Gazi Üniv. Müh. Mim. Fak. Der.
Cilt 27, No 4, 785-793, 2012

Journal of the Faculty of Engineering and Architecture of Gazi University
Vol 27, No 4, 785-793, 2012

GERÇEK ZAMANLI KALABALIK BENZETİMLERİNDE YENİ BİR DİZİNLEME VE POTANSİYEL ALAN SAKLAMA TEKNİĞİ

Murat HACIÖMEROĞLU*
*Gazi Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Eti Mah. Yükseliş Sok. No:5 MALTEPE / ANKARA
murath@gazi.edu.tr

(Geliş/Received: 25.07.2011; Kabul/Accepted: 21.06.2012)

ÖZET

Gerçek zamanlı kalabalık benzetimlerinde sanal girdilerin dizinlenerek konumsal sorguların karmaşıklığının düşürülmesi ve böylece benzetimin performansının her zaman olabilecek en yüksek seviyede tutulması gerekmektedir. Bu makalede geliştirilen yeni dizinleme yöntemi anlatılmaktadır. Geliştirilen sistem, yol bulma algoritması için literatürde kullanılan çizge veri yapısını kullanarak dizinleme gerçekleştirmektedir. Böylelikle, benzetimde dizinleme için ayrı bir veri yapısı kullanmaya gerek kalmayacağı gibi, uzun ve dar koridorlu (şehir ortamındaki kaldırımlar gibi) benzetim ortamlarında daha performanslı çalıştığı gösterilmektedir. Ayrıca geliştirilen sistem, sabit nesne konum bilgilerini çizge veri yapısının içerisine sıkıştırılarak gömmekte ve geleneksel yöntemlere kıyasla hafıza alanında oldukça yüksek kazanımlar sağlamaktadır. Geliştirilen sistemin geleneksel yöntemler ile performans açısından sınaması, sanal bir şehir ortamında gerçekleştirilmiştir. Sonuç olarak geliştirilen sistem, geleneksel ızgara tabanlı dizinlemeye göre %8 performans artışı sağlarken sabit nesnelerin dizinlenmesinde çok daha az hafıza alanına ihtiyaç duymaktadır.

Anahtar Kelimeler: Gerçek Zamanlı Kalabalık Benzetimleri, Geometrik Dizinleme.

A NEW INDEXING AND POTENTIAL FIELD STORAGE TECHNIQUE FOR REAL-TIME CROWD SIMULATIONS

ABSTRACT

In real-time crowd simulations, the complexity of indexing the virtual entities should be kept low at all times in order to maximize the simulation performance. In this paper, a novel indexing technique is proposed. The proposed system is based on the navigation graph that is already commonly used for path finding in the current virtual crowd simulations. By this way, the simulation does not need a separate data structure for indexing the virtual entities, and at the same time it performs faster than the traditional grid based simulations that have narrow and long corridors (such as urban environments). In addition, while indexing static obstacles, the proposed system compresses and embeds the static obstacle information into the graph structure thus requires less memory than the traditional systems. The performance is evaluated for both the proposed and the traditional grid based systems on a virtual city environment. As a result, it has been shown that the proposed system on average works 8% faster than the traditional grid based system and requires significantly less memory while indexing static obstacles.

Keywords: Real-Time Crowd Simulations, Geometric Indexing.

1. GİRİŞ (INTRODUCTION)

Üç boyutlu sanal ortamlarda kalabalık canlandırılması günümüzde önemli bir araştırma alanı haline gelmiştir. Bunun nedeni bilgisayar oyunları [1], eğitim benzetimleri [2,3], şehir planlama [4,5], tahliye benzetimleri [2,6] ve sinema filmleri [7] gibi uygulamaların, sanal kalabalıkları otomatik canlandırma tekniklerini her geçen gün daha fazla kullanmalarıdır. Sanal kalabalıklar ile ilgili yürütülen akademik ve endüstriyel çalışmaların kapsamlı bir incelemesi [8] numaralı referansta bulunabilmektedir. Sanal kalabalıkları canlandırırken 3 boyutlu modellerin görsel kalitesi yanında davranışlarının da modellenmesi istenilen hususlardandır. Kalabalık içerisindeki her bir bireyin, içinde bulunduğu senaryoya uygun bir biçimde davranması gerekmektedir. Kalabalık benzetimlerinde en temel davranış gezinme davranışıdır [9]. Sanal girdilerin bulundukları ortamda istenilen herhangi bir konuma otomatik olarak gidebilmesi gerekmektedir. Örneğin bu makalede incelenen şehir benzetimlerinde tüm sanal yayalar tıpkı gerçek yayalar gibi devamlı bir yerden diğer bir yere ve diğer nesnelerle çarpışma olmaksızın gidebilmelidirler. Gerçek zamanlı sistemlerde ise gerçekçi bir davranış motoruna ek olarak, kullanılacak tüm algoritmaların olabildiğince efektif çalışması gerekliliği gibi bir kısıtlama bulunmaktadır. Bunun nedeni yüzlerce hatta bazen binlerce sanal girdi için yapılacak hesaplamaların sınırlı sayıda işlem gücü ile gerçekleştirilme zorunluluğudur. Gerçek zamanlı bir benzetimde hareket eden binlerce sanal girdinin sık aralıklarla (ideal olarak en az saniyede 30 kere) konum ve yön bilgilerinin güncellenmesi gerekmektedir. Bunu yaparken de çarpışma olmamasına dikkat edilmedir. Olası çarpışma durumlarına önceden önlem almak ya da çarpışma olduysa tepki oluşturabilmek için ortamdaki sanal girdilerin birbirlerinin konumlarından haberdar olması gerekmektedir. Herhangi bir dizinleme tekniği kullanılmadığı zaman, benzetimdeki her sanal girdinin tüm diğer sanal girdileri sorgulaması gerekeceğinden, konum sorgularının karmaşıklığı n adet sanal girdi bulunması durumunda $O(n^2)$ olmaktadır. Elbette gerçek zamanlı bir kalabalık benzetimi için bu kabul edilemez bir karmaşıklık değeridir. Bu nedenle araştırmacılar ortamdaki sanal girdileri değişik teknikler kullanarak dizinlemişler ve karmaşıklığı $O(n)$ düzeyine indirmeye çalışmışlardır [10,11].

Bu makalede gerçek zamanlı kalabalık benzetimlerinde dizinleme sorgularının performansını artıracak çizge veri yapısına dayalı yeni bir model önerilmektedir. Geliştirilen sistem geleneksel yöntemlerin aksine, hem hareketli hem de sabit nesnelerin dizinlenmesini gezinti çizgesi üzerinde gerçekleştirmektedir. Makale şu şekilde düzenlenmiştir; Bölüm 2'de literatürde şu ana kadar konuyla ilgili olarak yapılan çalışmalar tartışılmıştır, Bölüm 3,4 ve 5'de önerilen sistem detaylı olarak anlatılmıştır. Bölüm 6'da ise önerilen sistemin geleneksel yöntemlerle performans ve hafıza kullanımı açısından karşılaştırması yapılmıştır ve Bölüm 7'de bu çalışmada elde edilen sonuçlar sunulmuş ve bazı hususlar tartışılmıştır. Aynı zamanda Bölüm 7'de olası gelecek çalışma alanları üzerine değerlendirmeler sunulmuştur.

2. LİTERATÜRDEKİ ÇALIŞMALAR (LITERATURE BACKGROUND)

Sanal girdilerin ortamda istenilen noktalara otomatik olarak ulaşabilmelerini sağlayabilmek için birçok araştırmacı gezinme çizgesini kullanmaktadır [12-14]. Bu çizgeyi ortamdan otomatik olarak çıkarmak için değişik teknikler kullanılmıştır. Mesela Kavraki [12] Olasılıksal Yol Haritası (Probabilistic Roadmaps, PRM) yöntemini kullanarak robotlar için yürünebilir alanı tanımlayan bir çizge oluşturmuştur. Fakat PRM'ler çok fazla gereksiz uç nokta tanımladığı için gerçek zamanlı kalabalık benzetimlerinde kullanımı kısıtlı kalmıştır. Arikan ve arkadaşları [13] görünebilirlik çizgesini (visibility graph) gezinti çizgesi olarak kullandılar. Fakat görünebilirlik çizgesini kullanan sanal girdiler, çizgenin yapısı dolayısıyla nesnelere çok yakın gezinmektedirler. Bu durum sanal girdisi çok olan benzetimler için olası çarpışma sayısını artırdığından uygun değildir. Böylece araştırmacılar, nesnelere olabilecek en fazla uzaklık sağlayan ve aynı zamanda en az uç sayısı ile ortamı tasvir edebilen çizge türlerinin en uygun olacağı sonucuna vardıklar ve kalabalık benzetimlerinde Voronoi Diyagramları (VD) [14] ya da Doğrusal İskelet (Straight Skeleton) [15] gibi çizge algoritmalarını kullanmaya başlamışlardır.

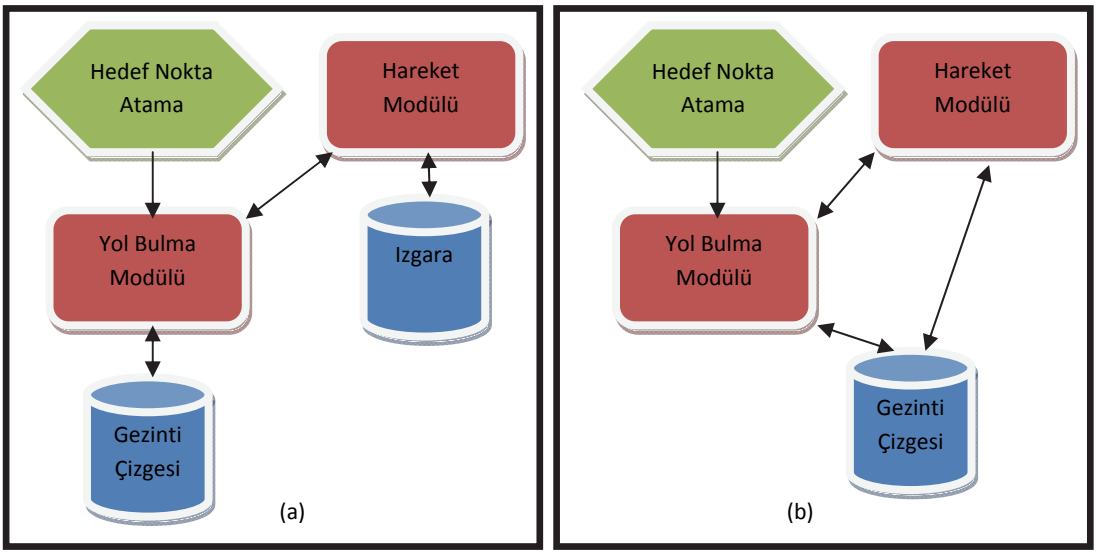
Gerçek zamanlı bir kalabalık gezinme benzetimlerinde efektif bir gezinme çizgesi yanında, dizinleme için ortamın küçük dışbükey poligonlara bölünmesi de gerekmektedir. Bazı araştırmacılar, ortamı Delaunay Üçgenlemesi (DT) ile bölmeyi denemiştir [9,16]. DT'ler, böldükleri alanı tam anlamıyla temsil etmektedirler (hücrelerin birleşimi tam olarak boş alana eşittir). Fakat benzetim sırasında hücreler simetrik olmadıkları için dizinleme ve dizin sorgulama süresi gerçek zamanlı benzetimler için genellikle çok uzamaktadır. Bu nedenle birçok araştırmacı sanal girdilerini dizinlemek için her hücresi eşit boyutlarda ve kare olan ızgara veri yapısını kullanmışlardır [17,19]. Kare tabanlı iki boyutlu ızgara gerçek zamanlı kalabalık benzetimlerinde oldukça sıklıkla kullanılmıştır. Yakın zamanda Joselli ve arkadaşları [10] sanal girdileri üç boyutlu bir komşuluk ızgara yapısına, grafik işlemcisini ve Bitonic sıralama algoritmasını kullanarak yerleştirmişlerdir. Bu sistemde çok hızlı yakınlık (proksimite) sorguları

çalıştırılabilse de sorgular ancak en yakın sanal girdiyi geri döndürebilmektedir. Ancak kalabalık benzetimlerinde sanal girdilerin ortamın durumuna göre çok daha uzak diğer sanal girdilerden haberdar olması gerekmektedir. Örneğin bir şehir benzetiminde sanal yayalar gezindikleri uzun bir kaldırımın yoğunluk durumu ve diğer yayaların konum ve yönlerini hareket algoritmasına bildirmek zorundadır. Izgara tabanlı dizinlemeler bu gibi durumlarda efektif bir çözüm sunamamaktadırlar. Bu nedenle önce Pettrè ve arkadaşları [20] dizinlemeyi çizge tabanında gerçekleştirdiğini belirtmiş ancak gerçekleştirim detaylarından bahsetmemiştir. Bu makale çalışması, çizge tabanlı dizinlemenin performans artışı sağlayacak şekilde nasıl gerçekleştirilebileceğini anlatan bir dizi yeni yöntem önermektedir. Önerilen benzetim sistemi ayrı bir ızgara veri yapısına ihtiyaç duymadığından daha az hafıza alanı gerektirmektedir. Ayrıca uzun ve dar koridorlardan oluşan benzetim ortamlarında ızgara tabanlı dizinlemeden daha performanslı çalışmaktadır.

3. GELİŞTİRİLEN YÖNTEM: ÇİZGE TABANLI DİZİNLEME (THE PROPOSED SYSTEM: GRAPH BASED INDEXING)

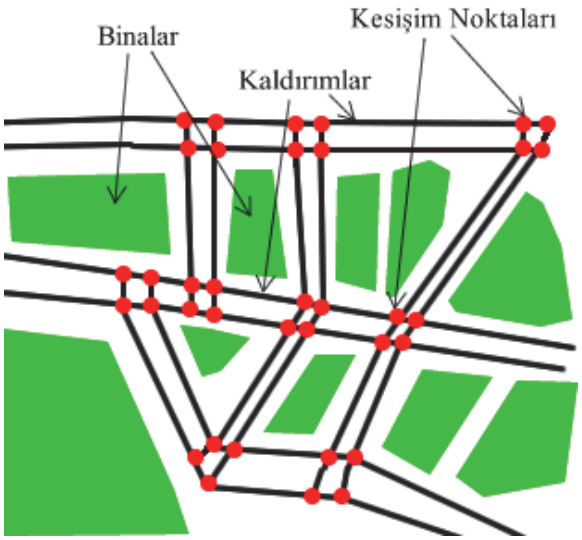
Geleneksel ızgara tabanlı dizinleme yöntemlerinde her sanal girdi kendisini bir ızgara hücresine kaydetmekte ve çevresindeki diğer girdileri sorgularken sadece bulunduğu ya da ek olarak komşu hücreleri sorgulamaktadır. Önerilen sistem de buna benzer çalışmakla beraber, girdilerin dizinlendiği alanlarda farklılık göstermektedir. Önerilen çizge tabanlı dizinlemede her yaya kendini, üzerinde gezindiği çizge kenarına kaydeder. Yayaların bulundukları ortamlarda hedeflerine ulaşmalarını sağlayan bir nokta listesi bulunmaktadır. Bu nokta listesi yol bulma algoritmalarından birini kullanarak oluşturulmaktadır. Yayalar P_n noktasından P_{n+1} noktasına giderken aslında $E_{P_n \rightarrow P_{n+1}}$ kenarından geçmektedirler. Kenarlar, üzerlerinde gezmekte olan yayaların hafıza referanslarını saklamaktadırlar. Dolayısıyla ne zaman bir yaya sorgu yapsa çizge sadece bulunduğu ya da ek olarak komşu kenarların üzerinde bulunan yayaların hafıza referanslarını döndürmektedir. Eğer üzerinde bulunulan kenar ve birinci derece komşu kenarların uzaklığı yeterince büyük değilse, öz yinelemeli olarak komşu (komşunun komşusu) kenarlar da sorgulanacak kenarlar listesine eklenmektedir. Bu işlem yeterli uzaklık sağlanana kadar devam etmektedir. Bu işlem önceden belirlenecek bir en fazla uzaklık değerine göre benzetimden önce ön işlem olarak her kenar için hesaplanıp o kenara kaydedilir. Örneğin, bazı kenarların komşuları bile sorgu listesine eklense yine de yaya istenilen uzaklıktaki yayaları sorgulayamamaktadır. Bu durumda komşuların komşuları da sorguya katılmalıdır.

Önerilen sistemin ızgara tabanlı sistemlerden farkı Şekil 1'de özetlenmiştir. Önerilen sistemde gezinti çizgesi, dizinleme ızgarasının görevini de üstlenmektedir. Izgara, geleneksel gerçek zamanlı kalabalık benzetimlerinde (Şekil 1(a)) sanal girdileri dizinlemek ve hareket algoritmasının konumsal sorgularına cevap vermekle sorumludur. Önerilen sistemde ise (Şekil 1(b)) dizinleme ve konumsal sorgulara cevap verme işlemleri, yol bulma işlemleri ile birlikte, gezinti çizgesi tarafından gerçekleştirilmektedir.



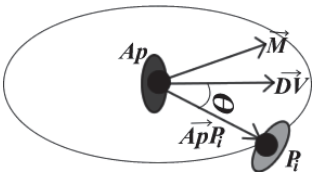
Şekil 1. (a) Izgara tabanlı çalışan sistemin genel yapısı. **(b)** Önerilen çizge tabanlı sistemin genel yapısı. (**(a)** General structure of the grid based system. **(b)** General structure of the proposed graph based system.)

Gezinti çizgesi birçok yolla ortaya çıkarılabilir. Örneğin sabit nesneleri kullanarak ortaya çıkarılacak bir VD [14] ya da DT [9] çizgesi buna örnek verilebilir. Bu makalede bahsedilen gezinti çizgesi sanal bir şehrin kaldırım doğrularının birleşiminden oluşmaktadır. Bu çizgenin ve benzetim ortamının küçük bir örneklemesi Şekil 2'de verilmiştir. Şekil 2'deki kırmızı noktalar gezinti çizgesinin uç noktalarını, siyah çizgiler çizgenin kenarlarını (kaldırımlar) ve yeşil poligonlar ise binaların yere olan dikey izdüşümlerini simgelemektedir. Sanal yayalar bulundukları noktadan diledikleri başka bir noktaya olan bağlı yolu, yol bulma modülüne sormaktadırlar. Yol bulma modülü ise kaynak ve hedef noktalar arasında en kısa ya da kısa bir yolun içerdiği nokta listesini sıralı olarak döndürmektedir. Yol bulunurken, kalabalık benzetimlerinde genellikle tercih edilen A-star algoritması [21] kullanılmıştır. A-star algoritması yolun en kısa olduğunu her zaman garanti etmemektedir ancak, A-Star algoritması sezgisel fonksiyonu *h(x)* sayesinde klasik yol bulma algoritmalarından çok daha hızlı çalışabilmektedir. Yine de geliştirilen sistem yol bulma algoritmasından bağımsız çalıştığı için, bir başka yol bulma algoritması da tercih edilebilmektedir. Daha sonra sanal yayalar bir bir yolu oluşturan noktaları takip ederek hedeflerine ulaşabilmektedirler.



Şekil 2. Yol bulma çizgesi ve benzetim ortamının iki boyutlu iz düşümünün bir örneği. (An example of the navigation graph and the 2D projection the simulation environment)

Ortamda birden çok sanal yaya olduğu düşünülürse, yayaların birbirleri ile çarpışmadan kaçınmaları ve gerçekçi bir şekilde yürümeleri gerekmektedir. Önerilen sistemin hareket algoritması iki katmandan oluşmaktadır. Bunların ilki Reynolds'ın kaçınma kuvvetinin [22] uygulanmasından ibarettir. Ancak kaçınma kuvvetinin tek başına, özellikle yoğun bölgelerde çarpışmayı engelleyemediği gözlemlenmiştir. Bunun nedeni kaçınma kuvvetinin daima en yakındaki tekbir olası çarpışmayı dikkate almasıdır. Bu nedenle ikinci bir katman geliştirilmiş ve bu katmanın çıktısı ilk katman ile doğrusal olarak birleştirilmektedir. Esasen ikinci katman Helbing ve Molnar'ın sosyal güç tabanlı benzetimlerine benzemektedir [23]. İkinci katmanda her yaya için, aynı ya da komşu çizge kenarlarında olan diğer yayaların konumları hesaba katılarak bir hareket vektörü hesaplanmaktadır. Yayanın bir sonraki yapacağı hareketin yön ve hızını belirleyen hareket vektörü PV Denklem 1 kullanılarak hesaplanır. Denklem 1'deki A_p noktası kendisi için hesaplama yapılan yayanın konumunu, P_i noktası diğer yayanın konumunu ve DV vektörü de yayanın o anki yönünü belirtmektedir. Burada dikkat edilmesi gereken bir nokta da PV vektörünün yayanın vücut yönünü belirlemediğidir. Yayanın gerçek yönü ilk aşamada hesaplanır ve sadece ilk aşama tarafından değiştirilir. PV vektörü yayanın kalabalık bölgelerde yavaşlamasını sağlamaktadır. Güçler tabanlı hareket algoritmasının bir örnek çalışma biçimi Şekil 3'de sunulmaktadır.



Şekil 3. Kullanılan güçler tabanlı hareket algoritmasının bir örneklemesi. (An example of the employed force based motion algorithm)

Denklem 1'deki β değişkeni yayaların birbirlerinden ne kadar uzaklaşma eğiliminde olacaklarını belirleyen deneysel bir değişkendir. β değişkeni bu çalışmada *1,0* olarak alınmıştır.

$$\overrightarrow{PV} = \sum_{i=0}^{n} \left(\frac{-\overrightarrow{A_p P_i}}{(\|\overrightarrow{A_p P_i}\|)^2 x\, \beta} \right) x\left(1 + \overrightarrow{DV} \cdot \overrightarrow{A_p P_i}\right) \quad (1)$$

Her sanal insan hareket algoritmasının gerekli hesaplamalarını gerçekleştirmek üzere çevresindeki diğer sanal insanların ve engellerin pozisyonlarını dizinleme modülünden sormaktadır. Dizinleme modülü sanal insanların sorgularına cevap vermenin yanında onların sürekli değişen konum bilgilerine göre dizinlerini güncellemekle de sorumludur. Benzetim sırasında bir çizge kenarında yüzlerce yaya hafıza referansı ekleme ve çıkarma işlemi gerçekleştirilmektedir. Dizi veri yapısındaki pahalı silme sonrası kaydırma işleminden kaçınmak için bağlı listeler kullanılmıştır.

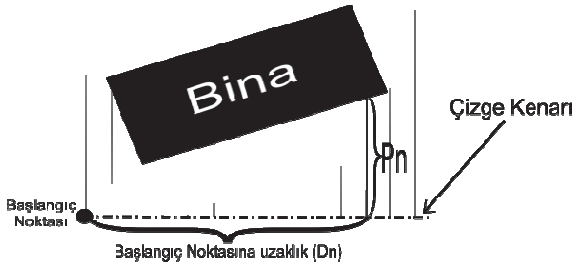
Izgara veri yapısı hareketli nesneleri dizinlemenin yanında, sabit nesneleri de dizinlemektedir. Örneğin şehir ortamında binalar ızgara yapısına dizinlenerek yakın yayalar tarafından kolaylıkla sorgulana-bilmektedirler. Öyleyse önerilen sistemin de sabit nesneleri dizinleyebilmesi gerekmektedir. Bir sonraki bölümde sabit nesnelerin çizge veri yapısına nasıl dizinleneceği anlatılacaktır.

4. ÇİZGE TABANLI DİZİNLEMEDE SABİT NESNELERİN POTANSİYEL ALANLARI (POTENTIAL FIELDS OF THE STATIC ONSTACLES IN THE GRAPH BASED INDEXING)

Izgara tabanlı dizinlemenin önemli bir avantajı, sabit nesnelerin oluşturacakları potansiyel alanları kolaylıkla içinde barındırabilmesidir. Bu potansiyel alan, benzetim sırasında sanal insanlar tarafından sorgulanır ve hareket algoritmasına ayrı bir güç vektörü olarak katılır. Önerilen sistem ızgara yapısını ortadan kaldırdığı için önerilen sistemin de buna benzer bir güç vektörünü benzetim sırasında sanal girdilere sunabilmesi gerekmektedir.

Izgara tabanlı dizinlemede sanal yayalar içinde bulundukları hücre ve çevre hücrelerdeki önceden hesaplanmış sabit engel potansiyelini kullanarak efektif bir şekilde ortamdaki sabit engellerden kaçınabilmektedirler. Önceden hesaplama sürecinde her hücreye, kendisine yakın sabit engellerin uzaklıkları ve yönleri hesaplanıp kayıt edilir. Benzetim sırasında sanal girdiler ek bir hesaplama yapmadan kaydedilen değerleri kullanarak sabit nesnelerden kaçınabilmektedirler. Oysaki çizge tabanlı dizinlemede bu bilgi bulunmamaktadır. İşte bu yüzden sabit nesne potansiyel bilgisini çizge veri yapısının içerisine yerleştirecek bir tekniğe ihtiyaç duyulmaktadır. Bunun için önceki çalışma [24] dizinleme çizgesi veri yapısına uyarlanmıştır.

Bölüm 3'de anlatıldığı gibi sistemdeki çizge veri yapısının temel taşı çizgenin kenarlarıdır. Her kenar aslında sağında ve solunda kalan kendisine en yakın iki taraftaki sabit nesnelerin bilgilerini tutabilir. Bunun için benzetim başlamadan ön işlem ile her bir kenar için başlangıç noktasından başlayarak belirli aralıklarla kenara dik ışınlar gönderilir. Bir ışının sabit bir nesneye ilk defa çarptığı nokta ile ışının çıkış noktasının oluşturduğu doğru parçasının uzunluğu, kenarın o noktadaki potansiyelini belirleyebilmektedir. Bu durumun bir örneği Şekil 4'de gösterilmiştir. Bundan sonra anlatılan yöntemler kenarın bir tarafı dikkate alınarak ifade edilmiş olsa da aslında tüm işlemler kenarın her iki tarafı (sağ ve sol) için de yapılmaktadır.



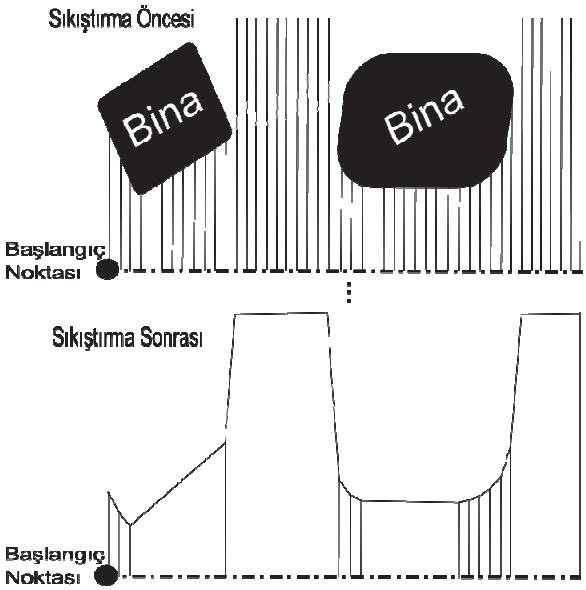
Şekil 4. Her çizge kenarı için sabit nesne potansiyelinin hesaplanması. (Determining the static obstacle potential for each edge of the graph)

Ön hesaplama sonucunda her kenarda, Şekil 4'de ki bir dizi (belirli aralıklarla) D_n ve P_n uzaklıkları tutulmaktadır. Hafıza kullanımını daha da aza indirmek için D_n ve P_n değerleri sıkıştırılmaktadır. Aslında birçok sıralı potansiyel değeri birbiri ile korelasyon göstermektedir. Örneğin bir binanın bir kenarı genellikle doğrusal olduğundan bu kenara çarpan örnekleme ışınları da Şekil 5'de bir örneğinin verildiği gibi doğrusal bir yapılandırmada olacaktır. İşte bu durum göz önüne alınarak Algoritma 1 geliştirilmiştir. Gerçekte, Algoritma 1 sıkıştırma yaparken çoğu potansiyel uzaklığın aynı doğru üzerinde olmasının avantajını kullanmaktadır. Şekil 5'de örnek bir sabit nesne potansiyel hesaplanması ve Algoritma 1 kullanılarak sıkıştırılması verilmektedir.

Algoritma 1: Potansiyel alanları sıkıştırma

Girdi: *N tane potansiyel değer dizisi L*
Çıktı: *Sıkıştırılmış değerler dizisi CL*
i=0
while *i≤N* **do**
    L[i]'yi CL dizisine ekle
    i'yi artır
    fark = L[i-1] - L[i]
    while *L[i-1] - L[i] = fark AND i ≤ N*
        i'yi artır
    end
    L[i-1]'yi CL dizisine ekle
end

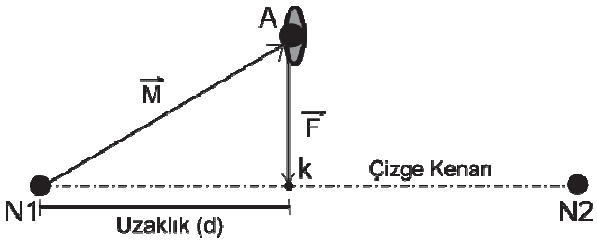
Sabit nesne potansiyeli sıkıştırma sonucunda düzensiz bir hal almaktadır. Örneğin kenarın ilk beş metresi bir potansiyel değerler kümesini (doğru parçası) ifade ederken sonraki iki metre başka bir potansiyel değerler kümesini ifade etmektedir. Dolayısıyla sorgulama ızgara tabanlı dizinlemede olduğu gibi $O(1)$ zamanda gerçekleştirilememektedir. Ancak potansiyel bilgisi çizge kenarlarında ikili ağaç yapısı içerisinde tutularak arama $O(\log n)$ zamana indirilmiştir (n kenardaki farklı potansiyel alan sayısını belirtmektedir). İkili ağaçta kenarın ortasındaki potansiyel noktası (*CL[N/2]*) kök olarak alınmakta ve öz yineli olarak her iki taraftaki noktalara da aynı işlem uygulanarak ata-çocuk ilişkisi belirlenmektedir. Bu ek işlem maliyetine rağmen çizge tabanlı potansiyel sıkıştırması Bölüm 6'da değinileceği gibi oldukça yüksek hafıza kazancı sağlamaktadır.



Şekil 5. Potansiyel alanların sıkıştırma işleminin öncesi ve sonrası. (Before and after the compressing process of the potential fields)

5. ÇİZGEYE GÖMÜLÜ POTANSİYEL ALANIN SORGULANMASI VE KULLANIMI (QUERYING THE EMBEDDED POTENTIAL INFORMATION)

Saklanan potansiyel bilgisinin benzetim sırasında sorgulanması ve kullanılması basit bir takım vektörel işlemler sonucunda kolaylıkla gerçekleştirilebilmektedir. Örneğin Şekil 6'da gezinti ve dizinleme çizgesinin bir kenarı örneklenmiştir. Kenarın uç noktaları olan N1 ve N2 noktalarının L vektörünü oluşturduğu düşünülürse Denklem 2'yi kullanarak uzaklık değeri kolaylıkla bulunabilir.



Şekil 6. Çizge kenarında gezinen bir sanal yaya. Çizgedeki potansiyel alanı bulabilmek için N1 noktasının k noktasına olan uzaklığın bulunması gerekmektedir. (A virtual pedestrian navigating the graph edge. In order to find the potential field, the distance between N1 and k points should be determined)

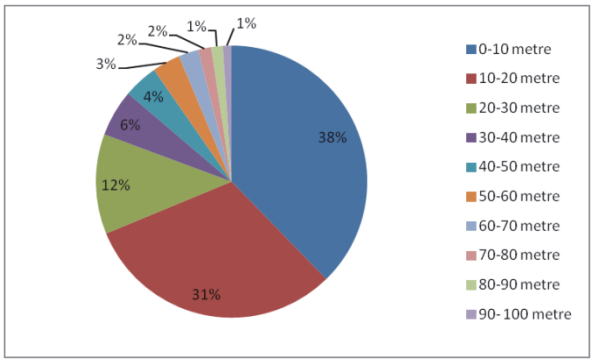
$$d = \frac{L \cdot M}{|L|} \tag{2}$$

Kendilerine uygulamaları gereken potansiyel vektörü Denklem 3'ü kullanarak bulunabilmektedir. Ancak bu vektörün yönü doğru olsa da boyutu bu aşamada yanlıştır. Çünkü bulunması gereken sanal yayanın çizge kenarına olan uzaklığı değil, kenarın o noktada önceden hesaplanan potansiyelidir. Bu potansiyel bilgisi Denklem 2'de hesaplanan uzaklık verisini kullanarak çizge kenarından getirilir. Gelen değere göre F vektörünün uzunluğu (gücü) belirlenir. Daha sonra hesaplanan potansiyel vektörü son hareket vektörüne katılır.

$$F = d\frac{L}{|L|} - M \tag{3}$$

6. SINAMA ORTAMI VE PERFORMANS ANALİZİ (THE TEST ENVIRONEMT AND THE PERFORMANCE ANALYSIS)

Geleneksel ızgara tabanlı ve önerilen çizge tabanlı dizinleme yöntemlerini karşılaştırmak amacıyla sanal bir sınama ortamı yaratılmıştır. Sınama, yaklaşık 10000 kenardan oluşan, Şekil 2'de bir kısmı gösterilmiş bir çizge üzerinde gerçekleştirilmiştir. Gezinti çizgesinin kenarlarının boy dağılımları Şekil 7'de verilmiştir. Sanal şehir *2km x 2km* boyutlarındadır. Sınama sırasında 10000 sanal yaya devamlı şehirde gezinti yapmaktadır. Yayaların başlangıç ve bitiş noktaları her iki dizinleme sistemini de sınarken aynı olarak belirlenmiştir. Her iki sınamada da tüm yayaların hareket ettirilmesini sağlayan döngünün çalışma zamanı kaydedilmiştir. Izgara tabanlı dizinlemede ortam her bir kenarı 10 metre olan eşit karelere bölünmüştür. Çevredeki yayaları tespit etmek için, içinde bulunulan ve ek olarak komşu hücreler sorgulanırken, çizge tabanlı dizinlemede ise içinde bulunulan kenar ve ona komşu olan kenarlar sorgulanmıştır. Komşu kenarlar bir önceki bölümde anlatıldığı gibi önceden hesaplanmış ve yayaların her zaman en az 10 metreye kadar diğer yayaları fark etmesi sağlanmıştır. Benzetim, Intel® Pentium Dual Core® 2.4GHz işlemcili, 3GB hafızaya sahip ve grafik kartı olarak da Geforce® 9800GTX bulunduran bir kişisel bilgisayarda gerçekleştirilmiştir.



Şekil 7. Benzetim ortamındaki gezinme çizgesinin kenar-boyut dağılımı. (The edge-magnitude distribution of the navigation graph used in the simulation environment)

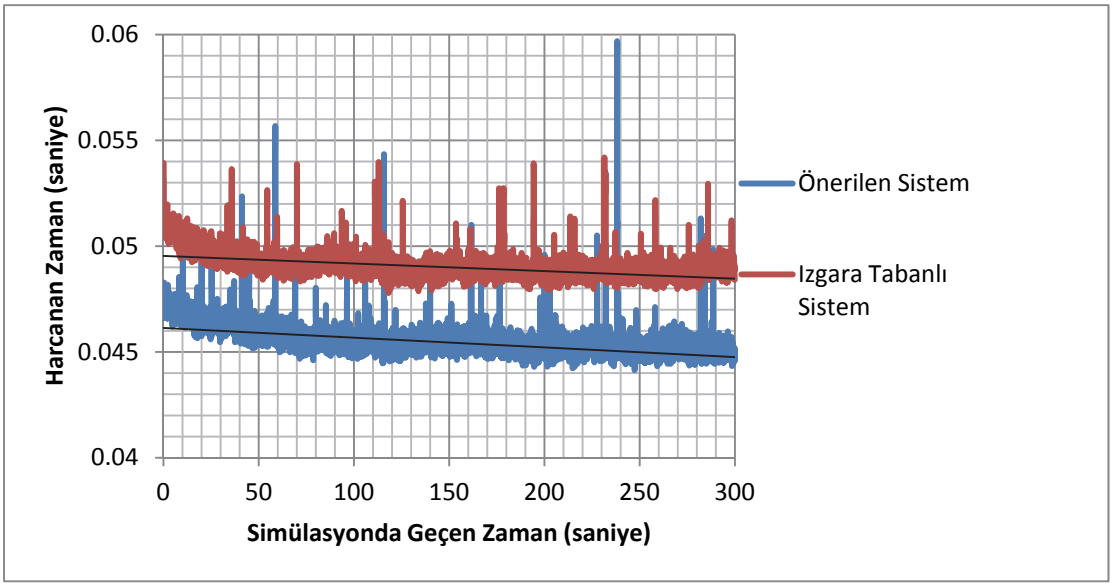
Şekil 8, her iki dizinleme sisteminin de 5'er dakikalık çalıştırılmasının performans sonucunu göstermektedir. Izgara tabanlı dizinleme kullanan benzetimde tüm yayaların hareket ettirilmesi için harcanan ortalama işlemci zamanı *0,049* saniye iken çizge tabanlı

dizinleme kullanan benzetimde ortalama zaman *0,045* saniye olmuştur. Önerilen sistem sınama ortamında geleneksel ızgara tabanlı dizinlemeye göre *%8*'lik bir performans artışı sağlanmıştır. Şekil 8 yakından incelendiğinde, çizge tabanlı benzetimin bazı anlarda ızgara tabanlı benzetimden daha yavaş çalıştığı gözlemlenmektedir. Bunun nedeni çizgenin bazı yerlerinde çok fazla ve kısa kenarın birbirine bağlı olması ve istenilen sorgu uzaklığına erişilebilmek için çok fazla kenarın sorgulanması gerekliliğidir. Önerilen sistemin kullanılması halinde bu durumun dikkate alınmasının önemli olduğu düşünülmektedir.

Önerilen çizge tabanlı dizinlemenin performans avantajının dışında hafıza kullanımı da sınanmıştır. Izgara tabanlı dizinleme kullanıldığında, koşulan benzetim alanında her ızgara hücresi *0,5 metre* boyutlarındadır (potansiyel hesaplama için kullanılan örnekleme ışınların aralarındaki uzaklık da *0,5 metre* olarak alınmıştır). Bu durumda ortamdaki sabit nesneleri dizinleyebilmek için 4000×4000 boyutlarında bir ızgara gerekmektedir. Böylece potansiyel bilgisinin depolanması için *61 MB* hafıza alanı gerekmektedir. Ancak potansiyeli çizge kenarlarında tutuğumuz zaman sadece *0,9 MB* alana ihtiyaç duyulmuştur.

7. SONUÇ (CONCLUSION)

Gerçek zamanlı kalabalık benzetimlerinde çizgelerin yol bulmadaki sistem performansına olan olumlu etkileri sıklıkla tartışılmaktadır. Benzetim ortamının yürünebilir boş alanları çizgeler ile tasvir edilebilmekte ve sanal girdilerin yol bulma modülünün kullanımına sunulmaktadır. Çizge yapısı yol bulma için olmazsa olmaz bir bileşendir. Bu makalede önerilen yöntemler ile aslında sistemde yol bulma için kullanılan çizge veri yapısının dizinleme için de kullanılması bazı durumlarda (ör. Şehir benzetimi) hem daha az hafıza kullanımı hem de daha iyi performans sunduğu ortaya konulmaktadır. Önerilen sistem sanal bir şehir parçasında sınanmış ve geleneksel ızgara tabanlı dizinlemeden %8'lik bir performans artışı sağlanmıştır. Buna ek olarak önerilen sistem, geleneksel olarak iki boyutlu hücresel otomata şeklinde saklanan sabit engel potansiyel bilgisini tek boyutlu iz düşüm türevleri halinde saklamakta ve bunun sonucu olarak da oldukça düşük hafıza alanı kullanımı sağlamaktadır. Ancak unutulmamalıdır ki bu makalede sunulan çizge tabanlı dizinleme yöntemi her ortam için uygun olamamaktadır. Örneğin yol bulma çizgesi kenarlarının çok küçük ve sık olduğu ortamlarda (bazen bu 1 metreden küçük yüzlerce komşu çizge kenarı demektir) dizinlemenin çizge ile yapılmasının performans açısından olumsuz etkiler doğurabileceği açıktır. Buna ek olarak, simetrik ızgara tabanlı sistemlerin grafik işlemcisindeki doku hafızası ile birebir örtüşebildiği için (her piksel bir ızgara hücresine denk gelebilmektedir) paralel işlenebilme açısından avantajının olduğu da unutulmamalıdır. Önerilen sistem ile simetrik ızgara ve dışbükey poligonlar tabanlı dizinleme tekniklerini avantaj ve dezavantajları açısından incelenmesi araştırmacı ve geliştiricilere yön göstermesi amacıyla Tablo 1'de verilmiştir.



Şekil 8. Önerilen sistem ve ızgara tabanlı dizinleme tekniğinin sınama sonrası performans açısından kıyaslanması. (The performance comparison between the proposed and grid based indexing techniques)

Tablo 1. Dizinleme tekniklerinin karşılaştırılması. (Comparison of the indexing techniques)

Dizinleme Tekniği	Avantajı	Dezavantajı	Uygun Kullanım Yeri
Dışbükey Asimetrik Poligonlar [9,16]	Boş alanı eksiksiz biçimde ifade eder.	Dizinleme safhası pahalı. Kalabalık benzetimleri için ideal değil.	Boş alanların küçük ve sık olduğu ortamlar.
Simetrik Izgara [25,26]	Hızlı dizinleme, kolay gerçekleştirim.	Boş alanı tam ifade edemez. Dar ve uzun koridorlarda ancak önceden belirlenen mesafedeki sorgulara cevap verebilir.	Boş alanların geniş ve büyük olduğu ortamlar.
Önerilen Sistem	Doğru ortamlarda en az hafıza gereksinimi ile en hızlı dizinleme.	Geniş boş alan olan ortamlarda boş alanları yeterince kaplayamaz.	Boş alanların uzun ve dar olduğu ortamlar.

Bazen de bir ortam hem sık ve küçük hem de uzun ve dar çizge kenarlarından oluşmaktadır. Bu gibi durumlarda ise hem ızgara hem de çizge tabanlı dizinleme yapısı birleşik (hybrid) bir şekilde kullanılabilir. Ayrıca hangi dizinleme yönteminin daha iyi sonuç verebileceğini otomatik olarak bulan bir model de gelecek çalışmaların konusunu teşkil etmektedir.

KAYNAKLAR (References)

1. Reese, B. ve Stout, B., "Finding a Pathfinder", **AAAI**, California, USA, 69-72, 1999.
2. Courty, N. ve Musse, R.S., "Simulation of Large Crowds in Emergency Situations Including Gaseous Phenomena", **Computer Graphics International**, New York, USA, 206-212, 2005.
3. Takahashi, S. ve arkadaşları, "Spectral-Based Group Formation Control", **Eurographics**, Munich, 639-648, 2009.
4. Penn, A. ve Turner, A., "Space Syntax Based Agent Simulation", **International Conference on Pedestrian and Evacuation Dynamics**, 99-114, 2002.
5. Turner, A. ve Penn, A., "Encoding Natural Movement as an Agent-Based System: An Investigation into Human Pedestrian Behaviour in the Built Environment", **Environment and Planning B: Planning and Design**, Cilt 29, No 4, 473-490, 2002.
6. Braun, Adriana, J., E.Bardo ve Musse, R.Soraia, "Simulating Virtual Crowds in Emergency Situations", **ACM Symposium on Virtual Reality Software and Technology**, Monterey, USA, 244-252, 2005.
7. Ulicny, B., Heras, P. ve Thalmann, D., "Crowdbrush: Interactive Authoring of Real-Time Crowd Scenes", **ACM SIGGRAPH/Eurographics Symposium on Computer Animation**, Grenoble, France, 243-252, 2004.
8. Pelechano, N., Allbeck, J.M. ve Badler, N.I., **Virtual Crowds: Methods, Simulation, and Control**.: Morgan & Claypool Publishers, 2008.
9. Lamarche, F. ve Donikian, S., "Crowd of Virtual Humans: A New Approach for Real Time Navigation in Complex and Structured Environments", **Computer Graphics Forum**, Cilt 23, No 3, 509-518, 2004.
10. Joselli, M., Passos, E.B., Zamith, M., Clua, Montenegro, A.E., ve Feijo, B., "A Neighborhood Grid Data Structure for Massive 3D Crowd Simulation on GPU", **VIII Brazilian Symposium on Digital Games and Entertainment**, Rio de Janeiro, Brazil, 121-131, 2009.
11. Bandi, S. ve Thalmann, D., "Space Discretization for Efficient Human Navigation", **Eurographics**, vol. 17, Lisbon, Portugal, 195--206, 1998.
12. Kavraki, L. ve Latombe, J.C., "Probabilistic Roadmaps for Robot Path Planning", **Practical Motion Planning in Robotics: Current Approaches and Future Directions**., 33-53, 1998.
13. Arikan, O., Chenney, S. ve Forsyth, A.D., "Efficient multi-agent path planning", **Eurographics Workshop on Computer Animation and Simulation**, Manchester, UK, 151-162, 2001.
14. Pettrè, J., Laumond, J.P. ve Thalmann, D., "A Navigation Graph for Real-Time Crowd Animation on Multilayered and Uneven Terrain", **V-CROWDS**, Lausanne, Switzerland, 81-89, 2005.
15. Haciomeroglu, M., Laycock, R.G. ve Day, A.M., "Automatic Spatial Analysis and Pedestrian Flow Control for Real-Time Crowd Simulation in an Urban Environment", **The Visual Computer**, Cilt 24, No 10, 889-899, 2008.
16. Kallmann, M., Bieri, H. ve Thalmann, D., "Fully Dynamic Constrained Delaunay Triangulations", **Geometric Modelling for Scientific**

Visualization, 241-257, 2003, ISBN 3-540-40116-4.

17. Loscos, C., Marchal, D. ve Meyer, A., "Intuitive Crowd Behaviour in Dense Urban Environments Using Local Laws", **Theory and Practice of Computer Graphics**, Birmingham, UK, p. 122, 2003.

18. Tecchia, F., Loscos, C. ve Chrysanthou, Y., "Visualizing Crowds in Real-Time", **Computer Graphics forum**, Cilt 21, No 4, 753-765, 2002.

19. Shao, W. ve Terzopoulos, D., "Environmental Modeling for Autonomous Virtual Pedestrians.", **Symposium on Digital Human Modeling for Design and Engineering**, Iowa City, USA, 1-8, 2005.

20. Pettrè, J., Grillon, H. ve Thalmann, D., "Crowds of Moving Oobjects: Navigation Planning and Simulation", **IEEE International Conference on Robotics and Automation**, Roma, Italy, 1-7, 2007.

21. Nilsson, N.J., **Principles of Artificial Intelligence**.: Tioga Pub. Co, 1980.

22. Reynolds, W.C., "Steering Behaviors for Autonomous Characters", **Game Developers Conference**, California, USA, 763-782, 1999.

23. Helbing, D. ve Molnar, P., "Social Force Model for Pedestrian Dynamics", **Physical Review**, Cilt 51, No 5, 4282-4286, 1995.

24. Haciomeroglu, M., Laycock, R.G. ve Day, A.M., "Efficient Encoding of Potential Field Maps into Agent Navigation Graphs", **Computer Graphics International (Kısa Bildiriler)**, İstanbul, 320-323, 2008.

25. Treuille, A., Cooper, S., Popovic, ve Z., "Continuum crowds", **SIGGRAPH**, California, USA, 1160-1168, 2006.

26. Oguz, O., Akaydin, A., Yılmaz, T. ve Güdükbay, U., "Emergency Crowd Simulation for Outdoor Environments", **Computers & Graphics**, Cilt 34, No 2, 136-144, 2010.