GPU Programlama Tekniği ile Yüksek Performanslı Araç Takibi

Yasemin POYRAZ, Selçuk SEVGEN

Bilgisayar Programcılığı, İstanbul Üniversitesi, İstanbul, Türkiye
yasemin.poyraz@istanbul.edu.tr, sevgens@istanbul.edu.tr
(Geliş/Received:17.11.2016; Kabul/Accepted:04.05.2017)
DOI: 10.17671/gazibtd.331036

Özet —Bu çalışmada, Grafik İşlemci Birimi (GPU) Programlaması kullanılarak yüksek performanslı bir araç takip uygulaması geliştirilmiştir. GPU programlama ortamı olarak Birleşik Hesap Cihazı Mimarisi (CUDA) kullanılmıştır. Uygulama, GeForce GT 630 ve GeForce GTX 550Ti isimli iki farklı ekran kartı üzerinde test edilmiş ve farklı GPU'lara sahip ekran kartlarının uygulamanın performansına olan etkisi incelenmiştir. Ayrıca uygulama MATLAB ortamında Merkezi İşlemci Birimi (CPU) üzerinde de gerçeklenerek, CPU-GPU karşılaştırılması yapılmış ve ayrıntılı sonuçlar sunulmuştur. Elde edilen sonuçlar, araç takibi işleminde GPU programlamanın kullanılmasının yüksek performans kazanımı getirdiğini göstermiştir.

Anahtar Kelimeler— Grafik işlemci birimi, CUDA, araç takip, görüntü işleme.

High Performance Vehicle Tracking Implemented by Using GPU Programming

Abstract— In this study, a high performance vehicle tracking application was implemented using the Graphic Processing Unit (GPU) Programming. For the GPU programming environment, Compute Unified Device Architecture (CUDA) was used. The application was tested on two different graphics card, namely GeForce GT 630 and GeForce GTX 550Ti and the effect of the graphics cards with different GPUs on the performance of the application was inspected. Also, the application was implemented for the Central Processing Unit (CPU) in MATLAB environment, CPU vs GPU comparisons were done and detailed results are presented. The detailed results show that in vehicle tracking operation, using GPU programming achieves high performance.

Keywords— Graphic processing unit, CUDA, vehicle tracking, image processing.

1. GİRİŞ (INTRODUCTION)

Son yıllarda görüntü işleme alanında yapılan çalışmaların sayısı oldukça artmıştır. Araç tespiti ve takibi ise görüntü işleme uygulamalarında önemli bir yere sahiptir. Bu konu ile ilgili literatürde gerçekleştirilmiş pek çok çalışma bulunmaktadır [1 -10].

Araç takip işleminde model tabanlı, aktif kontur tabanlı, özellik tabanlı ve bölge tabanlı olmak üzere 4 farklı yaklaşım bulunmaktadır. Coifman ve diğerleri, araçları tespit etmek için özellik-tabanlı izleme sistemi kullanmışlardır. Araçların tüm özelliklerini takip etmek yerine tek bir özellik ile takip işlemini gerçekleştirmişlerdir. Böylece daha sağlıklı bir takip işlemi yapılmıştır. Çünkü sistem aracın en belirgin özelliğini belirleyerek takip işlemini gerçekleştirdiğinden, ışık gibi ortam değişikliğinden kaynaklanan olumsuz etki oranı azaltılmıştır [1].

Betke ve diğerleri, gerçek zamanlı bir görüntü sistemi oluşturmuşlardır. Kamera, araçların geliş doğrultusunda yerleştirilmiş ve renkli bir video oluşturulmuştur. Sistemde araç tanıma ve izleme işlemi için renk, kenar ve hareket bilgisi kullanılmıştır. Deneysel sonuçlar sağlam ve gerçek zamanlı bir araç algılama sistemi gerçekleştirildiğini göstermiştir [2].

Derek ise araç takip işlemi için yeni bir yaklaşım sunmuştur. Her piksel için Gauss tabanlı bir arka plan modeli ve aracın boyutu pozisyonu, hızı ve renk dağılımından oluşan bir ön plan modelinin birleşimini ortaya koymuştur. Her bir piksel ya ön plan ya arka plan ya da gürültü pikseli olabilir. Aracın boyutu ve hızı hakkında tutarlılık olması için zemin-düzlem dönüşümü

(ground-plane transform) kullanılmıştır. Bu sistemde kullanılan videonun hızı 20 kare/sn'dir [3].

Gerçek zamanlı trafik izleme makine öğrenmesi alanında en zorlu çalışmalardan biridir. Rad ve Jamzad, şerit değiştirme tespiti, araç sınıflandırma ve araç sayma işlemleri için gerçek zamanlı bir algoritma sunmuşlardır. Bu sistemde Kalman filtresi, arka plan ayrımı yöntemleri ve morfolojik işlemler kullanılmıştır. Algoritma CPU üzerinde çalışmaktadır, çalışma hızı 11 kare/sn ve doğruluk oranı %96 olarak ölçülmüştür. Bu çalışmada kullanılan videonun saniyedeki görüntü karesi oldukça düşüktür [4].

Haag ve Nagel, geliştirdikleri model-tabanlı araç takip sistemini, şehir içi trafiğinden alınan 15 dakikalık video üzerinde denemişlerdir. 400 araç geçişinde 335 tanesini başarılı bir şekilde takip ederek %80 başarı oranı elde etmişlerdir. Model tabanlı araç takip algoritması ile bu sistemin başarı oranının videonun kalitesine ve araçların geçme sıklığına bağlı olduğu görülmüştür. Sistem değişik videolar üzerinde denetilerek %65 ve %80 arasında başarı oranı göstermiştir [5].

Model tabanlı başka bir araç takip sistemi ise Kim ve Malik tarafından geliştirilmiştir. Buradaki araç belirleme algoritması, olasılık yoğunluk fonksiyonlarını temel alarak gerçekleştirilmiştir. Böylece araçların 3 boyutlu sınıflandırma işlemi de gerçekleştirmiştir. Bu yüzden çalışmanın performansı bakış açısının değişmesine bağlı değildir [6].

Model tabanlı bir diğer uygulama Hu ve diğerleri tarafından gerçekleştirilmiştir. Bu çalışmada, trafik kazalarını tahmin etmek için 3 boyutlu model tabanlı bir olasılık modeli oluşturulmuştur. Bulanık, kendi kendini düzenleyen yapay sinir ağı algoritması geliştirilmiştir. Yapay sinir ağları kullanılarak araçların aktiviteleri öğrenilmiştir. Böylece araçların olası hamleleri tahmin edilerek kaza yapma olasılığı belirlenmiştir [7].

Hsieh ve diğerleri, araçları sadece "araçtır" veya "araç değildir" olarak sınıflandırmışlardır. Yapılan çalışma gölgeden dolayı oluşan gürültüden çok az etkilenmektedir. Bu çalışmada kullanılan yöntem çizgi tabanlı gölge algoritmasıdır. Gürültüye sebep olan gölgelerin etkisini yok etmek için çizgiler kullanılmıştır ve güvenilir sonuçlar elde edilmiştir [8].

Kastrinaki ve diğerleri, trafik uygulamaları için çok geniş kapsamlı bir araştırma çalışması gerçekleştirmişlerdir. Trafik için yapılan çalışmaları özellik-tabanlı, bölge-tabanlı ve model-tabanlı olmak üzere kategorilere ayırmıştır. Aynı zamanda durağan kameradan alınan görüntüler ve hareketli kameradan alınan görüntülerle yapılan çalışmaları da ayrı kategori altında incelemişlerdir. Yapılan çalışmaların birbirine göre olumlu ve olumsuz yönlerini sunmuşlardır [9].

Sivaraman ve Manubhai, aracın izlediği yoldaki, sol ve sağ şerit sınırları, uzunlamasına mesafe, şerit içindeki yan konum, şerit genişliği, sol ve sağ şerit sınır eğimi, eğrilik, şerit durumu tahmini gibi tüm bilgileri kullanarak takip işlemi yapmışlardır. 11 kare/sn özellikli videolar için sistem gerçek zamanlı olarak çalışmaktadır [10].

Do ve Woo, araç takibi sırasında tahmin edilemeyen ortam değişikliklerinden kaynaklanan çözünürlük problemine çözüm bulmayı hedeflemişlerdir. Araç tespiti için SIFT algoritması ve araç takibi için de Piramit Lucad Kanada Optik akış algoritması kullanılmıştır [11].

Zhao ve diğerleri, araçların izini sürmek için yeni bir bilgisayarlı görme tabanlı trafik gözetim sistemi sunmuşlardır. Sezgisel ve makine öğrenmesi yaklaşımları yanında doğruluk oranını artırmak için SURF algoritması da kullanılmıştır [12].

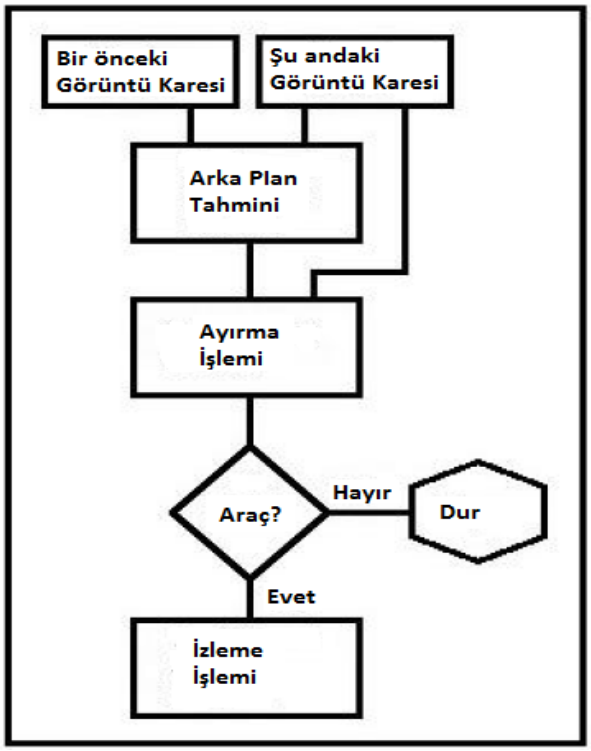
Lee ve diğerleri, hızlı ve çok fazla işlem gerektirmeyen bir algoritma kullanarak araç takip işlemini gerçekleştirmiştir. Ani görünüş değişikliklerini hesaplamak için, manifold öğrenme yöntemi kullanılmıştır. Hareketteki ani değişimleri yönetmek için izleyicinin bir sonraki muhtemel merkez alanı yörünge bilgisi kullanılarak tahmin edilmiştir. Daha sonra konum benzerliği, izleyicinin öngörülen sonraki konumu ve ilerleme yönüne dayanarak hesaplanmıştır [13].

Bu çalışmada ise bölge tabanlı izleme yöntemi kullanılmıştır. Çalışmanın temel amacı gerçek zamanlı araç takip uygulaması gerçekleştirmektir. [2, 4, 13] çalışmalarında da gerçek zamanlı araç takip sistemi yapılmıştır. Ancak bu çalışmalarda kullanılan videoların saniyedeki görüntü karesi diğer uygulamalara göre oldukça düşüktür. Bu çalışmada ise videoların saniyedeki görüntü karesi sayısının düşürülmesi yerine NVIDIA firmasının sağlamış olduğu GPU programlama tekniği ile çalışma performansı artırılmıştır.

Görüntü işleme uygulamaları yüksek hesaplama gücü gerektiren uygulamalardır. Bu yüzden hala klasik CPU üzerinde gerçekleşen uygulamalar ihtiyacı karşılayamamaktadır. Diğer taraftan, var olan görüntü işleme teknikleri ve metotları büyük verileri işleyecek kapasitede olmayabilir [14]. Bu yüzden paralel hesaplama teknikleri ve metotları görüntü işleme alanındaki problemleri çözmede büyük öneme sahiptir. Dolayısıyla bu çalışmada paralel işlem yapabilen NVIDIA CUDA C ve C++ dilleri kullanılarak, OpenCV entegrasyonu ile araç takibi uygulaması gerçekleştirilmiştir. Yazılım Visual Studio 2010 ortamında hazırlanmıştır. CUDA C ile Visual Studio ortamında debug işlemi gerçekleştirilemediği için NVIDIA Parallel Nsight eklentisinden yararlanılmıştır. Video okuma işlemleri OpenCV 2.4.5 versiyonu kullanılarak gerçekleştirilmiştir.

2. MALZEME ve YÖNTEM (MATERIAL and METHOD)

Bu bölümde, çalışmada kullanılan araç takip uygulaması anlatılacaktır. Araç takip uygulamasının tasarımı temel olarak 2 bölümden oluşmaktadır. Birinci kısım uygulamanın CPU tarafında çalışacak olan kısmıdır. İkinci kısım ise görüntü işleme adımlarını yürütecek olan GPU kısmıdır. Bu temel ayrımın sebebi uygulamanın iki farklı alan arasında gerçekleşen bir iş akışına sahip olmasıdır. Araç takip algoritmasının GPU tarafından yürütülecek olan bölümün akış şeması Şekil 1'de gösterilmiştir.



Şekil 1. Araç Takip Algoritması Akış Şeması.
(Flowchart of Vehicle Tracking Algorithm)

Kullanılan algoritmanın adımları ayrıntılı olarak aşağıda verilmiştir.

2.1. Video Okuma İşlemi (Capturing Images)

Değişen video formatları, bir görüntü işleme uygulaması için en önemli sorunlardan biridir. Bu uygulama içerisinde video okuma fonksiyonu CPU üzerinde çalışmaktadır. OpenCV, HighGUI sınıfı altında sağladığı yapılar ile bu işlemi en basit şekilde gerçekleştirmektedir. Şekil 2'de dosyadan görüntü okuması yapan kod görülmektedir.

```
CvCapture* capture = cvCreateFileCapture("c:\\arac.avi");
```

Şekil 2. OpenCV video okuma fonksiyonu.
(OpenCV function for capturing Images)

Bu fonksiyon ile videonun ve videodaki her bir görüntü karesinin özellikleri hakkında bilgi sahibi olunabilmektedir.

2.2. Gri Formata Çevirme (Gray-Scale Transform)

Video içerisindeki her bir görüntü karesi RGB (kırmızı-yeşil-mavi) formatında elde edilmiştir. Bu görüntü karelerinin işlenebilmesi için ilk olarak gri formata çevrilmesi gerekmektedir. Bu çevirim işlemi için aşağıdaki denklem kullanılmıştır.

$$Gri\_Format = Kırmızı*0.3 + Yeşil*0.6 + Mavi*0.1 \quad (1)$$

Bütün piksel değerleri rgb_to_kernel isimli çekirdek ile aynı anda gri formata çevrilmiştir. Şekil 3'de gri formata çevrilmiş bir görüntü karesi örneği gösterilmiştir.

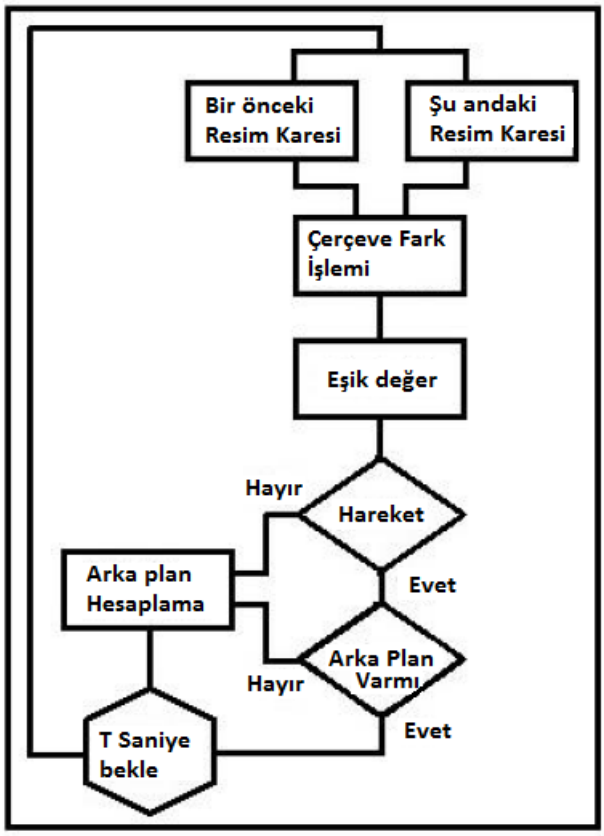
Şekil 2. Gri formata çevrilmiş bir görüntü karesi.
(An example of background image frame)

2.3. *Arka Plan Ayrımı ve İkili Seviyeye Dönüşüm İşlemi (Background Substraction and Binary Transform)*

Arka plan ayrımı işlemi araç takip uygulamasının başarı oranını etkileyen önemli bir adımdır. Takip algoritmasının başarısı arka plan ayrımı işlemine bağlıdır. Arka plan ayrımı işlemi ne kadar başarılı gerçekleştirilirse ön plandaki araçların görüntüleri de o kadar net elde edilir. Literatürde arka plan ayrımı işlemi için çeşitli algoritmalar geliştirilmiştir. Bunlardan bazıları Çerçeve Fark Algoritması [15, 16], Running Gaussian Average [17], Geçici Medyan Filtreleme [18], Gaussian Mixture Model (GMM) [19], Kernel Density Estimation (KDE) [20] ve Kalman Filtresi [21]'dir.

Bu çalışmada arka plan ayrımı işlemi ön plandaki araçları algılama işlemine dayanmaktadır. Bu işlem için çerçeve fark algoritması kullanılmıştır. Elde edilen arka plan görüntüsü Şekil 4'de, çerçeve fark algoritmasının akış şeması Şekil 5'de [22, 23] gösterilmiştir.

Şekil 3. Elde edilen arka plan görüntüsü.
(Background image frame.)



Şekil 4. Arka Plan Ayrımı için Çerçeve Fark Algoritması Akış Şeması.
(The frame difference algorithm flowchart for Background Substraction)

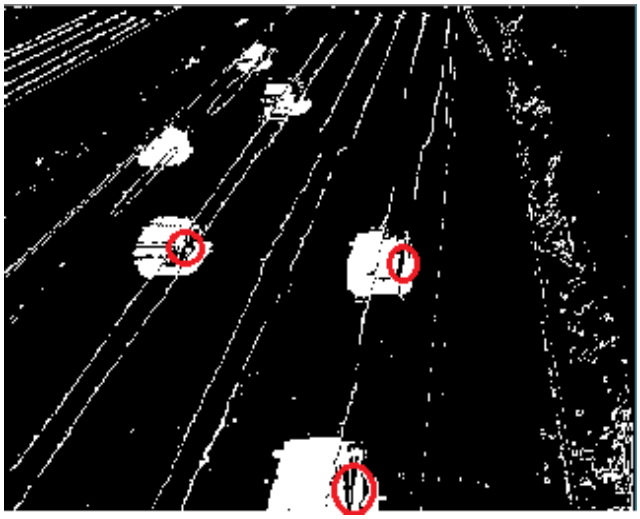
Bu algoritmada görüntü üzerinde kameradan ya da ışık değişiminden kaynaklanarak oluşan gürültü piksellerini azaltmak için başlangıçta ortalama filtre metodu kullanılmıştır.

Her bir görüntü karesinin aynı pozisyonda bulunan piksel değerlerinin ortalama değeri hesaplanmıştır. Bu ortalama değer arka plan resminin o pozisyondaki piksel değerini temsil etmektedir.

Daha sonra her bir görüntü karesinden oluşturulan arka plan resminin mutlak farkı alınmıştır. Fark değerinden oluşan yeni görüntü karesinde sıfırdan farklı değerler hareketin olduğu yerleri temsil etmektedir. Bazı durumlarda, hareket olmadığı halde kamera ya da ışık değişimi hareket var olgusu oluşturabilmektedir. Bu durumu önlemek için bir eşik değer belirlenmiştir. Görüntü kareleri ve arka plan resmi arasında mutlak fark alma işlemi yapılırken, fark değeri belirlenen eşik değerinden yüksek ise o piksel değeri ön plan olarak kabul edilmiş ve 1 değeri atanmıştır; eşik değerinden küçük ise arka plan olarak kabul edilmiş ve 0 değeri atanmıştır. Böylece her bir görüntü karesi ikili seviye dönüştürülmüştür.

2.4. Morfolojik İşlemler (Morphological Operations)

Arka plan ayrımı işleminde kullanılan eşik değer, oluşturulan ikili resimlerdeki bütün gürültü değerlerini yok edememektedir. Bu yüzden algoritma içerisinde açma işlemi uygulanmıştır. Gürültünün var olduğu bir görüntü karesi örneği Şekil 6'da kırmızı halkalar ile gösterilmiştir. Açma işlemi erozyon ve genişletme adında iki işlemden oluşmaktadır. Erozyon işleminin amacı ön plandaki yani 1 değerine sahip piksel değerlerini sınırlandırmaktır. Böylece ön plandaki nesneler daha küçük olacak ve bu nesnelerin içerisinde olan delikler daha da büyüyecektir. Genişletme işleminin amacı ise erozyon işlemine uğramış görüntü karesinde bulunan ön plandaki nesnelerin sınırlarını genişletmek ve bu nesnelerdeki delikleri küçültmek ya da tamamen yok etmektir. Böylece erozyon işlemi ile çok küçük nesneler yok edilerek genişletme işleminde tekrardan oluşması engellenecektir.



Şekil 5. Gürültülü görüntü karesi örneği.
(A sample of noisy image frame)

Erozyon ve genişletme işlemlerinin denklemleri aşağıda verilmiştir:

Erozyon denklemi:

$$\in_B (X) = X \ominus B = \{\varkappa \backslash B_x \subset X\} \qquad (2)$$

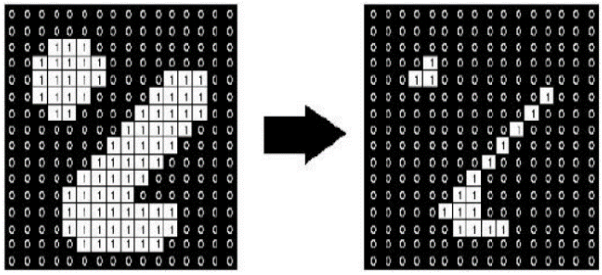
Genişletme denklemi:

$$\delta_B(X) = X \oplus B = \{x + b \backslash b \in B, x \in X\} \quad (3)$$

Burada X değeri B ve $\in$'nin alt kümesi olan yapı elementidir [14].

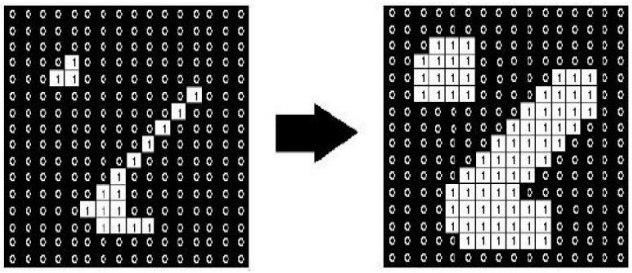
Erozyon ve genişletme işlemi 3x3'lük bir yapısal eleman ile bütün görüntü karelerine uygulanmıştır. Şekil 7 ve 8'de erozyon ve genişletme işlemi sırası ile gösterilmiştir.

Erozyon: 3x3 boyutundaki yapısal elemanın merkezi, 1 değerine sahip ön plandaki piksel değeri ile üst üste gelecek şekilde yerleştirilir. Ön plandaki piksel değerinin 3x3'lük bütün komşu değerleri 1 değerine sahip ise piksel değeri değişmez. Aksi halde 0 değeri atanır.



Şekil 6. Erozyon işlemi.
(Eroded image)

Genişletme: 3x3'lük yapısal eleman görüntünün bütün piksel değerleri üzerinde gezdirilir. 3x3'lük komşu değerlerinden en az bir tanesi 1 değerine sahip ise merkezdeki piksel değerine 1 atanır.



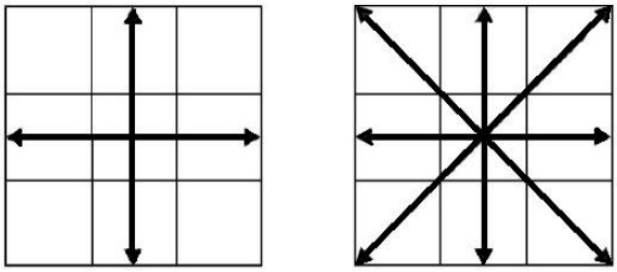
Şekil 7. Genişletme işlemi.
(Dilated image)

2.5. Araç Tespiti ve Takibi (Vehicle Detecting and Tracking)

Nesne tespit algoritmaları genel olarak görüntü üzerinde her bir pikseli değerlendikleri için sıralı programlama teknikleri ile uygulandıklarında düşük performans göstermektedirler.

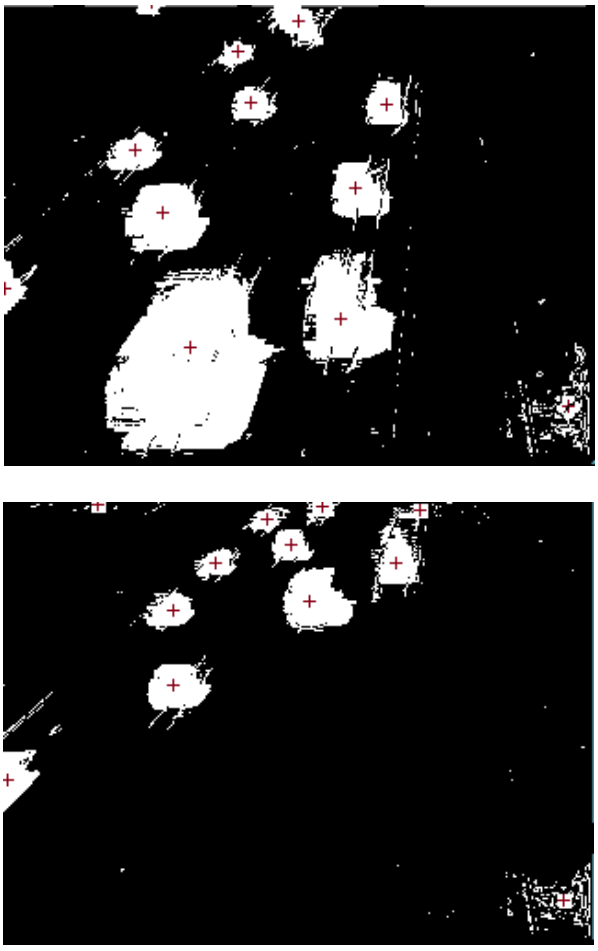
Araç izleme için aracın tüm özelliklerini tanımlamak gereklidir. Bu işleme blob analizi denmektedir. Bu çalışmada araç konum tespiti için blob bulma yönteminden faydalanılmıştır. Hareketli bir nesne belirlendiğinde araç olup olmadığını belirlemek için kameranın pozisyonuna göre ortalama bir değer belirlenmiştir.

Blob analizi yönteminde birbirine bağlı pikselleri bulmanın iki farklı yolu vardır. Birincisi yatay-dikey bitişik pikselleri analiz etmek, ikincisi de birbirine çapraz bağlı pikselleri analiz etmektir (Şekil 9).



Şekil 8. Birbirine bağlı piksel konumları örneği.
(Two rules of adjacent pixels)

Blob analizi yönteminin performansı morfolojik işlemlere ve ayırma işlemine direkt olarak bağlıdır. Yetersiz ayırma işlemi gürültü piksel değerlerini blob olarak algılanmasına neden olmaktadır.



Şekil 9.Takip sonucu görüntüleri.
(Result images)

Takip işleminin birinci adımında Şekil 9'da gösterilen komşu değerler ile merkezdeki pikselin değeri karşılaştırılır. İşaretli olan komşulardan değeri 255 olan ve etiket değeri en küçük olan pikselin etiket değeri merkez piksele atanır. Bu çalışmada ise eğer bir pikselin kendinden önceki dört komşusunun değerleri 255 değilse veya etiket değerleri merkez pikselden küçükse sıralı olarak düzenlenen etiket numarası yerine pikselin görüntüdeki piksel dizisi içindeki sırası etiket değeri olarak kullanılmıştır. Bu aşama CUDA C ile düzenlendiği için her bir kanalda bir piksel ve onun komşuları arasındaki işlemler gerçekleştirir. Yani birinci adım hiçbir döngü kullanılmadan bütün resim için gerçekleştirilir. Çünkü bütün kanallar paralel olarak pikselleri işlemektedir. Sıralı çözümlerde böyle bir yaklaşım performansı dikkat çekici bir şekilde düşürmektedir.

3. BULGULAR (RESULTS)

Nesne takibi son 20 yıldır görüntü işleme alanında çok fazla araştırmanın yapıldığı bir konudur. Bu çalışmada paralel programlama ile yüksek performanslı işlemler gerçekleştirme imkânı sağlayan CUDA C dili kullanılarak, araç tespiti ve takibi uygulaması gerçekleştirilmiştir. Bu uygulama için kullanılan video görüntüsünün özellikleri Tablo 1'de verilmiştir.

Tablo 1. Videonun özellikleri.
(Properties of the video)

Görüntü Karesi Sayısı	Süresi	Çözünürlük
462	33 Saniye	240x320

Algoritmanın uygulanmasında kullanılan GeForce GT 630 2GB ve GeForce GTX 550 Ti ekran kartlarının özellikleri ise Tablo 2'de sunulmuştur.

Tablo 2: Ekran Kartlarının Özellikleri.
(Properties of two GPU graphic cards)

Özellikler	GeForce GT 630 2GB	GeForce GTX 550 Ti
Toplam Hafıza	2048 MB	1024 MB
GPU hızı	1,62 GHz	1,82 GHz
Toplam Doku Hafıza	65536	65536
Toplam Sabit Hafıza	65536	65536
Her Blok için Toplam Paylaşılan Hafıza	49152	49152
Warp Boyutu	32	32
Thread Sayısı	1024	1024
Bir Blok İçerisindeki Thread Sayısı	< 1024; 1024; 64 >	< 1024; 1024; 64 >
Grid Boyutu	< 65535; 65535; 65535 >	< 65535; 65535; 65535 >
CUDA Çekirdek Sayısı	96	192

Tablo 3'de algoritmanın GeForce GT 630 2GB ve GeForce GTX 550 Ti ekran kartları üzerinde çalışma zamanı ayrıntılı olarak gösterilmiştir.

Tablo 3'deki sonuçlardan toplam çalışma zamanının direkt olarak ekran kartının özelliğine bağlı olduğu görülebilmektedir. Uygulamanın toplam çalışma zamanı Tablo 4'de gösterilmiştir.

Tablo 3:Her ekran kartı için algoritmanın çalışma süresi.
(Running time of the algorithm for each graphic card)

	GeForce GT 630 2GB	GeForce GTX 550 Ti
Griye Çevirme	0,046 ms	0,020 ms
Arka Plan Tahmini	0,680 ms	0,392 ms
Erozyon	0,070 ms	0,029 ms
Genişletme	0,142 ms	0,063 ms
Araç tespiti ve Takibi	35,951 ms	20,106 ms

Tablo 4: Uygulamanın toplam çalışma süresi.
(The total running times)

GeForce GT 630 2GB	GeForce GTX 550 Ti
17,042718 sn + 33 sn	9,52182 sn + 33 sn

Bu çalışmada kullanılan algoritma CPU üzerinde MATLAB ortamı kullanılarak da gerçekleştirilmiştir. Intel Core i3 2.1 GHz işlemci, 3Gb RAM ve 64 bit işletim sistemine sahip bir bilgisayar kullanılmıştır. Aynı video ile toplam çalışma süresi 20.1 dakika olarak ölçülmüştür. Hesaplanan bu değer Tablo 4'de bulunan GPU programlama tekniği kullanılarak bulunan çalışma süreleri ile karşılaştırıldığında önemli ölçüde bir performans kazancı elde edildiği görülmektedir.

4. SONUÇLAR (CONCLUSIONS)

Bu çalışmada, GPU programlama kullanılarak bir araç tespit ve takip uygulaması gerçekleştirilmiştir. GeForce GT 630 2GB ve GeForce GTX 550 Ti isimli iki adet ekran kartı üzerinde aynı video ile işlemler yapılmıştır. Elde edilen sonuçlar ekran kartları üzerinde bulunan GPU'ların araç tespit ve takip işleminin çalışma süresi üzerinde çok önemli rolü olduğunu göstermiştir. Aynı algoritma CPU üzerinde MATLAB ortamı kullanılarak da gerçekleştirilmiştir. Bulunan süreler incelendiğinde, GPU programlamanın paralel işlem özelliği kullanılarak, yüksek performanslı araç tespit ve takip işleminin gerçekleştirilebileceği görülmektedir. Ayrıca literatürde bulunan bazı çalışmalarda [2, 4, 13] olduğu gibi kullanılan videonun saniyedeki görüntü sayısı düşürülmeden yüksek performanslı bir sistem gerçekleştirilmiştir.

Bu çalışmada gerçekleştirilen uygulama ve elde edilen sonuçlar GPU programlamanın, sahip olduğu paralel işlem özelliği sayesinde özellikle yüksek işlem hızına ihtiyaç duyulan alanlarda çok etkili olabileceğini göstermektedir.

Elde edilen sonuçlar değerlendirildiğinde, araç tespiti ve takibi işleminin performansının daha da artırılması yönünde yeni çalışmaların yapılabileceği düşünülmektedir. Bu amaçla, yapay zekâ gibi farklı yöntemler kullanılabilir.

KAYNAKLAR (REFERENCES)

[1] Coifman, B., Beymer, D., McLauchlan, P., & Malik, J. (1998). A real-time computer vision system for vehicle tracking and traffic surveillance. Transportation Research, Part C, 6, 271-288.

[2] Betke, M. T., Haritaoglu, E., & Davis, L. S. (2000). Real-time multiple vehicle detection and tracking from a moving vehicle. Machine Vision and Applications, 12, 69–83.

[3] Derek, M. R. (2004). Tracking multiple vehicles using foreground, background and motion models. Image and Vision Computing 22, 143–155.

[4] Rad, R., & Jamzad, M. (2005). Real time classification and tracking of multiple vehicles in highways. Pattern Recognition Letters, 26, 1597–1607.

[5] Haag, M., & Nagel, H. H. (1999). Combination of edge element and optical flow estimates for 3D-model-based vehicle tracking in traffic image sequences. International Journal of Computer Vision, 35, 3, 295–319.

[6] Kim, Z., & Malik, J. (2003). Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking. Proceedings of the Ninth IEEE International Conference on Computer Vision, 524–531.

[7] Hu, W., Xiao, X., Xie, D., Tan, T., & Maybank, S. (2001,4). Traffic accident prediction using 3D model-based vehicle tracking. IEEE Transactions on Vehicular Technology, 53, 3, 677 - 694.

[8] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, W.-F. Hu, Automatic Traffic Surveillance System for Vehicle Tracking and Classification, IEEE Transactions on Intelligent Transportation Systems, Vol. 7, No. 2, Haziran 2006.

[9] Kastrinaki, V., Zervakis, M., & Kalaitzakis, K. (2003). A survey of video processing techniques for traffic applications. Image and Vision Computing, 21, 359–381.

[10] Sivaraman, S., & Manubhai, M. (2013). Integrated lane and vehicle detection, localization, and tracking: a synergistic approach. IEEE Transactions on Transportation Systems, 14, 2, 906 - 917.

[11] Do, V. D., & Woo, D.M. (2016). Optical flow on vehicle tracking under unpredictable environments. Advanced Science and Technology Letters, 126, 32-36.

[12] Zhao, X., Dawson, D., Sarasua, W. A., & Birchfield, S.T. (2016). Automated traffic surveillance system with aerial camera arrays imagery: macroscopic data collection with vehicle tracking, American Society of Civil Engineers, 10.1061/(ASCE)CP.1943-5487.0000646.

[13] Lee, G., Mallipeddi, R., & Lee, M. (2017). trajectory-based vehicle tracking at low frame rates. Expert Systems With Applications, 10.1016/j.eswa.2017.03.023.

[14] Saxena, S., & Sharma, N. (2016). Parallel image processing techniques, benefits and limitations. Research Journal of Applied Sciences, Engineering and Technology, 12, 2, 223-238.

[15] Güler, E., & Geçer, B. (2013). People Counting. http://www.ebubekirguler.com/goruntu-isleme-yontemleri-ile-insan-sayma/.

[16] El-Azim, S. A., Ismail I., & El-Lati, H. A. (2002). An efficient object tracking technique using block-matching algorithm. Proc. Of the Nineteenth National, Radio Science Conf., 427-433.

[17] Wren, C., Azarhayejani, A. Darrell, T., & Pentland, A.P. (1997). Pfinder: real-time tracking of the human body. IEEE Trans. on Pattern Analysis. And Machine Intelligence, 19, 7, 780-785.

[18] Lo, B.P.L., & Velastin, S.A. (2001). Automatic congestion detection system for underground platforms. Proc. Of Int. Symposium on Intelligent Multimedia, Video and Speech Processing, 158-161.

[19] Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. Proceedings of conference on Computer Vision and Pattern Recognition, 2, 246-25.

[20] Elgammal, A., Hanvood, D., & Davis, L.S. (2000). Nonparametric model for background subtraction. Proc. Of European Conf. on Computer Vision, 751-767.

[21] Koller, D., Weber, J., Huang, T., Malik, J., Ogasawara, G., Rao, B., & Russell, S. (1994). Towards robust automatic traffic scene analysis in real-time, Proc. Of Int. Conf. on Pattern Recognition, 126-131.

[22] Leoch, D. (2007). Real-time people counting system using video camera, Yüksek Lisans Tezi, Gjvik University College, 2007.

[23] Gutchess, D., Trajkonic, M., Cohen-Solal, E., Lyons, D., & Jain, A. K. (2001). A background model initialization algorithm for video surveillance. The 8th IEEE Int. Conf. on Computer Vision, 733-740.