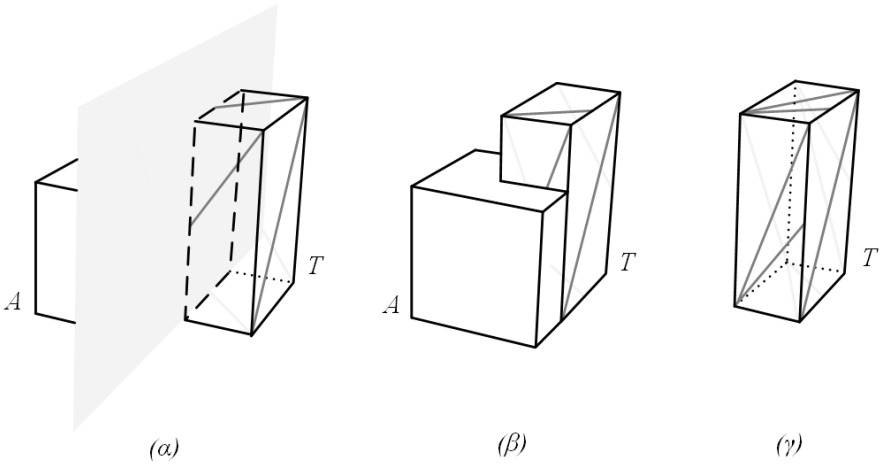
Χρησιμοποιώντας το επίπεδο της αριστερής έδρας του πολυέδρου A, παρατηρούμε ότι δεν επηρεάζεται η γεωμετρία του πολυέδρου T αφού όλα τα τρίγωνα του δεύτερου βρίσκονται στον ίδιο ημιχώρο με το A και κατά συνέπεια διατηρούνται ως έχουν (βλ. Σχ. 10.8). Το ίδιο συμβαίνει τόσο για την εμπρόσθια όσο και την κάτω έδρα

του A.



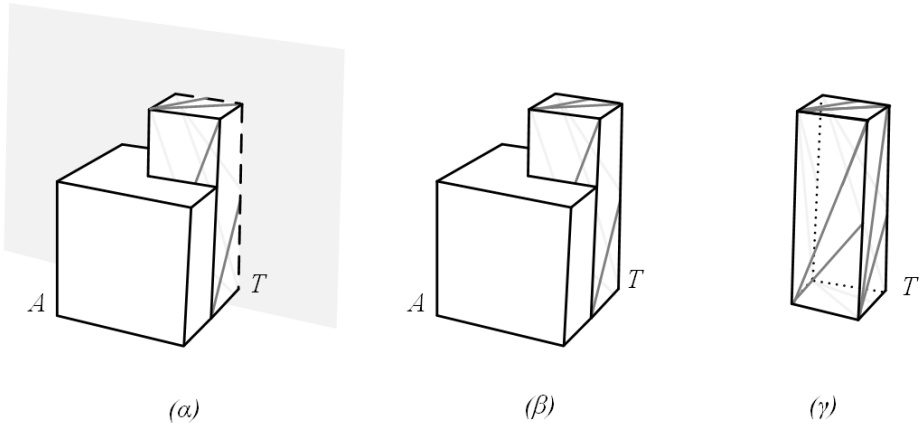
Σχήμα 10.8: Συγκρίνοντας τα τρίγωνα του T με το επίπεδο της αριστερής έδρας του A

Χρησιμοποιώντας τη δεξιά έδρα του πολυέδρου A (βλ. Σχ. 10.9α) διαπιστώνουμε ότι τα τρίγωνα της εμπρόσθιας, της οπίσθιας, της άνω και της κάτω έδρας του T τέμνουν το επίπεδό της οπότε πρέπει να βρεθούν τα σημεία τομής τους και να δημιουργηθούν νέα τρίγωνα ή τετράπλευρα όπου απαιτείται (βλ. Σχ. 10.9β). Όπου προκύπτουν νέα τετράπλευρα τότε αυτά τριγωνίζονται (βλ. Σχ. 10.9γ).



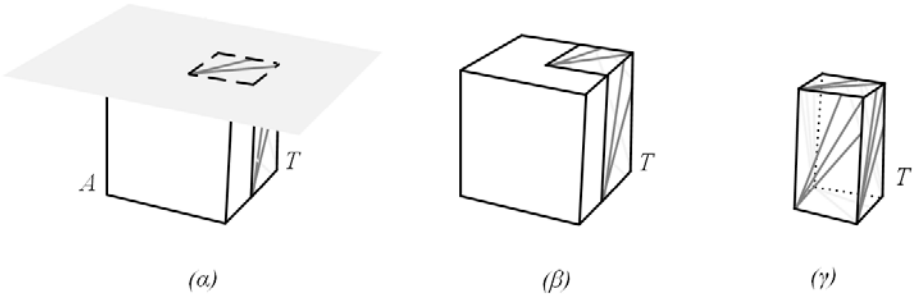
Σχήμα 10.9: (α) Χρησιμοποιώντας το επίπεδο της δεξιάς έδρας του πολυέδρου A, (β) Αποτέλεσμα μετά τον έλεγχο της τομής των τριγώνων του T με το τρέχον επίπεδο, (γ) Τριγωνισμός των νέων τετραπλεύρων

Προχωρώντας στο επόμενο βήμα και χρησιμοποιώντας την οπίσθια έδρα του πολυέδρου A (βλ. Σχ. 10.10α) διαπιστώνουμε ότι τα τρίγωνα της αριστερής, της δεξιάς, της άνω και της κάτω έδρας του T τέμνουν το επίπεδό της οπότε πρέπει να βρεθούν τα μεταξύ τους σημεία τομής και να δημιουργηθούν νέα τρίγωνα ή τετράπλευρα (βλ. Σχ. 10.10β). Ξανά, όπου προκύπτουν νέα τετράπλευρα τότε αυτά τριγωνίζονται (βλ. Σχ. 10.10γ).



Σχήμα 10.10: (α) Χρησιμοποιώντας το επίπεδο της οπίσθιας έδρας του πολυέδρου A, (β) Αποτέλεσμα μετά τον έλεγχο της τομής των τριγώνων του T με το τρέχον επίπεδο, (γ) Τριγωνισμός των νέων τετραπλεύρων

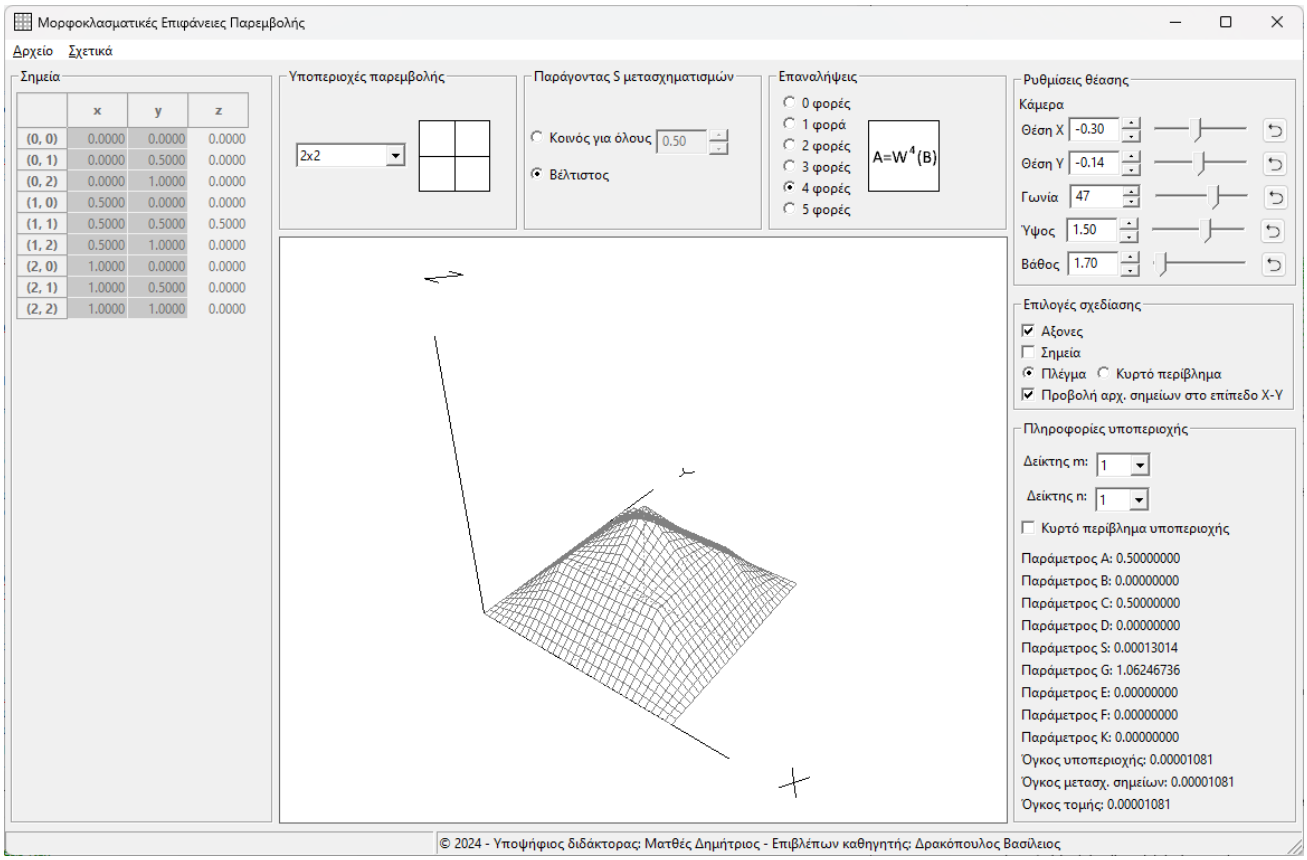
Τέλος, χρησιμοποιώντας την άνω έδρα του πολυέδρου A (βλ. Σχ. 10.11α) διαπιστώνουμε ότι τα τρίγωνα της αριστερής, της δεξιάς, της εμπρόσθιας και της πίσω έδρας του T τέμνουν το επίπεδό της οπότε πρέπει να βρεθούν τα μεταξύ τους σημεία τομής και να δημιουργηθούν νέα τρίγωνα ή τετράπλευρα (βλ. Σχ. 10.11β). Για τα νέα τετράπλευρα εφαρμόζεται τριγωνισμός. Η τελική μορφή του πολυέδρου T αποτελεί και την τομή μεταξύ των δύο κυρτών πολυέδρων A και B (βλ. Σχ. 10.11γ).



Σχήμα 10.11: (α) Χρησιμοποιώντας το επίπεδο της άνω έδρας του πολυέδρου A, (β) Αποτέλεσμα μετά τον έλεγχο της τομής των τριγώνων του T με το τρέχον επίπεδο, (γ) Μετά τον τριγωνισμό το πολύεδρο T αποτελεί την τομή μεταξύ των δύο κυρτών πολυέδρων A και B

11. ΕΦΑΡΜΟΓΗ «ΜΟΡΦΟΚΛΑΣΜΑΤΙΚΕΣ ΕΠΙΦΑΝΕΙΕΣ ΠΑΡΕΜΒΟΛΗΣ»

Για τη δημιουργία Μορφοκλασματικών Επιφανειών Παρεμβολής καθώς και την υλοποίηση όλης της πληροφορίας που περιγράφηκε στα κεφάλαια του Β' Μέρους, δημιουργήθηκε η εφαρμογή «Μορφοκλασματικές Επιφάνειες Παρεμβολής» (βλ. Σχ. 11.1). Μέσω αυτής ο χρήστης μπορεί να δημιουργήσει Μ.Ε.Π. μέσω απλών βημάτων καθώς και να αξιολογήσει το οπτικό αποτέλεσμά τους σε πραγματικό χρόνο.



Σχήμα 11.1: Η εφαρμογή «Μορφοκλασματικές Επιφάνειες Παρεμβολής»

Μπορεί κανείς να την μεταφορτώσει από: https://matthes.sites.sch.gr/fis/setup.exe

11.1 Τεχνικά χαρακτηριστικά - Απαιτήσεις - Εγκατάσταση

Η εφαρμογή δημιουργήθηκε σε γλώσσα C++ χρησιμοποιώντας τις βιβλιοθήκες CGAL, WxWidgets και OpenGL.

Η CGAL είναι λογισμικό ανοιχτού κώδικα το οποίο παρέχει εύκολη πρόσβαση σε αποδοτικούς και αξιόπιστους γεωμετρικούς αλγόριθμους υπό τη μορφή βιβλιοθήκης. Χρησιμοποιείται σε διάφορους τομείς που απαιτούνται γεωμετρικοί υπολογισμοί, όπως συστήματα γεωγραφικών πληροφοριών, σχεδιασμός με τη βοήθεια υπολογιστή, μοριακή βιολογία, ιατρική απεικόνιση, γραφικά υπολογιστών και ρομποτική. Η βιβλιοθήκη προσφέρει δομές δεδομένων και αλγόριθμους για τριγωνισμούς, διαγράμματα Voronoi, πράξεις Boolean σε πολύγωνα και πολύεδρα, επεξεργασία συνόλων σημείων, ρυθμίσεις καμπυλών, δημιουργία πλέγματος επιφάνειας και όγκου, επεξεργασία γεωμετρίας, σχήματα άλφα, αλγόριθμους κυρτού περιβλήματος, ανακατασκευή σχημάτων, κυτία AABB κ.α.

Η wxWidgets είναι μία βιβλιοθήκη που επιτρέπει στους προγραμματιστές να δημιουργήσουν Γραφική Διεπαφή Χρήστη (ή απλώς Γ.Δ.Χ.) για διάφορα Λ/Σ, όπως Windows, macOS, Linux καθώς και άλλες πλατφόρμες, σε μία ενιαία βάση κώδικα. Μπορεί να χρησιμοποιηθεί μαζί με δημοφιλείς γλώσσες προγραμματισμού, όπως οι C++, Python, Ruby, Lua, Perl, και, σε αντίθεση με άλλα αντίστοιχα λογισμικά δημιουργίας Γραφικής Διεπαφής Χρήστη, δίνει στα προγράμματα μία πραγματικά εγγενή με το Λ/Σ εμφάνιση και αίσθηση αφού χρησιμοποιεί απευθείας τις εντολές δημιουργίας Γ.Δ.Χ. του ίδιου του Λειτουργικού αντί να τις μιμείται. Είναι πλήρης, δωρεάν, ανοιχτού κώδικα και θεωρείται πλέον «ώριμη» αφού τα τελευταία χρόνια έχει υιοθετηθεί από μεγάλες εταιρείες κατασκευής λογισμικού.

Η OpenGL (Open Graphics Library) είναι μία ανοιχτού κώδικα βιβλιοθήκη που χρησιμοποιείται για τη δημιουργία και την απεικόνιση 2Δ και 3Δ γραφικών. Αρχικά αναπτύχθηκε από την εταιρεία Silicon Graphics Inc., γνωστή και ως SGI, τη δεκαετία του 1990 και εξελίχθηκε σε ένα από τα πιο δημοφιλή και ευρέως χρησιμοποιούμενα πρότυπα γραφικών προγραμματισμού. Παρέχει μία σειρά από λειτουργίες και δυνατότητες για την κατασκευή γραφικών περιβαλλόντων, εφαρμογών προσομοίωσης, παιχνιδιών και άλλων διαδραστικών εφαρμογών. Εκμεταλλεύεται την υπολογιστική ισχύ των καρτών γραφικών καθώς και των υπολογιστικών μονάδων για την απεικόνιση πολύπλοκων σκηνών και εφέ. μία από τις βασικές ιδέες πίσω από την OpenGL είναι η δυνατότητα αφαιρετικής προσέγγισης, δηλαδή η δυνατότητα να περιγράψει κανείς την επιθυμητή απεικόνιση χωρίς να ανησυχεί για τις λεπτομέρειες εκτέλεσης σε διάφορες πλατφόρμες ή κάρτες γραφικών. Αυτό την καθιστά ένα ισχυρό εργαλείο για τους προγραμματιστές που θέλουν να δημιουργήσουν πολύπλοκες γραφικές εφαρμογές και παιχνίδια.

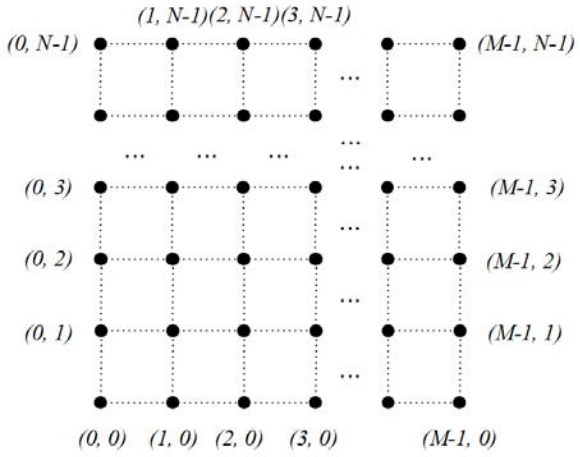
Οι ελάχιστες απαιτήσεις για τις «Μορφοκλασματικές Επιφάνειες Παρεμβολής» είναι:

- Λειτουργικό Σύστημα Windows 10
- Μνήμη 2GB RAM
- 40MB ελεύθερο χώρο στον δίσκο.

Η εγκατάστασή είναι αρκετά απλή. Εκτελώντας κανείς το πρόγραμμα της αυτοματοποιημένης εγκατάστασης, η εφαρμογή εγκαθίσταται στον υπολογιστή μας και είναι έτοιμη προς χρήση.

11.2 Λειτουργίες και δυνατότητες

Όπως προαναφέραμε, με την εν λόγω εφαρμογή ο χρήστης μπορεί να δημιουργήσει Μ.Ε.Π. χρησιμοποιώντας τις μεθόδους που περιγράφηκαν στα προηγούμενα κεφάλαια. Μετά την εκκίνηση, το βασικό παράθυρο της εφαρμογής είναι χωρισμένο σε περιοχές. Στο κέντρο, είναι το οπτικό αποτέλεσμα που προκύπτει ανάλογα με τις παραμέτρους που εισάγει ο χρήστης. Στα αριστερά φαίνονται τα σημεία παρεμβολής και οι συντεταγμένες τους. Ο χρήστης έχει τη δυνατότητα να αλλάξει μόνο την συντεταγμένη του βάθους (z) αφού οι άλλες δύο συντεταγμένες (x, y) λαμβάνουν συγκεκριμένες τιμές. Τα σημεία παρεμβολής είναι MxN, προσδιορίζονται από ένα ζεύγος ακεραίων και έχουν την μορφή (m, n), όπου $m \in [0 \dots M-1]$ και $n \in [0 \dots N-1]$ (βλ. Σχ. 11.2).



Σχήμα 11.2: Ο προσδιορισμός των σημείων παρεμβολής

Πέρα από τα σημεία παρεμβολής, ο χρήστης μπορεί να επιλέξει το πλήθος των υποπεριοχών παρεμβολής. Οι υποπεριοχές κυμαίνονται από 2×2 έως 10×10.

Μία άλλη επιλογή αφορά τις τιμές των ελεύθερων παραμέτρων S που σχετίζονται με την κατακόρυφη κλιμάκωση, αν θα είναι συγκεκριμένες για κάθε υποπεριοχή μετά από έναν μετασχηματισμό ή αν θα υπολογίζονται

αυτόματα από την εφαρμογή ανά περίπτωση.

Επιπλέον, ο χρήστης μπορεί να επιλέξει το πλήθος των επαναλήψεων των μετασχηματισμών το οποίο μπορεί να είναι από 0 έως 5 φορές. Έχοντας επιλεγμένη την τιμή «0 φορές», μπορεί να επεξεργαστεί τα σημεία παρεμβολής και δουλέψει με την αρχική επιφάνεια. Για μεγαλύτερες από μηδέν φορές, τα σημεία παρεμβολής δεν επιδέχονται επεξεργασία και ορατό είναι μόνο το αποτέλεσμα των επαναλήψεων σε πραγματικό χρόνο.

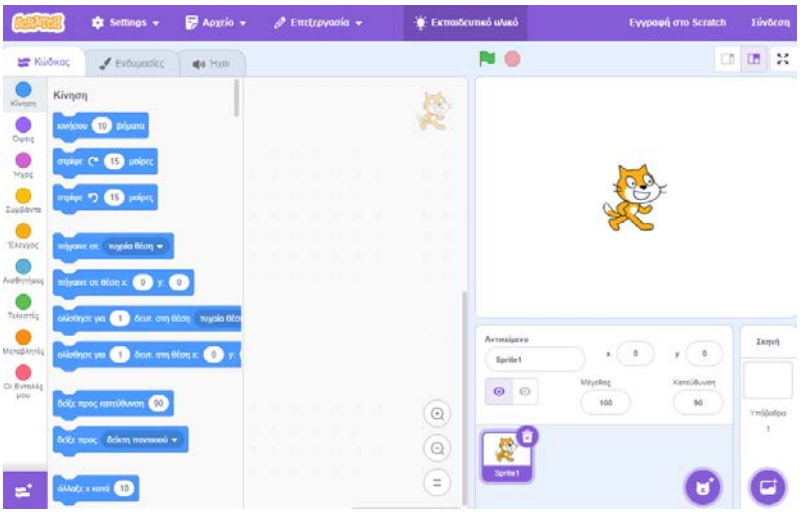
Στην περιοχή «Ρυθμίσεις θέασης» δύναται να αλλάξει η θέση της κάμερας ως προς τους άξονες X, Y, η γωνία, το ύψος και το βάθος αυτής. Στην περιοχή «Επιλογές σχεδίασης», μπορεί επιλεχθεί, αν θα εμφανίζονται οι άξονες X, Y, Z, αν θα εμφανίζονται τα σημεία παρεμβολής κατά τη σχεδίαση των Μ.Ε.Π., αν ο τρόπος σχεδιασμού θα είναι πλέγμα ή κυρτό περίβλημα και αν θα γίνεται προβολή των σημείων παρεμβολής στο επίπεδο X-Y. Τέλος, στην περιοχή «Πληροφορίες υποπεριοχής», εμφανίζονται οι τιμές των παραμέτρων $A, B, C, D, S, G, E, F, K$ για την επιλεγμένη υποπεριοχή μετά τους μετασχηματισμούς καθώς και οι όγκοι υποπεριοχής, μετασχηματισμένων σημείων και τομής τους.

12. ΣΥΜΠΕΡΑΣΜΑΤΑ-ΠΡΟΟΠΤΙΚΕΣ

Στην παρούσα διατριβή εστιάσαμε σε δύο αντικείμενα: στην αποκοπή ευθυγράμμων τμημάτων και στη διδιάστατη μορφοκλασματική παρεμβολή.

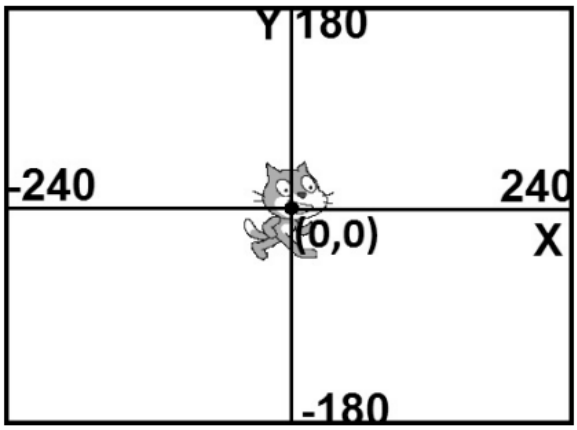
Όσον αφορά την αποκοπή ευθυγράμμων τμημάτων, αυτή χωρίστηκε σε τρεις κατηγορίες: α) Όταν το παράθυρο αποκοπής είναι ορθογώνιο παραλληλόγραμμο (2Δ), β) όταν το παράθυρο αποκοπής είναι κυρτό πολύγωνο (2Δ) και γ) όταν ο όγκος θέασης είναι ορθογώνιος (3Δ). Σε κάθε μία από τις κατηγορίες αυτές υλοποιήσαμε και παρουσιάσαμε τους αντιπροσωπευτικότερους αλγορίθμους ενώ, επιπλέον, προτείναμε και έναν νέο αλγόριθμο αποκοπής ανά περίπτωση. Εκτός από την υλοποίηση και την παρουσίαση, πραγματοποιήθηκε και αξιολόγηση όλων των αλγορίθμων μεταξύ τους με τη βοήθεια μίας διαδικτυακής εφαρμογής που δημιουργήθηκε για τον σκοπό αυτό. Η εφαρμογή είναι βασισμένη στην HTML και την Javascript και είναι προσβάσιμη από τη διεύθυνση: https://matthes.sites.sch.gr/clipping. Η διαδικασία που ακολουθήθηκε κατά την αξιολόγηση ήταν η εξής: Κατά την εκτέλεση κάθε αλγορίθμου αποκοπτόταν ένα μεγάλο πλήθος αυθαιρέτων ευθυγράμμων τμημάτων. Κάθε αλγόριθμος εκτελείτω δέκα φορές και ο χρόνος που χρειαζόταν για να αποκόψει και να σχεδιάσει τα ευθύγραμμα τμήματα κάθε φορά, καταγραφόταν. Στο τέλος, ο μέσος χρόνος των δέκα εκτελέσεων αποτέλεσε το κριτήριο αξιολόγησης του κάθε αλγορίθμου ως προς την ταχύτητα. Όλες οι μετρήσεις πραγματοποιήθηκαν με χρήση του ίδιου υπολογιστικού συστήματος. Επιπλέον της εφαρμογής, και προκειμένου να διαπιστωθεί αν τα αριθμητικά αποτελέσματα που προέκυψαν είναι έγκυρα και στατιστικώς ορθά, χρησιμοποιήθηκε η δοκιμασία αθροίσματος κατάταξης Wilcoxon (Wilcoxon Rank Sum Test). Τα τελικά αποτελέσματα δείχνουν ότι οι προτεινόμενοι νέοι αλγόριθμοι αποκοπής είναι ταχύτεροι από τους κλασικούς-αντιπροσωπευτικούς της κάθε κατηγορίας.

Όπως αναφέρθηκε στην ενότητα 2.9, η αποκοπή ευθυγράμμων τμημάτων χρειάστηκε να εφαρμοστεί στα πλαίσια μίας ερευνητικής εργασίας (project). Η ερευνητική εργασία αυτή πραγματοποιήθηκε σε ένα σχολείο της Δευτεροβάθμιας Εκπαίδευσης, συγκεκριμένα σε ένα Ημερήσιο Γενικό Λύκειο, χρησιμοποιώντας το προγραμματιστικό περιβάλλον Scratch (βλ. Σχ. 12.1).



Σχήμα 12.1: Το προγραμματιστικό περιβάλλον Scratch 3.0

Το Scratch διαθέτει μία οθόνη σταθερών διαστάσεων (480×360 εικονοστοιχεία) για την απεικόνιση αντικειμένων η οποία ονομάζεται Σκηνή (βλ. Σχ. 12.2). Η Σκηνή δεν επιτρέπει στα αντικείμενα του Scratch να βγουν εκτός των ορίων της. Το ίδιο συμβαίνει και με την σχεδίαση των γραφικών. Επειδή πραγματοποιείται με χρήση αντικειμένων, περιορίζεται εντός των ορίων της Σκηνής. Οι μαθητές που έλαβαν μέρος στην ερευνητική εργασία αναρωτήθηκαν πως θα μπορούσαν να ξεπεραστούν τα όρια αυτά και το αποτέλεσμα να αποκόπτεται εντός της θύρας θέασης.



Σχήμα 12.2: Οι συντεταγμένες και τα όρια της Σκηνής του Scratch

Όλοι οι σχετικοί με την αποκοπή αλγόριθμοι που δημιουργήθηκαν και προτάθηκαν δίνουν εύκολη λύση στο πρόβλημα αυτό. Επιπλέον, έχουν ένα κοινό χαρακτηριστικό στον τρόπο λειτουργίας τους. Σε όλες τις περιπτώσεις αποκοπής ευθυγράμμων τμημάτων, είτε στις δύο είτε στις τρεις διαστάσεις, εφ' όσον η περιοχή ή ο όγκος αποκοπής είναι κυρτός, ο εκάστοτε αλγόριθμος εξετάζει το σύνορο αποκοπής «εναγκαλιζόμενος» το ευθύγραμμο τμήμα

και αποκόπτοντάς το σταδιακά. Η διαπίστωση αυτή οδήγησε την έρευνα στο δεύτερο αντικείμενο της διατριβής, τη διδιάστατη μορφοκλασματική παρεμβολή. Εκεί, η λογική του εναγκαλισμού βρίσκει απόλυτη εφαρμογή στην κατασκευή Μορφοκλασματικών Επιφανειών Παρεμβολής (Μ.Ε.Π.).

Κατά την μορφοκλασματική παρεμβολή για την κατασκευή Μ.Ε.Π. επιχειρείται η δημιουργία νέων σημείων δεδομένων εντός του εύρους ενός διακριτού συνόλου γνωστών σημείων δεδομένων. Σε αντίθεση με τις παραδοσιακές μεθόδους παρεμβολής που χρησιμοποιούν συναρτήσεις πολυωνύμων ή καμπύλες spline, η μορφοκλασματική παρεμβολή χρησιμοποιεί Επαναλαμβανόμενα Συστήματα Συναρτήσεων (Ε.Σ.Σ.) για τη δημιουργία αυτοόμοιων δομών μέσω της επαναλαμβανόμενης εφαρμογής καταλλήλως ορισμένων συναρτήσεων. Στους μετασχηματισμούς αυτούς, ο «συντελεστής κατακόρυφης κλιμάκωσης», γνωστός και ως «παράμετρος συσταλτικότητας», ο οποίος συμβολίζεται με S, είναι ελεύθερος δηλαδή δεν μπορεί να υπολογιστεί άμεσα. Η εύρεση της βέλτιστης τιμής του S αποτελεί ένα σημαντικό ζήτημα το οποίο τις περισσότερες φορές επιλύεται προσεγγιστικά χρησιμοποιώντας αλγεβρικές ή γεωμετρικές μεθόδους.

Ένας τρόπος υπολογισμού της βέλτιστης τιμής του συντελεστή κατακόρυφης κλιμάκωσης προτάθηκε στην παρούσα διατριβή με το όνομα «Μέθοδος των Κυρτών Περιβλημάτων (Μ.Κ.Π.)». Βασίζεται στην χρήση περιβαλλόντων όγκων και την ελαχιστοποίηση της συμμετρικής διαφοράς τους έτσι ώστε το μετασχηματισμένο σύνολο να αποτελεί μία όσο το δυνατόν καλύτερη προσέγγιση του αρχικού συνόλου. Για την υλοποίηση της Μ.Κ.Π. απαιτούνται: i) ο υπολογισμός του κυρτού περιβλήματος ενός συνόλου σημείων, ii) ο υπολογισμός του όγκου ενός κυρτού περιβλήματος και iii) ο υπολογισμός της τομής δύο κυρτών περιβλημάτων. Στην περίπτωση (i), ο υπολογισμός του κυρτού περιβλήματος ενός συνόλου σημείων μπορεί να υλοποιηθεί αποτελεσματικά με τον Αλγόριθμο Ταχέως Περιβλήματος. Για την υλοποίηση των (ii) και (iii) προτείναμε μεθοδολογίες και αλγορίθμους οι οποίοι υπολογίζουν αποδοτικά τόσο τον όγκο ενός κυρτού περιβλήματος όσο και την τομή δύο κυρτών περιβλημάτων. Πιο συγκεκριμένα, παρουσιάσαμε μία μεθοδολογία για την εύρεση του όγκου ενός κυρτού πολυέδρου δια μέσω διαμερισμού του σε τετράεδρα καθώς και έναν αλγόριθμο υπολογισμού της τομής δύο κυρτών πολυέδρων εφ' όσον το ένα εκ των δύο πολύεδρων είναι τριγωνισμένο.

Τέλος, δημιουργήσαμε μία εφαρμογή με το όνομα «Μορφοκλασματικές Επιφάνειες Παρεμβολής» μέσω της οποίας υλοποιήθηκαν οι προτεινόμενες μεθοδολογίες και αλγόριθμοι σχετικοί τις Μ.Ε.Π. Η εν λόγω εφαρμογή δέχεται ως είσοδο τα δεδομένα σημεία της παρεμβολής και θέτοντας τις κατάλληλες παραμέτρους απεικονίζει σε πραγματικό χρόνο τα αντίστοιχα αποτελέσματα τα οποία μας βοηθούν στην καλύτερη κατανόηση των Μ.Ε.Π. αλλά και στον τρόπο δημιουργίας τους.

Στα άμεσα μελλοντικά σχέδια του συγγραφέα συγκαταλέγεται η δημοσίευση των προτεινόμενων αδημοσίευτων αλγορίθμων καθώς και η συνέχιση της έρευνας σχετική με την Γραφική Υπολογιστών και τα μορφοκλάσματα.

Εν κατακλείδι, θεωρούμε ότι αποτελέσματα που προέκυψαν, τόσο στο πεδίο της αποκοπής όσο και σε εκείνο της διδιάστατης μορφοκλασματικής παρεμβολής, δημιουργούν προοπτικές για μελλοντικούς ερευνητές να επεκτείνουν περαιτέρω την παρούσα έρευνα. Ως προς την αποκοπή, η επέκταση αυτή θα μπορούσε πραγματοποιηθεί σε διάφορους τομείς της Γραφικής Υπολογιστών, όπως η αποκοπή πολυγώνων, ενώ στην διδιάστατη μορφοκλασματική παρεμβολή θα μπορούσε να επεκταθεί σε τομείς, όπως η μορφοκλασματική συμπίεση, δηλαδή συμπίεσης με απώλειες για ψηφιακές εικόνες βασισμένες σε μορφοκλάσματα.

ΞΕΝΟΓΛΩΣΣΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

[AS91] R. Andreev and Elena Sofianska. “New algorithm for two-dimensional line clipping”. In: *Comput. Graph.* 15.4 (1991), pp. 519–526. DOI: 10.1016/0097-8493(91)90051-I.

[Aur91] Franz Aurenhammer. “Voronoi diagrams—a survey of a fundamental geometric data structure”. In: *ACM Comput. Surv.* 23.3 (1991), pp. 345–405. ISSN: 0360-0300. DOI: 10.1145/116873.116880. URL: https://doi.org/10.1145/116873.116880.

[BDD06] P. Bouboulis, Leoni Dalla, and V. Drakopoulos. “Construction of recurrent bivariate fractal interpolation surfaces and computation of their box-counting dimension”. In: *Journal of Approximation Theory* 141.2 (2006), pp. 99–117. ISSN: 0021-9045. DOI: https://doi.org/10.1016/j.jat.2006.01.006. URL: https://www.sciencedirect.com/science/article/pii/S0021904506000207.

[BDH96] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. “The quickhull algorithm for convex hulls”. In: *ACM Trans. Math. Softw.* 22 (1996), pp. 469–483. URL: https://api.semanticscholar.org/CorpusID:9793096.

[Bro79] Kevin Q. Brown. “Voronoi diagrams from convex hulls”. In: *Information Processing Letters* 9.5 (1979), pp. 223–228. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(79)90074-7. URL: https://www.sciencedirect.com/science/article/pii/0020019079900747.

[BT93] Jean-Daniel Boissonnat and Monique Teillaud. “On the randomized construction of the Delaunay tree”. In: *Theoretical Computer Science* 112.2 (1993), pp. 339–354. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(93)90024-N. URL: https://www.sciencedirect.com/science/article/pii/030439759390024N.

[CB78] M. Cyrus and J. Beck. “Generalized two- and three-dimensional clipping”. In: *Comput. Graph.* 3 (1978), pp. 23–28. DOI: 10.1016/0097-8493(78)90021-3.

[CD87] B. Chazelle and D. P. Dopkin. “Intersection of convex objects in two and three dimensions”. In: *Journal of the ACM* 34.1 (1987), pp. 1–27.

[Cha92] Bernard Chazelle. “An Optimal Algorithm for Intersecting Three-Dimensional Convex Polyhedra”. In: *SIAM Journal on Computing* 21.4 (1992), pp. 671–696. DOI: 10.1137/0221041. eprint: https://doi.org/10.1137/0221041. URL: https://doi.org/10.1137/0221041.

[CK70] Donald R. Chand and Sham S. Kapur. “An Algorithm for Convex Polytopes”. In: *J. ACM* 17 (1970), pp. 78–86. URL: https://api.semanticscholar.org/CorpusID:8366936.

[Com06] P. Comninos. *Mathematical and Computer Programming Techniques for Computer Graphics*. London: Springer-Verlag, 2006. DOI: 10.1007/978-1-84628-292-8.

[CS89] K. L. Clarkson and P. W. Shor. “Applications of random sampling in computational geometry, II”. In: *Discrete Comput. Geom.* 4 (1989), pp. 387–421.

[Dal02] L. Dalla. “Bivariate fractal interpolation functions on grids”. In: *Fractals* 1 (Mar. 2002). DOI: 10.1142/S0218348X02000951.

[Day92] J. D. Day. “An algorithm for clipping lines in object and image space”. In: *Comput. Graph.* 16.4 (1992), pp. 421–426. DOI: 10.1016/0097-8493(92)90029-U.

[Dim15] S. C. Dimri. “A simple and efficient algorithm for line and polygon clipping in 2-D computer graphics”. In: *International Journal of Computer Applications* 127.3 (2015), pp. 31–34.

[DJ94] Marie-Pierre Dubuisson and Anil K. Jain. “A modified Hausdorff distance for object matching”. In: *Proceedings of 12th International Conference on Pattern Recognition* 1 (1994), 566–568 vol.1. URL: https://api.semanticscholar.org/CorpusID:46947070.

[DK83] D. Dobkin and D. Kirkpatrick. “Fast detection of polyhedral intersection”. In: *Theoretical Computer Science* 27.3 (1983), pp. 241–253.

[DK90] D. Dobkin and D. Kirkpatrick. “Determining the separation of preprocessed polyhedra – a unified approach”. In: *Automata, Languages and Programming* (1990), pp. 400–413.

[DM20] Vasileios Drakopoulos and Polychronis Manousopoulos. “On Non-Tensor Product Bivariate Fractal Interpolation Surfaces on Rectangular Grids”. In: *Mathematics* 8.4 (2020). ISSN: 2227-7390. DOI: 10.3390/math8040525. URL: https://www.mdpi.com/2227-7390/8/4/525.

[DMS24] Vasileios Drakopoulos, Dimitrios Matthes, and Dimitrios Sgourdos. “Detecting and Finding the Intersection of Two Convex Polyhedra”. In: (2024).

[Dra+23] Vasileios Drakopoulos, Dimitrios Matthes, Dimitrios Sgourdos, and Nallapu Vijender. “Parameter Identification of Bivariate Fractal Interpolation Surfaces by Using Convex Hulls”. In: *Mathematics* 11.13 (2023). ISSN: 2227-7390. DOI: 10.3390/math11132850. URL: https://www.mdpi.com/2227-7390/11/13/2850.

[Dye84] M. E. Dyer. “Linear Time Algorithms for Two- and Three-Variable Linear Programs”. In: *SIAM Journal on Computing* 13.1 (1984), pp. 31–45. DOI: 10.1137/0213003. eprint: https://doi.org/10.1137/0213003. URL: https://doi.org/10.1137/0213003.

[Ede87] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Monographs in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 1987. ISBN: 9783540137221. URL: https://books.google.gr/books?id=mxugg47mzK4C.

[ERA19] M. Elliriki, C. Reddy, and K. Anand. “An efficient line clipping algorithm in 2D space”. In: *Int. Arab J. of Info. Tech.* 16.5 (2019), pp. 798–807.

[ES92] H. Edelsbrunner and N. R. Shah. “Incremental topological flipping works for regular triangulations”. In: *Proceedings of the Eighth Annual Symposium on Computational Geometry*. SCG ’92. Berlin, Germany: Association for Computing Machinery, 1992, pp. 43–52. ISBN: 0897915178. DOI: 10.1145/142675.142688. URL: https://doi.org/10.1145/142675.142688.

[Fen08] Zhigang Feng. “Variation and Minkowski dimension of fractal interpolation surface”. In: *Journal of Mathematical Analysis and Applications* 345.1 (2008), pp. 322–334. ISSN: 0022-247X. DOI: https://doi.org/10.1016/j.jmaa.2008.03.075. URL: https://www.sciencedirect.com/science/article/pii/S0022247X08003296.

[FK93] Wm. Randolph Franklin and Mohan S. Kankanhalli. “Volumes from overlaying 3-D triangulations in parallel”. In: *Advances in Spatial Databases*. Ed. by David Abel and Beng Chin Ooi. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 477–489. ISBN: 978-3-540-47765-5.

[Fol+90] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics Principles and Practice*. 2nd. Reading, MA: Addison-Wesley, 1990.

[GH93] J.S. Geronimo and D. Hardin. “Fractal Interpolation Surfaces and a Related 2-D Multiresolution Analysis”. In: *Journal of Mathematical Analysis and Applications* 176.2 (1993), pp. 561–586. ISSN: 0022-247X. DOI: https://doi.org/10.1006/jmaa.1993.1232. URL: https://www.sciencedirect.com/science/article/pii/S0022247X83712321.

[GH98] G. Greiner and K. Hormann. “Efficient clipping of arbitrary polygons”. In: *ACM Transactions on Graphics* 17 (1998), pp. 71–83. DOI: 10.1145/274363.274364.

[Grü63] Branko Grünbaum. “Measures of symmetry for convex sets”. In: 1963. URL: https://api.semanticscholar.org/CorpusID:117201662.

[HBC14] D. Hearn, M. Pauline Baker, and W. R. Carithers. *Computer Graphics with Open GL*. 4th. Edinburgh Gate, Harlow, Essex CM20 2JE: Pearson Education Limited, 2014.

[Hua10] Wenjun Huang. “The Line Clipping Algorithm Basing on Affine Transformation”. In: *Intelligent Information Management* 2.6 (2010), 2029, 6 pages. DOI: 10.4236/iim.2010.26046.

[HW09] Wenjun Huang and Wangyong. “A Novel Algorithm for Line Clipping”. In: *2009 International Conference on Computational Intelligence and Software Engineering*. 2009, pp. 1–5. DOI: 10.1109/CISE.2009.5366550.

[Joe91] Barry Joe. “Construction of three-dimensional Delaunay triangulations using local transformations”. In: *Computer Aided Geometric Design* 8.2 (1991), pp. 123–142. ISSN: 0167-8396. DOI: https://doi.org/10.1016/0167-8396(91)90038-D. URL: https://www.sciencedirect.com/science/article/pii/016783969190038D.

[Kat79] Victor J. Katz. “The History of Stokes’ Theorem”. In: *Mathematics Magazine* 52.3 (1979), pp. 146–156.

[Kol94] I. Kolingerová. “3D-line clipping algorithms — a comparative study”. In: *The Visual Computer* 11 (1994), pp. 96–104. DOI: https://doi.org/10.1007/BF01889980.

[KP05] R. Kobes and A. J. Penner. “Nonlinear fractal interpolating functions of one and two variables”. In: *Fractals* 13.02 (2005), pp. 179–186. DOI: 10.1142/S0218348X05002817. eprint: https://doi.org/10.1142/S0218348X05002817. URL: https://doi.org/10.1142/S0218348X05002817.

[KWC12] S. R. Kodituwakku, K. R. Wijeweera, and M. A. P. Chamikara. “An efficient line clipping algorithm for 3D space”. In: *International Journal of Advanced Research in Computer Science and Software Engineering* 2.5 (2012), pp. 96–101.

[KWC13] S. R. Kodituwakku, K. R. Wijeweera, and M. A. P. Chamikara. “An efficient algorithm for line clipping in computer graphics programming”. In: *Ceylon Journal of Science (Physical Sciences)* 1.17 (2013), pp. 1–7.

[LB84] Y-D. Liang and B. A. Barsky. “A new concept and method for line clipping”. In: *tog* 3.1 (1984), pp. 1–22. DOI: 10.1145/357332.357333.

[LZZ07] Sheng Liu, Jian Zhang, and Binhai Zhu. “Volume Computation Using a Direct Monte Carlo Method”. In: vol. 4598. July 2007, pp. 198–209. ISBN: 978-3-540-73544-1. DOI: 10.1007/978-3-540-73545-8_21.

[Mał06] Robert Małysz. “The Minkowski dimension of the bivariate fractal interpolation surfaces”. In: *Chaos, Solitons & Fractals* 27.5 (2006), pp. 1147–1156. ISSN: 0960-0779. DOI: https://doi.org/10.1016/j.chaos.2005.05.007. URL: https://www.sciencedirect.com/science/article/pii/S0960077905004686.

[Mas90] Peter R Massopust. “Fractal surfaces”. In: *Journal of Mathematical Analysis and Applications* 151.1 (1990), pp. 275–290. ISSN: 0022-247X. DOI: https://doi.org/10.1016/0022-247X(90)90257-G. URL: https://www.sciencedirect.com/science/article/pii/0022247X9090257G.

[Mas93] Peter Massopust. “Fractal interpolation functions from R^n into R^m and their projections”. In: *Zeitschrift fur Analysis und ihre Anwendungen = Journal of analysis and its applications* 12 (Jan. 1993), pp. 535–548.

[Mas95] Peter Massopust. *Fractal Functions, Fractal Surfaces, and Wavelets*. Vol. 22. Jan. 1995. ISBN: 978-0124788404. DOI: 10.1016/S0098-3004(96)80467-5.

[MD19a] D. Matthes and V. Drakopoulos. “A simple and fast line-clipping method as a Scratch extension for computer graphics education”. In: *Computer Science and Information Technology* 7.2 (2019), pp. 40–47. DOI: 10.13189/csit.2019.070202.

[MD19b] D. Matthes and V. Drakopoulos. “Another simple but faster method for 2D line clipping”. In: *International Journal of Computer Graphics & Animation* 9.1/2/3 (2019), pp. 1–15. DOI: 10.5121/ijcga.2019.9301.

[MD22] D. Matthes and V. Drakopoulos. “Line Clipping in 2D: Overview, Techniques and Algorithms”. In: *Journal of Imaging* 8.10 (2022). ISSN: 2313-433X. DOI: 10.3390/jimaging8100286. URL: https://www.mdpi.com/2313-433X/8/10/286.

[MD23] D. Matthes and V. Drakopoulos. “Line Clipping in 3D: Overview, Techniques and Algorithms”. In: *Algorithms* 16.4 (2023). ISSN: 1999-4893. DOI: 10.3390/a16040201. URL: https://www.mdpi.com/1999-4893/16/4/201.

[Meg82] Nimrod Megiddo. “Linear-time algorithms for linear programming in R3 and related problems”. In: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. 1982, pp. 329–338. DOI: 10.1109/SFCS.1982.24.

[Meh84] Kurt Mehlhorn. *Data structures and algorithms 3: multi-dimensional searching and computational geometry*. Berlin, Heidelberg: Springer-Verlag, 1984. ISBN: 0387136428.

[MS06] Kurt Mehlhorn and Klaus Simon. “Intersecting two polyhedra one of which is convex”. In: Apr. 2006, pp. 534–542. ISBN: 3-540-15689-5. DOI: 10.1007/BFb0028837.

[MY08] W. Metzler and C. Yun. “Construction of fractal interpolation surfaces on rectangular grids”. In: *International Journal of Bifurcation and Chaos* (2008).

[Nis17] Nisha. “Comparison of various line clipping algorithms : Review”. In: *International Journal of Advanced Research in Computer Science and Software Engineering* 7.1 (2017), pp. 68–71. DOI: 10.23956/ijarcsse/V7I1/0149.

[NLN87] Tina M. Nicholl, D.T. Lee, and Robin A. Nicholl. “An efficient new algorithm for 2-D line clipping : Its development and analysis”. In: *Comput. Graph.* 21.4 (1987), pp. 253–262. DOI: 10.1145/37402.37432.

[ORo+82] Joseph O’Rourke, Chi-Bin Chien, Thomas Olson, and David Naddor. “A New Linear Algorithm for Intersecting Convex Polygons”. In: *Computer Graphics and Image Processing* 19 (1982), pp. 384–391.

[PH77] F.P. Preparata and S.J. Hong. “Convex Hulls of Finite Sets of Points in Two and Three Dimensions”. In: *Communications of the ACM* 20 (1977), pp. 87–93.

[PH98] J.R. Price and Monson Hayes. “Fractal interpolation of images and volumes”. In: vol. 2. Dec. 1998, pp. 1698–1702. DOI: 10.1109/ACSSC.1998.751615.

[PJ13] A. Pandey and S. Jain. “Comparison of various line clipping algorithms for Improvement”. In: *International Journal of Modern Engineering Research* 3.1 (2013), pp. 69–74.

[Pre+92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Second. Cambridge, USA: Cambridge University Press, 1992.

[PS84] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry. An Introduction*. Springer New York, NY, 1984. ISBN: 978-960-8407-43-5. DOI: 10.1007/978-1-4612-1098-6.

[Qia02] Xiao-yuan Qian. “Bivariate Fractal Interpolation Functions on Rectangular Domains”. In: *Journal of Computational Mathematics* 20.4 (2002), pp. 349–362. ISSN: 02549409, 19917139. URL: http://www.jstor.org/stable/43693001 (visited on 06/26/2024).

[Rap91] A. Rappoport. “An efficient algorithm for line and polygon clipping”. In: *The Visual Computer* 7.1 (1991), pp. 19–28. DOI: 10.1007/BF01994114.

[Ray12] B. K. Ray. “A Line Segment Clipping Algorithm in 2D”. In: *International Journal of Computer Graphics* 3.2 (2012), pp. 51–76.

[Rog97] David F. Rogers. *Procedural Elements for Computer Graphics (2nd Ed.)* USA: McGraw-Hill, Inc., 1997. ISBN: 0070535485. DOI: 10.5555/265140.

[SA16] SHILPA and Prince Arora. “Clipping in Computer Graphics-A review”. In: *International Journal Of Advance Research And Innovative Ideas In Education* 2.6 (2016), pp. 1725–1730. DOI: 16.0415/IJARIIE-3590. URL: http://ijariie.com/AdminUploadPdf/Clipping_in_Computer_Graphics_A_review_ijariie3590.pdf.

[SH74] I. Sutherland and G. Hodgman. “Reentrant Polygon Clipping”. In: *Communications of the ACM* 17 (1974), pp. 32–42. DOI: 10.1145/360767.360802.

[SH76] M .I. Shamos and O. Hoey. “Geometric intersection problems”. In: *Proc. 17th Annual IEEE Symposium on the Foundations of Computer Science – IEEE Computer Society* (1976), pp. 208–215.

[Ska05] V. Skala. “A new approach to line and line segment clipping in homogeneous coordinates”. In: *Visual Comput.* 21 (2005), pp. 905–914. DOI: 10.1007/s00371-005-0305-3.

[Ska12] V. Skala. “S-clip E2: A new concept of clipping algorithms”. In: *SIGGRAPH Asia* (2012). DOI: 10.1145/2407156.2407200.

[Ska93] V. Skala. “An efficient algorithm for line clipping by convex polygon”. In: *Comput. Graph.* 17.4 (1993), pp. 417–421. DOI: 10.1016/0097-8493(93)90030-D.

[Ska94] V. Skala. “O(lg N) Line clipping algorithm in E^2”. In: *Comput. Graph.* 18.4 (1994), pp. 517–524. DOI: 10.1016/0097-8493(94)90064-7.

[SPY87] M. S. Sobkow, P. Pospisil, and Y. Yang. “A Fast Two-Dimensional Line Clipping Algorithm Via Line Encoding”. In: *Comput. Graph.* 11.4 (1987), pp. 459–467. DOI: 10.1016/0097-8493(87)90061-6.

[SX04] Hongquan Sun and Heping Xie. “Study on the Improved Fractal Interpolation Surface of the Attitude and Surface of Fault”. In: *Thinking in Patterns*. 2004, pp. 243–254. DOI: 10.1142/9789812702746_0020. eprint: https://www.worldscientific.com/doi/pdf/10.1142/9789812702746_0020. URL: https://www.worldscientific.com/doi/abs/10.1142/9789812702746_0020.

[Vat92] B. R. Vatti. “A generic solution to polygon clipping”. In: *Communications of the ACM* 35 (1992), pp. 56–63. DOI: 10.1145/129902.129906.

[WA77] Kevin Weiler and Peter Atherton. “Hidden Surface Removal Using Polygon Area Sorting”. In: SIGGRAPH ’77. San Jose, California: Association for Computing Machinery, 1977, pp. 214–222. ISBN: 9781450373555. DOI: 10.1145/563858.563896. URL: https://doi.org/10.1145/563858.563896.

[Wan06] Hong-Yong Wang. “On smoothness for a class of fractal interpolation surfaces”. In: *Fractals* 14 (Sept. 2006), pp. 223–230. DOI: 10.1142/S0218348X06003222.

[Web94] R. Webster. *Convexity*. Oxford science publications. Oxford University Press, 1994. ISBN: 9780198531470. URL: https://books.google.gr/books?id=kZZVngEACAAJ.

[Wit95] C.M. Wittenbrink. “IFS fractal interpolation for 2D and 3D visualization”. In: *Proceedings Visualization ’95*. 1995, pp. 77–84. DOI: 10.1109/VISUAL.1995.480798.

[XF99] H. Xie and Z. Feng. “On star product fractal surfaces and their dimensions”. In: *Applied Mathematics and Mechanics* 20.11 (1999), pp. 1183–1189.

[Xie+01] Heping Xie, Hongquan Sun, Yang Ju, and Zhigang Feng. “Study on generation of rock fracture surfaces by using fractal interpolation”. In: *International Journal of Solids and Structures* 38.32 (2001), pp. 5765–5787. ISSN: 0020-7683. DOI: https://doi.org/10.1016/S0020-7683(00)00390-5. URL: https://www.sciencedirect.com/science/article/pii/S0020768300003905.

[XS11] Heping Xie and Hongquan Sun. “The Study on Bivariate Fractal Interpolation Functions and Creation of Fractal Interpolated Surfaces”. In: *Fractals* 05 (Nov. 2011). DOI: 10.1142/S0218348X97000504.

[XS98] Heping Xie and Hongquan Sun. “The theory of fractal interpolated surface and its applications”. In: *Applied Mathematics and Mechanics* 19.4 (1998), pp. 321–331. DOI: 10.1007/BF02457536.

[Zha96] Nailiang Zhao. “Construction and application of fractal interpolation surfaces”. In: *The Visual Computer* 12 (1996), pp. 132–146.

ΕΛΛΗΝΟΓΛΩΣΣΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

[Δρα19] Β. Δρακόπουλος. *Εισαγωγή εις την γεωμετρίαν των μορφοκλασματικών συνόλων*. Σημειώσεις παραδόσεων μαθήματος «Μορφοκλασματική και Υπολογιστική Γεωμετρία» τμήματος Πληροφορικής με Εφαρμογές στη Βιοϊατρική. Πανεπιστήμιο Θεσσαλίας. 2019.

[Θεο+15] Θ. Θεοχάρης, Γ. Παπαϊωάννου, Ν. Πλατής και Ν. Μ. Πατρικαλάκης. *Γραφικά και Οπτικοποίηση: Αρχές & Αλγόριθμοι*. 3η έκδοση. Εκδόσεις Συμμετρία, 2015.

[Μαν09] Π. Μανουσόπουλος. "Εξακρίβωση παραμέτρων και αλγοριθμική κατασκευή μορφοκλασματικών συναρτήσεων παρεμβολής: Εφαρμογές στον ψηφιακό εικονισμό και την οπτικοποίηση". Διδακτορική Διατριβή. 2009.

[Μπο06] Π. Μπουμπούλης. "Fractal επιφάνειες παρεμβολής, Θεωρία και εφαρμογές στη συμπίεση εικόνας". Διδακτορική Διατριβή. 2006.

[Σγο12] Δ. Σγούρδος. "Εξακρίβωση παραμέτρων διδιάστατων μορφοκλασματικών συναρτήσεων παρεμβολής χρησιμοποιώντας περιβάλλοντες όγκους". Πτυχιακή εργασία. 2012.

ΕΥΡΕΤΗΡΙΟ

ΓΛΩΣΣΑΡΙ

Αναδίπλωση	Fold	105
Ανάκλαση	Reflection	105
Αποκοπή	Clipping	4
Αρχέγονο	Primitive	5
Αστρικό γινόμενο	Star product	94
Αυξητικός	Incremental	124
Διαίρει και Βασίλευε	Divide and Conquer	26, 126
Διαμέριση	Partitioning	129
Γραφική Διεπαφή Χρήστη	Graphical User Interface	157
Δοκιμασία t	T-test	48
Δοκιμασία αθροίσματος κατάταξης Wilcoxon	Wilcoxon rank sum test	48
Δοκιμασία με επίπεδα	Plane test	81
Δυαδικό ψηφίο	Bit	9
Δυνητική Εισχώρηση	Potentially Entering	15
Δυνητική Αποχώρηση	Potentially Leaving	15
Εισερχόμενο φυσιολογικό διάνυσμα	Inward normal vector	14
Εξωκωδικός	Outcode	10, 27, 32
Επαναλαμβανόμενο Σύστημα Συναρτήσεων	Iterated Function System	104
Ερευνητική εργασία	Project	39

ΠΑΡΑΡΤΗΜΑ

Α. ΚΩΔΙΚΕΣ ΑΛΓΟΡΙΘΜΩΝ

```
const int INSIDE = 0; // 0000
const int LEFT = 1;   // 0001
const int RIGHT = 2;  // 0010
const int BOTTOM = 4; // 0100
const int TOP = 8;    // 1000

int compute_outcode(float x, float y)
{
  int code = INSIDE;      // initialised as being inside of clip are

  if (x < xmin)           // left to the clip area
    code |= LEFT;
  else if (x > xmax)      // right to the clip area
    code |= RIGHT;

  if (y < ymin)           // below the clip area
    code |= BOTTOM;
  else if (y > ymax)      // above the clip area
    code |= TOP;

  return code;
}

void cohen_sutherland(float x1, float y1, float x2, float y2,
float xmin, float ymin, float xmax, float ymax)
{
  int accept = false;
  int done = false;
  int outcode1 = compute_outcode(x1, y1);
  int outcode2 = compute_outcode(x2, y2);

  do
  {
    if (outcode1 == 0 && outcode2 == 0)
    {
      accept = true;
      done = true;
    }
    else if (outcode1 & outcode2)
    {
      done = true;
    }
    else
    {
      float x, y;
      int outcode_ex = outcode1 ? outcode1 : outcode2;

      if (outcode_ex & TOP)
      {
        x = x1 + (x2 - x1) * (ymax - y1) / (y2 - y1);
        y = ymax;
      }
      else if (outcode_ex & BOTTOM)
      {
        x = x1 + (x2 - x1) * (ymin - y1) / (y2 - y1);
        y = ymin;
```

```
        }
        else if (outcode_ex & RIGHT)
        {
          y = y1 + (y2 - y1) * (xmax - x1) / (x2 - x1);
          x = xmax;
        }
        else
        {
          y = y1 + (y2 - y1) * (xmin - x1) / (x2 - x1);
          x = xmin;
        }

        if (outcode_ex == outcode1)
        {
          x1 = x;
          y1 = y;
          outcode1 = compute_outcode(x1, y1);
        }
        else
        {
          x2 = x;
          y2 = y;
          outcode2 = compute_outcode(x2, y2);
        }
      }
    }
    while (!done);

    if(accept)
      draw_line(x1, y1, x2, y2);
  }
```

Κώδικας Α.1: Ο κώδικας του αλγορίθμου Cohen-Sutherland

```
  struct point { float x, y; };   // define type for a 2D point

  void cyrus_beck(point &p1, point &p2, point (&polygon)[N])
  {
    float wn, dn, sn, tE, tL, t;
    point normal, d, w, pE, pL;
    bool draw = true;
    d.x = p2.x - p1.x;
    d.y = p2.y - p1.y;
    tE = 0;
    tL = 1;

    for(int i = 0; i < N ; i++)
    {
      // calculate the inward normal vector n(dy, -dx),
      // where dx = x2 - x1 & dy = y2 - y1
      normal.x =  (polygon[i + 1].y - polygon[i].y);
      normal.y = -(polygon[i + 1].x - polygon[i].x);

      // calculate co-ordinates of W (p1 - Ei), where Ei --> boundary/edge
      w.x = p1.x - polygon[i].x;
      w.y = p1.y - polygon[i].y;

      // dot product of W*n
      wn = w.x * normal.x + w.y * normal.y;

      // dot product of D*n
      dn = d.x * normal.x + d.y * normal.y;

      // check D*n. If D*n==0, the line is parallel to the edge
      if(dn == 0)
      {
        // If W*n < 0, the line is completely outside the polygon
        if(wn < 0)
```

```
        {
          draw = false;
          break;
        }
      }
      else
      {
        // calculate the t value of the intersection between line & edge
        t = -float(wn / dn);
        if(dn > 0)
          tE = (t > tE ? t : tE); //tE = max(t, tE);
        else if(dn < 0)
          tL = (t < tL ? t : tL); //tL = min(t, tL);
      }
    }

    if(tE < tL && draw)
    {
      pE.x = p1.x + d.x * tE;
      pE.y = p1.y + d.y * tE;
      pL.x = p1.x + d.x * tL;
      pL.y = p1.y + d.y * tL;

      // draw the clipped line
      draw_line(pE, pL);
    }
  }
```

Κώδικας Α.2: Ο κώδικας του αλγορίθμου Cyrus-Beck

```
  void liang_barsky(float x0, float y0, float x1, float y1,
  float xmin, float ymin, float xmax, float ymax)
  {

    float t[2], dx, dy, p[4], q[4], r[4], xa, ya, xb, yb;

    t[0] = 0;
    t[1] = 1;

    dx = x1 - x0;
    dy = y1 - y0;

    p[0] = -dx; q[0] = x0 - xmin;
    p[1] =  dx; q[1] = xmax - x0;
    p[2] = -dy; q[2] = y0 - ymin;
    p[3] =  dy; q[3] = ymax - y0;

    if (((p[0] == 0.0) && (q[0] < 0.0)) || ((p[1] == 0.0) && (q[1] < 0.0)) ||
    ((p[2] == 0.0) && (q[2] < 0.0)) || ((p[3] == 0.0) && (q[3] < 0.0))
    )
    {
      /*Line is rejected*/
      return;
    }
    else
    {
      for(int i = 0; i <= 3; i++)
      {
        if (p[i] != 0.0)
        {
          r[i] = q[i] / p[i];
          if (p[i] < 0)
            t[0] = r[i] > t[0] ? r[i] : t[0];
          else
            t[1] = r[i] < t[1] ? r[i] : t[1];
        }
      }
```

```
      if (t[0] > t[1])
      {
        /*Line is rejected*/
        return;
      }
      else
      {
        xa = x0 + t[0] * dx;
        ya = y0 + t[0] * dy;
        xb = x0 + t[1] * dx;
        yb = y0 + t[1] * dy;
        draw_line(xa, ya, xb, yb);
      }
    }
  }
```

Κώδικας Α.3: Ο κώδικας του αλγορίθμου Liang-Barsky

```
//Rotation of 90° clockwise about the origin
void rotate90c(float &x, float &y) { float t = x; x = y; y = -t; }

//Rotation of 180° clockwise about the origin
void rotate180c(float &x, float &y) { x = -x; y = -y; }

//Rotation of 270° clockwise about the origin
void rotate270c(float &x, float &y) { float t = x; x = -y; y = t; }

//Reflection about the line x = -y
void reflectxminusy(float &x, float &y) { float t = x; x = -y; y = -t; }

//Reflection about the x-axis
void reflectxaxis(float &x, float &y) { y = -y; }

void p2lefttop(float xleft, float ytop, float xright, float ybottom, float &x1, float &y1, float &
  x2, float &y2)
{
  float relx2, rely2, leftproduct, topproduct;
  relx2 = x2 - x1; rely2 = y2 - y1;
  leftproduct = rely2 * (xleft - x1);
  topproduct = relx2 * (ytop - y1);
  if (topproduct > leftproduct)
  {
    x2 = x1 + topproduct / rely2;
    y2 = ytop;
  }
  else
  {
    y2 = y1 + leftproduct / relx2;
    x2 = xleft;
  }
}

void p2left(float xleft, float ytop, float xright, float ybottom, float &x1, float &y1, float &x2,
   float &y2)
{
  if (y2 > ytop)
    p2lefttop(xleft, ytop, xright, ybottom, x1, y1, x2, y2);
  else if (y2 < ybottom)
  {
    rotate90c(x1, y1); rotate90c(x2, y2);
    p2lefttop(ybottom, -xleft, ytop, -xright, x1, y1, x2, y2);
    rotate270c(x1, y1); rotate270c(x2, y2);
  }
  else
  {
    y2 = y1 + (y2 - y1) * (xleft - x1) / (x2 - x1);
    x2 = xleft;
  }
```

```
}

void inside(float xleft, float ytop, float xright, float ybottom, float &x1, float &y1, float &x2,
   float &y2, bool &display)
{
  display = true;
  if (x2 < xleft)
    p2left(xleft, ytop, xright, ybottom, x1, y1, x2, y2);
  else if (x2 > xright)
  {
    rotate180c(x1, y1); rotate180c(x2, y2);
    p2left(-xright, -ybottom, -xleft, -ytop, x1, y1, x2, y2);
    rotate180c(x1, y1); rotate180c(x2, y2);
  }
  else if (y2 > ytop)
  {
    x2 = x1 + (x2 - x1) * (ytop - y1)/(y2 - y1);
    y2 = ytop;
  }
  else if (y2 < ybottom)
  {
    x2 = x1 + (x2 - x1) * (ybottom - y1) / (y2 - y1);
    y2 = ybottom;
  }
  // else P2 is inside, just display, no need to clip
}

void p2bottom(float xleft, float ytop, float xright, float ybottom, float &x1, float &y1, float &
  x2, float &y2, bool &display)
{
  float leftproduct, bottomproduct, rightproduct, relx2, rely2;

  relx2 = x2 - x1; rely2 = y2 - y1;
  leftproduct = (xleft - x1) * rely2;
  bottomproduct = (ybottom - y1) * relx2;
  if (bottomproduct > leftproduct)
    display = false;
  else
  {
    if (x2 <= xright)
    {
      x2 = x1 + bottomproduct / rely2;
      y2 = ybottom;
    }
    else
    {
      rightproduct = (xright - x1) * rely2;
      if (bottomproduct > rightproduct)
      {
        x2 = x1 + bottomproduct / rely2;
        y2 = ybottom;
      }
      else
      {
        y2 = y1 + rightproduct / relx2;
        x2 = xright;
      }
    }
    y1 = y1 + leftproduct / relx2;
    x1 = xleft;
    display = true;
  }
}

void leftedge(float xleft, float ytop, float xright, float ybottom, float &x1, float &y1, float &
  x2, float &y2, bool &display)
{
  float relx2, rely2;

  if (x2 < xleft)
```

```
      display = false;
    else if (y2 < ybottom)
      p2bottom(xleft, ytop, xright, ybottom, x1, y1, x2, y2, display);
    else if (y2 > ytop)
    {
      reflectxaxis(x1, y1); reflectxaxis(x2, y2);
      p2bottom(xleft, -ybottom, xright, -ytop, x1, y1, x2, y2, display);
      reflectxaxis(x1, y1); reflectxaxis(x2, y2);
    }
    else
    {
      relx2 = x2 - x1; rely2 = y2 - y1;
      if (x2 > xright)
      {
        y2 = y1 + rely2 * (xright - x1) / relx2;
        x2 = xright;
      }
      y1 = y1 + rely2 * (xleft - x1) / relx2;
      x1 = xleft;
      display = true;
    }
  }

  void centrecolumn(float xleft, float ytop, float xright, float ybottom, float &x1, float &y1,
    float &x2, float &y2, bool &display)
  {
    if (y1 > ytop)
    {
      rotate270c(x1, y1); rotate270c(x2, y2);
      leftedge(-ytop, xright, -ybottom, xleft, x1, y1, x2, y2, display);
      rotate90c(x1, y1); rotate90c(x2, y2);
    }
    else if (y1 < ybottom)
    {
      rotate90c(x1, y1); rotate90c(x2, y2);
      leftedge(ybottom, -xleft, ytop, -xright, x1, y1, x2, y2, display);
      rotate270c(x1, y1); rotate270c(x2, y2);
    }
    else
    inside(xleft, ytop, xright, ybottom, x1, y1, x2, y2, display);
  }

  void leftbottomregion(float xleft, float ytop, float xright, float ybottom, float &x1, float &y1,
    float &x2, float &y2, bool &display, float relx2, float rely2, float leftproduct)
  {
    float bottomproduct, rightproduct;

    if (y2 >= ybottom)
    {
      if (x2 > xright)
      {
        y2 = y1 + (xright - x1) * rely2 / relx2;
        x2 = xright;
      }
      y1 = y1 + leftproduct / relx2;
      x1 = xleft;
      display = true;
    }
    else
    {
      bottomproduct = (ybottom - y1) * relx2;
      if (bottomproduct > leftproduct)
        display = false;
      else {
        if (x2 > xright)
        {
          rightproduct = (xright - x1) * rely2;
          if (bottomproduct > rightproduct)
          {
            x2 = x1 + bottomproduct / rely2;
```

```
            y2 = ybottom;
          }
          else
          {
            y2 = y1 + rightproduct / relx2;
            x2 = xright;
          }
        }
        else {
          x2 = x1 + bottomproduct / rely2;
          y2 = ybottom;
        }
        y1 = y1 + leftproduct / relx2;
        x1 = xleft;
        display = true;
      }
    }
  }

  void topleftcorner(float xleft, float ytop, float xright, float ybottom, float &x1, float &y1,
    float &x2, float &y2, bool &display)
  {
    float relx2, rely2, topproduct, leftproduct;

    if (y2 > ytop)
      display = false;
    else
    {
      relx2 = x2 - x1; rely2 = y2 - y1;
      topproduct = (ytop - y1)* relx2;
      leftproduct = (xleft - x1) * rely2;
      if (topproduct > leftproduct)
        leftbottomregion(xleft, ytop, xright, ybottom, x1, y1, x2, y2, display, relx2, rely2,
    leftproduct);
      else
      {
        reflectxminusy(x1, y1); reflectxminusy(x2, y2);
        leftbottomregion(-ytop, -xleft, -ybottom, -xright, x1, y1, x2, y2, display, -rely2, -relx2,
    topproduct);
        reflectxminusy(x1, y1); reflectxminusy(x2, y2);
      }
    }
  }

  void leftcolumn(float xleft, float ytop, float xright, float ybottom, float &x1, float &y1, float
    &x2, float &y2, bool &display)
  {
    if (x2 < xleft)
      display = false;
    else if (y1 > ytop)
      topleftcorner(xleft, ytop, xright, ybottom, x1, y1, x2, y2, display);
    else if (y1 < ybottom)
    {
      reflectxaxis(x1, y1); reflectxaxis(x2, y2);
      topleftcorner(xleft, -ybottom, xright, -ytop, x1, y1, x2, y2, display);
      reflectxaxis(x1, y1); reflectxaxis(x2, y2);
    }
    else
      leftedge(xleft, ytop, xright, ybottom, x1, y1, x2, y2, display);
  }

  void clip(float xleft, float ytop, float xright, float ybottom, float x1, float y1, float x2,
    float y2)
  {
    bool display=false;
    if (x1 < xleft)
      leftcolumn(xleft, ytop, xright, ybottom, x1, y1, x2, y2, display);
    else if (x1 > xright)
    {
      rotate180c(x1, y1); rotate180c(x2, y2);
```

```
      leftcolumn(-xright, -ybottom, -xleft, -ytop, x1, y1, x2, y2, display);
      rotate180c(x1, y1); rotate180c(x2, y2);
    }
    else
      centrecolumn(xleft, ytop, xright, ybottom,  x1, y1, x2, y2, display);

    if (display)
    {
      draw_line(x1, y1, x2, y2);
    }
  }
```

Κώδικας Α.4: Ο κώδικας του αλγορίθμου Nicholl-Lee-Nicholl

```
typedef unsigned OUTCODE;

#define XMIN -100
#define YMIN -75
#define XMAX 100
#define YMAX 75

// Not strictly portable, but usually fine.
#define SIGN_BIT (~(~0u >> 1))

#define LEFT SIGN_BIT
#define TOP (LEFT >> 1)
#define RIGHT (TOP >> 1)
#define BOTTOM (RIGHT >> 1)
#define ALL (LEFT | BOTTOM | RIGHT | TOP)

// Mask the sign bit.
#define M(X) ((X) & SIGN_BIT)

// Shift previous value and mask in the new sign bit.
#define SM(Prev, New) (((OUTCODE)(Prev) >> 1) | M(New))

__inline OUTCODE outcode(int x, int y)
{
  return SM(SM(SM(M(YMAX - y), XMAX - x), y - YMIN), x - XMIN);
}

// In the S-T intinate system, pO is outside boundary C and will be moved
// to the boundary while pI doesn't move. I is the termination correction.
#define MOVE_TO_BOUNDARY(SO, TO, SI, TI, C, I, IS_OUTSIDE) do { \
  int tsi = SI, tti = TI; \
  while (SO IS_OUTSIDE C) { \
    int sm = (tsi + SO + I) >> 1; \
    int tm = (tti + TO + I) >> 1; \
    if (sm IS_OUTSIDE ## = C) { \
      SO = sm; \
      TO = tm; \
    } else { \
      tsi = sm; \
      tti = tm; \
    } \
  } \
} while (0)

bool midpoint_subdivision(float *x0p, float *y0p, float *x1p, float *y1p)
{
  int x0 = *x0p, y0 = *y0p, x1 = *x1p, y1 = *y1p;
  OUTCODE code0 = outcode(x0, y0);
  OUTCODE code1 = outcode(x1, y1);
  for (;;)
  {
    if ((code0 | code1) == 0)
    {
      *x0p = x0;
```

```
        *y0p = y0;
        *x1p = x1;
        *y1p = y1;
        return true;
      }
      else if (code0 & code1)
      {
        return false;
      }
      else if (code0)
      {
        if (code0 & BOTTOM)
          MOVE_TO_BOUNDARY(y0, x0, y1, x1, YMAX, 0, >);
        else if (code0 & RIGHT)
          MOVE_TO_BOUNDARY(x0, y0, x1, y1, XMAX, 0, >);
        else if (code0 & TOP)
          MOVE_TO_BOUNDARY(y0, x0, y1, x1, YMIN, 1, <);
        else /* LEFT */
          MOVE_TO_BOUNDARY(x0, y0, x1, y1, XMIN, 1, <);
        code0 = outcode(x0, y0);
      }
      else
      {
        if (code1 & BOTTOM)
          MOVE_TO_BOUNDARY(y1, x1, y0, x0, YMAX, 0, >);
        else if (code1 & RIGHT)
          MOVE_TO_BOUNDARY(x1, y1, x0, y0, XMAX, 0, >);
        else if (code1 & TOP)
          MOVE_TO_BOUNDARY(y1, x1, y0, x0, YMIN, 1, <);
        else /* LEFT */
          MOVE_TO_BOUNDARY(x1, y1, x0, y0, XMIN, 1, <);
        code1 = outcode(x1, y1);
      }
    }
  }
```

Κώδικας Α.5: Ο κώδικας του αλγορίθμου Midpoint Subdivision

```
int outcode(point p)
{
  int code = 0;

  if(p.x < XMIN)
    code = 8;
  else if (p.x > XMAX)
    code = 4;

  if(p.y < YMIN)
    code |= 9;
  else if (p.y > YMAX)
    code |= 2;

  return code;
}
```

Κώδικας Α.6: Κώδικας υπολογισμού του κωδικού περιοχής του αλγορίθμου Skala 2005

```
int calculate_c_index(point a, point b, point *polygon)
{
  // Coefficients of the line Ax + By + C = 0
  float A = a.y - b.y;
  float B = b.x - a.x;
  float C = a.x * b.y - a.y * b.x;

  // Calculate c value of TAB table
```

```
    int vertex[N], c = 0;
    for (int k = 0, power = 1; k < N; k++, power *= 2)
    {
      vertex[k] = polygon[k].x * A + polygon[k].y * B + C < 0;
      c += vertex[k] *  power;
    }
    return c;
  }
```

Κώδικας Α.7: Κώδικας για τον χαρακτηρισμό της θέσης των κορυφών με 0 ή 1 και υπολογισμός του δείκτη *c* στον Skala 2005

```
  void skala2005(point &a, point &b, point *polygon)
  {
    // calculate the outcode for each endpoint
    int codeA = outcode(a);
    int codeB = outcode(b);

    if( (codeA | codeB) == 0)
    {
      // line is totally inside
      draw_line(a, b);
      return;
    }

    if( (codeA & codeB) !=0)
    {
      // line is totally outside
      return;
    }

    int c = calculate_c_index(a, b, polygon);

    // Non available cases
    if(c == 0 || c == 15) return;

    // TAB1, TAB2 and MASK values
    int tab1[16] = { -1, 0, 0, 1, 1, -2, 0, 2, 2, 0, -2, 1, 1, 0, 0, -1 };
    int tab2[16] = { -1, 3, 1, 3, 2, -2, 2, 3, 3, 2, -2, 2, 3, 1, 3, -1 };
    int mask[16] = { -1, 1, 1, 4, 4, -2, 1, 4, 4, 1, -2, 4, 4, 1, 1, -1 };

    int i = tab1[c];
    int j = tab2[c];

    if(codeA != 0 && codeB != 0)
    {
      // there are two intersections
      point newpointA = crossProduct(a, b, polygon[i], polygon[i + 1]);
      point newpointB = crossProduct(a, b, polygon[j], polygon[j + 1]);
      draw_line(newpointA, newpointB);
    }
    else
    {
      if(codeA == 0)
      {
        // point A is inside, point B is outside
        point newpointB;
        if( (codeB & mask[c]) != 0)
          newpointB = crossProduct(a, b, polygon[i], polygon[i + 1]);
        else
          newpointB = crossProduct(a, b, polygon[j], polygon[j + 1]);
        draw_line(a, newpointB);
      }
      else if(codeB == 0)
      {
        // point B is inside, point A is outside
        point newpointA;
        if( (codeA & mask[c]) != 0)
          newpointA = crossProduct(a, b, polygon[i], polygon[i + 1]);
```

```
        else
          newpointA = crossProduct(a, b, polygon[j], polygon[j + 1]);
        draw_line(newpointA, b);
      }
    }
  }
```

Κώδικας Α.8: Ο κώδικας του αλγορίθμου Skala 2005 (ορθογώνια περιοχής αποκοπής)

```
void sclip(point &a, point &b, point *polygon)
{
  int c = calculate_c_index(a, b, polygon);

  // Exit for non available cases
  if(c == 0 || c == 15) return;

  // TAB1, TAB2 and MASK values
  int tab1[16] = { -1, 3, 0, 3, 1, -2, 0, 3, 2, 2, -2, 2, 1, 1, 0, -1 };
  int tab2[16] = { -1, 0, 1, 1, 2, -2, 2, 2, 3, 0, -2, 1, 3, 0, 3, -1 };
  int i = tab1[c];
  int j = tab2[c];

  // Calculate the two intersection points between line segment and edges
  point inter1 = cross_product(a, b, polygon[i], polygon[i + 1]);
  point inter2 = cross_product(a, b, polygon[j], polygon[j + 1]);

  // Calculate parameter "t" of each intersection point
  float t1, t2;
  if(a.x != b.x)
  {
    t1 = (inter1.x - a.x) / (b.x - a.x);
    t2 = (inter2.x - a.x) / (b.x - a.x);
  }
  else
  {
    t1 = (inter1.y - a.y) / (b.y - a.y);
    t2 = (inter2.y - a.y) / (b.y - a.y);
  }

  // Apply clipping
  if(t1 < 0 && t2 > 1)
    draw_line(a, b);
  else if((t1 >= 0 && t1 <= 1) && (t2 >= 0 && t2 <= 1))
    draw_line(inter1, inter2);
  else if((t1 >= 0 && t1 <= 1) && !(t2 >= 0 && t2 <= 1))
  {
    if(t2 < 0)
      draw_line(a, inter1);
    else
      draw_line(inter1, b);
  }
  else if(!(t1 >= 0 && t1 <= 1) && (t2 >= 0 && t2 <= 1))
  {
    if(t1 < 0)
      draw_line(a, inter2);
    else
      draw_line(inter2, b);
  }
}
```

Κώδικας Α.9: Ο κώδικας του αλγορίθμου S-Clip E^2

```
void clipKWC(float x0, float y0, float x1, float y1)
{
  int i;
```

```
    float m, c, x[2], y[2];

    x[0] = x0;
    y[0] = y0;
    x[1] = x1;
    y[1] = y1;

    // non vertical lines
    if (x[0] != x[1])
    {
      // non horizontal lines
      if (y[0] != y[1])
      {
        // calculate the gradient
        m = (y[0] - y[1]) / (x[0] - x[1]);
        // calculate the y-intercept
        c = (x[0] * y[1] - x[1] * y[0]) / (x[0] - x[1]);
        for (i = 0; i<2; i++)
        {
          if (x[i]<xmin)
          {
            x[i] = xmin;
            y[i] = m*xmin + c;
          }
          else if (x[i]>xmax)
          {
            x[i] = xmax;
            y[i] = m*xmax + c;
          }
          if (y[i]<ymin)
          {
            x[i] = (ymin - c) / m;
            y[i] = ymin;
          }
          else if (y[i]>ymax)
          {
            x[i] = (ymax - c) / m;
            y[i] = ymax;
          }
        }

        if ((x[0] - x[1] < 1) && (x[1] - x[0] < 1))
        {
          // initial line is completely outside, do nothing
        }
        else
        {
          // draw the clipped line
          draw_line(x[0], y[0], x[1], y[1]);
        }
      }
      // horizontal lines
      else
      {
        if ((y[0] <= ymin) || (y[0] >= ymax))
        {
          // initial line is completely outside, do nothing
        }
        else
        {
          for (i = 0; i<2; i++)
          {
            if (x[i]<xmin)
              x[i] = xmin;
            else if (x[i]>xmax)
              x[i] = xmax;
          }

          if ((x[0] - x[1]<1) && (x[1] - x[0]<1))
          {
```

```
            // initial line is completely outside, do nothing
          }
          else
          {
            // draw the clipped line
            draw_line(x[0], y[0], x[1], y[1]);
          }
        }
      }
    }
    // vertical lines
    else
    {
      // initial line is just a point
      if (y[0] == y[1])
      {
        // initial point is outside
        if ((y[0] <= ymin) || (y[0] >= ymax))
        {
          // do nothing
        }
        else if ((x[0] <= xmin) || (x[0] >= xmax))
        {
          // initial point is outside, do nothing
        }
        else
        {
          // initial point is inside
          draw_line(x[0], y[0], x[1], y[1]);
        }
      }
      else if ((x[0] <= xmin) || (x[0] >= xmax))
      {
        // initial line is completely outside, do nothing
      }
      else
      {
        for (i = 0; i<2; i++)
        {
          if (y[i]<ymin)
            y[i] = ymin;
          else if (y[i]>ymax)
            y[i] = ymax;
        }
        if ((y[0] - y[1]<1) && (y[1] - y[0]<1))
        {
          // initial line is completely outside, do nothing
        }
        else
        {
          // draw the clipped line
          draw_line(x[0], y[0], x[1], y[1]);
        }
      }
    }
  }
```

Κώδικας Α.10: Ο κώδικας του αλγορίθμου KWC

```
  void md_clip(float x0, float y0, float x1, float y1, float xmin, float ymin, float xmax, float
    ymax)
  {
    if(!(x0 < xmin && x1 < xmin) &&
    !(x0 > xmax && x1 > xmax) &&
    !(y0 < ymin && y1 < ymin) &&
    !(y0 > ymax && y1 > ymax))
    {
      float x[2], y[2];
```

```
      x[0] = x0;
      y[0] = y0;
      x[1] = x1;
      y[1] = y1;

      for(int i = 0; i <= 1; i++)
      {
        if(x[i] < xmin)
        {
          x[i] = xmin;
          y[i] = y0 + (y1 - y0) / (x1 - x0) * (xmin - x0);
        }
        else if(x[i] > xmax)
        {
          x[i] = xmax;
          y[i] = y0 + (y1 - y0) / (x1 - x0) * (xmax - x0);
        }
        if(y[i] < ymin)
        {
          y[i] = ymin;
          x[i] = x0 + (x1 - x0) / (y1 - y0) * (ymin - y0);
        }
        else if(y[i] > ymax)
        {
          y[i] = ymax;
          x[i] = x0 + (x1 - x0) / (y1 - y0) * (ymax - y0);
        }
      }
      if (!(x[0] < xmin && x[1] < xmin) && !(x[0] > xmax && x[1] > xmax))
        draw_line(x[0], y[0], x[1], y[1]);
    }
  }
```

Κώδικας Α.11: Ο κώδικας του προτεινόμενου αλγορίθμου

```
struct point { float x, y; };

// check if a point (p) is on the LEFT or the RIGHT side of a line segment (p1->p2)
bool isLeft(point p, point p1, point p2)
{
  return (p2.y - p1.y) * p.x + (p1.x - p2.x) * p.y + (p2.x * p1.y - p1.x * p2.y) < 0;
}

point crossProduct(point e1, point e2, point p1, point p2)
{
  // coefficients of the edge e1 --> e2 (equation: a1*x + b1*y + c1 = 0)
  float a1, b1, c1;
  a1 = e1.y - e2.y;
  b1 = e2.x - e1.x;
  c1 = e1.x * e2.y - e2.x * e1.y;

  // coefficients of the line segment p1 --> p2 (equation: a2*x + b2*y + c2 = 0)
  float a2, b2, c2;
  a2 = p1.y - p2.y;
  b2 = p2.x - p1.x;
  c2 = p1.x * p2.y - p2.x * p1.y;

  // 2 equations, 2 unknowns (x,y), solve the system with determinants
  float d, dx, dy;
  d = a1 * b2 - a2 * b1;
  dx = -c1 * b2 + c2 * b1;
  dy = -a1 * c2 + a2 * c1;

  return { dx/d, dy/d };
}

int calculateCode(point p, float xmin, float ymax, float xmax, float ymin)
```

```
{
  int code = 0;

  if(p.x < xmin)
    code = 8;
  else if (p.x > xmax)
    code = 4;

  if(p.y < ymin)
    code |= 1;
  else if (p.y > ymax)
    code |= 2;

  return code;
}

void skala2005(point &a, point &b, point (&polygon)N[])
{
  // classification of each vertex, check if it's LEFT (0) or RIGHT (1) of the line segment AB
  int vertex[N];
  for (int i = 0 ; i < N; i++)
    isLeft(polygon[i], a, b)? vertex[i] = 0 : vertex[i] = 1;

  // TAB1 & TAB2 values "on-the-fly" based on the alternations of digits 0 and 1
  int TAB[2] = {-1, -1}, k = 0;
  for(int i = 0; i < N; i++)
    if(vertex[i] != vertex[(i + 1)])
      TAB[k++] = i;

  if(TAB[0] != -1 && TAB[1] != -1)
  {
    // calculate intersection points between line and the clipping area
    point inter1 = crossProduct(a, b, polygon[TAB[0]], polygon[(TAB[0] + 1)]);
    point inter2 = crossProduct(a, b, polygon[TAB[1]], polygon[(TAB[1] + 1)]);

    // define coordinates of the rectangle that is formed by the intersection points
    // amount "0.001" fixes the problems with horizontal and vertical lines
    float xmin = min(inter1.x, inter2.x) - 0.001;
    float ymax = max(inter1.y, inter2.y) + 0.001;
    float xmax = max(inter1.x, inter2.x) + 0.001;
    float ymin = min(inter1.y, inter2.y) - 0.001;

    // define this rectangle as the new clipping area
    point rectangle[] = { {xmin, ymin}, {xmax, ymin}, {xmax, ymax}, {xmin, ymax}, {xmin, ymin} };

    // calculate the outcode for each point (Cohen-Sutherland based, 9 regions)
    int codeA = calculateCode(a, xmin, ymax, xmax, ymin);
    int codeB = calculateCode(b, xmin, ymax, xmax, ymin);

    if( (codeA | codeB) == 0)
    {
      // same outcodes means that the line segment is inside the same region, draw it and exit
      drawLine(a.x, a.y, b.x, b.y);
      return;
    }

    if( (codeA & codeB) !=0) return; // line is completely outside, do nothing and exit

    // all trivial cases solved, compare each edge of the rectangle against the line
    int c = 0, base = 1;
    for (int i = 0 ; i < 4; i++)
    {
      if(!isLeft(rectangle[i], a, b))
        c += base;
      base *= 2;
    }

    // both of line endpoints are outside, line crosses only the same edge of the rectangle, use
  the intersections to draw
    if(c == 0 || c == 15)
```

```
        return;

      // predefined TAB1, TAB2 and MASK values
      int TAB1[16] = { -1, 0, 0, 1, 1, -2, 0, 2, 2, 0, -2, 1, 1, 0, 0, -1 };
      int TAB2[16] = { -1, 3, 1, 3, 2, -2, 2, 3, 3, 2, -2, 2, 3, 1, 3, -1 };
      int MASK[16] = { -1, 1, 1, 4, 4, -2, 1, 4, 4, 1, -2, 4, 4, 1, 1, -1 };

      int i = TAB1[c];
      int j = TAB2[c];

      if(codeA != 0 && codeB != 0)
      {
        // there are two intersections
        point newpointA = crossProduct(a, b, rectangle[i], rectangle[i + 1]);
        point newpointB = crossProduct(a, b, rectangle[j], rectangle[j + 1]);
        drawLine(newpointA.x, newpointA.y, newpointB.x, newpointB.y);
      }
      else
      {
        if(codeA == 0)
        {
          // point A is inside, point B is outside
          point newpointB;
          if((codeB & MASK[c]) != 0)
            newpointB = crossProduct(a, b, rectangle[i], rectangle[i + 1]);
          else
            newpointB = crossProduct(a, b, rectangle[j], rectangle[j + 1]);
          drawLine(a.x, a.y, newpointB.x, newpointB.y);
        }
        else if(codeB == 0)
        {
          // point B is inside, point A is outside
          point newpointA;
          if((codeA & MASK[c]) != 0)
            newpointA = crossProduct(a, b, rectangle[i], rectangle[i + 1]);
          else
            newpointA = crossProduct(a, b, rectangle[j], rectangle[j + 1]);
          drawLine(newpointA.x, newpointA.y, b.x, b.y);
        }
      }
    }
  }
```

Κώδικας Α.12: Ο κώδικας του αλγορίθμου Skala 2005 (κυρτό πολύγωνο)

```
struct point { float x, y; };

point crossProduct(point e1, point e2, point p1, point p2)
{
  // coefficients of the edge e1 --> e2 (equation: a1*x + b1*y + c1 = 0)
  float a1, b1, c1;
  a1 = e1.y - e2.y;
  b1 = e2.x - e1.x;
  c1 = e1.x * e2.y - e2.x * e1.y;

  // coefficients of the line segment p1 --> p2 (equation: a2*x + b2*y + c2 = 0)
  float a2, b2, c2;
  a2 = p1.y - p2.y;
  b2 = p2.x - p1.x;
  c2 = p1.x * p2.y - p2.x * p1.y;

  // 2 equations, 2 unknowns (x,y), solve the system with determinants
  float d, dx, dy;
  d = a1 * b2 - a2 * b1;
  dx = -c1 * b2 + c2 * b1;
  dy = -a1 * c2 + a2 * c1;

  return { dx/d, dy/d };
```

```
}

int outcode(point p, float xmin, float ymax, float xmax, float ymin)
{
  int code = 0;

  if(p.x < xmin)
    code = 8;
  else if (p.x > xmax)
    code = 4;

  if(p.y < ymin)
    code |= 1;
  else if (p.y > ymax)
    code |= 2;

  return code;
}

void calculate_tab_values(point a, point b, point *polygon, int *tab)
{
  // Coefficients of the line Ax + By + C = 0
  float A = a.y - b.y;
  float B = b.x - a.x;
  float C = a.x * b.y - a.y * b.x;

  // Check each vertex against line and assign 0 (left) or 1 (right)
  int vertex[N + 1];
  for (int i = 0, power = 1; i < N; i++, power *= 2)
    vertex[i] = polygon[i].x * A + polygon[i].y * B + C < 0;
  vertex[N + 1] = vertex[0];

  // Calculate TAB values based on the alternations of digits 0 and 1
  int k = 0;
  for(int i = 0; i < N; i++)
    if(vertex[i] != vertex[i + 1])
      tab[k++] = i;
}

void sclip(point &a, point &b, point *polygon)
{
  // Calculate TAB1 & TAB2 values
  int tab[2] = {-1, -1};
  calculate_tab_values(a, b, polygon, tab);

  if(tab[0] != -1 && tab[1] != -1)
  {
    // Calculate intersection points between the line and the edges of the clipping area
    point inter1 = crossProduct(a, b, polygon[tab[0]], polygon[tab[0] + 1]);
    point inter2 = crossProduct(a, b, polygon[tab[1]], polygon[tab[1] + 1]);

    // Define coordinates of the rectangle that is formed by the intersection points
    float xmin = min(inter1.x, inter2.x);
    float ymax = max(inter1.y, inter2.y);
    float xmax = max(inter1.x, inter2.x);
    float ymin = min(inter1.y, inter2.y);

    // Define rectangle as the new clipping area
    point rectangle[] = { {xmin, ymin}, {xmax, ymin}, {xmax, ymax}, {xmin, ymax} };

    // Calculate the outcode for each point (Cohen-Sutherland based, 9 regions)
    int codeA = outcode(a, xmin, ymax, xmax, ymin);
    int codeB = outcode(b, xmin, ymax, xmax, ymin);

    if( (codeA | codeB) == 0)
    {
      // Same outcodes means that the line segment is inside the same region, draw and exit
      draw_line(a, b);
      return;
    }
```

```
    // Check the line and find the "t" parameters (scalar) of each intersection point
    float t1, t2;
    if(a.x != b.x)
    {
      t1 = (inter1.x - a.x) / (b.x - a.x);
      t2 = (inter2.x - a.x) / (b.x - a.x);
    }
    else
    {
      t1 = (inter1.y - a.y) / (b.y - a.y);
      t2 = (inter2.y - a.y) / (b.y - a.y);
    }

    // Clip the line
    if((t1 >= 0 && t1 <= 1) && !(t2 >= 0 && t2 <= 1))
    {
      if(t2 < 0)
        draw_line(inter1, a);
      else
        draw_line(inter1, b);
    }
    else if(!(t1 >= 0 && t1 <= 1) && (t2 >= 0 && t2 <= 1))
    {
      if(t1 < 0)
        draw_line(inter2, a);
      else
        draw_line(inter2, b);
    }
    else if(((t1 >= 0 && t1 <= 1) && (t2 >= 0 && t2 <= 1)))
    draw_line(inter1, inter2);
  }
}
```

Κώδικας Α.13: Ο κώδικας του αλγορίθμου S-Clip E^2 σε κυρτό πολύγωνο

```
//  INPUT: arbitary point P, line endpoints A & B
// OUTPUT: cross product (clockwise order)
//         > 0 : left side
//         < 0 : right side
//         = 0 : on the line

float cross_product(point P, point A, point B)
{
  return (B.x - A.x) * (P.y - A.y) - (B.y - A.y) * (P.x - A.x);
}
```

Κώδικας Α.14: Κώδικας για τον υπολογισμό του εξωτερικού γινομένου

```
point intersection(point A, point B, point C, point D)
{
  // calculate the intersection point between lines AB and CD
  point d1 = {B.x - A.x, B.y - A.y};
  point d2 = {D.x - C.x, D.y - C.y};
  float n1 = C.x * d2.y - C.y * d2.x;
  float n2 = A.x * d1.y - A.y * d1.x;
  float n3 = 1/(d2.y * d1.x - d1.y * d2.x);
  float x = (n1 * d1.x - n2 * d2.x) * n3;
  float y = (n1 * d1.y - n2 * d2.y) * n3;
  return {x, y};
}
```

Κώδικας Α.15: Κώδικας για τον υπολογισμό των σημείων τομής μεταξύ δύο ευθύγραμμων τμημάτων

```
//  INPUT : point A, point B, N, point polygon[N + 1] (last vertex is equal to first)
// OUTPUT : clipped line segment from point A to point B

float sideA, sideB;
bool draw = true;

for(int i = 0; i < N; i++)
{
  // sideX > 0 --> LEFT
  // sideX < 0 --> RIGHT
  // sideX = 0 --> ON THE EDGE
  sideA = cross_product(A, polygon[i], polygon[i + 1]);
  sideB = cross_product(B, polygon[i], polygon[i + 1]);

  if(sideA > 0 && sideB > 0)
  {
    // line is completely outside
    draw = false;
    break;
  }
  else if(sideA > 0 && sideB <= 0)
  {
    // point A is outside, point B is inside polygon or on the edge
    A = intersection(A, B, polygon[i], polygon[i + 1]);
  }
  else if(sideB > 0 && sideA <= 0)
  {
    // point B is outside, point A is inside polygon or on the edge
    B = intersection(A, B, polygon[i], polygon[i + 1]);
  }
}

if(draw)
  draw_line(A, B);
```

Κώδικας Α.16: Ο κώδικας του προτεινόμενου αλγορίθμου (Convex)

```
const int INSIDE = 0; // 000000
const int LEFT = 1;   // 000001
const int RIGHT = 2;  // 000010
const int BOTTOM = 4; // 000100
const int TOP = 8;    // 001000
const int BACK = 16;  // 010000
const int FRONT = 32; // 100000

int compute_outcode(float x, float y, float z)
{
  int code = 0;

  if(x > XMAX)
    code |= RIGHT;
  else if(x < XMIN)
    code |= LEFT;

  if(y > YMAX)
    code |= TOP;
  else if(y < YMIN)
    code |= BOTTOM;

  if(z > ZMAX)
    code |= FRONT;
  else if(z < ZMIN)
    code |= BACK;

  return code;
}
```

```
void cs3d(float x0, float y0, float z0, float x1, float y1, float z1)
{
  bool accept = false, done = false;

  int outcode1 = compute_outcode(x0, y0, z0);
  int outcode2 = compute_outcode(x1, y1, z1);

  do
  {
    if(outcode1 == 0 && outcode2 == 0)
    {
      accept = true;
      done = true;
    }
    else if(outcode1 & outcode2)
    {
      done = true;
    }
    else
    {
      float x, y, z;
      int outcode_ex = outcode1 ? outcode1 : outcode2;

      if(outcode_ex & TOP)
      {
        x = x0 + (x1 - x0) * (YMAX - y0) / (y1 - y0);
        z = z0 + (z1 - z0) * (YMAX - y0) / (y1 - y0);
        y = YMAX;
      }
      else if(outcode_ex & BOTTOM)
      {
        x = x0 + (x1 - x0) * (YMIN - y0) / (y1 - y0);
        z = z0 + (z1 - z0) * (YMIN - y0) / (y1 - y0);
        y = YMIN;
      }
      else if(outcode_ex & RIGHT)
      {
        y = y0 + (y1 - y0) * (XMAX - x0) / (x1 - x0);
        z = z0 + (z1 - z0) * (XMAX - x0) / (x1 - x0);
        x = XMAX;
      }
      else if(outcode_ex & LEFT)
      {
        y = y0 + (y1 - y0) * (XMIN - x0) / (x1 - x0);
        z = z0 + (z1 - z0) * (XMIN - x0) / (x1 - x0);
        x = XMIN;
      }
      else if(outcode_ex & FRONT)
      {
        x = x0 + (x1 - x0) * (ZMAX - z0) / (z1 - z0);
        y = y0 + (y1 - y0) * (ZMAX - z0) / (z1 - z0);
        z = ZMAX;
      }
      else if(outcode_ex & BACK)
      {
        x = x0 + (x1 - x0) * (ZMIN - z0) / (z1 - z0);
        y = y0 + (y1 - y0) * (ZMIN - z0) / (z1 - z0);
        z = ZMIN;
      }

      if(outcode_ex == outcode1)
      {
        x0 = x;
        y0 = y;
        z0 = z;
        outcode1 = compute_outcode(x0, y0, z0);
      }
      else
      {
        x1 = x;
```

```
          y1 = y;
          z1 = z;
          outcode2 = compute_outcode(x1, y1, z1);
        }
      }
    }
    while (!done);

    if(accept)
      draw_line(x0, y0, z0, x1, y1, z1);
  }
```

Κώδικας Α.17: Ο κώδικας του αλγορίθμου Cohen-Sutherland 3D

```
  struct Point3D { float x, y, z; };

  float dotProduct(Point3D a, Point3D b)
  {
    return a.x * b.x + a.y * b.y + a.z * b.z;
  }

  void CyrusBeck3D(float x0, float y0, float z0, float x1, float y1, float z1
           float xmin, float xmax, float ymin, float ymax, float zmin, float zmax)
  {
    Point3D p1 = {x0, y0, z0};
    Point3D p2 = {x1, y1, z1};

    // Define normals of clipping planes
    Point3D n[] = { {-1, 0, 0}, {1, 0, 0}, {0, -1, 0}, {0, 1, 0}, {0, 0, -1}, {0, 0, 1} };

    // Define points on clipping planes
    Point3D p[] = { {xmin, 0, 0}, {xmax, 0, 0}, {0, ymin, 0}, {0, ymax, 0}, {0, 0, zmin}, {0, 0,
    zmax} };

    // Define line direction
    Point3D d = { p2.x - p1.x, p2.y - p1.y, p2.z - p1.z };

    float tEnter = 0, tLeave = 1;

    for (int i = 0; i < 6; ++i)
    {
      float ndotd = dotProduct(n[i], d);
      float ndotp1 = dotProduct(n[i], { p1.x - p[i].x, p1.y - p[i].y, p1.z - p[i].z });

      if (ndotd == 0)
      {
        if (ndotp1 > 0)
        return; // Line is parallel to the plane and outside
      }
      else
      {
        float t = -ndotp1 / ndotd;
        if (ndotd < 0)
        {
          if (t > tEnter) tEnter = t;
        }
        else
        {
          if (t < tLeave) tLeave = t;
        }
      }
    }

    if (tEnter <= tLeave)
    {
      x0 = p1.x + tEnter * d.x;
      y0 = p1.y + tEnter * d.y;
      z0 = p1.z + tEnter * d.z;
```

```
      x1 = p1.x + tLeave * d.x;
      y1 = p1.y + tLeave * d.y;
      z1 = p1.z + tLeave * d.z;

      draw_line(x0, y0, z0, x1, y1, z1);
    }
  }
```

Κώδικας Α.18: Ο κώδικας του αλγορίθμου Cyrus-Beck 3D

```
void LiangBarsky3D(float x0, float y0, float z0, float x1, float y1, float z1)
{
  float u1 = 0.0, u2 = 1.0;
  float p1, q1, p2, q2, p3, q3, p4, q4, p5, q5, p6, q6;
  float r1, r2, r3, r4, r5, r6;
  float xa, ya, za, xb, yb, zb;

  p1 = -(x1 - x0);  q1 = x0 - xmin;
  p2 =  (x1 - x0);  q2 = xmax - x0;
  p3 = -(y1 - y0);  q3 = y0 - ymin;
  p4 =  (y1 - y0);  q4 = ymax - y0;
  p5 = -(z1 - z0);  q5 = z0 - zmin;
  p6 =  (z1 - z0);  q6 = zmax - z0;

  if ((p1 == 0.0 && q1 < 0.0) ||
    (p2 == 0.0 && q2 < 0.0) ||
    (p3 == 0.0 && q3 < 0.0) ||
    (p4 == 0.0 && q4 < 0.0) ||
    (p5 == 0.0 && q5 < 0.0) ||
    (p6 == 0.0 && q6 < 0.0)
  )
  {
    /*Line is rejected*/
  }
  else
  {
    if (p1 != 0.0)
    {
      r1 = (float)q1 / p1;
      if (p1 < 0)
      u1 = r1 > u1 ? r1 : u1;
      else
      u2 = r1 < u2 ? r1 : u2;
    }
    if (p2 != 0.0)
    {
      r2 = (float)q2 / p2;
      if (p2 < 0)
      u1 = r2 > u1 ? r2 : u1;
      else
      u2 = r2 < u2 ? r2 : u2;

    }
    if (p3 != 0.0)
    {
      r3 = (float)q3 / p3;
      if (p3 < 0)
      u1 = r3 > u1 ? r3 : u1;
      else
      u2 = r3 < u2 ? r3 : u2;
    }
    if (p4 != 0.0)
    {
      r4 = (float)q4 / p4;
      if (p4 < 0)
      u1 = r4 > u1 ? r4 : u1;
      else
```

```
        u2 = r4 < u2 ? r4 : u2;
      }
      if (p5 != 0.0)
      {
        r5 = (float)q5 / p5;
        if (p5 < 0)
        u1 = r5 > u1 ? r5 : u1;
        else
        u2 = r5 < u2 ? r5 : u2;
      }
      if (p6 != 0.0)
      {
        r6 = (float)q6 / p6;
        if (p6 < 0)
        u1 = r6 > u1 ? r6 : u1;
        else
        u2 = r6 < u2 ? r6 : u2;
      }

      if (u1 > u2)
      {
        // line rejected
        return;
      }
      else
      {
        x0 = x0 + u1 * (x1 - x0);
        y0 = y0 + u1 * (y1 - y0);
        z0 = z0 + u1 * (z1 - z0);

        x1 = x0 + u2 * (x1 - x0);
        y1 = y0 + u2 * (y1 - y0);
        z1 = z0 + u2 * (z1 - z0);

        draw_line(x0, y0, z0, x1, y1, z1);
      }
    }
  }
```

Κώδικας Α.19: Ο κώδικας του αλγορίθμου Liang-Barsky 3D

```
  struct point3d { float x, y, z; };
  struct vec3d { float x, y, z; };
  struct triangle { point3d p1, p2, p3; };

  triangle CV[12]; // Clipping volume triangles

  vec3d create_vector(point3d a, point3d b)
  {
    return { b.x - a.x, b.y - a.y, b.z - a.z };
  }

  vec3d cross_product(vec3d a, vec3d b)
  {
    return { a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x };
  }

  float dot_product(vec3d a, vec3d b)
  {
    return { a.x * b.x + a.y * b.y + a.z * b.z };
  }

  bool is_inside_clipping_volume(point3d p)
  {
    bool is_inside = false;
    if(p.x >= XMIN && p.x <= XMAX)
    if(p.y >= YMIN && p.y <= YMAX)
    if(p.z >= ZMIN && p.z <= ZMAX)
```

```
    is_inside = true;
    return is_inside;
  }

  void kolingerova_clipping(float x0, float y0, float z0, float x1, float y1, float z1)
  {
    point3d x1, x2, x3, xa, xb;
    vec3d s2, s3, s, s1, n, n1;
    float p, q, t, sn, tp[2];
    int intersections = 0, i = 0;

    xa.x = x0;   xa.y = y0;    xa.z = z0;
    xb.x = x1;   xb.y = y1;    xb.z = z1;

    while(intersections < 2 && i < 12) //for(i = 0; i <= 11; i++)
    {
      x1 = CV[i].p1;
      x2 = CV[i].p2;
      x3 = CV[i].p3;

      s2 = create_vector(x2, x1);
      s3 = create_vector(x1, x3);
      s1 = create_vector(x1, xa);
      s = create_vector(xa, xb);

      n = cross_product(s2, s3);
      n1 = cross_product(s, s1);

      sn = dot_product(s, n);

      if(sn != 0)
      {
        p = -dot_product(s3, n1) / sn;
        q = -dot_product(s2, n1) / sn;
        t = -dot_product(s1, n) / sn;

        if(p >= 0 && p <= 1 && q >=0 && q <= 1 - p && t >= 0 && t <= 1)
        {
          tp[intersections++] = t;
          draw_point(xa.x + s.x * t, xa.y + s.y * t, xa.z + s.z * t);
        }
      }
      i++;
    }

    if(intersections == 2)
    {
      x0 = xa.x + s.x * tp[0];
      y0 = xa.y + s.y * tp[0];
      z0 = xa.z + s.z * tp[0];
      x1 = xa.x + s.x * tp[1];
      y1 = xa.y + s.y * tp[1];
      z1 = xa.z + s.z * tp[1];
      draw_line(x0, y0, z0, x1, y1, z1);
    }
    else if(intersections == 1)
    {
      if(is_inside_clipping_volume(xa))
      {
        x0 = xa.x;
        y0 = xa.y;
        z0 = xa.z;
        x1 = xa.x + s.x * tp[0];
        y1 = xa.y + s.y * tp[0];
        z1 = xa.z + s.z * tp[0];
        draw_line(x0, y0, z0, x1, y1, z1);
      }
      else if(is_inside_clipping_volume(xb))
      {
        x0 = xb.x;
```

```
        y0 = xb.y;
        z0 = xb.z;
        x1 = xa.x + s.x * tp[0];
        y1 = xa.y + s.y * tp[0];
        z1 = xa.z + s.z * tp[0];
        draw_line(x0, y0, z0, x1, y1, z1);
      }
    }
    else if(intersections == 0)
    {
      if(is_inside_clipping_volume(xa) && is_inside_clipping_volume(xb))
      {
        x0 = xa.x;
        y0 = xa.y;
        z0 = xa.z;
        x1 = xb.x;
        y1 = xb.y;
        z1 = xb.z;
        draw_line(x0, y0, z0, x1, y1, z1);
      }
    }
  }
```

Κώδικας Α.20: Ο κώδικας του αλγορίθμου Kolingerová 3D

```
  void clip_3d_line(float x[], float y[], float z[])
  {
    if( !((x[0] < XMIN && x[1] < XMIN) ||
    (x[0] > XMAX && x[1] > XMAX) ||
    (y[0] < YMIN && y[1] < YMIN) ||
    (y[0] > YMAX && y[1] > YMAX) ||
    (z[0] < ZMIN && z[1] < ZMIN) ||
    (z[0] > ZMAX && z[1] > ZMAX))
    )
    {
      float dx = x[1] - x[0];
      float dy = y[1] - y[0];
      float dz = z[1] - z[0];
      for(int i = 0; i <= 1; i++)
      {
        if(x[i] < XMIN)
        {
          y[i] = dy / dx * (XMIN - x[0]) + y[0];
          z[i] = dz / dx * (XMIN - x[0]) + z[0];
          x[i] = XMIN;
        }
        else if(x[i] > XMAX)
        {
          y[i] = dy / dx * (XMAX - x[0]) + y[0];
          z[i] = dz / dx * (XMAX - x[0]) + z[0];
          x[i] = XMAX;
        }

        if(y[i] < YMIN)
        {
          x[i] = dx / dy * (YMIN - y[0]) + x[0];
          z[i] = dz / dy * (YMIN - y[0]) + z[0];
          y[i] = YMIN;
        }
        else if(y[i] > YMAX)
        {
          x[i] = dx / dy * (YMAX - y[0]) + x[0];
          z[i] = dz / dy * (YMAX - y[0]) + z[0];
          y[i] = YMAX;
        }

        if(z[i] < ZMIN)
        {
```

```
            x[i] = dx / dz * (ZMIN - z[0]) + x[0];
            y[i] = dy / dz * (ZMIN - z[0]) + y[0];
            z[i] = ZMIN;
          }
          else if(z[i] > ZMAX)
          {
            x[i] = dx / dz * (ZMAX - z[0]) + x[0];
            y[i] = dy / dz * (ZMAX - z[0]) + y[0];
            z[i] = ZMAX;
          }
        }
      }
      draw_line(x[0], y[0], z[0], x[1], y[1], z[1]);
    }
```

Κώδικας Α.21: Ο κώδικας του προτεινόμενου αλγορίθμου (3D)