T.C
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAREKETLİ NESNELERİN İNDEKSLENMESİ

Seçkin MANDACI

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Yrd.Doç.Dr. Cavit TEZCAN

Edirne-2010

T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAREKETLİ NESNELERİN İNDEKSLENMESİ

Seçkin MANDACI
Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı

Bu tez 21/12/2010 tarihinde aşağıdaki jüri tarafından kabul edilmiştir..

Yrd. Doç. Dr. Cavit TEZCAN
Danışman

Prof. Dr. Mesut RAZBONYALI
Üye

Yrd. Doç. Dr. Deniz TAŞKIN
Üye

ÖZET

Yüksek Lisans Tezi, Hareketli Nesnelerin İndekslenmesi, T.C. Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı.

Gelişen ve değişen dünya şartlarında teknolojinin girmediği ve kullanılmadığı alan artık yok gibidir. Bilgisayarların ve elektronik diğer araçların bu gelişime ve kullanım yaygınlığına paralel olarak görüntü, ses gibi çoklu ortam verilerinin depolanması ve istenildiğinde tekrar kullanılması gibi sorunları halledebilecek çözüm ve tekniklerin geliştirilmesi önem arz etmektedir. Çoklu ortam veri tabanlarında tutulan verilere hızlı ve kolay erişmenin yollarından birisi etkili bir indeksleme yöntemi kullanmaktır.

Bu çalışmada; hareketli nesne tespit ve takip metotları ve indeksleme teknikleri incelenmiştir.

Bu tez 2010 yılında yapılmıştır ve 64 sayfadan oluşmaktadır.

ANAHTAR KELİMELER: Hareketli Nesne, R Ağaç, İndeksleme, Nesne Tespit, Veri Seti

ABSTRACT

Postgraduate Thesis, Indexing of Moving Objects, T.C. Trakya University, Institute of Natural Sciences, Department of Computer Engineering.

In the developing and changing world conditions, there is hardly any area that technology does not get into and is not used. In parallel with this development and penetration, it is highly important that computers and other electronical devices develop solutions and techniques for the problems such as storing and reusing the multimedia data such as image and sound. One of the rapid and easy ways of reaching the data that is kept in the multimedia databases is using an effective indexing technique.

In this study; methods of detecting and tracking moving object and indexing techniques are examined.

This thesis is completed in 2010 and consists of 64 pages.

KEY WORDS: Moving Objects, R Tree, Indexing, Object Detection, Data Set

TEŞEKKÜR

Bu çalışmanın hazırlanmasında bana yol gösteren, destek ve yardımlarını esirgemeyen danışman hocam Yrd. Doç. Dr. Cavit TEZCAN'a, çalışmalarımda bana yardımcı olan Yrd. Doç. Dr. Deniz TAŞKIN'a ve değerli hocalarıma, aileme teşekkürlerimi sunarım.

ÖZET……i
ABSTRACT……ii
TEŞEKKÜR……iii
1. GİRİŞ 1
2. HAREKETLİ NESNELERİN TESPİTİ VE TAKİBİ 2
2.1. Arka Plan Modelleme 2
2.1.1. Arka Plan Modelleme Yöntemleri 3
2.1.1.1. PFinder 5
2.1.1.2. HRR (En çok tekrarlanma oranı) 6
2.1.1.3. PFinder ve HRR Yöntemlerinin Karşılaştırması 7
2.2. Sıralı Görüntülerin Farkının Alınmasıyla Nesne Tespit Etme 7
2.3. Bölütleme ile Nesne Bulma 8
2.4. Cam Shift Yöntemi ile Nesne Bulma 10
2.5. Nesne Takibinde Dikkat Edilmesi Gereken Noktalar 10
2.6. Nesne Takibi Metotları 10
2.6.1. Nokta Takibi 12
2.6.2. Çekirdek Takibi 12
2.6.3. Silüet Takibi 13
2.6.4. Kalman Filtresi ile Nesne Takibi 14
2.6.5. Koşullu Yoğunluk Yayılımı Metodu ile Nesne Takibi 15
3. İNDEKSLEME YÖNTEMLERİ 17
3.1. İndeksleme Nedir? 17
3.2. İndeksleme Niçin Gereklidir? 17
3.3. İndeksleme için Kullanılan Teknikler 17
3.3.1. B+ Ağacı 17
3.3.2. R-Ağaç 21
3.3.2.1. R-Ağaç İndeks Yapısı 21
3.3.2.1.1. M ve m parametrelerinin ayarlanması 25
3.3.2.1.2 R-Ağaç ve İkincil Bellek Kullanılması 25
3.3.2.1.3. Taşma ve aşağı taşma 25
3.3.2.2. Algoritmalar 26
3.3.2.2.1. Arama 28
3.3.2.2.2. Ekleme 29
3.3.2.2.3. Silme 31
3.3.2.2.3. Bir Düğümü Bölme 32
3.3.2.2.3.1. Ayrıntılı bölme algoritması 33
3.3.2.2.3.2. İkinci dereceden yük algoritması 33
3.3.2.2.3.3. Doğrusal yük algoritması 33
3.3.2.3. R Ağacın türevleri 34
3.3.2.3.1. R* Ağaç 34
4. VİDEO DOSYALARININ VERİ SETİNE DÖNÜŞTÜRÜLMESİ 35
4.1. Nesnelerin Çıkarımı 35
4.2. İndeksleme Yöntemi 38
4.3. Kullanılacak Nesne Çıkarım Yönteminin Belirlenmesi 40
4.4. Kullanılacak Erişim Yönteminin Belirlenmesi 40
5. R-AĞAÇ YAPISI İLE HAREKETLİ NESNELERİN İNDEKSLENMESİ 43
6. SONUÇ 62
7.KAYNAKLAR……63

1. GİRİŞ

İçinde bulunduğumuz zaman dilimi "bilgi çağı", "teknoloji çağı", "bilgisayar çağı" gibi çeşitli isimlerle adlandırılmaktadır. Şüphesiz ki her geçen gün belirli bir zaman periyodundaki gelişmeler ve yenilikler geçmişteki herhangi bir zamandaki aynı uzunluktaki periyotlardaki gelişmelerden çok daha fazladır. Bunun sonucu olarak bu yeniliklerin hayatımızdaki, günlük yaşantımızdaki yeri ve önemi giderek artmakta, teknolojinin nimetlerinden hemen her yerde yararlanmaya çalışmaktayız. Bilgisayar, internet, mobil cihazlar ile bilgiye ulaşmak hemen her yerde mümkün hale gelmiştir.

İnternet, bilgisayar, görüntü teknolojilerinin bu hızlı gelişimine paralel bilgilerin saklanması ve ulaşılması gibi ihtiyaçlarda ortaya çıkmıştır. Artan bu bilgi miktarını en uygun şekilde saklamayı, ihtiyaç duyulduğunda tekrar nasıl ulaşabiliriz sorusu ve bu alandaki çalışmalar devam etmekle birlikte gelecekte yapılacak çalışmalar içersinde daha da önemli bir yere sahip olacağı düşünülmektedir.

Görüntü dosyalarının boyutlarının çok yer tutması ve değişen ortamlarda çekilmesinden dolayı bunların indekslenmesi ve tekrar başarılı ve hızlı bir şekilde ulaşılması çokta kolay olmamakta ya da uzun zaman almaktadır. Bu konularla ilgili birçok akademik çalışma yapılmış ve halen de yapılmaktadır. Ancak çok başarılı ve sorunsuza yakın bir yöntem şu an için mevcut değildir. Her geçen gün geliştirilen bazı metotlarla en iyi performansı yakalamak için çalışmalar sürmektedir. Bu tezde de amaçlanan hareketli nesneleri takip ederken ve indekslerken kullanılan bir metodun incelenmesi ve bir veri seti üzerine uygulanması olacaktır.

Hareket halindeki nesnelerin konum-zaman boyutunda sürekli bir değişim halinde olması, sabit resim ve görüntülere göre çalışılması daha zor kılmaktadır. Bu başarı durumunu etkileyen faktörlerin başındadır. Öncelikle hareketli nesnelerin tespit edilmesi ve takip edilmesi daha sonrada bunların belirli indeksleme metotları ile veri tabanlarına kaydedilmesi gerekmektedir.

2. HAREKETLİ NESNELERİN TESPİTİ VE TAKİBİ

İçinde bulunduğumuz evren sürekli bir hareket ve değişim halindedir. Güneş, dünya, yıldızlar, galaksiler evren içinde sürekli bir hareket halinde yine gezegenler güneşin etrafında ve kendi etrafında, bunların uyduları da aynı şekilde sürekli bir değişim içindedir. İçinde canlıların yaşadığı ve yaşam olan güneşin bir gezegeni olan dünyada da sürekli bir değişim vardır. İnsanoğlu tarih boyunca sürekli olarak yeniliklerin peşinde koşmuş ve değişimin bir parçası olmuştur. Son çeyrek asırdaki gelişmeler baş döndürücü niteliktedir. Bu zaman diliminde bilgisayar, internet, cep telefonu, video kamera gibi teknoloji ürünleri hayatın bir parçası olmuştur.

Video görüntüleri, coğrafik bilgi sistemleri, taşıt tanıma sistemleri gibi hareketli görüntüler üzerinde işlem yapabilmek için değişik metotlar denenmiştir. Bazılarında çok uzun işlemler gerekmekte ve çok büyük zaman kayıplarına neden olmaktadır. Bu tür çalışmalarda hareketli nesnelerin öncelikle tespit edilmesi gerekir.

Hareketli nesnelerin tespiti ile ilgili yapılan çalışmalarda en çok başvurulan yöntem ardışık iki çerçeve arasında aynı olmayan pikselleri bulma yöntemidir. İki çerçeve arasındaki piksel farklılıklarına bakılarak nesnenin hareketi hakkında bilgi ve çıkarımlarda bulunulabilir. Bir kameradan gelen görüntü analiz edilirken ilk önce hareketli bir nesne olup olmadığına bakılır, varsa boyut, konum gibi özellikleri elde etmek gerekir. Hareketli nesneleri tespit için kullanılan yöntemlerden ikisi: arka plan modelleme ve ışıl akış yöntemleridir.

2.1. Arka Plan Modelleme

Hareket eden nesnelerin tespitinin en kolay yolu, var olan kare ile bir önceki karenin karşılaştırılmasıdır. Bu karşılaştırma sadece iki kare arasındaki farkı bulmayı sağlar. Hareket eden nesnelerin tutarlı ve tam şekilde bulunması işlemi ise ancak arka planın oluşturulması, bu modelin ani ışık değişimi, arka planda oluşan değişiklikler gibi etmenlere karşı modelin güncellenebilmesiyle mümkündür.

Arka plan modelleme konusunda uygulanabilen en temel varsayım, hareket etmeyen cisimler haricinde kalan görüntünün istatistiksel verileriyle modellenebilecek

düzgün bir davranış sergilemesidir. Eğer bunu sağlayan bir model oluşturulursa, modele uymayan parçalar; sahne alanına giren, hareket eden nesneleri belirtir. Bu işlem *Arka Plan Modelleme* ya da *Arka Plan* olarak adlandırılmaktadır. Arka plan modelleme için kullanılan yöntemlerin en büyük bölümünü tümevarım yöntemler oluşturur. Bu yöntemler, görüntüyü oluşturan her bir karenin piksel değerlerinin dağılımını kullanarak arka planı modeller. Arka planın gerçekçi bir şekilde modellenmesi, eğitim için 10 – 30 saniye süresince video karelerinin istatistiksel değerlenin yorumlanması ile olur [Ekinci ve Gedikli, 2005].

Eğitim süreci içeren tekniklerde, arka planın gerçeğine en yakın şekilde modellenebilmesi için, eğitim süresince hareketli nesnelerin mümkün olduğunca az, hatta hiç olmaması gerekmektedir. Fakat gerçek durumlarda bu çoğu zaman mümkün değildir. Kalabalık alışveriş merkezlerinden, sürekli trafik akışı olan yollara kadar çok geniş bir aralıktaki sahnelerde arka plan modelleme çalışması yapılması gerekebilir. Bu yüzden modelin öğrenim süresinde bu tarz durumlara karşı çözüm getirmesi gerekmektedir.

2.1.1. Arka Plan Modelleme Yöntemleri

Arka plan modelleme yöntemlerinden, piksel değerlerini kullanan yöntemlerinden *PFinder* ve *HRR* incelenecektir. Yapılan uygulamalarda her iki yöntem için de piksel değerlerinin gri tonlamalı değerleri kullanılmaktadır. Şekil 2.1 ’de, bu modellerin uygulamasında kullanılan test videosunun 25 karede bir alınmış görüntüleri bulunmaktadır.



Şekil 2.1. Örnek bir videodan 25 karede bir alınmış görüntüler

2.1.1.1. PFinder

PFinder yöntemi, arka plan modelini yeni karedeki piksel değerlerini α katsayısıyla çarpıp güncellemesi ile arka plan modeli oluşturan bir yöntemdir [Wren vd., 1997]. Bu model var olan arka planı, ışık değişimi, arka plan yapısının farklılaşması gibi değişikliklerle arka planı dinamik olarak güncellemektedir. *PFinder* yönteminde kullanılan arka plan piksel değerinin hesaplama yöntemi formül (2.1)'deki eşitlikte gösterilmiştir. Eşitlikteki α katsayısı, her yeni karenin arka plan modelini güncelleme katsayısıdır. Bu değerin az veya fazla seçilmesine göre model yapısı değişmektedir. Az seçilen α değeri, modelin değişikliklere karsı tepki süresini düşürürken, hareketli nesnenin de bir kısmının arka plana dahil edilmesine yol açar. Fazla seçilen α değeri ise modelin değişikliklere karşı tepki süresini artırır, bu da ani ışık değişimi vb. değişikliklerin modele yansımasının zaman almasına yol açar.

$$B_t = (1- \alpha)B_{t-1} + \alpha\, I_t \qquad (2.1)$$

B_t = t anındaki arka plan piksel değeri

B_{t-1} = t-1 anındaki arka plan piksel değeri

I_t = t anındaki görüntünün piksel değeri

α = güncelleme katsayısı ([0,005 – 0,100] arası değerler kullanılmaktadır.)

Artıları:

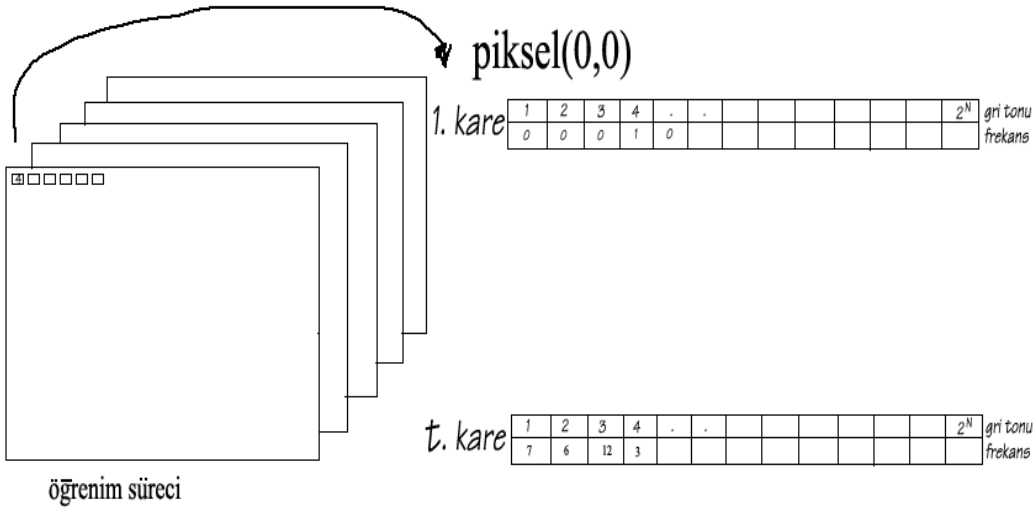
- Dinamik olarak arka planı güncellemesi
- Arka planın değişikliklerinin modele dinamik olarak yansıması
- Bellek ihtiyacının az olması

Eksileri:

- Ani değişikliklerin arka plan modeline yansımasının zaman alması
- Arka planda hareket çok ise (özellikle belli alanda yoğunlaşmış hareketler var ise) arka plan modelinin bozulmaya uğraması
- (1) eşitliğindeki α değeri optimum seçilmezse modelin verimli olmaması
- İlk alınan karenin arka plan olarak atanmasından (ilk kare hareketli nesneyi de içeriyor olabilir) dolayı arka planın düzeltilmesinin zaman alması

2.1.1.2. HRR (En çok tekrarlanma oranı)

HRR modelinin çıkış noktası, görüntüyü oluşturan karelerdeki piksellerin en sık görülenlerinin arka planı oluşturduğu düşüncesidir [Ekinci ve Gedikli, 2005]. Belirli bir öğrenme süresince karelerdeki piksel değerleri alınır, frekansı en çok olan pikseller arka plan değeri olarak atanır. Bir dizide parlaklık değerlerinin görünme frekansı tutulur, öğrenme süresi içinde her karede bu değerler güncellenir. Öğrenme süreci tamamlandığında, en çok frekansa sahip piksel değerleri arka plan modelini oluştur. Şekil.2.2'de her piksel için tutulan tablo gösterilmiştir. Bu tabloda her piksel için *t* sayıdaki öğrenim karesi boyunca gri ton değerlerinin frekansı tutulmaktadır. Öğrenim sonunda her piksel için en yüksek frekansa sahip gri ton değeri, o pikselin arka plandaki değerini oluşturmaktadır.



Şekil 2.2. HRR öğrenme süreci gösterimi

Arka plan oluştuktan sonra, modeli arka planda oluşan değişikliklere karşı güncellemek için de öğrenim süreci belli bir süre sonra tekrarlanmalıdır.

Artıları:

- Modelin, aslından çok az sapmayla oluşması

Eksileri:

- Öğrenim süresince gerekli olan fazla bellek ihtiyacı
- Model oluşturma aşamasında fazla CPU zamanı tüketimi
- Arka plan değişiminin tekrar öğrenme sürecine girmeden modele yansımaması

2.1.1.3. PFinder ve HRR Yöntemlerinin Karşılaştırması

PFinder ve *HRR* yöntemleri piksel bazında arka plan oluşturma yöntemlerindendir. Bu iki yöntem arasındaki en belirgin fark; *PFinder* yönteminin dinamik olarak arka plan modelini oluşturup güncellemesine karşın, *HRR* yönteminin belirli bir öğrenim süreci sonrasında modeli oluşturup, öğrenme süreci yenilenene kadar modeli güncellememesidir.

2.2. Sıralı Görüntülerin Farkının Alınmasıyla Nesne Tespit Etme

Sıralı görüntülerin farkının alınmasıyla nesne bulma, video akısında arka arkaya gelen görüntüdeki piksellerin renk değerlerinin farkını alarak bu farkı bir eşik seviyesi ile karsılaştıran bir yöntemdir (McIvor, 2000).

Tespiti yapan yöntemin görüntü alımı yapan kameranın sabit, hedef nesnenin hareketli olduğu durumlarda kullanımı uygundur.

$$| \text{görüntü}_{i}(x,y) - \text{görüntü}_{i-1}(x,y)| > T$$

Bu fark sayesinde görüntüdeki yer değişimleri ortaya çıkmaktadır. Bu yöntemin diğer aşamaları arka plan farkı ile nesne bulma yöntemindeki gibidir. Farklı olarak diğer yöntemlerde arka planda yer almayan bütün nesneler bulunurken, bu yöntemde yalnızca

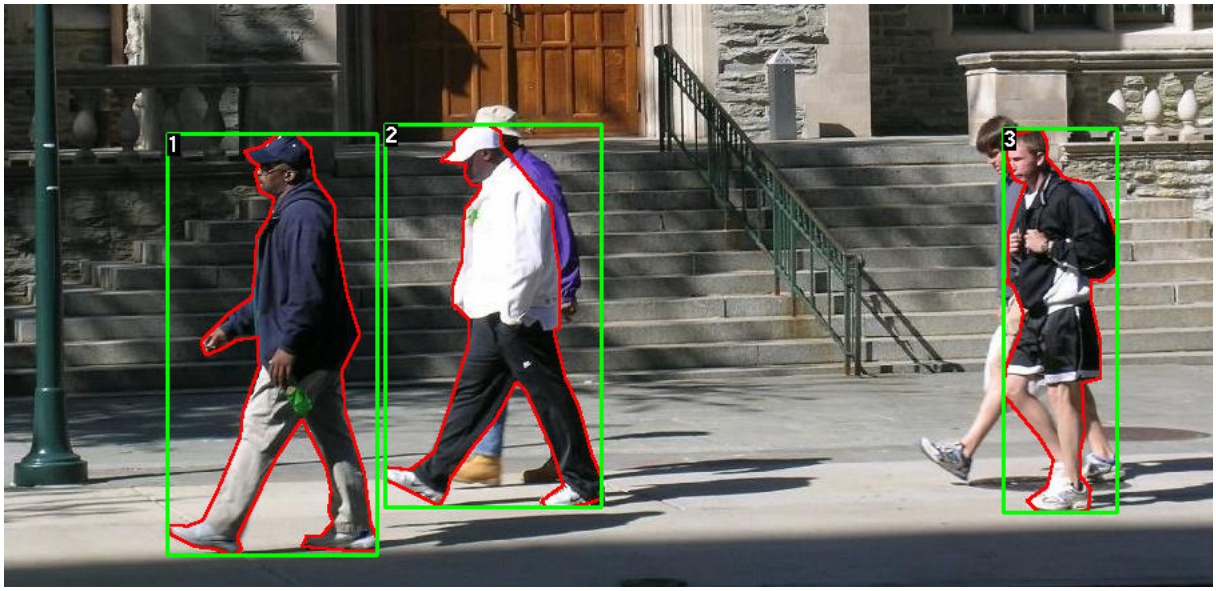
hareketli bölgeler, yani bir önceki görüntüye göre farklılık gösteren bölgeler tespit edilir.

2.3. Bölütleme ile Nesne Bulma

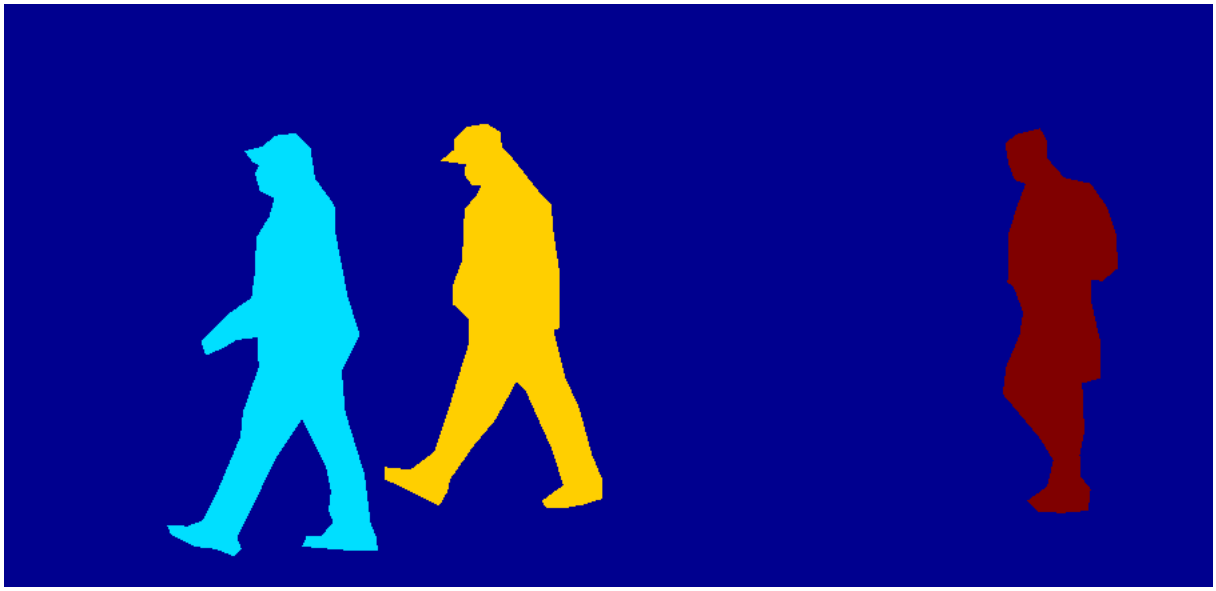
Görüntü bölütleme algoritmaları, görüntüdeki benzer renkli bölgeleri gruplamaktadır. Nesne bulma amacıyla görüntü bölütleme yapılarak elde edilen parçalar üzerinden hedef nesne belirlenebilir (Shi ve Malik, 2000).

Bölütleme ile nesne bulmada kullanılabilecek yöntemlerden biri öğretmenli öğrenme yöntemidir. Bu yöntemde, önceden özellikleri çıkarılıp öğretmenli öğrenme yöntemi ile sisteme öğretilmiş bir hedef nesne bulunmaktadır. Bölütleme işlemi sonucu elde edilen her bir parçanın özellikleri çıkarılır ve aday nesnenin hedef nesne olup olmadığı test edilir. Hedef nesne bulunduğunda takip aşamasına geçilir. Bu yöntemde hedef nesneyi temsil eden özelliklerin seçimi sistemin başarılı çalışmasında önemli bir rol oynar.

Bölütleme ile nesne bulmada kullanılabilecek başka bir yöntem ise Şablon Eşleştirme (Template Matching) yöntemidir. Takip edilmesi amaçlanan hedef nesne ile bölütleme sonucu elde edilen her bir parça şablon eşleştirme yöntemi ile birbiriyle kıyaslanır. Hedef nesneye en çok benzeyen parça eğer benzeme oranı eşik seviyesi değerini de geçmişse hedef nesne olarak tanınır ve takibe başlanır. Şekil 2.3'de bölütleme ile nesnelerin tespitine dair bir örnek görülmektedir.



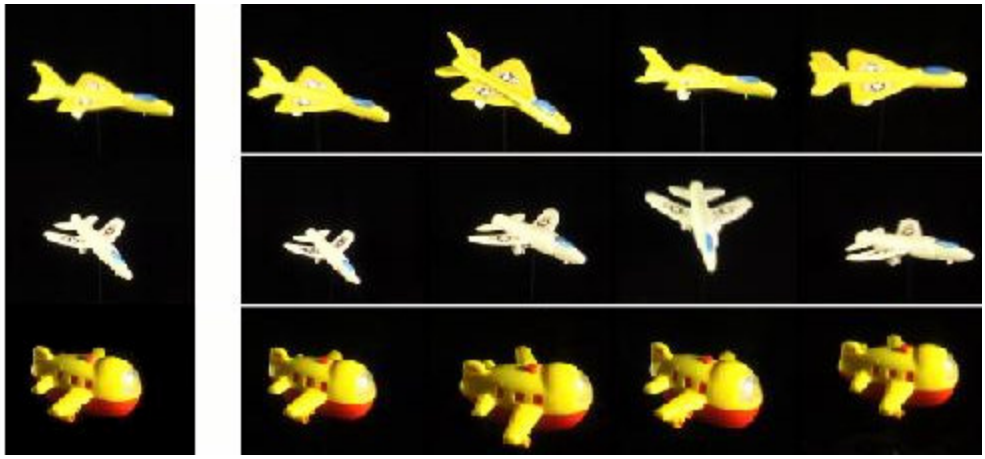
a) Ön plandaki nesneler



b) Ön plandaki nesnelerin Bölütleme ile çıkarılmış hali

Şekil 2.3. Ön plandaki nesnelerin bölütleme ile bulunması

Bölütleme ile nesne bulmada kullanılabilecek diğer bir yöntem ise takip edilecek hedef nesnenin değişik açılardan çekilmiş görüntülerinin sisteme öğretilmesidir (Şekil 2.4). Bu sayede nesnenin hareketi sırasında alabileceği değişik pozlar da tanınabilir. Olası nesneler Şablon Eşleştirme yöntemi kullanılarak kıyaslanır ve hedef nesne bulunur.



Şekil 2.4. Nesnelerin değişik açılardan çekilmiş pozları

2.4. Cam Shift Yöntemi ile Nesne Bulma

Nesne bulmak için kullanılan diğer bir yöntem de CamShift adı verilen yöntemdir. CamShift, Continuously Adaptive Mean Shift sözcüklerinin kısaltılmasıdır ve Türkçe anlamı Sürekli Uyarlamalı Ortalama Değer Kaymasıdır. Gary R. Bradski'nin 1998 yılında ortaya koyduğu bir yöntemdir. Çalışma temeli nesnelerin renk dağılımlarının (histogramlarının) arama kriteri olarak kullanılmasıdır.

2.5. Nesne Takibinde Dikkat Edilmesi Gereken Noktalar

Nesne takip edilmeye başlamadan önce bazı hususları öncelikle belirlemek gerekir. Bunlar Nesne takibi metotları, nesne gösterimleri ve nesne takibinde kullanılacak özellikler gibi kriterlerdir.

2.6. Nesne Takibi Metotları

Güncel olarak en çok tercih edilen Nesne takibi metotları Çizelge 2.1'de görülmektedir. Bu çizelge A.Yılmaz, O. Javed, M. Shah, 2006 tarafından hazırlanmıştır.

Hem parametrik hem de parametrik olmayan modeller nesnenin tanımını yapmak için kullanılabilirler.

Çizelge 2.1. Nesne takibi metotları

Kategoriler	Yapılan Çalışma
Nokta Takibi	
• Belirleyici(Determinist) Metotlar	MGE izleyicisi [Salari ve Sehi 1990], GOA izleyicisi [Veenman ve diğerleri 2001].
• İstatistiksel Metotlar	Kalman Filtresi[Broida ve Chellappa 1986], JPDAF (Joint Probabilistic Data Association Fitler) [Bar-Shalom ve Foreman 1988], PMHT (Probabilistic Multiple Hypothesis Tracking) [Streit ve Luginbuhl 1994].
Çekirdek Takibi	
• Şablon ve yoğunluk tabanlı görünüm modelleri	Mean-Shift [Comaniciu ve diğerleri 2003], KLT (Kanade Lucas Tomasi Feature Tracker) [Shi ve Tomasi 1994], Layering [Tao ve diğerleri 2002].
• Çoklu görüşlü görünüm modelleri	Eigentracking [Black ve Jepson 1998], Varyasyon metotları[Bertalmio ve diğerleri 2000],
Siluet Takibi	
• Çevre çizgisi yayılımı	Durum uzay modelleri [Isard ve Blake 1998], Varyasyon metotları [Betalmio ve diğ.2000], Sezgisel metotlar [Ronfard 1994].
• Şekil Eşleştirme	Hausdorff [Huttenlocher ve diğerleri 1993], Hough dönüşümü [Sato ve Aggarwal 2004], Historam [Kang ve diğerleri].

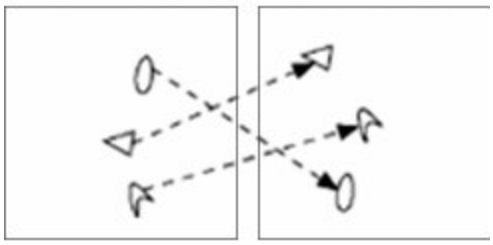
Örnek olarak bir nesneyi temsil etmek için seçilen model nokta olsun. Noktanın ve noktanın temsil ettiği nesnenin yapabileceği hareket sadece öteleme hareketidir.

Nokta yerine çember gibi bir geometrik şekil gösterimi seçilirse izdüşümsel dönüşümler gibi parametrik hareket modelleri daha uygundur. Bu tip gösterimler esnek olmayan nesnelerin hareketini çözümlemek için uygundurlar. Esnek olan nesneler için siluet ve çevre çizgisi yöntemleri daha uygundurlar.

Çizelgede gösterilen ve kategorilere ayrılmış olan metot ve modelleri sırasıyla incelenecektir.

2.6.1. Nokta Takibi

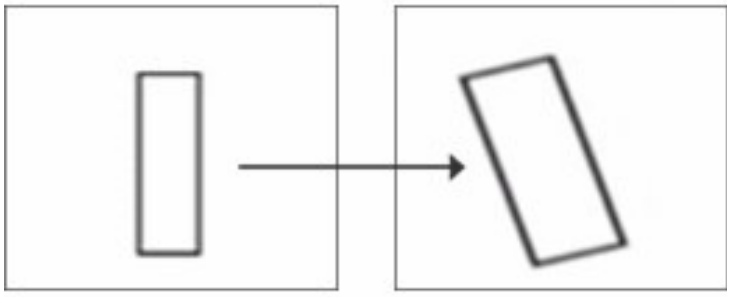
Bu modelde bir görüntüdeki ardışık çerçevelerdeki nesneler nokta ile temsil edilmektedir. Bu yaklaşımda her bir karedeki hedef nesneyi bulacak harici bir nesne bulma algoritmasına ihtiyaç vardır. Nokta eşleştirme özellikle nesnelerin görüntüye giriş ve çıkışlarının çok olduğu, özellikle kaybolma, yanlış bulma durumlarında oldukça zorlanan bir yöntemdir. Şekil 2.5'de yöntemin çalışması gösterilmektedir.



Şekil 2.5 Nokta takibi

2.6.2. Çekirdek Takibi

Bu yöntemde herhangi bir nesne çekirdek olarak adlandırılmaktadır ve bu çekirdek o bölgenin histogramı ile ilişkilendirilmiş bir dörtgen ile ifade edilmektedir. Bu nesnenin arka arkaya olan resim çerçevelerindeki hareketi hesaplanarak nesneler takip edilmektedir. Bu hareketler dönme, öteleme ve şekil değişimi gibi dönüşümler olabilir.



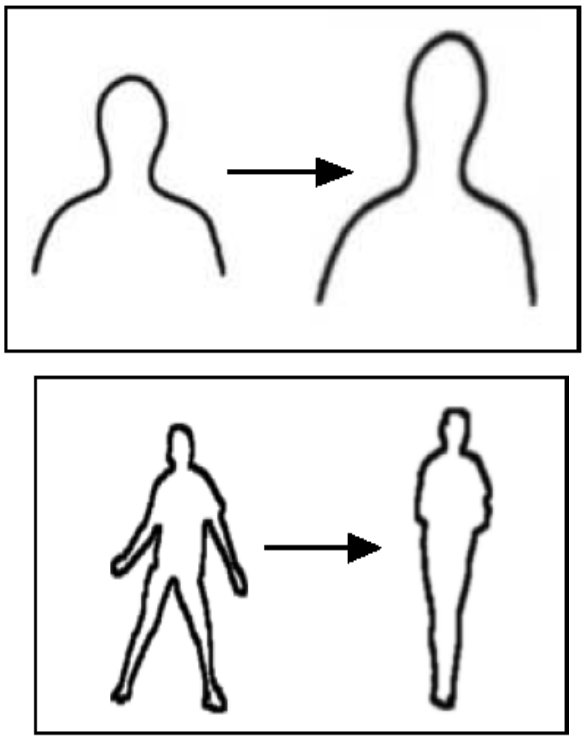
Şekil 2.6. Çekirdek takibi

Dikdörtgen şeklin parametrik dönüşümü Şekil 2.6'da görülmektedir (A. Yılmaz, O. Javed, M. Shah, 2006).

2.6.3. Silüet Takibi

Yaşadığımız dünyada, çevremizdeki her nesnenin sabit bir şekli yoktur. İnsanlar, hayvanlar ve bazı cisimler zamana ve konuma göre şekil değiştirebilmektedirler. Örneğin bir insanın dururken, yürürken, elleri ve bacakları hareket halindeyken, spor yaptığı sıradaki vücut şekli farklılıklar gösterebilir. Şekilleri zamanla değişiklik gösterebilen nesneler için *Siluet Takibi Yöntemi* geliştirilmiştir. Silüet Tabanlı Nesne takibindeki amaç daha önce kullanılmış nesne modeliyle daha sonra gelen görüntülerde nesne bölgesinin bulunarak takip edilmesidir. Bu nesne modeli nesne kenarı, nesne dış çizgisi ve renk histogramı olabilir. Siluet takibi temel olarak iki şekilde yapılabilir; şekil eşleştirme ve çevre dış çizgisi takibi.

Şekil eşleştirmede o andaki görüntüde takip edilen nesnenin silueti aranır. Çevre dış çizgisinde ise takip edilen nesnenin çevre dış çizgisi ile o andaki görüntüdeki değişmiş halinin çevre dış çizgisi ilişkilendirilmeye çalışılır.



Sekil 2.11. Çevre çizgisi oluşturma örnekleri (A. Yılmaz vd., 2006)

2.6.4. Kalman Filtresi ile Nesne Takibi

Nesne takibi amacıyla kullanılan yöntemlerden en bilineni Kalman filtresidir. (Kalman,1960). Bu yöntemin uygulamasının basit olmasından ve gerçek zamanlı çalışabilmesinden dolayı literatürde birçok çalışmada kullanılmıştır. Kalman filtresi, nesne takibinde hedef nesnenin doğrusal hareket ettiğini ve sistemde Gauss paraziti bulunduğunu kabul eder. Nesne takibi amacıyla kameralardan elde edilen ölçümler çoğu zaman parazit içermektedir. Bunun yanında nesne hareketleri de rastgele düzensizlikler içerebilir. Kalman filtresinin de dahil olduğu istatistiksel eşleme metotları, nesne durum tahmini boyunca ölçüm ve model belirsizliklerini de hesaba katarak bu gibi nesne takibi problemlerine çözüm üretmektedir. Kalman Filtresi de diğer istatistiksel eşleme metotları gibi pozisyon, hız ve ivme gibi nesne özelliklerini belirleyebilmek için *durum alan* yaklaşımını kullanır (A. Yılmaz, vd., 2006). Kalman filtresi, Gauss dağılımı olan durumlarda doğrusal bir sistemin durumunu tahmin etmek için kullanılmaktadır. Kalman filtrelemesi üç adımdan oluşur: önerme, doğrulama ve asimilasyon. Bu algoritmada amaç, *t-1* anındaki sistem model önermesi ile *t* anındaki ölçümler kullanılarak *t* anındaki sistemin durumunu bulmaktır. Önerme aşamasında, *t-1* anındaki

sistem modeline ve nesne durumuna dayanılarak *t* anındaki nesnenin durumu önerilir. Ölçüm aşamasında, *t* anındaki görüntü üzerindeki özellikler çıkarılır. Bu özellikler, nesne durumunun doğrusal dönüşümü olarak kabul edilir. Asimilasyon aşamasında ise, önerilmiş durumlar ile ölçülmüş durumlar kombine edilerek nesnenin yeni durumu çıkarılır (Neil Alldrin). Kalman filtresinin, tek şekilli ve Gauss olasılık dağılımının olduğu durumlarda çalışması bu algoritmanın önemli bir sınırlamasıdır. Çünkü genellikle nesneler Gauss dağılımlı değildir. Kalman filtresinin başka bir dezavantajı da karmaşık arka planlara karşı hassas olmasıdır. Kalman Filtresi, bir sonraki görüntü üzerinde nesnenin yeri hakkında tek bir tahmin yapar. Bu sebepten ötürü Kalman Filtresi ile yalnızca tek bir nesnenin takibi yapılabilir.

Bilindiği gibi nesneyi temsil etmesi için seçilen model, nesnenin hareketini kısıtlar. Örneğin; nesne yalnızca bir nokta ile temsil ediliyorsa, noktanın dolayısıyla nesnenin yapabileceği hareket yalnızca ötelemeli bir harekettir. Elips gibi bir geometrik şekil gösterimi kullanılması durumunda ise ilgin ve izdüşümsel dönüşümler gibi parametrik hareket modelleri uygundur (A. Yılmaz vd., 2006). Kalman Filtresi yönteminde hedef nesne, nokta ile temsil edilmektedir. Eğer hedef nesne geometrik bir şekil ile temsil edilebilseydi takip işlemi sırasında nesnenin boyutlarındaki değişiklik ölçülebilirdi. Bu nedenle Kalman Filtresi yöntemi ile yapılan nesne takibinde hedef nesnenin boyutları, kapladığı alan ölçülemez yalnızca görüntü üzerindeki yeri bulunabilir.

2.6.5. Koşullu Yoğunluk Yayılımı Metodu ile Nesne Takibi

Koşullu Yoğunluk Yayılımı yöntemi 1998 yılında Isard ve Blake tarafından geliştirilmiştir. Kullanım amacı karmaşık bir görüntü içerisinde takip edilecek nesnenin dış çizgilerinin tespit edilmesi ve bu ortam içerisinde nesnenin takip edilebilmesidir. Burada zor olan görüntü içinde hangi piksellerin nesnenin kenar çizgileri olduğunu tespit edebilmektir.

Koşullu Yoğunluk Yayılımı Metodunun en önemli farklılıklarından biriside görüntüdeki her piksel üzerinde işlem yapmıyor oluşudur. Bu algoritma işlenecek pikselleri rastgele seçer ve sadece bu piksellerin alt kümelerini işleme alır.

Bu algoritma önerme, güncelleme ve yeniden örnekleme adımlarından oluşan iteratif bir algoritmadır. Her iterasyondaki önerme adımında yeni bir parçacık elde edilerek parçacık kümesi oluşturulur ve her bir parçacığın hareket modeli örneklenir. Güncelleme adımında gözlem modeli kullanılarak her bir parçacığın ağırlığı hesaplanır. O anki gözlemle aynı sonu üreten parçacığın değeri arttırılır. Yeniden örnekleme adımında o andaki parçacıkların ağırlık dağılımlarından yararlanılarak yeni parçacık kümesi oluşturulur. Ağırlığın yüksek olması parçacığın bir sonraki kümede yer alma olasılığının yüksek olması anlamına gelmektedir (Barrera vd., 2005).

Avantajlarının yanında *Koşullu Yoğunluk Yayılımı Algoritmasının* dezavantajları da mevcuttur. Olasılık tavan değerine çıktığında yada yeni ölçümler öncekilerin kuyruğunda yer aldığında örneğin başarısız olduğu görülmektedir. Bu dezavantajların üstesinden gelmek için çeşitli çalışmalar yapılmaktadır. Ayrıca, önceki ve sonraki dağılımların örneklerinin sayısını azaltmak için de birçok çaba harcanmaktadır(Yalçın ve Gökmen, 2005).

3. İNDEKSLEME YÖNTEMLERİ

3.1. İndeksleme Nedir?

İndeksler, bulunan kayıtlı veriler içerisinde veri tabanı tablolarında gerekli kayıtlar aranırken kullanılırlar. Kullanım amaçları sorgu sırasında geçen süreyi en aza indirmek, veriye en hızlı şekilde ulaşmaktır. Bu çok fazla kayıt olmayan veri tabanları için çok önemli değilmiş gibi gözükse de çok fazla veriye sahip büyük ve günden güne kayıt sayısı artan veri tabanları için hayati önem taşımaktadır. Çünkü bu kadar büyük bir veri yumağı içinden istenilen verilere ulaşmak oldukça uzun süreler alabilmektedir.

3.2. İndeksleme Niçin Gereklidir?

Veri tabanlarına kaydedilen verilere istenildiğinde hızlı ve düşük maliyetli bir şekilde ulaşabilmek için verilerin indekslenmesi gereklidir. Özellikle çok büyük verileri barındıran veri ambarlarında bazı sorgulamaların çok uzun süre alacağı kaçınılmazdır. Bunların üstesinden gelmek ve tüm verileri taradıktan sonra istediğimiz bilgilere ulaşmak yerine arananı içeren bir kısım veriyi taramak daha az zaman alır. İndeks kullanılan veri tabanları daha hızlı veriye ulaşmaktadırlar.

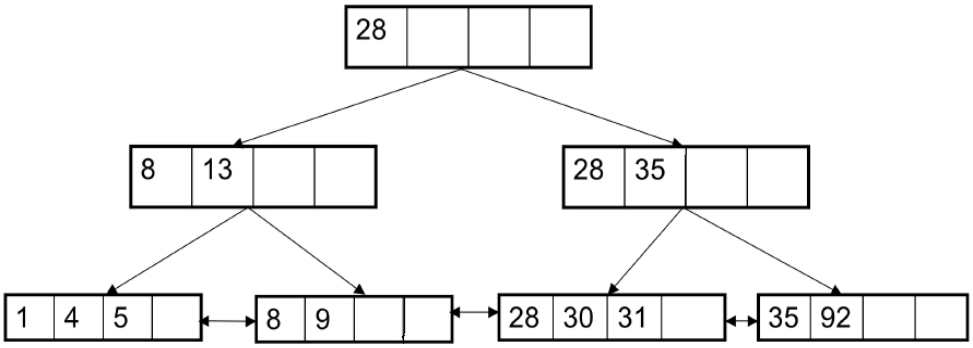
3.3. İndeksleme için Kullanılan Teknikler

3.3.1. B+ Ağacı

B+ ağacı, yeni veri eklerken, bu veriden eksiltme yaparken veya sadece veriye ulaşmak istediğimizde hızlı ve verimli bir şekilde ulaşmak için sıkça tercih edilen, indeksleme amacıyla kullanılan bir ağaç yapısıdır. İndeksleme, her bir veride bulunan ve sadece o veriye özel olan (yani her bir veri için farklı olan) bir değer seçilerek yapılır. Bu değere anahtar denmektedir. B+ ağaçları sahip olduğu verilerin ya da anahtarların sayısına bağlı olarak büyüyüp küçülebilir. Bu yüzden, dinamik bir yapıya sahip olan B+ ağaçlarının yüksekliği değişkendir; ancak dosyaya ekleme ve çıkarma yaparken kullanılan algoritmaların dizaynı dolayısıyla tüm yapraklar aynı yüksekliktedir ve bütün kayıtlar yaprak düzeyinde tutulur. Anahtarlar ise yaprak

olmayan noktalarda tutulur. B+ ağaçlarının her bir indeks parçasındaki anahtar sayısı bir minimum ve bir maksimum değer ile sınırlandırılmıştır. Bu sayıya ağacın derecesi denir ve bir düğümde bulunabilecek maksimum anahtar sayısı minimum anahtar sayısının iki katıdır. B+ ağaçlarını normal bir ikili ağaçtan ayıran en önemli özellik, derecesinin yüksek olabilmesidir. Bu sayede, yapılacak bir operasyonda disk okumaları minimize edilmiş olmakta ve fazla sayıdaki verilerde bile algoritma yüksek hızda çalışabilmektedir. Btrfs, NTFS, ReiserFS, NSS, XFS ve JFS gibi dosya sistemleri de bu ağaç yapısını kullanmaktadır. IBM DB2, Informix, Microsoft SQL Server, Oracle 8, Sybase ASI, PostgreSQL, Firebird ve MySQL gibi ilişkisel veritabanı yönetim sistemleri de tablo indeksleri için bu ağaç yapısını desteklemektedir.

Dosyada kayıtlar sıralı olarak tutulmaktadır. Kayıtların tutulduğu düzeydeki tüm düğümler hem kendi içinde hem de birbirlerine göre sıralıdırlar. Birbirlerine göre sıralı olması iki düğümden birinin içindeki tüm elemanların, diğerinin içindekilerden küçük anahtar değere sahip olduğu anlamına gelmektedir. Buradan da çıkarabileceğimiz gibi anahtar olarak seçtiğimiz değerlerin birbirleriyle kıyaslanabilir olması gerekmektedir. Sayılarda, büyüklük küçüklük ilişkisi kullanılırken, yazılarda alfabetik sıralama göz önünde bulundurulmaktadır. Tüm düğümlerde anahtarlar aynı şekilde sıralı halde tutulur. Anahtarların her birinin önünde ve arkasında bir alt seviyedeki düğümleri gösteren işaretçiler vardır. Alttaki düğümler de sıralıdır ve onu gösteren işaretçinin solundaki anahtarın değeri, alttaki düğümdeki minimum değere sahip anahtarın değerinden küçük veya eşittir. Benzer şekilde işaretçinin sağındaki anahtar değeri de işaretlenen düğümdeki tüm anahtar değerlerinden büyüktür. Veri aranırken düğümün işaretçisinin sağına veya soluna bakılır. Ebeveyn düğümlere bakma işlemi, bulunamayan her anahtar değer için kendini tekrar ederek yapılır; fakat sonunda bir değere ulaşılamıyorsa o düğümün bir alt sınırının olmadığını gösterir. Benzer şekilde düğümün en sonundaysa, o düğümün bir üst sınırının olmadığını ifade eder.



Şekil 3.1. B+ ağaç yapısına bir örnek

B+ ağaç yapısında Arama algoritması şu şekilde işlemektedir:

1. Kök düğümü bakılan düğüm olarak seçilir.

2. Bakılan düğüm bir yaprak değilse;

    - Bakılan düğümde aranan kaydın anahtar değerine göre arama yapılır
    - Aradığımız anahtar değerden küçük olan en yüksek değerli anahtardan sonraki işaretleyicinin gösterdiği düğüme gidilir
    - Bu düğüm bakılan düğüm olarak seçilir
    - 2. adıma geri dönülür

2. Bakılan düğüm bir yaprak ise aradığımız düğüm ya bu yaprak düğümdedir ya da kayıtların arasında yoktur. Bu yüzden son olarak bu düğümde bir arama yapılır ve aranılan düğüm bulmuş olur.

Ekleme algoritması şu şekildedir:

1. Yeni eklenecek kaydın anahtar değerine göre, olması gereken düğüm bulunur.

2. Eğer bu yaprak düğüm, derecesinin iki katından daha az kayıt tutuyorsa; yani düğümde boşluk varsa kayıt, düğüm içinde sırayı koruyacak şekilde doğru yere eklenir. Kayıt eklendikten sonra algoritma son bulur.
3. Boşluk yoksa düğüm ikiye ayrılır.
4. Yeni bir yaprak oluşturulur ve düğümdeki elemanların yarısı, sıralanmış hali bozmadan bu yeni yaprağa eklenir.
5. Yeni düğümdeki en küçük elemanın anahtar değeri ve adresi ebeveyn düğüme eklenir.
6. Eğer ebeveyn düğüm doluysa o da ikiye ayrılır.
7. Sonra onun ortasındaki anahtar sadece yukarıya kaydırılır.
8. Bu işlem bölünme gerektirmeyen bir ebeveyn düğüm bulunana kadar tekrar eder.
9. Eğer en son kök düğümü de ikiye bölünürse tek bir anahtar değeri ve iki işaretleyicisi bulunan yeni bir kök düğüm oluşturulur. B+ ağaçlarında tüm düğümlerdeki anahtar sayısı derece ve derecenin iki katı arasındadır; ancak kök düğüm buna istisnadır. Kökte, birden derecenin iki katına kadar herhangi bir sayıda anahtar bulunabilir.

Anahtarı verilmiş bir kaydı silmek için ise şu yol izlenir:

1. Anahtarı verilen kayıt, yaprak düğümler arasından bulunur.
2. Düğümde tutulan kayıtlar ağacın derecesine eşit veya ondan büyükse silme işlemi gerçekleştirilir.
3. Daha az kayıt varsa yanındaki aynı ebeveyne sahip düğümden ödünç alınıp bu düğüme eklenir. Bu işleme tekrar dağıtma denilmektedir.
4. Eğer tekrar dağıtma başarısız olursa; yani kardeş düğüm de derecenin altında kayda sahipse, iki düğüm birleştirilir.
5. Birleştirilme yapıldıysa sağdaki düğümü gösteren işaretleyici ve onun anahtarı ebeveynden silinir. Bu durumda ebeveyn de anahtar sayısını azalırken o da aynı şekilde kök düğüme kadar silme operasyonuna tabi tutulur.

B+ ağaçlarının özellikleri aşağıdaki gibidir.

- Ağacın derecesi d, yüksekliği y ve eleman sayısını da n ile gösterilir
- En fazla $n=(2d)^y$ kayıt tutulur

- En az $2d^{y-1}$ anahtar bulunur
- Ağacı tutmak için gerekli olan yer miktarı O(n)'dir
- Yeni bir kayıt eklemek en kötü durumda $O(\log_d n)$'dir
- Bir kaydı aramak en kötü durumda $O(\log_d n)$'dir
- Bir kaydı silmek en kötü durumda $O(\log_d n)$'dir
- Belli bir aralıkta k adet elemanla sıralama yapmak en kötü durumda $O(\log_d n + k)$ kadar işlem gerektirir

Sonuç olarak B+ ağaçları, biraz karmaşık yapısına rağmen sağladığı faydalar sayesinde sıkça tercih edilen bir indeksleme yöntemidir.

3.3.2. R-Ağaç

Zaman içinde teknolojinin gelişimine paralel olarak tek indeksli verileri yönetmede etkili olan B-Ağaç'ın yetersiz kaldığı durumlar ortaya çıkmaya başlamıştır. Multimedya uygulamaları ile birlikte görüntü ve ses ile ilgili saklanan verilerin hızla yaygınlaşması ve bu tür uygulamalardaki verilerin indekslenmesinde tek boyutlu indeksleme yöntemlerinin yetersiz kalması yeni arayışlara başlanmasına neden olmuştur. Geliştirilen ilk yöntemlerden birisi Antonin Guttman tarafından sunulan arama, silme, ekleme gibi algoritmaları etkili olan R-Ağaç yapısıdır [Guttman, 1984].

3.3.2.1. R-Ağaç İndeks Yapısı

R-Ağaç B+ Ağaç yapısına benzer bir yapıdır. B+ Ağaç yapısının *n* boyuta genişletilmiş, çok boyutlu uzayda hacmi olan nesneleri dinamik olarak kontrol edebilen bir indeks yapısıdır. R-Ağacı diğer yapılardan üstün kılan yanı bir dönüşüm tekniği değil gerçek bir uzaysal erişim metodu olmasıdır. Çok boyutlu nesneleri dönüşüm teknikleriyle tek boyuta indirip indekslemek yerine nesneleri çok boyutlu bir ağaç yapısı kullanarak indekslemektedir.

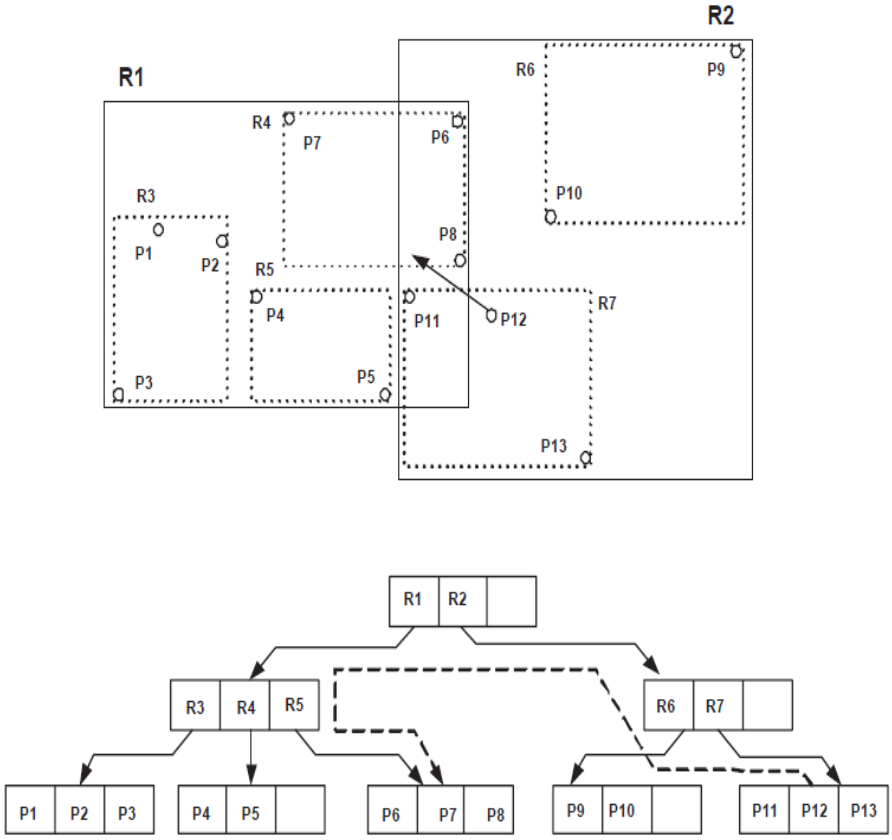
Tamamen dinamik bir indeks var olduğundan, eklemeler ve silmeler arama ile birlikte yapılmalıdırlar. Böylece periyodik olmayan tekrar düzenlemelere gerek kalmaz.

Yaprak düğümler ve yaprak olmayan düğümler olmak üzere R-Ağaç yapısında iki tip düğüm bulunmaktadır. Yaprak düğümler (I, Nesne_P) şeklinde nesneleri sınırlayan dikdörtgeni ifade eden (I_0, $I_{1,...,}$ I_{n-1}) vektör bilgisine ve Nesne_P nesne işaretçisine sahiptirler. Yaprak olmayan düğümler ise (I, cocuk_P) şeklinde alt ağaçlarda bulunan sınırlayıcı dikdörtgenleri sınırlayan dikdörtgenleri ifade eden (I_0, $I_{1,...,}$ I_{n-1}) vektörel bilgisine ve cocuk_P alt çocuk düğümü işaretçisine sahiptirler.

R-Ağaç'ın derecesi (m,M) ikilisi ile ifade edilir. R-Ağaç yapısında her iki düğüm yapısı en az m<m/2 kayıt en fazla M kayıt sayısına sahip olmalıdır.

Guttman'ın R-Ağaç yönteminde her çeşit geometrik nesne indislenebilmektedir. Bununla beraber, ağaçta saklanan veri region tipindeki, d-boyutlu uzlamsal nesneleri çevreleyen minimum d-boyutlu dörtgenlerdir. Bu dörtgenlere **M**inimum **B**ounding **R**ectangles, MBR adı verilir. *MBR*'ların kenarları ortamdaki koordinat sistemine paralel olmalıdır. Bu yapısal düzenlemeler sayesinde ağaç hiyerarşik ve iç içe düzenli bir yapı olabilmektedir.

Bir R-Ağaç'ta iç düğümlerde yer alan MBR'lar bir alt seviyedeki düğümleri kapsamaktadır. R-Ağaç'ta iç içe MBR'lardan oluşan uzlamsal parçalama ile başarılı bir filtreleme sağlanmaktadır. Böylece bir sorgu için R-Ağaç ile filtrelenerek erişilen yaprak düğümlerdeki nesneler tasfiye aşamasıyla, yani nesnenin gerçek uzlamsal şekli analiz edilerek, sorgu koşuluna uygunluğu kontrol edilir.



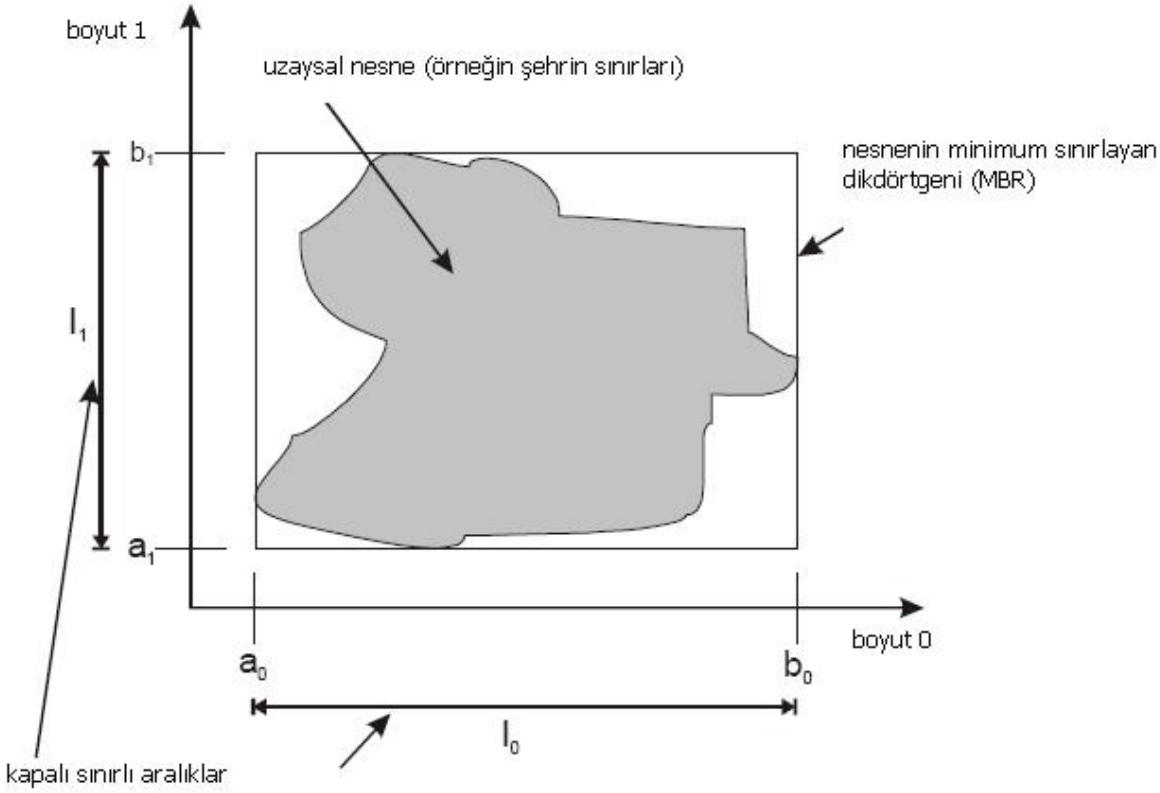
Şekil3.2. R-Ağaç yapısına bir örnek

Şekil 3.2'de örnek bir R-Ağaç yapısı görülmektedir. Kök R1 ve R2 olmak üzere 2 bölgeye ayrılmış yine bu bölgelerde kendi içlerinde alt bölgelere ayrılmış durumdalar.

Bir R-Ağaç, (m,M) parametre ikilisi ile tanımlanmıştır. Buna göre,

- Ağaçtaki her düğüm $m \leq (M/2)$ olmak üzere, en az m, en fazla M indis kaydı tutar. Kök düğüm ise, (yaprak düğüm olmaması kaydıyla) en az iki indis kaydı tutar.
- Yaprak düğümlerdeki her indis kaydı (I,id) bilgi ikilisini tutar. I, id ile işaret edilen ve uzlamsal nesneyi çevreleyen MBR'dır.
- Yaprak olmayan iç düğümlerdeki her indis kaydı , (I, child-id) bilgi ikilisini tutar. I child-id ile işaret edilen bir alt seviyedeki çocuk düğümündeki nesneleri çevreleyen MBR'dır.

Şekil 3.3'te minimum sınırlayan dikdörtgen görülmektedir.



Şekil 3.3. Kapalı sınırlı aralık ve minimum sınırlayan dikdörtgen

Eğer M (makinenin disk sayfası ve n boyut sayısına bağlı olarak) bir düğüme sığabilecek maksimum kayıt sayısı ise, $m \leq M/2$ bir düğümdeki minimum kayıt sayısıdır. Bir R-Ağaç aşağıdaki özellikleri taşımalıdır:

- Kök olmayan her yaprak düğümü m ve M arasında indeks kaydı tutabilir.
- Yapraktaki her bir indeks kaydı (I, yaprak-belirteci) için, I tuple tarafından temsil edilen n-boyutlu veri nesnesini içeren en küçük dikdörtgendir.
- Kökte olmayan, her yaprak olmayan düğümün m ve M arasında çocuğu olabilir.
- Yaprak olmayan düğümdeki her bir kayıt (I, çocuk-işaretçi) için, I çocuk düğümü içeren en küçük dikdörtgendir.

- Kök düğüm yaprak düğüm olmadıkça en az iki çocuğa sahip olmalıdır.

3.3.2.1.1. M ve m parametrelerinin ayarlanması

M ve m'in ayarlanması veritabanının verimi için çok önemlidir. M verinin saklanacağı sabit diskin özelliklerinden, örneğin disk sayfa boyutu ve kapasite, çıkarılan parametrelere göre hesaplanır.

m ise veritabanı performansı için temel öğedir. Eğer veritabanı sadece arama sorguları için gerekliyse ve az güncelleme olacaksa, yüksek bir m değeri tavsiye edilir. Böylece R-Ağacın yüksekliği düşük tutulur ve arama performansı artar. Fakat bu taşma ve aşağı taşma riskini arttırır.

Diğer durumda m düşük bir değer olarak ayarlanır. Böylece veritabanı sık güncelleme ve değişikliklere karşı iyi durumda olur.

3.3.2.1.2 R-Ağaç ve İkincil Bellek Kullanılması

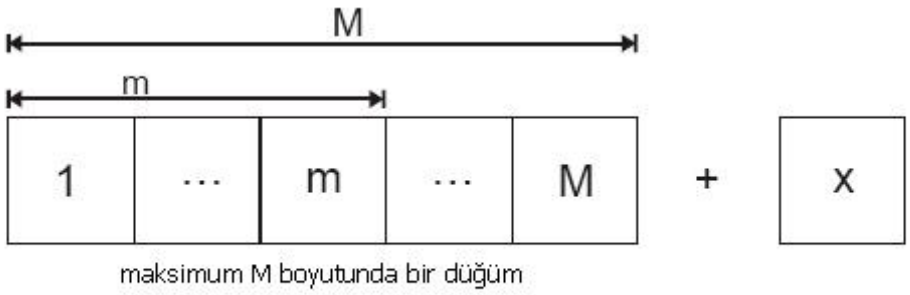
Veri tabanı dosyaları disk üzerinde bulunurlar ve sayfa denilen yapılardan oluşurlar. Bir sayfa 8 Kbyte'lık bir hafıza bloğudur. Çeşitli ek bilgiler için ayrılanların dışında sadece 8060 byte'lık alanında veri saklayabilir. Buna satırların nerede başlayıp bittiğini tutan satır ofset bilgileri de dahildir.

R-Ağaç ikincil bellek kullanımı için çok uygun bir yöntemdir. Çünkü R-Ağaç'ın her bir düğümü ayrı disk sayfasında saklanmaktadır. Bu da indeks yapısı belleğe sığmayacak kadar büyük veri tabanlarında R-Ağaç yönteminin kullanılmasını uygun kılmaktadır.

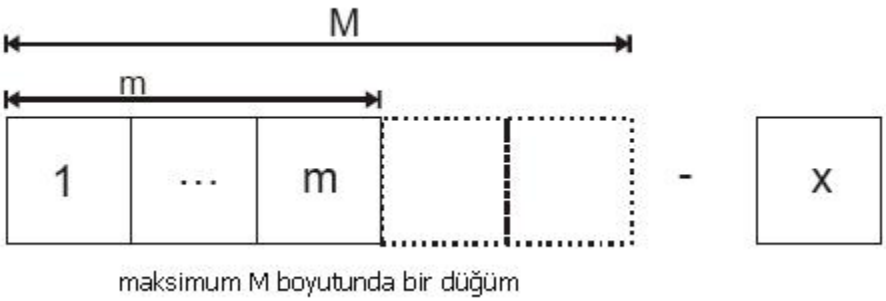
3.3.2.1.3. Taşma ve aşağı taşma

Eğer m çok yüksek ayarlanırsa (M'e yakın) düğüm çok yoğun dolar. Şekil 3.4'te görüldüğü gibi eğer bir ya da daha fazla kayıt bu düğüme yazılırsa maksimum kayıt sayısı M geçilir ve düğüm taşar.

Benzer şekilde m ve M arası çok geniş ayarlanırsa bir ya da daha fazla kaydın silinmesi durumunda şekil 3.5'te görüldüğü gibi kayıt sayısı m'in altına düşebilir.



Şekil3.4. Düğüm taşması



Şekil 3.5. Düğüm aşağı taşması

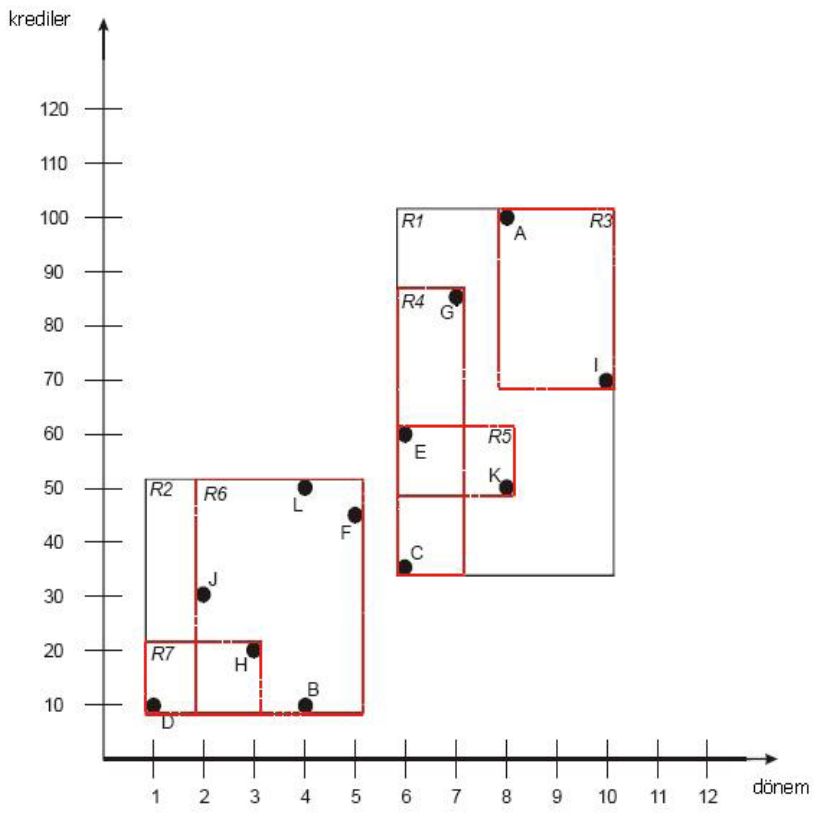
3.3.2.2. Algoritmalar

Guttman R-Ağaç için temel algoritmaları geliştirmiştir [Guttman,1984]. R-Ağacın bu metotları B-Ağaçtakilerin benzerleridir; sadece taşma ve aşağı taşma yönetimi verinin uzaysal yeri yüzünden farklıdır.

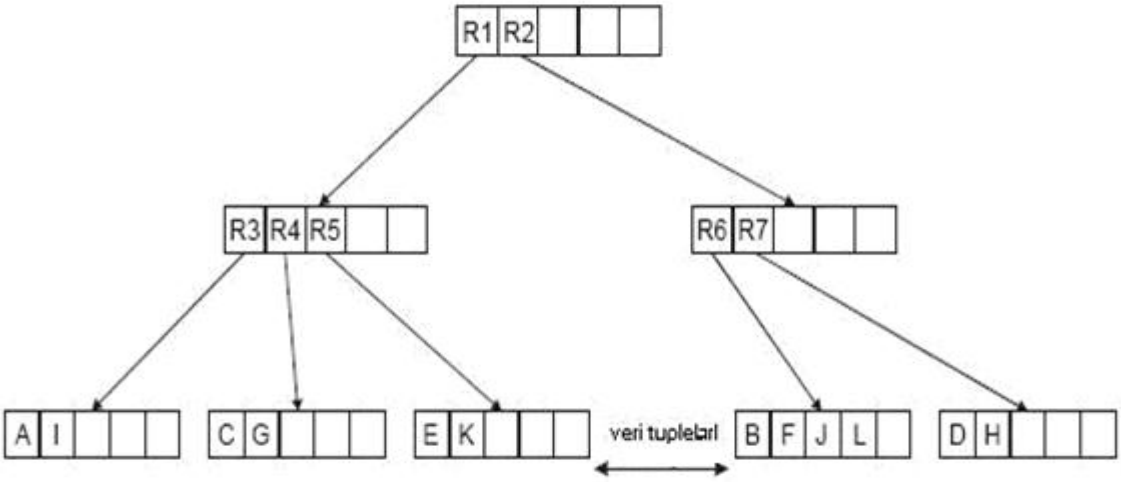
Devam eden kısımda farklı algoritmaları açıklamak amacıyla 2 boyutlu örnek bir veritabanı kullanılacaktır. Bu örnekte m=2 ve M=5 olarak seçilmiştir.

ad	dönem	kredi
A	8	100
B	4	10
C	6	35
D	1	10
E	6	40
F	5	45
G	7	85
H	3	20
I	10	70
J	2	30
K	8	50
L	4	50

Şekil 3.6. Örnek veri seti



Şekil 3.7. Örnek veri seti grafik gösterimi



Şekil 3.8. R ağaç yapısında örnek veri seti

3.3.2.2.1. Arama

R-Ağaçta arama B-Ağaçtaki arama gibi çalışır, ağaç kökten itibaren aşağı doğru gezilir. B-Ağaçtan farklı olarak aranması gereken birkaç dikdörtgen üst üste binmiş olabilir. (Şekil 2.7) Bütün bu alt dikdörtgenler ziyaret edilmek zorunda olduğundan hiçbir iyi en kötü durum performansı garanti edilemez.

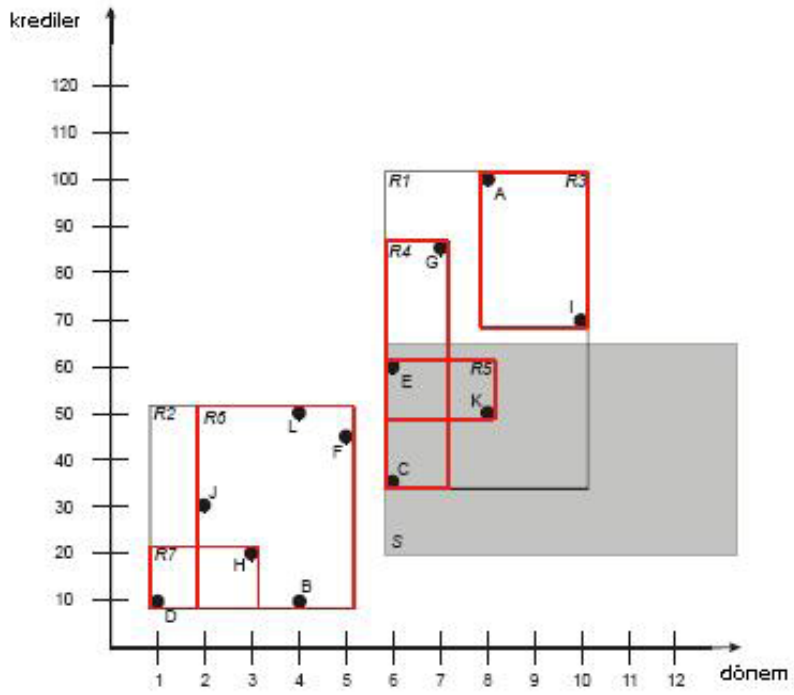
Algoritma: *Arama*

T, R-Ağacın kökü olsun. Dikdörtgenleri arama dikdörtgeni *S* ile üst üste çakışan bütün indeks kayıtları aranır.

• Eğer T bir yaprak değilse S ile üst üste çakışan ve kökü *çocuk-işaretçi* tarafından işaret edilen her çocuğa ***Arama*** yı uygula.

• Eğer T bir yapraksa, S ile üst üste çakışan bütün kayıtları sonuç kümesi olarak döndür.

Örnek veri tabanında *arama* için bir örnek:



Şekil 3.9. Örnek veri seti grafiksel gösterimi üzerinde S arama dikdörtgeni

Bu örnekte 6. dönemde veya daha üst dönemde ve 20 ile 65 arası kredi kazanan bütün öğrenciler bulunmaya çalışılıyor.

R1 sorgu dikdörtgeni S ile üst üste biniyor, R2 ile S arasında üst üste binme olmuyor. Bu yüzden R1 aranmalıdır. Bir sonraki adımda R4 ve R5 S ile üst üste biniyor. Bu dikdörtgenlerin içinde sonuç kayıtlar bulunmaktadır. R4'ten C ve R5'ten E ve K. Sonuç kümesi {C, E, K}.

Nokta verisi için arama aynı şekilde yapılır, sadece sorgu dikdörtgeni sadece bir sorgu noktasıdır. (örneğin (6,35) C'yi sonuç olarak döndürür.)

3.3.2.2.2. Ekleme

Eğer veri tabanına yeni bir kayıt eklemek gerekirse, R-Ağaca yeni bir indeks kaydı eklenmelidir. Bu aynı zamanda R-Ağacın yüksekliğini arttıracak tek faktördür. Yani bir düğüm taşması varsa, düğüm bölünmek zorundadır. Bu durumda bölünme köke ulaşır ve yükseklik artar.

Algoritma: *Ekleme*

- E yeni bir kayıt olsun

- E'nin yerleşmesi gereken yaprak düğüm L'yi bulmak için **YaprakSeçme** algoritmasını kullan.
- Eğer L'de yeteri kadar yer varsa, E'yi ekle. Yoksa **DüğümBölme** algoritmasını
  L'ye uygula. Bu algoritma L ve L' olmak üzere E'yi ve L'nin önceki tüm kayıtlarını içeren iki düğüm döndürür.
- L üzerinde **AğaçDüzeltme** algoritmasını uygula. Eğer daha önce bir bölünme varsa bunu L' üzerinde de uygula.
- Eğer bölünme köke ulaşırsa ve kök bölünmek zorundaysa, çocukları kökün bölümünden çıkan iki düğüm olan yeni bir kök yarat.

Algoritma: *YaprakSeçme*

Yeni kayıt E için uygun bir yaprak düğüm seç.

N kök düğüm olsun.

- Eğer N yapraksa, onu döndür.
- N yaprak değilse, kayıt F_k'yı N içinde bul. Bu kayıt E'yi düğüme eklemek için dikdörtgeni en az genişlemeye maruz kalacak kayıttır. Birden fazla Fk bulunması durumunda, en küçük olanı seç.
- Bir yaprağa ulaşana kadar **YaprakSeçme**'yi F_k 'ya uygula.

Algoritma: *AğaçDüzeltme*

Yaprak düğüm L'den köke kadar tırman. Tırmanırken dikdörtgenleri ayarla ve gerekli düğüm bölünmelerini gerçekleştir.

N=L yap. Eğer L daha önceden bölünmüşse N'=L' yap.

- Eğer N kökse, sonlandır.
- P N'nin ebeveyni olsun. P'nin içinde N'nin kaydını düzenle. Böylece P N'nin içerdiği bütün dikdörtgenleri içerebilsin.
- Eğer bir bölünme meydana gelmişse P'ye N' gösteren yeni bir kayıt ekle. Eğer ebeveyn düğüm taşarsa **DüğümBölme** algoritmasını kullan.

Ekleme için bir örnek:

(Q,10,65) şeklinde yeni bir öğrenci eklensin. *YaprakSeçme* R1'i ilk yeni düğüm olarak döndürür. Sonradan R3 kaydın ekleneceği dikdörtgen olarak seçilir ve burada bir taşma olmaz. Böylece Q R3'e eklenmiş olur. Bundan sonra *AğaçDüzeltme* R1 ve R3'ün dikdörtgenlerini günceller.

3.3.2.2.3. Silme

Eğer bir nesne veritabanından silinmek zorundaysa, kayıt E için ilgili indeksi bulup silmeniz gerekir.

Bu ayrıca R-Ağacın yüksekliğini azaltmanın tek yoludur.

Algoritma: *Silme*

- Kayıt E'yi içeren yaprak L'yi bulmak için ***YaprakBulma*** algoritmasını uygula. E bulunmazsa algoritmayı sonlandır.
- L'den E'yi çıkar.
- Dolu olmayan düğümleri yoğunlaştırmak için ***AğaçYoğunlaştır*** algoritmasını kullan.
- Eğer ayarlamalardan sonra kök sadece tek çocuğa sahipse, çocuğu yeni kök yap. (Ağacın yüksekliği azalır)

Algoritma: *YaprakBulma*

Kayıt E'yi içeren yaprak düğümü bul. T kök olsun,

- Eğer T yaprak değilse, dikdörtgenleri E ile üst üste binen bütün çocuklara ***YaprakBulma***'yı uygula. Eğer E bulunursa onu döndür.
- Eğer T bir yapraksa, her kaydı E ile karşılaştır ve eşleşirse T'yi döndür.

Algoritma: *AğaçYoğunlaştır*

Bu algoritma kaydın silindiği L yaprak düğümünü alır ve eğer yaprak m'den az kayda sahipse bu düğümü kaldırır.

Algoritma ağaç boyunca yukarıya doğru ilerler ve bütün dikdörtgenleri ayarlar (eğer gerekirse onları küçültür)

N=L olsun ve Q kaldırılan düğümlerin boş bir kümesi olsun.

- Eğer N kökse son adıma git. Diğer hallerde P'yi N'in ebeveyni olarak ele al.

- Eğer N m kayıttan daha az kayda sahipse (aşağı taşma), P'deki N'nin kayıtlarını kaldır ve onları Q'ya ekle.
- Eğer N'de bir aşağı taşma yoksa, dikdörtgenini (MBR) N'in diğer tüm kayıtlarını da içerecek şekilde ayarla.
- N=P yap ve ilk adıma dön.
- Q'daki bütün yaprakları *ekleme* kullanarak ağaca tekrar ekle. Q'da saklanan bütün yaprak olmayan düğümler ağacın daha üst düzeylerine eklenmelidir böylece ağaç yükseklik bakımından dengeli kalabilir.

Guttman şuna işaret etmiştir ki, prosedürlerin çoğu B-Ağaç'takiyle aynıdır. Fakat B-Ağaç'tan farklı olarak birleştirilen düğümler yerine R-Ağaç'ta tekrar ekleme vardır.

Silme için bir örnek:
Öğrenci K veritabanından silinmek istenirse, *YaprakBulma* algoritması uygulanır ve bunun sonucunda R5 dikdörtgeni elde edilir. Kayıt K R5'ten kaldırılır ve bundan sonra R5 aşağı taşar. Ardından R5'e *AğaçYoğunlaştır* algoritması uygulanır. Bu algoritma R5'i R-Ağaç'tan kaldırır, E'yi eklemeyi kullanrak R4'e ekler ve R1'in dikdörtgenini günceller.

3.3.2.2.3. Bir Düğümü Bölme

Dolu bir düğüme yeni bir kayıt eklenirken, M+1 kaydı iki düğüme bölmek şarttır. Düğümlerin bölümünde dikkate alınacak kısım bölümden sonra oluşan iki düğümün de dikdörtgenlerinin büyüklüğünü minimize etmektir. Çünkü bu dikdörtgenler küçük olurlarsa arama durumunda sadece gerekli düğümlerin ziyaret edilme olasılığı fazla olur. Bunun nedeni daha küçük dikdörtgenlerin diğer dikdörtgenlerle üst üste binme olasılığının daha az olmasıdır.
Guttman tarafından önerilen üç tane düğüm bölme algoritması vardır:

3.3.2.2.3.1. Ayrıntılı bölme algoritması

Bu algoritma bütün olası gruplamaları dener ve en iyisini seçer. En kalitelisidir fakat 2M+1 ihtimal yüzünden CPU kullanımı üssel olarak artar. Bu yüzden M'in çok yüksek olduğu (M>50) büyük veri tabanları için uygun değildir.

3.3.2.2.3.2. İkinci dereceden yük algoritması

Bu algoritma birbirine uzaklığı en fazla olan iki kaydı seçer ve bunları farklı düğümlere koyar. Kalan bütün elemanlar aynı şekilde düğümlere dağıtılır. Her bir düğüm için alan artışı hesaplanır. Her bir düğüm için bu düğümlerin N ve N' 'e eklenmesi için gerekli alanlar hesaplanır ve daha sonra bu iki grup arasında en büyük farkı olan düğüm başkasına eklendiğinde daha az alan gerektiren düğüme eklenir.

İkinci dereceden yük algoritması en iyi bölünmeleri sağlamaz fakat 50 kayıttan fazlasını içeren büyük veri tabanları için daha etkilidir.

Bu algoritma için kullanılan diğer iki yöntem:

KaynakSeçme: Grupların ilk iki elemanını seçer.

SonrakiniSeçme: Gruba koyulacak sonraki elemanı seçer.

3.3.2.2.3.3. Doğrusal yük algoritması

Bu algoritma her bir boyut için birbirinden en uzak iki elemanı seçer ve bunları farklı düğümlere koyar. Kalan elemanlar rastgele dağıtılır. Bu yöntemin ikindi dereceden yük algoritmasından farkı sınırlandırılmış *KaynakSeçme* ve *SonrakiniSeçme* yöntemleridir.

Bu algoritma çok hızlıdır ancak arama performansı diğer yöntemlere göre daha kötüdür.

3.3.2.3. R Ağacın türevleri

Orijinal R-Ağacın geliştirilmesinden sonra orijinal yapı ve algoritmalara aşağıdaki bazı geliştirme ve özel yetenekler eklenmiştir.

Paketlenmiş R-Ağaç (1985): Ağacın indeks yapısı tarafından kullanılmayan alan serbest bırakılır. Genelde ekleme ve çıkarılma yapılmayacak veritabanlarında kullanılır.

R+-Ağaç(1987): Üst üste binen alanları engellemeye çalışan yeni bir yöntemdir. Disk kullanımı daha yüksek maliyetle gerçekleşir fakat daha hızlı arama yapar.

R-Ağaç (1990):* Yapı R-Ağaçla aynıdır. R-Ağaçtan farkı bazı geliştirilmiş ekleme ve bölünme yöntemleridir.

X-Ağaç (1996): Düğüm boyutunun değişimi ile üst üste binen alanlar engellenir

3.3.2.3.1. R* Ağaç

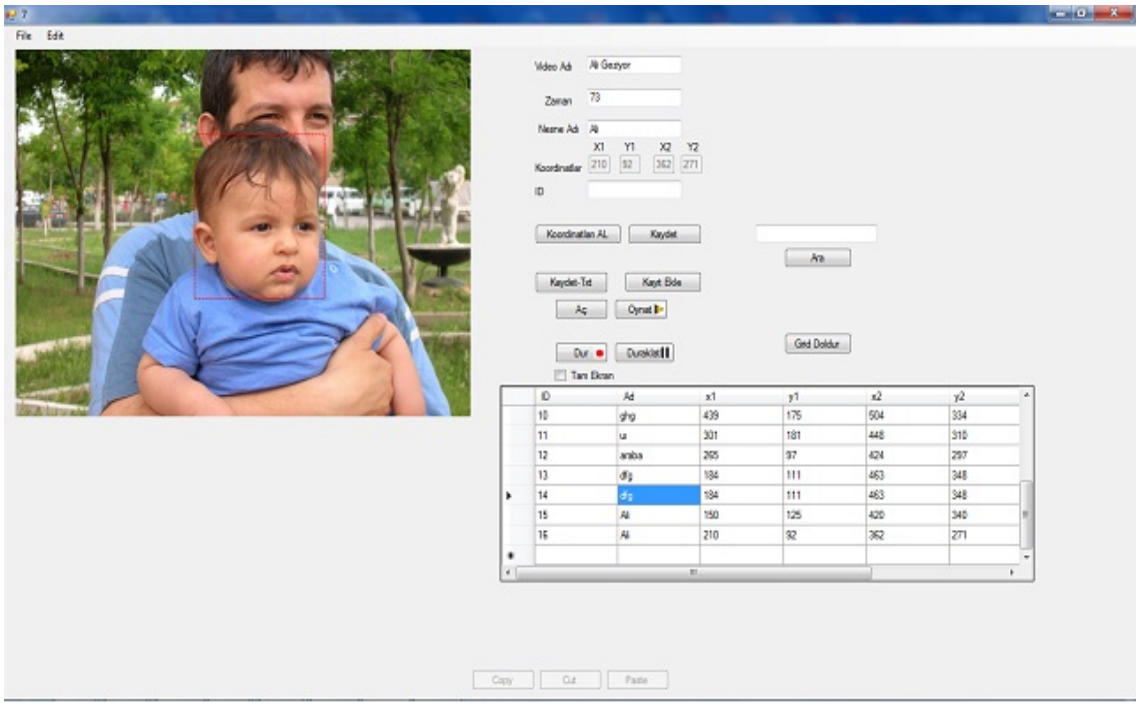
R Ağaçta optimizasyon kriteri iç düğümlerdeki çevreleyen dikdörtgenlerin en az alan kaplamasıydı. Fakat R Ağaçtan sonra şu sorular soruldu: Neden bu kriter kenar uzunluğunu kısaltmak ya da üst üste binen dikdörtgenlerin alanını düşürmek değildi? Neden bellek kullanımını optimize etmek değildi? Neden bunların hepsini yapmaya çalışmıyordu? Bu sorudan yola çıkan Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider ve Bernhard Seeger adlı araştırmacılar 1990'da yeni bir yöntem olarak R* Ağacı sundular [Beckmann vd., 1990]. R* Ağaç diğer R Ağaç türevlerinden belirgin bir biçimde iyiydi. Ayrıca hem nokta veri hem de uzaysal veriyi tutmada başarılıydı.

4. VİDEO DOSYALARININ VERİ SETİNE DÖNÜŞTÜRÜLMESİ

4.1. Nesnelerin Çıkarımı

Hareketli Görüntülerdeki Nesnelerin tespiti ve takibine dair kullanılan yöntemlere ikinci bölümde değinilmiş ve bir kısmı incelenmiştir. Video görüntüleri, coğrafik bilgi sistemleri, taşıt tanıma sistemleri gibi hareketli görüntüler üzerinde işlem yapabilmek için değişik metotlar bulunmaktadır. Hareketli görüntüler üzerinde çalışırken bu görüntüler içinde nesnelerin tespit edilmesi için bu yöntemlerden birisi kullanılır. Bir video dosyası üzerinde nesneleri tespit etmek ve bulmak için yarı otomatik bir uygulama yazılımı geliştirilmiştir.

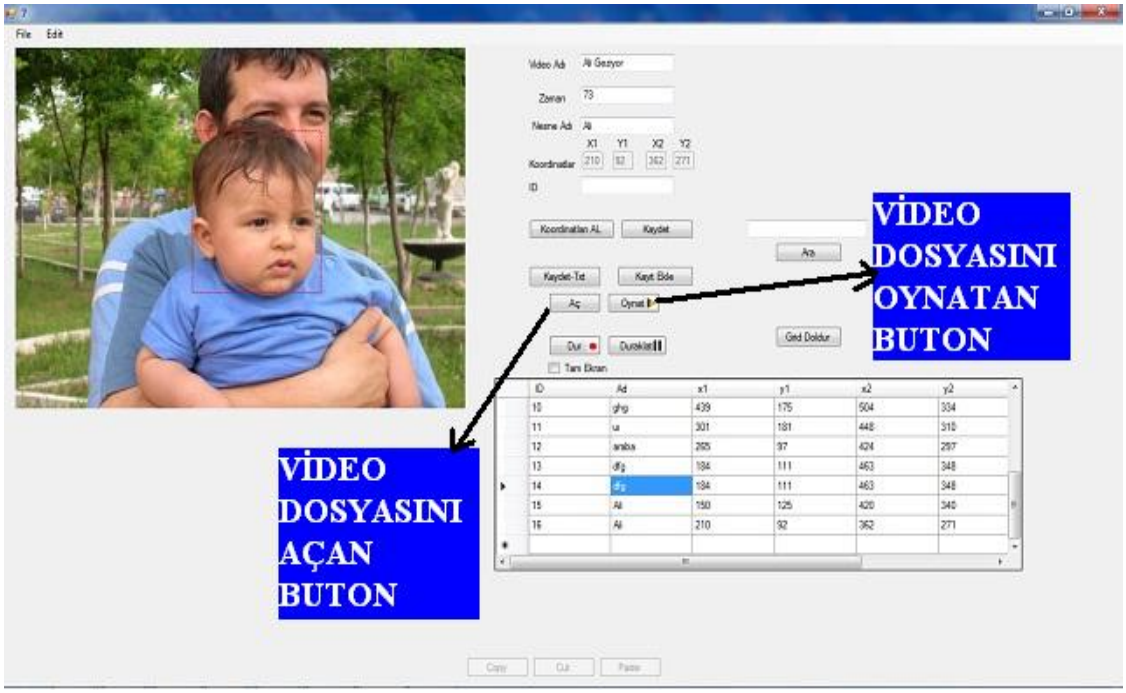
Şekil 4.1'de uygulama yazılımı ekran görüntüsü görülmektedir. Uygulama yazılımında video dosyası açılıp oynatılarak, istenildiğinde video dosyası durdurularak o anki ekran görüntüsünün fare imleci yardımıyla bir bölgesinin seçilerek o bölgenin o anki ekran üzerindeki konum koordinatlarını ve video dosyasındaki zaman ile birlikte kaydedilmesini sağlamaktadır.



Şekil 4.1. Program ekran görüntüsü

Uygulama programı çalıştırıldığında ilk önce açılmak istenen video dosyası seçilmektedir. Bunun için Şekil 4.2'de görülen "*Aç*" düğmesi kullanılmaktadır. .NET ortamında C# programlama dili ile geliştirilen uygulamada "*Aç*" düğmesi aracılığıyla video dosyasını açarak, "*PictureBox*" nesnesinin "*Image*" özelliğini kullanarak video dosyası oynatılacak hale getirilmektedir.

Video dosası açılıp oynatılacak hale geldikten sonra "*Oynat*" düğmesi aracılığıyla da açtığımız video dosyası oynatılabilmektedir.



Şekil 4.2. Video Dosyasını Açıp Oynatma

Video dosyasını seçip, açtıktan ve oynattıktan sonra video görüntülerinin istediğimiz karesini alabilmek için *"Durdur"* butonu kullanılmaktadır. Oynayan görüntü üzerinde istediğimiz bir sahnede *"Durdur"* ile görüntüyü durdurup seçme ve kaydetme işlemi yapılabilmekte ve *"Oynat"* butonu ile videonun kaldığı yerden tekrar oynatılarak sonraki sahnelerin izlenmesi ve gerektiğinde durdur, seç ve kayıt işlemleri tekrar yapılabilmektedir.

Başka bir video dosyası üzerinde çalışmak istenildiğinde bu sefer "*Dur*" butonu yardımıyla video dosyası sonlandırılıp başka bir video dosyası "*Aç*" ile tekrar açılabilmektedir.

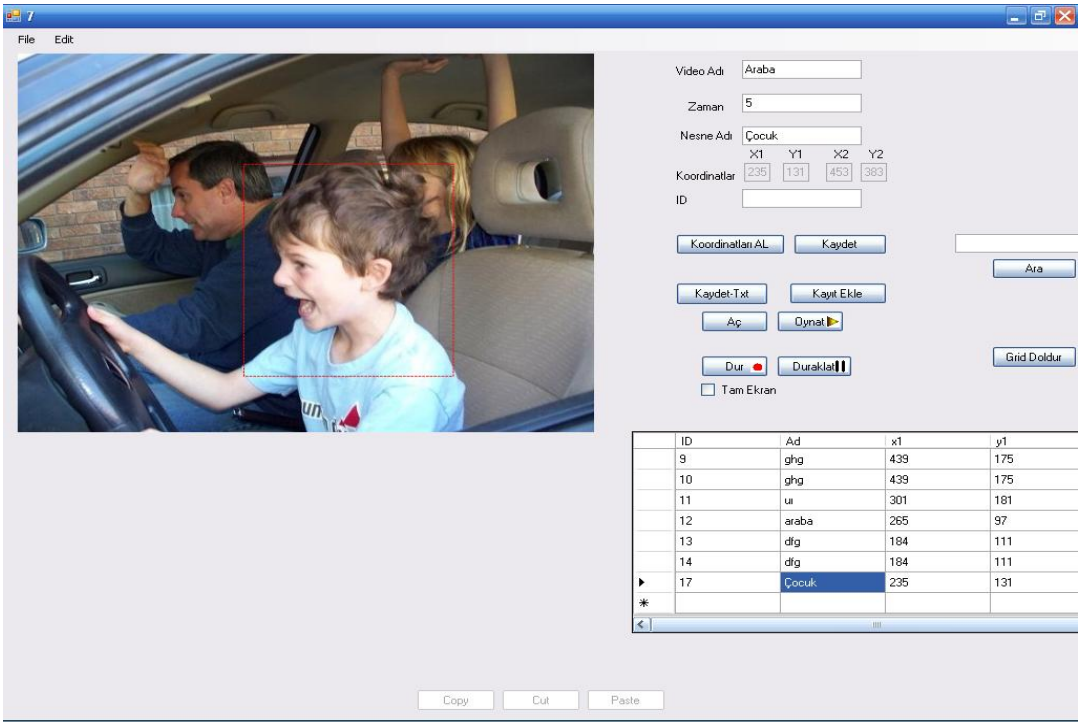
Video dosyalarındaki istenilen kareler yakalanarak bu karelerdeki nesneler koordinat ve zamanları ile birlikte veri tabanına kaydedilmektedir.

Nesne çıkarımını yaparken fare imleciyle seçilen nesnenin koordinatlarını, videodaki zamanını, video adını veri tabanında hazırladığımız tablolara uygulama programı ile kaydedilmektedir. "*Goruntu_Indeks.mdb*" veri tabanı dosyasında oluşturulan;

- Indeks
- Nesneler
- Bolgeler
- Vgs

isimli tablolar mevcuttur.

"*Indeks*" isimli tabloda "ID, Ad, x1, y1, x2, y2, zaman, VGS_ID" alanları oluşturulmuş durumdadır. Uygulama programı açılıp videodan tespit edilen nesneler Şekil 4.3'te görüldüğü gibi seçili alan kırmızı kesik çizgili dikdörtgen içine alınmaktadır. Bu bölge, koordinatları, zaman ve nesne adı ile birlikte "*Kayıt Ekle*" butonuna tıklandığında "*Goruntu_Indeks.mdb*" dosyasındaki "*Indeks*" tablosundaki "*ID, Ad, x1, y1, x2, y2, zaman*" alanlarına kaydedilmektedir.



Şekil 4.3. Seçilen nesnenin kaydedilmesi

4.2. İndeksleme Yöntemi

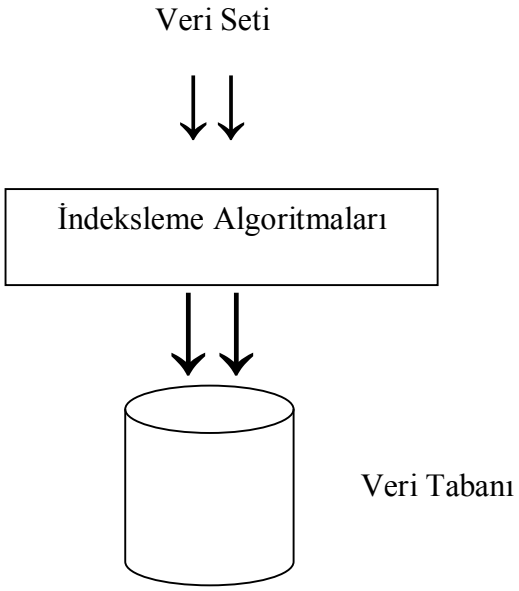
İndeksleme yapısı olarak çok boyutlu veri yapılarında en çok kullanılan, oldukça başarılı olan ve birçok indeks yapısına temel teşkil eden R-Ağaç yapısı esas alınarak veriler kaydedilmektedir.

Çizelge 4.1. Örnek bir veri tablosu

ID	Adı	X1	Y1	X2	Y2	Zaman	Video_ID
1	Çocuk1	239	131	324	408	12	1
2	Araba	102	85	655	567	15	1
3	Baba	232	123	345	543	18	1
4	Çocuk2	290	156	45	345	21	1
5	Direksiyon	234	433	434	344	25	1
6	Ali	290	678	345	543	12	2
7	Oyuncak	223	256	285	277	5	2
8	Ağaç	322	345	489	680	9	2
9	Pencere	156	9	345	98	22	2
10	Bayrak	34	43	67	356	1	3
11	Kapı	315	510	345	450	5	2
12	Top	332	234	380	266	8	3
13	Masa	456	34	680	211	22	3
14	Sandalye	390	24	430	56	25	3
15	Tahta	78	55	325	246	29	3
16	Bilgisayar	88	180	276	332	32	3
17	Dolap	66	90	268	356	38	3
18	Telefon	110	70	164	90	38	3
19	Priz	200	80	242	110	38	3
20	Tahta	400	100	634	345	38	3
21	Kalem	580	100	620	110	38	3
22	Saksı	34	56	80	148	46	3
23	Otobüs	45	67	577	354	2	4

24	Kamyon	58	86	432	324	5	4
25	Araba	22	33	245	432	9	4
26	Ambulans	32	90	321	345	15	4
27	İtfaiye	110	45	325	234	35	4
28	Trafik ışık	233	256	290	689	42	4
29	Polis	249	267	290	435	46	4
30	Bisiklet	34	65	136	123	48	4
31	Lastik	34	65	78	120	48	4
32	Traktör	432	68	543	145	55	4
33	Kitap	234	325	270	421	1	5
34	Defter	187	74	277	168	1	5
35	Silgi	155	66	198	86	1	5
36	Kağıt	23	34	145	154	1	5
37	Cetvel	67	90	236	123	1	5
38	Kalemlik	45	64	234	456	1	5
39	Kalemtraş	233	322	267	355	1	5
40	Pergel	47	34	135	232	2	5

Yukarıda Çizelge 4.1'de veriler R-Ağaç yapısı kullanılarak veri tabanına kaydedilmektedir.



Şekil 4.4. İndeksleme işlemi

Çizelge 4.2. Bölgeler Tablosu

ID	0
ID_ADI	0
ALT BÖLGE1ID	-
ALTBÖLGE2 ID	-
BOLGE MINX	-
BOLGE MINY	-
BOLGE MAXX	-
BOLGE MAXY	-
BOLGE MINT	-
BOLGE MAXT	-
BOLGE ID MIN	-
BOLGE ID MAX	-
YAPRAK MI?	0

Veriler "*Indeks*" tablosuna kaydedildikçe indeks yapısını tutan "*Bolgeler*" isimli tablonun da güncellenmesi yapılmaktadır.

4.3. Kullanılacak Nesne Çıkarım Yönteminin Belirlenmesi

Bu tez çalışmasında nesnelerin çıkarımı işlemi için operatör yardımı alınması önerilmektedir. Bir video dosyasındaki nesneleri, operatör tarafından belirlenmekte ve isimlendirilmektedir. Operatöre kolaylık sağlaması için bir video dosyası oynatmaya yarayan ve geçici olarak durdurmaya yarayan tuşlar eklenmiştir. Video oynatılarak indekslenmek istenen bir karede bulunan nesneler seçilir, kaydet tuşuna basılır. Geliştirilen program seçilen nesneyi, o anki video karesindeki konumu ve zamanını alarak veri tabanına kaydeder.

4.4. Kullanılacak Erişim Yönteminin Belirlenmesi

Veri Tabanları büyük veri yığınlarını saklamak ve bunlara erişimi hedef almaktadır. Hedef bilgi kümesini saklamak ve buna hızlı erişmek bir takım sorunları da beraberinde getirmektedir. Bunlardan en önde geleni veri tabanının ihtiyaç duyulan bölümü için ana hafızanın yetersiz kalmasıdır. Bu yüzden ikincil hafızaya erişim süresi sorgulama başarımı ve performansı açısından önemli hale gelmektedir. Erişim yöntemleri ikincil hafızada yer alan bilgiye erişim yükünü azaltmak için tasarlanmış

yapılardır. Genellendiğinde, erişim yöntemleri indeksleme veya hesaba dayalı adresleme yöntemleri ile gerçekleştirilmektedir. Yapılan çalışmalar incelendiğinde bu iki yöntem en sık başvurulan erişim yöntemi olarak karşımıza çıkmaktadır.

Geleneksel ve tek boyutlu bilginin saklandığı veri tabanlarındaki indisleme ihtiyacı düşünüldüğünde, hareketli nesne veri tabanlarında veri yapılarının ne kadar kritik olduğu daha iyi anlaşılabilir. Böyle bir veri tabanının dinamik yapısı, indisleme ve diğer veri tabanı ana modüllerinin bu dinamik ortama ayak uydurmasını gerekli kılmaktadır. Son zamanlarda popüler olmaya başlayan bir araştırma konusu olan hareketli veri tabanları, veri tabanı alanında birçok yeniliğe zemin hazırlamaktadır. Örnek olarak büyük bir şehirdeki tüm araçların konumlarının hareketli nesne veri tabanında tutulduğu durumda; veri tabanındaki zaman-uzlamsal bir sorgu için bütün nesnelerin konumlarının test edilmesi gerçeklenebilir bir çözüm değildir. Bununla birlikte indis olarak geleneksel veya uzlamsal indis yapılarını kullanmak da bir çözüm olmayacaktır. Çünkü belirli bir model üzerine konum bilgisi yenilenen nesneler için indisin devamlı güncellenmesi ve güncelleme işleminin maliyetinin yüksek olması bu yaklaşımı yetersiz kılmaktadır. O zaman dinamik ortama uyumluluk, hareketli nesnelere ait indislerden beklenen temel özelliklerin başında gelmektedir.

Bir indis veri yapısından beklenen şey, belirli bir arama anahtarı üzerinden dosyadaki kayıt veya kayıtlara hızlı erişmeye olanak sağlamasıdır. Hızlı erişimden kastedilen, disk tabanlı bir indis yapısı için disk erişim sayısıdır; ana hafıza tabanlı bir indis yapısı için ise çalışma zamanıdır. Bir sorgunun işlenmesinde indis üzerinden arama yapılması, veri tabanındaki tüm kayıtlara erişip arama yapılmasını engeller, bu da zaman açısından avantaj sağlamaktadır. Örneğin Bayer-McCreight'ın (1972) birlikte tasarladığı, dengeli ağaç yapısındaki B-Ağaç indisinin çalışma zamanı O ($\log_k N$)'dir. Bu çalışma zamanı ile kastedilen disk ortamında bulunan bir k-order bir B-Ağacı ile ağacın yüksekliği kadar adımda istenen kayıt ve kayıtları içeren disk sayfalarına erişilebilmesidir. N indislenen eleman sayısı olmak üzere ağacın yüksekliği $\log_k N$'dir ve ağaç her seviyede vereceği kararda bundan sonraki alt ağacın 1/K'lık kısmı ile ilgilenmektedir.

B-Ağaç gibi ağaç tabanlı indis yapıları zamansal veya uzlamsal veri yapıları için indis olarak doğrudan kullanılmasa bile bu gibi çok boyutlu veri için güçlü indis yapılarına temel teşkil edebilmektedir. Örneğin, çok boyutlu uzlamsal erişim yöntemleri

(Spatial Access Methods, SAM) adı altında birçok geometrik indis yapıları tasarlanmıştır. Bunlardan Guttman'ın (1984) tasarladığı R-Ağaç, B-Ağacın çok boyutlu genelleştirilmiş bir versiyonu olarak düşünülebilir. Diğer yandan zamansal veri için daha çok " ayrık zamanda değişen" tipte veriler için indis yapıları tasarlanmıştır. Gerek uzlamsal gerek zamansal indis yapıları, sürekli zamanda zaman-uzlamsal indis yapıları için temel kaynaklardır.

5. R-AĞAÇ YAPISI İLE HAREKETLİ NESNELERİN İNDEKSLENMESİ

Oluşturulan veri seti R-Ağaç yapısı kullanılarak indekslenmektedir. Veriler, 6 boyuta sahip olduğundan *uzaklık formülü* (5.1), bölgenin sınırlarının hesaplamalarında ve değerlendirmelerde kullanılmaktadır.

$$\text{Uzaklık}=\sqrt{(x1-x2)^2+(y1-y2)^2+(t1-t2)^2+(Video_ID1-Video_ID2)^2} \quad (5.1.)$$

R-Ağaç yapısını uygulamak için 40 adet kayıttan oluşan örnek bir veri seti kullanılacaktır. Bu örnekte M değeri 3 olarak belirlenmiştir.

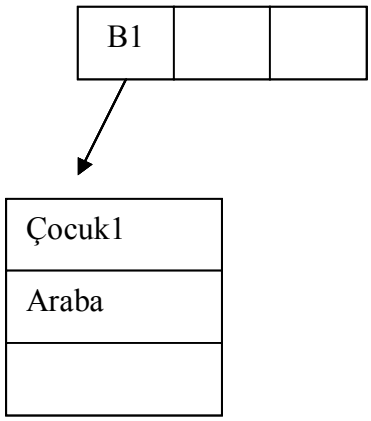
M=3 ise;

m=M/2 yani m=3/2=1,5 olduğundan m değerinin en az 2 olması gerekmektedir. Bu yüzden m=2 seçilmiştir.

1- Çocuk1 (239, 131, 324, 08, 12,1) verisi veri tabanına eklenmeden önce veri tabanı boş durumdadır. Kayıt sayısı *m=2*'den küçük olduğundan dolayı henüz bir bölge oluşmamıştır.

2- Araba (102, 85, 655, 567, 15, 1) verisi veri tabanına kaydedilirken kayıt sayısı *m*=2 değerine ulaşır. Bu durumda 1. Bölge ve kök düğüm oluşur. Bölge sınırları ve alt bölge sınırları kaydedilir.

Bölge isimleri B1, B2, B3….. şeklinde gösterilmektedir.



Şekil 5.1. Çocuk1 ve Araba verisinin eklenmesi

Araba eklendikten sonra bölge sınırları Çizelge 5.1'deki gibi olacaktır.

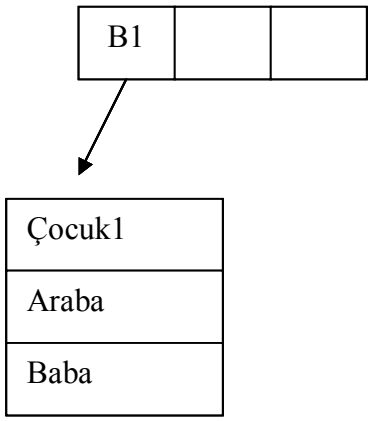
Çizelge 5.1 Bölge1 Sınırları

Alt Sınırlar	102, 85, 324, 408, 12, 1
Üst Sınırlar	239, 131, 655, 567, 15, 1

3- Baba(232, 123, 345, 543, 18, 1) verisi veri tabanına kaydedilirken öncelikle *Yaprak seç* algoritması çalışır. Kayıt eklenirken bir tane bölge bulunmaktadır. Bölgede kayıt işlemi için boş yer olup olmadığına bakılır. Bölgede boş yer olduğu için Bölge1 uygun olarak tespit edilmektedir. *Kayıt ekle* algoritması çalışır. Kayıt eklendikten sonra *Ağaç düzenle* algoritması çalışır bölge sınırları Çizelge 5.2'deki olur ve alt bölge bilgileri güncellenir.

Çizelge 5.2. Baba verisi eklendikten sonra Bölge1 sınırları

Alt Sınırlar	102, 85, 324, 408, 12, 1
Üst Sınırlar	239, 131, 655, 567, 18, 1



Şekil 5.2. Baba verisinin eklenmesi

4- Çocuk2 (290, 156, 45, 345, 21, 1) verisi veri tabanına kaydedilirken öncelikle *Yaprak seç* algoritması çalışır. Kayıt eklenirken bir tane bölge vardır. Bölgede kayıt için boş yer olup olmadığına bakılır. Bölge dolu olduğundan kayıt eklenebilmesi için bölgenin bölünmesi gerekir.

Kayıt sayısı = 3+1 >M olduğu için bölgenin bölünmesi gerekir. Bölge ikiye bölünür.

Elimizde 4 kayıt bulunmaktadır. 4 kayıt 2 ayrı bölgeye bölünmektedir. Bölünmenin uygun biçimde gerçekleşmesi için tüm bölünme olasılıkları göz önüne alınmalı. En uyun bölünme birbirine yakın olan kayıtların aynı bölgede bulunduğu durumdur. Bunun için verilerin birbirine olan uzaklıkları hesaplanır.

Uzaklık = $\sqrt{(Ust\ Sıurlar\ -\ Alt\ Sınırlar\ )^2}$

formülü ile elde edilen değerler karşılaştırılır.

Çocuk1 (239, 131, 324, 408, 12, 1)
Araba (102, 85, 655, 567, 15, 1)
Baba (232, 123, 345, 543, 18, 1)
Çocuk2 (290, 156, 45, 345,21, 1)

Uzaklık (Çocuk1-Araba) =

$$\sqrt{(239-102)^2+(131-85)^2+(324-655)^2+(408-567)^2+(12-15)^2(1-1)^2 * 100000}$$

Uzaklık (Çocuk1-Araba) $= \sqrt{18769+2116+109562+25281+9}$

$= \sqrt{155736}$

$= 394{,}634$

Uzaklık (Çocuk1-Baba) $= \sqrt{49+64+441+18225+36}$

$= \sqrt{18815}$

$= 137{,}167$

Uzaklık(Çocuk1-Çocuk2) $= \sqrt{2601+625+77841+3969+81}$

$= \sqrt{85036}$

$= 291{,}609$

Uzaklık (Araba-Baba) $= \sqrt{16900+1444+96100+576+9}$

$= \sqrt{115029}$

$= 339{,}159$

Uzaklık (Araba-Çocuk2) $= \sqrt{35344+5041+78400+49284+36}$

$= \sqrt{168105}$

$= 410{,}006$

Uzaklık (Baba-Çocuk2) $= \sqrt{3364+1089+90000+39204+9}$

$= \sqrt{133666}$

$= 365{,}603$

Çizelge 5.3. Bölünme olasılıklarının hesaplanması

Çocuk1	Araba	+	Çocuk1	Araba	= 760,237

Çocuk1	Baba	+	Araba	Çocuk2	= 547,173

Araba	Baba	+	Çocuk1	Çocuk2	= 630,768

547,173<630,768<760,237 olduğundan,
en küçük değer (Çocuk1,Baba), (Araba,Çocuk2) şeklindeki bölünmeden elde edileceğinden Bölge1'in yaprak düğümlerinde (Çocuk1, Baba), Bölge2'nin yaprak düğümlerinde (Çocuk2,Araba) verileri bulunur.

Ağaç düzenle algoritması çalışır. Bu algoritma bölge bilgilerini ve alt bölge sınırlarını günceller. Bölünmeden sonra bölge sınırları Çizelge 5.4'deki gibi olmaktadır.

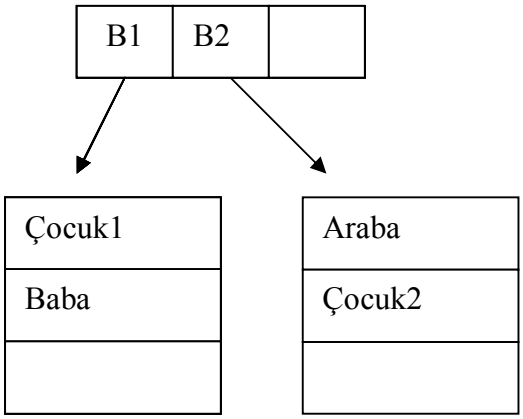
Çizelge 5.4. Çocuk2 verisi eklendikten sonra Bölge1 ve Bölge2 sınırları

<u>Bölge1:</u>

Alt Sınırlar	232, 123, 324, 408, 12, 1
Üst Sınırlar	239, 131, 345, 543, 18, 1

<u>Bölge2:</u>

Alt Sınırlar	102, 85, 45, 345, 15, 1
Üst Sınırlar	290, 156, 655, 567, 21, 1



Şekil 5.3. Bölge bölünmesinden sonra oluşan durum

5- Direksiyon (234, 433, 434, 344, 25, 1) verisi veri tabanına kaydedilirken öncelikle *Yaprak seç* algoritması çalışır. Verinin hangi yaprağa girmesi gerektiğine bakar. Bunun için kayıt eklendikten sonra bölgelerde olacak artışlara bakılır. Daha az artışın olacağı bölgeye kayıt eklenir.

Bölge1 ve Bölge2'ye eklenmesi halinde oluşabilecek yeni sınır değerleri çizelge 5.5'deki gibi olacaktır.

Çizelge 5.5. Direksiyon verisinin eklenmesi ile oluşabilecek sınır değerleri

Bölge1

Alt Sınırlar	232, 123, 324, 344, 12, 1
Üst Sınırlar	239, 433, 434, 543, 25, 1

Bölge2

Alt Sınırlar	102, 85, 45, 344, 15, 1
Üst Sınırlar	290, 433, 655, 567, 25, 1

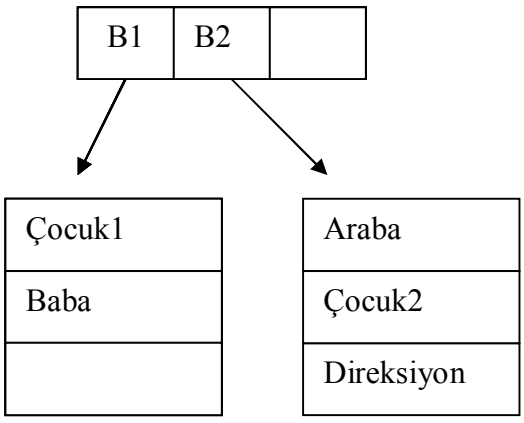
Bölge Adı	Eski Uzaklık	Yeni Uzaklık	Uzaklık Farkı
Bölge1	137,167	384,602	247,435
Bölge2	679,502	727,235	47,733

47,733<247,435 olduğu için, *Yaprak seç* çalışır. Bölge2'ye daha yakın olduğu için Direksiyon verimiz Bölge2'ye eklenmelidir. Bölge2'ye eklendikten sonra ağaç düzenle çalışarak bölge bilgilerini ve alt bölge bilgilerini günceller. Bölge2'nin sınırları çizelge 5.6'daki gibi olacaktır.

Çizelge 5.6. Direksiyon verisi eklendikten sonra oluşan Bölge2 sınırları

Bölge2

Alt Sınırlar	102, 85, 45, 344, 15, 1
Üst Sınırlar	290, 433, 655, 567, 25, 1



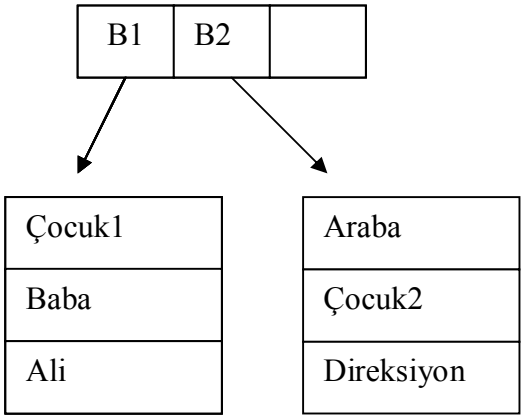
Şekil 5.4. Direksiyon verisi eklendikten sonra bölgelerin son durumu

6- Ali (290, 678, 345, 543,1 2, 2) verisi veri tabanına kaydedilirken öncelikle *Yaprak seç* algoritması çalışır. İki bölge var ve bunlardan Bölge2 dolu. Bölge1 boş ve uygun. Ali verisi Bölge1'e eklenir ve eklendikten sonra Bölge1 sınırları *Ağaç düzenle* algoritması çalışarak güncellenir.

Çizelge 5.7. Bölge1'in Ali verisi eklendikten sonraki sınırları

Bölge1

Alt Sınırlar	232, 123, 324, 344, 12, 1
Üst Sınırlar	290, 678, 434, 543, 25, 2



Şekil 5.5. Ali verisi eklendikten sonra bölgelerin durumu

7- Oyuncak (223, 256, 285, 277, 5, 2) verisi veri tabanına kaydedilirken öncelikle *Yaprak seç* algoritması çalışır. 2 bölge var ve bölgeler dolu. Bölge bölünmesi olacaktır. Oyuncak verisi eklendiğinde en az artış hangi bölgede olacak ise o bölge bölünecektir.

Bölge1 bölünür ise

Çocuk1 (239, 131, 324, 408, 12, 1)

Baba (232, 123, 345, 543, 18, 1)

Ali (290, 678, 345, 543, 12, 2)

Oyuncak (223, 256, 285, 277, 5, 2)

Uzaklık (Çocuk1-Baba) = 137,167

Uzaklık (Çocuk1- Ali) = 648,441

Uzaklık (Çocuk1-Oyuncak) = 366,828

Uzaklık (Baba-Ali) = 641,424

Uzaklık (Baba-Oyuncak) = 438,514

Uzaklık (Ali-Oyuncak) = 506,929

Çizelge 5.8. Oyuncak verisi eklenmesi durumunda oluşacak Bölge1 sınırları

Çocuk1	Baba	+	Ali	Oyuncak	= 644,096

Çocuk1	Baba	+	Araba	Çocuk2	= 1086,955

Araba	Baba	+	Çocuk1	Çocuk2	= 1008,252

Bölge2 bölünür ise

Araba (102, 85, 655, 567, 15, 1)

Çocuk2 (290, 156, 45, 345, 21, 1)

Direksiyon (234, 433, 434, 344, 25, 1)

Oyuncak (223, 256, 285, 277, 5, 2)

Uzaklık (Araba-Çocuk2) = 679,562

Uzaklık (Araba-Direksiyon) = 487,029

Uzaklık (Araba-Oyuncak) = 592,140

Uzaklık (Çocuk2-Direksiyon) = 480,835

Uzaklık (Çocuk2-Oyuncak) = 420,676

Uzaklık (Direksiyon-Oyuncak) = 398,170

Çizelge 5.9. Oyuncak verisi eklenmesi durumunda oluşacak Bölge2 sınırları

Araba	Çocuk2	+	Direksiyon	Oyuncak	= 1077,732

Araba	Direksiyon	+	Çocuk2	Oyuncak	= 907,705

Araba	Oyuncak	+	Çocuk2	Direksiyon	= 1072,975

Bölge1'in bölünmesi ile uzaklık farkı en az (Çocuk1-Baba)+(Ali-Oyuncak) = 644,096 şeklinde,

Bölge2'nin bölünmesi ile uzaklık farkı en az (Araba-Direksiyon)+(Çocuk2-Oyuncak) = 907,705 şeklinde,

gerçekleşeceğinden ve 644,096<907,705 olduğundan Bölge1 ikiye bölünerek Bölge1 ve Bölge3 oluşur. Bölge1'in içerisine (Çocuk1,Baba) verileri girerken, Bölge2 içerisine (Ali, Oyuncak) verileri girer.

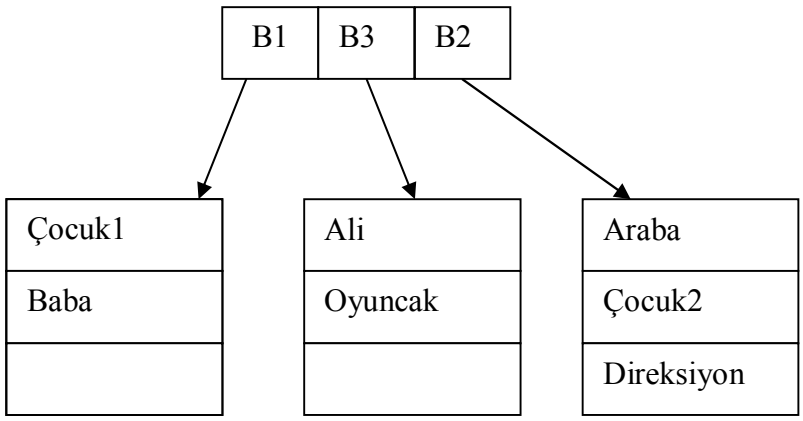
Çizelge 5.10. Oyuncak verisinin eklenmesi ile Bölge1 ve Bölge3'ün güncellenmiş sınır değerleri

Bölge1

Alt Sınırlar	232, 123, 324, 408, 12, 1
Üst Sınırlar	239, 131, 345, 543, 18, 1

Bölge3

Alt Sınırlar	223, 256, 285, 277, 5, 2
Üst Sınırlar	290, 678, 345, 543, 12, 2



Şekil 5.6. Oyuncak verisi ekledikten sonra bölgelerin durumu

8- Ağaç (322, 345, 489, 680, 9, 2) verisi veri tabanına kaydedilirken öncelikle *Yaprak seç* algoritması çalışır. Üç bölge var. Bölge2 dolu durumda. Bölge1 ve Bölge3 uygun. Verinin hangi bölgeye eklenmesi gerektiği yine uzaklık farkları bulunup karşılaştırılarak tespit edilir.

Bölge1 ve Bölge3'e eklenmesi halinde oluşabilecek yeni sınır değerleri çizelge 5.11'deki gibi olacaktır.

Çizelge 5.11. Ağaç verisi eklenmesi durumunda oluşacak sınırlar

Bölge1

Alt Sınırlar	232, 123, 324, 408, 9, 1
Üst Sınırlar	322, 345, 489, 680, 18, 2

Bölge3

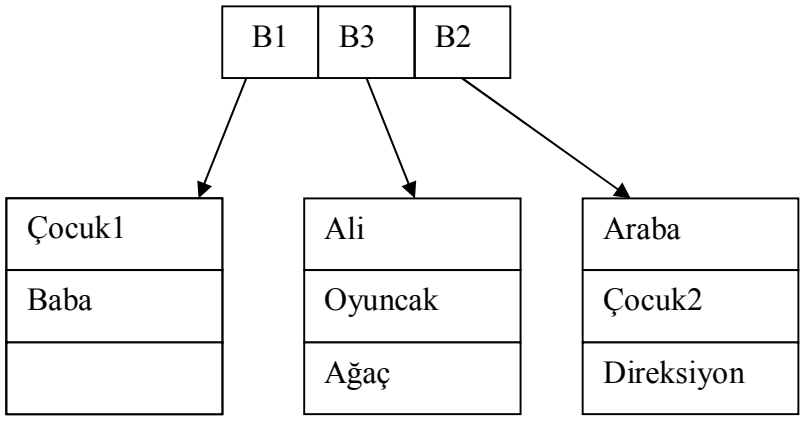
Alt Sınırlar	223, 256, 285, 277, 5, 2
Üst Sınırlar	322, 678, 489, 680, 12, 2

Bölge Adı	Eski Uzaklık	Yeni Uzaklık	Uzaklık Farkı
Bölge1	137,167	508,600	371,433
Bölge3	506,929	540,332	33,403

Ağaç verisi Bölge3'e eklendiğinde en az uzaklık farkı oluşacağından Bölge3'e eklenir. *Ağaç düzenle* algoritması çalışarak bölge ve alt bölgelerin sınırları güncellenir ve çizelge 5.12'deki durumunu alır.

Çizelge 5.12. Ağaç verisi eklendikten sonra Bölge3 sınırları

Alt Sınırlar	223, 256, 285, 277, 5, 2
Üst Sınırlar	322, 678, 489, 680, 12, 2



Şekil 5.7. Ağaç verisi eklendikten sonra bölge durumları

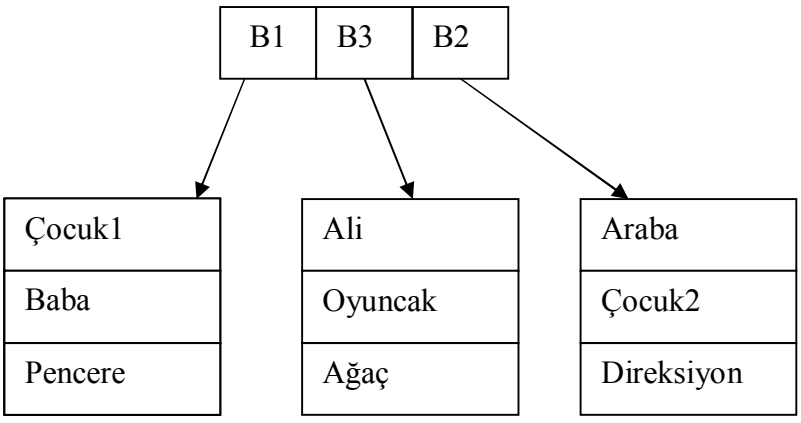
9- Pencere (156, 9, 345, 98, 22, 2) verisi veri tabanına kaydedilirken öncelikle *Yaprak seç* algoritması çalışır. Bölge2 ve Bölge3 dolu, Bölge1 uygun. Pencere verisi Bölge1'e eklenecektir. *Ağaç düzenle* algoritması çalışır ve bölge sınırları güncellenir.

Çizelge 5.13. Pencere verisi eklendikten sonra Bölge1 sınırları

Bölge1

Alt Sınırlar	156, 9, 324, 98, 12, 1
Üst Sınırlar	232, 131, 345, 543, 22, 2

En son ağaç yerleşimi aşağıdaki çizelgedeki gibi olur.



Şekil 5.8. Pencere verisi eklendikten sonra bölge durumları

10- Bayrak (34, 43, 67, 356, 1, 3) verisi veri tabanına kaydedilirken öncelikle *Yaprak seç* algoritması çalışır. Bölge1, Bölge2, Bölge3 olmak üzere şuanda tüm bölgeler dolu durumdalar. Bölge bölünmesi olacaktır.

Çizelge 5.14. Bölge sınır artışları

Bölge Adı	Eski Uzaklık	Yeni Uzaklık	Uzaklık Farkı
Bölge1	565,000	854,305	289,305
Bölge2	1019,824	727,235	292,589
Bölge3	962,695	540,332	422,363

En az artış Bölge1'de olduğundan Bölge1 alt bölgelere bölünür. Dört adet veri ikişer veri bir bölgede olmak üzere iki alt bölgeye bölünür.

Çizelge 5.15. Bölge1'de oluşabilecek artışlar

Çocuk1	Baba	+	Pencere	Bayrak	= 646,449

Çocuk1	Pencere	+	Baba	Bayrak	= 1214,438

Çocuk1	Bayrak	+	Baba	Pencere	= 1199,355

En az artış (Çocuk1, Baba) ve (Pencere, Bayrak) ikilileri şeklinde oluşmaktadır. Veriler bu şekilde kaydedilir ve *Ağaç düzenle* algoritması çalışarak güncellemeler yapılır. Bölge ve alt bölge sınırları güncellenir ve son durumu çizelge 5.16'daki gibi olur.

Çizelge 5.16. Ağaç verisi eklendikten sonra oluşan bölge sınırları

Bölge1

Alt Sınırlar	34, 9, 67, 98, 1, 1
Üst Sınırlar	239, 131, 345, 543, 22, 3

Bölge2

Alt Sınırlar	223, 256, 285, 277, 5, 2
Üst Sınırlar	322, 678, 489, 680, 12, 2

Bölge3

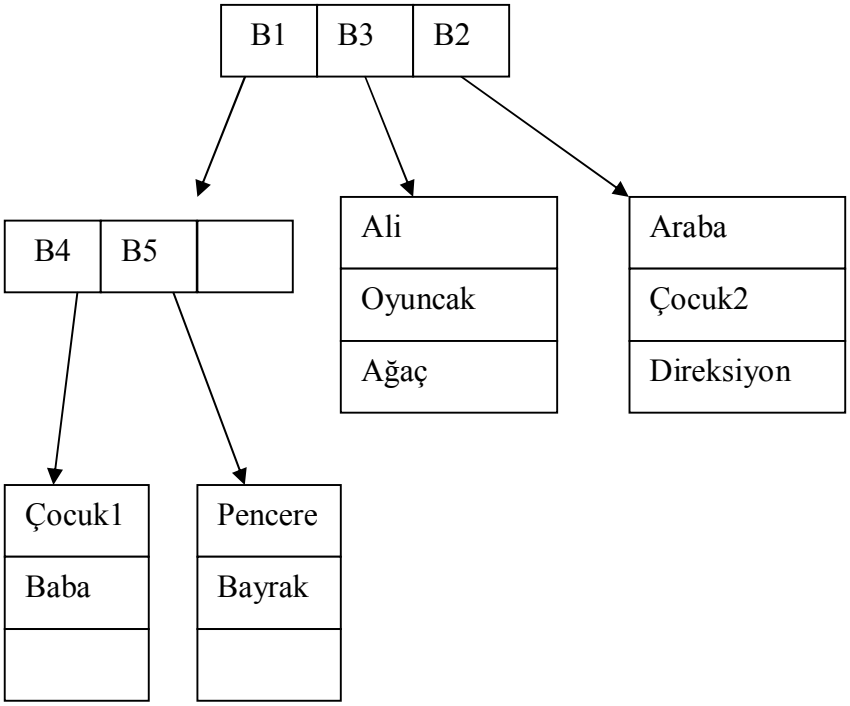
Alt Sınırlar	102, 85, 45, 344, 15, 1
Üst Sınırlar	290, 433, 655, 567, 25, 1

Bölge4

Alt Sınırlar	232, 123, 324, 408, 12, 1
Üst Sınırlar	239, 131, 345, 543, 18, 1

Bölge5

Alt Sınırlar	34, 9, 67, 98, 1, 2
Üst Sınırlar	156, 43, 345, 356, 22, 3



Şekil 5.9. Ağaç verisi eklendikten sonra bölge durumları

11- Kapı (315, 510, 345, 450, 5, 2) verisi veri tabanına kaydedilirken öncelikle *Yaprak seç* algoritması çalışır. Verinin hangi bölgeye gireceğine bakılır. Bunun için veri eklendiğinde bölgelerde oluşabilecek uzaklık farkları göz önüne alınarak bir kıyaslama işlemi yapılmaktadır. Çizelge 5.17'de oluşabilecek bu uzaklık farkları görülmektedir.

Çizelge 5.17. Kapı verisinin eklenmesi durumunda bölgelerde oluşacak uzaklık farkları

Bölge Adı	Eski Uzaklık	Yeni Uzaklık	Uzaklık Farkı
Bölge1	855,955	1002,852	146,897
Bölge2	760,511	864,998	104,487
Bölge3	626,066	626,066	0

Uzaklık farkı değeri en az Bölge3'te olacağından kapı verisi Bölge3 içerisine eklenir. Ancak verinin eklenebilmesi için yeterli yer olmadığından Bölge3 alt bölgelere bölünür. Yeni oluşan alt bölgelere verilerin nasıl yerleşeceğine yine uzaklık farklarına bakılarak en az uzaklık değeri olan veriler aynı bölgeye yerleştirilir.

Çizelge 5.18. Bölge3'ün alt bölgelere bölünmesi ile oluşabilecek durumlar

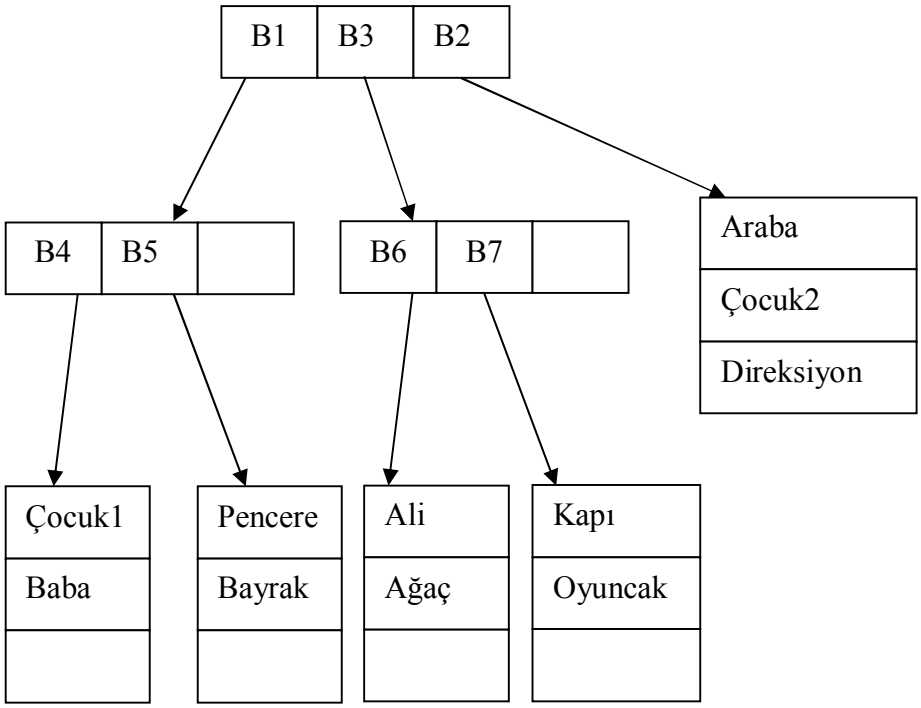
Ali	Kapı	+	Oyuncak	Ağaç	= 1259,932

Ali	Oyuncak	+	Kapı	Ağaç	= 1166,924

Ali	Ağaç	+	Kapı	Oyuncak	= 809,852

809,852<1166,924<1259,932 olduğundan,

Ali ve Ağaç aynı bölgede, Kapı ve Oyuncak aynı bölgede yer alır. Bölge3'ün bölünmesi ile oluşan alt bölgelerden Bölge6, Ali ve Ağaç verisini, Bölge7 Kapı ve Oyuncak verilerini içerir.



Şekil 5.10. Kapı verisi eklendikten sonra bölge durumları

6. SONUÇ

Konum-zaman veri tabanlarının alt bir grubu olarak nitelenen hareketli nesne veri tabanlarına günümüzde birçok alanda ihtiyaç duyulmaktadır. Trafik kontrol, uçuş kontrol, nesne takip sistemleri, konuma dayalı servisler örnek verilebilecek uygulama alanlarından birkaçıdır.

Bu tezde hareketli nesnelerin tespiti, takibi ve indekslenmeleri incelenmiş ve örnek bir veri seti üzerinden bir uygulama yazılımı ile verilerin R ağaç yapısı kullanılarak disk ortamına kaydedilmesi gerçekleştirilmiştir. Nesne tespiti ve takibi için mevcut kullanılan yöntemler ele alınarak bunların nasıl ve ne amaçla kullanıldığı incelendikten sonra indeksleme yöntemleri ile indekslemede kullanılan ağaç yapıları ele alınmıştır.

R ağaç yapısının tutulduğu bölgeler oluşturulurken eklenen verilerin bulunduğu video dosyası, saniye cinsinden bulunduğu konum, ekranda bulunduğu koordinatlar dikkate alınmaktadır. Bu parametrelerin hesaplanması sonucunda eklenen verinin hangi bölge içerisinde olacağı, bölge dolu ise nasıl bir bölge bölünmesi olacağı, bölünen bölgelerin içerisinde hangi verilerin yer alacağı bulunmaktadır. Eklediğimiz 11 veri, m değeri 3 olduğundan kök düzeyde Bölge1, Bölge2, Bölge3 olmak üzere 3 bölge oluşturmuştur.

Yapılan bu çalışmada R ağaç yapısı kullanılarak video dosyalarından istenilen nesnelerin görüntü kareleri koordinat ve zaman ile birlikte alınarak kaydedilmiştir. Uzaklık hesaplaması sayesinde benzer özelliklerdeki verilerin aynı bölgelerde tutulması sağlanmıştır. Bölgelerde bulunan verilerin benzer özelliklerde olması sayesinde sorgulama işlemleri daha hızlı sonuçlanmaktadır.

Çok fazla sayıda video dosyasının tutulduğu çoklu ortam veri tabanlarında aranılan nesnenin ve sahnenin kısa zamanda bulunması önemli bir husustur. Bu uygulama R ağaç yapısının çok boyutlu video dosyalarında kullanılan etkin bir indeksleme yöntemi olduğunu göstermektedir.

KAYNAKLAR

Almeida and Güting R.H., "Indexing the Trajectories of Moving Objects in Networks", Fernuniversität Hagen, Informatik-Report 309, March 2004.

Brinkhoff T., "A Framework for Generating Network-Based Moving Objects", GeoInformatica (2002)

Dedeoğlu Y., "Moving Object Detection, tracking and Classification For Smart Video Surveillence", 2004

Ergezer H., "Visual Detection and Tracking of MovingObjects", 2007

Frentzos, E., "Indexing Objects Moving on Fixed Networks", *8th ISSTD, 2003*

Gözet O., " Çok Boyutlu Verilerin İndekslenmesi İçin Kullanılan Yöntemler ve Bu Yöntemlerin Karşılaştırılması", 2008

Guttman A., "R-trees: A Dynamic Index Structure for Spatial Searching", Proceedings of the ACM SIGMOD, 1984

Hu W., Sha L., Sperber M., "Moving Object Spatial Queries and Indexing Techniques", 2006

Kalay U., "Hareketli Nesne Veri Tabanlarında Modelleme ve Erişim Yapılarının Tasarımı ve Gerçeklenmesi", 2007

Kolahdouzan M., Shahabi C., "Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases" Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

Kutluay H., "Dijital Videolarda Arka Plan Modelleme ve Hareketli Nesne Çıkarımı", 2008

Oracle Spatial User Guide and Manual, *Oracle 9i Documentation*

Sistla P., O. Wolfson, S. Chamberlain, S. Dao, "Modeling and Querying Moving Objects", Proceedings of the Thirteenth International Conference on Data Engineering (ICDE13), Birmingham, UK, Apr. 1997

Taşkın D., "Çoklu Ortam Veri Tabanları ve Uygulamaları", 2004

Tayeb J., Ulusoy O., and Wolfson O., "A Quadtree Based Dynamic Attribute Indexing Method," Computer Journal, vol. 41, no. 3, 1998.

Theodoridis, Y.,Vazirgiannis, M., Sellis, T. K."Spatio-Temporal Indexing for Large Multimedia Applications" In International Conference on Multimedia Computing and Systems (1996)

Yilmaz A., Javed O and Shah M., "Object Tracking: A Survey", ACM Computing Surveys, Vol. 38, No. 4, Article 13, Publication date: December 2006.

http://e-bergi.com/2009/Aralik/B+-Agaclari