Journal of the Faculty of Engineering and Architecture of Gazi University 32:2(2017) 439-448

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University
Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Üretim aşamasında ray ve profilde oluşan kusurların tespitine yönelik bir paralel kusur algılama algoritması

İlhami Muharrem Orak[1], Ahmet Çelik[2*]
[1]Mühendislik Fakültesi, Karabük Üniversitesi, Bilgisayar Mühendisliği Bölümü, 78050, Karabük, Türkiye
[2]Tavşanlı Meslek Yüksekokulu, Dumlupınar Üniversitesi, Bilgisayar Teknolojileri Bölümü, 43300, Kütahya, Türkiye

Ö N E Ç I K A N L A R

- Ray ve profil kusurlarının bilgisayar destekli tespit edilmesi
- Paralel görüntü işleme metodu
- CUDA kernel yapısının performans kıyaslaması

Makale Bilgileri
Geliş: 17.03.2016
Kabul: 14.11.16

DOI:
10.17341/gazimmfd.322168

Anahtar Kelimeler:
Sıcak haddeleme işlemi, kusur tespiti, ray, profil, grafik işlemcisi, cuda, paralel işlem

ÖZET

Otomatik bir sistem tarafından görüntü işlenerek sonuç elde etmek günümüzde pek çok alanda gerekli olabilmektedir. Kusurlu ürün üretimi birçok alanda karşılaşılan üreticiler tarafından da istenmeyen bir durumdur. Görüntü işleyerek görüntü üzerindeki kusurların tespit edilmesi bu alanda kullanılan bir yöntemdir. Görüntü işleme piksel temelli yapıldığından dolayı çok iş yükü oluşturmaktadır. Hızın işlem sürecinde önem arz ettiği durumlarda paralel görüntü işlemenin yapılması bir çözüm olabilmektedir. Bundan dolayı mevcut çok çekirdekli bilgisayarların donanım ya da yazılım yardımıyla paralelleştirilerek görüntülerin işlenmesi performans sağlayacaktır. Paralel görüntü işlemede elde edilen performans, kullanılan algoritmanın paralelliğe uygun olması ve işlemcilere doğru bir şekilde dağıtım yapılması ile ilişkilidir. Kaynakların ortak kullanımı ve veri alışverişinin fazla olması performansı doğrudan etkiler. Bu çalışmada; Kardemir A.Ş. haddehanede, haddeleme işlemi sırasında ray ve profil yüzeylerinde meydana gelen kusurların tespit edilmesine yönelik geliştirilen COLMSTD algoritmasının paralel uygulaması, iki farklı şekilde gerçekleştirilmiştir. 1. Yöntemde GPU'da çalışacak CUDA çekirdek sayıları yazılımsal olarak değiştirilmiş ve 2. Yöntemde ise tek CUDA çekirdeğindeki blok sayıları değiştirilerek, test gerçekleştirilmiştir. GPU (Graphics Processing Unit) üzerinde CUDA (Compute Unified Device Architecture) arayüz desteğinin uygulanması ile elde edilen değerler ile CPU üzerinde elde edilen değerler kıyaslanmıştır.

A parallel algorithm for defect detection of rail and profile in the manufacturing

H I G H L I G H T S

- Computer aided defect detection of rail and profile
- Parallel image processing method
- Comparing performance of CUDA kernel structure

Article Info
Received: 17.03.2016
Accepted: 14.11.16

DOI:
10.17341/gazimmfd.322168

Keywords:
Hot rolling processing, defect detection, rail, profile, graphic processor, cuda, parallel processing

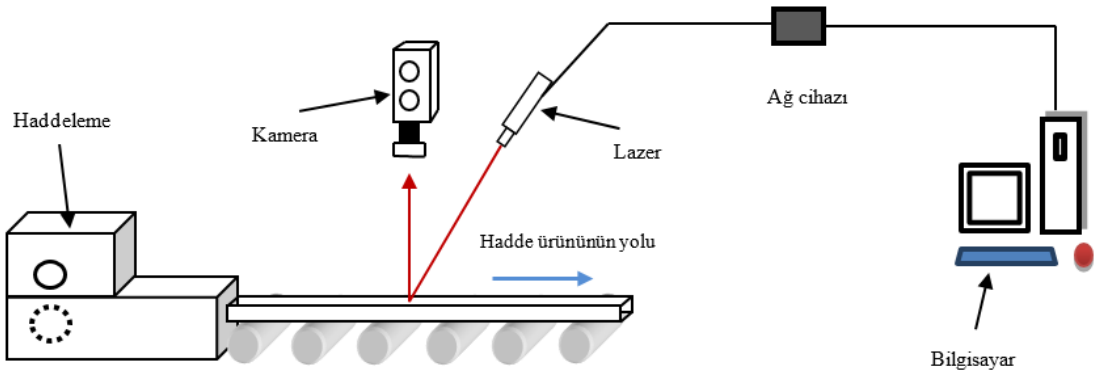
ABSTRACT

Obtaining a result by processing an image via an automatic system may be useful in many fields today. Manufacturing a defective product is an undesired case for manufacturers in many fields. Processing images is an efficient method used to detect defects on images to eliminate the defective products. Since image processing is conducted on pixel basis, it entails great workload. In cases where speed is important in processing, parallel image processing might be a solution. Therefore, processing images in the current multi-core computers by paralleling them with additional hardware and software can boost the performance. The performance in parallel image processing is related to relevance of the algorithm to the parallelism and its accurate distribution to the processors. Common use of the resources and excess of data exchange affect the performance directly. In this study, parallel application of COLMSTD algorithm developed to detect the defects on rail and profile surface during rolling in Kardemir Inc. rolling plant was conducted in two different ways. The 1st method was carried out by selecting the CUDA core numbers in GPU structure by software and the 2nd method was conducted by using single CUDA core. The performance of the results obtained on GPU (Graphics Processing Unit) with the support of CUDA (Compute Unified Device Architecture) interface was compared with that of CPU values.

*Sorumlu Yazar/Corresponding Author: ahmet.celik@dpu.edu.tr / Tel: +90 274 614 8671

1. GİRİŞ (INTRODUCTION)

Son yıllarda üretim aşaması sürecinde otomatik kusur tespit sistemlerine ihtiyaç artmaktadır. Kusur tespiti, kalite kontrol aşamasında çok önemli ve gerekli bir basamaktır. Kalite kontrolün amacı ürünlerin kusurlarını tespit etmek ve konumlarını belirlemektir. Bu bilgilere dayanarak da kısa zamanda kusur nedeni tespit edilmekte ve buna çözüm bulunması amaçlanmaktadır. Günümüzde hala üretim sürecinde geleneksel yöntem olarak insan faktörü kullanılmaktadır. Sadece insana dayalı sistemlerde birçok faktöre bağlı olarak tehlikeli durumlarla karşılaşılmakta ve hata oranı da artmaktadır [1]. Ayrıca kusurlu ürün üretimi, hem ekonomik hem de ticari açıdan büyük kayıplar oluşturabilmektedir. Bilgisayarlı görüntü işleme teknikleri endüstriyel işlemlerde karşılaşılan kusur gibi problemlerin çözümlerinde sık kullanılan yöntemlerdir. İlk olarak görüntünün elde edilmesi sonra görüntüler üzerinde kusur tespiti yapabilmek için matematiksel algoritmaların kullanılması gerekmektedir. Kusur tespitinde pikselin rengi önemli rol oynamaktadır. Özellikle komşu piksel değerleriyle kıyaslanarak kusurlu bölge tespiti yapılabilmektedir [2]. Ham görüntü üzerinde belirli algoritmalar uygulanarak görüntünün daha net duruma getirilmesi sağlanabilir. Böylece tespiti istenen nesne daha kolay tespit edilebilir [3]. Metal yüzeylerindeki kusurların erken ve gerçek zamanlı tespiti üretim performansına önemli derece olumlu etki yapmaktadır. Birçok görüntü verisinin bilgisayar sistemi üzerinde işlenmesi kusur tespit ve çözüm süresini azaltacaktır. Kusurlu ürünlerinin tespit edilmesi üretim performansı arttırmaktadır. Karmaşık ve çok olan verilerin bilgisayar destekli incelenmesi hızlı bir şekilde gerçekleştirilebilir [4]. Metal üreticileri kusurlu ürünleri tespit ederek istenmeyen kusurların sebeplerini ortadan kaldırabilir ve müşterilerine devamlı olarak kaliteli ürün temin edebilirler [5]. Haddeleme işlemi sırasında oluşan kusurların tespitine yönelik Wu, K ve arkadaşları, yaptıkları çalışmada FFT özellik çıkarımı metodu kullanılarak genetik algoritma ve yapay sinir ağları yöntemleriyle kusur tespiti yapmışlardır [6]. Xu, K ve arkadaşları ise CCD kameralar yardımıyla elde edilen gri görüntüler üzerinde arkaplan çıkarımı yöntemi kullanılarak CPU üzerinde kusur tespit uygulaması çalışmışlardır [7]. Sıcak haddehanelerde bilgisayar destekli kusurlu ürün tespitinde, çok fazla görüntünün işlenmesi, sıcaklık etkisinden ve ışık yansımasından kaynaklanan kalitesiz görüntülerin olması gibi zorluklarla karşılaşılmaktadır. Kusurlu ürün tespitinin sıcak haddeleme işlemi devam ederken insan faktörü olmayan bilgisayar destekli hızlı bir sistemle yapılması hem hata oranını düşürecek hem de gerçek zamanlı üretim sürecinde zaman kaybını engelleyecektir. Daha önceki çalışmamızda ray görüntülerindeki kusurların tespitine yönelik COLMSTD algoritması geliştirilmişti [8]. Bu algoritmanın, kusur tespiti için kullanılan 7 farklı algoritmayla kıyaslandığında %85 başarı oranıyla en iyi sonucu verdiği ortaya konulmuştu. Bu başarı oranı yanında bütün kusur tiplerine yönelik kullanılabilecek bir algoritma olduğu da görülmüştür. Gerçek zamanlı kusur tespiti, hızlı bir şekilde kusur nedenini bulmayı sağlamaktadır. Nextsense firması bu alanda çalışmalar yapmaktadır. Kardemir de ürünün tek yüzeyi üzerinden görüntü alınarak çalışmalar yapılmıştır. Tek yüzey üzerinden görüntü alınmasına rağmen, bir sonraki malzemenin haddelenmesi bitene kadar CPU işlemeyi tamamlayamamaktadır. Bütün yüzeylerden görüntü alabilmek için en az 4 kameraya ihtiyaç vardır. Bundan dolayı paralel bir sistem tarafından görüntüleri işleyen bir sistemin oluşturulması ihtiyacı ortaya çıkmıştır. COLMSTD algoritması uygun bir yöntem kullanılarak paralelleştirilirse, üretim ortamında (haddehanede) gerçek zamanlı olarak başarılı bir şekilde kullanılabileceği öngörülmektedir. Bu amaçla ekran kartının işlemci çekirdekleri kullanılarak paralelleştirme sağlanabilir [9]. Görüntü işleme, merkezi işlemci üzerinde çok büyük iş yükü oluşturduğundan paralel olarak işlenmesi performans katkısı sağlamaktadır. Spinola. G. C ve arkadaşları, paslanmaz çelik yüzeylerindeki kusur tespitini GPU üzerinde paralel işlemle gerçekleştirmiş ve 9 ile 14 kat arasında hız artışı elde etmişlerdir. Bu uygulama yassı çelik üzerinde geliştirmiş bir uygulamadır [10]. Bu çalışmamızda ise ray ve profil de birden fazla yüzeye sahip bir yapı üzerinde işlem gerçekleştirilecektir. Sadeghi M. ve arkadaşları yassı çelik üzerinde oluşan kusurların tespitinde ve sınıflandırılmasında



Şekil 1. Haddehanede ray ve profil görüntüsü elde etme (Obtaining image of rail and profile in rolling mill)

Gabor filtresinden yararlanarak %90'ın üzerinde korozyon, delik ve kırışıklıkları tespit ettiler. Bu uygulamadaki matris çarpımı aşamasını paralel uygulamayla gerçekleştirdiler [11]. Her problemin ya da işlenecek verinin paralel olarak en uygun şekilde paralelleştirilmesi gerekmektedir. Aksi takdirde beklenen performans kazancı elde edilememekte aksine performans kaybıyla karşılaşılmaktadır [12].

2. RAY VE PROFİL GÖRÜNTÜLERİNİN ELDE EDİLMESİ
(OBTAINING RAIL AND PROFIL IMAGES)

Bu çalışma, Kardemir A.Ş. haddehanelerindeki mevcut prototip sistem yardımıyla ray ve profil ürünlerinin üst yüzeyinden elde edilen görüntüler üzerinde gerçekleştirilmiştir. Prototip sistem Şekil 1'de görülmektedir. Bu sistemde rayın tek yüzeyinin (üstten) görüntüsü; haddelenmiş rayın soğutma alanına geçiş yolu üzerindeki CMOS kamera yardımıyla elde edilmektedir. Görüntüler KARDEMİR haddehanesinde çalışan bir kamera sistemi üzerinden araştırma ve inceleme amaçlı alınmıştır. Üzerinde çalışılan görüntüler 2013 -2014 yılına ait görüntülerdir. Rayın gerçek görüntüsü 416x197 piksel boyutlarındadır. Haddelenen farklı uzunluktaki raylara göre elde edilen görüntü sayısı 100-1152 arasında değişebilmektedir. Gerçek rayın görüntüsü Şekil 2'de gösterildiği gibi gri renk tonludur. Ray; taban, baş ve gövde bölümlerinden oluşmaktadır.

3. COLMSTD ALGORİTMASI
(STANDARD DEVIATION AND MEAN OF COLUMN ALGORITHM)

Görüntü işlenerek nesne tespiti istatiksel temelli, yapısal temelli, filtre temelli ve model temelli olarak dört yolla yapılmaktadır. En çok kullanılan istatiksel yaklaşım ile görüntü piksel değerlerinin dağılımları incelenerek kusur tespiti gerçekleştirilir [13]. Daha önce geliştirmiş olduğumuz COLMSTD algoritması, istatistiksel hesaplama temeline dayanmaktadır. Bu yöntemde gri piksel değerleri üzerindeki işlemler sütun temeline dayalı olarak gerçekleştirilmiştir. Aritmetik ortalama ve standart sapma kavramları kullanılarak oluşturulan COLMSTD görüntü işleme algoritmasıyla belirli bir eşik seviyesi üzerindeki gri değerler sınıflandırılmıştır [8]. Eş. 1, COLMSTD algoritmasının çalışmasını göstermektedir.

$$\begin{cases} P_{i,j} \neq 0 \; olmak \; üzere \\ if\ (P_{i,j} - S_{m,n}) \leq StSpm\ \{Not\ defect \rightarrow P_{i,j} = 0\} \\ elseif\ \begin{pmatrix} P_{i,j} - S_{m,n}) > hk\ \{\ Depht\ defect \rightarrow P_{i,j} \\ = 255\} //h = 1.8\ olmak \; üzere \end{pmatrix} \\ elseif\ \begin{pmatrix} P_{i,j} - S_{m,n}) > hk\ \{Medium\ Depht\ defect \rightarrow P_{i,j} \\ = 150\} //h = 1.5\ olmak \; üzere \end{pmatrix} \\ else\ \{Low\ Depht\ defect \rightarrow P_{i,j} = 50\} \end{cases} \quad (1)$$

Burada $P_{i,j}$ her pikselin gri değeri, *hk* ise hassasiyet katsayısıdır. $S_{m,n}$ ise sütun ortalama değeri ve *StSpm* ise sütun piksel değerlerinin standart sapmasıdır. $P_{i,j}=0$, $P_{i,j}=150$, $P_{i,j}=255$ $P_{i,j}=50$ ise COLMSTD algoritmasının sonuç piksel değerleridir. Piksel değeri sütun ortalaması değerinden küçük oldursa kusur türü çukur, büyük olursa tümsek olarak sınıflandırılır. Sütun ortalama (mean) değeri Eş. 2'de gösterilmektedir [8].

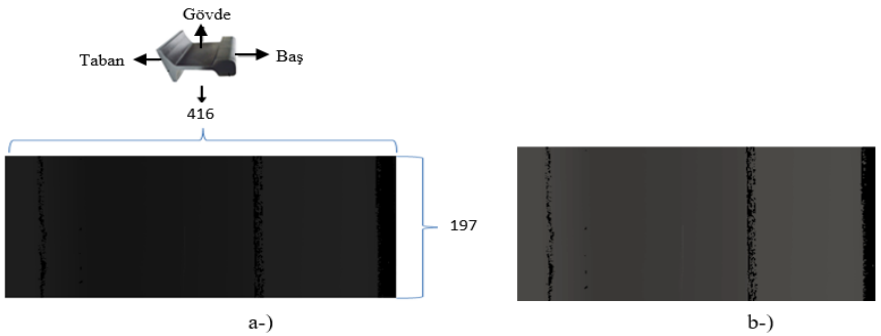
$$S_{mn} = \frac{1}{N}\sum_{i=0,j=0}^{N} P_{i,j} = \frac{P_{0,0}+P_{0,1}+P_{0,2}\ldots\ldots+P_{0,N}}{N} \quad (2)$$

Buradaki *N* değeri yükseklik değeridir. Standart sapma değeri ise Eş. 3'de ifade edilmektedir. *M* değeri görüntü genişliğidir [8].

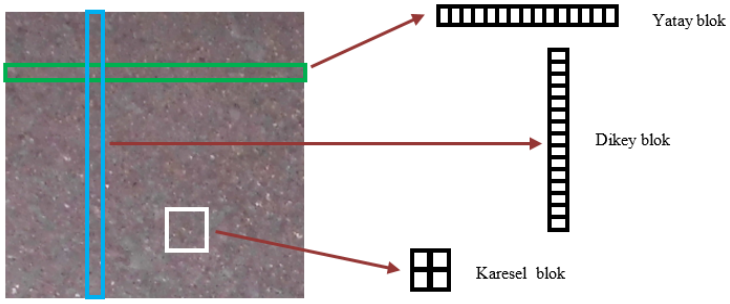
$$StSpm_{i=0}^{M} = \sqrt{\frac{1}{N}\sum_{j=0}^{N}(P_{i,j} - S_{mn})^2} \quad (3)$$

4. PARALEL OLARAK GÖRÜNTÜ İŞLEME
(PARALLEL IMAGE PROCESSING)

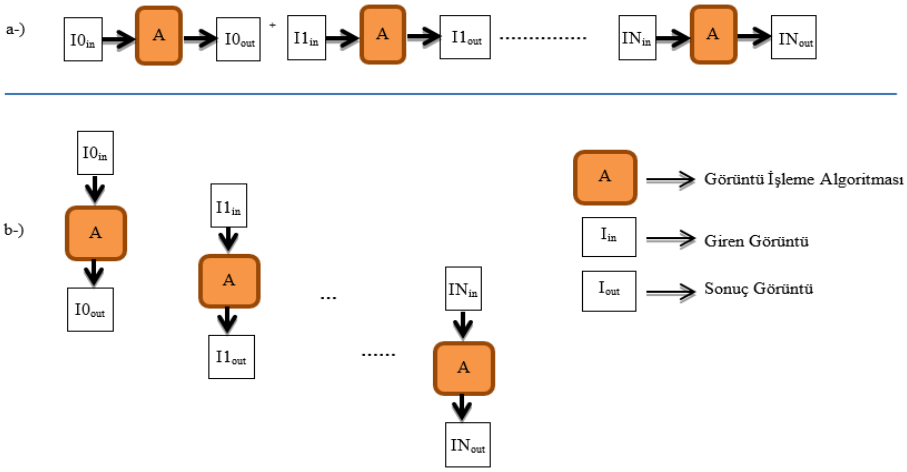
Görüntü işleme uygulamalarında, görüntü ya da görüntü parçaları piksel değerleri için, matris yapısı kullanılır. Paralel işlemede; küçük parçalara bölünen iş parçaları ayrı işlemci yapılarında çalıştırılır. Görüntü işleme yöntemi, yatay bloklar, dikey bloklar halinde veya düzgün blok parçalarına bölünerek gerçekleştirilebilir (Şekil 3). Her blok farklı işlemcilerde veya thread'lerde çalıştırılabilir. Özellikle yoğun işlem gerektiren görüntü işleme algoritması GPU yapısında paralel işlemle etkin olarak gerçekleştirilebilir [14]. Bu çalışmada; GPU yapısında problem mümkün olduğunca çok sayıda küçük parçalara bölünerek, paralel



Şekil 2. Gri ray görüntüsü ve çözünürlüğü a) Kameranın çektiği ham görüntü b) Parlaklığı %20 arttırılmış ray görüntüsü
(Gray rail image and resolution, a) Raw image obtained by camera b)Rail image having brightness increased by 20%)



Şekil 3. Paralel görüntü işlemenin türleri (Types of parallel image processing)



Şekil 4. Algoritma Çalışma Süreçleri a) Seri algoritma b) Paralel algoritma
(Processing Types of Algorithms a) Serial algorithm b) Parallel algorithm)

biçimde çözerek performans iyileştirmesi sağlanmıştır. Büyük görüntü verileri CUDA desteğiyle grafik kartı üzerindeki işlemcilerde paralel şekilde işlenebilmektedir. Seri algoritmalar sıralama tekniğine dayanmakta olup bir problemin işlem süreci bittiğinde sıradaki diğer işleme geçilir. Paralel algoritmalar ise aynı anda birçok işlemin gerçekleştirebilmesine olanak sağlar. Şekil 4'de seri ve paralel algoritmaların yürütülme aşamaları görülmektedir. Seri görüntü işleme CPU üzerinde, paralel sistemler ise ortak bellekli bir donanım olan GPU üzerinde uygulanabilmektedir. N adet I_i görüntüsü üzerinde, işlem seri olarak uygulandığında sırasıyla her I_i görüntüsünden sonra I_{i+1} görüntüsüne geçilerek devam edilmektedir (Şekil 4.a). En son görüntü işlendikten sonra işlem süreci tamamlanmaktadır. Görüntü işleme GPU çekirdek sayısı değişken (dinamik) seçilerek paralelleştirildiğinde ise (Şekil 4.b), müsait olan çekirdek sayısı kadar görüntü I_i, ortak bellek kullanılarak işlenmektedir.

Seri olarak görüntü işleme basamakları şu şekildedir:

1- Bütün görüntülerinin matris biçiminde Host'a yüklenmesi
2- Görüntü sayısına kadar döngünün başlaması
3- Sırasıyla görüntülerin CPU ya gönderilmesi
4- CPU üzerinde her görüntü işleme algoritmasının uygulanması
5- Görüntü üzerinde kusurlu bölgelerin gösterilmesi

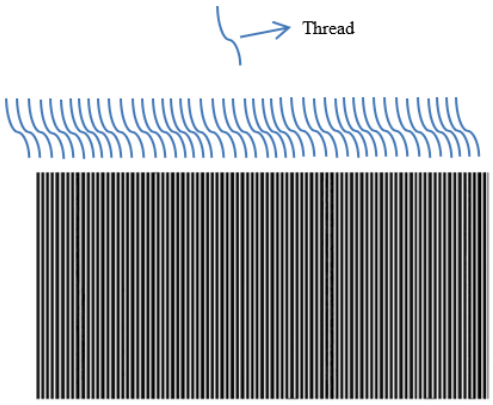
Paralel görüntü basamakları ise aşağıdaki şekildedir:

1- Bütün ray görüntülerinin matris biçiminde Host'a yüklenmesi
2- Görüntü sayısına kadar döngünün başlaması
3- Görüntülerin GPU'ya aktarılması
4- GPU üzerinde her görüntü üzerinde algoritmanın paralel olarak uygulanması
5- Görüntülerin Host'a yüklenmesi
6- Görüntü üzerinde kusurlu bölgelerin gösterilmesi

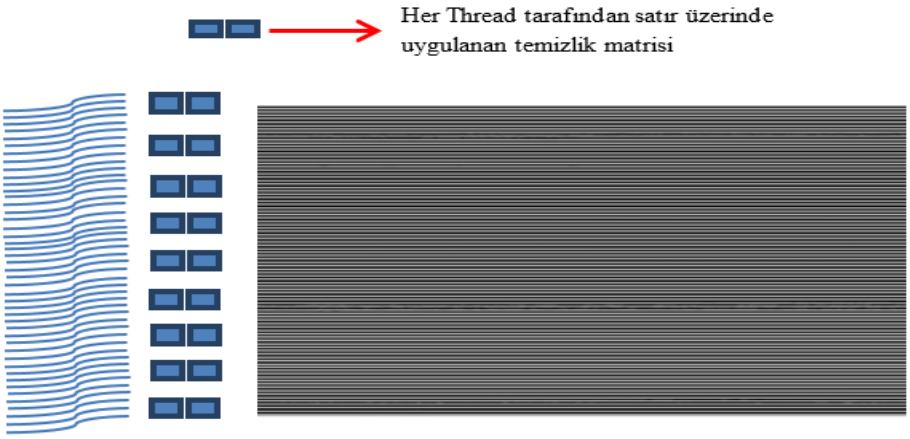
Paralel işlemede kernel görevlendirmesi yapılırken görev sırası önem arz etmektedir. Görev alacak çekirdekler algoritma hesaplama sırasına göre yapılmadığı takdirde aynı kaynak kullanımı gereksiniminden dolayı performans kayıpları oluşabilmektedir.

5. COLMSTD ALGORİTMASININ PARALEL UYGULAMASI
(PARALLEL APPLICATION OF COLMSTD ALGORITHM)

Daha önceki çalışmalarımızda görüntü işlemenin ilk aşaması olan istenmeyen piksellerin paralel olarak temizlenmesi (ön temizleme) başarıyla gerçekleştirilmiştir [15]. Bu çalışmada ise istatiksel tabanlı olarak kusurlu bölgelerin tanınması ve kusur özelliği taşımayan istenmeyen piksellerin temizlenmesini içeren adımların paralelleştirilmesini



Şekil 5. Threadlerin yaptığı sütun işlemleri (Column operations performed by threads)

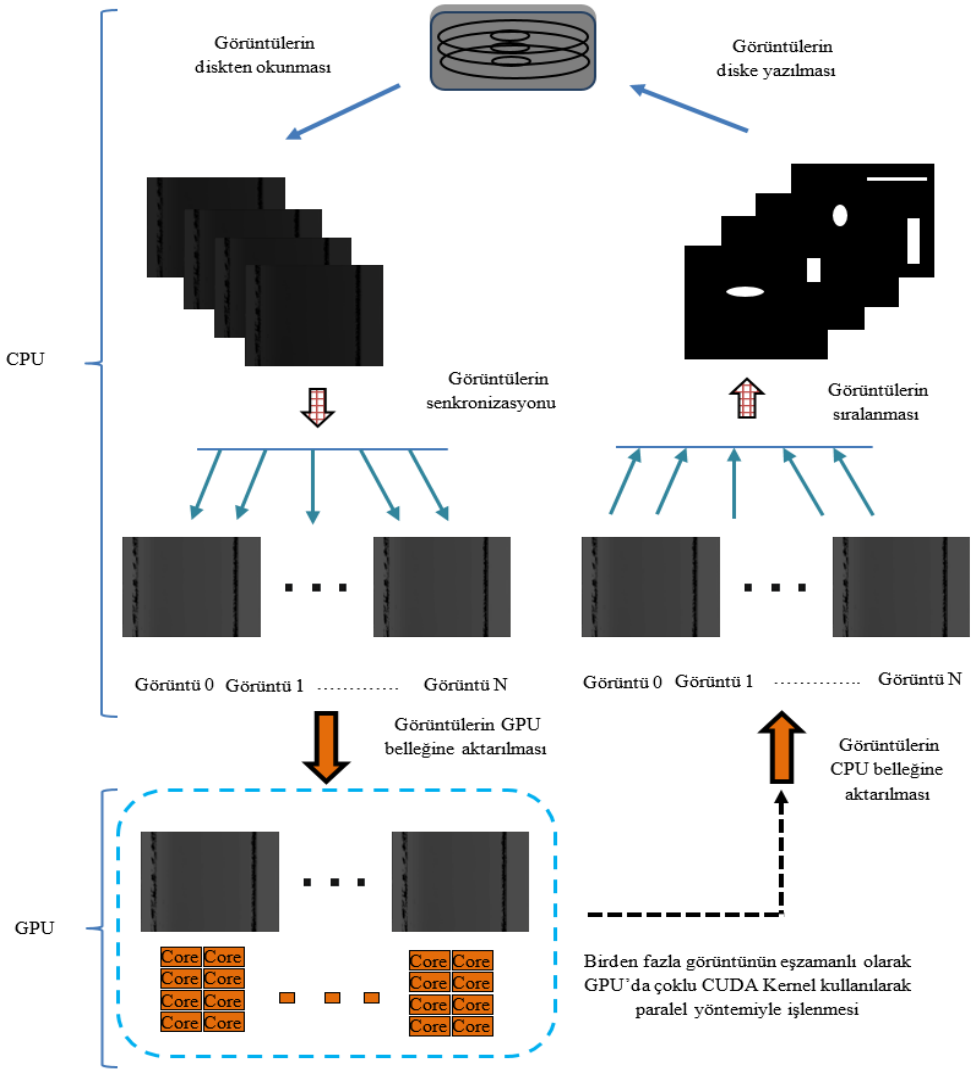


Şekil 6. Threadlerin yaptığı satır işlemleri (Row operations performed by threads)

sağlanmaktadır. Bu üç adımdaki her bir algoritmanın başarılı bir şekilde paralel olarak geçekleştirilmesi sistemin tamamının paralel bir sitemde uygulanabilirliğini ortaya koymaktadır. Kusur tespiti yapan kısım, sütun temelli olarak paralel çalışmayla gerçekleştirilmektedir. Şekil 5 üzerinde gri ray görüntüsünün her bir sütununun threadler tarafından işlendiği görülmektedir. Gri ray görüntüsü 416x197 çözünürlüğe sahip olduğundan ilk aşamada 416 thread görevlendirilmiştir. Bu threadlerin her biri, bir sütun için ağırlıklı ortalama ve standart sapma değerlerini hesaplayacaktır. Sonra standart sapma değerlerini dikkate alarak piksel gri değer renklendirmesi gerçekleştirecektir. Renkli olan bölgeler kusurları temsil edecektir. Şekil 6 üzerinde ikinci aşama olan temizlik işlemi gerçekleştirilecektir. Bu aşamada işlemler satır temelli olarak gerçekleşmektedir. Temizlik işleminde 1x2 boyutlarında bir matris kullanılmaktadır. Bu matris her satır içinde ayrı thread tarafından gezdirilir ve 2 birim genişliğe sahip olmayan pikseller yok edilir [16]. Görüntünün threadler tarafından işlenmesi daha iyi performans değeri oluşturmaktadır. Ancak performans değeri hesaplama işlemlerine, döngülere ve mantıksal işlemlere bağlıdır. GPU, çoklu thread ve çoklu işlemci çekirdeği yardımıyla ortak bellekli yüksek derece paralellik sağlamaktadır [17]. Ray görüntülerindeki kusurları tespit etmek için iki farklı paralel yöntem çalışma kapsamında değerlendirilmiştir.

1- Tek bloklu, çok görüntü işleme yöntemi (Şekil 7).
2- Çok bloklu, tek görüntü işleme yöntemi (Şekil 8).

Bu yöntemlerin farkı, görüntü işleme aşamasındaki kullandıkları blok ve işlenen görüntü sayılarıdır. Yöntem-1 de tek blok, Yöntem-2 de ise 2 ile 32 arasında blok kullanılmıştır. Her görüntü bloğu ayrı bellek kullanmaktadır. İşlem büyüklüğüne bağlı olarak belleğe okuma yazma erişimi gerekmektedir. Ayrıca ortak kullanılan bellek alanlarına erişmede senkronizasyon işlemine gerek duyulmaktadır. Bu sürelerin her birinin minimize edilmesi paralel olarak gerçekleştirilecek olan işlemlerin performansını arttırmaktadır [18]. Yöntem 1: Şekil 7'de de gözükmekte olduğu gibi bu yöntemde aynı anda birden fazla görüntünün eş zamanlı olarak işlenmesi sağlanmaktadır. Çekirdekte 512 thread görevlendirilir ve her görüntü 197x1 boyutunda blok parçalarına ayrılır. Her çekirdekte bir görüntü işlenmektedir. Çekirdekler içindeki threadlerin her biri 197x1 boyutundaki istatistiksel hesaplamaları gerçekleştirerek COLMSTD algoritmasını yürütür. Yöntem-2: Bu yöntemde GPU üzerinde her defada tek görüntü işlenmektedir (Şekil 8). Farklı blok adetleri kullanılarak işlemler gerçekleştirilebilmektedir. Bu aşamadan sonra tespit edilen kusurlu görüntü sabit diske kaydedilmektedir. Bir sonraki görüntü üzerinde aynı işlemler yapılmaktadır. Bu çalışmada OpenCV açık kaynak kodlu görüntü işleme



Şekil 7. Çok kernel kullanarak aynı anda çok görüntü işleme (Yöntem-1)
(Image processing simultaneously using multi kernel (Method -1))

kütüphanesi ve CUDA birlikte kullanılmıştır. OpenCV açık kaynak kodlu görüntü işleme kütüphanesi Visual Studio C++ üzerine eklenmiş ve daha sonra CUDA arayüzü C++ üzerine eklenmiştir. C++ kod bölümünden hem CPU üzerinde OpenCV hem de GPU üzerinde OpenCV kütüphaneleri çağrılarak işlemler gerçekleştirilmiştir.

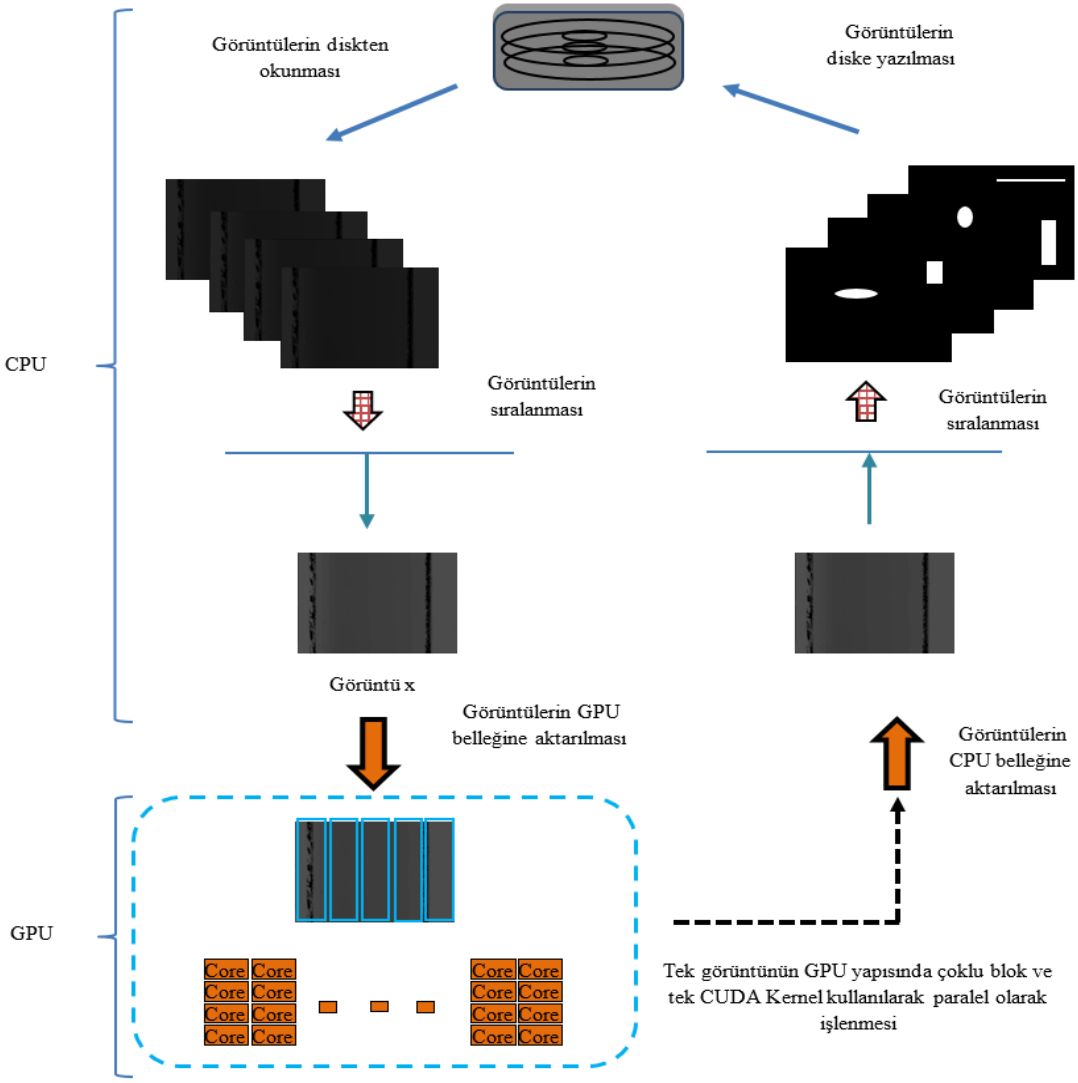
7. SONUÇLAR VE TARTIŞMALAR
(RESULTS AND DISCUSSIONS)

Yapılan çalışmada ray ve profil yüzeylerindeki kusurların tespiti paralel sistem üzerinde elde edilmiş ve sonuçları seri sistem ile kıyaslanmıştır. Her iki sistemin, 1-1152 adet görüntü üzerinde, COLMSTD algoritmasını çalıştırma süre performansları değerlendirilmiştir.

Bu çalışmada, paralel işlemlerin gerçekleştirilmesi için NVIDIA Geforce 9800 GT ekran kartı kullanılmıştır. Bu kart, 112 çekirdek, 1 GB bellek ve 600 MHz işlemci frekansı özelliklerine sahiptir. Seri olarak algoritmanın çalıştırılması ise Intel Core 2 Duo işlemciye sahip CPU üzerinde sağlanmıştır. Bu işlemcide, 2 core, 3,17 GHz frekans ve 2 GB ana bellek özellikleri mevcuttur.

7.1. GPU üzerinde Paralel Yöntemlerin Performansı
(The Performance of Parallel Methods Running on GPU)

Yöntem-1, kullanılarak geliştirilen sistemin görüntü ve kullanılan kernel sayısına bağlı olarak çalıştırma zaman değerleri Tablo 1'de gösterilmektedir. Maksimum 32 adet kernel kullanılarak denemeler yapılmıştır. GPU'da eş zamanlı işlenen görüntü sayısı arttıkça, çalıştırma süre değerleri azalmakta, buna bağlı olarak da performans kazancı elde edilmektedir. 32 görüntünün aynı anda işlenmesi durumunda en iyi performans elde edilmiştir. Yöntem-2 kullanılarak geliştirilen sistemin, görüntü ve blok sayısına bağlı olarak çalıştırma zaman değerleri Tablo 2 de gösterilmektedir.



Şekil 8. Tek kernel kullanarak tek görüntü işleme (Yöntem-2) (Image processing using single kernel (Method -2))

Tablo 1. Çoklu CUDA kernel ve Çoklu görüntü sayısına göre Paralel COLMSTD algoritmasının çalışma performansı (Zaman birimi saniyedir)
(Running Performance of Parallel COLMSTD algorithm by multi CUDA kernels and multi images (Time unit in seconds))

CUDA Kernel Sayısı	Görüntü Sayısı							
	32	64	128	256	512	768	1024	1152
32	0,109	0,217	0,433	0,864	1,727	2,597	3,468	3,905
16	0,11	0,222	0,442	0,88	1,765	2,666	3,545	4,105
8	0,115	0,229	0,458	0,914	1,833	2,757	3,684	4,153
4	0,125	0,245	0,491	0,985	1,980	2,972	3,978	4,542
2	0,139	0,277	0,557	1,115	2,243	3,880	4,540	5,145
1	0,171	0,345	0,676	1,374	2,783	4,193	5,621	6,376

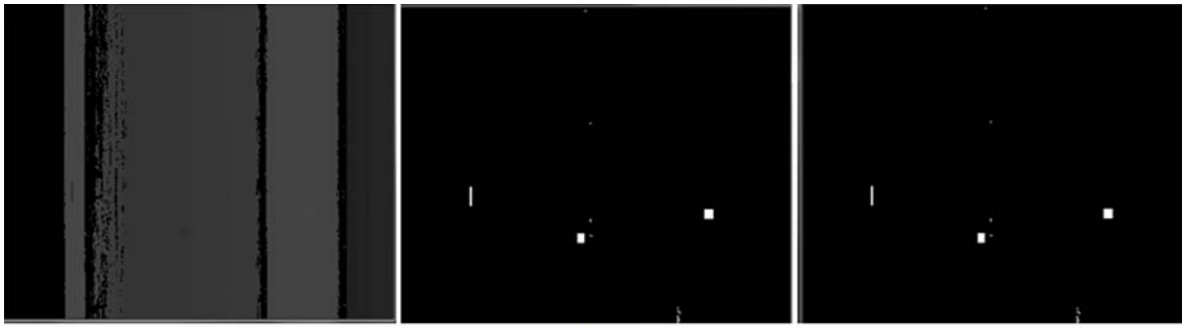
Bu çalışmada 32'li, 16'li, 8'li, 4'lü ve 2'li bloklar kullanarak denemeler yapılmıştır. GPU üzerinde işlenen blok sayısı arttıkça çalıştırma süre değerleri de artmaktadır. Buna bağlı olarak GPU'da 1 görüntünün, 1 blok halinde işlenmesi durumunda, en iyi performans elde edilmiştir. Blok sayısının artması çalıştırma zaman değerlerini arttırmıştır. Yöntem-1 de daha az bellek erişimine ihtiyaç duyulduğundan dolayı daha iyi performans elde edilmiştir.

Tablo 2. Çoklu CUDA Blok ve Tekli görüntü sayısına göre Paralel COLMSTD algoritmasının çalışma performansı (Zaman birimi saniyedir)
(Running Performance of Parallel COLMSTD algorithm by multi CUDA blocks and single image (Time unit in seconds))

CUDA Block Sayısı	Görüntü Sayısı							
	32	64	128	256	512	768	1024	1152
32	1,121	2,122	4,412	8,846	17.551	26,289	35,082	39,501
16	0,656	1,215	2,555	5,101	10,15	15,186	20,264	22,889
8	0,38	0,692	1,481	2,958	5,585	8,844	11,815	13,311
4	0,205	0,401	0,82	1,651	3,299	4,982	6,678	7,555
2	0,176	0,343	0,706	1,412	2,625	4,278	5,741	6,485
1	0,171	0,345	0,676	1,374	2,783	4,193	5,621	6,376

Tablo 3. Yöntem-1 ve Yöntem-2'nin kıyaslanması (Comparation of method -1 and method -2)

Görüntü Sayısı	Yöntem-1	Yöntem-2	Hızlanma Oranı (Yöntem-2 / Yöntem-1)
32	0,109	1,121	10,28
1152	3,905	39,501	10,12



Şekil 9. CPU ve GPU işlemcilerinin kusur tespit kabiliyetleri
a) Parlaklığı %20 Arttırılmış Ray Görüntüsü b) CPU'nun tespit ettiği kusurlar c) GPU'nun tespit ettiği kusurlar
(Defect detection capability by CPU and GPU a) Brightness increased by 20% b) Defects detected by CPU c) Defect detected by GPU)

Tablo 3'de 32 ve 1152 görüntü için Yöntem-1 ve Yöntem-2'nin performansının kıyaslanması görülmektedir. Yöntem-1 kullanılarak yaklaşık 10 kat daha hızlı kusur tespiti yapılmaktadır.

7.2. GPU ve CPU'nun Kusur Tespit Kabiliyetlerinin Kıyaslanması
(Comparison of Defect Detecting Capabilities of GPU and CPU)

Grafik İşlemci Birimi (GPU) üzerinde görüntülerin işlenmesi ile kusur tespiti sağlanmış ve paralel algoritmanın başarısını ölçebilmek için sonuçlar Merkezi İşlemci Birimi (CPU) üzerinde seri olarak gerçekleştirilen COLMSTD algoritmanın sonuçları ile kıyaslanmıştır. Yapısında çizgisel, noktasal ve bölgesel kusur bulunan bir görüntünün işlenmesinde iki yaklaşımın da aynı sonuçlar elde ettiği görülmektedir (Şekil 9). Bu sonuç, paralel COLMSTD algoritmasının kusur tespitini seri olarak gerçekleştirilen algoritma ile aynı başarı seviyesinde sağladığını göstermektedir. Tablo 4 üzerinde; COLMSTD algoritmasının CPU üzerinde seri olarak ve GPU üzerinde paralel olarak çalıştırıldığında noktasal, çatlak ve bölgesel kusur tiplerini aynı başarı oranıyla tespit ettiği görülmektedir.

Tablo 4. Seri ve paralel algoritmanın kusur tespit performansı
(Defect detection performance of serial and parallel algorithms)

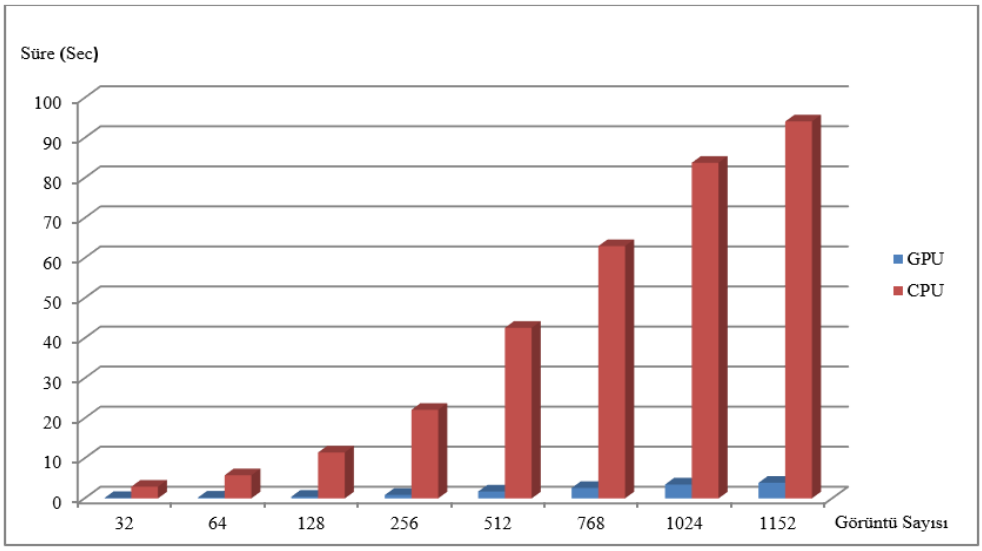
Kusur Tipleri	Seri Algoritma	Paralel Algoritma
Noktasal	%92	%92
Çatlak	%100	%100
Bölgesel	%58	%58

7.3. CPU ve GPU İşlemcilerinin Süre Performans Kıyaslaması
(Time Performace Comparison of CPU and GPU Processors)

Bu çalışmada görüntülerin alındığı haddehanede azami 72 metre uzunluğuna sahip ray ve profil üretilebilmektedir. En

Tablo 5. CPU ve GPU üzerinde algoritma çalıştırma değerleri (The running time of algorithm on CPU and GPU)

Görüntü sayısı	GPU (sec)	CPU (sec)	Hızlanma Oranı
32	0,109	2,912	26,72
64	0,217	5,778	26,63
128	0,433	11,469	26,49
256	0,864	22,125	25,61
512	1,727	42,612	24,67
768	2,597	63,006	24,26
1024	3,468	83,814	24,17
1152	3,905	94,122	24,10



Şekil 10. CPU ve GPU üzerinde algoritma çalıştırma performans grafiği (Perfomance graphic of algorithm on CPU and GPU)

uzun ürün üretiminde CMOS kamera yardımıyla 1152 gri görüntü elde edilmektedir. COLMSTD algoritmasının GPU üzerinde çalışması ile elde edilen değerler CPU üzerindekilerle kıyaslanmış ve sonuçlar Tablo 5'de gösterilmiştir. GPU üzerinde paralel olarak işleme, daha iyi performans gösteren Yöntem-1 kullanılarak gerçekleştirilmiştir. 32 den 1152 ye kadar değişik sayılarda görüntü işlenmiş ve performans süreleri saniye olarak belirtilmiştir. Tablo 5'de yer alan değerlere göre GPU ortalama olarak 25,30 kat daha hızlı kusur tespiti yapabilmektedir. Bu sadece çalıştırma işlemi dikkate alındığındaki değerlerdir. En yüksek başarı oranı, 32 görüntü üzerinde 26,72 kat daha hızlı performans ile elde edilmiştir. En düşük başarı oranı ise 24,10 değeriyle 1152 görüntü üzerinde gerçekleştirilmiştir. Tablo 5'den elde edilen değerlerin grafiği Şekil 10 üzerinde gösterilmiştir. Şekil üzerinde, görüntü sayısı artıkça, GPU ve CPU işlem sürelerinin de doğrusal olarak arttığı görülmektedir. Elde edilen sonuçlara göre GPU üzerinde paralel olarak kusur tespiti algoritmasının çalıştırılması ile oldukça iyi performans elde edildiği görülmüştür. Paralel olarak algoritmanın çalıştırılması aşamasında her bir görüntünün okunması 0,001 saniye ve yazması 0,002 saniye zaman almıştır. COLMSTD algoritmasının her bir görüntüyü işleme süresi ise yaklaşık 0,004 saniyedir. Ürünün, maksimum hız olan 7 m/s ile kamera önünden geçmesi durumunda, 1 saniyede 112 görüntü elde edilmektedir. 112 görüntünün tümü kusurlu olduğu varsayıldığında, görüntüler, okuma ve yazma dahil 0,784 saniyede işlenebilmektedir. Kusurlu olmayan görüntülerin diske yazılmasına gerek olmadığından sistemin çalışması daha kısa süre almaktadır.

8. SONUÇLAR (CONCLUSIONS)

Bu çalışmada, Kardemir A.Ş. ye ait haddehanede üretilen ray ve profillerin yüzey kusurlarını tespit etmek için geliştirilen COLMSTD algoritması paralel olarak gerçekleştirilmiş ve performansı seri COLMSTD algoritması ile kıyaslanmıştır. Paralel yöntemlerin sağladığı performans katkısı görüntü işleme hesaplamalarında çok net ortaya çıkmaktadır. Ray ve profil görüntülerinin çok fazla olmasından dolayı performans artışı GPU kullanılarak elde edilmeye çalışılmıştır. Sonuçlar COLMSTD algoritmasının, ray ve profil yüzeylerindeki kusurları, GPU üzerinde hızlı bir şekilde tespit edebileceğini ortaya çıkarmıştır. COLMSTD algoritmasının paralelleştirilmesi performans kazanımı oluşturmuştur. Böylece paralel bir sistem üzerinde iş paylaşımı yapılarak COLMSTD algoritmasıyla hızlı bir

şekilde kusur tespitinin yapılabileceği sonucu elde edilmiştir. Bu çalışmada incelenen COLMSTD algoritmasının çalıştırma performans değerleri dikkate alındığında GPU üzerinde ortalama 25,30 kat daha hızlı işlem yapılabileceği görülmüştür. Bu çalışmada işlenecek görüntülerin tek tek veya birden fazlasının aynı anda paralel olarak yürütülmesi incelenmiştir. Aynı anda birden fazla görüntünün paralel olarak işlenmesinin daha iyi performans gösterdiği sonucuna varılmıştır. Bu sonuçta, bellek okuma-yazma (özellikle ana belleğe erişim) ihtiyacının aynı anda çok görüntü işleme yönteminde daha az olmasının etkisi olduğu değerlendirilmiştir.

Elde edilen maksimum 0,784 saniyelik bütün aşamaları içeren işlem süresi, geliştirilen bu sistemin gerçek zamanlı olarak uygulanabilirliğini kanıtlamaktadır. Mevcut sistemde sadece üstten görüntü elde edildiğinden sadece ray ve profil yüzeylerinin üst yüzeyindeki kusurlar tespit edilmiştir. Bu aşamadan sonra ray ve profil yüzeylerinin tümü (yan yüzeyler ve taban yüzeyi) paralel sistem üzerinde işlenerek kusur tespiti yapılabilir. Ancak haddehanedeki çevresel etkiler (sıcaklık, buhar ve ışık vb.) azaltılarak, daha kaliteli görüntü elde etme ortamının oluşturulması gerekmektedir.

9. SEMBOLLER (SYMBOLS)

$P_{i,j}$	Görüntü piksellerinin gri değeri
I_{in}	Giren görüntü değeri
I_{out}	Sonuç görüntü değeri
A	COLMSTD algoritması

Kısaltmalar (Abbreviations)

CPU	Merkezi İşlemci Birimi
GPU	Grafik İşlemci Birimi
COLMSTD Algoritma	Sütun Ortalama Standart Sapma Temelli
CUDA	Tümleşik Hesaplama Aygıt Mimarisi
OpenCV	Açık Kaynak Kodlu Görüntü Kütüphanesi

TEŞEKKÜR (ACKNOWLEDGEMENT)

Bu çalışmaya katkılılarından ve desteklerinden dolayı KARDEMİR A. Ş. yönetici ve çalışanlarına teşekkür ederiz.

KAYNAKLAR (REFECENCES)

1. Henry Y.T.N., Grantham K.H.P., Nelson H.C.Y., Automated Fabric Defect Detection - A Review, Image and Vision Computing, 29, 442-458, 2011.
2. Funck J.W., Zhong Y., Butler D.A., Brunner C.C., Forrer J.B., Image Segmentation Algorithms Applied to Wood Defect Detection, Computers and Electronics in Agriculture, 41, 157-179, 2003.
3. Kasım Ö., Kuzucuoğlu, AE., Detection and Classification of Leukocyte Cells From Smear Image, Journal of The Faculty of Engineering and architecture of Gazi University, 30 (1), 95-109, 2015.
4. Calle F.J., Bulnes F.G., Garcia D.F., Usamentiaga R,, Molleda J., A Parallel Genetic Algorithm for Configuring Defect Detection Methods, Latin America Transactions, 13, 1462-1468, 2015.
5. Yazdchi M., Yazdi M., Mahyari A.G., Steel Surface Defect Detection Using Texture Segmentation Based on Multifractal Dimension, International Digital Image Processing Conference, Bangkok, Thailand, 346-350, 2009.
6. Wu G., Zhang H., Sun X., Xu J., Xu K., A bran-new Feature Extraction Method and Its Application to Surface Defect Recognition of Hot Rolled Strips, International Conference on Automation and Logistics, Jinan, 2069-2074, 2007.
7. Xu K., Yang C., On-line Defect Detection Algorithms for Surface Inspection of Hot Rolled Strips, Mechanic Automation and Control Engineering (MACE), Wuhan, 2350-2353, 2010.
8. Orak İ.M., Çelik A., An Algorithm (COLMSTD) for Detection of Defects on Rail and Profile Surfaces, International Journal of Computer Science and Information Security (IJCSIS), 14 (4), 45-50, 2016.
9. Gebali F., Algorithms-Parallel-Computing, New Jersey, USA: John Wiley Sons, 2011.
10. Spinola G.C., Canero-Nietro M. J., Bonelo M. J., Real-time Image Processing for Edge Inspection and Defect Detecion in Stainless Steel Production Lines, Imaging Systems and Techniques, Penang, Malaysia, 170-175, 2011.
11. Sadeghi M., Soltani H., Zamanifar K., Application of Parallel Algorithm in Image Processing of Steel Surfaces for Defect Detection, Cumhuriyet University Science Journal, 36, 263-173, 2015.
12. Jaja J., An Introduction to Paralel Algoritms, Redwood City, USA: Addison Wesley Longman Publishing, 1992.
13. Xianghua X., A Review of Recent Advances in Surface Defect Detection Texture Analysis Techniques, Electronic Letters on Computer Vision and Image Analysis, 3, 1-22, 2008.
14. Weimer D., Thamer H., Thoben D. K., GPU Architecture for Unsupervised Surface Inspection Using Multiscale Texture Analysis, Procedia Technology, 15, 278-284, 2014.
15. Orak M. I., Çelik A., Parallel Noise Removing Process on Hot Rolled Rail Images for Defect Detection, Global Journal on Technology, 136-143, 2015.
16. Shapiro L., Stockman G., Computer Vision, Washington, USA: Pearson, 2001.
17. Vandal N.A., Savvides M., CUDA Accelerated Illumination Preprocessing on GPUs, 17th IEEE Digital Signal Processing (DSP 2011), Corfu, Greece, 1-6, 2011.
18. Nickolls J., Buck I., Garland M., Skadron K., Scalable Parallel Programming with CUDA, Queue - GPU Computing, 6 (2), 40-53, 2008.