T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SANAL HEYKELTRAŞLIKTA OPTİMİZE EDİLMİŞ HASH-TEMELLİ OCTREE VERİ YAPISININ KULLANILMASI

DOKTORA TEZİ

Gülüzar ÇİT

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Doç. Dr. Cemil ÖZ**

Ocak 2015

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SANAL HEYKELTRAŞLIKTA OPTİMİZE EDİLMİŞ HASH-TEMELLİ OCTREE VERİ YAPISININ KULLANILMASI

DOKTORA TEZİ

Gülüzar ÇİT

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ

Bu tez 05/01/2015 tarihinde aşağıdaki jüri tarafından oybirliği ile kabul edilmiştir.

Prof. Dr. **Nevcihan DURU** **Jüri Başkanı**	**Prof. Dr.** **Ümit KOCABIÇAK** **Üye**	**Prof. Dr.** **İbrahim ÇİL** **Üye**
Doç. Dr. **Cemil ÖZ** **Üye**	**Yrd. Doç. Dr.** **Sıtkı ÖZTÜRK** **Üye**	

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Gülüzar ÇİT
05.01.2015

TEŞEKKÜR

Doktora çalışmam boyunca hoşgörü ve desteğini esirgemeyen değerli danışman hocam Doç. Dr. Cemil ÖZ'e teşekkür ederim.

Sanal Gerçeklik laboratuarında birlikte çalıştığım ve tez çalışmamın şekillenmesinde doğrudan katkısı bulunan asistan arkadaşım Kayhan AYAR'a teşekkür ederim.

Hayatımın bu uzun soluklu ve yorucu yolculuğunda bana destek, anlayış ve sabır gösteren eşime, bana yaşam enerjisi veren kızıma ve bu günlere gelmemde büyük emeği olan sevgili aileme teşekkürlerimi sunarım.

Ayrıca bu çalışmanın maddi açıdan desteklenmesine olanak sağlayan Sakarya Üniversitesi Bilimsel Araştırma Projeleri (BAP) Komisyon Başkanlığına (Proje No: 2010-50-02-010) teşekkür ederim.

İÇİNDEKİLER

TEŞEKKÜR i
İÇİNDEKİLER ii
SİMGELER VE KISALTMALAR LİSTESİ v
ŞEKİLLER LİSTESİ vi
TABLOLAR LİSTESİ ix
ÖZET x
SUMMARY xi

BÖLÜM 1.
GİRİŞ 1
1.1. Tezin Amacı 2
1.2. Tezde Geliştirilenler 4
1.3. Önerilen Sanal Heykeltraşlık Modeli 6
1.3.1. Donanım 7
1.3.2. Yazılım 8
1.3.2.1. Grafik Sahneleme Modülü 8
1.3.2.2. Simülasyon Modülü 9
1.3.2.3. Haptic Sahneleme Modülü 9
1.4. Tezin Organizasyonu 9

BÖLÜM 2.
LİTERATÜR ÇALIŞMASI 11
2.1. Sanal Heykeltraşlık 11

BÖLÜM 3.
GRAFİK SAHNELEME MODÜLÜ 17
3.1. Ön-İşleme 17
3.1.1. Vokselleştirme 18
3.1.1.1. Voksel 23
3.1.1.2. Kaplayan hacim hesaplama 25
3.1.1.3. Möller'in 3B üçgen/kutu kesişim testi 28
3.1.1.4. İç hacim doldurma 37
3.1.1.5. Voksel-hücre eşleştirme 42
3.1.1.6. Optimize edilmiş hash-temelli octree veri yapısı oluşturulması 47
3.1.2. Grafik sahneleme modülü 47
3.1.2.1. Marching Cubes 48

BÖLÜM 4.
OPTİMİZE EDİLMİŞ HASH-TEMELLİ OCTREE 54
4.1. Octree 55
4.2. Octree Yaklaşımları 57
4.2.1. Yukarıdan-aşağıya octree oluşturma 57
4.2.2. Aşağıdan-yukarıya octree oluşturma 59
4.3. Octree Çeşitleri 62
4.3.1. Doğrusal Octree 62
4.3.2. İşaretçi Temelli octree 62
4.3.3. Hash-temelli octree 64
4.3.3.1. Anahtar hesaplama 65
4.3.4. Optimize edilmiş hash-temelli octree 71

BÖLÜM 5.
SİMÜLASYON MODÜLÜ 74
4.1. Çarpışma Tespiti 74
4.2. Çarpışma Cevabı 76

BÖLÜM 6.
HAPTIC SAHNELEME MODÜLÜ 79
4.1. Sanal Araç..... 80
4.2. Geri-Besleme Kuvvetinin Hesaplanması..... 81

BÖLÜM 7.
SONUÇLAR VE ÖNERİLER 87

KAYNAKLAR 88
ÖZGEÇMİŞ 95

SİMGELER VE KISALTMALAR LİSTESİ

2B	: İki-boyutlu
3B	: Üç-boyutlu
AABB	: Axis-Aligned Bounding Box
B-rep	: Boundary representation
BSP	: Binary Search Partitioning / İkili uzay bölümleme
CSG	: Constructive Solid Geometry / Yapısal Katı Geometri
CT	: Computer Tomography / Bilgisayarlı Tomografi
dof	: Degree of Freedom / Serbestlik Derecesi
GPU	: Graphics Processing Unit
HCI	: Human-Computer Interaction / İnsan-Bilgisayar Etkileşimi
HIP	: Haptic Interaction Point / Haptic Etkileşim Noktası
kHz	: Kilo Hertz
MC	: Marching Cubes
MRI	: Magnetic Resonance Imaging / Manyetik Rezonans Görüntüleme
OBB	: Oriented Bounding Box
OpenGL	: Open Graphics Library / Açık Grafik Kütüphanesi
sn	: Saniye

ŞEKİLLER LİSTESİ

Şekil 1.1. Önerilen haptic-temelli sanal heykeltıraşlık sisteminin mimarisi..... 7
Şekil 1.2. Geomagic Phantom Omni Haptic Cihazı..... 8
Şekil 3.1. 2B tarama-dönüşümü örneği..... 19
Şekil 3.2. 3B üçgen kafes modelinden hacimsel ikili voksel veri kümesi elde etmek için gerçekleştirilen algoritmanın sözde kodu..... 22
Şekil 3.3. 4x4x4 3B ayrık voksel uzayı ve voksel..... 23
Şekil 3.4. 2B piksel ve 3B voksel komşuluğu (a) Ortadaki koyu renkli piksele N-komşu 2B pikseller kümesi (N Є {4,8}) (b) Ortadaki voksele N-komşu 2B vokseller kümesi (N Є {6,18,26})..... 24
Şekil 3.5. Temel kaplayan hacim şekilleri (a) küre, (b) eksen-hizalı kaplayan hacim, (c) yönlü kaplayan hacim ve (d) slab..... 25
Şekil 3.6. Üçgen için eksen-hizalı kaplayan hacim hesaplamayı gerçekleştiren algoritmanın sözde kodu..... 27
Şekil 3.7. Üçgen kafes modeli için AABB hesaplamayı gerçekleştiren algoritmanın sözde kodu..... 28
Şekil 3.8. Bir üçgen ve karenin x eksenine göre izdüşümleri arasındaki ayıran eksen..... 29
Şekil 3.9. Örnek 3B voksel ve üçgenin başlangıç ve taşınmış konumları..... 30
Şekil 3.10. Üçgenin normal düzlemi ile voksel kesişim testinin 2B gösterimi..... 31
Şekil 3.11. Üçgen kenarları ve ilgili voksel yüksekliğinin aij'ye göre örnek izdüşümlerinin 2B gösterimi..... 33
Şekil 3.12. Bir voksel ile üçgenin kesişimini test eden algoritmanın sözde kodu..... 34
Şekil 3.13. Çok-kanallı vokselleştirme modülünün çalışma prensibi..... 35
Şekil 3.14. xy voksel diliminin iç hacminin Feng ve Soon'un tohum-doldurma algoritmasına göre doldurulması..... 38

Şekil 3.15. 3B yüzey verisinin iç hacminin doldurulmasını hesaplayan algoritmanın sözde kodu 39
Şekil 3.16. Bir voksel ve eş-hücre..... 43
Şekil 3.17. Voksel-hücre eşleştirmesinin 2B gösterimi..... 43
Şekil 3.18. 3B hacim voksel ızgarasından eş-hücre ızgarasının elde edilmesini sağlayan algoritmanın sözde kodu..... 44
Şekil 3.19. Marching Cubes algoritmasında eş-yüzey oluşturmak için kullanılan 15 kalıp 49
Şekil 3.20. Örnek ikili voksel için indeks değeri hesaplama 50
Şekil 3.21. Örnek voksel için kenar değeri hesaplama 50
Şekil 3.22. Marching Cubes algoritmasının 2B örnek gösterimi..... 51
Şekil 3.23. Çok-kanallı MC modülünün çalışma prensibi..... 52
Şekil 4.1. 2B/3B (z)yx sıralamasına göre herhangi bir derinlikteki dolanım yönü ve yer kodları..... 56
Şekil 4.2. 8x8'lik örnek bir ikili resimden yukarıdan-aşağıya yaklaşımı ile quadtree oluşturulması 58
Şekil 4.3. 8x8'lik örnek bir ikili resimden aşağıdan-yukarıya yaklaşımı ile quadtree oluşturulması 59
Şekil 4.4. 8x8'lik örnek resim için hiyerarşik quadtree gösterimi..... 61
Şekil 4.5. 8x8'lik örnek ikili resimden işaretçi-temelli quadtree oluşturulması (a) Örnek 8x8'lik ikili resim (b) ters Z dolanım yönüne göre etiketlenmiş quadtree alt uzayları (c) işaretçi-temelli quadtree gösterimi 64
Şekil 4.6. 8x8'lik örnek ikili resmin ters Z dolanım yönü kullanılarak uzay/alt uzaylarının ilgili yer kodlarından anahtar değerlerinin hesaplanması (a) Örnek 8x8'lik ikili resim (b) ters Z dolanım yönüne göre ilgili yer kodlarından anahtar değerleri hesaplanmış quadtree uzayı ve tüm alt uzayları (c) işaretçi-temelli quadtree gösterimi..... 66
Şekil 4.7. 8x8'lik örnek ikili resim üzerindeki bir pikselin ters Z dolanım yönü kullanılarak anahtar değerinin hesaplanması (a) Örnek 8x8'lik ikili resim ve anahtarı hesaplanacak piksel (b) ilgili pikselin ters Z dolanım yönüne göre yer kodlarından anahtar değerlerinin hesaplanması (c) ilgili pikselin

uzaydaki konumuna göre anahtar değerinin hesaplanması (d) ilgili pikselin işaretçi-temelli quadtree gösterimindeki konumu 68
Şekil 4.8. 8x8'lik örnek ikili resime ait hash-temelli octree veri yapısı oluşturulması (a) Örnek 8x8'lik ikili resim (b) ters Z dolanım yönüne göre ilgili yer kodlarından anahtar değerleri hesaplanmış quadtree uzayı ve tüm alt uzayları (c) hash-temelli quadtree gösterimi 70
Şekil 4.9. 8x8'lik örnek ikili resime ait optimize edilmiş hash-temelli octree veri yapısı oluşturulması (a) Örnek 8x8'lik ikili resim (b) ters Z dolanım yönüne göre ilgili yer kodlarından anahtar değerleri hesaplanmış quadtree uzayı ve tüm alt uzayları (c) optimize edilmiş hash-temelli quadtree gösterimi 72
Şekil 5.1. 4x4 çözünürlüklü örnek resimde sanal araç ile çarpışan pikselin tespit edilmesi..... 75
Şekil 5.2. 4x4'lük 2B voksel ızgarasının (a) çarpışmadan önce ve (b) çarpışmadan sonraki eş-yüzey ve optimize-edilmiş hash-temelli quadtree gösterimi.... 78
Şekil 6.1. Haptic Etkileşim Noktası (a) Gerçek dünya (b) Sanal dünya..... 80
Şekil 6.2. Sanal aracın çalışma uzayı ile 3B uzayın dünya uzayının eşleştirilmesi... 81
Şekil 6.3. Geri-besleme kuvvetinin hesaplanması 82
Şekil 7.1. İşaretçi-temelli ve optimize edilmiş hash-temelli octree veri yapıları kullanılarak saklanan 3B tavşan modeli üzerinde aynı deliğin oluşturulması sırasında geçen toplam gerçek-zaman dolanımı ve eş-yüzeyin bölgesel olarak yeniden oluşturulma süreleri 86
Şekil 7.2. Önerilen haptic-temelli sanal heykeltıraşlık sisteminin arayüzü 87

TABLOLAR LİSTESİ

Tablo 3.1. Farklı kanal sayılarına ve çözünürlüğe göre vokselleştirme süreleri..... 36
Tablo 3.2. Farklı kanal sayılarına ve çözünürlüğe göre voksel yüzey modelinin iç hacminin doldurulma süreleri 41
Tablo 3.3. Farklı kanal sayılarına ve çözünürlüğe göre MC süreleri..... 53
Tablo 7.1. İşaretçi-temelli hash-temelli ve optimize edilmiş hash-temelli octree veri yapıları için gerekli düğüm sayıları ve ilgili oluşturma süreleri 84
Tablo 7.2. İşaretçi-temelli ve optimize edilmiş hash-temelli octree veri yapıları için Marching Cubes uygulanmasının farklı çözünürlüklerdeki ön-işleme boyunca geçen süreleri..... 85

ÖZET

Anahtar kelimeler: Sanal Heykeltaşlık, Haptics, Vokselleştirme, Octree, Hashing

Sanal heykeltıraşlık, kullanıcılara sanat, tasarım ve hızlı prototipleme alanlarında sanal gerçeklik donanım ve yazılımları tarafından sağlanan bir sanal dünya içerisinde yeni 3B katı nesne oluşturma veya mevcut nesneleri değiştirebilme imkânı sağlayan 3B bir modelleme işlemidir. Bu tez çalışmasında haptic kuvvet geri-beslemesi ile voksel-temelli bir sanal heykeltıraşlık uygulaması için optimize edilmiş hash-temelli bir octree veri yapısı kullanımı önerilmektedir. Amaç, hacim voksel veri kümesini saklamak için gerekli hafızayı ve hesaplama maliyetini düşürmek, aynı zamanda da gerçek-zamanda etkileşim esnasında gerçekleşen octree kullanımından doğan ağaç dolanım süresini azaltarak model yüzeyinin lokal olarak yeniden oluşturulma süresini kısaltmaktır.

İlk önce, üzerinde çalışılan sanal ham maddeye ait hacim verisi daha az hafıza kullanarak saklayabilmek ve gerçek-zamanda yontabilmek amacıyla vokselleştirilerek optimize-edilmiş hash-temelli bir octree veri yapısına dönüştürülmektedir. Daha sonra, küre olarak tasarlanan sanal araç tarafından heykelin hangi voksellerine dokunulduğu belirlenerek bu vokseller veri yapısından çıkarılmaktadır. Sonuç verisine gerçekçi bir görüntü verebilmek amacıyla üçgen kafes modelini yeniden oluşturmak için Marching Cubes algoritması kullanılmaktadır. Tüm hacim için hesaplama maliyeti yüksek olduğundan dolayı bu çalışmada sadece yontma işleminden sonra modifiye edilen vokseller tarafından etkilenen eş yüzey yeniden hesaplanarak lokal güncelleme gerçekleştirildi.

İkinci olarak, önerilen sanal heykeltıraşlık sistemine, sanal yontma aracının üç-boyutlu kontrolünü sağlamak ve kullanıcıların yontma işlemi sırasında heykel üzerinde uyguladıkları kuvvete karşı meydana gelen direnci hissedebilmelerine imkan sağlamak amacıyla bir haptic cihazı yolu ile haptic kuvvet-geri beslemesi entegre edilmiştir.

Sanal heykeltıraşlıkta hafıza optimizasyonu ve gerçek-zaman etkileşimi üzerine odaklanılan bu çalışmada, önerilen optimize edilmiş hash-temelli octree veri yapısının performansını test etmek amacıyla hafıza maliyetleri ve çalışma süreleri, işaretçi-temelli ve hash temelli veri yapıları ile karşılaştırılmıştır. Sonuç olarak, bu yeni optimize edilmiş hash-temelli octree veri yapısının hem ön-işleme zamanında hem de gerçek zamanda hafıza maliyetleri ve çalışma sürelerindeki düşüşler gösterilmiştir.

USING AN OPTIMIZED HASH-BASED OCTREE DATA STRUCTURE IN VIRTUAL SCULPTING

SUMMARY

Keywords: Virtual Sculpting, Haptics, Voxelization, Octree, Hashing

Virtual sculpting is a 3D modelling process which allows users to create new 3D solid models or modify existing objects provided by virtual reality software and hardware in art, design and rapid prototyping areas. In this thesis, an optimized hash-based octree data structure in a voxel-based virtual sculpting application with haptic force feedback is proposed. The goal is to reduce the memory and computation costs to store volumetric voxel dataset and also to reduce the local surface reconstruction times of the model by decreasing tree traversal time caused by octree during real-time interaction.

First, in order to store with less memory and carve in real-time, volumetric data of virtual workpiece is converted into an optimized hash-based octree data structure by voxelizing them. Then, voxels collided with the carving tool that is designed as a sphere are removed from this data structure. Marching Cubes algorithm is used to reconstruct the triangular mesh model in order to give a realistic display of the voxel data. Since the computational cost is very high for the whole volume, in this study, local update is performed by reconstructing the isosurface affected from the modified voxels after carving process.

Afterwards, by the way of a haptic device, a haptic force feedback is integrated in the proposed virtual sculpting application in order to provide 3D control of the virtual tool and allow to feel the resistance against the applied force on the sculpture object.

This study focused on memory optimization and real-time interaction, memory costs and runtimes of the proposed optimized hash-based octree data structure are compared with the pointer-based and hash-based ones in order to test the performance. Consequently, memory cost and working time decreases on both pre-processing and runtime of this new optimized hash-based octree data structure are shown.

BÖLÜM 1. GİRİŞ

Mühendisler ve tasarımcılar üç boyutlu modelleme ve tasarım yapmak için sayısal bilgisayarları kullanmaktadırlar. 90'lı yılların başından itibaren bilgisayarların hız ve kapasiteleri sadece artmakla kalmamış aynı zamanda maliyetleri de kayda değer bir oranda azalmıştır. Bu teknolojik gelişme ile birlikte sanatçılar ve tasarımcılar çalışmalarını üç-boyutlu (3B) modelleme yazılımları kullanarak yapmaya doğru yönelmişlerdir. Ancak, günümüz 3B modelleme araçları, sanal gerçeklik ortamları tarafından sağlanan gerçeklik ve etkileşim yeteneğine sahip değillerdir. Bunun için de, deneyimli bir tasarımcının gerçek bir fiziksel ortamda yaptığı gibi sanal araçlar kullanarak tasarım modelleri oluşturmasına imkân veren sanal gerçeklik ortamları geliştirilmiştir. Sanal gerçekliğin bu uygulama alanına da genel olarak "Sanal Heykeltıraşlık" adı verilmiştir.

Zhang sanal heykeltıraşlığı, gerçek bir heykeltraşın kil, balmumu veya bir tahta parçası üzerinde yapabildiklerini kullanıcının bilgisayar ekranındaki bir çalışma parçasını oyarak etkileşimli olarak üç boyutlu modeller oluşturabilmesi işlemi olarak tanımlamaktadır [1].

Başka bir şekilde ifade edecek olursak, sanal heykeltıraşlık, sanatsal çalışmalar yapmak veya hızlı bir şekilde prototip modeller oluşturmak için kullanıcıların sanal gerçekliğin sunduğu sanal bir dünyada üç-boyutlu nesneler oluşturmalarını mümkün kılan etkileşimli bir modelleme işlemidir. Sanal heykeltıraşlık, aynı zamanda, mühendislik, tasarım ve mimari alanlarında kendini ispatlamış olan geleneksel tasarım programlarının aksine, kullanıcıların hiçbir özel eğitim ve uzmanlık gerektirmeksizin 3B modelleri sezgisel olarak değiştirebilecekleri serbest-formlu tasarım imkânı sağlar. Geleneksel tasarım programlarının bir diğer kısıtı da

tasarımcıların üç-boyutlu nesneleri oluşturmak için genellikle mouse, vb. gibi iki boyutlu giriş cihazlarını kullanmak zorunda kalmalarıdır.

Bir sanal heykeltıraşlık sisteminin temel amacı, tasarımcıların sanal gerçeklik donanım ve yazılımları ile donatılmış etkileşimli bir ortam yolu ile üç-boyutlu serbest-formlu nesneleri oluşturmalarına ve yeniden şekillendirebilmelerine imkân sağlamaktır. Bu amaçla, bir sanal heykeltıraşlık sistemi kullanıcıya sanal ortamdaki gerçekçi geri beslemeyi haptic cihazları (örneğin, veri eldiveni), başa takılan cihaz ve aktif gözlük gibi sanal heykeltıraşlık donanımları yolu ile iletir. Örneğin, kullanıcıların sanal aracın model üzerine uyguladığı kuvvet sonucu meydana gelen direnci hissedebilmesi, sisteme bir haptic cihazı eklenmesi yolu ile gerçekleştirilebilir. Yunanca “haptesthai“ kelimesinden gelen ve “dokunma duyusu ile ilgili” anlamına gelen “haptics”, kullanıcıların sanal ortamlardaki nesneleri hissetmesine imkân sağlayan kuvvet geri-beslemesi teknolojisi ile ilgili bir mühendislik alanıdır [2]. Ayrıca, bu cihazlar ve kullanıcı arasındaki etkileşim uygulama yazılımları yolu ile gerçekleştirilir.

Literatürde tasarım ve modelleme alanlarında eğitimi amaçlayarak önerilen sanal heykeltıraşlık çalışmaları mevcuttur [3-16]. Aynı zamanda, sanal tıp alanındaki cerrahi ve diş simülasyonlarında eğitim amacı ile geliştirilmiş uygulamalar da bilim literatüründe kendisine yer edinmiştir [17-22].

1.1. Tezin Amacı

Bir sanal heykeltıraşlık çalışması, sanal çalışma parçasına madde ekleme şeklinde yapılabileceği gibi sanal araç ile hammaddeyi yontma olmak üzere temel olarak iki farklı şekilde yapılabilir. Bu çalışmada önerilen heykeltıraşlık uygulaması, sanal araç ve çalışma parçası arasında Boolean fark işlemleri kullanılarak yapılmaktadır.

Üç-boyutlu bir nesneden haptic cihazı kullanarak etkileşimli olarak yontma yani hacim çıkartma işlemini gerçekleyen sanal heykeltıraşlık sistemi için amaçlanan temel katkılar aşağıdaki gibidir:

Gerçekçi görüntü: Yontma işleminin sanal çalışma parçası ve haptic cihazı üzerindeki etkileri gerçekçi olarak görüntülenmelidir. Örneğin, sanal model üzerinde yumuşak bir darbe yapıldı ise oluşan sonuç yüzey görüntüsü ve aynı zamanda kullanıcıya yansıtılan geri-besleme kuvveti de buna uygun olmalıdır.

Gerçek-zaman performansı: Yontma işleminin etkileri kullanıcıya gerçek-zamanlı olarak kullanıcının gözüne ve eline olmak üzere iki şekilde yansıtılmalıdır. Göze yansıtılan görüntünün güncellenmesi, insan gözünün bunu gerçekçi olarak algılayabilmesi için saniyenin 1/24 ünden daha az animasyon hızlarında gerçeklenmelidir. Haptic-temelli bir sistemde ele yansıtılan kuvvet ile ilgili temel sorun ise sert yüzeylerin gerçekçi algısını mümkün kılmak için yaklaşık olarak 1 kHz haptic yenileme hızı (kuvvetleri haptic cihazlarına gösterildiği hız) sağlamaktır. Bu hızın altındaki örnekleme etki kuvvetinin titreşmesi veya istenilmeyen şekilde yumuşak hissedilmesi ile sonuçlanır.

Hafıza-optimizasyonu: Sanal heykeltıraşlık uygulamalarında çok büyük hacim verileri üzerinde çalıştığından dolayı bu verilerin hafızaya yükledikleri maliyetin azaltılması gerekir.

Doğruluk: Sanal heykeltıraşlık çalışmalarında, sanal yontma aracının yontulacak nesneye çarpması sonucu sanal nesneden çıkartılacak hacim miktarının gerçekte çıkartılması muhtemel hacim miktarına mümkün olduğunca yakın olması beklenir.

Bu çalışmanın temel amacı, hafıza gereksinimi ve hesaplama maliyetini düşürmek için optimize edilmiş hash-temelli yeni bir octree veri yapısı kullanan gerçek-zamanlı bir sanal heykeltıraşlık sistemi geliştirmektir. 3B giriş modelimiz kapalı üçgen ızgara modelidir ve başlangıçta hacimsel veri kümesini elde etmek için modelin yüzeyi ve iç hacmi vokselleştirilir. Daha sonra, 3B hacimsel veri kümesini saklamak için

gerekli hafızayı azaltan octree veri yapısı kullanılır ve ağaçtaki bir düğüme ulaşmak için gerekli dolanım sürelerini kısaltmak için ise octree veri yapısı hash-temelli olarak optimize edilmiş yeni bir yaklaşımla oluşturulur. Sanal çalışma parçasından madde çıkarıldıktan sonra, en popüler eş yüzey oluşturma algoritması olarak bilinen Marching Cubes algoritması [24] kullanılarak sanal modelin yüzeyi bölgesel olarak yeniden oluşturulur. Kullanıcıya, gerçek hayattaki heytkeltraşlık uygulamalarında eli ile model arasında hissettiği geri-besleme kuvveti ise sanal ortamda bir haptic cihazı yolu ile hissettirilir.

1.2. Tezde Geliştirilenler

Bu çalışmada önerilen sanal heykeltıraşlık sistemi geliştirilirken, voksel-temelli sanal heykeltıraşlık, görüntüleme ve haptic alanlarında katkılar yapıldı. Voksel-temelli heykeltıraşlık için üçgen kafeslerden oluşan giriş modelleri kullanıldı ve başlangıçta sanal çalışma parçasına ait voksel-temelli hacimsel veri kümesini elde etmek için modelin yüzeyi ve iç hacmi vokselleştirildi. Möller'in tek üçgen ve kutu arasındaki kesişimi test etmek için geliştirdiği üçgen/kutu kesişim testi algoritması [25,26] yüksek doğruluk elde etmek için üçgen yüzey modeline adapte edildi. Sanal modelin iç hacminde yer alan vokseller ise tohum-doldurma olarak adlandırılan algoritma kullanılarak belirlendi [27]. Voksel-temelli 3B hacim verisi elde etmek amacıyla gerçekleştirilen yüzey ve hacim vokselleştirmedeki yüksek hesaplama maliyetinden doğan süreyi kısaltmak için çok-kanallı programlama kullanıldı [28].

Sanal heykelin voksel-temelli hacim verisini saklamak için üç-boyutlu ayrık vokseller kümesi yerine gerekli hafıza maliyetini düşürmek amacıyla uzayın yinelemeli olarak alt bölümlenmesi sonucu elde edilen octree veri yapısı kullanıldı. Önerilen çalışmadaki temel katkı, 3B octree-temelli hacimsel voksel veri kümesini saklamak için geleneksel octree veri yapısındaki işaretçi-temelli veri yapısı yerine optimize edilmiş hash-temelli bir octree veri yapısı kullanılmasıdır. Optimize edilmiş bu yeni yaklaşım ile octree veri yapısını oluşturmak için gerekli hafıza maliyeti ve hesaplama zamanı diğer octree gösterimlerine kıyasla düşürüldü.

Daha sonra, sanal yontma aracının herhangi bir oryantasyonda iken heykele uygulanması sağlandı. Buradaki problemin karmaşıklığı, gerçek-zaman performansını yakalayabilmek için belirli bir zaman aralığında sanal araç tarafından heykelin hangi voksellerine temas edildiğinin belirlenmesidir. Bu çalışmada kullanılan sanal araç basit bir küre olarak belirlenmiştir. Böylece problemin karmaşıklığı sanal araç ve heykel vokselleri arasındaki kaplayan-küre veya eksen-hizalı kutu gibi hızlı çarpışma tespiti testlerine indirgenerek etkin olarak azaltılır.

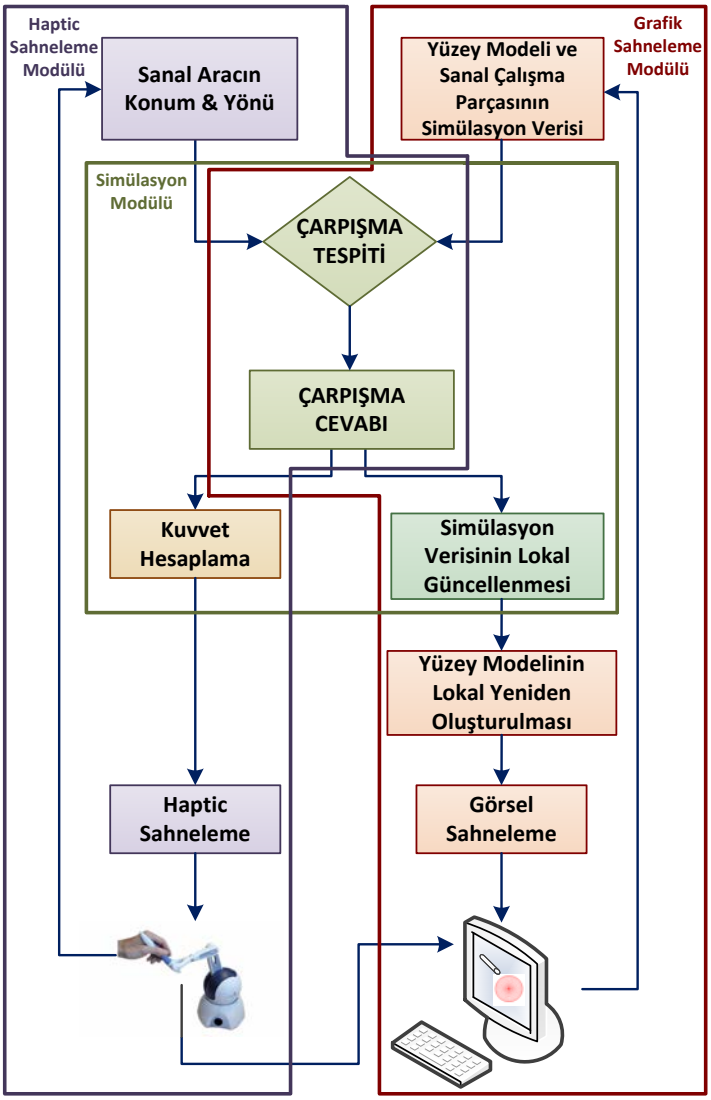
Heykelin hacimsel gösteriminin yüzeyini voksel verisinin pürüzlü ve aliased görüntüsü yerine daha gerçekçi ve estetik olarak yeniden görüntülemek için ilk olarak Lorenson ve Cline tarafından geliştirilen Marching Cubes algoritması uygulandı [24]. Hesaplama maliyeti oldukça yüksek olan ve maliyeti çözünürlüğe bağlı olarak arttığı algoritmanın gerçek-zamanda uygulanabilirliğini sağlamak için algoritmanın çalışmasına iki temel katkı yapıldı. Birinci olarak, Marching Cubes algoritmasının voksel komşuluğu prensibine göre sekiz komşu vokselin değerlerine bakarak üçgen modelleri yeniden oluşturduğundan dolayı octree veri yapısındaki ağaç dolanım maliyetini ekarte etmek amacıyla ilgili sekiz komşu voksel ön-işleme zamanında birleştirilerek her bir vokselden bir eş-hücre oluşturuldu. İkinci olarak ise, algoritma gerçek-zamanda sadece çarpışmanın gerçekleştiği, yani eşyüzey değişiminin olacağı hacme bölgesel olarak uygulandı. Çalışmada önerilen optimize-edilmiş hash-temelli yeni octree veri yapısının kullanımı ile birlikte etkileşim sırasındaki çarpışma tespiti için gerekli ağaç dolanım süresi ve dolayısıyla da yüzeyin bölgesel olarak yeniden oluşturulma süresi kısaltıldı. Bunların yanı sıra, gerçekçi ve hızlı görüntüleme amacıyla Marching Cubes ile eşyüzeyin sahnelenmesi ve ışıklandırılması GPU (Graphics Processig Unit) destekli grafik donanımlarından yararlanıldı. Ayrıca, ön-işleme zamanında eş-yüzeyin yeniden oluşturulması sırasında geçen süreyi kısaltmak için çok-kanallı programlama kullanıldı [29].

Önerilen sanal heykeltıraşlık çalışmasında kullanıcıların dokundukları sanal model üzerinde uyguladıkları basıncı hissetmelerini sağlayan kuvvet geri-beslemesi, bir haptic cihazı kullanımı yolu ile sağlandı.

Tüm bunların yanı sıra, sanal heykeltıraşlık sisteminin farklı bileşenlerinde çok-kanallı mimarinin kullanımı ile önerilen sanal heykeltıraşlık sisteminin çalışma sürelerinin kısaltılması sağlandı.

1.3. Önerilen Sanal Heykeltraşlık Modeli

Bu çalışmada önerilen haptic-temelli sanal heykeltıraşlık sistemi, yazılım ve donanım olmak üzere iki temel kısımdan oluşmaktadır. Donanım kısmı yüksek kapasiteli ekran kartı ile donatılmış bir bilgisayar ve bir haptic cihazından oluşmaktadır. Yazılım kısmı ise haptic sahneleme modülü, grafik sahneleme modülü ve simülasyon modülü olmak üzere üç ana modülden oluşmaktadır. Şekil 1.1'de bu çalışmada önerilen haptic-temelli sanal heykeltıraşlık sistem mimarisinin blok diyagramı gösterilmektedir. Sistem bileşenlerinin detayları sonraki iki bölümde detaylı olarak açıklanmıştır.



Şekil 1.1. Önerilen haptic-temelli sanal heykeltıraşlık sisteminin mimarisi

1.3.1. Donanım

Önerilen haptic-temelli sanal heykeltıraşlık sisteminde kullanılan yüksek kapasiteli ekran kartı ile donatılmış olan bilgisayar haptic cihazı ve sistem yazılımı arasındaki iletişimi sağlar. Haptic cihazı, Şekil 1.2'de görülen Geomagic Phantom Omni cihazıdır. Bu altı-eklemli haptic cihazı benzer amaçla kullanılan diğer cihazlara kıyasla daha ekonomiktir. Ayrıca, uygulanmasının kolay olması da bu cihazın seçilmesinin başlıca nedenleridir.

Şekil 1.2. Geomagic Phantom Omni Haptic Cihazı

Haptic cihazının ucu sistemde yontma için kullanılan sanal aracı yönetir ve sanal aracın ucu sanal heykel nesnesinin yüzeyine temas ettiğinde, bir geri besleme sinyali hesaplanır ve bu sinyale bağlı oluşan etkileşimli kuvvet haptic cihazı yolu ile kullanıcıya iletilir.

1.3.2. Yazılım

Bu çalışmada önerilen sanal heykeltıraşlık sisteminin yazılım kısmı grafik sahneleme modülü, simülasyon modülü ve haptic sahneleme modülü olmak üzere üç farklı modülden oluşmaktadır.

Önerilen haptic-temelli sanal heykeltıraşlık sisteminin simülasyon yazılımı C++ platformunda yazılırken, sanal sahne ve tüm elemanlarını içeren grafik yazılımı OpenGL, haptic yazılımı ise OpenHaptics Toolkit kullanılarak yazılmıştır.

1.3.2.1. Grafik sahneleme modülü

Grafik sahneleme modülünde çalışma uzayı olarak adlandırılan 3B sanal sahnenin, yontma işlemini gerçekleştiren sanal aracın ve çalışma parçası olarak adlandırılan yontulacak nesnenin sahnelenmesi esas alınır. Çalışma uzayının, sanal aracın ve çalışma parçasının renk, sertlik, sürtünme gibi fiziksel özellikleri de aynı zamanda modülün başlangıç aşamadınsa tanımlanır.

1.3.2.2. Simülasyon modülü

Hem grafik hem de haptic sahneleme modülleri simülasyon modülü tarafından harekete geçirilir/tetiklenir. Haptic cihazının miline bağla olan sanal araç, heykel nesnesinin yüzeyi ile çarpışır çarpışmaz, bu modül harekete geçer. "Çarpışma" olarak adlandırılan bu "etki" tespit edildiğinde, hem grafik hem de haptic modüllerinde bu çarpışmaya bir "tepki" meydana gelir. İlk önce, simülasyon modülünde sanal araç ve model arasında meydana gelen çarpışmadan etkilenen lokal voksel alanı hesaplanır ve etkilenen bu vokseller optimize-edilmiş hash-temelli octree veri yapısından silinir. Daha sonra, grafik sahneleme döngüsünde bu lokal alanda Marching Cubes algoritması kullanılarak sanal modelin eşyüzeyi bölgesel olarak yeniden hesaplanır [24]. İkinci olarak, simülasyon döngüsünde bir geri besleme sinyali hesaplanır ve bu sinyal Phantom cihazı yolu ile etkileşimli kuvvet formunda bir geri-besleme sinyali olarak kullanıcıya iletilir. Bu etkileşim kuvveti, çalışma parçasının ve sanal aracın fiziksel özelliklerine bağlı olarak Open Haptics Toolkit tarafından hesaplanır.

1.3.2.3. Haptic sahneleme modülü

Haptic sahneleme modülü haptic cihazına geri gönderilen geri-besleme kuvvetini hesaplar. Haptic cihazının miline bağlanan sanal araç çalışma parçasının yüzeyi ile çarpıştığında, milin konumu ve oryantasyonu elde edilir, geri-besleme kuvveti hesaplanır ve haptic cihazı yolu ile kullanıcıya iletilir.

1.4. Tezin Organizasyonu

Tezin bundan sonraki bölümleri aşağıdaki gibi düzenlenmiştir:

Bölüm 2'de genel olarak sanal heykeltıraşlık ve önerilen sistemin temel bileşenleri olan vokselleştirme, eş-yüzeyin oluşturulması ve etkileşimi sağlayan haptic sahneleme konularında yapılan literatür çalışmalarına yer verilmiştir.

Bölüm 3’te üzerinde işlem yapılacak olan sanal çalışma parçasının ön-işleme adımı ve grafik sahneleme döngüsü olmak üzere iki temel adımdan oluşan grafik sahneleme modülü ile ilgili ayrıntılı verilmiştir.

Bölüm 4’te, grafik sahneleme modülünün bir bileşeni olan ve tezde önerilen optimize-edilmiş hash-temelli octree veri yapısı detaylı olarak anlatılmıştır.

Bölüm 5’de, simülasyon modülünü başlatan çarpışmanın nasıl tespit edildiği ve meydana gelen çarpışmaya grafik ve haptic sahneleme modülleri tarafından nasıl cevap verildiği anlatılmıştır.

Bölüm 6’te haptic sahneleme modülü ve simülasyon modülünde tespit edilen çarpışma sonucu tetiklenen haptic sahnelemenin nasıl gerçeklendiği anlatılmıştır.

Bölüm 7’de ise tezde önerilen optimize-edilmiş hash-temelli octree veri yapısının diğer octree temelli veri yapılarına göre (işaretçi-temelli ve hash-temelli) avantajları anlatılmış, ön-işleme ve gerçek-zaman performansları tablolar ile gösterilmiştir.

BÖLÜM 2. LİTERATÜR ÇALIŞMASI

Bir sanal heykeltraşlık uygulaması geliştirmek için literatürde önerilen pek çok çalışma yer almaktadır. Bu uygulamalar geliştirilmeden önce uygulamanın amacına uygun olarak önceden belirlenmesi beklenen kriterlerin bir kaçı aşağıda listelenmiştir:

- Giriş modelinin türü (B-rep, CT/MRI verisi, vs.)
- Heykel verisinin hafızada saklanması
- Yontulan model yüzeyinin yeniden oluşturulması
- Etkileşim var ise geri-besleme kuvvetinin hesaplanması
- Sanal yontma aracının fonksiyonları ve şekilleri

Önerilen çalışmalarda bu soruların cevaplanmasının ardından, amaca uygun yaklaşımlara odaklanılarak bazıları yukarıda sayılan kriterlerden bir veya bir kaçına temel katkılar yapılmıştır.

Bu bölümde, hafıza optimizasyonu ve gerçek-zaman etkileşimine odaklanan ve bir haptic-temelli sanal heykeltraşlık uygulaması geliştirilmesi amacıyla önerilen sanal heykeltraşlık sistemi için literatürde yapılan çalışmalara değinilmiştir.

2.1. Sanal Heykeltraşlık

Sanal heykeltraşlık, kullanıcılara sanat, tasarım ve hızlı prototipleme alanlarında sanal gerçeklik donanım ve yazılımlarının birlikte kullanılmasıyla oluşturulan sanal bir dünya içerisinde yeni 3B nesne oluşturma veya mevcut nesneleri değiştirebilme imkânı sağlayan 3B bir modelleme işlemidir. Bir sanal heykeltıraşlık sisteminin temel amacı, kullanıcıların sanal gerçeklik donanım ve yazılımları ile donatılmış

etkileşimli bir ortam yolu ile 3B nesneleri oluşturmalarına veya şekillendirebilmelerine imkân sağlamaktır.

Literatürde, tasarım ve modelleme alanlarında eğitimi amaçlayarak önerilen sanal heykeltıraşlık çalışmaları mevcuttur [3-16]. Bilinen ilk sanal heykeltıraşlık çalışması Galyean ve Hughes tarafından 1992'de önerilmiştir [3]. Bu çalışmada, uniform ayrık voksel ızgarası olarak tanımlanan başlangıç modeli, 3B serbest-formlu şekiller oluşturmak amacıyla etkileşimli olarak ekleme ve çıkarma gibi temel işlemler gerçekleştirilerek işlenmektedir. Ayrıca, ısı tabancası, zımparalama ve renk değiştirici gibi araç tanımlamaları önerilmiştir. Daha sonra, Wang ve Kaufman tarafından yontma ve kesme temelli araçlara sahip benzer başka bir heykeltıraşlık sistemi önerilmiştir [4].

Williams ve diğerleri tarafından 3B sanal bir nesneyi gerçek-zamanlı olarak yontabilmek amacıyla ayrık vokseller kümesi üzerinde hacim çıkartma işlemi gerçekleştirilmiştir [5]. Bu çalışmada kullanılan giriş modeli CT/MRI görüntülerinden alınan vokseller kümesi veya voksel hacim modeline dönüştürülen bir üçgen ızgaradır. Burada, modelin eş-yüzeyini yeniden oluşturmak amacıyla voksellerin oluşturduğu nokta bulutlarından üçgen ızgaralar elde eden BPA (Ball Pivoting Algorithm) [30], modelin yontulan bölgesel alanından yeniden üçgen ızgara oluşturmak amacıyla DBPA (Dynamic Ball Pivoting Algorithm) olarak genişletilmiştir. Daha sonra, O'Neill ve arkadaşları tarafından, yontma işlemini gerçek-zamanda gerçekleştirebilmek ve sanal nesne yüzeyi ile etkileşim sağlayabilmek amacıyla bu çalışmaya 3B mouse gibi davranan kalem-temelli bir haptic cihazı kullanarak gerçekçi kuvvet-geri beslemesi eklenmiştir [6].

Yukarıda değinilen çalışmalarda, 3B uzayın eşit-boyutlu birim küplere bölünmesi ile oluşturulan 3B ayrık bir dizi kullanılarak gösterildiği uniform voksel gösterimi kullanılmıştır. Bu vokseller, nesnenin içinde olup olmadıklarına göre etiketlenirler. Bu gösterimde, çözünürlük arttıkça sanal nesneyi göstermek için gerekli voksel sayısından dolayı hafıza gereksinimi de artar. Örneğin, 1024x1024x1024 çözünürlükte sanal bir nesneyi göstermek için bir milyardan fazla voksel gereklidir.

Son zamanlarda, 3B modeller için gerekli hacim miktarının büyüklüğü ile başa çıkmak için araştırmacılar başta hiyerarşik veri yapıları olmak üzere farklı yaklaşımlar kullanmaya başlamıştır.

Barentzen tarafından, daha düşük hafıza maliyeti ile daha yüksek çözünürlüklerde çalışabilmek amacıyla voksel-temelli hacimsel verinin yinelemeli olarak sekiz eşit oktanta bölen octree veri yapısı kullanımı önerilmiştir [7]. Burada kullanılan sanal araç bir küre, işlemler ise hacim ekleme ve çıkarma şeklindedir. Ferley ve diğerleri tarafından ise yüksek detay seviyelerine ulaşmak amacıyla bir vokselin 27 parçaya (26 komşusu ve kendisi) bölünebileceği bir hiyerarşik bir yapı önerilmiştir [8,9].

Perng ve diğerleri tarafından, Galyean ve Hughes tarafından geliştirilmiş sanal heykeltraşlık sistemi temel alınarak gerçek-zamanlı ve etkileşimli yeni bir modelleme sistemi önerilmiştir [10]. Haptic etkileşimin 3D tracker kullanılarak sağlandığı bu uygulamada kesme aracının sınırlarındaki voksellerin durumları değiştirilerek yontma, doldurma, çömlek yapma ve boyama gibi işlemler gerçekleştirilmiştir. Aynı zamanda, heykel nesnesinin eş-yüzeyi, octree kullanılarak saklanan bu voksellere her etkileşim sonrası kenar çevirme tekniği [31] uygulanarak yeniden oluşturulmuştur.

Ho ve arkadaşları tarafından önerilen haptic-temelli sanal heykeltraşlık sisteminde sanal modele ait vokseller hafızayı ve yürütme sürelerini optimize etmek amacıyla octree kullanılarak saklanmış ve Boolean işlemler kullanılarak değiştirilmiştir [11]. Sanal model eş-yüzeyinin yeniden oluşturulması sırasında octree veri yapısının seviyelerinden dolayı oluşan üçgenler arasındaki çatlakların kübik ilerleyen kareler algoritması ile iyileştirilmesi önerilmiştir [32].

Raffin ve diğerleri tarafından önerilen sanal heykeltraşlık çalışmasında voksel-temelli hacimsel kodlama ve B-rep gösteriminin bir kombinasyonu mevcuttur [12]. Etkileşimlerin bir uzay topu, klavye ve mouse yolu ile kontrol edildiği bu çalışmada hafıza maliyetini düşürmek amacıyla octree-temelli çok-çözünürlüklü bir yaklaşım önerilmiştir ve hem heykel nesnesi hem de yontma aracı octree olarak kodlanmıştır.

Ayrıca, hacimsel kodlamadan çıkartılan B-rep ise etkileşim sonucu heykel nesnesinin hızlı bir görüntülenmesine imkân sağlar.

Heurtebise ve Thon tarafından önerilen 3B Haar dalgacık dönüşümüne dayanan çok-çözünürlüklü modelde dalgacık dönüşümünün hiyerarşik yapısı kullanılarak ve her detay seviyesinde bu dönüşümler yinelemeli olarak x, y ve z yönlerinde uygulanarak hem sanal heykel nesnesi hem de yontma aracı ayrık uniform vokseller kümesi olarak gösterilmiştir [13]. Böylece, uniform uzaysal numaralandırma gösterimleri, işlem ve görüntüleme zamanları bakımından maliyetli olduğundan dolayı, görüntüleme, etkileşim ve heykeltraşlık zamanlarını hızlandırmak amacıyla dalgacık dönüşümü tarafından sağlanan detay seviyesinin avantajından faydalanılmıştır.

3B ayrık voksel dizilerinin hafıza maliyeti ve hesaplama sürelerindeki dezavantajını gidermek amacıyla, yine Heurtebise ve Thon tarafından octree ve dalgacık dönüşümlerini birleştiren yeni çok-çözünürlüklü bir model önerilmiştir [14]. Burada, 3B nesne, veri içeren her düğümün 3B Haar dalgacık dönüşümü sayesinde örneklendiği bir octree içerisine yerleştirilmiştir.

Daha sonra, yine Heurtebise ve Thon tarafından çok büyük ve/veya detaylı nesneler ile başa çıkmak ve heykel nesnesi ile yontma aracı arasındaki etkileşimi sağlayan çarpışma tespitini geliştirmek amacıyla "yeniden bölme yapısı" olarak adlandırılan bir min/max yapısı önerilmiştir [15].

Sanal heykeltıraşlık uygulamalarında heykeltıraşlık işlemini gerçekleştirecek olan sanal aracının şekilleri ve fonksiyonları da literatürde odaklanılan temel konulardan birisidir. Chen ve arkadaşları tarafından 3B hacim uzayında gerçek-zamanda erime, yanma, damgalama, boyama, bloklama ve soyma işlemlerini destekleyen haptic-temelli bir sanal heykeltraşlık sistemi önerilmiştir [16].

Sanal tıp alanındaki cerrahi ve diş simülasyonlarında eğitim amacı ile geliştirilmiş sanal heykeltıraşlık uygulamaları da bilim literatüründe kendisine yer edinmiştir [17-22]. Agus ve diğerleri tarafından, temporal kemik cerrahisi için eğitim

simülatörünün bir bileşeni olarak geliştirilmiş olan kemik kesmenin gerçek-zamanlı haptic ve görsel olarak uygulanması önerilmiştir [17]. Voxel-temelli hasta CT/MRI verisi ve her iki el için de haptic cihazı kullanılan bu çalışmada, kemik delme işlemi için voksel yoğunluk değerlerine bağlı bir erozyon fonksiyonu önerilmiştir. Daha sonra, voksel verisini saklamak için hafızayı optimize etmek amacıyla octree veri yapısının kullanımı önerilmiştir [18].

Duriez ve Syllebranque tarafından diş implanataloji simülasyonu için sürtünmenin sahnelenmesi ve gerçek-zamanlı güncellenmesini içeren üç-boyutlu bir 6-dof haptic frezeleme algoritması önerilmiştir [19]. Gerçek hasta çene kemiği CT (Bilgisayarlı Tomografi) verisi kullanılan bu çalışmada kemiklere ait voxel-temelli verilere delik açmak amacıyla önerilen voxel-temelli çarpışma tespiti algoritması ve 6-dof haptic geri-besleme metodu uygulanmıştır.

Eriksson ve arkadaşları tarafından cerrahi frezeleme işlemini simüle edip kanser tümörlerinin çıkartılması gibi karmaşık kafatası kemiği işlemlerini gerçekleştirmek için cerrahları eğitmek amacıyla bir haptic ve sanal gerçeklik simülatörü geliştirilmiştir [20,21]. Burada kullanılan yüksek çözünürlüklü hacimsel veri, frezeleme süresince geçen hızlı dinamik değişimleri optimize etmek amacıyla octree veri yapısı kullanılarak saklanmıştır. Aynı zamanda, önerilen proxy-temelli haptic sahneleme algoritması ile sanal ucun kemik yüzeyinde kalması sağlanarak haptic istikrar temin edilmiş olur.

Daha sonra, Eriksson ve Wikander tarafından 2012 yılında cerrahi frezeleme işleminde daha gerçekçi bir haptic tecrübesi sağlamak amacıyla, önerilen önceki çalışmalarına hem baskı hem de dönme kuvveti geri-beslemesi sağlayan 6-dof eklenerek genişletilmiştir [22]. 6-dof haptic algoritması, Barbic ve James tarafından önerilen çalışma [23] referans alınarak, madde çıkarımından kaynaklanan hacim verisi değişimlerinin gerçek-zamanlı sahnelenmesi ile başa çıkabilmek amacıyla frezeleme işlemleri için modifiye ve optimize edilmiştir.

Çit ve diğerleri tarafından ise gerçek-zaman etkileşimini hızlandırmak amacıyla geleneksel işaretçi-temelli octree veri yapısının aksine her düğümün önceden belirlenen bir indekse göre bir hash tablosuna kaydedildiği haptic-temelli bir sanal heykeltraşlık uygulaması önerilmiştir [33]. Bu çalışmanın devamı olarak geliştirilmiş olan bu tez çalışmasında ise, voksel-temelli hacimsel veriyi saklamak için octree veri yapısının hash tabloları kullanılarak saklanmasını öneren çalışmadaki ilgili hash tablolarının optimize edilerek hafıza ve hesaplama maliyetlerinin kısaltılması önerilmiştir.

İlerleyen bölümde, bu çalışmada önerilen sanal heykeltıraşlık sisteminin üç ana modülünden birisi olan grafik sahneleme modülü ve bileşenleri detaylı olarak anlatılmıştır.

BÖLÜM 3. GRAFİK SAHNELEME MODÜLÜ

Grafik sahneleme modülü, ön-işleme ve grafik sahneleme döngüsü olmak üzere iki temel adımdan oluşmaktadır. Ön-işleme adımında sırasıyla giriş üçgen kafes modelinin vokselleştirilmesi, oluşturulan 3B voksel ızgarasındaki her voksele karşılık gelen eş-hücrelerinin elde edilmesi ve son olarak da optimize edilmiş hash-temelli octree veri yapısının oluşturulması işlemleri sırasıyla gerçekleştirilir. Grafik sahneleme döngüsünde ise, optimize edilmiş hash-temelli hacimsel veriye bağlı olarak sanal çalışma parçasına ait grafiğin 30 Hz'lik bir frekansta sürekli olarak yeniden sahnelenmesini sağlanır.

3.1. Ön-İşleme

Üçgen ızgaradan oluşan giriş modelinin yontma işlemi sırasında sanal araç ile çarpışmasının sırasında geçen süreyi optimize etmek ve modelin sadece yüzey bilgisi değil, aynı zamanda iç hacim bilgisini de elde etmek amacıyla voksel-temelli veriye dönüştürülmesi için "ön-işleme" olarak nitelenen işlem boyunca sırasıyla aşağıdaki üç temel adım gerçekleştirilir:

- Vokselleştirme: Üçgen kafes modelinin 3B hacimsel voksel verisine dönüştürülmesi
- Voksel-hücre eşliği: 3B hacimsel voksel verisindeki her bir voksele karşılık gelen eş-hücrenin oluşturulması
- Optimize edilmiş hash-temelli octree oluşturulması: Eş-hücrelerden oluşan 3B voksel uzayının hiyerarşik olarak alt bölümlere ayrılması ile optimize edilmiş hash-temelli bir octree veri yapısının oluşturulması

İlerleyen bölümlerde sırasıyla bu üç temel adım detaylı olarak anlatılmıştır.

3.1.1. Vokselleştirme

Önerilen haptic-temelli sanal heykeltıraşlık sistemine ait grafik sahneleme döngüsünün başlangıç adımı olan ön-işlemedeki temel amaç, üçgen kafes model verisinin voksel-temelli hacimsel veriye dönüştürülmesidir. Voksel verisi, üçgen yüzey modellerinin aksine eksen-hizalıdır, bakış açısından bağımsızdır ve hesaplanması halinde modelin sadece yüzey bilgisi değil iç hacim bilgisi de elde edilebilir. Buna ek olarak, voksel verisi kullanılarak ekleme ve çıkarma gibi çarpışma tespitinde zaman-kritik bir problem olan Boolean işlemlerin hesaplanması daha kolay ve hızlıdır. Bu nedenle, önerilen çalışmada, modelin yüzey bilgisini barındıran giriş üçgen modeli ilgili voksel hacim modelini elde etmek üzere vokselleştirilir.

Bilgisayar grafikleri teknolojisinin gelişmesi ile birlikte ortaya çıkan 3B bilgisayar grafiklerinde ve geometrik modellemede 3B geometrik nesneleri tanımlamak için genellikle vektör grafiklerinin 3B veri gösterimine adaptasyonundan doğmuş poligon ızgaralar (B-rep, vs.), parametrik yüzeyler (Bezier eğrileri, NURBS, vs.) ve yapısal katı geometri (CSG – Constructive Solid Geometry) kullanılır. Ancak, iki boyutlu bilgisayar grafiklerindeki vektör grafiklerine alternatif olarak kullanılan piksel modeline karşılık günümüz bilgisayar yazılım ve donanım teknolojileri ile birlikte gelişen üç boyutlu bilgisayar grafiklerinde geometrik nesneleri modellemek için iki boyutlu piksel modelinin üç boyutlu karşılığı olan voksel hacim modeli geliştirilmiştir [34].

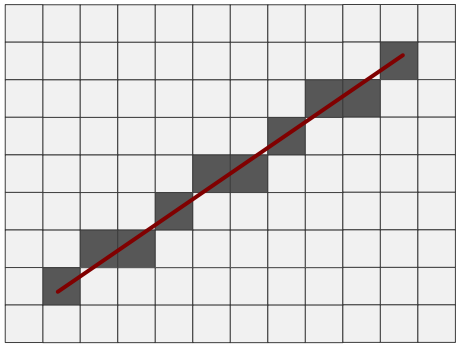
Voksel en basit haliyle Öklid uzayında (*x*,*y*,*z*) merkezli üç boyutlu bir küptür. Geleneksel yüzey gösterimi sunan yaklaşımların aksine, bir voksel karşılık geldiği birim hacmin içinde olan renk, sıcaklık, yoğunluk, opaklık, sertlik vb. gibi modelin gerçek nesne fenomenleri olan ölçülebilir fiziksel özelliklerini de barındırabilir.

Voksel-temelli modeller, son yıllarda üç boyutlu hacim verisinin öneminin artması ile birlikte hacim modelleme/sahneleme [35-37], tıbbi görüntüleme [38], sanal tıp [39], akış dinamiği [40], sonlu eleman analizi [41], çarpışma tespiti [42,43], CSG modelleme [44] gibi pek çok alanda kullanılmaya başlanmıştır.

Geleneksel üç boyutlu geometrik yüzey gösterimlere alternatif olarak ortaya çıkan ve hacim grafiklerinin temelini oluşturan yeni 3B veri gösterimini elde etmek için bir dönüşüm işlemi gerekir. “Vokselleştirme” olarak adlandırılan bu dönüşüm işlemi nesnenin 3B ayrık bir ızgaradaki vokseller kümesi olarak gösterilmesi işlemi olarak tanımlanır. Diğer bir deyişle, vokselleştirme, nesnenin 3B ayrık bir ızgarada yer alan tüm vokseller için bir üyelik sınıflandırma problemidir.

Kaufman ve arkadaşları “Hacim Grafikleri” isimli makalede [45] vokselleştirme için şu cümleleri söylemiştir: “...Vokselleştirme olarak adlandırılan bu adım geometrik nesnelerin sürekli gösterimlerinden sürekli nesneye en yakın vokseller kümesine dönüştürülmesi olarak ele alınır. 2B nesneleri pikselleştiren tarama-dönüşümü (scan-conversion) işlemini taklit ettiğinden dolayı, 3B tarama dönüşümü olarak da anılır…”.

Kaufman ve Shimony tarafından geliştirilen literatürdeki ilk vokselleştirme çalışmasında, algoritma doğrular, daireler, poligonlar, çok yüzlü ve quadratik nesneler gibi çeşitli yüzeyler ve nesneleri vokselleştirir [34]. Bu çalışmada, iki boyutlu vektör grafiklerinden raster grafiğe geçiş için yapılan 2B tarama-dönüşümü işleminin [46,47] üç boyutlu uzantısı gerçeklenmiştir. Şekil 3.1’de 2B tarama-dönüşümü algoritmalarından birisi olan ve Kaufman ve Shimony’nin önerdikleri vokselleştirme çalışmasına ışık tutan Bresenham’ın doğru çizme algoritması gösterilmektedir [48].



Şekil 3.1. 2B tarama-dönüşümü örneği

Başka bir şekilde ifade edecek olursak, “vokselleştirme” yukarıda da bahsedildiği gibi sürekli geometrik nesneden kendisine en yakın artık olarak sayısallaştırılmış veritabanını veya bir diğer değişle voksel veri kümesini üreten bir dönüşüm işlemidir. Elde edilen voksel-temelli veri, sahneleme amacı ile kullanılabileceği gibi [35-37], bu çalışmada olduğu gibi çarpışma tespiti gibi karmaşık işlemleri hızlandırmak ve modelin iç hacim bilgisini elde etmek amacıyla bir ara işlem adımı olarak da kullanılabilir.

Üç boyutlu geometrik modelleri voksel gösterim modeline dönüştürmek için geliştirilen bu algoritmaların temel prensibi hangi voksellerin geometrik model ile kesiştiğinin belirlenmesidir. Diğer bir deyişle, vokselleştirme algoritmalarındaki en basit ve temel fikir her bir vokselin nesneye ait olup olmadığının belirlenmesidir. Buna bağlı olarak da eğer nesne tarafından kapsanıyor ise ilgili voksele "1" değilse de "0" değeri atanır. Sadece nesnenin var olup olmadığını ifade eden ikili (boolean) bir değer kullanan bu tarz vokselleştirme yaklaşımları “ikili vokselleştirme” olarak adlandırılır.

Ancak bu yaklaşım sahnelemede mutlak evet/hayır durumundan kaynaklanan önemli aliasing hatalarını da beraberlerinde getirir. Bu nedenle de her vokselin yoğunluk olarak adlandırılan [0;1] aralığında bir reel sayı ile gösterildiği “çok-değerli vokselleştirme” yaklaşımları geliştirilmiştir [49,50]. Alias-free/smooth gösterim üzerine odaklanan bu metotta iki yaklaşım mevcuttur. Bu tekniklerdeki temel fikir nesnenin içi ve dış yüzeyi arasındaki voksellere aşamalı olarak değişen bir yoğunluk değeri atanması ve buna bağlı olarak da yumuşak yoğunluk geçişi yolu ile aliasing hatalarının önlenmesidir.

Bilgisayar grafiklerinde üç boyutlu nesneler iki kategoriye ayrılırlar: Nesneye ait sadece yüzey bilgisinin tutulduğu “yüzey gösterimleri” veya bir diğer ifade ile “sınır gösterimleri” (kısaca B-rep) ve dış yüzey bilgisinin yanında nesnenin iç kısmının bilgisinin de tutulduğu “hacim gösterimleri” ve bir diğer ifade ile “katı modeller”. Bu bağlamda vokselleştirme, üç boyutlu nesneye ait örneklenen verilerin veya geometrik modellerin hacim uzayını dolduran “hacim temelli yaklaşımlar” [50-54] ve sadece

yüzeyi dolduran “yüzey temelli yaklaşımlar” [55-57] olmak üzere iki kategoriye ayrılır. Hacim temelli yaklaşımların, yüzey temelli olanlara göre tek farkı, 3B katı nesnenin iç yüzeyi de dâhil olmak üzere nesneye ait tüm vokseller kümesi ile ifade edilmesidir.

Hacim temelli yaklaşımlar nesnenin iç hacminin önemli olduğu uygulamalarda kullanılırken yüzey temelli yaklaşımlar da hafıza alanını korur. Çünkü yüzey verisinin miktarı nesnenin yüzeyi ile orantılı iken hacim verisinin miktarı da nesnenin uzayda kapladığı hacim ile orantılıdır. Sonuç olarak çoğunlukla yüzey verisi hacim verisinden daha az yer kaplar, daha kolay değiştirilebilir ve görüntülenebilir ancak 3B nesnenin modellenmesinde hangi yaklaşımın kullanılacağı amaca uygun olarak seçilir. Bununla birlikte, vokselleştirmenin seksenli yılların ortasında yapılan ilk uygulamasından [34] günümüze kadar uzanan tarihine bakıldığında hacim grafiklerinin artan popülerliği ile birlikte yüzey temelli yaklaşımlardan hacim temelli yaklaşımlara geçiş belirgin bir şekilde gözlenir.

Bu tez çalışmasında önerilen haptic-temelli sanal heykeltıraşlık uygulamasında yontulacak 3B modelin iç hacim bilgisini elde etmek ve sanal araç ile çarpışmasını optimize etmek amacıyla üçgen yüzey modeli voksel-temelli hacim verisi oluşturacak şekilde vokselleştirilmiştir. Ayrıca, vokselleştirme işlemi sonucu elde edilecek olan hacimsel voksel veri kümesi sahnelemede kullanılmayacağından ve çok-değerli vokselleştirmenin yüksek hesaplama maliyetinden kaçınmak için üçgen yüzey modeli ikili vokselleştirme yaklaşımı kullanılarak gerçekleştirilmiştir. 3B üçgen kafes modelinden hacimsel ikili voksel veri kümesi elde etmek için gerçekleştirilen vokselleştirme algoritmasının sözde kodu, Şekil 3.2’de verilmiştir.

vokselleştirme (Tri[n])

/ 3B üçgen kafes modelinden hacimsel ikili voksel veri kümesi oluştur */*
GİRİŞ: Tri[n], 3B kafes modelindeki üçgenler kümesi, **res**, 3B çıkış ayrık ızgara çözünürlüğü ve **index**, üç-boyutlu konumu (x,y,z) olan vokselin tek-boyutlu konum indexi
ÇIKIŞ: x,y,z ϵ {0,1,…,**res**} olacak şekilde **VoxelData[res^3]** 3B modelin resxresxres boyutundaki ayrık vokseller kümesi

```
/* 3B üçgen kafes modelinin eksen-hizalı kaplayan hacmini hesapla */
modelAABB_hesapla(Tri[n]);
/* Kaplayan hacim içinde bir res çözünürlüğüne göre resxresxres boyutunda 3B bir
voksel ızgarası oluştur ve bu hacimdeki her vokselin değerini 0 olarak ata. */
for (x=0; x<res; x++)
    for (y=0; y<res; y++)
        for (z=0; z<res; z++)
            index = x*(res)^2 + y*res + z;
            VoxelData[index] ← 0 ;

/* voksel yüksekliğini hesapla */
h ← ((AABB.max – AABB.min ) / res) * 0.5;
/* Yüzey Vokselleştirme */
for (modeldeki her Tri[i] için)
    for (ilgili üçgenin kaplayan hacimdeki tüm VoxelData[index] için)
        if (VoxelData[index] = 0)
            if (kesisim_testi(Tri[i] , VoxelData[index],h) != 0)
                VoxelData[index] ← 1 ;

/* Hacim Vokselleştirme */
tohum_doldur(VoxelData[res^3]);
/* Voksel-Hücre Eşleştirme */
voksel_eslestir(VoxelData[res^3]);
```

Şekil 3.2. 3B üçgen kafes modelinden hacimsel ikili voksel veri kümesi elde etmek için gerçekleştirilen algoritmanın sözde kodu

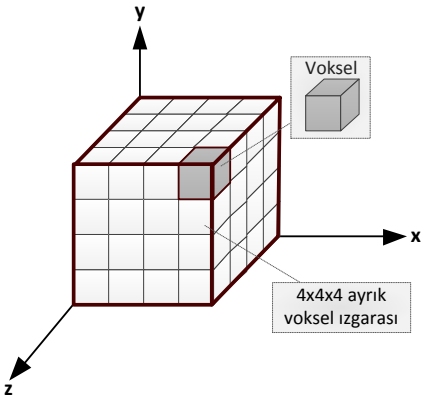
Burada, *n* tane üçgenden oluşan üçgen kafes modelinin eksen-hizalı kaplayan hacminin nasıl hesaplandığını açıklayan "modelAABB_hesapla" isimli fonksiyona ait sözde kod Şekil 3.7'de gösterilmektedir. Başlangıçta belirtilen bir çözünürlüğüne göre oluşturulan 3B boş voksel ızgarası üzerine yerleştirilen üçgen kafes modelinin yüzeyinin, boş ızgaradaki hangi vokseller ile çakıştığının nasıl hesaplandığını açıklayan "kesisim_testi" isimli fonksiyona ait sözde kod Şekil 3.12'de

gösterilmektedir. Daha sonra, oluşturulan 3B yüzey voksel verisinin iç haminin doldurulmasının nasıl gerçekleştirildiğini açıklayan “tohum_doldur” isimli fonksiyona ait sözde kod Şekil 3.15’de gösterilmektedir. Son olarak, 3B ikili hacimsel voksel verisindeki her voksele karşılık gelen eş-hücrelerinin oluşturulması işleminin nasıl gerçekleştirildiğini açıklayan “voksel_eslestir” isimli sözde kod ise 3.18’de gösterilmektedir.

İlerleyen bölümlerde, sırasıyla Şekil 3.2’deki sözde kodda bahsi geçen voksel kavramı, kaplayan hacim hesaplama, yüzey vokselleştirme için kullanılan Möller’in üçgen/kutu kesişim testi algoritması, yüzey verisinin iç hacmini doldurmak için kullanılan tohum doldurma algoritması ve elde edilen 3B voksel ızgarasının her bir vokseline karşılık gelen eş-hücre hesaplama işlemi detaylı olarak açıklanmıştır.

3.1.1.1. Voksel

Piksel iki boyutlu uzayda belirli bir koordinatı gösterirken, voksel de benzer şekilde üç boyutlu hacim uzayında belirli bir koordinatı gösteren veri elemanıdır. En basit haliyle voksel, Öklid uzayında (*x*,*y*,*z*) merkezli basit bir küptür denilebilir. Şekil 3.3’de 3B ayrık voksel uzayı ve bu uzayda yer alan tek bir voksel gösterilmektedir. Buradaki voksel uzayının çözünürlüğü 4x4x4‘tür. Yani, uzay her üç eksende (*x*,*y*,*z*) dört eşit parçaya bölünmüştür. Dolayısıyla, 3B ayrık uzayda eşit-boyutlu 64 adet voksel yer almaktadır. Bu voksellerin oluşturduğu vokseller kümesi “voksel verisi” olarak da adlandırılır.

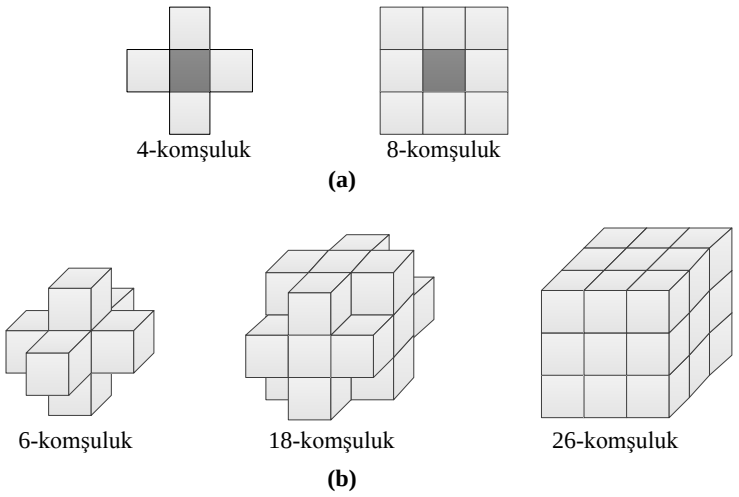


Şekil 3.3. 4x4x4 3B ayrık voksel uzayı ve voksel

3B ayrık uzaydaki bir vokselin değeri {0,1} arasındadır: “1” olarak atanan vokseller 3B nesneleri gösteren “siyah” veya “dolu” vokseller olarak adlandırılırken ve “0” olarak atanan vokseller ise şeffaf arka planı gösteren “beyaz” veya “boş” vokseller olarak adlandırılır. Bunun yanı sıra voksel değerlerinin [0,1] aralığında olduğu ikili olmayan yaklaşımlarda ise [0,1] aralığındaki gri-seviyeli değerler vokselin yoğunluğu olarak ifade edilir.

Geleneksel yüzey gösterimi sunan yaklaşımların aksine, bir voksel karşılık geldiği birim hacmin içinde olan renk, sıcaklık, yoğunluk, opaklık, sertlik vb. gibi modelin gerçek nesne fenomenleri olan ölçülebilir fiziksel özelliklerini de barındırabilir.

3B ayrık voksel uzayında her bir vokselin diğer vokseller ile yüz, kenar ve köşelerinin komşu olup olmadığına bağlı olarak 3 tip komşuluk ilişkisi vardır. Şekil 3.4’de, sırasıyla 2B piksel komşuluğu ve 3B voksel komşulukları gösterilmektedir. Şekil 3.4a’da ortadaki gri piksele sırasıyla etrafındaki dört ve sekiz voksel komşudur. Dolayısıyla, ortadaki pikselin kenarlarını paylaşan ilk dört voksel ilgili vokselin 4-komşusu, hem kenar hem de köşelerin paylaşan sekiz piksele de 8-komşusu denir. 3B voksel uzayında (Şekil 3.4b) ise bir voksel, etrafındaki vokseller ile 6 adet yüzünü paylaşıyor ise bu voksellere 6-komşu, 6 yüz ve 12 kenarını paylaşıyor ise 18-komşu, hem 6 yüz, hem 12 kenar, hem de 8 köşesini paylaşıyor, yani tüm olası voksel komşuluklarını içeriyor ise bunlara da 26-komşu denir [58].

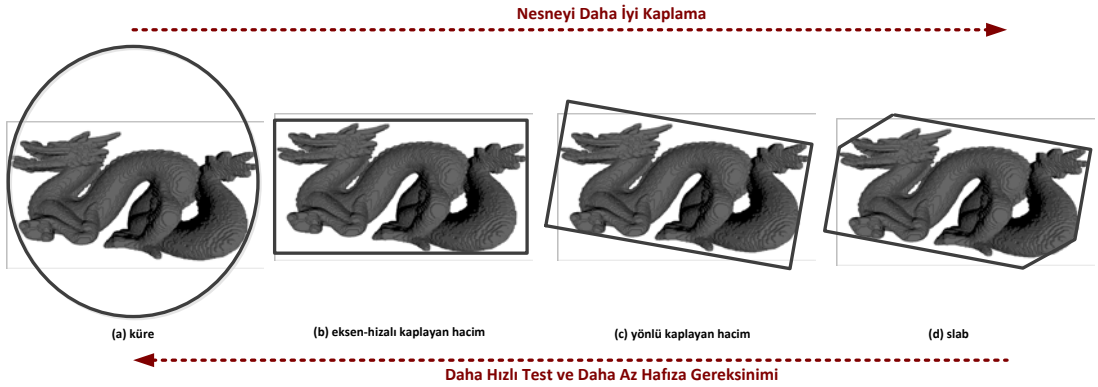


Şekil 3.4. 2B piksel ve 3B voksel komşuluğu (a) Ortadaki koyu renkli piksele N-komşu 2B pikseller kümesi (N Є {4,8}) (b) Ortadaki voksele N-komşu 2B vokseller kümesi (N Є {6, 18, 26}) [58].

3.1.1.2. Kaplayan hacim hesaplama

3B uzaydaki bir nesnenin tüm yüzeyini kaplayan en küçük hacmine o nesnenin "kaplayan hacmi" denir. Bir nesneyi kaplayan hacim ile çevreleme fikri ilk olarak Clark tarafından gizli-yüzey algoritmasının performansını geliştirmek için önerilmiştir [59] ve daha sonra başta çarpışma tespiti olmak üzere bilgisayar grafiklerindeki farklı çalışmalarda nesnenin çalışma alanını optimize etmek amacıyla kullanılmıştır [60,61].

Bir nesnenin kaplayan hacmi pek çok değişik şekilde hesaplanabilir. Şekil 3.5'de sırasıyla küre, eksen-hizalı kaplayan hacim (AABB), yönlü kaplayan hacim (OBB) ve slab olmak üzere dört temel kaplayan hacim şekli gösterilmektedir.



Şekil 3.5. Temel kaplayan hacim şekilleri (a) küre, (b) eksen-hizalı kaplayan hacim, (c) yönlü kaplayan hacim ve (d) slab

Kaplayan hacim hesaplamanın en basit olanı kaplayan hacmi bir küre kullanarak hesaplamaktır. Burada, nesneyi kaplayan en küçük yarıçaplı küre kaplayan hacim olarak belirlenir.

Kaplayan hacim hesaplamanın ikinci temel yöntemi de eksen-hizalı kaplayan hacim kullanmaktır. Burada, koordinat düzlemindeki her üç eksene paralel en küçük alanlar belirlenip birleştirilerek oluşturulan eksen-hizalı prizma kaplayan hacim olur.

Üçüncü temel yöntem ise yönlü kaplayan hacim hesaplamadır. Buradaki yöntem, eksen-hizalı yönteme benzerdir. Fakat burada, ilk önce nesnenin yönü hesaplanarak nesneyi kaplayan prizmanın yönü buna bağlı olarak döndürülür.

Kaplayan hacim hesaplamanın dördüncü temel yolu ise nesneyi minimum ve maksimum noktalarından kesen ikili slableri belirleyerek en az altı slable nesneyi bu paralel düzlemler arasında sıkıştırmaktır.

Karmaşık kaplayan hacimler nesneyi daha sıkı kapsar, fakat hesaplama ve hafıza maliyetleri yüksektir. Şekil 3.5'deki kaplayan hacimlere bakıldığında en soldaki en hızlı, en basit ve en az hafıza gerektiren iken, aynı zamanda da nesneyi en kötü kapsayandır. En sağdaki ise nesneyi en sıkı kapsayandır, fakat diğer yandan da hesaplanması en yavaş olanıdır ve en fazla hafızayı kullanır. Buradan yola çıkarak kaplayan hacim oluşturmanın hesaplama ve hafıza maliyeti ile sıkılığı arasında ters orantı vardır denilebilir. Yani, küre ve eksen-hizalı kaplayan hacim gibi kolay ve hızlıca oluşturulan kaplayan hacimler aslında nesneyi en iyi kaplayanlar olmamalarına rağmen zaman-kritik uygulamalarda tercih edilirler. Bu çalışmada da, voksel yapısına uygun olması ve kolay hesaplanması nedeniyle eksen-hizalı kaplayan hacim kullanılmıştır.

Şekil 3.6 ve Şekil 3.7'de sırasıyla tek bir üçgen ve bir üçgen kafes modelinin kaplayan hacimlerinin nasıl hesaplandığını gösteren sözde kodlar verilmiştir.

ucgenAABB_hesapla (Tri)

/ Verilen üçgenin eksen-hizalı kaplayan hacmini hesapla */*

GİRİŞ: V_0, V_1, V_2, Üçgenin köşe konum vektörleri

ÇIKIŞ: min, max, üçgenin eksen-hizalı kaplayan hacminin en küçük ve en büyük konum vektörleri

```
/* eksen-hizalı kaplayan hacmin en küçük ve en büyük köşe noktalarını
başlangıçta V0 olarak belirle */
min ← V0;
max ← V0;

/* eksen-hizalı kaplayan hacmin en küçük ve en büyük köşe noktalarının
değerlerini V1 ve V2 ile karşılaştır */
if (min.x > V1.x)
    min.x ←V1.x;
else if (min.x > V2.x)
    min.x ←V2.x;

if (min.y > V1.y)
    min.y ←V1.y;
else if (min.y > V2.y)
    min.y ←V2.y;

if (min.z > V1.z)
    min.z ←V1.z;
else if (min.z > V2.z)
    min.z ←V2.z;

if (max.x > V1.x)
    max.x ←V1.x;
else if (max.x > V2.x)
    max.x ←V2.x;

if (max.y > V1.y)
    max.y ←V1.y;
else if (max.y > V2.y)
    max.y ←V2.y;

if (max.z > V1.z)
    max.z ←V1.z;
else if (max.z > V2.z)
    max.z ←V2.z;
```

Şekil 3.6. Üçgen için eksen-hizalı kaplayan hacim hesaplamayı gerçekleştiren algoritmanın sözde kodu

modelAABB_hesapla (Tri[n])

/ Verilen üçgenin kafes modelinin eksen-hizalı kaplayan hacmini hesapla */*

GİRİŞ: n, üçgen sayısı ve **min**, **max** ise ilgili üçgenin en küçük ve en büyük konum vektörleri

ÇIKIŞ: AABBmin, **AABBmax**, modelin eksen-hizalı kaplayan hacminin en küçük ve en büyük konum vektörleri

```
/* eksen-hizalı kaplayan hacmin en küçük noktasının başlangıç değerini  çok büyük,
en büyük noktasının başlangıç   değerini ise çok küçük olarak ata */
AABBmin ← 1.0e+30;
AABBmax ← -1.0e+30;

for (int i = 0; i < n; i++)
    if (Tri[i].min < min.x) AABBmin .x  ← Tri[i]min.x;
    if (Tri[i].min < min.x) AABBmin .y  ← Tri[i]min.y;
    if (Tri[i].min < min.x) AABBmin .z  ← Tri[i]min.z;
    if (Tri[i].min < min.x) AABBmax .x ← Tri[i]min.x;
    if (Tri[i].min < min.x) AABBmax .y ← Tri[i]min.y;
    if (Tri[i].min < min.x) AABBmax .z ← Tri[i]min.z;
```

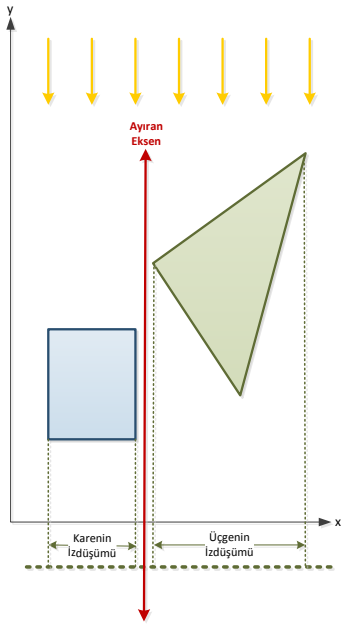
Şekil 3.7. Üçgen kafes modeli için AABB hesaplamayı gerçekleştiren algoritmanın sözde kodu

3.1.1.3. Möller'in 3B üçgen/kutu kesişim testi

Möller'in 3B üçgen/kutu kesişim testi üç-boyutlu bir üçgenin bir prizma ile kesişip kesişmediğini test eder. Eksen ayırma teoremine (SAT – Seperating Axis Theorem) dayanan bu algoritmanın bir üçgen ile prizma arasında ayıran bir eksen olup olmadığını test etmek için toplam 13 kriteri mevcuttur. Eğer, tüm testlerin geçilmesi sonucunda hiçbir ayıran eksen bulunmuyor ise üçgen, kutu ile kesişiyor demektir.

Tek bir üçgen ve tek bir kutu arasındaki çarpışmayı hızlı bir şekilde test etmek için geliştirilmiş olan algoritma genişletilerek, 3B üçgen modellerin yüzeyini vokselleştirmek amacıyla kullanılmıştır [5,62]. Burada, yüzey vokselleştirme işlemi, 3B ayrık voksel ızgarası içine yerleştirilen üçgen kafes modelinin yüzeyinin hangi vokseller ile hesaplanarak gerçekleştirilir. Bir üçgenin bir voksel ile kesişimi, dolayısıyla ilgili vokselin 3B modelin yüzeyi ile kesişimi bir kere doğrulanmış ise bu vokselin kesişim testi modeldeki diğer üçgenler için tekrarlanmaz. Böylece, gereksiz işlem yapılmaz. Bu algoritma ufak kayan-nokta hataları dışında tam sonuç verir.

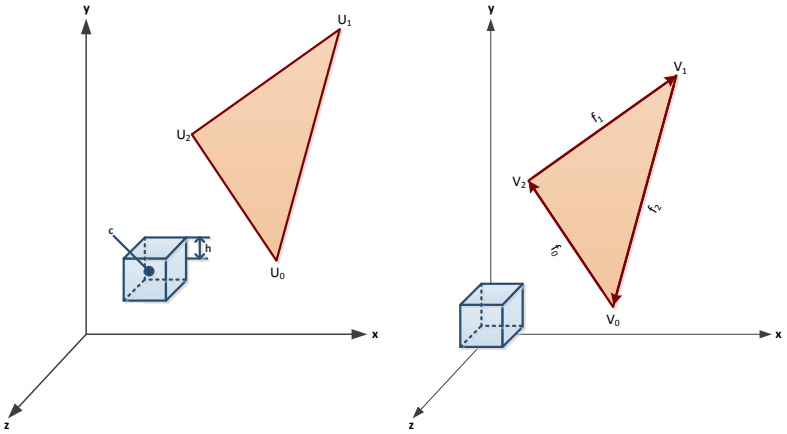
Eksen ayırma teoremi iki dışbükey çok yüzlü poligonun kesişip kesişmediğini, bu poligonların izdüşümlerinin kesişme durumuna göre belirler. Yani, iki dışbükey nesnenin izdüşümleri birbirine çakışmayacak bir eksen çizebiliyorsa bu iki nesne kesişmiyor denir. İki nesnenin kesişmediğine karar vermek için tek bir ayıran eksen bulunması yeterlidir, kesişmesi için de hiçbir ayıran eksen olmaması gerekir. Şekil 3.8'de bir üçgen ve karenin x eksenine göre izdüşümleri ve izdüşüm düzlemine dik örnek bir ayıran eksen gösterilmektedir.



Şekil 3.8. Bir üçgen ve karenin x eksenine göre izdüşümleri arasındaki ayıran eksen

Möller'in 3B bir üçgen ve kutunun kesişimini eksen ayırma teoremine göre test eden algoritmasının toplam on üç kriteri mevcuttur. Bunlardan üçü, üçgenin kaplayan hacmine karşı kutunun kaplayan hacminin kesişimini test ederken, biri, üçgen normal vektörüne dik düzlem ile kutunun kesişimini ve diğer dokuzu ise üçgen ile kutunun izdüşümlerinin kesişimini test eder. Bu çalışmada, üçgenin kaplayan hacmindeki vokseller alanının sınırları Şekil 3.2'deki sözde kodda da görüleceği gibi kesişim testinden önce hesaplandığından dolayı, ilgili voksel zaten üçgenin kaplayan hacmi ile kesişecektir. Dolayısıyla kaplayan hacimlerin kesişip kesişmediğini tekrar test etmeye gerek yoktur. O halde ilk üç test atlanır ve geriye 10 test kalır.

Merkez noktası c, yüksekliği h olan voksel ile köşe noktaları u_0, u_1, u_2 olan üçgenin kesişim testi için ilk önce ilgili voksel ile üçgen, ilgili vokseli, merkezi koordinat düzleminin merkezine yerleşecek şekilde taşınır. Bu durumda üçgenin yeni koordinatları $v_i = u_i - c, i \in \{0,1,2\}$ şeklinde olacaktır. Şekil 3.9'da örnek 3B voksel ve üçgenin başlangıç ve sonraki taşınmış konumları gösterilmektedir.



Şekil 3.9. Örnek 3B voksel ve üçgenin başlangıç ve taşınmış konumları

Üçgenin kenarlarından geçen f_0, f_1 ve f_2 vektörleri Denklem 3.1, Denklem 3.2 ve Denklem 3.3'deki gibi hesaplanır:

$$f_0 = v_1 - v_0 \tag{3.1}$$

$$f_1 = v_2 - v_1 \tag{3.2}$$

$$f_2 = v_0 - v_2 \tag{3.3}$$

Üçgenin normal düzlemi ile ilgili vokselin kesişip kesişmediğini test etmek için Denklem 3.4, Denklem 3.5, Denklem 3.6 ve Denklem 3.7'deki işlem adımları takip edilir:

$$n = f_0 x f_1 \tag{3.4}$$

$$d = n x v_0 \tag{3.5}$$

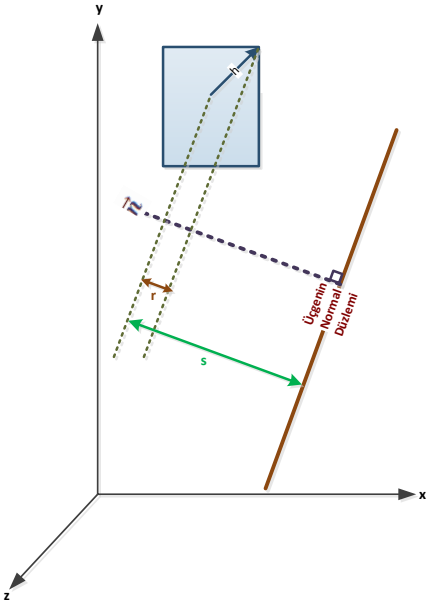
$$r = h \times \left(|n_x| + |n_y| + |n_z|\right) \tag{3.6}$$

$$s = n.c - n \times v_0 \tag{3.7}$$

Burada, *n*, üçgenin normal düzlemi, *s* voksel merkezinden düzleme dik uzaklığı, *r* ise normal düzlemine göre *h*'nin izdüşümüdür. Matematiksel olarak ifade edecek olursak;

EĞER *(s-r = 0)* İSE üçgenin normal düzlemi ile voksel kesişiyor
AKSİ HALDE üçgenin normal düzlemi ile voksel kesişmiyor

denir. Şekil 3.10'da üçgenin normal düzlemi ile ilgili vokselin kesişim testi 2B olarak gösterilmektedir.



Şekil 3.10. Üçgenin normal düzlemi ile voksel kesişim testinin 2B gösterimi

Daha sonra ise, üçgenin kenarları ile ilgili vokselin yüzeylerinin izdüşümlerinin kesişip kesişmediğini test eden toplam dokuz test uygulanır. Testlerden bir tanesi Denklem 3.8, Denklem 3.9 ve Denklem 3.10'daki gibidir:

$$e_0 = (1,0,0) \quad (3.8)$$

$$e_1 = (0,1,0) \quad (3.9)$$

$$e_2 = (0,0,1) \quad (3.10)$$

Burada e_0, e_1 ve e_2 küpün sırasıyla x, y ve z eksenlerindeki normalleridir ve $i, j \in \{0,1,2\}$ olmak üzere $a_{ij} = e_i \times f_j$dir ve a_{00} için kesişim testi Denklem 3.11 ve Denklem 3.12'deki gibi yapılır:

$$a_{00} = e_0 \times f_0 \tag{3.11}$$

$$a_{00} = (0, -f_{0z}, f_{oy}) \tag{3.12}$$

Üçgen köşelerinin a_{00} a göre izdüşümünün uzunluğu Denklem 3.13, Denklem 3.14 ve Denklem 3.15'deki gibi hesaplanır:

$$P_0 = a_{00}.V_0 = V_{0z}.V_{1y} - V_{0y}.V_{1z} \tag{3.13}$$

$$P_1 = a_{00}.V_1 = V_{0z}.V_{1y} - V_{0y}.V_{1z} = \mathrm{P}_0 \tag{3.14}$$

$$P_2 = a_{00}.V_2 = (V_{1y} - V_{0y})V_{2z} - (V_{1z} - V_{0z})V_{2y} \tag{3.15}$$

Voksel yüksekliğinin a_{00}'a göre izdüşümünün uzunluğu Denklem 3.16'daki gibi hesaplanır:

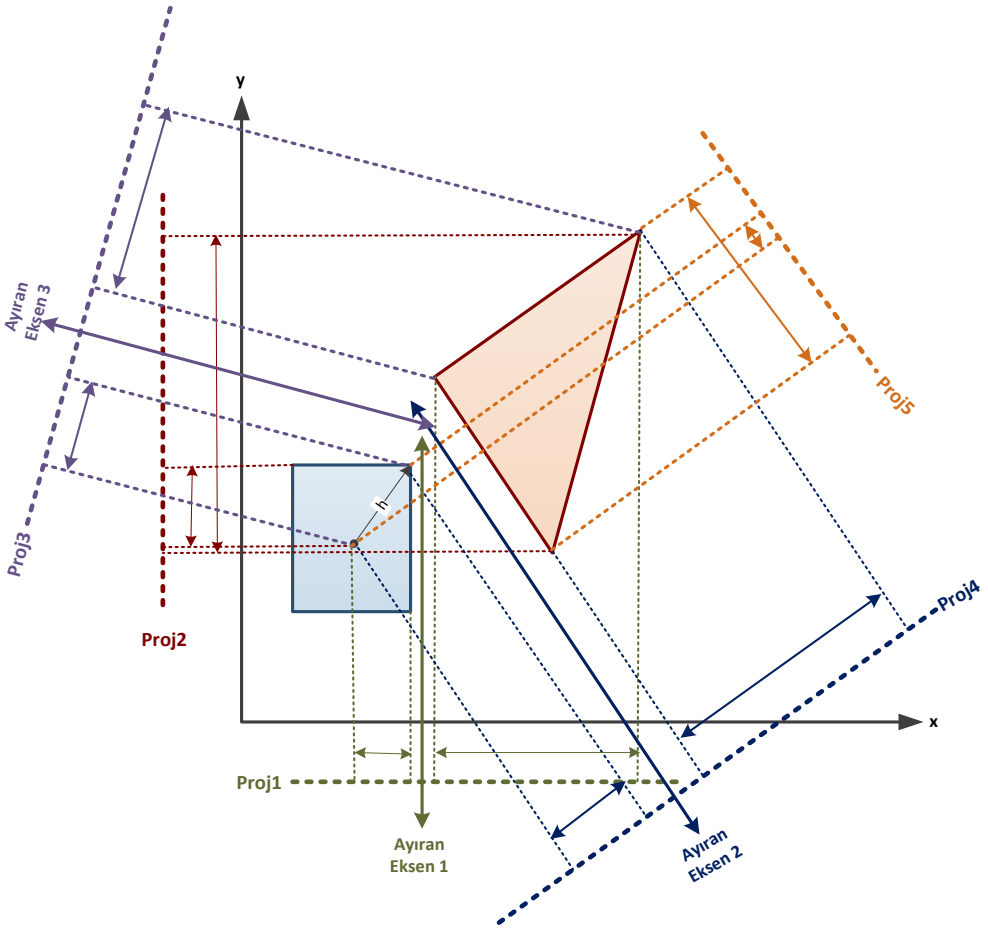
$$r = h \times (|a_{00x}| + |a_{00y}| + |a_{00z}|) \tag{3.16}$$

Üçgen kenarları ve ilgili voksel yüksekliğinin izdüşümleri çakışıyor ise ayıran eksen var demektir. Matematiksel olarak ifade edecek olursak;

EĞER $(min(P_0, P_2) > r \| max(P_0, P_{12}) > -r)$ İSE ayıran eksen vardır
AKSİ HALDE ayıran eksen yoktur

denir.

Şekil 3.11'de üçgen kenarları ve ilgili voksel yüksekliğinin örnek izdüşümleri 2B olarak gösterilmektedir. Burada, prizma ile üçgenin hesaplanan izdüşümlerinin üçünde ayıran eksen bulunuyor iken ikisinde yoktur. O halde, bu iki nesne kesişmiyor denir.



Şekil 3.11. Üçgen kenarları ve ilgili voksel yüksekliğinin a_{ij}'ye göre örnek izdüşümlerinin 2B gösterimi

Bu tez çalışmasında kullanılan algoritmada ilgili voksel ile üçgenin kesişip kesişmediğine, iki nesne arasında ayıran bir eksen bulunup bulunmamasına göre karar verilir. Gerçekleştirilen on testten herhangi birisinde bile ayıran eksen bulunuyor ise iki nesne çakışmıyor denir ve algoritma sonlandırılır. Şekil 3.12'de Möller'in bu çalışmaya uyarlanan sözde kodu gösterilmektedir.

kesisim_testi (Tri, VoxelData[pos], h)

/* 3B üçgen kafes modelindeki bir üçgen ile kaplayan hacmi içerindeki bir vokselin kesişip kesişmediğini test et */
GİRİŞ: u_0, u_1, u_2, üçgenin köşe noktaları, **c,** vokselin merkezi ve **h**, vokselin yüksekliği
ÇIKIŞ: üçgen ile vokselin kesişim kesişmediğini saklayan **boolean** değer

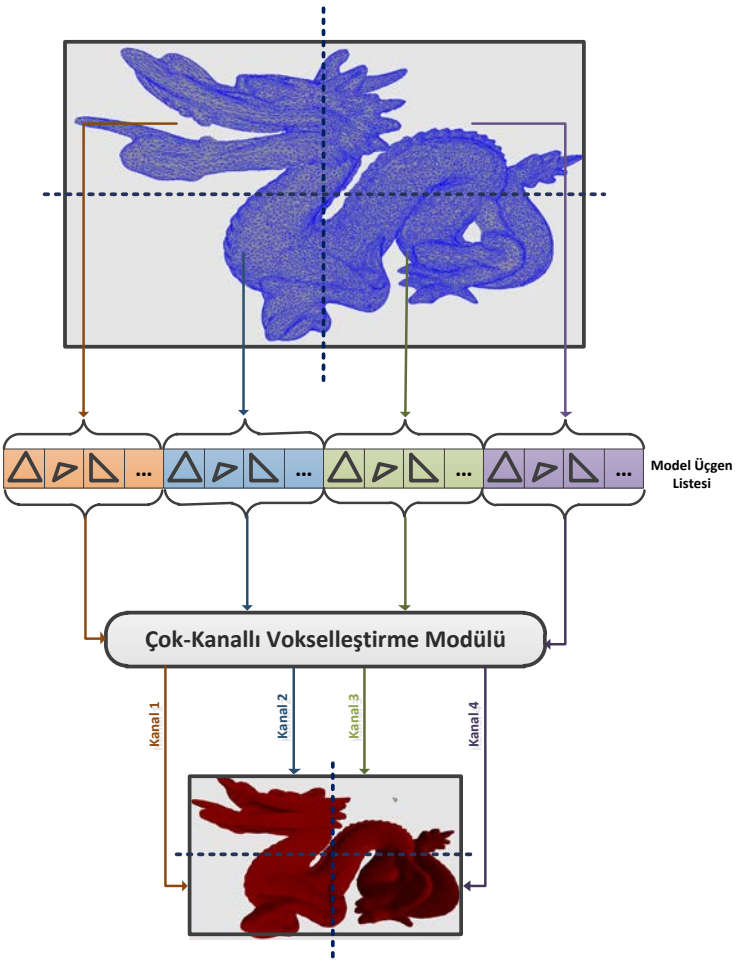
/ voksel ile üçgen başlangıçta kesişmiyor */*
test ← 0;
/ ilgili voksel ve üçgeni, vokselin merkezi koordinat düzleminin merkezi olacak şekilde taşı */*
$v_0 \leftarrow u_0 - c;$
$v_1 \leftarrow u_1 - c;$
$v_2 \leftarrow u_2 - c;$
/ Üçgenin normal düzlemi ile ilgili vokselin kesişimini test et */*
$n \leftarrow f_0 \times f_1;$
$d \leftarrow n \times v_0;$
$r \leftarrow h \times \left(|n_x| + |n_y| + |n_z|\right);$
$s \leftarrow n \bullet c - n \times v_0;$
if ((s - r) <> 0)
    return false;
/ Üçgen kenarları ile voksel yüzeylerinin izdüşümlerinin kesişimini test et */*
$e_0 \leftarrow (1,0,0);$
$e_1 \leftarrow (0,1,0);$
$e_2 \leftarrow (0,0,1);$
for (i=0; i≤2; i++)
    for (j=0; j≤2; j++)
        $a_{ij} \leftarrow e_i \times f_j;$
        for (k=0; k≤2; k++)
            $P_k \leftarrow a_{ij} \cdot v_k;$
            $r \leftarrow h \times \left(|a_{00x}| + |a_{00y}| + |a_{00z}|\right);$
            if $\left(\min(P_0, P_1, P_2) > r \,||\, \max(P_0, P_1, P_2) > -r\right)$
                return false;
return true;

Şekil 3.12. Bir voksel ile üçgenin kesişimini test eden algoritmanın sözde kodu

Geleneksel vokselleştirme algoritmalarının çoğu poligon kesişimini test etmek için sadece voksellerin merkezi kullanılırken burada ise kesişim testi için bir vokselin tüm hacmi ele alınır. Böylece, vokselleştirme algoritmaları için önemli bir kriter olan "doğruluk" artar. Bunun yanı sıra, kesişimi test eden kriter sayısının çok olmasından

dolayı vokselleştirme boyunca harcanan zaman diğer ön-işleme adımlarıma göre daha uzundur. “Hız” ve “doğruluk” arasındaki ters orantı vokselleştirme algoritmalarının tümünde mevcuttur. Fakat bu çalışmada kullanılan kesişim testi algoritmasında her bir üçgen diğerlerinden bağımsız olarak test edildiği için paralleştirmeye uygundur. Böylece, çalışma süresinin uzun olmasından doğan dezavantaj çok-kanallı programlama kullanılarak kayda değer ölçüde azaltılmıştır [28].

Şekil 3.13’de çok-kanallı vokselleştirme modülünün çalışma prensibi gösterilmektedir. Burada, başlangıçta belirtilen kanal sayısına, göre (yani şekle göre dörde) bölünen 3B model üçgen listesindeki üçgenlerin farklı kanallarda (şekle göre dört farklı kanalda) eş-zamanlı olarak vokselleştirilmesi sağlanmaktadır. Böylece, vokselleştirme süreleri kanal sayısı ile orantılı olarak azaltılmış olacaktır.



Şekil 3.13. Çok-kanallı vokselleştirme modülünün çalışma prensibi

Tablo 3.1'de ise farklı üçgen sayılarından oluşan küre, küp tavşan ve ejderha modellerinin farklı kanal sayılarına göre vokselleştirme süreleri gösterilmektedir. Örneğin, 69664 üçgenden oluşan küre modelinin 512x512x512 çözünürlükte sırasıyla 1, 2, 4 ve 8 kanal ile vokselleştirilmesi sonucu geçen süreler üç-basamaklı hassasiyet ile 15.315, 8.490, 4.741, 4.333 şeklindedir. Burada, kanal sayısının sırasıyla 1, 2 ve 4 olması durumlarında vokselleştirme işlemi için geçen sürenin bir öncekine göre yaklaşık olarak %50 oranında azaldığı açıkça görülmektedir. Ancak, hesaplama testlerinin gerçekleştirildiği bilgisayarın 4-çekirdekli olmasından dolayı dört kanal ile yapılan vokselleştirme sekiz kanal ile yapıldığında süre yaklaşık olarak %10-20 civarında düşmektedir. Bu sürelerin 8-çekirdekli bir bilgisayarda test edilmesi durumunda ise 8 kanal ile yapılan vokselleştirme süresinin de aynı şekilde %50 civarında azalacağı ön görülmektedir. Dikkat edilmesi gereken bir diğer nokta ise, çok-kanallı programlama kullanmanın düşük çözünürlüklerde vokselleştirme sürelerini çok fazla etkilemiyor oluşudur. Diğer bir deyişle, çok-kanallı programlama yüksek çözünürlüklerdeki performansı kayda değerdir denilebilir.

Tablo 3.1. Farklı kanal sayılarına ve çözünürlüğe göre vokselleştirme süreleri

Model Adı	**Üçgen Sayısı**	**Çözünürlük**	**Vokselleştirme Süresi (sn)**			
			1 Kanal	**2 Kanal**	**4 Kanal**	**8 Kanal**
Küre	4800	64x64x64	0,508	0,238	0,162	0,127
		128x128x128	2,541	1,136	0,927	0,657
		256x256x256	15,677	7,953	5,205	4,853
		512x512x512	104,855	52,305	38,432	30,194
Küp	12	64x64x64	0,017	0,110	0,070	0,100
		128x128x128	0,747	0,418	0,290	0,347
		256x256x256	3,555	2,096	1,457	1,439
		512x512x512	14,138	8,489	5,998	6,631
Tavşan	69664	64x64x64	0,318	0,160	0,083	0,067
		128x128x128	0,917	0,479	0,267	0,233
		256x256x256	3,203	1,741	1,056	0,949
		512x512x512	15,315	8,490	4,741	4,333
Ejderha	74998	64x64x64	0,226	0,109	0,056	0,052
		128x128x128	0,727	0,362	0,191	0,239
		256x256x256	2,926	1,383	0,839	0,774
		512x512x512	13,998	6,956	3,791	3,769

Bir sonraki bölümde, 3B modelin dış yüzeyi vokselleştirilerek elde edilen 3B ayrık vokseller kümesinin iç hacminin de doldurulabilmesi için kullanılan tohum doldurma algoritması detaylı olarak anlatılmaktadır.

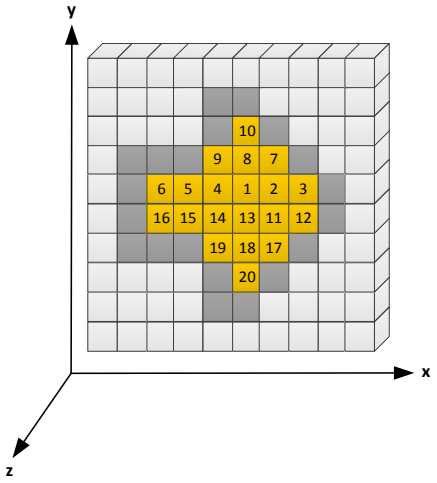
3.1.1.4. İç hacim doldurma

Tohum-doldurma algoritması, raster grafiklerinin temel algoritmalarından birisidir [63]. Algoritma, eğri sınırlarında bir tohum pikseli verildiğinde, bu sınır tarafından çevrelenen tüm komşu pikselleri sırasıyla dolaşır ve bunlara istenilen rengi atar. Böylece, tohum-doldurma algoritması kullanılarak 2B içi dolu nesneler oluşturulabilir. Benzer mantıkla dış yüzeyi vokselleştirilen bir nesnenin iç hacmindeki dolu vokseller de belirlenebilir. Burada da, algoritma iç hacimde rastgele seçilen bir tohum vokselinden başlar ve ilgili vokselin 26-komşusunu sırasıyla ziyaret ederek nesnenin iç hacmini doldurmaya devam eder.

Tüm tohum-doldurma algoritmaları yeni tohumların koordinatlarını saklamak için bir yığıt kullanırlar. Raster grafiklerinde kullanılan tohum doldurma algoritmasını 3B hacim grafiklerine adapte etmek için gereken ilave boyut yığıtın derinliğinin çok hızlı bir şekilde artmasına neden olabilir. Feng ve Soon tarafından geliştirilen algoritma literatürdeki ilk 3B tohum-doldurma algoritması olarak bilinir [27]. Burada bir tohumun sol ve sağ aralıkları yığıttan çıkarıldıktan hemen sonra doldurulur. Daha sonra güncel tohum dört komşu tarama çizgisi boyunca yeni tohumlar arar. Her tarama çizgisi boyunca sadece bir voksel alıyor olması özellikle aralıklar çok geniş olduğunda pek çok tohumun oluşturulmasını ve yığıta eklenmesini engeller. Ancak, algoritma katı model içerisindeki boşlukları ve sınırlardaki iç bukey kısımları atlıyor olması algoritmanın bir dezavantajı olarak sayılabilir.

Önerilen tez çalışmasında kullanılan bu tohum-doldurma algoritmasında +x ve –x yönleri doldurma yönü olarak seçilmiştir ve x koordinatları aynı, fakat y ve z koordinatları 1 artan veya azalan 4 komşu aralıktan seçilen bir veya daha fazla tohumu yığıta ekler. Şekil 3.14'de sabit bir *z* diliminde "1" olarak etiketlenmiş vokselin başlangıç tohumu olması durumunda ilgili dilimin doldurulma sırası

gösterilmektedir. Burada, +x ve –x eksenleri doldurma yönünü ve 2-20 numaralı vokseller de doldurma sırasını göstermektedir.



Şekil 3.14. xy voksel diliminin iç hacminin Feng ve Soon'un tohum-doldurma algoritmasına göre doldurulması

Şekil 3.15'de ise 3B bir yüzey verisinin iç hacminin nasıl doldurulduğunu hesaplayan algoritmanın sözde kodu gösterilmektedir. Bu çalışmada, başlangıç yığıt konumu modelin merkezi olarak belirlenmiştir.

tohum_doldur (VoxelData[res^3])

/ 3B yüzey voksel verisinin iç hacmindeki vokselleri belirle */*
GİRİŞ: VoxelData[res^3], res çözünürlüklü 3B voksel yüzey verisi ve **tohumPos**, iç hacmi doldurmak için rasgele seçilen voksel konum vektörü
ÇIKIŞ: VoxelData[res^3], res çözünürlüklü 3B voksel hacim verisi

```
doluSayisi = 0;
/* stk isimli boş bir yığıt oluştur ve ızgaranın merkezi olarak belirlenen tohum
konumunu ilgili yığıta ekle */
create stack stk;
tohumPos = (res/2, res/2, res/2);
push (stk, tohumPos);

while (stk boş değil)
    pos = pop (stk);
    voxelData[pos] = 1;
    doluSayisi ++;
    /* tohumun x koordinatını sakla */
    X_sakla = pos.x;
    /* tohumun sol mesafesini doldur */
    pos.x ++;
    while (voxelData[pos]) ≠ 1)
        voxelData[pos] = 1;
        doluSayisi ++;
        pos.x ++;

    /* sağ uç vokselin x koordinatını sakla ve bu x koordinatını yeniden ata */
    X_sag = pos.x -1;
    pos.x = X_sakla;
    /* tohumun sol mesafesini doldur */
    pos.x --;
    while (voxelData[pos]) ≠ 1)
        voxelData[pos] = 1;
        doluSayisi ++;
        pos.x --;
```

Şekil 3.15. 3B yüzey verisinin iç hacminin doldurulmasını hesaplayan algoritmanın sözde kodu

```
/* sol uç vokselin koordinatını sakla */
Xsol = pos.x + 1;;
/* 'y+1' olan ön tarama çizgisinin bir sınır veya bir önceki doldurulan olup
olmadığını kontrol et; değilse sol uçtan başlayarak tarama çizgisini doldur */
pos.y ++;
pos.x = Xsol;
/* ilk vokselin değerini sakla */
ilkVoxel = (voxelData[pos] ≠ 1);
pos.x ++;

while (pos.x ≠ Xsag)
    ikinciVoxel = (voxelData[pos] ≠ 1);
    /* iç ve dış vokseller arasındaki sınırları bul */
    if ((ilkVoxel ≠ ikinciVoxel) && (ilkVoxel = true))
        push(stk, pos(x-1,y,z));
    ilkVoxel = ikinciVoxel;
    pos.x ++;

/* son vokseli kontrol et */
if (ilkVoxel = true)
    push(stk, pos(x-1,y,z));

/* 'y-1' üst tarama çizgisini kontrol et */
pos.y = pos.y-2;
/* aynısını 'y+1' için yap */
/* 'z+1' ön tarama çizgisini kontrol et */
pos.y ++;
pos.z ++;
/* aynısını 'y+1' için yap */
/* 'z-1' arka tarama çizgisini kontrol et */
pos.z = pos.z - 2;
/* aynısını 'y+1' için yap */

delete stk;
```

Şekil 3.15. (Devamı)

Vokselleştirme işleminde olduğu gibi çok-kanallı programlama kullanılarak yüzeyi voksel verisi elde edilen modelin, önceden belirlenen kanal sayısına bölünmek suretiyle eş-zamanlı olarak iç hacmi doldurulabilir.

Tablo 3.2'de farklı üçgen sayılarından oluşan küre, küp tavşan ve ejderha modellerinin farklı kanal sayılarına göre tohum doldurma süreleri gösterilmektedir. Örneğin, 69664 üçgenden oluşan küre modelinin 512x512x512 çözünürlükte sırasıyla 1, 2, 4 ve 8 kanal ile iç hacminin doldurulması sonucu geçen süreler üç-basamaklı hassasiyet ile 5.018, 3.191, 2.263, 1.931 şeklindedir. Burada, kanal sayısının sırasıyla 1, 2 ve 4 olması durumlarında tohum doldurma işlemi için geçen sürenin bir öncekine göre azaldığı açıkça görülmektedir. Ancak, hesaplama testlerinin gerçekleştirildiği bilgisayarın 4-çekirdekli olmasından dolayı dört kanal ile yapılan tohum doldurma işlemi sekiz kanal ile yapıldığında süre yaklaşık olarak %10-20 civarında düşmektedir. Bu sürelerin 8-çekirdekli bir bilgisayarda test edilmesi durumunda ise 8 kanal ile yapılan tohum doldurma süresinin de aynı şekilde azalacağı ön görülmektedir. Dikkat edilmesi gereken bir diğer nokta ise, çok-kanallı programlama kullanmanın düşük çözünürlüklerde tohum doldurma sürelerini azaltmak yerine aksine artırıyor oluşudur. Diğer bir deyişle, çok-kanallı programlama yüksek çözünürlüklerdeki performansı kayda değerdir denilebilir.

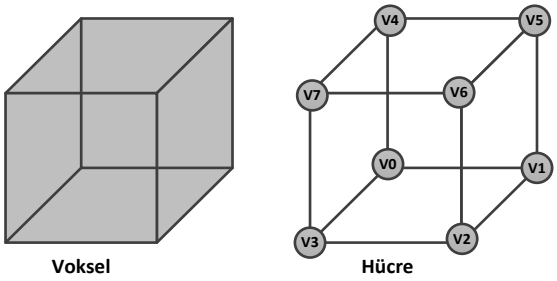
Tablo 3.2. Farklı kanal sayılarına ve çözünürlüğe göre voksel yüzey modelinin iç hacminin doldurulma süreleri

Model Adı	**Üçgen Sayısı**	**Çözünürlük**	**Tohum Doldurma Süresi (sn)**			
			1 Kanal	**2 Kanal**	**4 Kanal**	**8 Kanal**
Küre	4800	64x64x64	0,022	0,023	0,026	0,036
		128x128x128	0,137	0,086	0,128	0,150
		256x256x256	1,133	0,761	0,822	0,718
		512x512x512	24,563	9,974	7,202	6,957
Küp	12	64x64x64	0,032	0,046	0,035	0,041
		128x128x128	0,336	0,238	0,201	0,189
		256x256x256	2,241	1,803	1,828	1,303
		512x512x512	35,153	20,494	19,086	11,480
Tavşan	69664	64x64x64	0,012	0,018	0,019	0,025
		128x128x128	0,064	0,046	0,048	0,035
		256x256x256	0,465	0,322	0,249	0,137
		512x512x512	5,018	3,191	2,463	1,931
Ejderha	74998	64x64x64	0,006	0,006	0,088	0,089
		128x128x128	0,038	0,029	0,120	0,132
		256x256x256	0,209	0,156	0,067	0,095
		512x512x512	1,459	0,956	0,568	0,279

3.1.1.5. Voksel-hücre eşleştirme

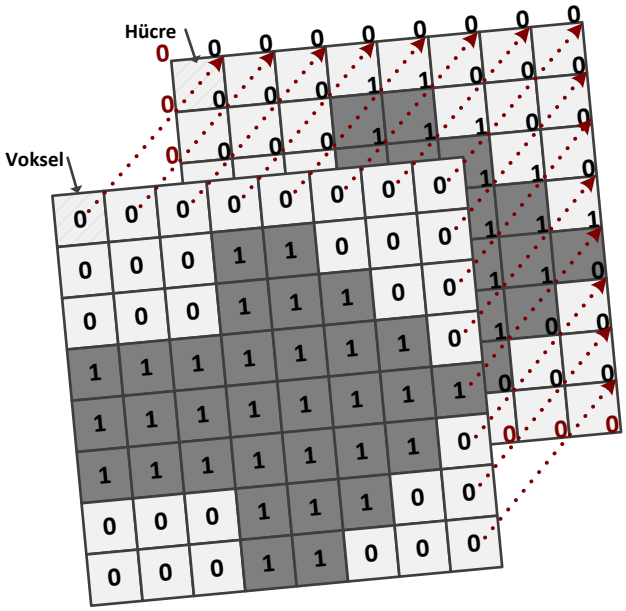
Bir voksel 3B uzayda bir küptür ve tüm hacminde değeri aynıdır. Ayrık ızgarası oluşturulan 3B modelin yüzeyini yeniden oluşturmak için ızgaradaki voksellerin hacim değerleri yerine her bir köşe noktasındaki değerleri gereklidir. Bunun için birbirine komşu sekiz voksel birleştirilip her birinin değeri eş-boyutlu yeni küpün bir köşe noktasına atanır ve bu yeni küp artık voksel yerine hücre olarak adlandırılır.

Aynı şekilde 3B uzayda bir küp olan hücre ise birbirine komşu sekiz vokselin birleşiminden oluşur. Şekil 3.16a'da görüldüğü gibi bu komşu voksellerin değerleri hücrenin ilgili köşe noktalarına atanır. Bu hücre yapısı Knoll ve diğerlerinin geniş sıkıştırılmış hacimlerin eş yüzeylerini ışın izleme ile yeniden oluşturmak için tüm voksel konumlarının yarım voksel genişliğinde geriye doğru lojik olarak kaydırılarak eş hücrelere yeniden atanan çalışmasındakine [64] benzer bir yaklaşımla 3B voksel uzayındaki her bir vokselin değeri "sağ-üst-arka" köşe konumuna kaydırılarak yeniden oluşturulur. Dolayısıyla, hücrenin her köşe noktasına ilgili vokselin ikili değeri atanır ve son olarak da her köşe noktasına atanan bu ikili değerler "$V_7V_6V_5\ldots V_0$" şeklinde birleştirilerek ilgili hücreye [0..255] arasında tamsayı bir değer atanır. Örneğin, bir hücrenin sekiz köşe noktasındaki değerler sırasıyla "0,1,0,0,1,1,0,1" şeklinde ise ilgili hücrenin ikili karşılığı "10110010" ve tamsayı karşılığı ise "178" olur. Bu metot ile optimize edilmiş hash-temelli octree veri yapısı oluşturulmadan önce, 3B hacim ızgarası hücre yapısına göre yeniden düzenlenir ve bundan sonraki işlem adımlarında vokseller yerine her bir voksel hacmine atanan eş hücre değerleri kullanılır. Şekil 3.16'da bir voksel ve bir hücre arasındaki fark gösterilmektedir. Burada, soldaki vokselin tüm hacminde (dış yüzey ve iç kısım) değer aynı iken sağdaki eş hücrenin her köşe noktasındaki değer karşılık gelen vokselin değerine bağlı olarak birbirinden farklı olabilmektedir.



Şekil 3.16. Bir voksel ve eş-hücre

Şekil 3.17'de örnek bir piksel ızgarasından eş-hücre ızgarasının elde edilmesi 2B olarak gösterilmektedir. Burada, başlangıçta tüm değerleri 0 olan eş-hücre ızgarası, öndeki her pikselin değeri arkasındaki eş-hücre ızgarasındaki ilgili alanın sağ üst köşesine atanarak doldurulur. Bu çalışmada, atama işlemi vokselin değeri ilgili eş-hücrenin sağ-üst-arka köşesine kaydırılarak gerçekleştirildi.



Şekil 3.17. Voksel-hücre eşleştirmesinin 2B gösterimi

Şekil 3.18'de ise 3B ayrık voksel veri kümesinde eş-hücre veri kümesinin elde edilmesini sağlayan algoritmanın sözde kodu gösterilmektedir.

voksel_eslestir (VoxelData[res^3])

/ 3B hacim voksel verisinden eş-hücre verisini oluştur */*
GİRİŞ: VoxelData[res^3], res çözünürlüklü 3B voksel hacim verisi
ÇIKIŞ: CellData[res^3], res çözünürlüklü 3B eş-hücre verisi

```
/* x,y,z ϵ {1,2,…,res} olacak şekilde ilgili hacimdeki tüm voksellerin değerlerini
eş-hücrelerin ilgili köşelerine bitsel operatörler kullanarak ata */
for (x=1; x<res; x++)
    for (y=1; y<res; y++)
        for (z=1; z<res; z++)
            index = x*res*res + y*res + z;
            dual = (voxelData[index-res*res-res-1]&64)>>6;
            dual |= (voxelData[index-res-1]&64)>>5;
            dual |= (voxelData[index-res]&64)>>4;
            dual |= (voxelData[index-res*res-res]&64)>>3;
            dual |= (voxelData[index-res*res-1]&64)>>2;
            dual |= (voxelData[index-1]&64)>>1;
            dual |= (voxelData[index]&64);
            dual |= (voxelData[index-res*res]&64)<<1;
            cellData[index] ← dual;

/* x=0 ve y,z ϵ {1,2,…,res} olan sol dilimdeki tüm voksellerin değerlerini
eş-hücrelerin ilgili köşelerine bitsel operatörler kullanarak ata */
x ← 0;
for (y=1; y<res; y++) {
    for (z=1; z<res; z++) {
        index = x*res*res + y*res + z;
        dual  = (voxelData[index-res-1]&64)>>5;
        dual |= (voxelData[index-res]&64)>>4;
        dual |= (voxelData[index-1]&64)>>1;
        dual |= (voxelData[index]&64);
        cellData[index] ← dual;
```

Şekil 3.18. 3B hacim voksel ızgarasından eş-hücre ızgarasının elde edilmesini sağlayan algoritmanın sözde kodu

```
/* x=0 ve y,z ϵ {1,2,…,res} olan sol dilimdeki tüm voksellerin değerlerini
eş-hücrelerin ilgili köşelerine bitsel operatörler kullanarak ata */
x ← 0;
for (y=1; y<res; y++) {
    for (z=1; z<res; z++) {
        index = x*res*res + y*res + z;
        dual  = (voxelData[index-res-1]&64)>>5;
        dual |= (voxelData[index-res]&64)>>4;
        dual |= (voxelData[index-1]&64)>>1;
        dual |= (voxelData[index]&64);
        cellData[index] ← dual;

/* y=0 ve x,z ϵ {1,2,…,res} olan alt dilimdeki tüm voksellerin değerlerini
eş-hücrelerin ilgili köşelerine bitsel operatörler kullanarak ata */
y ← 0;
for (x=1; x<res; x++) {
    for (z=1; z<res; z++) {
        index = x*res*res + y*res + z;
        dual = (voxelData[index-res*res-1]&64)>>2;
        dual |= (voxelData[index-1]&64)>>1;
        dual |= (voxelData[index]&64);
        dual |= (voxelData[index-res*res]&64)<<1;
        cellData[index] ← dual;

/* z=0 ve x,y ϵ {1,2,…,res} olan arka dilimdeki tüm voksellerin değerlerini
eş-hücrelerin ilgili köşelerine bitsel operatörler kullanarak ata */
z ← 0;
for (x=1; x<res; x++)
    for (y=1; y<res; y++)
        index = x*res*res + y*res + z;
        dual  = (voxelData[index-res]&64)>>4;
        dual |= (voxelData[index-res*res-res]&64)>>3;
        dual |= (voxelData[index]&64);
        dual |= (voxelData[index-res*res]&64)<<1;
        cellData[index] ← dual;
```

Şekil 3.18. (Devamı)

```
/* x=0, y=0 ve z ϵ {1,2,…,res} olan sol alt tarama çizgisindeki tüm voksellerin
değerlerini eş-hücrelerin ilgili köşelerine bitsel operatörler kullanarak ata */
x ← 0;
y ← 0;
for (z=1; z<res; z++)
    index = x*res*res + y*res + z;
    dual = (voxelData[index]&64);
    dual |= (voxelData[index-1]&64)>>1;
    cellData[index] ← dual;

/* x=0, z=0 ve y ϵ {1,2,…,res} olan sol arka tarama çizgisindeki tüm voksellerin
değerlerini eş-hücrelerin ilgili köşelerine bitsel operatörler kullanarak ata */
z ← 0;
for (y=1; y<res; y++)
    index = x*res*res + y*res + z;
    dual  = (voxelData[index-res]&64)>>4;
    dual |= (voxelData[index]&64);
    cellData[index] ← dual;

/* y=0, z=0 ve x ϵ {1,2,…,res} olan alt arka tarama çizgisindeki tüm voksellerin
değerlerini eş-hücrelerin ilgili köşelerine bitsel operatörler kullanarak ata */
y ← 0;
for (x=1; x<res; x++)
    index = x*res*res + y*res + z;
    dual  = (voxelData[index]&64);
    dual |= (voxelData[index-res*res]&64)<<1;
    cellData[index] ← dual;

/* x=0, y=0 ve z=0 olan vokselin değerleni eş-hücrenin ilgili köşesine bitsel
operatörler kullanarak ata */
cellData[0] ← voxelData[0]&64;
```

Şekil 3.18. (Devamı)

Voksel-hücre eşleştirmesi üçgen/kutu kesişim testi algoritması kullanılarak vokselleştirilen yüzeyin, yani voksel yüzeyinden üçgen ızgara modelinin yeniden oluşturulması için gereklidir ve bu eşleştirme gerçek-zaman etkileşimi esnasında yapılabileceği gibi burada oluğu gibi ön-işleme zamanında da yapılabilir. Voksel değerlerini ilgili hücrelere atama gibi zaman alıcı bir işi gerçek-zaman etkileşimi

esnasında yapmak zaman-kritik uygulamalar için uygun değildir. Biz, bu işlemi gerçek-zaman yerine ön-işleme sırasında yaparak yüzeyin hem global hem de gerçek-zamanda lokal olarak yeniden oluşturulması sırasında geçen süreyi, böylece de grafik sahneleme döngüsünü kısaltması hedeflenmektedir.

Tezin bundan sonraki bölümlerinde, "eş-hücre" kelimesi yerine "voksel" kelimesi kullanılmaya devam edecektir.

Ön-işlemenin son adımı hash-temelli octree veri yapısının oluşturulmasıdır. Vokselleştirilen üçgen ızgara modelindeki voksellere komşu voksellerin değerlerine göre hücre değerlerini atadıktan sonra oluşturulan sonuç 3B ikili hacim verisi hafıza gereksinimini, hesaplama maliyetini ve ağaç dolanım süresini kısaltmak amacı ile optimize edilmiş hash-temelli bir octree veri yapısına dönüştürülür.

3.1.1.6. Optimize edilmiş hash-temelli octree veri yapısı oluşturulması

Ön-işlemenin son adımı olan optimize edilmiş hash-temelli octree oluşturulması işlemi, çalışmada önerilen temel katkı olduğundan dolayı Bölüm 4'te detaylı olarak incelenmiştir.

3.1.2. Grafik sahneleme döngüsü

Grafik sahneleme modülünün ikinci adımı olan grafik sahneleme döngüsü, sanal model ve sanal araçtan oluşan sahnenin 30 Hz'lik bir frekansta sürekli olarak sahnelenmesi ile ilgilenmektedir.

Voksel yarıçapında oluşturulmuş bir küre olan sanal aracın boyutu ve şekli grafik sahneleme döngüsünün her çevriminde aynı iken konumu, önerilen sanal heykeltraşlık sistemine entegre edilmiş olan haptic cihazının mili tarafından kontrol edilmektedir.

Sanal modelin, yani üzerinde yontma işlemi gerçekleştirilen sanal çalışma parçasının yüzeyi, optimize edilmiş hash-temelli octree veri yapısındaki voksel verilerine bağlı olarak grafik sahneleme döngüsünün her çevriminde yeniden oluşturulmaktadır.

Grafik sahneleme döngüsünün en başında, ön-işleme adımının hemen ardından sanal modelin eş-yüzeyinin tümüyle yeniden oluşturulması gerekmektedir. İlerleyen çevrimlerde ise, haptic cihazının miline bağlı olan sanal aracın sanal model ile çarpışması durumunda, modelin eş-yüzeyi optimize edilmiş hash-temelli octree veri yapısında güncellenen voksellerin değerlerine göre bölgesel olarak yeniden oluşturulmaktadır.

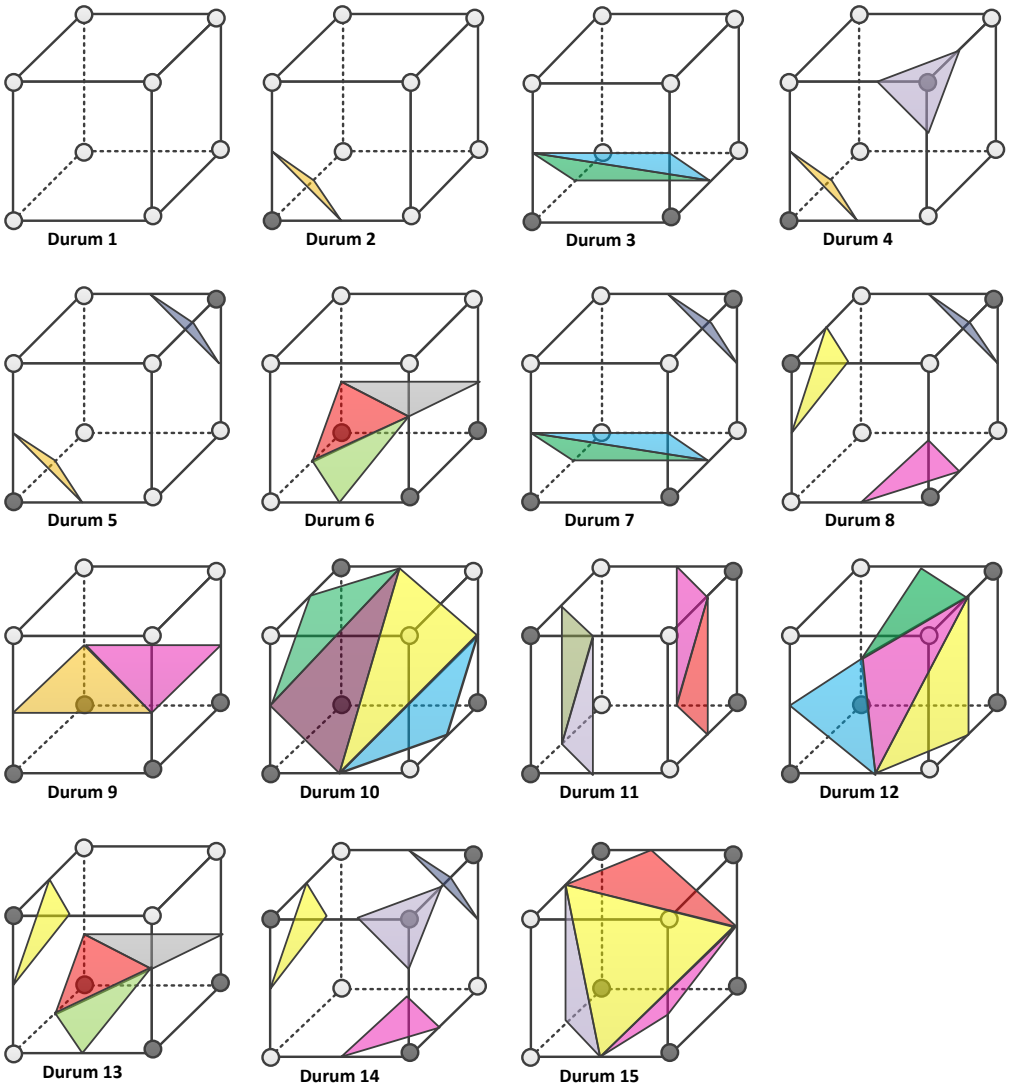
Bu bölümde, eş-yüzeyin global ve lokal oluşturulmasını sağlayan Marching Cubes algoritması detaylı olarak açıklanmıştır.

3.1.2.1. Marching Cubes

Marching Cubes, Lorenson ve Cline tarafından 3B hacimsel voksel verisinden üçgen ızgara oluşturan bir eş-yüzey oluşturma algoritmasıdır [24]. Algoritmanın temel amacı, 3B hacimsel verideki her bir voksele karşılık bir veya birden fazla eşyüzey oluşturmaktır.

Algoritma, 3B hacimsel veride voksel olarak adlandırılan bir küpün sekiz köşe noktasının önceden belirlenen bir eşik değerine göre “0” veya “1” olarak etiketlenmesi ile başlar. Bu çalışmada, 3B hacimsel veri kümesindeki her bir vokselin köşe noktaları, zaten “0” veya “1”lerden oluşan ikili değerler aldığından dolayı yeniden etiketlenmesine gerek yoktur.

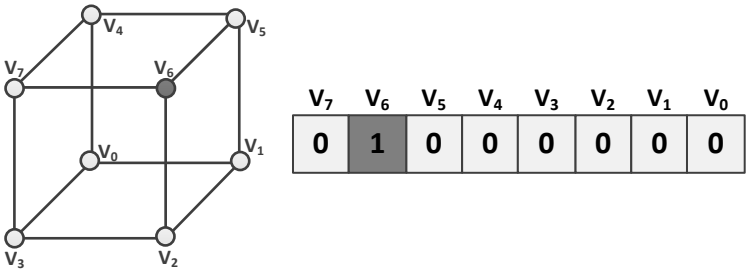
Bir vokselin 8 köşe noktasının her birinin 2 farklı değeri (“0” veya “1”) olmasından dolayı, voksele karşılık gelen üçgen(ler) hesaplanırken toplamda 2^8=256 farklı sonuç elde edilmektedir. Ancak, bunlar simetri düşünüldüğünde 15 farklı duruma indirgenir [64]. Şekil 3.19’da voksellere karşılık gelen üçgen(ler)in oluşturulması için kullanılan 15 farklı kalıp gösterilmektedir.



Şekil 3.19. Marching Cubes algoritmasında eş-yüzey oluşturmak için kullanılan 15 kalıp

Marching Cubes algoritmasında giriş, bir voksel, çıkış ise bir veya birden fazla üçgendir. Algoritma, “0” veya “1” lerden oluşan kısmen dolu değere sahip vokselleri kullandığından dolayı 3B hacimsel veri Algoritma, kümesindeki bu voksellerden elde edilen üçgen ızgara ise “eş-yüzey” olarak adlandırılır.

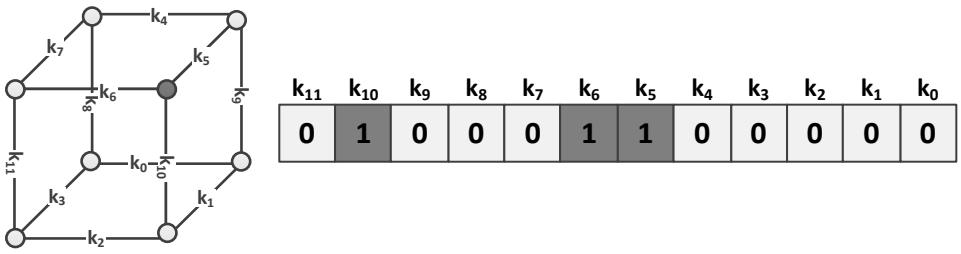
Vokselin köşe noktalarının birleştirilmesiyle oluşan bir tamsayı değere göre yürütülen algoritmada indeks adı verilen bu değerin nasıl hesaplandığı Şekil 3.20’de gösterilmektedir.



Şekil 3.20.Örnek ikili voksel için indeks değeri hesaplama

Burada, vokselin köşelerinin ikili değerlerinin birleştirilmesiyle elde edilen vokselin indeks değeri 64 ‘tür.

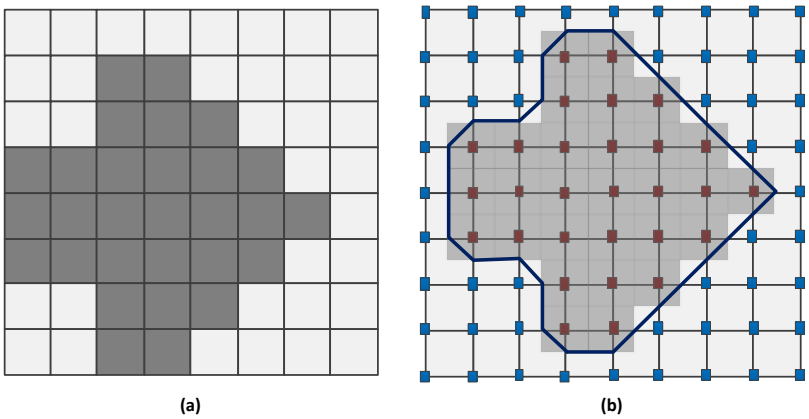
Algoritmada, kenar ve üçgen tablosu olarak adlandırılan iki okuma tablosu kullanılır. Kenar tablosu, voksellerin indeks değerlerine bağlı olarak oluşturulacak üçgenlerin, vokselin hangi kenarlarını kestiğini gösteren ve 256 farklı durum içeren tek-boyutlu bir dizidir. Şekil 3.21’de örnek ikili voksel için kenar değerinin hesaplanması gösterilmektedir.



Şekil 3.21. Örnek voksel için kenar değeri hesaplama

Bir kenarın iki köşesindeki voksellerin değerlerinden biri “1” diğeri “0” ise “1”, aksi halde “0” atanır. Burada, “1” değerine sahip V6 köşe noktasına göre oluşturulacak üçgen, k5, k6, k10 köşelerini kesmektedir. Buna göre, üçgen tablosunda bu kenar değerine karşılık gelen üçgen(ler)in verisi bulunarak elde edilen veri grafik sahneleme döngüsüne gönderilmek suretiyle üçgen listesinde kaydedilmektedir.

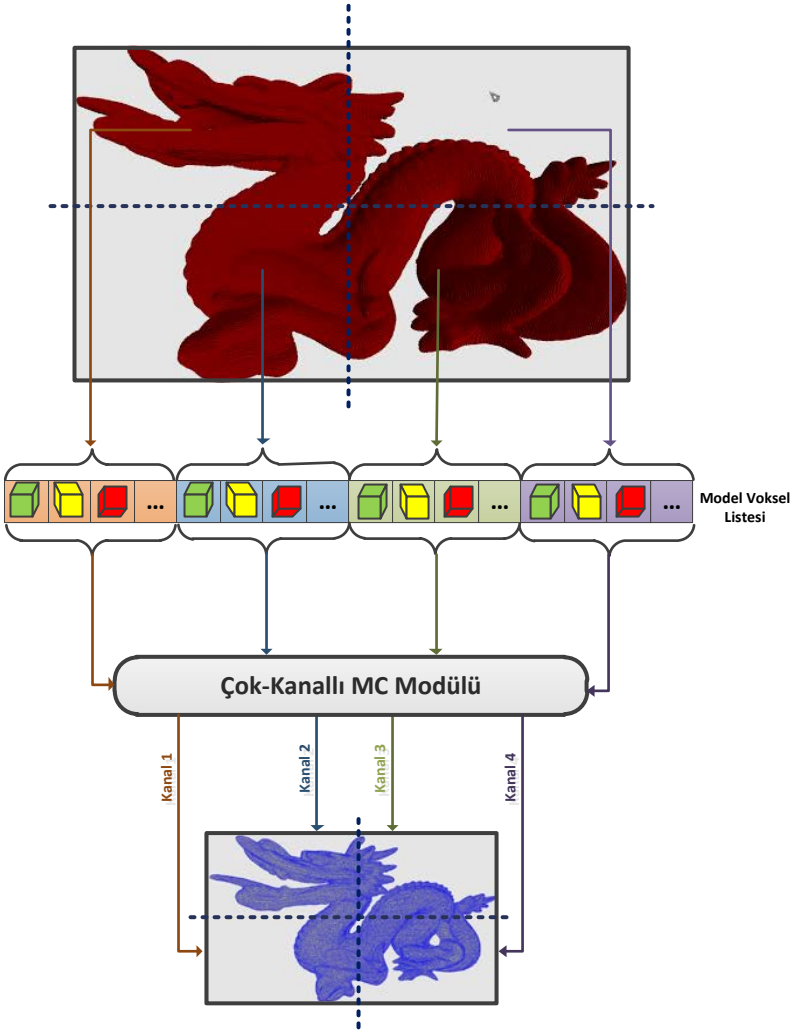
Şekil 3.22’de Marching Cubes algoritmasının örnek 2B gösterimi verilmektedir. Burada, 8x8’lik örnek piksel ızgarasında bulunan resmin algoritma uygulanmadan önce eş-hücre ızgarasına dönüştürüldüğüne dikkat edilmelidir.



Şekil 3.22. Marching Cubes algoritmasının 2B örnek gösterimi

Bu çalışmada, vokselleştirme işleminde olduğu gibi çok-kanallı programlama kullanılarak önceden belirlenen kanal sayısına bölünmek suretiyle hacimsel voksel verisi elde edilen modelden, eş-zamanlı olarak ilgili modelin eş-yüzeyi elde edilmektedir. [29].

Şekil 3.23 ’de çok-kanallı MC modülünün çalışma prensibi gösterilmektedir. Burada, başlangıçta belirtilen kanal sayısına, göre (yani şekle göre dörde) bölünen 3B model üçgen listesindeki üçgenlerin farklı kanallarda (şekle göre dört farklı kanalda) eş-zamanlı olarak vokselleştirilmesi sağlanmaktadır. Böylece, vokselleştirme süreleri kanal sayısı ile orantılı olarak azaltılmış olacaktır.



Şekil 3.23. Çok-kanallı MC modülünün çalışma prensibi

Tablo 3.3'de ise farklı üçgen sayılarından oluşan küre, küp, tavşan ve ejderha modellerinin farklı kanal sayılarına göre MC algoritması kullanılarak eş-yüzey oluşturma süreleri gösterilmektedir. Örneğin, 69664 üçgenden oluşan küre modelinin 512x512x512 çözünürlükte sırasıyla 1, 2, 4 ve 8 kanal ile eş-yüzeyinin oluşturulması sonucu geçen süreler üç-basamaklı hassasiyet ile 6.972, 3.675, 2.121, 1.869 şeklindedir. Burada, kanal sayısının sırasıyla 1, 2 ve 4 olması durumlarında MC kullanılarak eş-yüzey oluşturma işlemi için harcanan sürenin bir öncekine göre yaklaşık olarak %50 oranında azaldığı açıkça görülmektedir. Ancak, hesaplama testlerinin gerçekleştirildiği bilgisayarın 4-çekirdekli olmasından dolayı dört kanal ile yapılan eş-yüzey oluşturma işlemi sekiz kanal ile yapıldığında süre yaklaşık olarak %10-20 civarında düşmektedir. Bu sürelerin 8-çekirdekli bir bilgisayarda test edilmesi durumunda ise 8 kanal ile yapılan eş-yüzey oluşturma süresinin de aynı

şekilde azalacağı ön görülmektedir. Dikkat edilmesi gereken bir diğer nokta ise, çok-kanallı programlama kullanmanın düşük çözünürlüklerde eş-yüzey oluşturma sürelerini çok fazla etkilemiyor oluşudur. Diğer bir deyişle, çok-kanallı programlama yüksek çözünürlüklerdeki performansı kayda değerdir denilebilir.

Tablo 3.3. Farklı kanal sayılarına ve çözünürlüğe göre MC süreleri

Model Adı	Üçgen Sayısı	Çözünürlük	Eş-yüzey Oluşturma Süresi (sn)			
			1 Kanal	2 Kanal	4 Kanal	8 Kanal
Küre	4800	64x64x64	0,177	0,093	0,041	0,037
		128x128x128	0,714	0,421	0,301	0,278
		256x256x256	2,872	1,685	0,962	0,876
		512x512x512	11,426	6,421	3,169	2,989
Küp	12	64x64x64	0,242	0,122	0,093	0,086
		128x128x128	1,343	0,657	0,405	0,386
		256x256x256	8,218	4,126	2,308	1,789
		512x512x512	11,439	6,105	3,720	2,124
Tavşan	69664	64x64x64	0,108	0,084	0,079	0,062
		128x128x128	0,432	0,265	0,149	0,126
		256x256x256	1,765	0,981	0,576	0,441
		512x512x512	6,972	3,657	2,121	1,469
Ejderha	74998	64x64x64	0,099	0,064	0,048	0,039
		128x128x128	0,392	0,215	0,189	0,122
		256x256x256	1,549	0,769	0,386	0,275
		512x512x512	6,250	3,124	2,241	1,785

İlerleyen bölümde, önerilen haptic-temelli sanal heykeltıraşlık uygulamasında hafıza optimizasyonu sağlamak amacıyla kullanılan octree veri yapısı ve bu veri yapısının hash tabloları kullanılarak optimize edilmesini öneren yaklaşım detaylı olarak açıklanmaktadır.

BÖLÜM 4. OPTİMİZE EDİLMİŞ HASH-TEMELLİ OCTREE

Voksel gösteriminin ilk ve en basit yolu her bir vokselin uzayda eşit boyutlu bir küp olduğu 3B ayrık ızgara gösterimidir. Ancak çözünürlük arttıkça nesneyi göstermek için gerekli olan voksel sayısı ve buna bağlı olarak da hafıza maliyeti artar. Örneğin, 3B bir nesneyi 1024x1024x1024 çözünürlükte göstermek için bir milyardan fazla voksele ihtiyaç duyulur. Hafıza maliyeti voksel-temelli uygulamalarda kritik bir problemdir ve bu maliyeti optimize etmek için literatürde 3B uzayın rekürsif olarak alt bölümlere ayrıldığı ikili uzay bölümleme (BSP) ağacı [66], quadtree [39,67] ve octree [7,15] gibi hiyerarşik veri yapıları kullanılmıştır. Sanal heykeltraşlık uygulamalarında bu veri yapıları arasından en çok tercih edileni, uzayı her üç eksende de eşit olarak bölüyor olması sebebiyle octree olmuştur.

Sanal heykeltıraşlık uygulaması boyunca gerekli olan 3B voksel hacim verisinin yüksek hafıza ve hesaplama maliyetini düşürmeyi, aynı zamanda da heykeltıraşlık işlemine gerçek-zaman performansı kazandırabilmeyi amaçlayan bu tez çalışmasında da octree veri yapısını kullanılmıştır. Bu amaçla, optimize-edilmiş hash-temelli bir octree veri yapısı önerilmiştir.

Bu bölümde, sırasıyla, octree kavramı, ağacı oluşturmak için kullanılan yaklaşımlar, önerilen optimize-edilmiş hash-temelli octree veri yapısının voksel verisi üzerine uygulanmasının detayları ve bu veri yapısının diğer octree veri yapılarına göre (işaretçi-temelli ve hash-temelli) avantajları anlatılmaktadır.

Bu çalışmada kullanılan şekillerde octree ve önerilen optimize-edilmiş hash-temelli octree veri yapısını anlatmak için kâğıt üzerinde 3B ifade etmenin zorluğundan dolayı 2B piksel-temelli quadtree kullanılacaktır.

4.1. Octree

Octree, 3B nesneleri tanımlamak için kullanılan bir veri yapısıdır. Bilgisayar grafiklerinde, bilgisayar görmesinde, görüntüleme ve görüntü işlemede yaygın olarak kullanılır. Octree veri yapısının temeli, voksel değerlerine bağlı olarak uzayın alt bölümlere ayrılmasıdır [68,69].

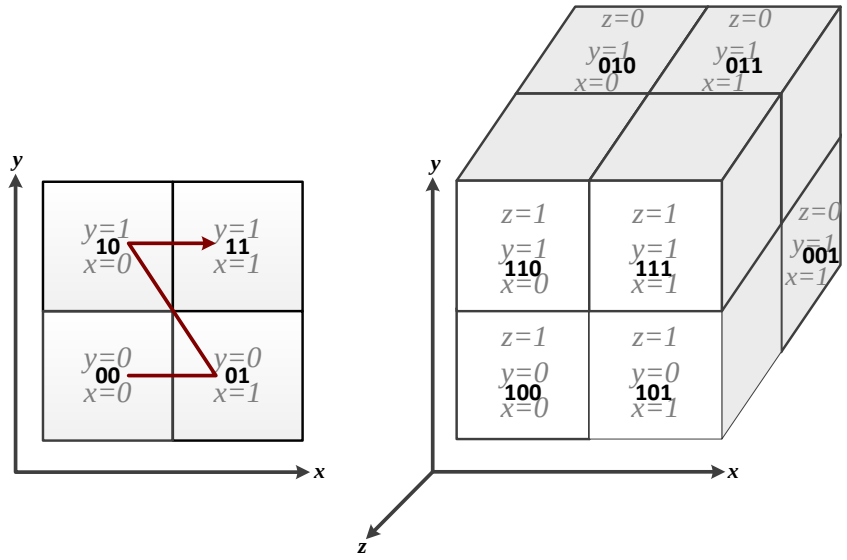
Octree veri yapısının içerisinde önceden bilinmesi gereken terimler ve açıklamalar aşağıda listelenmiştir. Bazı terimlerin detayları ilerleyen bölümlerde ayrıca verilecektir.

- Düğüm: Alt uzayı gösteren hacimdir. Her düğüm, sekiz alt düğümden oluşur (yaprak düğüm hariç).
- Kök Düğüm: Tüm voksel hacmini gösteren ağacın en tepesindeki düğümdür.
- Yaprak Düğüm: Bölünebilecek en küçük alt uzayı gösteren ağacın en ucundaki düğümdür.
- Ebeveyn Düğüm: Alt uzayı gösteren düğümün içinde bulunduğu üst uzayı gösteren düğümdür. Her alt uzay bir ebeveyn düğüme bağlıdır (kök düğüm hariç).
- Çocuk Düğüm: Üst uzayı gösteren ebeveyn düğümün sekiz alt uzayından birisini gösteren düğümdür. Bu düğüm, octree veri yapısında oktant, quadtree veri yapısında ise quadrant olarak adlandırılır.
- Seviye / Derinlik: Çözünürlüğe bağlı olarak ağacın alt uzaylara bölünme sayısıdır. (64x64x64 çözünürlüğe sahip bir hacim için octree seviyesi, $l = \log_2 64 = 6$'dır.)
- Dolanım yönü: Ağacın, çocuk düğümleri ziyaret etme sırasıdır. Ağaç oluşturulmadan önce eksenlerin dolaşılacağı sıraya göre belirlenen bu yön, ağacın her seviyesinde aynıdır.
- Yer Kodu: Düğümün, uzaydaki konumunu gösteren etikettir. Bu kod, önceden belirlenen bir eksen sıralamasına göre atanır.
- Eksen Sıralaması: Yer kodunu belirlemek için alt uzayın, çalışılan uzayın merkezinin altında/üstünde, sağında/solunda veya önünde/arkasında

kalmasına göre ilgili alt uzaya bitsel “0” veya “1” değerlerinin atanacağı sıralamadır (xy(z), (z)yx, vs.) [70].

- Anahtar: 3B voksel hacmindeki yer kodlarına bağlı olarak hesaplanan ve uzayın/alt uzayın ağaçtaki konumunu gösteren değerdir.

2B/3B ağaç oluşturulurken çocuk düğümlerin ziyaret edilme sırasına ağacın dolanım yönü, ziyaret edilme sırasına göre ilgili düğümlere verilen etiketlere de yer kodu denir. Burada dikkat edilmesi gereken husus seçilen dolanım yönünün ve eksen sıralamasının her derinlikte aynı olması gerektiğidir. Aynı durum 3B octree veri yapısı için de geçerlidir Şekil 4.1’de 2B/3B uzayda (z)yx sıralamasına göre herhangi bir derinlikteki örnek dolanım yönleri ve bu sıralamaya göre uzay/alt uzaya karşılık gelen yer kodları gösterilmektedir



Şekil 4.1. 2B/3B (z)yx sıralamasına göre herhangi bir derinlikteki dolanım yönü ve yer kodları

Burada, (z)yx eksen sıralamasına göre atanan alt uzayların üzerilerindeki bitsel ifadeler yer kodlarını göstermektedir. Yer kodlarının tamsayı karşılıkları ise dolanım yönünü belirler. Sol taraftaki 2B uzaya bakacak olursak, sırasıyla sol at, sağ alt, sol üst ve sağ üst düğümlerin ziyaret edileceğini gösteren kırmızı ok dolanım yönünü göstermektedir ve “ters Z” olarak ifade edilir. Sağ taraftaki 3B uzayda ise dolanım yönü, önce arka, sonra ön hacim dilimlerinin sırasıyla sol alt, sağ alt, sol üst ve sağ

üst düğümlerinin ziyaret edilmesi şeklindedir. Bu çalışmada kullanılan octree dolanım yönü de Şekil 4.1'deki gibi belirlenmiştir.

4.2. Octree Yaklaşımları

Octree veri yapısı, ağacı oluşturmak için kullanılan dolanım yönüne bağlı olarak yukarıdan-aşağıya ve aşağıdan-yukarıya olmak üzere genel olarak iki farklı yaklaşımla oluşturulur.

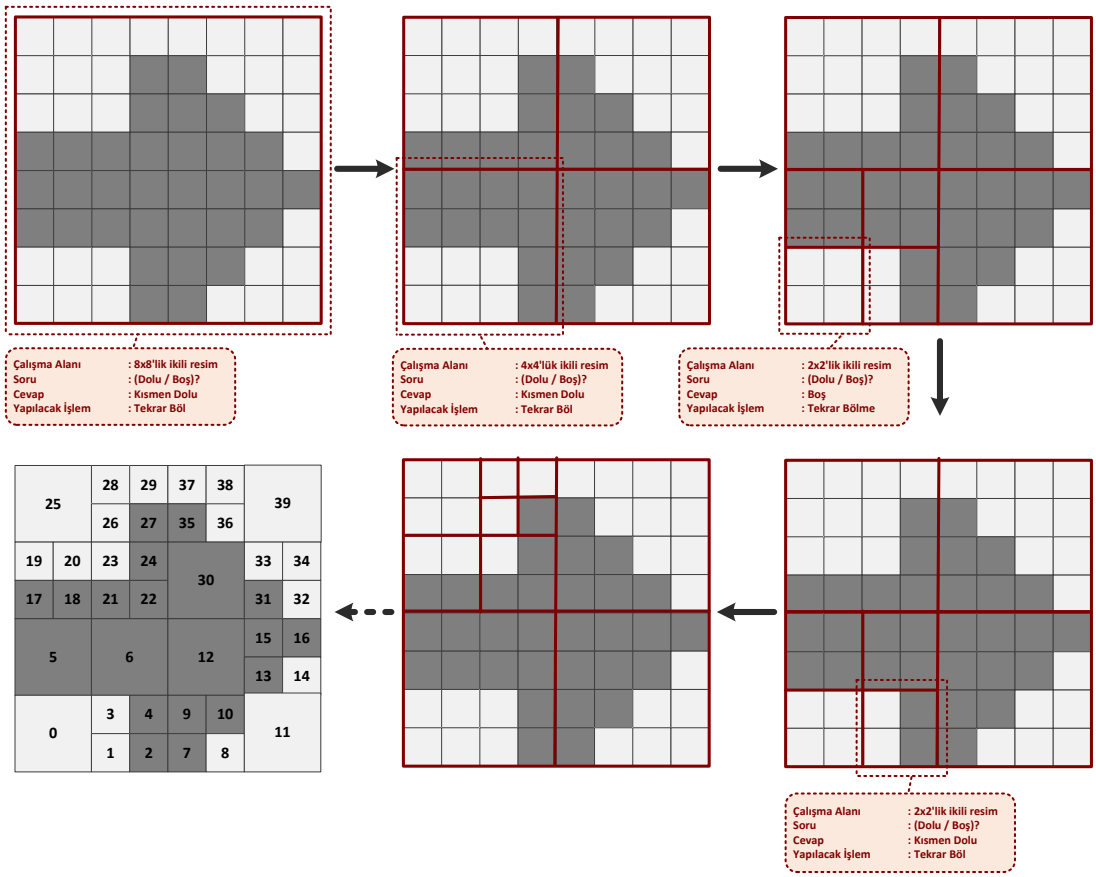
Bu iki farklı yaklaşımın temel kriteri yeni düğüm oluşturmak veya silmek için işlem yapılan hacim üzerindeki voksellerin tamamen dolu (tümü nesne vokseli), tamamen boş (tümü boş uzay vokseli) veya kısmen dolu (kısmen nesne, kısmen boş uzay vokseli) olup olmadığının belirlenmesidir.

4.2.1. Yukarıdan-aşağıya octree oluşturma

Yukarıdan-aşağıya octree yaklaşımında, ağaç, önce tüm hacim içerisindeki voksellerin tamamen dolu, boş veya kısmen dolu olup olmadığına bakar. Hacim kısmen dolu ise sekiz alt hacime bölünerek aynı işlem sırayla ilgili alt uzaylar üzerinde yapılır. Bu işlem, çözünürlüğe bağlı olarak hesaplanan ağaç derinliğine ulaşılıncaya kadar devam eder.

Bu yaklaşım, düşük çözünürlükte başarılı sonuç verirken, çözünürlük yüksek olduğunda uygulanması çok uzun zamanlar alacağı için tercih edilmez.

Şekil 4.2'de 8x8'lik örnek piksel-temelli ikili resim için yukarıdan-aşağıya yaklaşımı ile quadtree oluşturulmasının ilk bir kaç adımı ve görüntünün alt uzaylarının oluşturulma sırasına göre etiketlenmiş son hali gösterilmektedir.



Şekil 4.2. 8x8'lik örnek bir ikili resimden yukarıdan-aşağıya yaklaşımı ile quadtree oluşturulması

Burada dolanım yönü ters Z olarak belirlenmiştir. Yani, ağacı oluşturmak için düğümlerin dolanım sıralaması ilk önce sol alt, daha sonra da sırasıyla sağ alt, sol üst ve sağ üst şeklindedir. Yukarıdan-aşağıya quadtree oluşturmak için ilk önce "0" ve "1"lerden oluşan ikili resim, 4x4'lük 4 resim parçası oluşturacak şekilde dört eşit alt uzaya bölünerek başlanır. Daha sonra, sol alt düğümdeki ikili resim parçasının tamamen "0" veya tamamen "1"lerden mi yoksa "0" ve "1"lerden mi oluştuğuna bakılır. Resim parçası kısmen dolu ise, yani "0" ve "1"lerden oluşuyor ise tekrar dörde bölünür. Yeniden bölünen 2x2'lik resim parçasında dolanım yönünün aynı olması kuralına bağlı kalarak sırasıyla sol alt, sağ alt, sol üst ve sağ üstte yer alan piksellerin değerlerine bakılır. Bu piksellerin tamamı "0" veya tamamı "1" mi yoksa kısmen "0" ve "1" mi olduğu kontrol edilerek 2x2'lik resmin bölünüp bölünmeyeceğine karar verilir. Resim parçası tamamen "0" veya tamamen "1"lerden oluşuyor ise bölünmez. Aksi halde, yani resim parçası kısmen "0" ve kısmen "1"lerden oluşuyor ise resim tekrar dörde bölünür. Dolanım, sağ altta yer alan

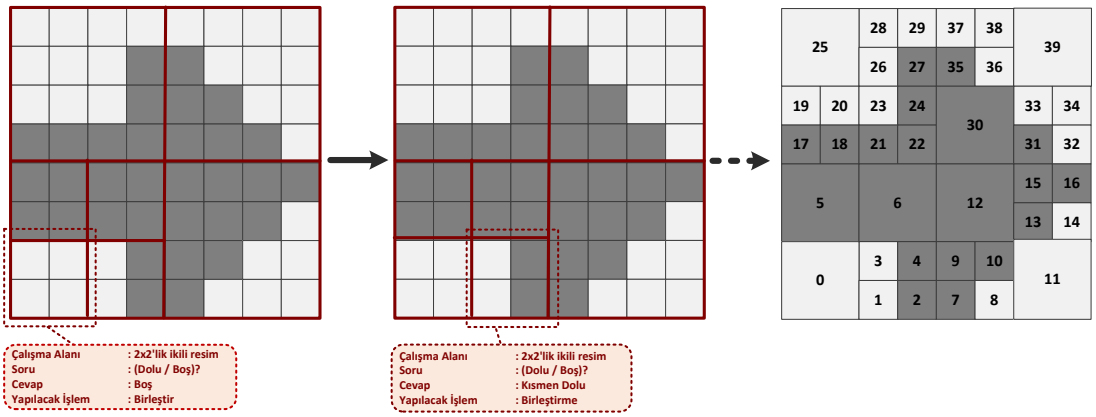
4x4'lük diğer resim parçasına geçerek devam eder ve tüm resmin bu şekilde gezilmesi ile yukarıdan-aşağıya yaklaşımı ile ağaç oluşturulmuş olur.

4.2.2. Aşağıdan-yukarıya octree oluşturma

Aşağıdan-yukarıya octree yaklaşımında, tüm yaprak düğümleri içerecek şekilde boş olarak oluşturulan ağaca, uygun dolanım sırasına göre karşılık gelen vokseller yerleştirilerek başlanır. Daha sonra, alt uzayların tamamen dolu, boş veya kısmen dolu olmasına bağlı olarak düğümler birleştirilerek aşağıdan-yukarıya doğru ilerlenir. Bu şekilde, tamamen dolu veya boş alt uzay kalmayıncaya kadar devam edilir.

Bu yaklaşım, yukarıdan-aşağıya yaklaşımının aksine tek seferde çok büyük alt uzayların dolu veya boş olmasına bakmak yerine sadece sekiz çocuk düğümün değeri ile ilgilendiği için çok büyük boyutlu görüntülere uygulanabilir.

Şekil 4.3'de, 8x8'lik resim için aşağıda-yukarıya yaklaşımı ile quadtree oluşturulmasının ilk iki adımı ve görüntünün alt uzayların oluşturulma sırasına göre etiketlenmiş son hali gösterilmektedir.

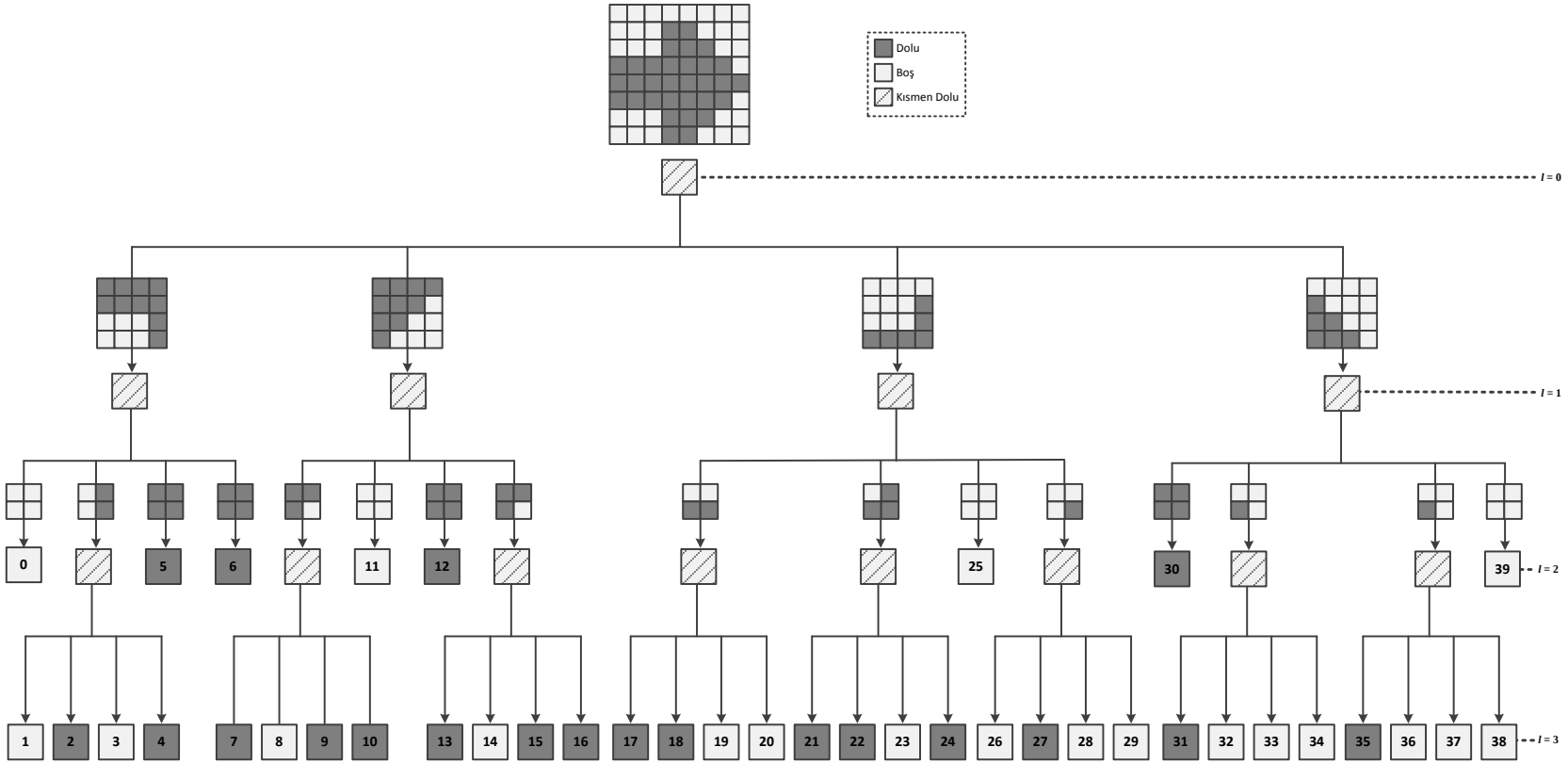


Şekil 4.3. 8x8'lik örnek bir ikili resimden aşağıdan-yukarıya yaklaşımı ile quadtree oluşturulması

Burada, dolanım yönü aynı şekilde ters Z olarak belirlenmiştir. Aşağıdan-yukarıya quadtree oluşturmak için ilk önce, "0" ve "1"lerden oluşan ikili resmin sol alt köşesinde yer alan 2x2'lik resim parçası ele alınır. Daha sonra, bu resim parçasının

tüm pikselleri tamamen “0” veya tamamen “1” veya kısmen “0” ve “1”lerden mi oluştuğuna bakılır. Resim parçası, eğer tamamen dolu veya boş ise bu dört piksel birleştirilir ve dolanım sırasına göre sırasıyla sağ alt, sol üst ve sağ üst 2x2’lik resim parçasının değerlerine bakılır. Bu dört alt resim parçasının herhangi bir tanesinin bile kısmen dolu olması bu alt uzayları içeren 4x4’lük resim parçasının (sol alt) da kısmen dolu olduğu anlamına gelir. Bu durumda, sol alttaki 4x4’lük resim parçası atlanarak, dolanım sağ alt 4x4’lük resim parçasının sol alt 2x2’lik resim parçasına bakılarak devam eder. Bu şekilde, tüm resim parçaları ziyaret edilir ve ağaç oluşturulur.

Şekil 4.4’de her iki yaklaşımla oluşturulan quadtree veri yapısının ağaç gösterimi verilmektedir. Burada, ağacın yaprakları önceden belirlenen ters Z dolanım yönüne uygun olarak yerleştirilir. Yani, soldan sağa doğru düğümlerin sıralaması, sırasıyla ikili resimdeki sol alt, sağ alt, sol üst ve sağ üst alt uzaylar şeklindedir.



Şekil 4.4. 8x8'lik örnek resim için hiyerarşik quadtree gösterimi

Burada, derinlik çözünürlüğe bağlı olarak hesaplanır. Şekildeki resmin çözünürlüğü 8x8 olduğundan dolayı $l = log_2 8 = 3$ olarak bulunur. Başlangıçta kök düğümün seviyesi "0"dır ve hesaplanan derinliğe ulaşıncaya kadar (yani, l = 3 oluncaya kadar) ağacın her seviyesinde "1" artar.

4.3. Octree Çeşitleri

Octree tüm uzayı gösteren ağacın bir kök düğümü ile başlar. Daha sonra kök düğüm, her bir düğümü, çocuk düğüm olarak adlandırılan ve uzayın *x*, *y* ve *z* yönlerinde eşit olarak bölünen sekiz alt uzayından birini gösteren sekiz oktanta sahip olur. Bu bölme işlemi her çocuk düğüm için önceden belirlenen bir octree derinliğine veya çözünürlüğe erişinceye kadar devam eder. Ağacın en alt seviyesindeki oktantlar ise 3B hacim uzayındaki voksellere karşılık gelir.

Literatürde, octree veri yapılarını hafızada saklamak amacıyla çeşitli veri yapıları önerilmiştir. Doğrusal ve işaretçi-temelli metotlar bunlar arasında en yaygın olanlarıdır [71].

4.3.1. Doğrusal octree

Doğrusal octree, hafızada sadece yaprak düğümleri, yani voksellerin 3B hacim uzayındaki konumlarına göre hesaplanan ve anahtar olarak adlandırılan değerlerini sırasıyla saklar. Böylece, doğrusal octree metotlarında işaretçilere ihtiyaç duyulmaz. Her bir yaprak düğüm şifrelenip saklanarak hafızadan tasarruf edilir. Ancak, doğrusal octree de bir düğüme ulaşmak (komşu voksellerin bulmak, vs.) için düğümlerin kaydedildiği veri yapısının tek boyutlu olmasından dolayı ağacı dolaşmak karmaşık ve zaman alıcı bir işlemdir.

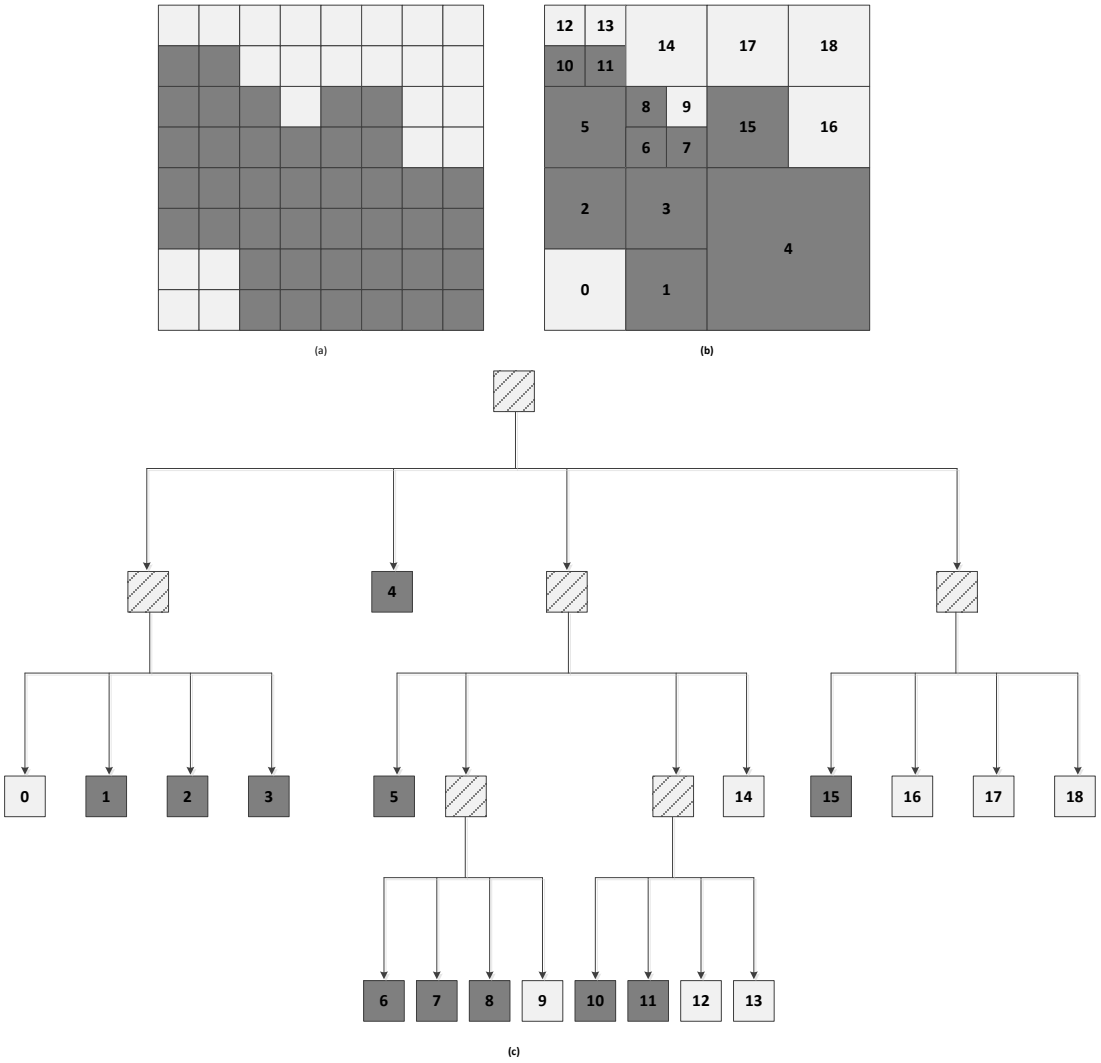
4.3.2. İşaretçi-temelli octree

Doğrusal octree metotlarının aksine, işaretçi-temelli octree metotlarında her bir ebeveyn düğüm, sekiz çocuk düğümünün adreslerini saklamak zorundadır. Oktant

olarak da adlandırılan bu çocuk düğümler modeli nasıl kapsadığına bağlı olarak dolu, boş veya kısmen dolu olarak etiketlenir.

İşaretçi-temelli octree veri yapısında ağacı dolaşmak ve bir düğüme ulaşmak doğrusal octree veri yapısına kıyasla daha kolay ve hızlıdır. Üstelik 3B ayrık ızgara modelinde olduğu gibi uzaydaki her bir vokseli kaydetmeye gerek olmadığı için hafızadan da tasarruf edilmiş olunmaktadır.

Şekil 4.5'te 8x8'lik örnek ikili resimden ters Z dolanım yönüne göre işaretçi-temelli quadtree oluşturulması ve ilgili dolanım yönüne göre etiketlenmiş quadtree alt uzayları gösterilmektedir.



Şekil 4.5. 8x8'lik örnek ikili resimden işaretçi-temelli quadtree oluşturulması (a) Örnek 8x8'lik ikili resim (b) ters Z dolanım yönüne göre etiketlenmiş quadtree alt uzayları (c) işaretçi-temelli quadtree gösterimi

Burada, sağ üstteki etiketlenmiş alt uzaylar, sol üstte yer alan 8x8'lik ikili resim kullanılarak ters Z dolanım yönüne göre oluşturulur. Alttaki ağaç gösterimi ise, 8x8'lik ikili resime ait işaretçi-temelli quadtree gösterimidir.

4.3.3. Hash-temelli octree

Octree veri yapısını hafızada saklamanın bir diğer yolu da ağacı işaretçiler veya doğrusal diziler yerine hash tablosu kullanarak oluşturmaktır [72]. Böylece, işaretçi-temelli octree veri yapısının karmaşık hiyerarşisi içerinde dolaşmak yerine düğümlere hash tablosundaki indeksine göre kolayca ulaşabilir. Bu amaçla, 2B/3B uzayın, alt uzay bölgelerini gösteren quadtree/octree düğümlerinin hafızada

saklanabilmesi için işaretçiler yerine düğümün uzaydaki konumunu gösteren anahtar değerleri kullanılabilir [70].

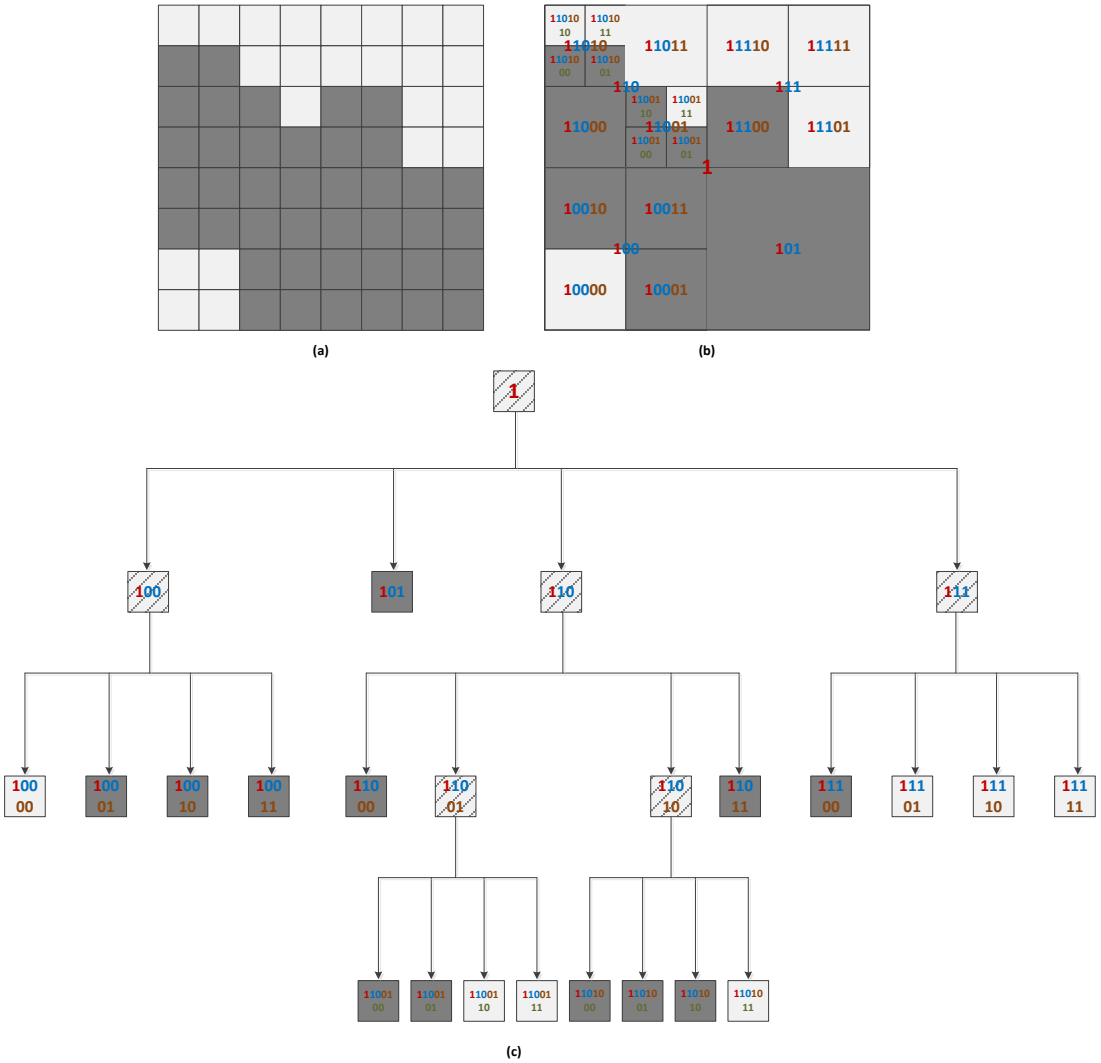
4.3.3.1. Anahtar hesaplama

Anahtar hesaplama işlemi, ilgili düğümün işaret ettiği yer kodlarını birleştirilmesi veya uzayın/alt uzayın ağacın ilgili seviyesine göre uzaydaki konumunun (x,y,(z)) kullanılması olmak üzere iki şekilde hesaplanabilir.

Quadtree/octree düğümünün anahtar değerini hesaplayan birinci yolda, ilgili düğümün işaret ettiği uzayın/alt uzayın yer kodları birleştirilir. Bir l seviyesindeki çocuk düğümün anahtarı, ebeveyn düğümünün anahtar değerinin sağına, bulunduğu alt uzayın 2/3 bitlik yer kodu eklenerek hesaplanır. Bu işlem, yaprak düğümlere ulaşıncaya kadar her seviyede anahtarın sağına 2/3 bitlik yer kodu eklenerek devam eder.

Ağaçtaki her düğümün tek bir anahtar değeri vardır ve bu değer diğer tüm düğümlerden farklıdır. Ancak, ters Z dolanım yönüne göre sol alt (arka) uzayın yer kodu “00(0)” olduğundan dolayı, bu uzaya/alt uzaya tekabül eden anahtar değeri her seviyede “0” olarak hesaplanır. Bu çalışmada, bu şekilde tekrarlayan “0” anahtar değerini engellemek amacıyla anahtar hesaplama işlemi, ağacın en tepesinde yer alan ve tüm uzayı gösteren kök düğüme “1” değeri atanarak başlar [72].

Şekil 4.6'da örnek 8x8'lik ikili resmin ters Z dolanım yönü kullanılarak uzay/alt uzaylarının ilgili yer kodlarından hesaplanan anahtar değerleri ve işaretçi-temelli quadtree veri yapısının ağaç gösterimi verilmektedir.



Şekil 4.6. 8x8'lik örnek ikili resmin ters Z dolanım yönü kullanılarak uzay/alt uzaylarının ilgili yer kodlarından anahtar değerlerinin hesaplanması (a) Örnek 8x8'lik ikili resim (b) ters Z dolanım yönüne göre ilgili yer kodlarından anahtar değerleri hesaplanmış quadtree uzayı ve tüm alt uzayları (c) işaretçi-temelli quadtree gösterimi

Buradaki ters Z olarak adlandırılan dolanım yönü, sol alt uç noktada yer alan pikselin resmin en küçük noktası (yani, (0,0)), sağ üstte uç noktada yer alan pikselin ise resmin en büyük noktası olduğu kabul edilerek belirlenmiştir. Dolayısıyla, pikselin/alt uzayın uzaydaki konumundan kendisine karşılık gelen anahtar değeri hesaplanabilir. Bu işlem, 3B uzayda da, derinlik (z) uzayının işleme dâhil edilmesi ile birlikte aynı şekilde gerçekleştirilir.

Anahtar hesaplamanın ikinci yolu ise uzayın/alt uzayın, ağacın seviyesine göre uzaydaki konumunu (x,y,(z)) kullanarak hesaplamaktır. Daha genel olarak, düğüm *n*'nin *k(n)* anahtarı ilgili düğümü yer koduna genişleme ve girişikleme yaparak ve en

baştaki bitinin soluna kök düğümün anahtarı olan "1" ekleyerek elde edilir. Girişikleme, iki veya daha fazla tamsayıyı kaydırarak genişletme işlemidir [73]. Genişletme ise bit formatındaki bir tamsayının her bir bitinin arasına bir dizi sıfır eklemektir [74]. Verilen bir *l* seviyesi için bir düğümün uzaydaki (x,y,z) konumuna göre anahtar değerinin hesaplanması sırasıyla Denklem 4.1, Denklem 4.2 ve Denklem 4.3'te verilmiştir.

$$\mathrm{DIL(x)} = \mathrm{x_l 00 x_{l-1} 00 .. x_1 00} x_0 \tag{4.1}$$

$$\mathrm{DIL(y)} = \mathrm{y_l 00 y_{l-1} 00 .. y_1 00 y_0} \tag{4.2}$$

$$\mathrm{DIL(z)} = \mathrm{z_l 00 z_{l-1} 00 .. z_1 00 z_0} \tag{4.3}$$

Burada, $\mathrm{x} = \mathrm{x_l x_{l-1} .. x_1 x_0}, \mathrm{y} = \mathrm{y_l y_{l-1} .. y_1 y_0}$ ve $\mathrm{z} = \mathrm{z_l z_{l-1} .. z_1 z_0}$, (x,y,z) konumundaki bir düğümün koordinatlarının bit formatıdır. DIL(x), DIL(y) ve DIL(z) ise, sırasıyla 1 bit yer kodu ve 2 sıfır olacak şekilde x, y ve z'nin homojen olmayan genişlemesidir. DIL(x), DIL(y) ve DIL(z)'nin eksenlerin bit değerlerinin z,y,x sıralamasına göre girişiklemesi Denklem 4.4 ve Denklem 4.5'te verilmiştir.

$$\mathrm{INT(x,y,z)} = \mathrm{DIL(z)} \ll 4 \mid \mathrm{DIL(y)} \ll 2 \mid \mathrm{DIL(x)} \tag{4.4}$$

$$\mathrm{INT(x,y,z)} = \mathrm{z_l y_l x_l z_{l-1} y_{l-1} x_{l-1}} \ldots \mathrm{z_1 y_1 x_1 z_0 y_0 x_0} \tag{4.5}$$

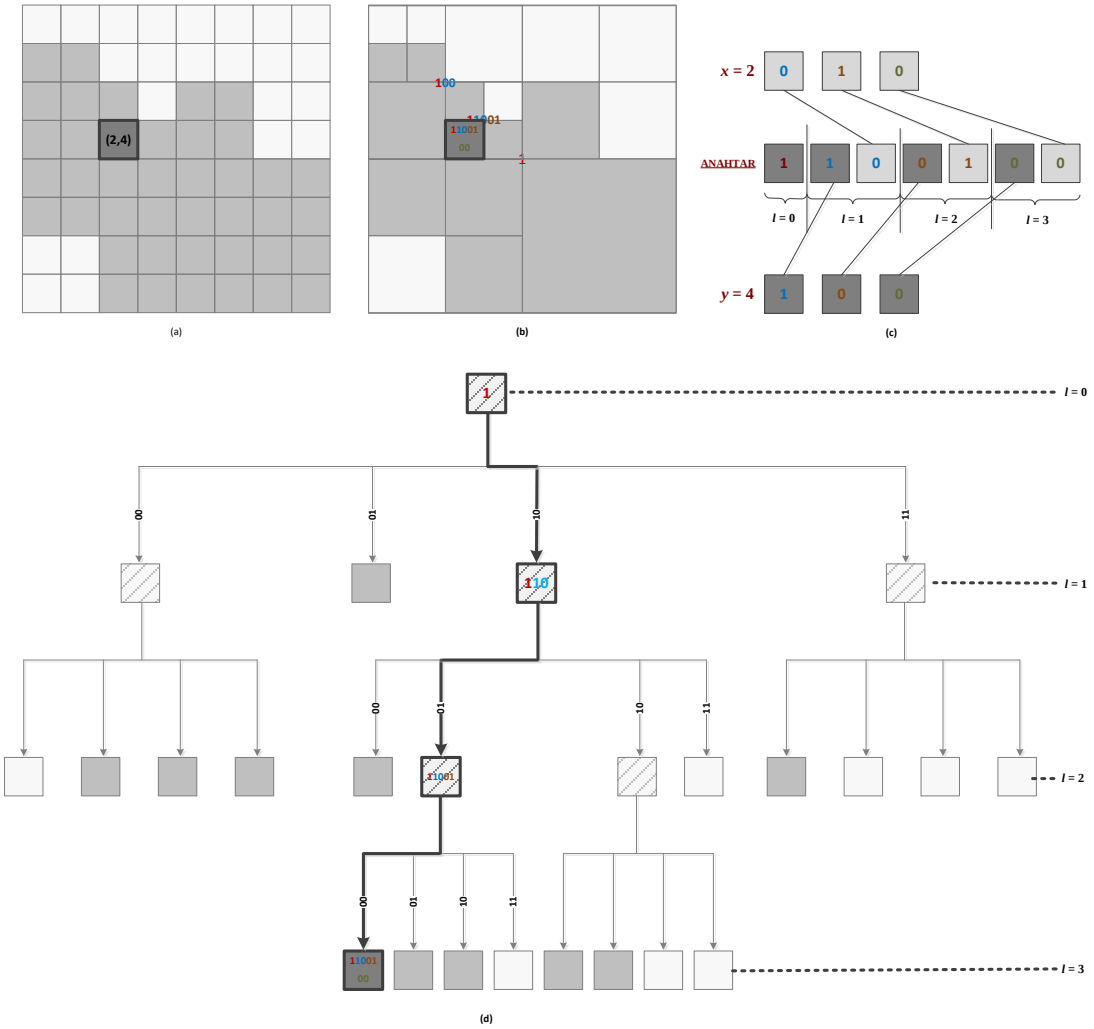
DIL(x), DIL(y) ve DIL(z)'nin eksenlerin bit değerlerinin z,y,x sıralamasına göre girişiklemesi olan INT(x,y,z)'den anahtar değerinin hesaplanması Denklem 4.6 ve Denklem 4.7'de verilmiştir.

$$\mathrm{KEY(x,y,z)} = 1 \,\&\, \mathrm{INT(x,y,z)} \tag{4.6}$$

$$\mathrm{KEY(x,y,z)} = 1\mathrm{z_l y_l x_l z_{l-1} y_{l-1} x_{l-1}} \ldots \mathrm{z_1 y_1 x_1 z_0 y_0 x_0} \tag{4.7}$$

Burada KEY(x,y,z), ilgili düğümün anahtar değeridir ve INT(x,y,z)'nin en soluna 1 bit eklenerek elde edilmiştir.

Şekil 4.7'de, 2B uzaydaki konumu (2,4) olan pikselden yer koduna ve pikselin uzaydaki konumuna bağlı olarak anahtar değerinin elde edilmesi gösterilmektedir.



Şekil 4.7. 8x8'lik örnek ikili resim üzerindeki bir pikselin ters Z dolanım yönü kullanılarak anahtar değerinin hesaplanması (a) Örnek 8x8'lik ikili resim ve anahtarı hesaplanacak piksel (b) ilgili pikselin ters Z dolanım yönüne göre yer kodlarından anahtar değerlerinin hesaplanması (c) ilgili pikselin uzaydaki konumuna göre anahtar değerinin hesaplanması (d) ilgili pikselin işaretçi-temelli quadtree gösterimindeki konumu

Burada, uzaydaki konumu (2,4) olan pikselin anahtar değeri uzaydaki konumu, genişleme ve girişikleme uygulanarak ya da piksele ulaşmak için uzaydan alt uzaylara doğru yer kodları kullanılarak hesaplandığında $(1100100)_2$ = 100 olarak bulunur.

3B uzaydaki bir voksele ulaşmak veya ilgili voksele komşu diğer voksellerin konumunu hesaplamak amacıyla işaretçi-temelli octree veri yapısında ilgili vokseli gösteren düğüme ulaşmak için ağaç üzerinde yukarıdan-aşağıya veya aşağıdan yukarıya dolaşmak gerekir. Bu da zaman-kritik uygulamalar için istenmeyen bir durumdur. Castro ve diğerleri tarafından, hem octree yapısının hafıza kazancından

faydalanmak hem de bu ağaç üzerinde geçen dolanım süresini azaltmak amacıyla octree veri yapısının ağaç yapısına sadık kalınarak düğümleri hash tabloları içine yerleştiren bir çalışma önerilmiştir [72]. Bu çalışmada, önerilen hash-temelli octree veri yapısı olarak adlandırılacak olan bu yeni veri yapısı belirlenen indekse göre düğümlerin sırasıyla (aşağıdan-yukarıya veya yukarıdan-aşağıya) hash tablosuna eklenmesi ilkesine dayanır.

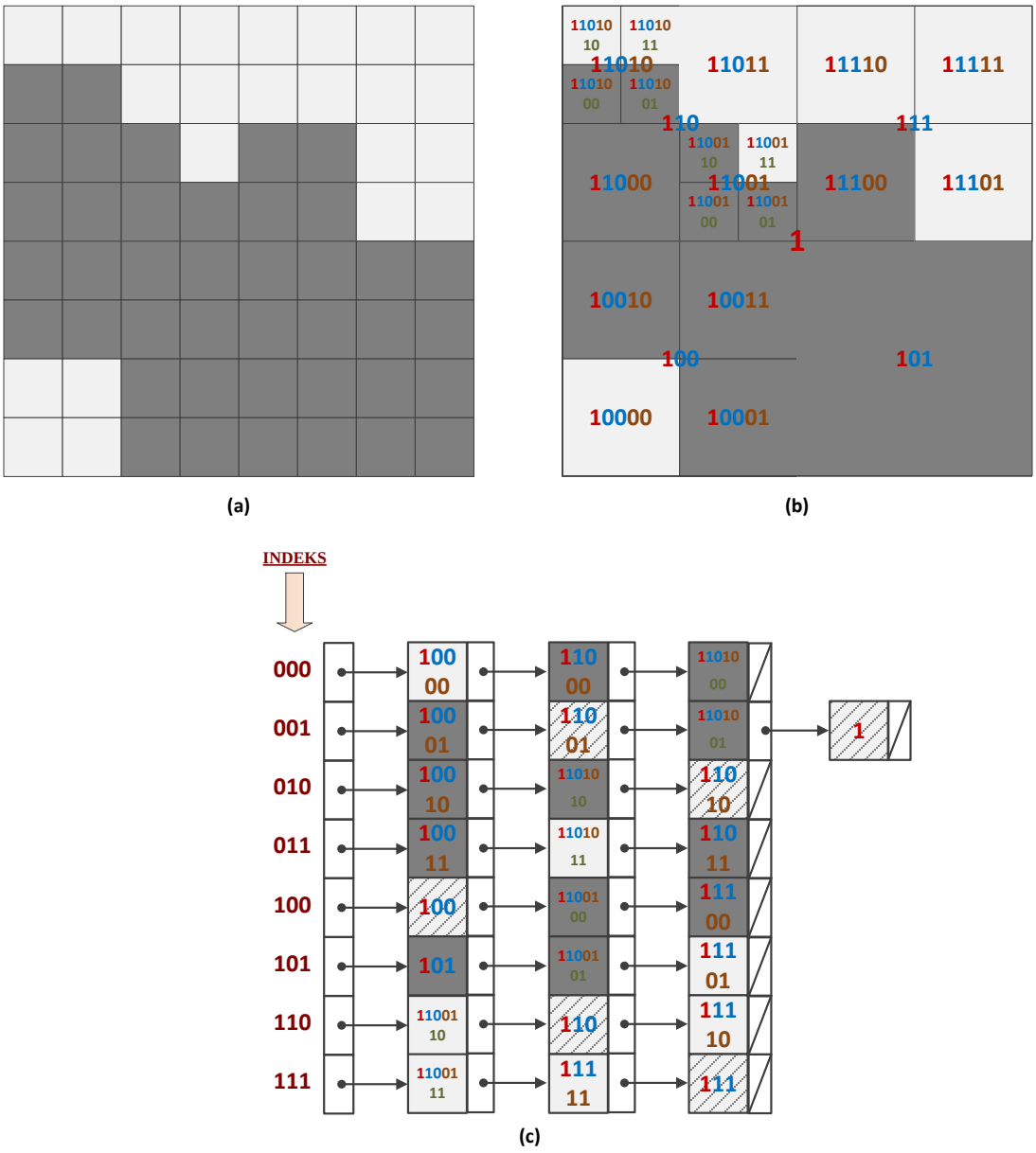
Basit bir liste ya da bağlı liste yapısında tasarlanan sabit boyutlu hash tablosuna veriler "anahtar" olarak belirlenen tekil değerlere bağlı olarak eklenir. Hash fonksiyonu olarak adlandırılan fonksiyon ile de verinin hash tablosunda hangi indekse atanacağı belirlenir.

Bu çalışmada kullanılan hash tablosunun boyutu, ağacın çözünürlüğüne bağlı olarak 2'nin üsleri (2^n, $n \in \{0,1,2,\ldots\}$) şeklinde belirlenmiştir. Ağaçtaki her düğüm, Denklem 4.8'de verilen ve aşağıdan-yukarıya octree yaklaşımındaki sıralamaya uygun olarak anahtar değerinin sağdan 2^n tane bitinin değerine bakan hash fonksiyonu kullanılarak hesaplanan bir indekse atanmaktadır.

$$\mathrm{hash}(x) = x \,\%\, 2^n;\ n \in \{0,1,2,\ldots\} \tag{4.8}$$

Burada *x*, octree düğümünün anahtar değeridir. Bu anahtarın başlangıçta belirlenen bir 2^n değerine göre modu hesaplanarak anahtarın işaret ettiği düğümün hash tablosunda hangi indekse atanacağı belirlenir ve düğüm ilgili indeksin sonuna eklenir.

Şekil 4.8'de örnek 8x8'lik ikili resmin ters Z dolanım yönü kullanılarak uzay/alt uzaylarının ilgili yer kodlarından hesaplanan anahtar değerlerine göre hash-temelli veri yapısına yerleştirilmesi gösterilmektedir.



Şekil 4.8. 8x8'lik örnek ikili resime ait hash-temelli octree veri yapısı oluşturulması (a) Örnek 8x8'lik ikili resim (b) ters Z dolanım yönüne göre ilgili yer kodlarından anahtar değerleri hesaplanmış quadtree uzayı ve tüm alt uzayları (c) hash-temelli quadtree gösterimi

Burada, hash tablosunun boyutu 8 (yani, 2^3) olarak belirlenmiştir ve ters Z dolanım yönüne göre anahtar değerleri hesaplanan her octree düğümü aşağıdan-yukarıya yaklaşımı ile anahtar değerinin son 3 bitinin değerine bakılarak ilgili indekse atanır. Örneğin, uzaydaki konumu (2,4) olan ve bu konuma göre anahtar değeri $(1100100)_2$ (yani, 100) olarak hesaplanan düğüm, 100%8=4 olduğundan dolayı hash tablosunda değeri 4 olan indeksin sonuna eklenir.

4.3.4. Optimize edilmiş hash-temelli octree

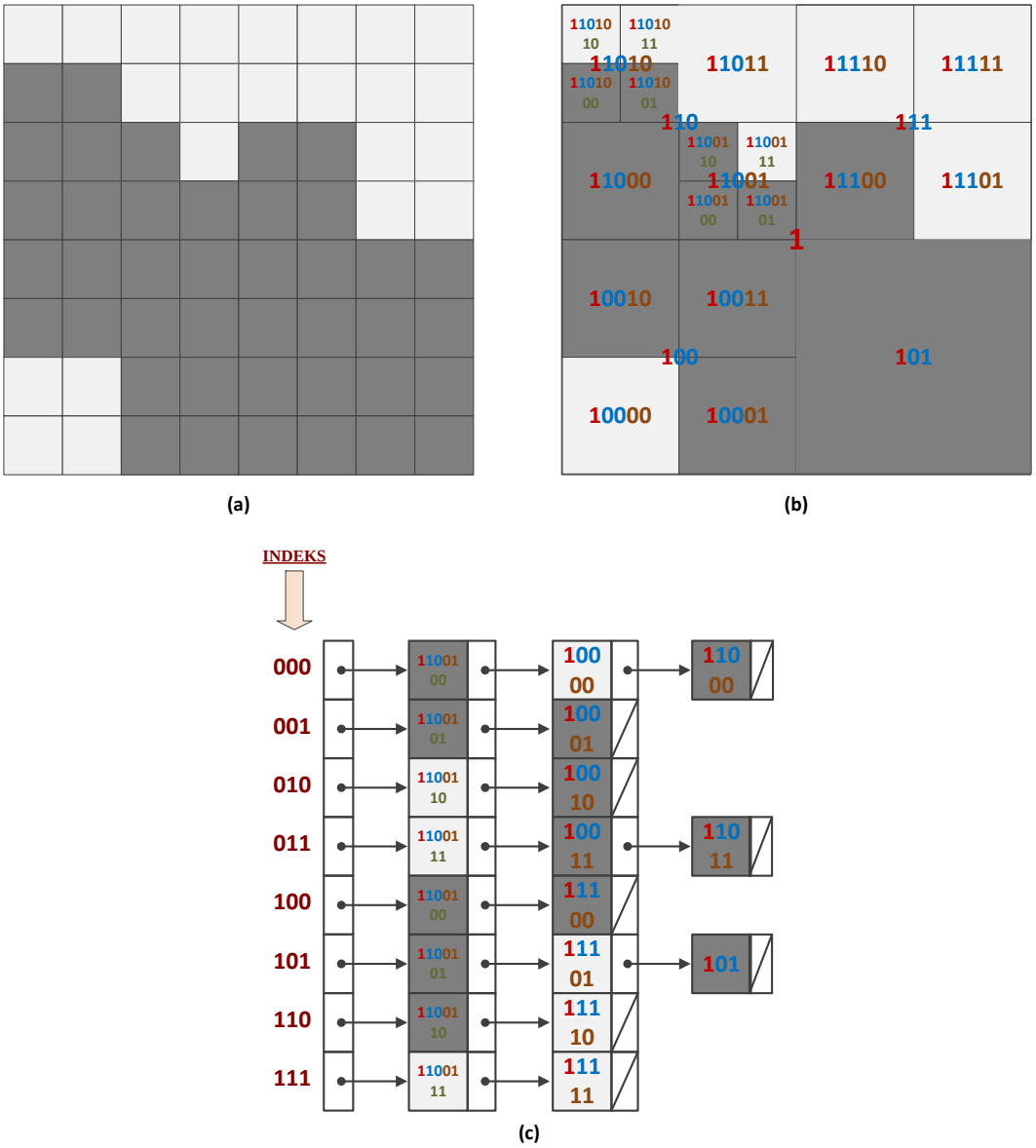
Octree veri yapısının sağladığı hafıza kazancı ile hash veri yapısının sunduğu veriye erişim kolaylığının birleştirilmesiyle elde edilen hash-temelli veri yapısında düğümler, ilgili anahtar değerlerine göre kaydedilirler.

Bu çalışmada, hash-temelli veri yapısının, hash tablosuna çocuk düğümleri olan ebeveyn düğümlerin eklenmemesi, yani sadece tamamen boş (“0”) veya tamamen dolu (“1”) olan yaprak düğümlerin eklenmesi yolu ile optimize edilmesi önerilmektedir. Bunun nedeni, çocuk düğümlerinin herhangi birisinin anahtar değerinden ebeveyn düğümünün anahtar değerine kolaylıkla erişilebileceğinden dolayı, ebeveyn düğümlerin tabloya eklenmesine gerek olmamasıdır. Bu sayede, hash tablosuna eklenecek olan düğüm sayısı azalacağından dolayı veri yapısını oluşturmak için gerekli toplam hafıza maliyeti de düşürülmüş olur.

Üç-boyutlu uzayda yer alan bir alt uzayın işaret ettiği bir düğümün çocuklarının olup olmadığını, yani işaret ettiği alt uzayın tamamen dolu (“1”) veya tamamen boş (“0”) olup olmadığını kontrol eden geçici bir yığıt yapısı kullanılmıştır. Düğümler hash tablosuna kaydedilirken oluşturulan bu geçici yığıt, ağaç tamamen dolaşıldıktan sonra hafızadan silinmektedir.

Ağaçta dolanım, yukarıdan-aşağıya veya aşağıdan-yukarıya yaklaşımlarından farklı olarak, ağaç dolanım yönlerine uygun şekilde aşağıdan-yukarıya her seviyenin tek tek dolaşılması, yani bir seviyenin dolaşılmasının tamamlanmasının ardından bir üst seviyeye geçilmesi yolu ile gerçekleştirilmektedir. Böylece, hash tablosundaki indekslerin başına voksellere karşılık gelen en alt seviyedeki düğümlerin (çocuk düğümler) yerleşmesi sağlanmıştır. Bu sayede, hash tablosunda bir voksele erişmek için geçen süre kısaltılması sağlanmıştır.

Şekil 4.9’da örnek 8x8'lik ikili resmin ters Z dolanım yönü kullanılarak uzay/alt uzaylarının ilgili yer kodlarından hesaplanan anahtar değerlerine göre optimize-edilmiş hash-temelli veri yapısına yerleştirilmesi gösterilmektedir.



Şekil 4.9. 8x8'lik örnek ikili resime ait optimize edilmiş hash-temelli octree veri yapısı oluşturulması (a) Örnek 8x8'lik ikili resim (b) ters Z dolanım yönüne göre ilgili yer kodlarından anahtar değerleri hesaplanmış quadtree uzayı ve tüm alt uzayları (c) optimize edilmiş hash-temelli quadtree gösterimi

Burada, hash tablosunun boyutu 8 (yani, 2^3) olarak belirlenmiştir ve ters Z dolanım yönüne göre anahtar değerleri hesaplanan her octree düğümü aşağıdan-yukarıya her seviyenin tek tek dolaşılması yaklaşımı ile anahtar değerinin son 3 bitinin değerine bakılarak ilgili indekse atanmaktadır. Bu yeni yaklaşım ile hash tablosundaki indekslerin başına, örnek 8x8'lik resimdeki piksellere karşılık gelen çocuk düğümlerin yerleştiği görülmektedir. Ağacı dolaşmak için kullanılan geçici yığıt

sayesinde de çocuğu olmayan düğümlerin belirlenerek tabloya eklenmediği de görülmektedir.

Hash-temelli octree yaklaşımında çocuk düğüme sahip düğümlerin çıkartılması ile oluşturulan bu yeni veri yapısı tezin bundan sonraki bölümlerinde optimize edilmiş hash-temelli octree veri yapısı olarak adlandırılacaktır. Bu yaklaşımın farklı çözünürlükler üzerindeki performansını test etmek için, ön-işleme zamanında hafıza maliyeti, hesaplama ve yüzeyin yeniden oluşturulma süreleri bakımından, gerçek-zamanda ise ilgili süreler bakımından işaretçi-temelli ve hash-temelli octree veri yapıları ile karşılaştırılmıştır. Elde edilen sonuçlara göre önerilen bu yeni yaklaşımın gerek hafıza maliyetindeki, gerekse hesaplama ve yüzeyin yeniden oluşturulması sürelerindeki düşüş bakımından sağladığı kazanç Bölüm 7'de verilmiştir.

BÖLÜM 5. SİMÜLASYON MODÜLÜ

Grafik ve haptic sahneleme modülleri, simülasyon modülü tarafından harekete tetiklenir. Simülasyon modülü, haptic cihazının miline bağlı olan sanal yontma aracı, sanal heykel nesnesinin yüzeyi ile çarpışır çarpışmaz çalışmaya başlamaktadır. "Çarpışma" olarak adlandırılan bu "etki" tespit edildiğinde, hem grafik hem de haptic sahneleme modüllerinde bu çarpışmaya karşılık bir "tepki" meydana gelir. Simülasyon modülünde, ilk olarak sanal araç ile model arasında meydana gelen çarpışmadan etkilenen bölgesel voksel alanı hesaplanır ve etkilenen voksellerin optimize-edilmiş hash-temelli octree veri yapısındaki değerleri güncellenir. Ardından, grafik sahneleme döngüsünde, güncellenmiş hash tablosu kullanılarak modelin eş yüzeyi bölgesel olarak yeniden oluşturulur. Daha sonra, simülasyon modülünde meydana gelen çarpışma sonucu bir geri-besleme sinyali hesaplanır ve bu sinyal Phantom cihazı ile insan-bilgisayar etkileşimi sağlayan bir kuvvet formunda kullanıcının eline iletilir.

Simülasyon modülünde, haptic cihazının miline bağlı olan sanal araç ile sanal heykel nesnesinin yüzeyinin çarpışmasından doğan etki "Çarpışma Tespiti", bu çarpışmaya karşılık meydana gelen tepki ise "Çarpışma Cevabı" olarak adlandırılır.

5.1. Çarpışma Tespiti

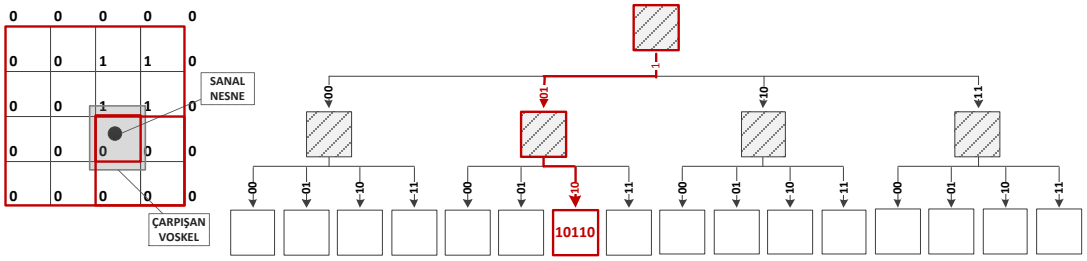
Çarpışma tespiti, optimize-edilmiş hash-temelli octree veri yapısında saklanan 3B voksel-temelli hacimsel sanal nesne ile haptic cihazının mili tarafından kontrol edilen sanal yontma aracının çarpışmasının test edilmesi anlamına gelmektedir. Bu amaçla, küre şeklinde tasarlanan sanal aracın kaplayan hacminin, yontulacak nesnenin hacimsel verisi içerisindeki vokseller ile çarpışıp çarpışmadığı sürekli olarak test

edilir. Eğer çarpışma var ise, çarpışma cevabı sırasıyla grafik ve haptic sahneleme döngülerine gönderilmektedir.

Bu çalışmada, çarpışma tespiti küre olarak tasarlanan sanal araç ile optimize-edilmiş hash-temelli octree veri yapısındaki vokseller arasında gerçekleşir. Dolayısıyla test, bir küre/küp kesişim testidir [75].

Çarpışma tespitinde, ilk önce haptic cihazının miline bağlı olan sanal aracın, modelin kaplayan hacmi içerisinde mi olduğu test edilir. Eğer, sanal araç, modelin kaplayan hacminin dışında ise "çarpışma yok" denir. Aksi halde, sanal aracın optimize-edilmiş hash-temelli octree veri yapısındaki hangi vokseller ile çarpıştığı test edilir ve ilgili vokseller ile komşu voksellerinin değerleri güncellenir.

Sanal aracın optimize-edilmiş hash-temelli octree veri yapısındaki hangi vokseller ile çarpıştığı ağacın kökünden yaprağına doğru sırasıyla hangi alt hacimin içerisinde yer aldığına bakılarak tespit edilir. Şekil 5.1'de sanal aracın merkezi ile aracın bulunduğu alt uzaydaki vokselin çarpışıp çarpışmadığının nasıl tespit edildiği 2B olarak gösterilmektedir.



Şekil 5.1. 4x4 çözünürlüklü örnek resimde sanal araç ile çarpışan pikselin tespit edilmesi

Burada, sanal modelin tüm kaplayan hacmi ağacın kök düğümü ile temsil edilirken, en uçtaki yaprak düğümlerin ise vokselleri temsil ettiği görülmektedir. Sanal aracın temsil ettiği noktanın hangi alt uzayın kaplayan hacmi içerisinde yer aldığı sırasıyla ağacın kökünden yaprağına doğru test edilerek tespit edilir. Ağacın çarpışmadan etkilenen pikselini göstermek için dolanım koyu renk ile gösterilmiştir ve ilgili pikselin ters Z dolanım yönüne göre hesaplanan anahtar değeri "10110"dır.

Eğer sanal aracın işaret ettiği vokselin optimize-edilmiş hash-temelli octree veri yapısındaki değeri "0"dan farklı ise "çarpışma var" denir. Aksi halde, "çarpışma yok" denir. Bu amaçla, çarpışma tespiti yapılan ve ters Z dolanım yönüne göre hesaplanan anahtar değerine sahip vokselin optimize-edilmiş hash-temelli octree veri yapısındaki hash tablosunda var olup olmadığına bakılır. Eğer voksel hash tablosunda bulunamadıysa vokseli içeren ebeveyn düğüm bulununcaya kadar ağaçta aşağıdan-yukarıya doğru seviye seviye çıkılır. Her seviyede, vokseli içeren ebeveyn düğümü hesaplamak için hash tablosunun boyutuna göre vokselin anahtar değerinin sağından Denklem 5.1'de hesaplanan kadar bit çıkartılır.

$$(l - d) * 3 \tag{5.1}$$

Burada, l, ağacın mevcut seviyesi, d ise derinliğidir. Özetle, Hash tablosunda aranan vokselin ebeveyni, ağacın hangi seviyesinde bulunuyorsa ve bu düğümün değeri "0"dan farklı ise "çarpışma var" denir.

Sanal araç ile model arasındaki çarpışmanın varlığı tespit edildikten sonra bu çarpışmaya grafik ve haptic sahneleme döngüleri tarafından verilecek çarpışma cevabı hesaplanmaktadır.

5.2. Çarpışma Cevabı

Haptic cihazının miline bağlı sanal araç ile hangi vokselin çarpıştığının simülasyon modülünde tespit edilmesinin ardından, oluşan çarpışma önce grafik, ardından da haptic sahneleme döngülerini tetikler.

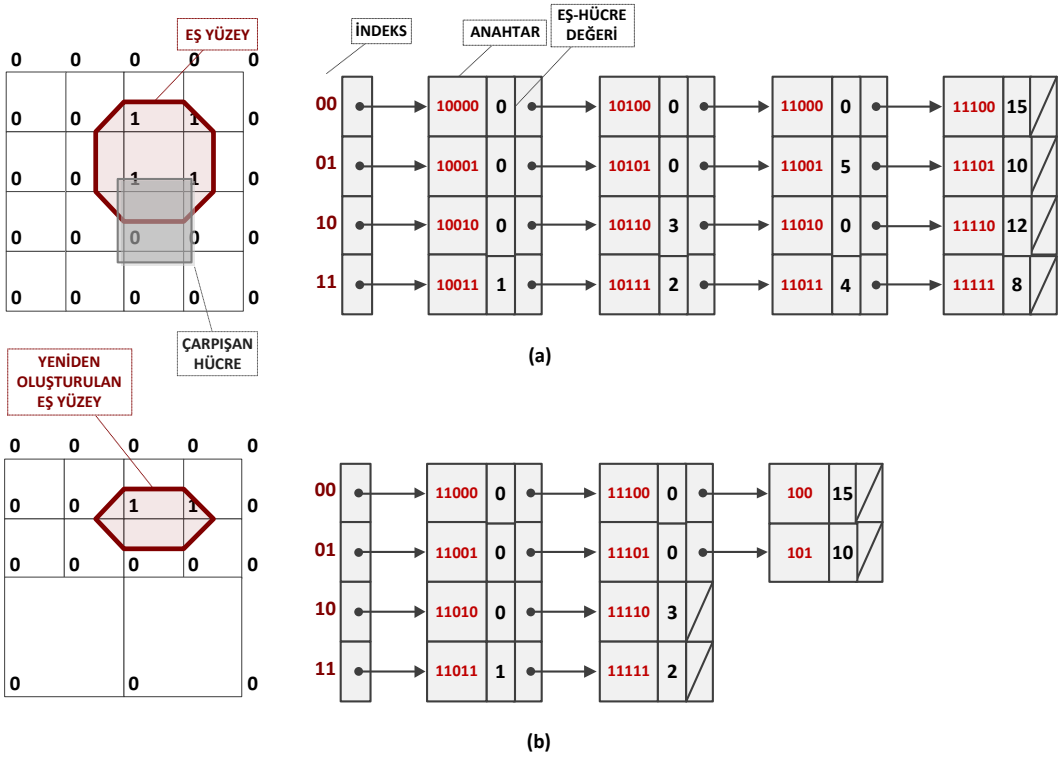
Burada, çarpışmadan etkilenen vokselin hash tablosunda kendisinin veya ebeveyninin bulunması gibi iki senaryo mevcuttur. Birinci senaryoda, yani çarpışmadan etkilenen vokselin hash tablosunda bulunması durumunda ilgili vokselin değeri "0" olarak güncellenir. Bu voksele komşu olan 26-komşu (kenar, köşe, yüzey)vokselinin değeri ise "0" olarak atanan köşe noktasının değerine bağlı olarak güncellenir.

İkinci senaryoda, yani çarpışmadan etkilenen vokselin hash tablosunda ebeveyninin bulunması durumunda ise çarpışmadan etkilenen vokselin ebeveyni, ağacın hangi seviyesinde bulundu ise ağaç bu seviyeden başlayarak en alt seviyeye, yani çocuk düğüme erişinceye kadar güncellenir. Bu güncelleme, ebeveyn düğümün hash tablosundan silinerek, ilgili düğümün tüm yaprak düğümlerinin eklenmesi şeklinde gerçekleştirilir. Bu yaprak düğümlere hash tablosundan silinen ebeveyn düğümün değeri atanır. Çarpışmadan etkilenen vokselin değeri ise "0" olarak güncellenir.

Çarpışmadan etkilenen vokselin 26-komşu (kenar, köşe, yüzey) vokseli de bu iki senaryoya uygun olarak hash tablosunda bulunur ve değerleri ilgili vokselin "0" olarak atanan köşe noktasının değerine bağlı olarak güncellenir.

Grafik sahneleme modülünün bir parçası olan grafik sahneleme döngüsünde, güncellenen hash tablosundaki voksellerin yeni değerlerine bağlı olarak hesaplanan bölgesel alanın eş-yüzeyi yeniden oluşturulur. Bu işlem, hash tablosundaki düğümlerin (voksellerin) çarpışmadan önceki değerlerine göre MC algoritması kullanılarak oluşturulan üçgenlerinin üçgen listesinden silinerek, güncellenen değerlere göre oluşturulacak yeni üçgenlerin listeye eklenmesi şeklinde gerçekleştirilir.

Şekil 5.2'de 2B voksel ızgarasının çarpışmadan önce ve sonraki eş-yüzey ve optimize-edilmiş hash-temelli quadtree gösterimleri verilmektedir.



Şekil 5.2. 4x4'lük 2B voksel ızgarasının (a) çarpışmadan önce ve (b) çarpışmadan sonraki eş-yüzey ve optimize-edilmiş hash-temelli quadtree gösterimi

Burada, sanal araç ile çarpışan pikselin değerinin “0” olarak güncellendiği ve bu değişikliğin çarpışan pikselin etrafındaki 5 komşu pikselin (sağ üst, üst, sol üst, sol, sol alt, alt, sağ alt) değerlerini de etkilediği görülmektedir. Dolayısıyla, bu pikselin hash tablosundaki değerinin güncellenmesinin ardından 5 komşu pikselinin de değerleri sırasıyla güncellenir.

Haptic cihazının miline bağlı araç ile sanal modelin çarpışması sonucu meydana gelen geri-besleme kuvvetinin hesaplanması aynı zamanda haptic sahneleme modülünün de bir parçasıdır ve bu kuvvetin nasıl hesaplandığı Bölüm 6’da detaylı olarak açıklanmıştır.

BÖLÜM 6. HAPTIC SAHNELEME MODÜLÜ

Dokunma hissi, son yıllarda insan-bilgisayar etkileşimi (HCI) sistemlerinin yaygınlaşmasından dolayı önemli bir araştırma konusu haline gelmiştir. Kuvvet veya dokunsal geri besleme yolu ile dokunma hissini bilgisayar uygulamaları ile birleştiren bilim dalı "Haptics" olarak adlandırılır. Haptic cihazı olarak adlandırılan özel giriş/çıkış cihazları, haptic desteği eklenmiş özel uygulamaları ile birlikte kullanılarak, kullanıcının sanal 3B nesneleri hissedebilmeleri veya değiştirebilmeleri sağlanmaktadır. "Haptic Sahneleme" ise, ilgili cihazlar kullanılarak sanal 3B nesnelere dokunulması durumunda, bu nesnelerin kullanıcı tarafından hissedilebilmesi olarak tanımlanabilir [76].

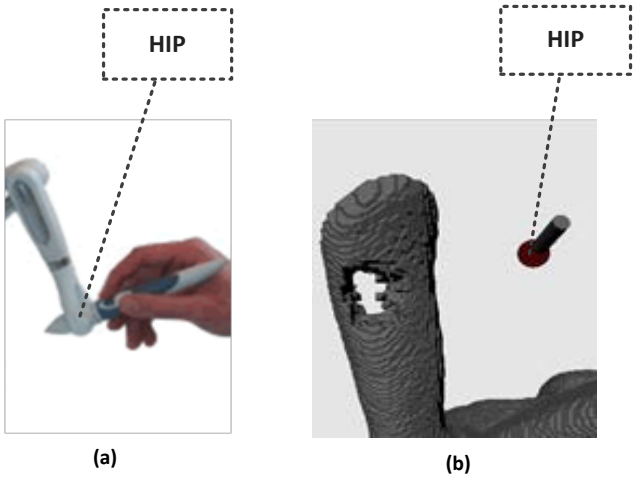
Bu çalışmada Phantom Omni haptic cihazı ile donatılmış önerilen sanal heykeltraşlık sistemindeki haptic sahneleme modülü, yontulacak nesne ile haptic cihazının mili tarafından kontrol edilen sanal aracın çarpışması sonucu meydana gelen geri-besleme sinyalinin hesaplanması ve bu etkileşimli kuvvetin haptic cihazı yolu ile kullanıcının eline iletilmesi ile ilgilenir.

Bu bölümde, haptic cihazının miline bağlı olan sanal aracın nasıl kontrol edildiği ve haptic sahneleme modülünü oluşturan haptic sahneleme döngüsü içerisindeki geri-besleme kuvvetinin nasıl hesaplandığı detaylı olarak açıklanmaktadır.

Haptic sahneleme modülü, 1 kHz'lik bir frekansa sahip sürekli bir döngüde çalışır. Haptic sahneleme döngüsü olarak adlandırılan bu döngünün her çevriminde sana aracın son konumu hesaplanır ve simülasyon modülünden gelen çarpışma bilgisine bağlı olarak geri-besleme kuvvetinin oluşturulması sağlanır.

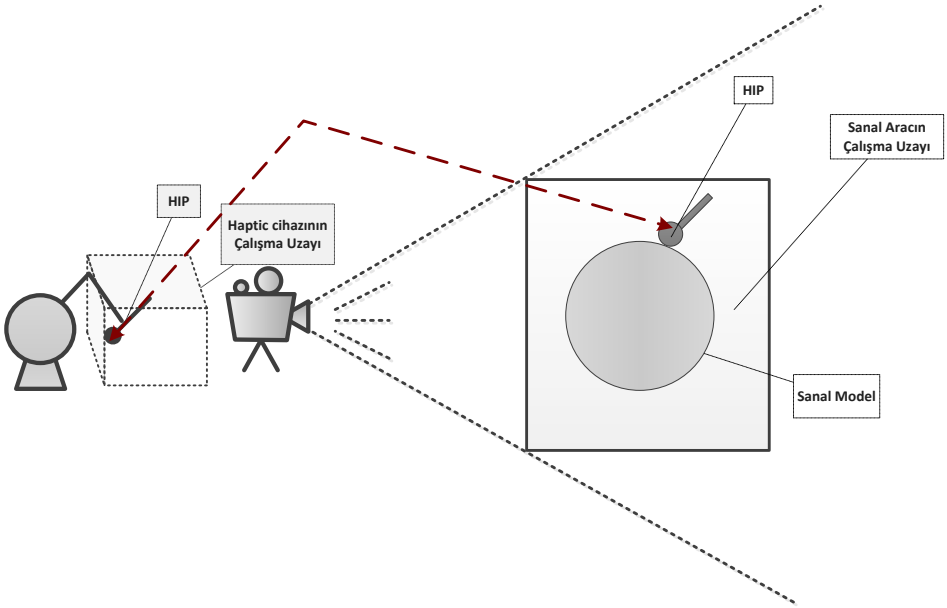
6.1. Sanal Araç

Bu çalışmada, haptic cihazının miline bağlı olan sanal araç bir küre olarak tasarlanmıştır ve kürenin merkezi, sanal dünya uzayında Phantom cihazının ikinci bağlantı noktasının cihaz uzayındaki hareketine bağlı olarak hareket eder [76]. Şekil 6.1'de haptic etkileşim noktası (HIP) olarak da adlandırılan bu noktanın gerçek ve sanal dünyadaki karşılıkları gösterilmektedir



Şekil 6.1. Haptic Etkileşim Noktası (a) Gerçek dünya (b) Sanal dünya

Şekil 6.2'de haptic cihazının çalışma uzayı ile 3B dünya uzayının haptic etkileşim noktalarına göre eşleştirilmesi gösterilmektedir.



Şekil 6.2. Sanal aracın çalışma uzayı ile 3B uzayın dünya uzayının eşleştirilmesi

6.2. Geri-Besleme Kuvvetinin Hesaplanması

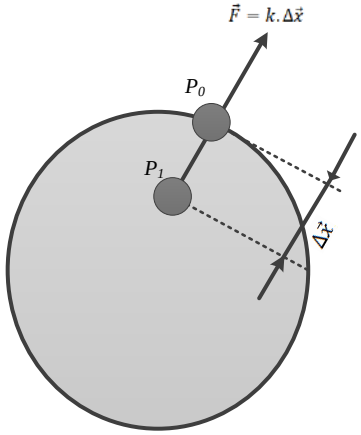
Bir haptic cihazı tarafından sanal nesnelere dokunulması yolu ile oluşturulan kuvvetlerin sahnelenmesi, yani haptic cihazı yolu ile kullanıcıya iletilmesi haptic sahneleme olarak adlandırılır. Kullanıcıya iletilecek olan geri-besleme kuvveti dokunulan sanal nesnenin sertlik, sürtünme, vs. gibi özelliklerine bağlı olarak hesaplanır. Bu çalışmada fiziksel özellikler sanal nesnenin her yerinde aynı değerlere sahip olacak şekilde belirlenmiştir. Aynı zamanda, geri-besleme kuvvetinin haptic cihazının mili tarafından kontrol edilen haptic etkileşim noktasına bağlı olarak hesaplanan nokta-temelli haptic sahneleme tekniği kullanılmıştır [77]. Nokta-temelli haptic sahneleme, en çok kullanılan yay kuvveti hesaplama yöntemi Denklem 6.1'deki gibi kullanılarak gerçekleştirilmiştir.

$$\vec{F} = k.\Delta\vec{x} \tag{6.1}$$

Burada, k, sanal modelin sertlik katsayısı, $\Delta\vec{x}$, haptic etkileşim noktasının toplam yer değiştirme vektörü ve $\vec{F}$, hesaplanan geri-besleme kuvvetidir. Toplam yer değiştirme vektörü Denklem 6.2'de gösterilmektedir.

$$\Delta\vec{x} = P_0 - P_1 \tag{6.2}$$

Burada P_0, haptic milinin yontulacak nesnenin yüzeyine dokunduğu nokta iken P_1, haptic milinin güncel konumu ve yani haptic etkileşim noktasıdır. Şekil 6.3'de geri-besleme kuvvetinin, sertlik katsayısı ve toplam yer değiştirme vektörüne göre hesaplanması gösterilmektedir.



Şekil 6.3. Geri-besleme kuvvetinin hesaplanması

Şekil 6.3'de, haptic cihazının çalışma uzayı ile 3B dünya uzayının eşleştirilmesi gösterilmektedir.

İlerleyen bölümde, optimize edilmiş hash-temelli octree veri yapısının kullanılmasını öneren haptic-temelli sanal heykeltıraşlık uygulamasının değerlendirmesi ve gelecek çalışması önerileri hakkında bilgi verilmektedir.

BÖLÜM 7. SONUÇLAR VE ÖNERİLER

Bu tezde, haptic cihazı ile insan-bilgisayar etkileşimi sağlanan bir sanal heykeltraşlık sistemi geliştirilmiştir. Önerilen voksel-temelli sanal heykeltraşlık sisteme hafıza optimizasyonu ve gerçek-zaman performansı alanlarında katkılar yapılmıştır. Sanal çalışma modelinin voksel-temelli hacimsel verisini saklamak için ayrık vokseller kümesi yerine hafıza maliyetini düşürmek amacıyla uzayın yinelemeli olarak alt uzaylara bölünmesi sonucu elde edilen octree veri yapısı kullanılmıştır. Önerilen çalışmadaki temel katkı, 3B octree-temelli hacimsel voksel veri kümesini saklamak için geleneksel octree veri yapısındaki işaretçi temelli octree yaklaşımı yerine optimize edilmiş hash-temelli bir octree veri yapısı kullanılmasıdır.

Bu çalışmada sonuçlar 4 GB hafızaya sahip Intel Core 2 Quad 2.5 GHz ana karta, 1 GB hafızaya sahip bir ATI Radeon HD 5850 grafik kartından oluşan ve 64 bit Windows 7 işletim sistemi yüklü bir bilgisayar kullanılarak elde edilmiştir.

Önerilen haptic-temelli sanal heykeltıraşlık sisteminin simülasyon yazılımı C++ platformunda yazılırken, sanal sahne ve tüm elemanlarını içeren grafik sahneleme modülü OpenGL, haptic sahneleme modülü ise OpenHaptics Toolkit kullanılarak yazılmıştır.

Optimize-edilmiş hash-temelli octree veri yapısının farklı çözünürlükler üzerindeki performansını test etmek için, önerilen optimize edilmiş hash-temelli octree yaklaşımı, ön-işleme zamanında hafıza maliyeti, hesaplama ve yüzeyin yeniden oluşturulma süreleri bakımından, gerçek-zamanda ise ilgili süreler bakımından işaretçi-temelli ve hash-temelli octree veri yapıları ile karşılaştırılmıştır.

Tablo 7.1'de 3B hacimsel voksel veri kümesinden işaretçi-temelli, hash-temelli ve optimize edilmiş hash-temelli octree oluşturma süreleri ve ilgili veri yapılarını oluşturmak için gerekli düğüm sayıları gösterilmektedir.

Tablo 7.1. İşaretçi-temelli hash-temelli ve optimize edilmiş hash-temelli octree veri yapıları için gerekli düğüm sayıları ve ilgili oluşturma süreleri

Model Adı	Üçgen Sayısı	Çözünürlük	Düğüm Sayısı		Oluşturma Süresi (sn)		
			İşaretçi & hash-temelli	Optimize edilmiş hash-temelli	İşaretçi-temelli	Hash-temelli	Optimize edilmiş hash-temelli
Küre	4800	64x64x64	50456	44150	0,099	0,059	0,034
		128x128x128	204560	179040	0,848	0,414	0,138
		256x256x256	820552	718187	6,724	2,760	0,723
		512x512x512	3281168	2871821	38,123	23,320	4,871
Küp	12	64x64x64	59848	52368	0,096	0,070	0,0354
		128x128x128	250376	219080	0,827	0,419	0,146
		256x256x256	1024584	896512	6,737	3,022	0,815
		512x512x512	4145800	3627576	39,895	17,233	5,404
Silindir	252	64x64x64	34736	30395	0,099	0,051	0,028
		128x128x128	142680	124846	0,810	0,341	0,109
		256x256x256	579744	507494	7,572	2,648	0,626
		512x512x512	2334264	2042699	41,232	19,427	4,488
Halka	512	64x64x64	33696	29492	0,104	0,052	0,022
		128x128x128	134272	117510	0,814	0,333	0,107
		256x256x256	547360	479018	6,801	2,636	0,623
		512x512x512	2187464	1911792	40,254	19,384	4,745
Tavşan	69664	64x64x64	37776	33055	0,192	0,124	0,118
		128x128x128	152784	133687	1,050	0,487	0,470
		256x256x256	613296	536635	7,517	1,962	1,962
		512x512x512	2446706	2141441	41,482	7,724	7,582
Ejderha	74998	64x64x64	31208	27308	0,205	0,165	0,107
		128x128x128	128416	112365	1,093	0,987	0,446
		256x256x256	514040	449786	6,919	1,635	1,710
		512x512x512	2055064	1798511	40,122	6,823	6,964

Veri yapılarını oluşturmak için belirtilen düğüm sayıları bize gerekli hafıza maliyetleri hakkında bilgi verir. Burada, hash-temelli octree veri yapısı ile işaretçi-temelli octree veri yapılarını oluşturmak için gerekli düğüm sayılarının aynı olduğu görülmektedir. Dolayısıyla, bu iki veri yapısının hafıza gereksinimleri de aynıdır denir. Fakat önerilen optimize edilmiş hash-temelli octree veri yapısını oluşturmak için gerekli düğüm sayısı diğer iki veri yapısındaki gerekli düğüm sayısından daha

azdır. Bunun nedeni, önerilen yeni hash-temelli octree yaklaşımında sadece yaprak düğümlerin, yani komple dolu veya boş olan düğümlerin tabloya eklenmesi, çocuk düğüme sahip ebeveyn düğümlerin ise eklenmemesidir. Önerilen octree yaklaşımı ile hafıza maliyetindeki hesaplanan düşüşün farklı çözünürlüklerdeki ortalama değeri %14-15 civarındadır. Üstelik önerilen optimize edilmiş hash-temelli octree veri yapısının oluşturulma süreleri diğer iki yaklaşımdan (işaretçi-temelli ve hash-temelli) daha kısadır.

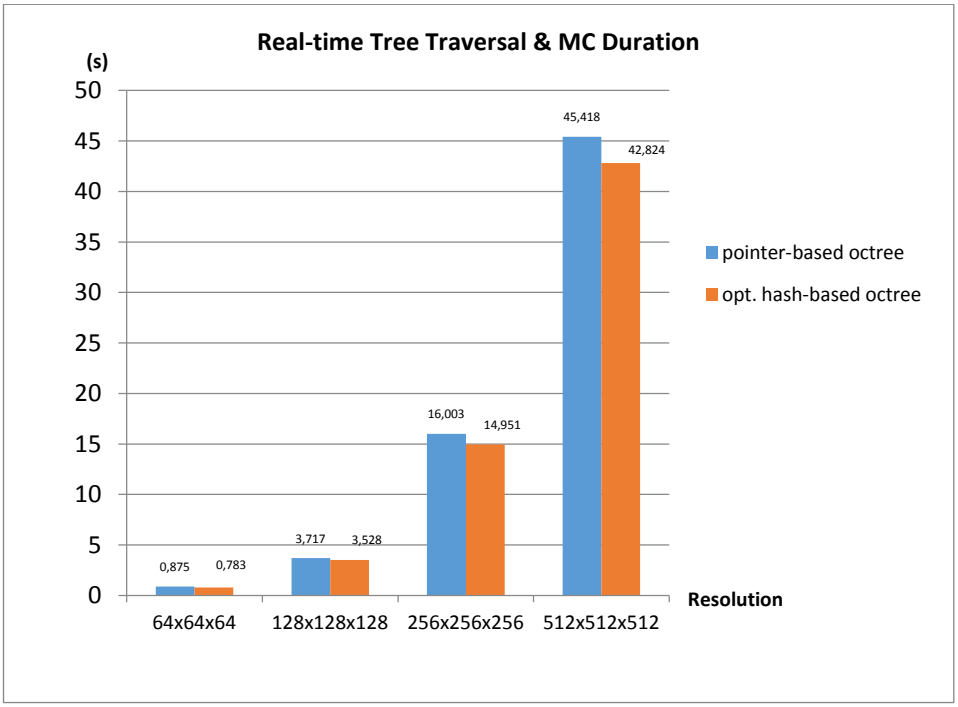
Tablo 7.2'de vokselleştirilerek önerilen optimize-edilmiş hash-temelli ve işaretçi-temelli octree veri yapıları içerisinde saklanan 3B modelin eş-yüzeyinin ön-işleme zamanında oluşturulması için geçen sürelerin farklı çözünürlüklerde karşılaştırılması gösterilmektedir.

Tablo 7.2. İşaretçi-temelli ve optimize edilmiş hash-temelli octree veri yapıları için Marching Cubes uygulanmasının farklı çözünürlüklerdeki ön-işleme boyunca geçen süreleri

Model Adı	Üçgen Sayısı	Çözünürlük	Ön-işleme MC Süreleri (sn)	
			İşaretçi-temelli	Optimize edilmiş hash-temelli
Küre	4800	64x64x64	0,173	0,169
		128x128x128	1,051	0,593
		256x256x256	7,146	2,355
		512x512x512	24,604	9,501
Küp	12	64x64x64	0,189	0,173
		128x128x128	1,154	0,705
		256x256x256	7,886	2,763
		512x512x512	23,105	11,475
Silindir	252	64x64x64	0,155	0,103
		128x128x128	0,891	0,406
		256x256x256	7,208	1,652
		512x512x512	14,413	6,601
Halka	512	64x64x64	0,132	0,122
		128x128x128	0,824	0,393
		256x256x256	5,835	1,567
		512x512x512	13,714	7,139
Tavşan	69664	64x64x64	0,146	0,104
		128x128x128	0,784	0,405
		256x256x256	8,008	1,481
		512x512x512	15,243	7,533
Ejderha	74998	64x64x64	0,174	0,128
		128x128x128	1,103	0,459
		256x256x256	7,451	1,674
		512x512x512	14,413	7,851

Burada, eş-yüzey oluşturma sürelerinin optimize-edilmiş hash-temelli octree veri yapısı kullanılarak saklanan 3B model verisinden işaretçi-temelli octree veri yapısı kullanılarak oluşturulmasına kıyasla daha kısa sürdüğü görülmektedir. Ayrıca, bu iki veri yapısının kullanılması ile ön-işleme zamanında eş-yüzey oluşturulması süresince geçen süreler arasındaki farkın daha da arttığı görülmektedir.

Şekil 7.1'deki grafikte, işaretçi-temelli ve optimize-edilmiş hash-temelli octree veri yapıları kullanılarak saklanan 3B tavşan modeli üzerinde aynı deliğin oluşturulması sırasında geçen toplam gerçek-zaman dolanımı ve eş-yüzeyin bölgesel olarak yeniden oluşturulma süreleri gösterilmektedir.

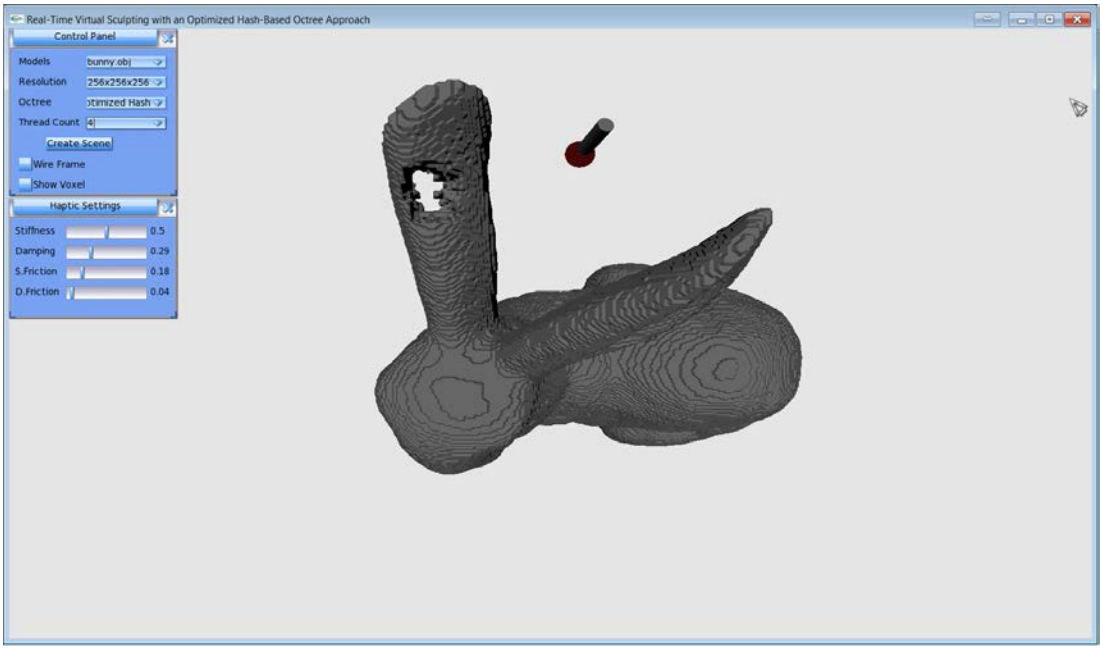


Şekil 7.1. İşaretçi-temelli ve optimize edilmiş hash-temelli octree veri yapıları kullanılarak saklanan 3B tavşan modeli üzerinde aynı deliğin oluşturulması sırasında geçen toplam gerçek-zaman dolanımı ve eş-yüzeyin bölgesel olarak yeniden oluşturulma süreleri

Burada, önerilen optimize edilmiş hash-temelli ve işaretçi-temelli octree veri yapıları kullanılarak saklanan sanal modelin üzerinde yapılan yontma işlemi sırasında geçen toplam gerçek-zaman dolanımı ve eş-yüzeyin bölgesel olarak yeniden oluşturulma sürelerinin çözünürlük arttıkça arttığı görülmektedir. Sanal model verisinin önerilen optimize-edilmiş hash-temelli octree yaklaşımı kullanılarak saklanması ile işaretçi-temelli octree veri yapısı kullanılarak saklanması kıyaslandığında ise önerilen

yaklaşımın gerçek-zaman dolanımı ve eş-yüzeyin bölgesel olarak yeniden oluşturulmasını daha kısa sürelerde tamamladığı görülmektedir.

Şekil 7.2'de önerilen haptic-temelli sanal heykeltraşlık sistemine ait yazılım ara yüzünün ekran çıktısı gösterilmektedir.



Şekil 7.2. Önerilen haptic-temelli sanal heykeltıraşlık sisteminin arayüzü

Yukarıdaki ekranda, 3B tavşan modelinin üzerinde yontma işlemleri yapılması sonucu yeniden oluşturulan eş-yüzeyin son hali gösterilmektedir.

Bu çalışmada önerilen haptic-temelli sanal heykeltraşlık sistemi sanat, cerrahi operasyonlar ve diş cerrahisi gibi yontma işlemi gerçekleştirilen çalışmaların eğitim amaçlı alt yapısını sağlamaktadır. Gerçekleştirilen sistemde hafıza optimizasyonu sağlamak ve gerçek-zaman performansını artırmak amacıyla kullanılan optimize edilmiş hash-temelli octree veri yapısının ilerleyen çalışmalarda yapay zekâ ve keşfe dayalı arama teknikleri kullanılarak geliştirilmesi hedeflenmektedir.

KAYNAKLAR

[1] ZHANG, W., Virtual Prototyping with Surface Reconstruction and Freeform Geometric Modeling Using Levet-set method. PhD, Missouri University of Science and Technology, 2008.

[2] EL SADDIK, A., OROZCO, M., EID, M., CHA, J., Haptics Technologies. Springer Series on Touch and Haptic Systems; Springer, pp. 3, 2011.

[3] GALYEAN, A., HUGHES, JF., Sculpting: An Interactive Volumetric Modeling Technique. Computer Graphics, 24:4:268-274, 1991.

[4] WANG, S., KAUFMAN, A.E., VolumeSculpting. In Symposium on Interactive Graphics; ACM SIGGRAPH, 1995.

[5] WILLIAMS, J., O'NEILL, G.T., LEE, W.S., Interactive 3D Haptic Carving using Combined Voxels and Mesh. In HAVE 2008 – IEEE International Workshop on Haptic Audio Visual Environments and their Applications; pp. 1008-113, 2008.

[6] O'NEILL, G.T., LEE, W.S., WILLIAMS, J., Haptic-Based 3D Carving Simulator, Advances in Haptics, pp.299-314, 2010.

[7] BARENTZEN, A., Octree-based Volume Sculpting. In IEEE Visualization '98, Late Breaking Hot Topics Preoceedings; IEEE Computer Society Press, pp. 9-12, 1998.

[8] FERLEY, E., CANI, M.P., GASCUEL, J.D., Practical Volumetric Sculpting. The Visual Computer; 16:8:469-480, 2000.

[9] FERLEY, E., CANI, M.P., GASCUEL, J.D., Resolution Adaptive Volume Sculpting. Journal of Graphical Models; 63:6:459-478, 2001.

[10] PERNG, K.L., WANG, W.T., FLANAGAN, M., OUHYOUNG, M., A real-time 3d virtual sculpting tool based on modified marching cubes. In Proceedings of International Conference on Artificial Reality and Teleexistence, pp. 64–72, 2001.

[11] HO, C.C., TU, C.H., OUHYOUNG, M., Detail Sculpting using Cubical Marching Squares. ICAT'0, pp.10-15, 2005.

[12] RAFFIN, R., GESQUIERE, G., REMY, E., THON, S., VirSculpt: A Virtual Sculpting Environment. In International Conference Graphicon, pp. 184–187, 2004.

[13] HEURTEBISE, X., THON, S., Discrete Tools for Virtual Sculpture. In GRAPP'06, pp. 415–422, 2006.

[14] HEURTEBISE, X., THON, S., Multiresolution representation and deformation of very large volume datasets based on Haar wavelets. In 3rd International Conference on Geometric Modeling and Imaging, pp. 34–40, 2008.

[15] HEURTEBISE, X., THON, S., A Repartition Structure for Collision Detection and Deformation of Discrete Objects Based on 3D Wavelets. International Journal of Computer Information Systems and Industrial Management Applications; 3:568–577, 2011.

[16] CHEN, H., SUN, H., Real-Time Haptic Sculpting in Virtual Volume Space. VRST'02 - Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp.81-88, 2002.

[17] AGUS, M., GIACHETTI, A., GOBBETTI, E., ZENETTI, G., Real–time haptic and visual simulation of bone dissection. In IEEE VR 2002, pp. 250-264, 2002.

[18] AGUS, M., GIACHETTI, A., GOBBETTI, E., ZENETTI, G., Adaptive techniques for real–time haptic and visual simulation of bone dissection. In IEEE Virtual Reality Conference; IEEE Computer Society Press, pp.102-109, 2003.

[19] DURIEZ, C., SYLLEBRANQUE, C., Siz Degree-of Freedom Haptic Rendering for Dental Implanatory Simulation, ISBMS 2010, pp.139-149, 2010.

[20] ERIKSSON, M.G., FLEMMER, H., WIKANDER, J., A Haptic and Virtual Reality Skull Bone Surgery Simulator, In World Haptics 2005 Conference, Italy, 2005.

[21] ERIKSSON, M.G., DIXON, M., WIKANDER, J., A Haptic VR Milling Surgery Simulator Using High-Resolution CT Data, In The 14th MMVR Conference in Los Angeles, USA, 2006.

[22] ERIKSSON, M.G., WIKANDER, J., A Face Validated Six Degrees-of-Freedom Haptic Bone Milling Algorithm, IEEE Transactions on Haptics, 2012.

[23] BARBIC, J., JAMES, D.L., Six-DoF Haptic Rendering of Contact between Geometrically Complex Reduced Deformable Models, IEEE Transactions on Haptics 1:1, pp. 39-52, 2008.

[24] LORENSEN, W.E., CLINE, H.E., Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics, 21(4): 163-169, 1987.

[25] MÖLLER, A.T., Fast 3D Triangle/Box Overlap Testing. Journal of Graphics Tools, 6(1):29-33, 2001.

[26] MÖLLER AT., HAINES, E., Real-Time Rendering. AK Peters Ltd., 2002.

[27] FENG, L. SOON, S.H., An Effective 3D Seed Fill Algorithm. Computers & Graphics, 22(5):641-644, 1998.

[28] ÖZ, C., ÇİT, G., AYAR, K., Multithreaded Voxelization Method for Virtual Sculpting. European Conference of Technology and Society, 2013.

[29] SERTTAS, S., AYAR, K., CIT, G., OZ, C., Multi-Threaded Application for Marching Cubes Algorithm. ISITES 2014 - 2nd International Symposium on Innovative Technologies in Engineering and Science, 2014.

[30] BERNARDINI, F., MITTLEMAN, J., RUSHMEIER, H., SILVA, C., TAUBIN, G., The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics; 5:4:349–359, 1999.

[31] KOBBELT, L.P., BOTSCH, M., SCHWANECKE, U., SEIDEL, H.P., Feature Sensitive Surface Extraction fromVolume Data. Proceedings of SIGGRAPH 2001, pp. 57-66, 2001.

[32] HO, C.C., WU, F.C., CHEN, B.Y., CHUANG, Y., OUHYOUNG, M., Cubical marching squares: Adaptive feature preserving surface extraction from volume data. Computer Graphics Forum; 24:2, 2005.

[33] ÇİT, G., AYAR, K., SERTTAŞ, S., ÖZ, C., A Real-Time Virtual Sculpting Application with a Haptic Device. Turkic World Mathematical Society Journal of Applied and Engineering Mathematics; 3:2:105–112, 2013.

[34] KAUFMAN, A., SHIMONY, E., 3D Scan Conversion Algorithms for Voxel-Based GRaphics. Proc. ACM 1986 Workshop on Interactive 3D Graphics, pp.45-76, 1986.

[35] CRASSIN, C., NEYRET, F., SAINZ, M., EISEMANN, E., Effcient Rendering of Highly Detailed Volumetric Scenes with GigaVoxels. GPU Pro; A K Peters, pp. 643–676, 2010.

[36] TIAN, F., HUA, W., DONG, Z. BAO, H, Adaptive voxels: interactive rendering of massive 3D models. Vis Comput; 26: 409–419, 2010.

[37] PIWOWAR, L., Voxel method for realistic rendering. PhD Thesis, 2008.

[38] NUBER, C., BRUCKSHEN, R.W., HAMANN, B., JOY, K.I., Interactive Visualization of Very Large Medical Datasets Using Point-Based Rendering. In Proc. SPIE 5029, Medical Imaging 2003: Visualization, Image-Guided Procedures, and Display, 27, 2003.

[39] NIU, Q., CHI, X., LEU, M.C., OCHOA, J., Image Processing, Geometric Modeling and Data Management for Development of a Virtual Bone Surgery System. Computer Aided Surgery, 13(1):30-40, 2008.

[40] YILDIRIM, G., KELEŞ, H.Y., İŞLER, V., Three Dimensional Smoke Simulation on Programmable Graphics Hardware. Parallel Computational Fluid Dynamics 2007, Lecture Notes in Computational Science and Engineering; 67:385-39, 2009.

[41] GROSLAND, N.M., BROWN, T.D., A Voxel-based Formulation for Contact Finite Element Analysis. Computer Methods in Biomechanics and Biomedical Engineering; 5:1:21-32, 2010.

[42] PRIOR, A., Real-Time Collaborative Volumetric Virtual Sculpting with Haptic Force-Feedback, PhD Thesis, 2009.

[43] ECHEGARAY, G., BORRO, D., A methodology for optimal voxel size computation in collision detection algorithms for virtual reality, Virtual Reality; 16:3:205-213, 2012.

[44] TUDORACHE, M., POPESCU, M, TANASIE, R., Constructive Volumetric Modeling.In Proceedings of the International Multiconference on Computer Science and Information Technology, pp. 755–758, 2010.

[45] KAUFMAN, A., COHEN, D., YAGEL, R., Volume Graphics. IEEE Computer; 26:7:51-64, 1993.

[46] FOLEY, J.D., VAN DAM, A., FEINER, S.K., HUGHES, J.F., Computer Graphics: Principles and Practice. 2nd Edition, Addison-Wesley, pp.92-99, 1990.

[47] HEARN, D., BAKER, M.P., Computer GRaphics, C Version. 2nd Edition, Donald Hearn, 1996.

[48] BRESENHAM, J.E., Algorithm for Computer Control of a Digital Plotter. IBM Systems Journal; 4:1:25-30, 1965.

[49] OOMES, S., SNOEREN, P., DIJKSTRA, T., 3D Shape Representation: Transforming Polygons into Voxels. Scale-Space Theory in Computer Vision, pp.349-352, Springer-Verlag, 1997.

[50] SRAMEK, M., KAUFMAN, A., “Alias-free voxelization of geometric objectss”, IEEE Transactions on Visualization and Computer Graphics; 5:3,1999.

[51] DONG, Z., CHEN, W., BAO, H., ZHANG, H., PENG, Q., Real-time Voxelization for Complex Models. PG '04 - Proceedings of the Computer Graphics and Applications, 12th Pacific Conference, pp.43-50, 2004.

[52] EISEMANN, E., DECORET, X., Single-Pass GPU Solid Voxelization for Real-Time Applications. GI '08 - Proceedings of Graphics Interface 2008; 322:73—80, 2008.

[53] OGAYAR, C.J., SEGURA, R.J., FEITO, F.R., Point in solid strategies. Computers & Graphics 29, pp. 616–624, 2005.

[54] SCHWARZ, M., SEIDEL, H.P., Fast parallel surface and solid voxelization on GPUs. ACM Transactions on Graphics; 29:6:179-187, 2010.

[55] JIA, J., QIN, Z., CHEN, J., A New Method on Voxelizing Triangular Mesh Model. Information Technology Journal; 6:8:1286-1289, 2007.

[56] PASSALIS, G., THEOHARIS, T., TODERICI, G., KAKADIARIS, I.A., General Voxelization Algorithm with Scalable GPU Implementation. Journal of Graphics, GPU, and Game Tools; 12:1:61-71, 2007.

[57] ZHANG, L., CHEN, W., EBERT, D. S., PENG, Q., Conservative voxelization. The Visual Computer;23:9-11,783-792, 2007.

[58] HUANG, J., YAGEL, R., FILIPPOV, V., KURZION, Y., An accurate method for voxelizing polygon meshes. In IEEE Symposium on Volume Visualization, pp.119–126, 1998.

[59] CLARK, J.H., Hierarchical geometric models for visible surface algorithms. Communications of the ACM, 19:10:547–554, 1976.

[60] ERICSON, C., Real-Time Collision Detection. Morgan Kaufmann, 2005.

[61] KLOWSKI, J.H., HELD, M., MITCHELL, J.S.B., SOWIZRAL, H., ZIKAN, K., Efficient collision detection using bounding volume hierarchies of k-dops. IEEE Transactions on Visualization and Computer Graphics; 4:1:21–36, 1998.

[62] MONTES, C.T., GPU Voxelization, MSc, Polytechnic University of Catalonia, 2009.

[63] ROGERS, D. F., Procedural Elements for Computer Graphics. McGraw-Hill, Inc., 1985.

[64] KNOLL, A., WALD, I., PARKER, S., HANSEN, C., Interactive Isosurface Ray Tracing of Large Octree Volumes. In Proceedings of the IEEE Symposium on Interactive Ray Tracing, pp.115-124, 2006.

[65] SHROEDER, W., MARTIN, K., LORENSEN, B., The Visualization Toolkit: An Object Oriented Approach to 3D Graphics, Prentice Hall, 2nd edition, pp.159-164, 1998.

[66] NAYLOR, B.F., SCULPT: An Interactive Solid Modeling Tool. Proceeding of Graphics Interface, 1990.

[67] THON, S., GESQUIERE, G., RAFFIN, R., A Low Cost Antialiased Space Filled Voxelization Of Polygonal Objects GraphiCon 2004 Preoceedings, pp.71-78, 2004.

[68] SAMET, H., Applications of Spatial Data Structures. Adison-Wesley, 1989.

[69] MEAGHER, D., Geometric Modeling Using Octree Encoding. Computer Graphics and Image Processing; 19:129-147, 1982.

[70] MORTON, G.M., A computer oriented geodetic database and a new technique in file sequencing 1966, IBM Ltd, 1966.

[71] GARGANTINI, I., Linear octrees for fast processing of three dimensional objects. Computer Graphics and Image Processing 1982; 4:20:365-374, 1982.

[72] CASTRO, R., LEWINER, T., LOPES, H., TAVARES, G., BORDIGNON, A., Statistical optimization of octree searches. Compuer Graphics Forum 2008; 27:6;1557-1566, 2008.

[73] STOCCO, L., SCHRACK, G., Integer Dilation and Contraction for Quadtrees and Octrees. In Proc. IEEE Pacific Rim Conf. Communications, Computers and Signal Processing (PACRIM ‘95), pp. 426-428, 1995.

[74] STOCCO, L., SCHRACK, G., On Spatial Orders and Location Codes. IEEE Transactions on Computers 2009; 58:3:424-432, 2009.

[75] LARSSON, T., MÖLLER-AKENINE, T., LENGYEL, E., On Faster Sphere-Box Overlap Testing. Journal of Graphics, GPU and Game Tools, 12(1): 3-8, 2007.

[76] Open Haptics 3.0 - Programmer’s Guide, SensAble Technologies, 2014.

[77] BASDOĞAN, C., SRINIVASAN, M.A., Haptic Rendering in Virtual Environments. Handbook of Virtual Environments: Design, Implementation, and Applications, pp. 117-134, 2002.

[78] YAU, HT., TSOU, LS., TSAI, MJ., Octree-based Virtual Dental Training System with a Haptic Device. Computer-Aided Design & Applications, 3:415-424, 2006.

ÖZGEÇMİŞ

Gülüzar ÇİT, 12.01.1980 tarihinde Trabzon'da doğdu. İlk, orta ve lise eğitimini Kocaeli ilinde sırayla İpraş İlkokulu ve Körfez Oruç Reis Anadolu Lisesi'nde tamamladı. 1998 yılında başladığı Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümünü lisans eğitiminden 2002 yılında mezun oldu. 2003 yılında başladığı Sakarya Üniversitesi Bilgisayar ve Bilişim Bilimleri Mühendisliği yüksek lisans eğitimini ise 2005 yılında tamamladı. Halen 2004 yılında girdiği Sakarya Üniversitesi Bilgisayar ve Bilişim Bilimleri Fakültesi Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak görev yapmaktadır.