T.C.
KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İPLİK YAPIMINDA KULLANILAN PAMUK İÇERİSİNDEKİ YABANCI MADDELERİN GPU KULLANILARAK SEZGİSEL BULANIK MANTIK İLE TESPİT EDİLMESİ

EYÜP YALÇIN

YÜKSEK LİSANS TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

KAHRAMANMARAŞ 2012

T.C.
KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İPLİK YAPIMINDA KULLANILAN PAMUK İÇERİSİNDEKİ YABANCI MADDELERİN GPU KULLANILARAK SEZGİSEL BULANIK MANTIK İLE TESPİT EDİLMESİ

EYÜP YALÇIN

Bu tez,
Elektrik-Elektronik Mühendisliği Anabilim Dalında
YÜKSEK LİSANS
derecesi için hazırlanmıştır.

KAHRAMANMARAŞ 2012

Kahramanmaraş Sütçü İmam Üniversitesi Fen Bilimleri Enstitüsü öğrencisi Eyüp Yalçın tarafından hazırlanan **"İPLİK YAPIMINDA KULLANILAN PAMUK İÇERİSİNDEKİ YABANCI MADDELERİN GPU KULLANILARAK SEZGİSEL BULANIK MANTIK İLE TESPİT EDİLMESİ"** adlı bu tez, jürimiz tarafından 10 / 07 / 2012 Tarihinde oy birliği ile Elektrik-Elektronik Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Yrd. Doç. Dr. Mahit GÜNEŞ (DANIŞMAN) …….
Elektrik-Elektronik Mühendisliği, KSÜ

Yrd. Doç. Dr. Mustafa ŞEKKELİ (ÜYE) …….
Elektrik-Elektronik Mühendisliği, KSÜ

Doç. Dr. M. Metin KÖSE (ÜYE) …….
İnşaat Mühendisliği, KSÜ

Yukarıdaki imzaların adı geçen öğretim üyelerine ait olduğunu onaylarım.

Prof. Dr. M. Hakkı ALMA …….

Fen Bilimleri Enstitüsü Müdürü

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Eyüp YALÇIN

Bu çalışma KSÜ-BAP tarafından desteklenmiştir.
Proje No:2011/ 3-31 YLS

Not: Bu tezde kullanılan özgün ve başka kaynaktan yapılan bildirişlerin, çizelge, şekil ve fotoğrafların kaynak gösterilmeden kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

İPLİK YAPIMINDA KULLANILAN PAMUK İÇERİSİNDEKİ YABANCI MADDELERİN GPU KULLANILARAK SEZGİSEL BULANIK MANTIK İLE TESPİT EDİLMESİ

ÖZET

Tekstil ve çırçır fabrikalarında kullanılan yün ve pamuk hammaddelerinde üretim ve işleme esnasında oluşan veya dışarıdan gelen yabancı maddeler, elde edilen kumaş veya ipliklerin kalitesinde önemli ölçüde düşüşe neden olmaktadır. Pamuk içerisindeki bu yabancı maddeler, genel olarak siyah pigment lifleri, naylon ve sentetik türden ambalaj artıkları, boya maddeleri ve bitkileri toplama esnasında gözden kaçan yabancı maddelerden oluşabilir.

Günümüzde tekstil ve tekstil işletmelerinde kullanılan metotlar, yabancı maddelerin ayırt edilebilmesi noktasında, hız ve kalite açısından verimli değildir. Son yıllarda geliştirilen bilgisayarlı görüş sistemleri, her alanda olduğu gibi tekstil alanında etkisini belli bir biçimde göstermektedir. Günümüzde kullanılan sistemlerde, öne çıkan sorunlardan kalite ve hız sorunları, önemli bir konu olarak karşımıza çıkmaktadır. Bu çalışmada, kameradan alınan görüntüdeki yabancı maddelerin görüntü içerisinden belirlenmesi işlemi için sezgisel bulanık mantık algoritması kullanılmıştır. Sezgisel bulanık mantık algoritmasının yapısı gereği, CPU (Central Processing Unit) tabanlı uygulamalarda hız problemlerinin olduğu görülmüştür. Bu nedenle, hız sorununu ortadan kaldırmak için GPU (Graphics Processing Unit) teknolojisi kullanılmıştır. Alınan görüntü üzerinde ilk aşamada, sezgisel bulanık mantık algoritmasında kullanılan eşik değeri için, görüntünün sayısal değerleri üzerinden dinamik eşik değeri üreten Otsu algoritması kullanılmıştır. Kameradan alınan her çerçevenin eşik değeri gerçek zamanda hesaplanmış ve görüntüye güncel olarak uygulanmıştır. Uygulanan bu algoritmalar, NVIDIA GTX 480 GPU destekli ekran kartı kullanılarak maksimum 262 kat hızlandırılmıştır. Sistem geliştirilirken yapılan denemelerde yakalanan görüntünün farklı çözünürlükteki örnekleri kullanılarak hem performans hem de kalite açısından, en uygun çözünürlük değerinin 640*100 olduğu görülmüştür. İşlenen bu görüntü, 80*100 çözünürlük değerinde, 8 eşit bölgeye ayrılmıştır. Bir bölgede yabancı madde olup olmadığına, ilgili bölgede kalan görüntü parçasının sayısal değerlerinin toplamı değerlendirilerek karar verilmiş ve ihmal edilebilirlik düzeyine göre yapılan denemeler sonucunda bir eşik değeri belirlenmiştir.

Yapılan denemeler sonucunda kullanılan metotların istenilen sonuçları eksiksiz olarak verdiği tespit edilmiştir. Bu metotlar sayesinde önceki çalışmaların aksine bilgisayarlı görüş sistemi hız ve kalite olarak çok daha önemli bir avantaj sağlamıştır.

Anahtar Sözcükler: *CUDA, GPU, Sezgisel Bulanık Mantık, Otsu Metodu, Pamuk Yabancı Madde Temizlenmesi, Grafik İşlemci*.

DETECTION OF FOREIGN SUBSTANCES IN COTTON USED FOR PRODUCTION OF YARN BY USING GPU AND INTUITIONISTIC FUZZY LOGIC METHOD

ABSTRACT

Foreign substances from outside or occurred during the production and processing of raw materials used for cotton and wool in textile and cotton gin factories cause to a significant decrease in the quality of the resulting fabric or yarn. Generally, these foreign substances consist of black pigment fibers, nylon and synthetic types of packaging waste, paint materials and foreign fibers from plants overlooked during collection.

Methods used todays are not sufficient for the extraction of the foreign substances quality and speed of processing in the textile industries. As in all areas in recent years, developments in the field of computer vision systems also show the effect of the textiles area. The quality and speed problems in existing systems used today are important selective factors. In this study, the intuitive fuzzy logic algorithm was used for the process of determination foreign substances from the images captured by camera. In this study, the Intuitionistic fuzzy logic algorithm was used for the process of determining foreign substances in image captured by camera within the images. As in the nature of Intuitionistic Fuzzy Logic Algorithm, the problems of speed are detected in CPU-based applications. Therefore, GPU technology is used to eliminate the problem of speed. At the first stage, Otsu algorithm that produces a dynamic threshold value over the numerical value of the image is used for the threshold value of the Intuitionistic Fuzzy Logic on the captured image. The threshold value of each frame captured from camera is calculated in real time and applied currently to the image. These applied algorithms are accelerated to 262-fold at maximum by using a graphics card supported with NVIDIA GeForce GTX 480 GPU. It is seen that the most appropriate resolution in terms of both performance and quality is 640*100 by using samples of different resolutions of the captured image value in experiments at the development of the system. This processed image is divided into 8 equal areas in resolution of 80 * 100. Whether foreign material stays in an area is determined on assessing the sum of numerical values of image part in the concerned area and a threshold value is determined as a result of experiments made in accordance with the level of negligence.

The result of the experiment shows that the used methods provide the wanted results completely. These methods allows for a greater advantage in terms of speed and quality contrast to previous studies with computerized vision system.

Key Words: *CUDA, GPU, Intuitionistic Fuzzy Logic, Otsu Method, Foreign Material Extraction from Cotton, Graphics Processor.*

TEŞEKKÜR

Yüksek lisans öğrenimim süresince danışmanlığımı yapan ve çalışmalarım sırasında desteğini esirgemeyen, Sayın Hocam Yrd. Doç. Dr. Mahit GÜNEŞ' e çok teşekkür ederim.

Yüksek lisans boyunca desteklerini esirgemeyen, Sayın Prof. Dr. Hasan Rıza ÖZÇALIK'a teşekkürü borç bilirim.

Yüksek lisans tez sürecinde, beraber çalışma olanağı bulduğum değerli arkadaşlarım Sayın Hasan BADEM ve İsmail EREN'e teşekkürlerimi sunarım.

Ayrıca, Yüksek Lisans Tez çalışmalarım esnasında maddi destek veren K.S.Ü. Bilimsel Araştırma Projeleri Birimi'ne (Proje No: 2011/ 3-31 YLS) teşekkür ederim.

Eyüp YALÇIN

ANNEME ve BABAMA…

İÇİNDEKİLER

Sayfa No

SİMGELER ve KISALTMALAR DİZİNİ

GPU : Graphics Processing Unit

CPU : Central Processing Unit

CUDA : Compute Unified Device Architecture

SDK : Software Development Kit

API : Application Programming Interface

NVCC : NVIDIA C Compiler

PTX : Parallel Thread Execution

Tpar : Paralel İşlem Süresi

Tse : Seri İşlem Süresi

RAM : Random Access Memory

DRAM : Dynamic Random Access Memory

GDDR5 : Graphic Double Data Rate 5

PCI : Peripheral Component Interconnect

AGP : Accelerated Graphics Port

IDE : Integrated Development Environment

RGB : Red Green Blue

SBM : Sezgisel Bulanık Mantık

NB : Negatif Büyük

SI : Sıfır

PK : Pozitif Küçük

PB : Pozitif Büyük

PO : Pozitif Orta

NO : Negatif Orta

MOM : Mean Of Maxima

COA : Center Of Area

IFE : Intuitionistic Fuzzy Entropy

IFD : Intuitionistic Fuzzy Divergence

FPS : Frame Per Second

OPENCV : Open Source Computer Vision Library

SM : Streaming Multiprocessor

ŞEKİLLER DİZİNİ

Sayfa No

ÇİZELGELER DİZİNİ

Sayfa No

KOD DİZİNİ

1. GİRİŞ

Pamuk, boyu 80-120 cm arasında değişen, yetişmesi için 7-9 haftalık süre gereken bir bitkidir. Yapılan tarihi kazılarda M.Ö. 5000 yılından günümüze kadar, pamuğun tekstil ürünlerinde ve kumaş üretiminde kullanıldığı görülmüştür. Pamuk, giydiğimiz elbiselerde, halılarda vb. tekstil ürünlerinde kullanılan temel hammaddedir. Pamuğun diğer materyallerden daha çok kullanılmasının nedeni olarak temizlenmesi kolay, dayanıklı, ucuz ve kolay boyanabilir olması gösterilmektedir. Bu nedenle, pamuğun insan hayatı üzerine etkisi ve önemi tartışılamazdır.

Pamuk endüstrisi, üretkenliğe ve fiyata bağlı olarak daralan küresel piyasalarda değişim gösteren çok önemli bir paya sahiptir. Pamuk temizleme endüstrisindeki son 10 yıldaki gelişmelerle, bu alandaki verim ve kalite hızlı bir ivme kazanmıştır. Daha önceleri pamuk içindeki yabancı maddelerin temizlenmesi, işçiler tarafından sağlandığından oldukça pahalı ve kalite açısından yetersiz bir çözümdü. 18. yüzyılda Eli Whitney'in icat ettiği pamuk temizleme makinesi ile atılan temelde, bugün tonlarca pamuğun ayıklanması yapılabilmektedir [1].Pamuk ipliği üretim tesislerinde, ürün kalitesi ve üretim maliyetleri işletmelerin karlılığını ve rekabet gücünü etkileyen en önemli faktörlerdendir. Her işletme öncelikle ürün kalitesini yükseltmeye ve üretim maliyetlerini en aza indirmeye çalıştırmaktadır. Üretim maliyetleri içinde, pamuk fiyatı ve pamuğun temizlenme işlemleri önemli bir yer tuttuğundan, işletmeler bu maliyetleri düşürme yollarını aramaktadır. Bu nedenle özellikle pamuk harmanı ve üretimi işleminde, başta telef olmak üzere ilgili sorunların en aza indirilebilmesi için yabancı madde miktarı az (toz, toprak, yaprak kırıntıları, vs.) kaliteli pamuk gruplarının kullanılması gerekmektedir. Bu nedenle işletmelerde, gelen pamuk balyalarının her biri üzerinde çeşitli fiziksel testler yapılarak üretilecek ipliklerin kalite özelliklerini bozmadan telef oranını en aza indirebilecek sistemlere ihtiyaç duyulmaktadır [2].

Pamuktan yabancı madde temizleme işlemlerinde temel unsur, statik elektriklenmeyi önleyip, daha sonra pamuk içerisindeki yabancı maddeleri çeşitli mekanik ve elektronik gereçlerle tespit etmektir. Bu şekilde elde edilen pamuğun kalitesi artmaktadır. Bu makinelerde, pamuk yumuşak bir akış sayesinde ayıklama makinesine akarak, daha düşük maliyet ile çabuk ve hızlı temizleme yapılarak verimi arttırmaktadır. İşletmelerde, yabancı madde ile temiz pamukların da atık olarak değerlendirilmesi temel sorundur. Bu sorunun çözümü işletme verimi açısından çok önem arz etmektedir [3].

Pamuktaki yabancı maddenin temizlemesi işlemleri için son yıllarda çeşitli sistemler geliştirildiği yapılan literatür araştırmalarında görülmektedir [1] [4] [5] [6]. Literatür araştırmalarındaki geliştirilen sistemler gerçek zamanlı olduğu için, kameradan alınan görüntülerin analiz edilip sonuç elde edilinceye kadar geçen süreyi en uygun kullanmakta bazı sorunlarla karşılaşıldığı görülmüştür. Örneğin, temizlenecek pamuk yığınından en iyi sonucu elde etmek için yavaş çalışacak şekilde tasarlanan bir sistem, birim zamanda üretilen pamuk miktarını azaltacağından üretim tesisinin kar oranını etkilemektedir. Diğer seçenek olarak hızlı çalışacak şekilde tasarlanan sistemlerde ise pamuk içinde yabancı maddelerin kalabilmesine yol açmaktadır. Bu şekilde tasarlanan sistemde de iplik kalitesi düşeceğinden yine üretim tesisinin kar oranını etkilemektedir.

Bu çalışmada, yukarıda bahsedilen sorunun çözülebilmesi için son zamanlarda çok hızlı şekilde gelişmekte olan GPU teknolojisi ile elde edilen görüntülerin analizi için paralelleşebilecek uygun algoritmaların geliştirilmesi düşünülmektedir.

2. ÖNCEKİ ÇALIŞMALAR

2.1. Pamuktan Yabancı Madde Ayıklanması ile İlgili Önceki Çalışmalar

Ajay Pai ve arkadaşları yapmış oldukları çalışmada; pamuk yığınındaki yabancı maddelerin ayıklanması için X-ray tarayıcı kullanarak elde ettikleri görüntüleri bilgisayara aktarmışlardır. Mikro Tomografi yöntemiyle yapmış oldukları bu çalışmada elde edilen görüntüleri her seviyede 6-10 arasında örnek alarak inceleyip, örneklerin bulanık mantık algoritması ile grafiğini çıkartmışlardır. Daha sonra elde edilen grafikten örnek içindeki yabancı cismi tespit etmeyi başarmışlardır [5].

Wenzhu Yang ve arkadaşları yapmış oldukları çalışmada; tarayıcıdan elde ettikleri 40 farklı görüntüleri 1700x2344 boyutlarına getirdikten sonra RGB renk uzayındaki resimleri gri renk uzayına getirip parçalı görüntülerin histogram analizini yapmışlardır. Daha sonra Otsu metoduyla görüntülerin eşik değerlerini ve gelişim hızlarını arama yöntemiyle optimize ettiler. Optimize edilmiş resimleri matematik histogramından geçirip oluşan pik sayısı ve görüntülerine göre yabancı maddenin cinsini veya biçimini tespit etmeyi başardılar [6].

Boshra D.Farah yapmış olduğu çalışmada; pamuk örneklerini 175x125 mm'lik bir alana 2 g pamuk örneğini eşit olarak dağıtarak, bu görüntüleri geçirilen ışık altında, **y**ansıyan ışık altında ve her ikisini beraber kullanmak üzere 3 farklı ölçüm yöntemiyle örneği incelemiştir. Aldığı resim görüntülerini video kart yardımıyla bilgisayara aktararak analiz yapmaya çalışmıştır. Alınan görüntüleri 256x256'lık resimler olarak binary formatına getirmiş ve kontrol örneği ile ölçülen örnek arasındaki renk histogramından hareketle eğer birden fazla pik noktası oluşmuşsa örnekte yabancı madde bulunduğuna dair kesin karar kılmıştır [7].

QU Xin, DING Tian-Huai yapmış oldukları çalışmada; pamuk içerisindeki yabancı maddeleri ayıklamak için kameradan alınan görüntülerin 2D-Wavelet transform yardımıyla RGB (Red Green Blue) renk uzayından YCrCb görüntü uzayına dönüşümünü sağladılar. Elde ettikleri görüntülere alçak, yüksek ve kompound filtreler uygulayarak örneğin içindeki yabancı maddeleri tespit etmeyi başardılar [8].

Ronghua Ji ve arkadaşları yapmış oldukları çalışmalarında; yüksek hızlı çizgi tarayan kamera kullanarak alınan örnekleri bilgisayar ortamına aktardıktan sonra, 10 farklı örneği 4000x500 boyutlarında resim dosyası olarak oluşturdular. Alınan bu resimlerin boy

oranı, yuvarlaklığını hesaplayıp, Support Vector Machine (SVM) algoritması kullanılarak sınıflandırılmasıyla %92 oranında başarı sağlamışlardır [9].

Yupent Zhang, Philip W.Smith yapmış oldukları çalışmada; CEILAB ve RGB ile alınan örnekleri 16.5x16.5 cm alanda 60 piksel çözünürlükle ve 3 MB boyutunda depoladıktan sonra alınan bu resimlerin her bir pikseline farklı ağırlık değerleri verilmiştir. Elde edilen bu örnek daha sonra Bayezian Weighted K-Means (BWKM) algoritmasıyla her bir pikselin komşuluğundaki değerle Öklid bağlantısı kullanılarak minimum uzaklığı hesaplayıp kazanan pikseli güncelleyerek yabancı maddelerin taramasını yapmışlardır [10].

2.2. Bulanık Mantık Teoremiyle İlgili Önceki Çalışmalar

Murali Siddaiah ve arkadaşları yapmış oldukları çalışmada; CDD renkli kamera ile aldıkları görüntüleri RGB renk uzayından HLS renk uzayına 640x480 piksel şeklinde dönüştürdüler. 25 tane farklı yabancı madde örneğini geri beslemeli yapay sinir ağları yöntemiyle öğretip, kameradan alınan görüntülerle beraber bulanık denetim uygulayarak karşılaştırdıktan sonra alınan örneklerin içindeki yabancı maddeleri sınıflandırmayı başarmışlardır. Daha sonra Fuzzy C ve ANFIS ile bu yöntemleri tekrar uyguladılar. Alınan sonuçlara göre %98 başarı ile ANFIS en başarılı yöntem olurken, yapay sinir ağları %93 ve Fuzzy C algoritması %86 başarı ile sonuçlanmıştır [4].

André Bigand ve Olivier Colot, yapmış oldukları çalışmada; görüntü işlemede gürültülü olarak elde edinilen bir görüntünün bulanık mantık kuramı ile histogram değerlerine göre temizlenebileceğini bir uygulama ile ortaya koyarak bulanık mantığın görüntü işlemede etkinliğini ortaya koymuşlardır [11].

2.3. Sezgisel Bulanık Mantık Teoremiyle İlgili Önceki Çalışmalar

Chaira T ve Ray A. K yapmış oldukları çalışmada; görüntü işleme yöntemlerinden kenar çıkartma (edge detection) işlemini, sezgisel bulanık küme (intuitionistic fuzzy set) teoremi tabanlı olarak geliştirmiştir. Yapılan bu çalışmada, görüntü işleme ile uzaklık ölçümünde sezgisel bulanık küme teoremi ile başarılı sonuçlar elde edilmiştir [12].

Milind M. Mushrif ve Ajoy K. Ray, yapmış oldukları çalışmada; renklere göre görüntü bölümleme (color image segmentation) için sezgisel bulanık küme (A-IFS Atanassov's intuitionistic fuzzy set) teoremini kullanarak görüntüde histogram renk ayarlarını daha kaliteli bir şekil elde eden bir algoritma geliştirmiştir [13].

2.4. GPU Teknolojiyle İlgili Önceki Çalışmalar

Kimberly Powell yapmış olduğu çalışmada; gerçek zamanlı uygulamalarda sistemin cevap verme süresini geciktiğine değinmiş ve bu sorunun GPU etkinliğini ortaya koymuştur [14].

Hiren Patel, yaptığı çalışmada; gerçek zamanlı olarak görüntü işlenme tekniklerinden görüntü polarize etme işlemini yaklaşık 600 kat hız elde etmiştir. Bu sayede Swap olayını azaltmıştır [15].

Jarno Mielikainen ve arkadaşları tarafında geliştirilen Hava Araştırma ve Tahmin modelinde, bulutların, aerosollerin, moleküllerin ve yüzeyin dağılımlarını ve emilmelerini hesaplamışlardır. GPU tabanlı Goddard kısa dalga şeması, CPU tabanlı tek iş parçacıklı benzerleriyle karşılaştırılırken hesaplama alanı 422* 297 yatay grid noktadan 34 dikey seviyede hesaplanmıştır. Tek hassas hesap ve daha az doğru hesap kullanılarak, hızlandırmalar I/O ile 536 kat, I/O olmadan 259 kat hızlandırılmıştır [16].

Jarno Mielikainen ve arkadaşları yaptıkları çalışmalarında; hava araştırma ve tahmin işlemi olan Stony Brook University şemasını optimize etmek için GPU tabanlı bir sistem geliştirmişlerdir. Bu sistemde mevcut duruma göre 422 ile 297 arasında değişen hız performansı sağlamışlardır [17].

Ashwin M. Aji ve arkadaşları yaptıkları çalışmada; biyomedikal üzerine yaptıkları çalışmalarında, genetik haritaların çıkartılmasında mevcut sistemlerini 14.5 kat hızlandırmışlardır [18].

.

3. MATERYAL ve METOT

3.1. Materyal

3.1.1. Donanımsal Materyaller

Geliştirilen sistemde 3 farklı donanım kullanılmıştır. Bunlar;

- Dijital Kamera
- Frame Grabber Kartı
- GPU Destekli Ekran Kartı (CUDA)

3.1.1.1. Dijital Kamera

Geliştirilen sistemde, analiz edilecek pamuk yığınlarının görüntülerini elde etmek için kullanılan araçtır. Sistem de kullanılan kamera 1920 x 1080 [16:9] çözünürlüğe ve 50 fps yakalama özelliğine sahiptir [19].

3.1.1.2. Frame Grabber Kart

Frame grabber kartlar, bir analog sinyali ya da dijital video yayınından çerçeveleri yakalayıp onu dijital çerçevelere dönüştüren bir elektronik cihazdır. Bu cihaz genellikle video yayınlarından aldığı görüntüleri depolama, görüntüleme ya da sıkıştırma işlemleri için kullanılır. USB, Ethernet ve IEEE 1394 bağlantıları kullanılarak bilgisayar ile bağlantı kurulabilir. Geliştirilen sistemde dijital video kameradan alınan görüntünün yakalanması için kullanılan karttır [20].

3.1.1.3. Gpu Destekli Ekran Kartı (Cuda)

Ekran kartı, bilgisayar üzerinde harici ya da ana kartta dâhili olarak bulunabilen PCI (Peripheral Component Interconnect), PCI-Express gibi bağlantı biçimlerine sahip görüntü çıkış birimidir.

GPU (Graphics Processing Unit) ise bu ekran kartların özelliklerinden birisi olup, yüksek çözünürlüklü video performansı isteyen uygulamalar ve bilimsel araştırmalar için kullanılan aygıtlardır. Modern GPU kartları yüksek grafik verisi işleme ve gösterme hızları, paralel yapıları nedeniyle CPU'lara göre çok yüksektir. GPU kayan nokta ve benzeri işlemleri hızlandırmak için bilgisayarın kendi CPU'sundan hariç ekran kartına eklenmiş bir tip CPU'dur. GPU'lar PCI-Express veya AGP (Accelerated Graphics Port) üzerinden ana karta bağlanabilmektedir.

Geliştirilen sistemde kullanılan NVIDIA firmasının ürettiği; 1,5 GB DDR5 bellek, 134.490 GB/s bandwidth ve 2 cache belleğe sahip olan GTX 480 ekran kartı kullanılmıştır.

3.1.2. Yazılımsal Materyaller

Geliştirilen sistemde 4 farklı yazılım kullanılmıştır. Bunlar;

- Microsoft Visual Studio 2010
- Cuda Toolkit
- OpenCV
- Cmake

3.1.2.1. Microsoft Visual Studio 2010

Microsoft Visual Studio, Microsoft tarafından uygulama geliştirmek için geliştirilen tümleşik bir IDE (Integrated Development Environment) 'dir. Windows, Windows Mobile, .NET Framework ve Microsoft Silverlight v.b. desteklenen bütün platformlar için Win32 ve Win64 tabanlı uygulamalar geliştirmek için kullanılır [21].

Microsoft Visual Studio özünde herhangi bir programlama dili desteklemek yerine VSPackage olarak kodlanmıştır. Microsoft Visual Studio ile web tabanlı uygulamalar, veri tabanı uygulamaları, form tasarımı, paket programları gibi çok yönlü uygulamalar geliştirilebilir. Dahili dil olarak C/C++, VB.NET, C#, F# içermektedir [21].

Projede Microsoft Visual Studio 2010 Professional sürümü kullanılmıştır. Bu IDE'ye eklenti olarak kullanılabilen Cuda Toolkit aracılığıyla sistem geliştirilmiştir.

3.1.2.2. Cuda Toolkit

CUDA Toolkit, NVIDIA tarafından CUDA destekli ekran kartları ile bütünleşik uygulamalar geliştirmek için çıkartılan yazılım geliştirme platformdur [22].

Geliştirilen sistemde görüntü işleme basamaklarını GPU (CUDA) tabanlı kısımlarının çalıştırılabilmesi için kullanılan platformdur.

3.1.2.3. OpenCV

OpenCV (Open Source Computer Vision Library) Intel desteği ile geliştirilen Windows, Linux, Mac OS X, PSP gibi farklı platformlarda çalışabilen, C ve C++ programlama dili ile yazılmış, görüntü işleme ve gerçek zamanlı bilgisayarlı görme işlemleri için kullanılabilen açık kaynak kodlu bir kütüphanedir. OpenCV'nin görüntü

işleme, tıbbi görüntüleme, robotik ve güvenlik gibi çok farklı alanlarda kullanımı yaygındır. OpenCV'de kullanılan yüzlerce fonksiyon işletim sisteminden bağımsız olarak çalışmaktadır. Bu sayede gerçek zamanlı uygulamalarda kullanıldığında geliştirilecek sistemler için maksimum verim sağlamaktadır. OpenCV lisansında belirtildiği üzere ticari kullanımında ürünün kodları açılmaksızın ücretsiz kullanılabilmektedir. Ayrıca, OpenCV 2.2 sürümünden itibaren CUDA desteği ile kullanıma sunulmuştur [23].

Geliştirilen sistemde, gerek CPU tabanlı gerekse GPU tabanlı kısmında, frame grabber kartından yakalan görüntüleri elde etmek için OpenCV 2.3.1 kullanılmıştır.

3.1.2.4. Cmake

Cmake, derleyiciden bağımsız olarak, birden fazla kütüphane ve bu kütüphanelerin Visual Studio ile beraber kullanılmaları için oluşturulmuş bir uygulamadır. Cmake lisansında belirtildiği üzere ücretsiz bir yazılımdır [24].

Geliştirilen sistemde;

- Kod yazmak için Microsoft Visual Studio 2010
- Derleyici olarak Cuda Toolkit
- Görüntü işleme kütüphanesi olarak OpenCV

kullanıldığından, bu yazılımların bir arada çalışabilmesi için, bağlayıcı olarak Cmake 2.8.4 sürümü kullanılmıştır.

3.2. Metot

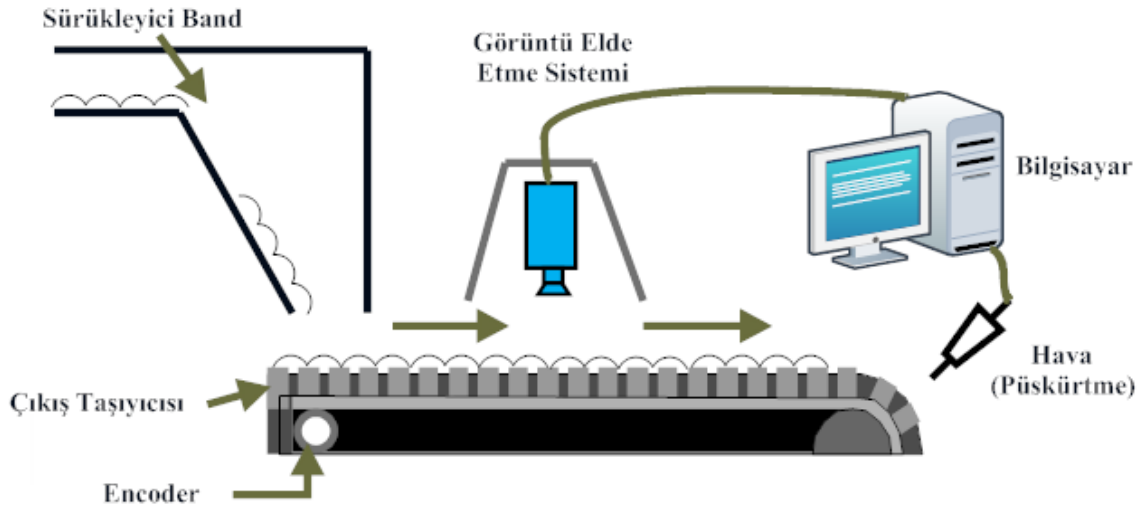
Dijital kamera ile alınan görüntüler frame grabber kartı ile yakalanarak geliştirilen yazılımlar aracılığıyla işlenmektedir. Geliştirilen yazılımlar, sezgisel bulanık mantık ve Otsu metodunun beraber kullanılmasıyla, dijital kameradan yakalanan görüntüdeki, yabancı maddenin olduğu alan veya alanlar belirlenmektedir. Belirlenen alan veya alanlar, sürücü devre aracılığıyla yabancı madde ayıklama bandındaki pistonları harekete geçirerek pamuk yığınındaki yabancı maddeler ayıklanmaktadır. Şekil 3.1'de, sistemin genel blok diyagramı görülmektedir.

Geliştirilen sistem, bazı alt birimler ve metotlardan oluşmaktadır. Bunlar;

- Mekatronik Sıralama Sistemi
- Makine Görüş Sistemi
- Video Görüntüsü Alma
- Görüntü İşleme Algoritması

şeklindedir.

3.2.1. Mekatronik Sıralama Sistemi



Şekil 3.1 Mekatronik Sıralama Sistemi

Şekil 3.1'de sistemin genel yapısı görülmektedir. Bu yapıya göre akış bandından gelen pamuk yığınlarının dijital kamera vasıtasıyla görüntüsü gerçek zamanlı alındıktan sonra frame grabber kartı ile çerçeve olarak RAM'e kaydedilmektedir. RAM'e alınan bu görüntüler GPU destekli ekran kartı ile işlendikten sonra yabancı madde bulunan ilgili bölgenin bilgilerine bağlı olarak sürücü devre vasıtasıyla pistonların kontrolü sağlanmaktadır. Bu şekilde gelen görüntü üzerindeki sadece ilgili alanda bulunan yabancı maddenin pamuk yığınından temizlenmesi sağlanmaktadır.

3.2.2. Makine Görüş Sistemi

Şekil 3.2 Görüntü Alma Blok Diyagramı

Makine görüş sistemi, dijital kameradan alınan görüntülerin frame grabber kartı ile yakalanarak bilgisayar ortamına aktarılması sürecine kadar olan aşamaları kapsamaktadır. Sistem geliştirilirken, birim zamanda işlenebilecek en optimal çözünürlük, gerek elde edilen görüntü kalitesi gerekse de birim zamanda analiz edilecek pamuk yığınının birim zamandaki işlem hızı dikkate alındığında 640x100 olduğu tespit edilmiştir. Görüntünün bu boyutlarda seçilmesinin en önemli nedenlerinden birisi de, işlem yapılacak görüntüyü 8 parçaya bölerek, görüntüdeki pamuk yığınının içerisinde yabancı maddenin bulunduğu ilgili parçanın pamuk yığından atılmasını sağlamaktır. Bu sayede, görüntüdeki pamuk yığınının tamamını üretim bandından çıkartmayarak üretim maliyetini düşürmüş olmaktadır.

3.2.3. Video Görüntüsü Alma

Dijital kameradan alınarak görüntünün gerek CPU tabanlı gerekse GPU (CUDA) tabanlı yazılım geliştirilebilmesi için frame grabber kartında yakalanan görüntünün yazılım geliştirme platformuna aktarılması gerekmektedir. Bu işlemin gerçekleştirilmesi için OpenCV adlı görüntü alma kütüphanesi kullanılmıştır. Bu kütüphane, görüntünün alınması, alınan görüntünün bütün özelliklerine erişilebilmesi, CUDA platformuna entegre çalışabilmesi ve yazılım geliştirmek için kullanılan programlama dili olan C dilini desteklemesi açısından en ideal kütüphanedir.

```
CvCapture* capture = 0;
capture  = cvCaptureFromCAM(1);
IplImage* videoFrame = NULL;
videoFrame = cvQueryFrame(capture);
```

Kod 3.1 OpenCV Görüntü Yakalama Kodu

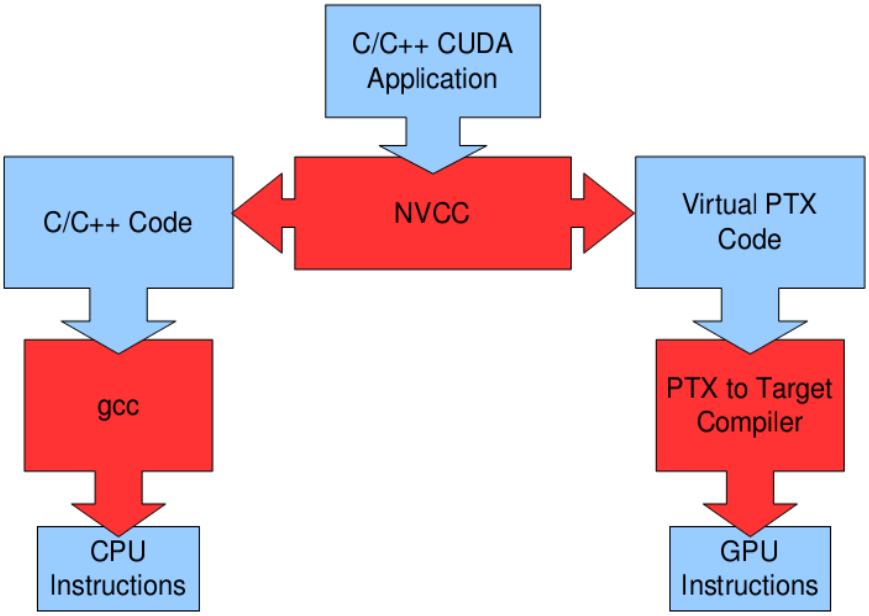
OpenCV kütüphanesi aracılığıyla görüntü yakalamanın temel kullanımı Kod 3.1'de görülmektedir. Sisteme bağlı dijital kameradan görüntüyü yakalamak için cvCaptureFromCAM yapısının parametresi 1 olarak ayarlanması gerekmektedir. Daha sonra ise videoFrame adlı boş bir pencere açıp, kameradan alınan görüntülerin burada depolanması sağlanmaktadır. videoFrame nesnesiyle hem CPU tabanlı hem de GPU tabanlı uygulamalarda görüntü işleme fonksiyonlarında görüntünün sayısal değerleriyle işlem yapabilmemizi sağlamaktadır [23].

3.2.4. Görüntü İşleme Algoritması

3.2.4.1. CUDA Temelli Programlama

GPU tabanlı yazılım geliştirmek için kullanılan Direct3D ve OpenGL platformları ile sınırlı sayıda GPU tabanlı uygulama geliştirilebilmektedir. Bu nedenle, Stanford Üniversitesinde grafik laboratuvarında başlatılan bir proje olan BrookGPU ile C programlama dili tabanına sahip bir platform geliştirilerek bu sorun çözülmek istendi [25]. Bu proje ekibinin NVIDIA firmasına katılmasıyla ilk programlanabilir GPU destekli ekran kartı olan GeForce 8800 GTX ürünü, kullanıcıların hizmetine sunuldu. 2007 yılında, bu ürünle birlikte firma tarafından ilk CUDA SDK (Software Development Kit) sürümü hizmete sunuldu. Bu sayede, NVIDIA CUDA'nın temelini atmış oldu.

CUDA, NVIDIA'nın GPU programlamak için C diline eklentilerle beraber kullanılabilen GPU programlama API (Application Programming Interface) 'sidir. Bu API, GPU'nun yapısından kaynaklanan paralelliği en rahat ve en verimli şekilde kullanmak için kullanıcılara sunulmuştur. CUDA API aracılığıyla uygulama geliştirebilmek için, sistemde bir adet NVIDIA GPU kartına ve ilgili kartın sürücülerine ihtiyaç vardır. CUDA API'leri sayesinde kullanıcılar, Stream Processing ve hafızalara sorunsuz erişim sağlamaktadır. Geliştirilmek istenen bir uygulamanın tamamı CUDA üzerinden çalıştırılamaz. Çünkü NVCC (NVIDIA C Compiler) derleme aşamasında kaynak kodlarını, CPU ve GPU kodları olarak ayırt etmek üzere çalışmaktadır.

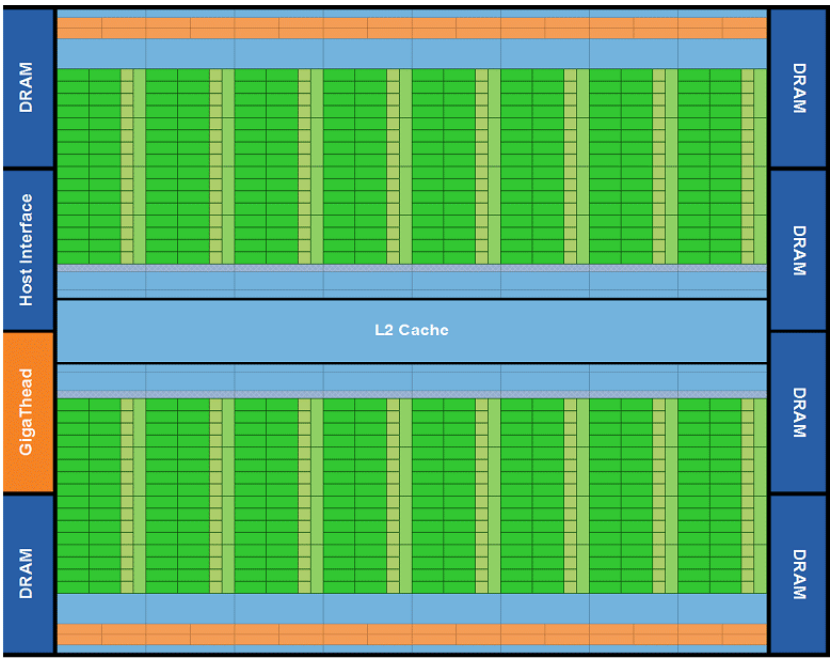


Şekil 3.3 Cuda Derleme İşlemleri [26]

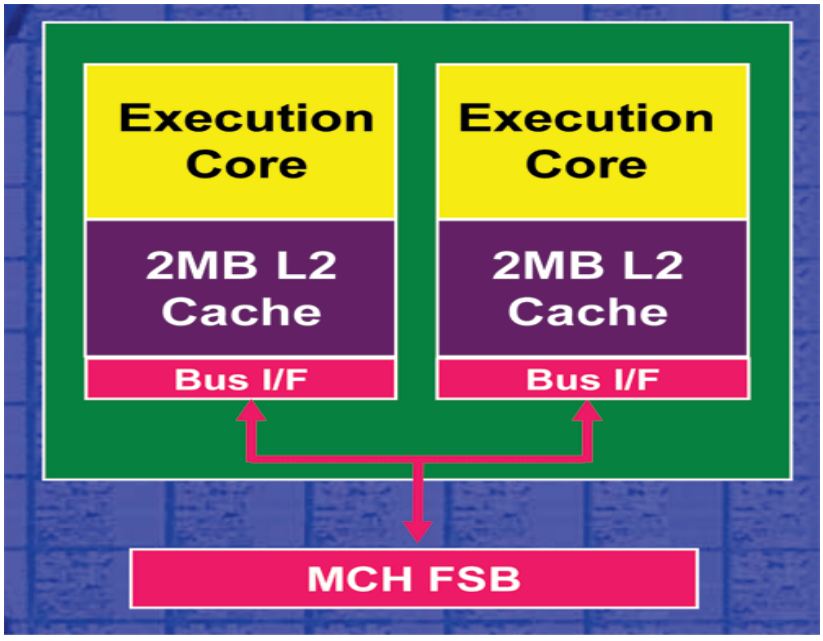
Şekil 3.3'de ".cu" uzantılı kaynak kodlarının NVCC derleyicisi ile derlenme işlemleri görülmektedir. NVCC derleyicisi gerekli derleyici araçlarını çağırarak, "cu" uzantılı kaynak kodları barındıran dosyanın derlenmesiyle, C kodu (CPU) ve PTX (Parallel Thread Execution) kodu (GPU) olmak üzere iki çıkış oluşturmaktadır. CUDA platformunda; CPU kodları C derleyicinde, PTX kodları ise CUDA Toolkit kullanılarak derlenmektedir. CUDA platformunda uygulama geliştirilirken, kullanılan GPU'nun hesaplama kapasitesine dikkat edilmesi gerekmektedir. Örneğin atomik işlemler ve çift hassasiyetli işlemler her versiyonda çalışmaz. Bu tür PTX kodları çalıştırabilmek için hesaplama kapasitesi, atomik işlemler için en az 1.1, çift hassasiyetli işlemler için ise 1.3 olan versiyonu kullanmak gerekir. Geliştirilen program, üst GPU versiyonlarda çalıştırılabilir ancak alt GPU versiyonda her zaman çalıştırılamayabilir. Bu yüzden CPU programlamadan farklı olarak CUDA ile program geliştirilirken donanımın özelliklerini ve kabiliyetlerini bilmek gerekmektedir.

Bu aşamada CUDA mimari yapısı ve programlama hiyerarşisi tanıtılacaktır.

3.2.4.1.1 CUDA Mimari Yapısı



Şekil 3.4 GPU Mimari Yapısı [27]



Şekil 3.5 CPU Mimari Yapısı [28]

Şekil 3.4 ve Şekil 3.5'te geliştirilen sistemde kullanılan CPU (Intel Dual-Core E5700) ve GPU (NVIDIA GTX 480) mimari yapıları görülmektedir. GPU mimari yapısında, en küçük dikdörtgenlerin her biri CUDA çekirdeklerini temsil etmektedir. Bu çekirdeklere, paralel işlemci denilmektedir. CUDA çekirdekleri 32'şerli gruplar halinde toplanmıştır ve bunlara Streaming Multiprocessor (SM) denilmektedir. Bu SM'ler iş

parçacıklarını 32'li gruplar halinde alırlar. Bu iş parçacıklarından oluşan bloğa da "warp" denilmektedir. Geliştirilen sistemde kullanılan GTX 480 ekran kartında 16 adet SM bloğu bulunmaktadır. Toplam çekirdek sayısı ise 480'dir.

GTX480 ekran kartlarında L1 ve L2 adında iki tane önbellek bulunmaktadır. L2 önbelleği, Şekil 3.4'te görülen birimler için ortak kullanımdadır. NVIDIA tarafından geliştirilen daha önceki ekran kartı mimarilerinde 16 KB önbellek bulunurken, GTX 480'de ise 64 KB önbellek bulunmaktadır. Önbellek kapasiteleri, yazılımsal olarak 2 türlü şekilde ayarlanabilmektedir.

- L1 önbelleğini 16 KB ve L2 belleğini 48 KB
- L1 belleğini 48 KB ve L2 önbelleğini 16 KB

CPU mimari yapısı ise 2 çekirdeklidir. Bu çekirdekler 3,0 Ghz'lik tek çekirdekli yapı halinde 2 adet olarak modellenmiştir. Dolayısıyla işlemci hızı teorik olarak 3 x 2 = 6 Ghz ulaşabilir.

İki hafıza modeli arasındaki farkı çizelge halinde gösterilirse;

Çizelge 3.1 Projede Kullanılan CPU-GPU Özellikleri

	CPU	**GPU**
Bellek	2.0 GB	1.5 GB
Bellek Band Hızı	12.8 GB/s	134.490 GB/s
L2 Cache	2 MB	16 KB > 48 KB
L1 Cache	64 KB > 128 KB	48 KB > 16 KB

CPU ile GPU arasındaki çalışma farkını daha iyi anlamak için 1 ile 16 sayısı arasındaki 16 adet sayının toplamı incelenirse;

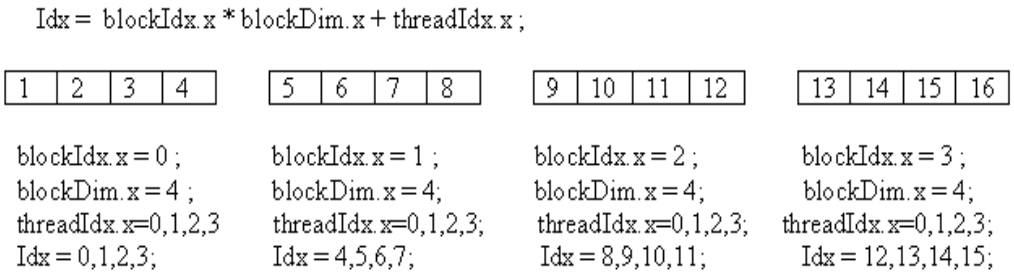
CPU üzerinden toplama;

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Şekil 3.6 CPU Toplama Dizisi

Şekil 3.6'da sayı dizisinin CPU üzerinden toplamı görülmektedir. CPU seri işlemci olduğundan toplama işlemini diziye tek tek erişerek yapmaktadır.

GPU üzerinden toplama ise;

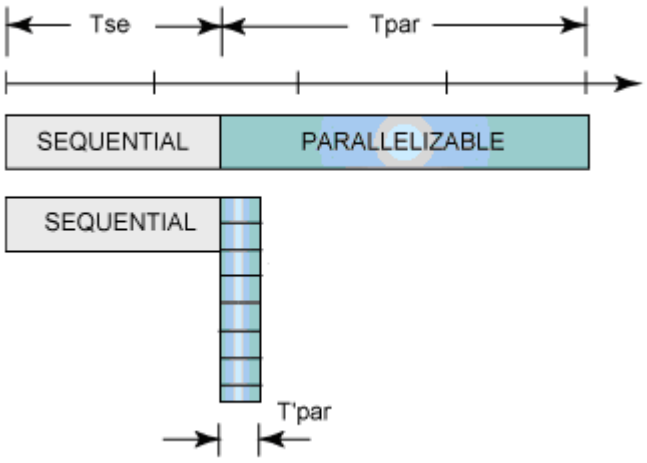


Şekil 3.7 GPU Toplama Dizisi

Şekil 3.7'de ise GPU üzerinden dizinin toplama işlemi görülmektedir. GPU 16'lık diziyi 4'lü bloklar halinde parçalayarak toplama işlemi için diziyi ayırmaktadır. Her bloğun içindeki 4'lü threadler sayesinde ise dizinin elemanlarına bloklar içerisinden erişilmektedir. GPU'nun yapısı gereği threadler ve bloklar senkron olarak çalışabilirler. Yani thread ve blokların hepsi çalışır ama karışık indisle çalışabilirler.

GPU'daki bu hesaplama şekli hız bakımından CPU'dan çok daha hızlı olmaktadır. GPU'daki bu hız farkını teorik olarak hesaplamak için Amdahl Yasası kullanılmaktadır.

Amdahl Yasası, Gene Amdahl tarafından öne sürülen bir algoritmanın paralel olarak çalıştırılması ile seri çalıştırılması arasındaki hız farkını öngören bir modeldir. Genel olarak, çoklu işlemci kullanıldığında erişilebilecek maksimum hızlanmayı hesaplayabilmek için kullanılmaktadır [29].



Şekil 3.8 Amdahl Yasası [30]

Şekil 3.8'de, paralel işlemler için yeterli sayıda işlemci kullanılarak T_{par} sıfıra doğru yaklaştırılabilir. Ancak, T_{se} hiçbir zaman değişmez. T_{par} süresini en kısa tutmak için paralel işlemci sayısını maksimum tutmak gerekir. Örneğin, bir programın toplam çalışma süresi 1 birim kabul edelim. Bu programın seri olarak çalışacak kısmına 1/N denilirse, paralel çalışacak kısım (1-1/N) olur. Teorik olarak, paralel çalışacak kısmı sıfıra yaklaştırmak için sonsuz sayıda çekirdek kullanılabilir ama seri kısmı her zaman 1/N şeklinde kalır. Sonuç olarak, programın çalışma süresi minimum, seri kısım olan 1/N kadar olacaktır. Bu yüzden, yazılacak programdaki en fazla işlem yükü gerektiren kısımlar (örneğin filtreleme, dizi işlemleri) CUDA vasıtasıyla GPU üzerinden çalıştırılabilir.

CUDA multi thread yapısı sayesinde programı paralelleştirdiği için hız olarak CPU'dan çok daha hızlı şekilde çalıştırılır. Seri işlemlerde ve giriş çıkış işlemlerinde ise CPU'yu kullanmak GPU' ya göre avantaj sağlar. Sistemin en verimli halini kullanmak için, heterojen yani CPU ve GPU'yu beraber kullanmak gerekir. Program yapısında, hesaplanması uzun süren algoritmaların GPU'da diğer işlemlerin CPU'da yapılması hem kullanıcı açısından hem de hız açısından çok daha verimli olur.

Hızlanma ise seri olarak (tek işlemci) çalıştırılan bir programın, paralel olarak (çok işlemci) çalıştırılan bir programın harcadığı süreye bölünmesine denir. Bu ifade;

$$S = \frac{1}{(1-P) + \frac{P}{N}} \tag{3.1}$$

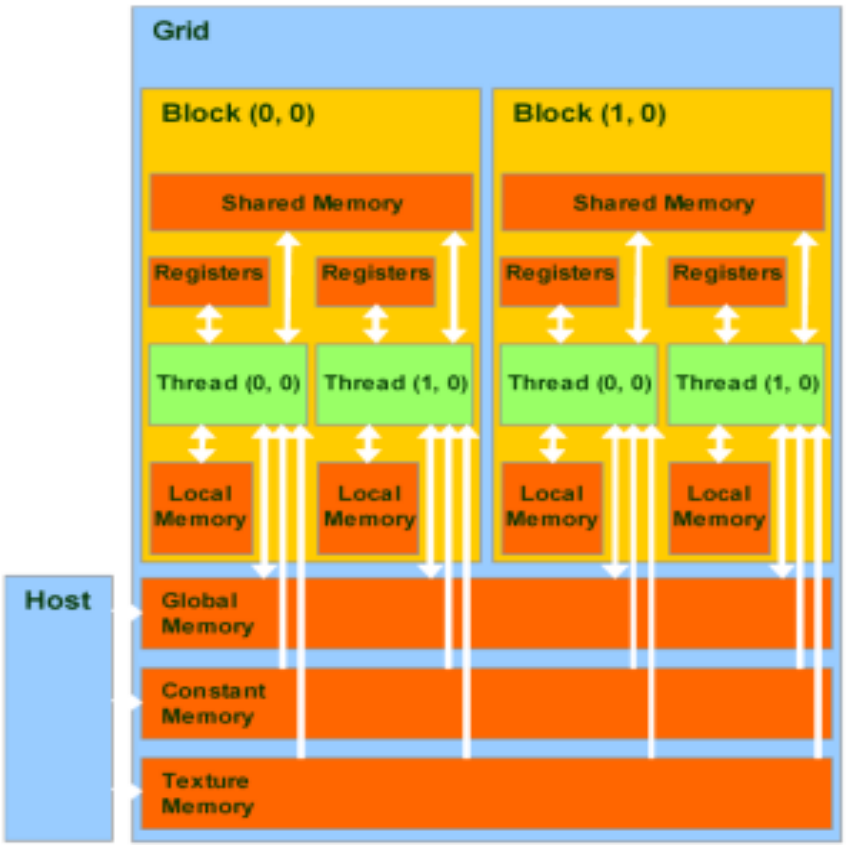
denklemi ile gösterilir.

- S: Algoritmanın Hızlanma Oranı,
- P: Algoritmanın Paralel İşlem Süresi,
- N: Çekirdek Sayısı

olarak gösterilir. Bu ifade de N çekirdek sayısı büyüdükçe, $\frac{P}{N}$ değeri $(1-P)$ değerine göre ihmal edilebilir bir düzeyde olacaktır. Böylece Denklem 3.1'deki ifade $S = \frac{1}{1-P}$ haline dönüşür. Programın %75'lik kısmı paralel olarak çalışabiliyorsa;

$S = \frac{1}{1-\frac{3}{4}} = 4$ kat hızlanma olur.

3.2.4.1.2 CUDA Çekirdek Yapısı



Şekil 3.9 CUDA Çekirdek Yapısı [31]

Şekil 3.9'da CUDA çekirdek yapısı gösterilen şemada ;

Host Bölgesi: Yapılan uygulama, ilk olarak CPU'da çalıştırılır. İşlenmek istenen verinin aygıta aktarımı, aygıt yönetimi gibi komutlar bu kısımda oluşturulmaktadır. CUDA teknolojisi sayesinde GPU + CPU yani heterojen programlama ile yazılan programları, maksimum verim sağlamak için birlikte çalıştırılabilmekte mümkündür.

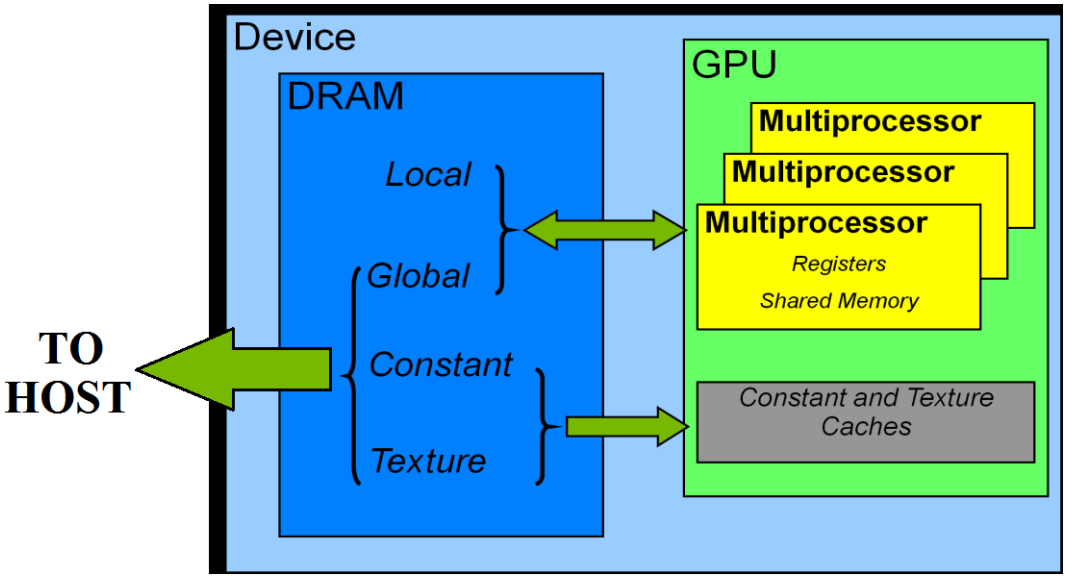
Kernel Bölgesi: Aygıt içinde kümelenmiş kerneller bulunmaktadır. CPU'da çalıştırılan bir uygulama GPU üzerinden çalıştırmak istendiğinde, C diline özel eklentiler sayesinde bu bölümde çalışması sağlanabilir. Hesaplama kapasitesi 2.0 olan GTX 480 modelinde birden çok kernel aynı anda çalıştırılabilmektedir. Çalışma şekli asenkron olduğundan, kerneller çalışırken kontrol tekrar CPU'ya geçebilmektedir.

Kernellerin Yapısı: Kernellerin içindeki görev parçacıklarına thread denilmektedir. Bu threadler tek, iki ve üç boyutlu olarak kümelenmişlerdir. Bu thread kümelerinin oluşturduğu yapıya ise blok denilmektedir. Bloklar ise yine tek, iki veya üç boyutlu şekilde kümelenmişlerdir. Bu kümelenmenin oluşturduğu yapıya da grid denilmektedir. Çizelge 3.2'de görüldüğü üzere, NVIDIA GTX 480 ekran kartında, her kernel için maksimum

65535x65535x65535 blok kümesi bulunmaktadır. Bir blokta en fazla oluşturulabilecek thread sayısı önceki CUDA ekran kartı versiyonlarında 512 iken yeni Fermi teknolojisiyle üretilen ekran kartlarında bu sayı 1024'e çıkarılmıştır. Bir bloktaki thread kümeleri ise maksimum 1024 x 1024 x 64 olarak tanımlanmıştır.

Kernellerde bu yapıları ve kullanılacak thread ve blok sayıları dim3 komutu ile istenildiği şekilde ayarlanabilir.

3.2.4.1.3 CUDA Hafıza Çeşitleri



Şekil 3.10 CUDA Hafıza Çeşitleri [32]

Şekil 3.10'da CUDA'nın bütün GPU modellerinde kullanılan genel hafıza çeşitleri ve bölgeleri görülmektedir. GPU hafıza bölümleri CPU ile PCI-Express veri yolu üzerinden veri alış-verişi yapmaktadır.

3.2.4.1.3.1 Host Bellek

Host bölümü CPU tarafından kontrol edilen RAM (Random Access Memory) bellektir. RAM belleği, CUDA threadleri doğrudan kullanamaz. Host bellek sadece global, constant ve texture bellekten okuma yazma işlemi yapabilmektedir. Veriler, uygulama sonuna kadar bellekte saklı kalmaktadır.

3.2.4.1.3.2 Shared Bellek

CUDA mimarisinde, Her thread bloğu kendi shared belleğine sahiptir. Shared belleğe, ilgili bloğun içinde sadece ilgili threadler tarafından erişilebilmektedir. Bu nedenle, diğer bloğun threadleri tarafından kullanılamamaktadır. Threadler kendi bloğunun

hafıza bölgesi olan shared belleğe direkt olarak erişebildiğinden en hızlı hafıza modelidir. Kaydedilen veriler blokların sonuna kadar saklanmaktadır.

3.2.4.1.3.3 Global Bellek

Global bellek, aygıt üzerinde DRAM (Dynamic Random Access Memory) bölgesinde bulunmaktadır. CPU ve aygıt içindeki threadler tarafından doğrudan kullanılabilmektedir. Global bellek, shared belleğe göre oldukça yavaş kaldığından, programları parçalar halinde shared bellekte işlemek çok daha avantajlıdır. Global belleğe diğer hafıza modelleri doğrudan erişememektedir. Global belleğe kaydedilen veriler hafıza bırakımına kadar saklanmaktadır. Sistemde kullanılan GTX480 Ekran Kartı, Çizelge 3.2'de görüldüğü üzere 1.5 GB 'lık GDDR5 (Graphic Double Data Rate 5) belleğe sahiptir.

3.2.4.1.3.4 Constant Bellek

Constant bellekte DRAM üzerinde ön bellek olarak yer almaktadır. Ön bellek boyutu 64 KB olarak sınırlandırılmıştır. Sadece okuma yapılacak veriler için avantajlıdır.

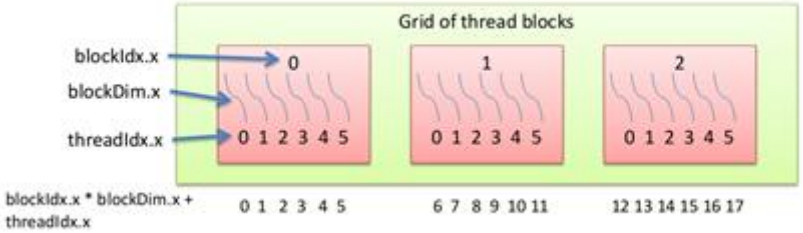
3.2.4.1.3.5 Texture Bellek

Aygıt üzerinde DRAM bölgesinde ön bellek olarak depolanan ve sadece veri okuma işlemleri için kullanılan bellek birimidir. Önbellek boyutu her çekirdekte 6~8 KB arasında değişir. Özellikle filtreleme, görüntü işleme ve 3 boyutlu alan bilgileri hesabı için kullanılmaktadır. Veri yazma işlemleri CPU üzerinden, okuma işlemleri ise GPU threadleri üzerinden sağlanır.

3.2.4.1.3.6 Local Bellek

CUDA derleyicisi tarafından derleme esnasında otomatik olarak oluşturulan değişkenlerin saklandığı bellek türüdür. Bu nedenle, programcıya açık olmayan bellektir.

3.2.4.1.4 Bellek Erişim Hiyerarşisi



Şekil 3.11 Bellek Erişim Hiyerarşisi [33]

3.2.4.1.4.1 Thread Hiyerarşisi

Threadler, thread blokları içerisinde 3 boyutlu ve paralel olarak çalışabilirler. Kendi aralarında çalışması senkron ve shared bellek ile haberleşebilirler. Her threadin kendi özel

hafıza bölgesi vardır. Bunlara local (yerel) bellek denir. Bu bellekteki verilerin ömrü, threadlerin işlenmesi bitene kadardır. Yani thread sırasını bitirince local bellekteki verinin de ömrü biter.

Bloklar içindeki threadleri kontrol etmek için iki yapı kullanılır.

threadIdx: Bloklar içindeki threadlerin index numaraları ile kontrol edilmesini sağlar.

blockDimx: Bloklar içindeki threadlerin sayısını tanımlamayı sağlar.

Threadler; Register, shared bellek, global bellek ve local bellek üzerinde veri okuma yazma işlemi yapabilirler. Ama constant bellek ve texture bellekten veri okuyabilmesine rağmen veri yazamazlar.

3.2.4.1.4.2 Blok Hiyerarşisi

Bloklar, gridler içerisinde 3 boyutlu olarak çalışırlar. Bloklar herhangi bir sıra ile çalışabilirler. Yani bloklar koordinelidir (hepsi çalışır) ama senkron değildir (sıra ile çalışmayabilir). Bloklar global bellek ile haberleşebilir.

Gridler içindeki blokları kontrol etmek için iki yapı kullanılır.

blockIdx: Gridler içindeki blokların index numaraları ile kontrol edilmesini sağlar.

gridDimx: Gridler içindeki blokların sayısını tanımlamayı sağlar.

3.2.4.1.5 Cuda Fonksiyon Tanımlamaları

__device__

Global bellekte stoklanır (cachelenmemiştir, yüksek gizlilik). Bütün threadler tarafından çalıştırılabilir. CPU tarafından çağrılamazlar. Veriler uygulama sonuna kadar saklanır.

__constant__

DRAM bölgesinde stoklanır (cachelenmiştir) CPU tarafından yazılabilir. Threadler tarafından sadece okunabilir. Veriler uygulama sonuna kadar saklanır.

__global__

CPU tarafından çağrılabilir. GPU tarafından çağrılamazlar. (diğer globalden çağrılamaz). Dönüş tipi void olmalıdır.

__host__

Sadece CPU tarafından çağrılabilir.

__host____device__

Kombine olarak çalıştırılabilir. Bu şekilde hem CPU hem de GPU üzerinde koordineli olarak çalışabilirler.

CUDA'nın avantajları olarak C diline küçük eklentilerle çalışması, kodların rastgele erişimli belleğe yazılması, GPU hafıza yönetimini ve kernelleri çalıştırmayı sağlaması ve heterojen (CPU+GPU) programlamaya izin vermesi gösterilebilir. Dezavantajları olarak recursive (özyinelemeli) fonksiyonların ve double veri tipinin desteklenmemesi, program parçacıklarının en iyi 32'li gruplar halinde yazılması, fonksiyon işaretçilerine izin vermemesi ve yazılan programların her ekran kartında çalışmaması gösterilebilir.

Program yazılırken avantaj sağlayacak konular;

- ✓ Büyük ölçekli bir transfer, küçük ölçekli çoklu transferden daha hızlıdır.
- ✓ Her Stream Proccessing de 192 den daha az thread kullanılması avantaj sağlamaz.
- ✓ Her bloktaki thread sayısı warp sayısı olan 32 sayısının katları olmalıdır. Aksi takdirde bir sonraki warp tam verimli haliyle çalıştırılmamış olur. Bu da süreyi uzatır.
- ✓ Her bloktaki thread sayısı maksimuma yakın olursa daha az esnek, minimuma yakın olursa da GPU'nun avantajını tam olarak kullanılmamasına neden olur. İdeal olarak thread sayısı bir blok için 256-512 arasında olabilir.
- ✓ Band genişliği en hızlı olan shared bellekten en fazla şekilde yararlanmak, programın çalışma süresini kısaltır.
- ✓ Matematiksel ifadelerde;

sin(x) : Yavaş hesaplanır fakat doğruluk payı çok fazladır.

__sinf(x) : Hızlı hesaplanır fakat doğruluk payı daha azdır.

İki durum arasında öncelik durumunu iyi belirlemek gerekir.

İşleme yeteneği 2.0 olan GeForce GTX 480 ekran kartının teknik özellikleri;

Çizelge 3.2 NVIDIA GTX 480 Teknik Özellikleri

Cudadevice	0
Adı	GeForce GTX 480
Toplam Global Hafıza	1610612736
Her Bloktaki Shared Bellek	49152
Her Bloktaki Kaydedici Sayısı	32768
Warp Boyutu	32
Bellek Alanı	2147483647
Her Bloktaki Maksimum Thread Sayısı	1024
Maksimum Thread Ölçüleri	1024,1024,64
Maksimum Grid Ölçüleri	65535,65535,65535
Clock Rate	1401000
Toplam Constant Bellek	65536
Majör	2
Minör	0
Texture Gruplama	512
Multi İşlemci Sayısı	15

3.2.4.2. Görüntü İşleme Temelleri

Geliştirilen sistemde, dijital kameradan 640x100 boyutlarında alınan görüntünün, analiz edilmesi için kullanılacak algoritmanın yeterince hızlı olması gerekmekteydi. Yapılan literatür taramasına göre, ihtiyaca en uygun algoritma olarak sezgisel bulanık mantık nesne çıkartım algoritması olduğu görülmüştür. Sezgisel bulanık mantık nesne çıkartım algoritmasına tezin 3.2.4.4. bölümünde değinilmektedir.

Dijital kameradan alınan görüntülerin analiz edilmesinde, RGB renk uzayının kullanılması uzun zaman gerektirdiğinden, dijital kameranın da desteklediği gri seviyenin kullanılması tercih edilmiştir. Dijital kameradan görüntüler, OpenCV kütüphanesi ile elde edilmektedir. OpenCV kütüphanesinde, renk sıralaması standart olan RGB yerine BGR renk modu kullanılmaktadır. RGB renk uzayındaki bir görüntünün piksel değerini, gri seviye dönüştürme işlemi Denklem 3.2'den elde edilir [34].

$$P_G(ij) = 0.30V_R + 0.59V_G + 0.11V_B \tag{3.2}$$

Gri seviyede elde edilen görüntü, SBM (Sezgisel Bulanık Mantık) algoritmasının CUDA aracılığıyla GPU'da hesaplanması için öncelikle alınan her çerçevenin GPU belleğine yüklenmesi gerekmektedir.

Yükleme yapılmadan önce, GPU üzerindeki hafıza alanını en efektif kullanmak için, hafıza isteme ve hafıza boşaltma işleminin mutlaka yapılması gerekmektedir. Bu şekilde rastgele erişimli bellekte görüntünün kontrolü için maksimum verim sağlanmış olur. Bu işlemi gerçekleştirmek için aşağıdaki komut dizisi kullanılmaktadır.

```
cudaMalloc((void **) &DInImage, size);
cudaMalloc((void **) &DOutImage, size);
```

Kod 3.2 CUDA Hafıza Ayırma Kodu

Komut dizisindeki;

- Size: Görüntünün boyutunu temsil etmektedir.

Size değeri Denklem 3.3'e göre elde edilmektedir.

$$size = h_f \; w_f \; b \tag{3.3}$$

Denklemde;

- h_f : Görüntünün yüksekliği
- w_f : Görüntünün genişliği
- b : Görüntüdeki bir pikselin temsil edileceği bit türünden değeridir.

GPU belleğinde ayrılan belleğe görüntünün yüklenmesi işlemini de aşağıdaki komut dizisi gerçekleştirmektedir.

```
cudaMemcpy(DInImage, InImage, size, cudaMemcpyHostToDevice);
cudaMemcpy(DOutImage, OutImage, size, cudaMemcpyHostToDevice);
```

Kod 3.3 CUDA Hafıza Kopyalama Kodu

Burada;

- DInImage: Alınan görüntüyü
- DOutImage: GPU'dan işlendikten sonra yüklenecek boş çerçeveyi.
- cudaMemcpyHostToDevice: Kopyalama yapılacak belleklerin yönünü

belirtmektedir (Örnekte CPU'dan GPU'ya kopyalama yapılıyor).

Bellekler arasındaki kopyalama işlemi tamamlandıktan sonra çalıştırılacak kernellerin parametreleri olan thread sayısı ve blok sayısının tanımlanması gerekmektedir. Bu işlemi aşağıdaki komut dizi gerçekleştirmektedir.

```
dim3 blockSize  ((Width/20),(Height/10));
dim3 grid(20,10);
```

Kod 3.4 Grid-Block Ayarlama Kodu

Burada dim3, thread ve blok sayısının tanımlanmasına yarayan veri tipidir. İçerisinde iki tane yapı bulunur. Bunlar blockSize yani bir bloğun içerisinde çalıştırılabilir thread sayısıdır. Diğeri ise grid yani bir grid içerisinde çalıştırılabilir block sayısını temsil etmektedir. Daha önceki NVIDIA ekran kartlarında bir blok içerisinde bulunabilecek maksimum thread sayısı 512 iken, yeni nesil Fermi teknolojisinde bu sayı 1024'e çıkartılmıştır. Bir grid içerisindeki maksimum blok sayısı ise 65535 tanedir. Eğer çalıştırılan kernelin daha hızlı çalıştırılması isteniyorsa blok içerisindeki thread sayısı 1024'e en yakın değer olmalıdır.

Yapılan projede;

Width = 640

Height = 100

olduğundan

((Width/20),(Height/10))= ((640/20) x (100/10)) = 320

olur. Bu da istediğimiz hızı yakalamamızı sağlamıştır. Dim3 veri yapısı, 3 boyutlu bir veri yapısıdır. Ancak görüntü işlemede 2 boyut üzerinden çalışıldığından 3.boyutu girilmediği takdirde otomatik olarak 1 sayısı eklenir. Bu husus dikkate alındığında, komut dizisi yeniden düzenlenerek aşağıdaki gibi olur [35].

```
dim3 blockSize  ((Width/20),(Height/10),1);
dim3 grid(20,10,1);
```

Kod 3.5 Grid-Block 3 Boyutlu Ayarlama Kodu

Bu tanımlamaları yaptıktan sonra, kernellerin çağırılması gerekmektedir. Aşağıda geliştirilen sistemde, yazılan FuzzyStart kerneli çağrılmaktadır.

```
      FuzzyStart  <<<grid,blockSize>>> (DInImage, DOutImage,Height,
Width);
```

Kod 3.6 CUDA Kernel Çağırma İşlemi Kodu

CUDA'da kernelleri çağırma işlemi "<<< >>>" parametreleri ile sağlanmaktadır. Bu ifadelerin içerisine, thread ve blok sayıları yazılmaktadır. CPU programlamadaki fonksiyon çağırma işlemine benzer şekilde, çağrılan fonksiyona çeşitli parametreler gönderilebilmektedir. Bu parametreler CPU kısmında tanımlı parametreler olmak zorundadır [36].

Geliştirilen sistemde, GPU temelli görüntü işleme aşamasında, bulanık mantık teoreminin Atanassov tarafından geliştirilmesi sonucunda ortaya çıkan sezgisel bulanık mantık teoremine dayanan, Chira T. tarafından geliştirilerek başarılı bir şekilde uygulanan sezgisel bulanık mantık nesne çıkartım algoritması kullanılmıştır. Bu algoritmanın matematiksel temeli ile görüntü işlemedeki kullanımı, aşağıdaki bölümlerde anlatılmaktadır.

3.2.4.3. Bulanık Küme (Fuzzy Sets)

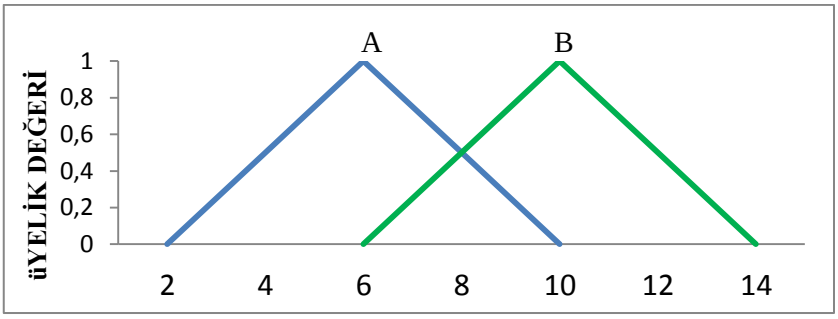
Bulanık mantık kuramı ilk kez, 1965 yılında L. A. Zadeh tarafından ortaya atılmıştır. Klasik küme kuramında, bir eleman kümeye ya aittir ya da değildir. Fakat bulanık mantık küme teorisinde bir eleman kümelere kısmen olarak da üye olabilmektedir. Bir bulanık mantık kümesi üyelik derecelerine göre karakterize edilmektedir. Bu üyelik fonksiyonu elemanın kümedeki ağırlık derecesini temsil etmektedir. Bulanık kümelerde, elemanın üyelik derecesi [0-1] aralığında yani tam üyelikten, üye olmamaya kadar değişebilmektedir [37].

Karmaşık sistemlerin, bulanık mantık ile analiz ve kontrol edilebileceği Zadeh tarafından ortaya konulduktan sonra, Mamdani E., bir buhar tribününün hızının denetlenmesine uygulamış ve bu amaçla, bir insanın davranışlarını taklit eden; "Eğer türbin hızı çok hızlı artıyorsa ve basınç da çok düşükse, buhar vanasını biraz aç" türünden kurallardan oluşan bir sistem geliştirmiştir. Mamdani, bulanık mantık temelli bu tür bir sistemle tribün hızının ve performansının çok başarılı bir şekilde denetlenebileceğini göstermiştir [38].

Bulanık mantık, insan düşüncesine ve doğal dile geleneksek mantık sistemlerinden daha yakın bir sistemdir. Bulanık mantık denetleyicisi uzman bilgisine ve tecrübesine dayalı olan sözel koşul cümleleriyle otomatik kontrol işlemi yapmaktadır [39].

Klasik küme de bir elemanın üyelik değeri ya 0'a eşit ya da 1'e eşittir. Fakat bulanık kümede Şekil 3.12'de görüldüğü üzere böyle bir durum söz konusu değildir. Bulanık küme de bir elemanın sahip olabileceği tüm üyelik değerleri evrensel küme olarak tanımlanmaktadır. Bu evrensel küme, tanımlı olan bütün bulanık alt kümeleri kapsamaktadır.

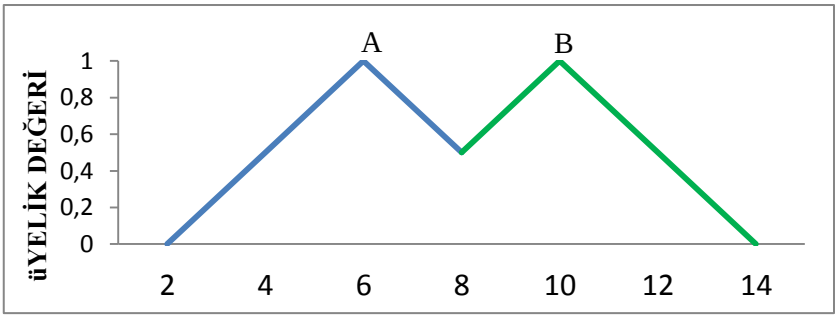
Klasik kümenin temel özellikleri, bulanık mantık kümelerde de geçerlidir. Bu özellikler için X evrensel kümesinde, Şekil 3.12'de görülen A ve B olmak üzere iki bulanık mantık küme tanımlı olsun;



Şekil 3.12 A ve B Bulanık Kümeleri

Birleşim: X evrensel kümesinde tanımlı olan A ve B kümeleri için $A \cup B$'nin üyelik fonksiyonunun değeri Denklem 3.4'den elde edilir. Bu denkleme göre elde edilen birleşim kümesi ise Şekil 3.13'de görülmektedir.

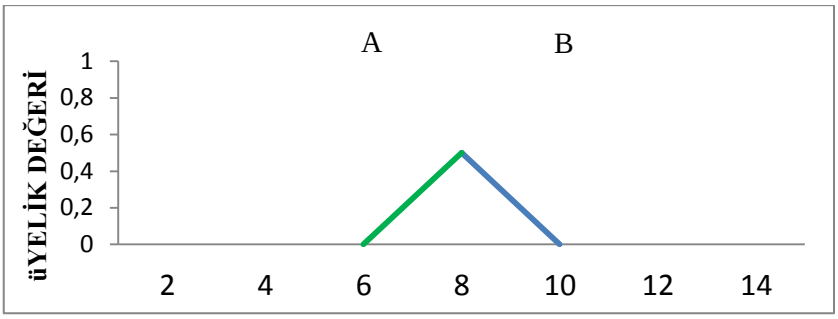
$$\mu_{A\cup B}(x) = Max\left[\mu_A(x), \mu_B(x)\right] \tag{3.4}$$



Şekil 3.13 $A \cup B$ Grafiği

Kesişim: X evrensel kümesinde tanımlı olan A ve B kümeleri için $A \cap B$'nin üyelik fonksiyonunun değeri Denklem 3.5'ten elde edilir. Bu denkleme göre elde edilen keşişim kümesi ise Şekil 3.14'de görülmektedir.

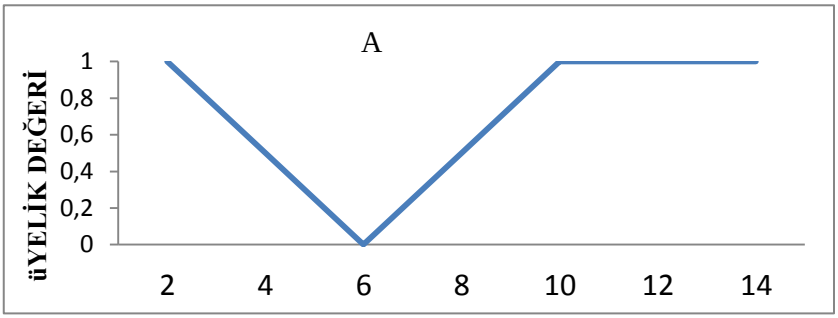
$$\mu_{A\cap B}(x) = Min\left[\mu_A(x), \mu_B(x)\right] \tag{3.5}$$



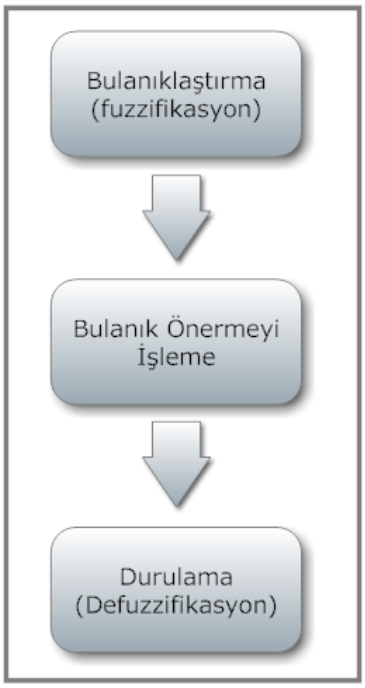
Şekil 3.14 $A \cap B$ Grafiği

Tümleme: X evrensel kümesinde tanımlı olan A kümesi için A^{-1}'nin üyelik fonksiyonunun değeri Denklem 3.6'dan elde edilir. Bu denkleme göre elde edilen A kümesinin tümleyen kümesi Şekil 3.15'de görülmektedir.

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{3.6}$$



Şekil 3.15 A' Grafiği

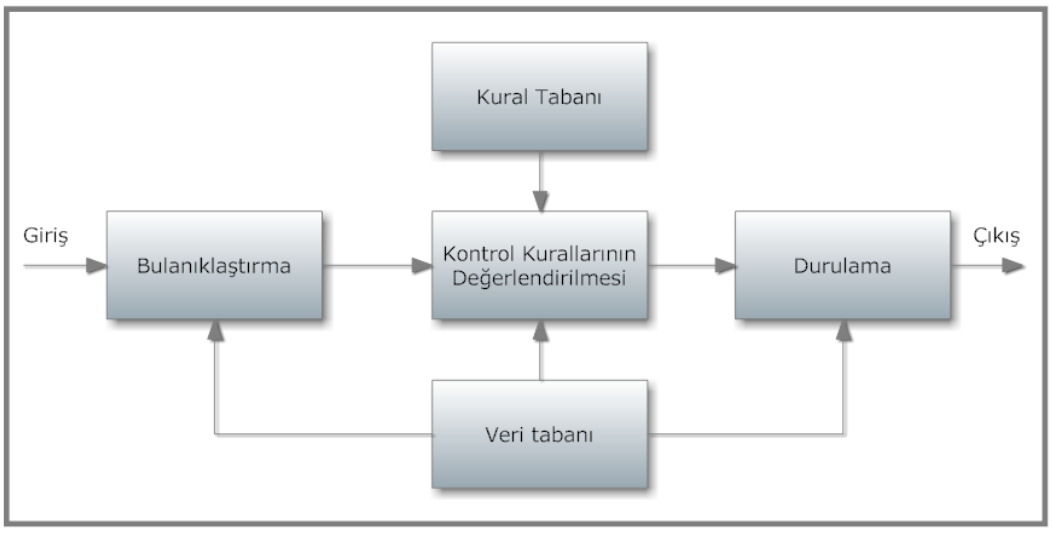


Şekil 3.16 Bulanık Mantık Süreçleri

Bulanık mantık uygulamaları, Şekil 3.16'da görüldüğü üzere;

- Bulanıklaştırma
- Bulanık Önermeyi İşleme
- Durulama

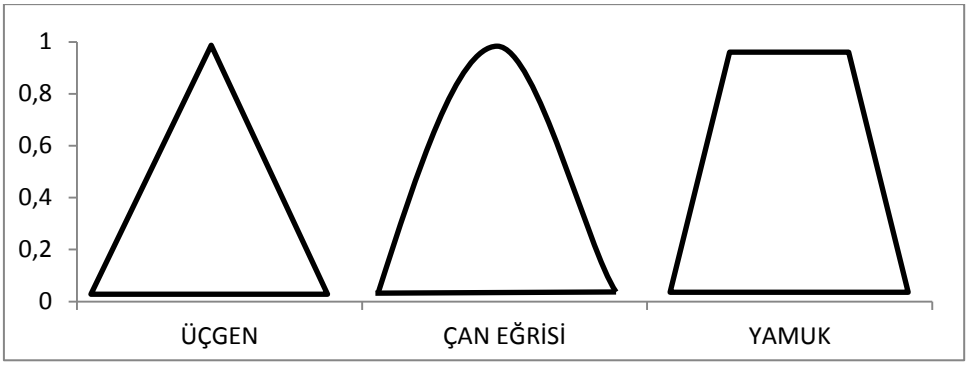
süreçlerinden geçmektedir. Bulanık mantık uygulamalarında kullanılan genel blok şeması Şekil 3.17'de görülmektedir.



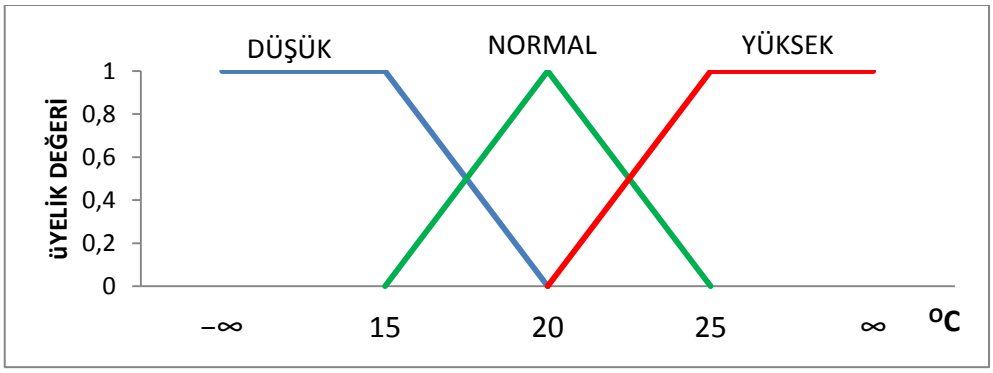
Şekil 3.17 Bulanık Mantık Genel Blok Şeması

3.2.4.3.1 Bulanıklaştırma Süreci

Çözülecek problem ile ilgili bulanık önerme değişkenlerinin ve karar verme kurallarının belirlenmesi ve üyelik fonksiyonunun oluşturulması işlemlerini kapsamaktadır. Bir bulanık değişken sözel dille ifade edilen değere sahiptir. Örneğin, hava sıcaklığı uzman kişi tarafından düşük, normal ve yüksek olarak tanımlanabilmektedir. Bu tanımlamalar Şekil 3.18'de görüldüğü üzere farklı tiplerde üyelik fonksiyonu kullanılarak tanımlama yapılabilmektedir. Hava sıcaklığının üçgen üyelik fonksiyonu kullanılarak bulanık küme tanımlaması ise Şekil 3.19'da görülmektedir.



Şekil 3.18 Bulanık Mantık Üyelik Fonksiyonları



Şekil 3.19 Bulanık Mantık Küme Tanımlaması

Bulanıklaştırma sürecinde, sayısal olarak alınan değerlerin dilsel olarak ifade edilen karşılıklarına üyelik derecelerince atanması ve bu ifadelere göre işlemlerin gerçekleştirilmesi süreçlerini kapsamaktadır.

3.2.4.3.2 Bulanık Önerme Süreci

Bulanık önerme süreci, kural tabanının oluşturulması ile uygulaması ve bulanık çıkartım işlemlerinden oluşmaktadır.

3.2.4.3.2.1 Kural Tabanı

Bulanık kontrol kural tabanı, uzmanlar tarafından oluşturulan kontrol kurallarının toplamına denilmektedir. Bulanık kontrol kural tabanı genellikle EĞER… O HALDE şeklindedir. Bir bulanık mantık kural tabanı;

$$\text{Kural1} : \text{Eğer}\left(X_{11} = A_{11}\right) \text{ VE…} \text{ VE } \left(X_{1m} = A_{1m}\right) \text{ ise U} = B_1$$
$$\text{Kural2} : \text{Eğer}\left(X_{21} = A_{21}\right) \text{ VE…} \text{ VE } \left(X_{2m} = A_{2m}\right) \text{ ise U} = B_2$$
$$\ldots$$
$$\text{Kuraln} : \text{Eğer}\left(X_{nm} = A_{nm}\right) \text{ VE…} \text{ VE } \left(X_{nm} = A_{nm}\right) \text{ ise U} = B_n$$

şeklinde olabilir. Örneğin, kural tabanı tanımlanmış Çizelge 3.3 ‘deki bulanık mantık denetiminde;

Çizelge 3.3 Bulanık Mantık Kural Çizelgesi

		G1				
	Ç	**NB**	**NK**	**SI**	**PK**	**PB**
G2	**NB**	NB	NB	NO	NK	SI
	NK	NB	NO	NK	SI	PK
	SI	NO	NK	SI	PK	PO
	PK	NK	SI	PK	PO	PB
	PB	SI	PK	PO	PB	PB

- Eğer G1=SI ve G2=NK ise Ç = NK

Kuralların “Varsayım” kısmı denetleyicinin giriş değişkenleri ile “Sonuç” kısmı ise çıkış değişkeni ile ilgilidir. Yukarıda verilen örnekte, her varsayım AND operatörü ile bağlanmış iki terimden oluşmaktadır. Varsayım kısmı ikiden daha fazla terimden oluşabileceği gibi OR ve NOT operatörleri ile de bağlanabilirler. Kuralların sonuç kısmında ise, eğer sistemin çıkışı birden fazla ise, bu kısımda birden fazla terim içerebilmektedir.

3.2.4.3.2.2 Bulanık Çıkartım

Bulanık mantık sonuç çıkartıma işlemi, sözel kuralların üzerinden bulanık çıkışı belirleme işlemlerini barındırmaktadır. Alınan giriş bilgileri için hangi kuralların uygulanacağı ve hangi uygun bulanık çıkış değerinin belirleneceği birçok çıkartım modelleri vardır. Fakat yapılan literatür taramalarına göre, en çok kullanılan yöntemler Takagi-Sugeno (MAX-DOT) ve Mamdani (MAX-MIN) yöntemleri olduğu görülmüştür [39].

İki tane bulanık kurala sahip bir bulanık kural tabanı olsun;

$$\text{Kural1} : \text{Eğer}\left(\text{X} = \text{A}_1\right)\text{VE}\left(\text{Y} = \text{B}_1\right) \text{ ise Z} = \text{C}_1$$
$$\text{Kural2} : \text{Eğer}\left(\text{X} = \text{A}_2\right)\text{VE}\left(\text{Y} = \text{B}_2\right) \text{ ise Z} = \text{C}_2$$

i. kurallın kullanılma ağırlığı α_i olsun. X_0 ve Y_0 girişleri için kural tabanının α_1 ve α_2 kullanılma ağırlığı Denklem 3.7 ve Denklem 3.8'den elde edilmektedir.

$$\alpha_1 = \mu_{A1}\left(X_0\right) \wedge \mu_{B1}\left(Y_0\right) \tag{3.7}$$

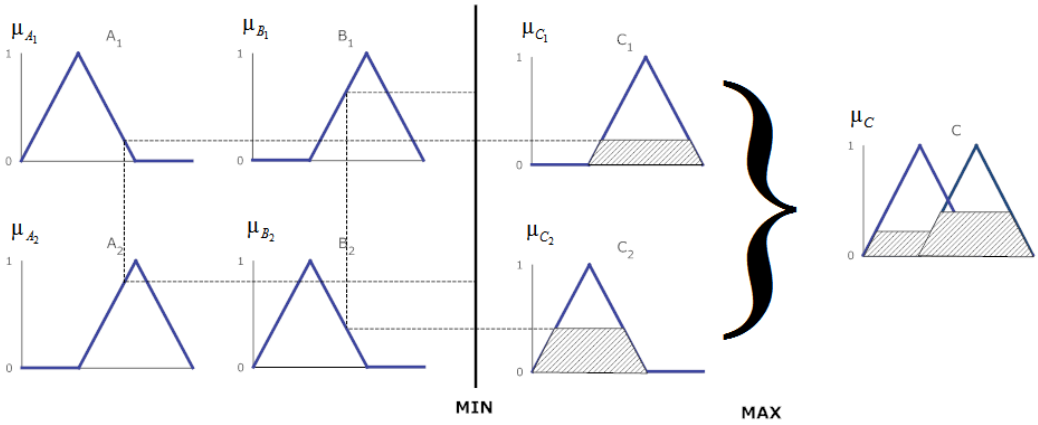
$$\alpha_2 = \mu_{A2}\left(X_0\right) \wedge \mu_{B2}\left(Y_0\right) \tag{3.8}$$

3.2.4.3.2.2.1 Mamdani (MAX-MIN) Bulanık Çıkartım Yöntemi

Mamdani bulanık çıkartım yöntemine göre, i. kural $\alpha_i \wedge \mu_{C_i}\left(\omega\right)$ ile ilişkilendirilerek kontrol kuralına ulaşılmaktadır. Böylece çıkarım sonucu C'nin üyeliği Denklem 3.9'dan elde edilir;

$$\mu_c\left(\omega\right) = \left(\alpha_1 \wedge \mu_{C_1}\left(\omega\right)\right) \vee \left(\alpha_2 \wedge \mu_{C_2}\left(\omega\right)\right) \tag{3.9}$$

Şekil 3.20'de kesin giriş değerleri x_0 ve y_0 için MAX-MIN çıkarım işlemi görülmektedir.



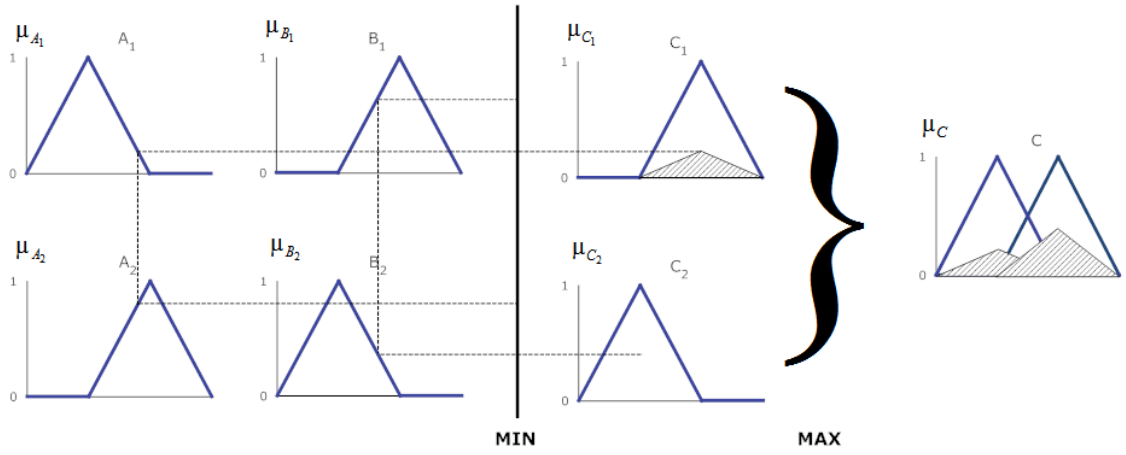
Şekil 3.20 Mamdani Bulanık Çıkartım İşlemi

3.2.4.3.2.2.2 Takagi-Sugeno (MAX-DOT) Bulanık Çıkartım Yöntemi

Takagi-Sugeno bulanık çıkartım yöntemine göre, i. kuralın $\alpha_i \mu_{C_i}(\omega)$ olarak açıklanması ile karar alınmaktadır. Çıkartım sonucunda C'nin üyelik değeri Denklem 3.10'dan elde edilmektedir.

$$\mu_c(\omega) = \left(\alpha_1 \mu_{C_1}(\omega)\right) \vee \left(\alpha_2 \mu_{C_2}(\omega)\right) \tag{3.10}$$

Şekil 3.21'de kesin giriş değerleri x_0 ve y_0 için MAX-DOT çıkarım işlemi görülmektedir.



Şekil 3.21 Takagi-Sugeno Bulanık Çıkartım İşlemi

3.2.4.3.3 Durulaştırma Süreci

Bulanık çıkartım yöntemlerinin çıkışı, evrensel kümede bulanık bir kümedir. Bu yüzden, nümerik değerlere çevrilmesi gerekmektedir. Bu çevirme işlemi durulaştırma (defuzzification) olarak adlandırılmaktadır. Durulaştırma işlemi Denklem 3.11'deki şekilde ifade edilebilmektedir.

$$y_0 = durulaştırıcı(y) \tag{3.11}$$

Burada; y bulanık çıkışı, y_0 nümerik çıkışı ve $durulaştırıcı()$ ise durulaştırma işlemini gerçekleştiren fonksiyonu temsil etmektedir.

Bulanık mantık uygulamalarında, maksimumların ortalaması (mean of maxima-MOM) ve alan merkezi (center of area-COA) durulaştırma işlemi için en çok kullanılan yöntemlerdir.

3.2.4.3.3.1 Maksimumların Ortalaması (Mean of Maxima-MOM)

Maksimumların ortalaması yöntemi, yükseklik durulaştırıcısı olarak da adlandırılmaktadır. Bu yöntemde Denklem 3.12'den anlaşılacağı üzere, üyelik değerleri maksimuma ulaşan bütün çıkışların ortalama değerini alarak çıkışın nümerik değerine dönüştürülmektedir.

$$Z = \frac{\sum_{j=1}^{n} z_j}{|M|}, z_j \in M \tag{3.12}$$

3.2.4.3.3.2 Alan Merkezi (Center of Area-COA)

Alan Merkezi yöntemi, Denklem 3.13'den anlaşılacağı üzere çıkış bulanık kümesin ağırlık merkezi nümerik çıkış değeridir.

$$Z = \frac{\sum_{i=1}^{n} u_i \mu_{OUT}(u_i)}{\sum_{i=1}^{n} \mu_{OUT}(u_i)} \tag{3.13}$$

3.2.4.4. Sezgisel Bulanık Küme (Intuitionistic Fuzzy Sets)

3.2.4.4.1 Konsept

Atanassov, Zadeh [37]'in klasik bulanık mantık küme üzerine yaptığı çalışmalarda, bulanık mantık küme teorisinin tanımının doğru olduğunu fakat gerçek hayatta bunun her zaman doğru yanıt vermeyeceğini belirtmektedir. Atanassov yaptığı çalışmalarda, gerçek hayatta insan deneyimlerine göre belirlenen bir kümenin gerçek üyelik değerinin tam olarak bilinemeyeceğini bu nedenle, sonucun istendiği gibi olamayacağını belirtmektedir. Bu problemin çözümü için, Zadeh'in bulanık mantık küme teoremine ek olarak sezgisel bulanık indisi ya da tereddüt derecesi olarak adlandırılan $\pi_A(x)$ katsayı tanımlamaktadır [40] [41].

Atanassov, bulanık kümelerin, sezgisel bulanık küme olarak genellenmesini önermektedir. Bir sezgisel bulanık A kümesi $X = \{x_1, x_2, \ldots, x_n\}$ şeklinde tanımlı ise matematiksel olarak gösterimi [42];

$$A = \{x, \mu_A(x), \upsilon_A(x) \mid x \in X\} \tag{3.14}$$

Burada $\mu_A(x) : X \rightarrow [0,1]$; X kümesinde tanımlı her x elemanının üyelik derecesi tanımlıdır. Buradan üyesizlik derecesi de;

$$\upsilon_A = 1 - \mu_A(x) \tag{3.15}$$

olarak bulunur.

Burada $\mu_A(x), \upsilon_A(x) : X \rightarrow [0,1]$; X kümesinde tanımlı her x elemanının üyelik derecesi $\mu_A(x)$, üyesizlik derecesi $\upsilon_A(x)$ fonksiyonları aşağıdaki şart sağlandığı sürece tanımlıdır.

$$0 \leq \mu_A(x) + \upsilon_A(x) \leq 1 \tag{3.16}$$

Buradan kolayca anlaşılacağı üzere her bulanık mantık kümesi, sezgisel bulanık mantık kümesinin bir özel durumudur. Bu durumun matematiksel ifadesi aşağıda verilmiştir.

$$A = \{(x, \mu_A(x), 1 - \mu_A(x)) \mid x \in X\} \tag{3.17}$$

Atanassov Denklem 3.16'da tanımlı olan bulanık mantık küme teoremine üçüncü bir parametre olarak, $\pi_A(x)$ sezgisel bulanık indisi veya tereddüt (hesitation) derecesi tanımlamaktadır. Tereddüt derecesi bilgi eksikliği veya kişisel hataları en aza indirmek için Denklem 3.16'ya eklenmiştir [43]. Bu durumda, tereddüt derecesi ile sezgisel bulanık mantık X kümesinin matematiksel ifadesi;

$$A = \{(x, \mu_A(x), \upsilon_A(x), \pi_A(x)) \mid x \in X\} \tag{3.18}$$

olarak tanımlanır. Sezgisel bulanık mantık A kümesinin tanımından;

$$\mu_A(x) + \upsilon_A(x) + \pi_A(x) = 1 \tag{3.19}$$

olarak ifade edilir. Elde edilen bu ifadeden;

$$0 \le \pi_A(x) \le 1 \mid x \in X \tag{3.201}$$

olabileceği çıkartılabilmektedir.

3.2.4.4.2 Sezgisel Bulanık Mantık Küme Hesaplamaları

Denklem 3.19'da görüldüğü üzere, sezgisel bulanık mantık küme işlemleri için 3 değerin hesaplanması gerekmektedir. Bunlar, ilgili küme elemanlarının üyelik derecesi $\mu_A(x)$, üyesizlik derecesi $\upsilon_A(x)$ ve tereddüt derecesi $\pi_A(x)$'dir [40] [42].

$A = \{x,\ \mu_A(x), \upsilon_A(x), \pi_A(x) \mid x \in X\}$, A sezgisel bulanık mantık kümesi tanımlı olsun;

- $\mu_A(x)$ Kullanılan görüntü kümesinin üyelik değeridir.
- $\upsilon_A(x)$ Sezgisel bulanık mantık üyesizlik değeri Denklem 3.21'den elde edilir.

$$\upsilon_A(x) = 1 - \mu_A(x) - \pi_A(x) \tag{3.21}$$

- $\pi_A(x)$:Sezgisel bulanık mantık tereddüt derecesi ise Denklem 3.22'den elde edilir.

$$\pi_A(x) = C^*[1 - \mu_A(x)] \tag{3.22}$$

3.2.4.4.3 Sezgisel Bulanık Nesne Çıkartma Algoritması (Intuitionistic Fuzzy Object Extraction)

Geliştirilen sistemde, dijital kameradan alınan görüntüdeki pamuk yığınındaki yabancı maddelerin tespit edilmesinde; Chira T ve Ray. K tarafından geliştirilen, sezgisel bulanık mantık teoremine dayanan sezgisel bulanık mantık nesne çıkartım algoritması kullanılmıştır [44] [12].

3.2.4.4.3.1 Temeli

Sezgisel bulanık mantık çıkartım algoritması, Chira T. ve Ray K. tarafından geliştirilen intuitionistic fuzzy divergence hesaplamasına dayanmaktadır. Bu hesaplama da**,** sezgisel bulanık mantık küme teoreminde bulunan tereddüt derecesi, olasılık hesaplamalarıyla belirlenmektedir. Chira T. ve Ray K tarafından geliştirilen sezgisel bulanık mantık diverjans hesabının çıkartım aşamaları aşağıda bulunmaktadır [44] [12].

Olasılık değerleri üzerinden mesafe ölçümü tanımlayan Shannon [45]'a göre, $P=\{p_1,p_2,\ldots,p_n\}$ ile tanımlı olan bir sezgisel bulanık mantık kümesinin her bir elemanın üyelik değerinin olasılık değerleri;

$$H\left(\{p_1, p_2, ..., p_n\}\right) = -\sum_{i=1}^{n}\left[p_i \log_2(p_i)\right] \tag{3.23}$$

denklemi ile gösterilir.

Shannon 'un bir sezgisel bulanık mantık kümesi üzerinde yaptığı araştırmasının üzerine, Kullback ve Leibler; iki sezgisel bulanık mantık kümesi arasında olasılık değerleri üzerinden mesafe ölçümünü tanımlamışlardır. Bu çalışmalarına göre, $P=\{p_1,p_2,\ldots,p_n\}$ ve $Q=\{q_1,q_2,\ldots,q_n\}$ ile tanımlı olan iki sezgisel bulanık mantık kümelerinin her bir elemanlarının üyelik değerlerinin olasılık değerlerini Denklem 3.24'den elde etmektedir [46].

$$D(P,Q) = \sum_{i=1}^{n} p_i \log_2 \frac{pi}{qi} \tag{3.24}$$

Kullback ve Leibler' in iki sezgisel bulanık mantık kümelerinin arasındaki mesafe ölçüm hesaplamasını, Szmidt E ve Kacprzyk J, iki görüntünün pikselleri arasındaki mesafe ölçmek için, Hamming mesafe [47]ve Euclidean mesafe [48] ölçme yöntemlerini sezgisel bulanık mantık teoreminde uygulayarak aşağıdaki bağıntıları elde etmişlerdir [49] [50].

Euclidean mesafe ölçüm' ün sezgisel bulanık mantık küme teoremine göre mesafe ölçümü;

$$E_{IFS}(A,B)=\sqrt{\sum_{i=1}^{n}\left[(\mu_A(x_i)-(\mu_B(x_i))^2+(\upsilon_A(x_i)-\upsilon_B(x_i))^2+(\pi_A(x_i)-\pi_B(x_i))^2\right]} \quad (3.25)$$

ile ifade edilir.

Hamming mesafe ölçüm' ün sezgisel bulanık mantık küme teoremine göre mesafe ölçmesi Denklem 3.26'da görülmektedir;

$$H_{IFS}(A,B)=\sum_{i=1}^{n}|\mu_A(x_i)-\mu_B(x_i)|+|\upsilon_A(x_i)-\upsilon_B(x_i)|+|\pi_A(x_i)-\pi_B(x_i)| \quad (3.26)$$

Chira T. ve Ray A.K.' ya göre, sezgisel bulanık mantık kümesinin iki resim arasındaki mesafeden yararlanılarak diverjans hesabı yapılabilmektedir [12].

Bu hesaplama yöntemine göre Denklem 3.21'de ifade edilen denklem;

$$\pi_A(x)=1-\mu_A(x)-\upsilon_A(x) \quad (3.27)$$

şeklinde yazılabilir.

Montes I. ve arkadaşlarına göre ; $A=\{x_1,x_2,\ldots,x_n\}$ sezgisel bulanık mantık kümesi tanımlı olmak üzere, $\mu_A(x)$, $\upsilon_A(x)$ ve $\pi_A(x)$ değerlerini üstel karşılığı;

$$IFE(A)=\sum_{i=1}^{n}\mu_A(x_i)e^{[1-\mu_A(x_i)]} \quad (3.28)$$

olarak elde edilir [51].

IFE (İntiutionistic Fuzzy Entropy) 'ye göre, A → B görüntüleri arasındaki diverjans değeri Denklem 3.29'dan elde edilir.

$$D_1(A,B) = \sum_i\sum_j\left(1-\left(1-\mu_A(a_{ij})\right)e^{\mu_A(a_{ij})-\mu_B(b_{ij})}-\mu_A(a_{ij})\,e^{\mu_B(b_{ij})-\mu_A(a_{ij})}\right) \quad (3.292)$$

IFE ‘ye göre, şekilde B → A olan diverjans değeri ise Denklem 3.30‘dan elde edilir.

$$D_1(B,A) = \sum_i \sum_j \left(1 - \left(1-\mu_B(b_{ij})\right) e^{\mu_B(a_{ij}) - \mu_A(b_{ij}) - \mu_B(b_{ij})} e^{\mu_A(b_{ij}) - \mu_B(a_{ij})}\right) \quad (3.303)$$

Sezgisel bulanık mantık küme teoremindeki tereddüt değeri dikkate alınmadan, toplam diverjans değeri Denklem 3.31’dan elde edilir.

$$D_1(A,B) + D_1(B,A) = \sum_i \sum_j 2 - \left(1-\mu_A(a_{ij})+\mu_B(b_{ij})\right) e^{\mu_A(a_{ij}) - \mu_B(b_{ij})} - \left(1-\mu_B(b_{ij})+\mu_A(a_{ij})\right) e^{\mu_B(b_{ij}) - \mu_A(a_{ij})} \quad (3.31)$$

Sezgisel bulanık mantık küme teoremindeki tereddüt değeri dikkate alınarak, toplam diverjans değeri Denklem 3.32‘den elde edilir;

$$D_2(A,B)+D_2(B,A) = \sum_i \sum_j 2-\left[1-\left(\mu_A(a_{ij}) - \mu_B(b_{ij})\right)+\left(\pi_B(b_{ij}) - \pi_A(a_{ij})\right)\right] e^{\mu_A(a_{ij}) - \mu_B(b_{ij})-\left(\pi_B(b_{ij}) - \pi_A(a_{ij})\right)} - \left[1-\left(\pi_B(b_{ij}) - \pi_A(a_{ij})\right)+\left(\mu_A(a_{ij}) - \mu_B(b_{ij})\right)\right] e^{\pi_B(b_{ij}) - \pi_A(a_{ij})-\left(\mu_A(a_{ij}) - \mu_B(b_{ij})\right)} \quad (3.32)$$

Denklem 3.31 ve Denklem 3.32’de hesaplanan, A ve B sezgisel bulanık mantık kümeleri arasındaki A → B ve B → A diverjansları ile elde edilen İntiutionistic Fuzzy Divergence formülü Denklem 3.33’de gösterildiği gibi elde edilir.

$$IFD(A,B) = D_1(A,B) + D_1(B,A) + D_2(A,B) + D_2(B,A) \quad (3.33)$$

$$IFD(A,B) = \sum_i \sum_j \Big(2 - \left[1 - \mu_A(a_{ij}) + \mu_B(b_{ij})\right] e^{\mu_A(a_{ij}) - \mu_B(b_{ij})} - \left[1 - \mu_B(b_{ij}) + \mu_A(a_{ij})\right] e^{\mu_B(b_{ij}) - \mu_A(a_{ij})} + \begin{pmatrix} 2 - \left[1 - \left(\mu_A(a_{ij}) - \mu_B(b_{ij})\right) + \left(\pi_B(b_{ij}) - \pi_A(a_{ij})\right)\right] e^{\mu_A(a_{ij}) - \mu_B(b_{ij}) - \left(\pi_B(b_{ij}) - \pi_A(a_{ij})\right)} \\ -\left[1 - \left(\pi_B(b_{ij}) - \pi_A(a_{ij})\right) + \left(\mu_A(a_{ij}) - \mu_B(b_{ij})\right)\right] e^{(\pi_B(b_{ij}) - \pi_A(a_{ij})) - \left(\mu_A(a_{ij}) - \mu_B(b_{ij})\right)} \end{pmatrix} \quad (3.34)$$

3.2.4.4.3.2 Nesne Çıkartım Algoritması

A={x, $\mu_A(x)$, $\upsilon_A(x)$| x ϵ X} ve B={x, $\mu_B(x)$, $\upsilon_B(x)$| x ϵ X} sezgisel iki bulanık kümeleri tanımlı olsun. Bunlardan A sezgisel bulanık kümesi giriş yapılan görüntü olarak, B sezgisel bulanık kümesi ise şablon olarak kullanılacak bulanık mantık çıkarımları içeren kümeler olarak kullanılmaktadır.

Sezgisel bulanık nesne çıkartma algoritması için 16 adet 3x3'lük 16 farklı durumu içeren bulanık çıkarım kümeleri oluşturulmuştur. Bunlar Şekil 3.22'de gösterilmektedir.

$$\begin{pmatrix} 0 & b & a \\ 0 & b & a \\ 0 & b & a \end{pmatrix}\begin{pmatrix} a & a & a \\ 0 & 0 & 0 \\ b & b & b \end{pmatrix}\begin{pmatrix} a & a & b \\ 0 & 0 & 0 \\ a & a & a \end{pmatrix}\begin{pmatrix} b & b & b \\ 0 & 0 & 0 \\ a & a & a \end{pmatrix}\begin{pmatrix} b & a & a \\ 0 & b & a \\ 0 & 0 & b \end{pmatrix}\begin{pmatrix} b & a & 0 \\ b & a & 0 \\ b & a & 0 \end{pmatrix}\begin{pmatrix} a & 0 & b \\ a & 0 & b \\ a & 0 & b \end{pmatrix}\begin{pmatrix} 0 & 0 & 0 \\ b & b & b \\ a & a & a \end{pmatrix}$$

$$\begin{pmatrix} a & a & a \\ b & b & b \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} a & b & 0 \\ a & b & 0 \\ a & b & 0 \end{pmatrix}\begin{pmatrix} 0 & 0 & 0 \\ a & a & a \\ b & b & b \end{pmatrix}\begin{pmatrix} 0 & a & b \\ 0 & a & b \\ 0 & a & b \end{pmatrix}\begin{pmatrix} b & b & b \\ a & a & a \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} b & 0 & a \\ b & 0 & a \\ b & 0 & a \end{pmatrix}\begin{pmatrix} b & 0 & 0 \\ a & b & 0 \\ a & a & b \end{pmatrix}\begin{pmatrix} 0 & 0 & b \\ 0 & b & a \\ b & a & a \end{pmatrix}$$

Şekil 3.22 Sezgisel Bulanık Mantık 16'lık Çıkartım Kümesi

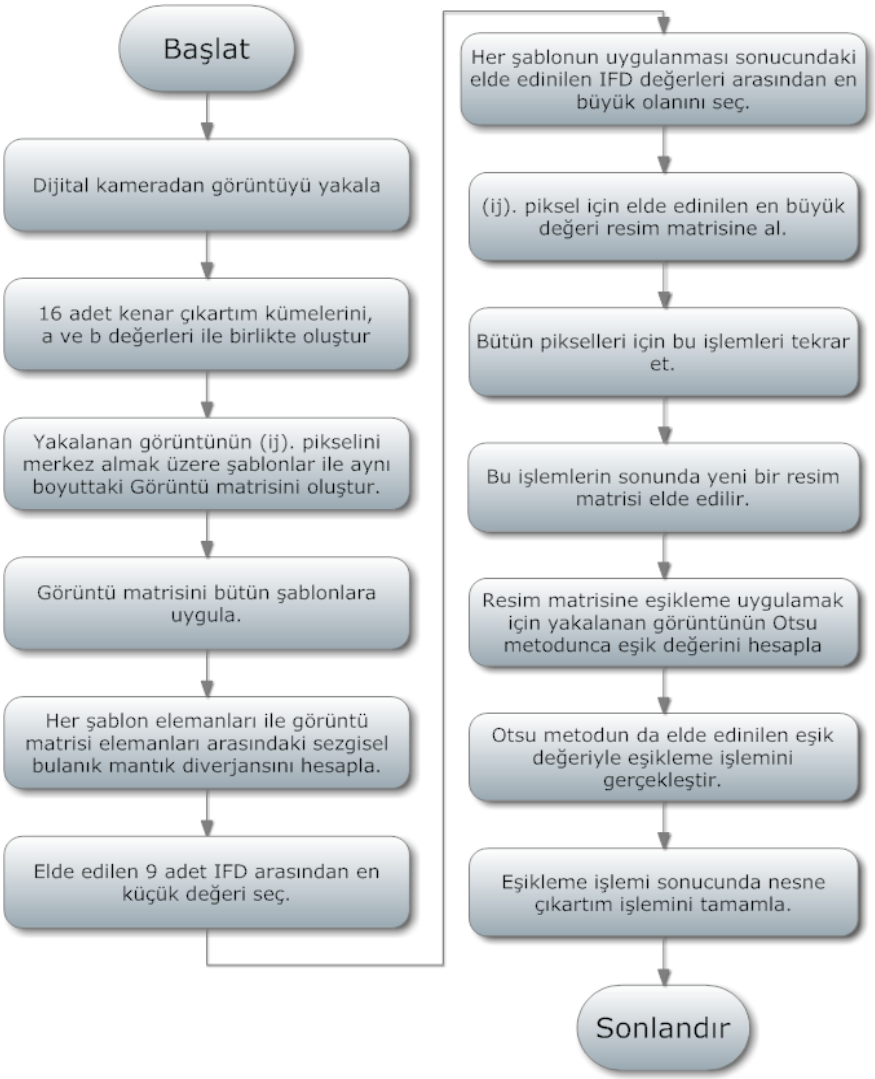
Bulanık Çıkarım kümelerinin seçimi çok önem arz etmektedir. Çünkü kenarın tipini ve yönünü belirtmektedir. Bu kümeler kenarların örneklerini aynı zamanda resimleri ifade eder. a, b ve 0 kenar örneklerinin piksel karşılığını ifade eder; a ve b nin değerleri tamamen deneme yanılma yöntemi ile bulunur. Fakat yapılan literatür taramasında en uygun a=0.3 b=0.8 olarak tahmin edildiği görülmüştür. Her bulanık çıkarım kümesinin merkezine (i,j) pozisyonuna normalize edilmiş resim üzerinden yerleştirilir. Resimdeki her (i,j) piksel pozisyonunda IFD (intuitionistic fuzzy divergence) ölçülür. IFD(i,j) değeri, bulanık çıkarım kümeleri ile aynı boyuttaki görüntü resmi ile bulanık çıkarım kümelerinin Denklem 3.35'den MAX-MIN ilişkisi ile elde edilir.

$$IFD(i,j) = \underset{N}{MAX}\left[\underset{r}{MIN}\left(IFD(A,B) \right) \right] \tag{3.354}$$

Denklem 3.35'deki IFD(a_{ij}, bij), bulanık çıkartım kümeleri (b_{ij}) ve onların görüntü resimleri olan (a_{ij})'nin her birisinin elemanları arasındadır. N bulanık çıkarım küme sayısını ve r bulanık çıkarım kümelerinin karelerinin eleman sayısını temsil etmektedir. IFD(i,j), resmin bütün piksel pozisyonlarının işlenmesinden sonra elde edilmektedir. Elde edilen IFD matrisi, orijinal resimle aynı ölçüde bir matris olmaktadır. IFD matrisini

eşikleme işlemine tabi tutarak kenar çıkarımları elde edilmiş görüntüyü elde ederiz. Yapılan literatür araştırmalarında, eşikleme işlemi, deneme yanılma yöntemiyle belirlenen sabit bir eşik değeri alınarak yapılmaktadır. Sabit eşik değeri, görüntüler arasında farklılık oluştuğunda hata payına neden olmaktadır. Bu nedenle, görüntünün sayısal değerlerine göre değişen dinamik bir eşik değeri belirleme ihtiyacı duyulmuştur. Yapılan literatür taramasında, görüntünün sayısal değerine göre otomatik eşik değeri hesaplama metodu olan Otsu yönteminin, geliştirilen sisteme oldukça uygun olduğuna karar verilmiştir.

Geliştirilen sistemde, pamuk yığınları arasındaki yabancı maddelerin tespit edilmesi için sezgisel bulanık mantık algoritması kullanılmaktadır. Sezgisel bulanık mantık nesne çıkartım algoritması Şekil 3.23‘ de görülmektedir.



Şekil 3.23 Sezgisel Bulanık Mantık Nesne Çıkartım Algoritması

3.2.4.4.3.3 Otsu Metodu

Otsu metodu, Nobuyuki Otsu tarafından öne sürülen gri seviyeli görüntüler üzerinde eşik değeri hesaplama metodudur. Bu metodun temeli, görüntünün arka plan ve ön plan olarak iki farklı renk sınıfından oluştuğu varsayılmaktadır. Varyansın minimum olması için, bu iki sınıfı ayıran en uygun eşik değeri hesaplanır. Eşik değeri hesaplandıktan sonra, değerin altındaki değerlerin tamamı 0'a, üstündeki değerlerin tamamı ise 255'e çevrilmektedir. Otsu metodu, renklerin görüntü üzerinde kaçar defa bulunduğuna baktığından, ilk olarak histogramı çıkartılmasına ve bu histogram üzerinden varyans hesaplama yoluyla eşik değeri hesaplanmasına dayanmaktadır [52].

Varyans, bir sayı dizisinin, aritmetik ortalaması üzerindeki dağılımının gözlemlenmesi ve yorumlanmasını sağlayan bir ölçüttür. Varyansın büyük olması sayı dizisindeki değerlerin aritmetik ortalamalarından uzak yani dağınık olduğunu, varyansın küçük olması ise sayı dizisindeki değerlerin aritmetik ortalamalarına yakın yani dağınık olmadığını göstermektedir [53]. Varyansın karekökü standart sapma olarak adlandırılmaktadır. Varyans değeri Denklem 3.36'dan elde edilir.

$$\sigma^2 = \sum_{i=1}^{N} \left(x_i - \overline{x}_i \right)^2 \Pr\left(x_i \right) \tag{3.365}$$

- x_i : Beklenen değer.
- $\Pr\left(x_i \right)$: Beklenen değerin olasılık değeri.
- $\overline{x}_i$: ağırlıklı ortalamadır.

Her x_i değerinin $\overline{x}_i$ ile farkı hesaplanıp karesi alınır ve bu değer olasılık değeri ile çarpılır. Görüntünün varyansı basit olarak, x_i değerinin resimdeki sayısının, toplam piksel sayınına bölümü olarak ifade edilebilmektedir.

Otsu metodu, sınıf içi varyanslarının minimum olduğu eşikler için, iki renk sınıfının varyanslarının ağırlıklı toplamları olarak tanımlanmaktadır. Otsu metodunda eşik değeri Denklem 3.37'den elde edilmektedir.

$$\sigma_{\omega}^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t) \tag{3.37}$$

- ω_i : ağırlık sınıfı (1 ve 2.sınıf)

- t : eşik değeri

Otsu, sınıflar içindeki varyansların minimum değeri ile sınıflar arası varyansların maksimum değerinin aynı olduğunu göstermiştir.

$$\sigma_b^2(t)=\sigma^2-\sigma_\omega^2(t)=\omega_1(t)\omega_2(t)\left[\mu_1(t)-\mu_2(t)\right]^2 \tag{3.38}$$

- μ_i : sınıf ortalaması
- ω_1 : sınıf olasılığı

Sınıf olasılığı olan $\omega_1(t)$, eşik değeri t olarak histogram üzerinden hesaplanmasıyla bulunmaktadır.

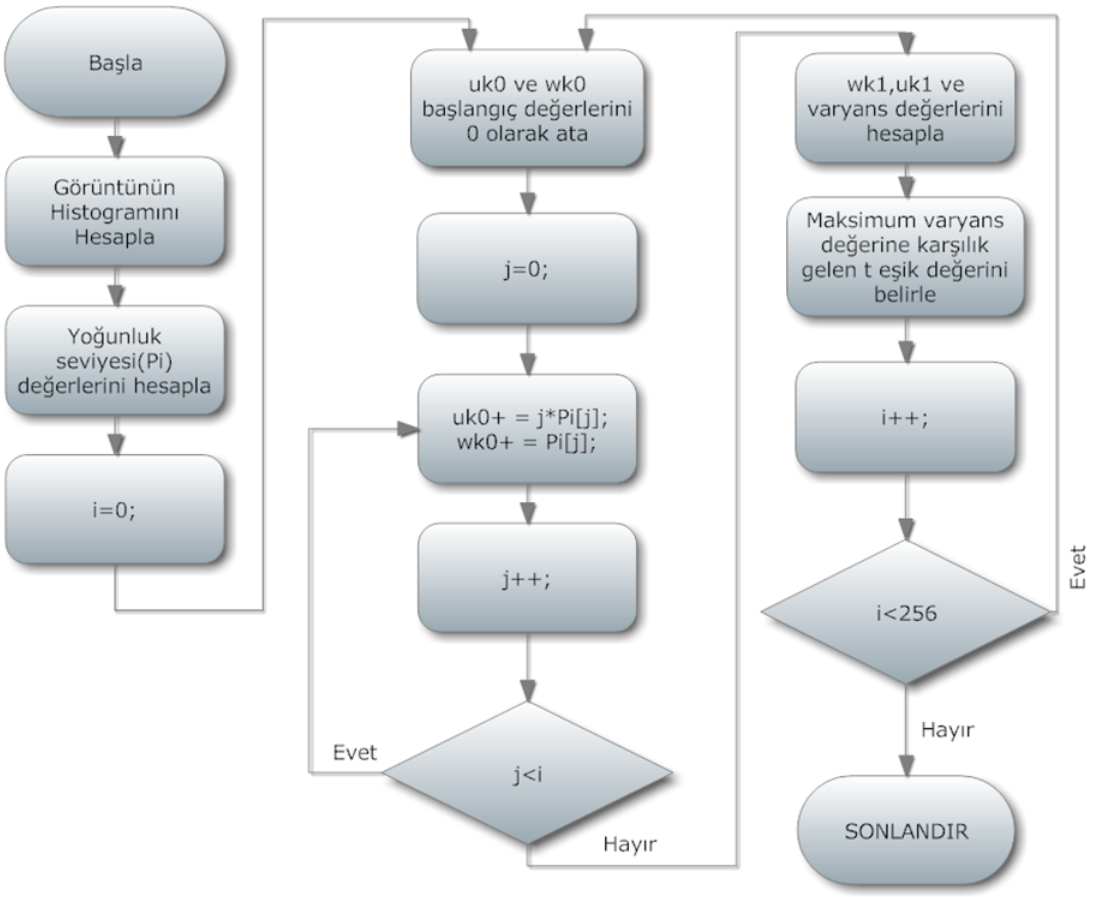
$$\omega_1(t)=\sum_0^t p(i) \tag{3.39}$$

Sınıf ortalaması μ_i ise;

$$\mu_1(t)=\sum_0^t p(i)\chi(i) \tag{3.406}$$

Denklem 3.40'daki $\chi(i)$ i. histogram değerinin merkez değeridir. Benzer olarak $\omega_2(t)$ değeri ise histogram üzerinde t eşik değerinden büyük olan değerler için hesaplanmaktadır.

Geliştirilen sistemde, otsu metodu ile sezgisel bulanık mantık nesne çıkartım algoritmasının otomatik eşikleme işlemi gerçekleştirilmiştir. Geliştirilen sistemde otsu metodunun algoritması Şekil 3.24'de görülmektedir.



Şekil 3.24 Otsu Metodu Algoritması

4. BULGULAR ve TARTIŞMA

4.1. Bulgular

Geliştirilen sistemin performansı, çeşitli örüntüler üzerinden sınanmış ve olumlu sonuçlar elde edilmiştir. Bu sınama süreçlerinde kullanılan örüntüler, hem CPU hem de GPU tabanlı uygulamada test edilmiştir. Test işlemlerinde, örüntünün kameradan alınan görüntüsü, CPU tabanlı uygulamanın çıktı görüntüsü ve GPU tabanlı uygulamanın çıktı görüntü elde edilerek, bu görüntülerin farklı çözünürlükteki örneklerinin çıktısının elde edilme sürelerine ilişkin veriler hesaplanmış olup ilgili örüntü örneğinde görülmektedir.

4.1.1. Birinci Örüntü Örneği

Şekil 4.1'de görülmekte olan kameradan yakalanan görüntünün CPU tabanlı uygulamada işlenmesi sonucu elde edilen görüntü Şekil 4.2'de ve GPU tabanlı uygulamada işlenmesi sonucu elde edilen görüntü ise Şekil 4.3'de görülmektedir.

Şekil 4.1 Kameradan Yakalanan Görüntünü

Şekil 4.2 CPU Tabanlı Uygulamanın Çıktısı

Şekil 4.3 GPU Tabanlı Uygulamanın Çıktısı

Çizelge 4.1'de görülmekte olan 640*100 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 1682 ms yani yaklaşık 1,7 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 9 ms yani 0,009 sn sürdüğü

görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 190 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği yaklaşık 1,5 saniyede 1 frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 120 frame incelendiğinden oldukça iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.2'de görülmekte olan 1280*200 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 7458 ms yani yaklaşık 7,5 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 30 ms yani 0,03 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 238 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 7,5 saniyede bir frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 30 frame incelendiğinden iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.3'te görülmekte olan 1920*300 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 17995 ms yani yaklaşık 18 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 71 ms yani 0,07 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 252 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 18 saniyede bir frame incelendiğinden mümkün olmamaktadır. Aynı zamanda, GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 14 frame olduğundan mümkün olmaktadır.

CPU tabanlı ve GPU tabanlı geliştirilen uygulamaların Çizelge 4.1, Çizelge 4.2 ve Çizelge 4.3'te görülmekte olan farklı çözünürlükteki görüntülerin cevap süreleri incelendiğinde, GPU tabanlı geliştirilen uygulamanın oldukça fazla performans sağladığı görülmektedir. Fakat her ne kadar CPU tabanlı geliştirilen uygulamaya göre performansı yüksek olsa da görüntü çözünürlüğü artıkça sistemin gerçek zamanlı uygulanabilirliği kaybolmaktadır.

Şekil 4.1'de görüldüğü üzere kameradan yakalanan görüntüde her hangi bir yabancı madde bulunan bölge yoktur. Bu da, hem Şekil 4.2'de görülmekte olan CPU tabanlı uygulama

çıktısında hem de Şekil 4.3'te görülmekte olan GPU tabanlı uygulama çıktısında görülmektedir.

Çizelge 4.1 Birinci Örüntünün 640*100 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	0,501737	0,14438
Eşik Değeri	195	195
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	1682,316753	8,685
Toplam Cevap Süresi (ms)	1682,818	8,82938
Yabancı Maddenin Bulunduğu Bölgeler	YOK	YOK
Otsu Metot Oranı (CPU/GPU)	3,475114	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	193,7037	
Cevap Suresi Oranı (CPU/GPU)	190,5931	

Çizelge 4.2 Birinci Örüntünün 1280*200 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	2,172485	0,42726
Eşik Değeri	195	195
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	7458,596148	30,874
Toplam Cevap Süresi (ms)	7460,769	31,30126
Yabancı Maddenin Bulunduğu Bölgeler	YOK	YOK
Otsu Metot Oranı (CPU/GPU)	5,084691	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	241,5818	
Cevap Suresi Oranı (CPU/GPU)	238,3536	

Çizelge 4.3 Birinci Örüntünün 1920*300 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	3,064007	0,29232
Eşik Değeri	195	195
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	17992,23912	70,932
Toplam Cevap Süresi (ms)	17995,3	71,22432
Yabancı Maddenin Bulunduğu Bölgeler	YOK	YOK
Otsu Metot Oranı (CPU/GPU)	10,48169	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	253,6548	
Cevap Suresi Oranı (CPU/GPU)	252,6567	

4.1.2. İkinci Örüntü Örneği

Şekil 4.4 Kameradan Yakalanan Görüntü

Şekil 4.5 CPU Tabanlı Uygulamanın Çıktısı

Şekil 4.6 GPU Tabanlı Uygulamanın Çıktısı

Çizelge 4.4'te görülmekte olan 640*100 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 1741 ms yani yaklaşık 1,7 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 9 ms yani 0,09 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 193 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 1,7 saniyede 1 frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 110 frame olduğundan oldukça iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.5'te görülmekte olan 1280*200 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 8043 ms yani yaklaşık 8 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 32 ms yani 0,03 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 247 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 8 saniyede bir

frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 30 frame incelendiğinden iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.6'da görülmekte olan 1920*300 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 17699 ms yani yaklaşık 18 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 71 ms yani 0,07 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 248 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 18 saniyede bir frame incelendiğinden mümkün olmamaktadır. Aynı zamanda, GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 14 frame incelendiğinden mümkün olmaktadır.

CPU tabanlı ve GPU tabanlı geliştirilen uygulamaların Çizelge 4.4, Çizelge 4.5 ve Çizelge 4.6'da görülmekte olan farklı çözünürlükteki görüntülerin cevap süreleri incelendiğinde, GPU tabanlı geliştirilen uygulamanın oldukça fazla performans sağladığı görülmektedir. Fakat her ne kadar CPU tabanlı geliştirilen uygulamaya göre performansı yüksek olsa da görüntü çözünürlüğü artıkça sistemin gerçek zamanlı uygulanabilirliği kaybolmaktadır.

Şekil 4.4'te görüldüğü üzere kameradan yakalanan görüntüde her hangi bir yabancı madde bulunan bölge yoktur. Bu da, hem Şekil 4.5'te görülmekte olan CPU tabanlı uygulama çıktısında hem de Şekil 4.6'da görülmekte olan GPU tabanlı uygulama çıktısında görülmektedir.

Çizelge 4.4 İkinci Örüntünün 640*100 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	0,49491	0,12541
Eşik Değeri	176	176
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	1741,095711	8,853
Toplam Cevap Süresi (ms)	1741,591	8,97841
Yabancı Maddenin Bulunduğu Bölgeler	YOK	YOK
Otsu Metot Oranı (CPU/GPU)	3,946336	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	196,6673	
Cevap Suresi Oranı (CPU/GPU)	193,9754	

Çizelge 4.5 İkinci Örüntünün 1280*200 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	1,447869	0,38314
Eşik Değeri	176	176
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	8042,243493	32,173
Toplam Cevap Süresi (ms)	8043,691	32,55614
Yabancı Maddenin Bulunduğu Bölgeler	YOK	YOK
Otsu Metot Oranı (CPU/GPU)	3,778955	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	249,9687	
Cevap Suresi Oranı (CPU/GPU)	247,0714	

Çizelge 4.6 İkinci Örüntünün 1920*300 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	3,082096	0,3105
Eşik Değeri	176	176
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	17696,11554	70,826
Toplam Cevap Süresi (ms)	17699,2	71,1365
Yabancı Maddenin Bulunduğu Bölgeler	YOK	YOK
Otsu Metot Oranı (CPU/GPU)	9,926235	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	249,8534	
Cevap Suresi Oranı (CPU/GPU)	248,8061	

4.1.3. Üçüncü Örüntü Örneği

Şekil 4.7 Kameradan Yakalanan Görüntü

Şekil 4.8 CPU Tabanlı Uygulamanın Çıktısı

Şekil 4.9 GPU Tabanlı Uygulamanın Çıktısı

Çizelge 4.7'de görülmekte olan 640*100 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 1711 ms yani yaklaşık 1,7 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 9 ms yani 0,009 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 194 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 1,7 saniyede 1 frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 110 frame incelendiğinden oldukça iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.8'de görülmekte olan 1280*200 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 8021 ms yani yaklaşık 8 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 31 ms yani 0,03 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 258 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 8 saniyede bir frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek

zamanlı olarak uygulanabilirliği, saniyede yaklaşık 32 frame incelendiğinden iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.9'da görülmekte olan 1920*300 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 17517 ms yani yaklaşık 18 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 71 ms yani 0,07 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 245 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 18 saniyede bir frame incelendiğinden mümkün olmamaktadır. Aynı zamanda, GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 14 frame incelendiğinden mümkün olmaktadır.

CPU tabanlı ve GPU tabanlı geliştirilen uygulamaların Çizelge 4.7, Çizelge 4.8 ve Çizelge 4.9'da görülmekte olan farklı çözünürlükteki görüntülerin cevap süreleri incelendiğinde, GPU tabanlı geliştirilen uygulamanın oldukça fazla performans sağladığı görülmektedir. Fakat her ne kadar CPU tabanlı geliştirilen uygulamaya göre performansı yüksek olsa da görüntü çözünürlüğü artıkça sistemin gerçek zamanlı uygulanabilirliği kaybolmaktadır.

Şekil 4.7'de görüldüğü üzere kameradan yakalanan görüntüde yabancı madde bulunmaktadır. Bu da, hem Şekil 4.8'de görülmekte olan CPU tabanlı uygulama çıktısında hem de Şekil 4.9'da görülmekte olan GPU tabanlı uygulama çıktısında görülmektedir. Çizelge 4.7, Çizelge 4.8 ve Çizelge 4.9 'da yabancı maddenin bulunduğu bölgeler incelendiğinde, 640x100 çözünürlükte bulunan yabancı madde bölgeleri 1-2 iken, daha büyük çözünürlük değerinde bu bölgeler 1-2-3-5-7 olmuştur. Bu nedenle çözünürlük değeri artıkça sistemin hatalı sonuçlar elde ettiği görülmektedir.

Çizelge 4.7 Üçüncü Örüntünün 640*100 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	0,496276	0,09811
Eşik Değeri	146	146
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	1710,541998	8,719
Toplam Cevap Süresi (ms)	1711,038	8,81711
Yabancı Maddenin Bulunduğu Bölgeler	1-2	1-2
Otsu Metot Oranı (CPU/GPU)	5,058363	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	196,1855	
Cevap Suresi Oranı (CPU/GPU)	194,0589	

Çizelge 4.8 Üçüncü Örüntünün 1280*200 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	1,530809	0,30256
Eşik Değeri	146	146
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	8019,550321	30,736
Toplam Cevap Süresi (ms)	8021,081	31,03856
Yabancı Maddenin Bulunduğu Bölgeler	1-2	1-2
Otsu Metot Oranı (CPU/GPU)	5,059522	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	260,9172	
Cevap Suresi Oranı (CPU/GPU)	258,4231	

Çizelge 4.9 Üçüncü Örüntünün 1920*300 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	3,040114	0,3495
Eşik Değeri	146	146
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	17517,07509	71,069
Toplam Cevap Süresi (ms)	17520,12	71,4185
Yabancı Maddenin Bulunduğu Bölgeler	1-2-3-5-7	1-2-3-5-7
Otsu Metot Oranı (CPU/GPU)	8,698466	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	246.479	
Cevap Suresi Oranı (CPU/GPU)	245.316	

4.1.4. Dördüncü Örüntü Örneği

Şekil 4.10 Kameradan Yakalanan Görüntü

Şekil 4.11 CPU Tabanlı Uygulamanın Çıktısı

Şekil 4.12 GPU Tabanlı Uygulamanın Çıktısı

Çizelge 4.10'te görülmekte olan 640*100 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 1714,777 ms yani yaklaşık 1,7 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 8,7593 ms yani 0,09 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 195 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 1,7 saniyede 1 frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 110 frame incelendiğinden oldukça iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.11'te görülmekte olan 1280*200 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 8078,566 ms yani yaklaşık 8 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 30,787ms yani 0,03 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 259 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 8 saniyede bir frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı

uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 32 frame incelendiğinden iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.12'tegörülmekte olan 1920*300 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 17512,87ms yani yaklaşık 18 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 71,257 ms yani 0,07 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 245 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 18 saniyede bir frame incelendiğinden mümkün olmamaktadır. Aynı zamanda, GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 13 frame incelendiğinden mümkün olmamaktadır.

CPU tabanlı ve GPU tabanlı geliştirilen uygulamaların Çizelge 4.10, Çizelge 4.11 ve Çizelge 4.12'te görülmekte olan farklı çözünürlükteki görüntülerin cevap süreleri incelendiğinde, GPU tabanlı geliştirilen uygulamanın oldukça fazla performans sağladığı görülmektedir. Fakat her ne kadar CPU tabanlı geliştirilen uygulamaya göre performansı yüksek olsa da görüntü çözünürlüğü artıkça sistemin gerçek zamanlı uygulanabilirliği kaybolmaktadır.

Şekil 4.10'da görüldüğü üzere kameradan yakalanan görüntüde yabancı madde bulunmaktadır. Bu da, hem Şekil 4.11' de görülmekte olan CPU tabanlı uygulama çıktısında hem de Şekil 4.12'de görülmekte olan GPU tabanlı uygulama çıktısında görülmektedir. Çizelge 4.10, Çizelge 4.11 ve Çizelge 4.12'te yabancı maddenin bulunduğu bölgeler incelendiğinde, 640x100 çözünürlükte bulunan yabancı madde bölgeleri 4-5 iken, daha büyük çözünürlük değerlerinde bu bölgeler 4-5-6 olmuştur. Bu nedenle çözünürlük değeri artıkça sistemin hatalı sonuçlar elde ettiği görülmektedir.

Çizelge 4.10 Dördüncü Örüntünün 640*100 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	0,480575	0,0983
Eşik Değeri	138	138
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	1714,29649	8,661
Toplam Cevap Süresi (ms)	1714,777	8,7593
Yabancı Maddenin Bulunduğu Bölgeler	4-5	4-5
Otsu Metot Oranı (CPU/GPU)	4,888861	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	197,9329	
Cevap Suresi Oranı (CPU/GPU)	195,7664	

Çizelge 4.11 Dördüncü Örüntünün 1280*200 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	1,465617	0,2951
Eşik Değeri	138	138
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	8078,566494	30,787
Toplam Cevap Süresi (ms)	8080,032	31,0821
Yabancı Maddenin Bulunduğu Bölgeler	4-5-6	4-5-6
Otsu Metot Oranı (CPU/GPU)	4,96651	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	262,4019	
Cevap Suresi Oranı (CPU/GPU)	259,9577	

Çizelge 4.12 Dördüncü Örüntünün 1920*300 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	3,043869	0,28736
Eşik Değeri	138	138
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	17509,82277	70,97
Toplam Cevap Süresi (ms)	17512,87	71,25736
Yabancı Maddenin Bulunduğu Bölgeler	4-5-6	4-5-6
Otsu Metot Oranı (CPU/GPU)	10,59253	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	246,7215	
Cevap Suresi Oranı (CPU/GPU)	245,7692	

4.1.5. Beşinci Örüntü Örneği

Şekil 4.13 Kameradan Yakalanan Görüntü

Şekil 4.14 CPU Tabanlı Uygulamanın Çıktısı

Şekil 4.15 GPU Tabanlı Uygulamanın Çıktısı

Çizelge 4.13'te görülmekte olan 640*100 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 1828,522 ms yani yaklaşık 1,8 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 9,698 ms yani 0,01 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 186 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 1,8 saniyede 1 frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 100 frame incelendiğinden oldukça iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.14'te görülmekte olan 1280*200 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 7883,594 ms yani yaklaşık 8 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 32,754 ms yani 0,03 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 240 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 8 saniyede bir frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek

zamanlı olarak uygulanabilirliği, saniyede yaklaşık 30 frame incelendiğinden iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.15'te görülmekte olan 1920*300 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 17517,077 ms yani yaklaşık 18 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 71,125ms yani 0,07 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 245 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 18 saniyede bir frame incelendiğinden mümkün olmamaktadır. Aynı zamanda, GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 13frame incelendiğinden mümkün olmamaktadır.

CPU tabanlı ve GPU tabanlı geliştirilen uygulamaların Çizelge 4.13, Çizelge 4.14 ve Çizelge 4.15'te görülmekte olan farklı çözünürlükteki görüntülerin cevap süreleri incelendiğinde, GPU tabanlı geliştirilen uygulamanın oldukça fazla performans sağladığı görülmektedir. Fakat her ne kadar CPU tabanlı geliştirilen uygulamaya göre performansı yüksek olsa da görüntü çözünürlüğü artıkça sistemin gerçek zamanlı uygulanabilirliği kaybolmaktadır.

Şekil 4.13'te görüldüğü üzere kameradan yakalanan görüntüde ikinci, üçüncü, beşinci, altıncı ve sekizinci bölgelerde yabancı madde bulunmaktadır. Bu da, hem Şekil 4.14'te görülmekte olan CPU tabanlı uygulama çıktısında hem de Şekil 4.15'te görülmekte olan GPU tabanlı uygulama çıktısında görülmektedir.

Çizelge 4.13 Beşinci Örüntünün 640*100 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	0,574096	0,09827
Eşik Değeri	145	145
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	1827,948027	9,698
Toplam Cevap Süresi (ms)	1828,522	9,79627
Yabancı Maddenin Bulunduğu Bölgeler	2-3-5-6-8	2-3-5-6-8
Otsu Metot Oranı (CPU/GPU)	5,842027	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	188,4871	
Cevap Suresi Oranı (CPU/GPU)	186,6549	

Çizelge 4.14 Beşinci Örüntünün 1280*200 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	1,451623	0,33725
Eşik Değeri	145	145
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	7882,142067	32,417
Toplam Cevap Süresi (ms)	7883,594	32,75425
Yabancı Maddenin Bulunduğu Bölgeler	2-3-5-6-8	2-3-5-6-8
Otsu Metot Oranı (CPU/GPU)	4,304294	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	243,1484	
Cevap Suresi Oranı (CPU/GPU)	240,6892	

Çizelge 4.15 Beşinci Örüntünün 1920*300 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	3,061959	0,32176
Eşik Değeri	145	145
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	17517,07714	71,125
Toplam Cevap Süresi (ms)	17520,14	71,44676
Yabancı Maddenin Bulunduğu Bölgeler	2-3-5-6-8	2-3-5-6-8
Otsu Metot Oranı (CPU/GPU)	9,516282	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	246.2857	
Cevap Suresi Oranı (CPU/GPU)	245,2195	

4.1.6. Altıncı Örüntü Örneği

Şekil 4.16 Kameradan Yakalanan Görüntü

Şekil 4.17 CPU Tabanlı Uygulamanın Çıktısı

Şekil 4.18 GPU Tabanlı Uygulamanın Çıktısı

Çizelge 4.16'da görülmekte olan 640*100 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 1724,038 ms yani yaklaşık 1,8 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 8,72408 ms yani 0,09 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 197 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 1,8 saniyede 1 frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 110 frame incelendiğinden oldukça iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.17'de görülmekte olan 1280*200 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 7883,594 ms yani yaklaşık 8 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 32,754 ms yani 0,03 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 253 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 8 saniyede bir frame incelendiğinden mümkün olmamaktadır. Fakat GPU tabanlı uygulamanın gerçek

zamanlı olarak uygulanabilirliği, saniyede yaklaşık 32 frame incelendiğinden iyi sonuç elde edilmesini sağlamaktadır.

Çizelge 4.18'da görülmekte olan 1920*300 çözünürlükteki görüntüye ait veriler incelendiğinde, CPU tabanlı uygulamanın cevap süresinin yaklaşık 17520,14ms yani yaklaşık 18 sn sürdüğü ve GPU tabanlı uygulamanın cevap süresinin yaklaşık 71,446 ms yani 0,07 sn sürdüğü görülmektedir. Sistemin cevap süreleri oranlandığında yaklaşık 246 kat fark olduğu görülmektedir. Sistemde kullanılan kameradan yakalanan görüntü sayısı 50 fps olduğundan, CPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, yaklaşık 18 saniyede bir frame incelendiğinden mümkün olmamaktadır. Aynı zamanda, GPU tabanlı uygulamanın gerçek zamanlı olarak uygulanabilirliği, saniyede yaklaşık 13 frame incelendiğinden mümkün olmaktadır.

CPU tabanlı ve GPU tabanlı geliştirilen uygulamaların Çizelge 4.16, Çizelge 4.17 ve Çizelge 4.18'de görülmekte olan farklı çözünürlükteki görüntülerin cevap süreleri incelendiğinde, GPU tabanlı geliştirilen uygulamanın oldukça fazla performans sağladığı görülmektedir. Şekil 4.16'da görüldüğü üzere kameradan yakalanan görüntüde birinci, ikinci, altıncı, yedinci ve sekizinci bölgelerde yabancı madde bulunmaktadır. Bu da, hem Şekil 4.17'de görülmekte olan CPU tabanlı uygulama çıktısında hem de Şekil 4.18'de görülmekte olan GPU tabanlı uygulama çıktısında görülmektedir.

Çizelge 4.16 Altıncı Örüntünün 640*100 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	0,47682	0,09808
Eşik Değeri	136	136
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	1723,561211	8,626
Toplam Cevap Süresi (ms)	1724,038	8,72408
Yabancı Maddenin Bulunduğu Bölgeler	1-2-6-7-8	1-2-6-7-8
Otsu Metot Oranı (CPU/GPU)	4,861542	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	199,81	
Cevap Suresi Oranı (CPU/GPU)	197,6183	

Çizelge 4.17 Altıncı Örüntünün 1280*200 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	1,759833	0,30237
Eşik Değeri	136	136
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	8017,585	31,60637
Toplam Cevap Süresi (ms)	8019,344	31,90874
Yabancı Maddenin Bulunduğu Bölgeler	1-2-6-7-8	1-2-6-7-8
Otsu Metot Oranı (CPU/GPU)	5,820131	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	253,6699	
Cevap Suresi Oranı (CPU/GPU)	251,3212	

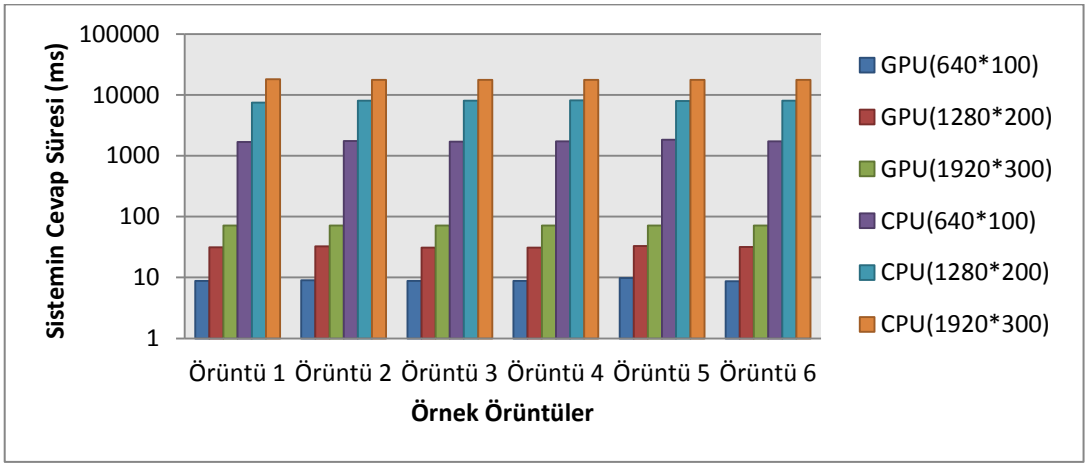
Çizelge 4.18 Altıncı Örüntünün 1920*300 Çözünürlüğündeki Görüntüsünün Çıktı Olarak Elde Edilmesine Kadar Geçen Sürelere İlişkin Veriler

Uygulama	**CPU tabanlı**	**GPU tabanlı**
Otsu Metodunun İşlenme Süresi (ms)	3,053084	0,32973
Eşik Değeri	136	136
Sezgisel Bulanık Mantık İşlenme Süresi(ms)	17575,31	71,33273
Toplam Cevap Süresi (ms)	17578,36	71,66246
Yabancı Maddenin Bulunduğu Bölgeler	1-2-6-7-8	1-2-6-7-8
Otsu Metot Oranı (CPU/GPU)	9,259346	
Sezgisel Bulanık Mantık Oranı (CPU/GPU)	246,3849344	
Cevap Suresi Oranı (CPU/GPU)	245,2938831	

4.2. Tartışma

İplik yapımında kullanılan pamuk yığının içindeki yabancı maddelerin ayıklanması işlemleri için çeşitli endüstriyel uygulamalar geliştirilmiştir. Bu uygulamalar incelendiğinde birim zamanda üretilecek iplik miktarını ve kalitesini artırmak için hızlandırılması gerekmektedir. Bu sorunun çözümü için yapılan literatür araştırmasında GPU ile hızlandırılmış sistemlerin onlarca hatta yüzlerce kat performans arttırıldığı görülmüştür [16] [17] [18]. Geliştirilen sistemde hem CPU tabanlı uygulama hem de GPU tabanlı uygulama geliştirilerek, sistemin sağladığı performansın ortaya konulması amaçlanmıştır.

Bulgular bölümündeki, farklı çözünürlük oranlarındaki örüntü örnekleri incelendiğinde sistemin cevap sürelerine ilişkin Şekil 4.19'daki grafik elde edilmektedir.



Şekil 4.19 Sistemin Hızlanma Oranı

Şekil 4.19'daki grafik incelendiğinde, GPU tabanlı uygulamanın çok belirgin bir hız farkı oluşturduğu görülmektedir. GPU tabanlı uygulamanın 640*100 oranındaki görüntülerde ortalama 193, 1280*200 oranındaki görüntülerde ortalama 249 ve 1920*300 oranındaki görüntülerde ise ortalama 247 kat daha hızlı olduğu görülmüştür. Ayrıca, tüm örüntülerde 186 ile 260 kat arasında hız oranı elde edilmiştir.

Yapılan literatür araştırmalarında, GPU ile hızlandırılan başarılı sistemlerde içinde 100 kat fark yakalayan nadir sistemlerin olduğu görülmüştür [18]. Geliştirilen sistem, literatürdeki sistemlerle mukayese edildiğinde, maksimum 260 kat fark sağlaması ile oldukça başarılı bir sonuç elde edilmiştir.

5. SONUÇ

Tekstil ve çırçır fabrikalarında kullanılan yün ve pamuk hammaddelerinden iplik üretimi ve işleme esnasında oluşan veya dışarıdan gelen yabancı maddeler, elde edilen kumaş veya ipliklerin kalitesinde önemli ölçüde düşüşe neden olmaktadır.

Günümüzde tekstil ve işletmelerinde kullanılan metotlar, yabancı maddelerin ayırt edilebilmesi için hız ve kalite açısından verimli değildir. Son yıllardaki bilgisayarlı görüş sistemleri her alanda olduğu gibi tekstil alanında da etkisini belli bir biçimde göstermektedir. Günümüzde kullanılan mevcut sistemlerde, öne çıkan sorunlardan kalite ve hız konusu önemli bir seçici unsurdur. Bu çalışmada ise kameradan alınan görüntüdeki yabancı maddelerin görüntü içerisinden belirlenmesi işlemi için Atanossov tarafından geliştirilen sezgisel bulanık mantık algoritması kullanılmıştır.

Sezgisel bulanık mantık algoritmasında, klasik bulanık mantık algoritmasında bulunmayan uzman deneyim hatalarını minimize etmek için tereddüt değerini hesaplayarak üyelik değerinden çıkartılmaktadır. Bu sayede daha kararlı ve istenilen sonuçlar elde edilmesine olanak tanımaktadır.

Sezgisel bulanık mantık algoritmasının yapısı gereği gerçek zamanlı CPU tabanlı uygulamalarda hız problemlerinin olduğu görülmüştür. Bu nedenle, hız sorununu ortadan kaldırmak için GPU teknolojisinden faydalanılmıştır. Kameradan alınan görüntüye ilk aşamada sezgisel bulanık mantık nesne çıkartma metodu uygulanmıştır. Yapılan literatür taramalarına göre sezgisel bulanık mantık algoritmalarında eşik değeri olarak sabit bir değerin kullanıldığı görülmüştür. Kameradan alınan her çerçeve farklı değerler üzerinden oluşacağından, algoritma istenilen sonuçları tam olarak yapamamaktaydı. Dinamik eşik değeri üreten Otsu metodu, ilk kez sezgisel bulanık mantık metodu ile bütünleşik olarak bu tezde kullanılmıştır. Bu sayede kameradan alınan her çerçevenin eşik değeri gerçek zamanda hesaplanmış ve görüntüye güncel olarak uygulanmıştır.

Sistem geliştirilirken, yapılan denemelerde yakalanan görüntünün farklı çözünürlükteki örnekleri kullanılarak hem performans hem de kalite açısından en uygun çözünürlük değeri 640*100 oranı olduğu görülmüştür. İşlenen bu görüntü, 80*100 oranında 8 eşit bölgeye ayrılmıştır. Her bölgedeki yabancı madde olup olmadığı ilgili bölgede kalan görüntü parçasının sayısal değerleri toplamı üzerinden karar verilmektedir. İlgili bölgede yabancı madde varlığıyla ilgili karar verilmesi için kullanılan bu toplam

değeri ihmal edilebilirlik düzeyine göre yapılan denemeler sonucunda bir eşik değeri belirlenmiştir.

Kullanılan algoritmalar, NVIDIA GTX 480 GPU destekli ekran kartı üzerinden farklı çözünürlükteki örüntü örneklerinde minimum 168 maksimum 262 kat hız farkı elde edilmiştir. Elde dilen bu hız farkı literatürdeki GPU ile hızlandırılmış sistemlere göre oldukça başarılı bir sonuç elde edilmiştir. Çünkü mevcut GPU uygulamalarında 100 kat oranını çok nadir uygulamanın geçtiği görülmüştür.

Yapılan denemeler sonucunda kullanılan metotların istenilen sonuçları eksiksiz olarak verdiği tespit edilmiştir. Bu metotlar sayesinde literatürdeki daha önce yapılan pamuktan yabancı madde ayıklama çalışmalarının aksine bilgisayarlı görüş sistemi hız ve kalite olarak çok daha önemli bir avantaj sağlamıştır.

6. KAYNAKLAR

[1] Smith, C.W., Cothren, J. T., Cotton: Origin, History, Technology, and Production., 1999, s. 64-69. (1999)

[2] Yitik B., Bir Pamuk İplik İşletmesinde Ortaya Çıkan Telef Miktarlarının Azaltılması Ve Bunun Çeşitli İplik Kalite Parametreleri Üzerine Etkisi. Yüksek Lisans Tezi. Marmara Üniversitesi. İstanbul.196 (2006).

[3] Koç, A. A., Dünya Pamuk Piyasalarında Eğilimler Ve Ulusal Tarım Politikasında Değişmelerin Türkiye Pamuk Pazarına Etkisi, Antalya, Turkey. Türkiye VI. Pamuk, Tekstil Ve Konfeksiyon Sempozyumu, 2003, Antalya s.11-20 (2003).

[4] Siddaiah, M., Hughs S.E., Lieberman, M.A. , Nadipuram, R.P., Identification of Trash Types in Ginned Cotton using Neuro Fuzzy Techniques, IEEE International Fuzzy Systems Conference, 1999, s.738-743 (1999).

[5] Pai, A., Sari-Sarraf, H., Hequest, E. F., Recognition of Cotton Contaminants via X-Ray Microtomographic Image Analysis. *IEEE Transactions on Industry applications*, 40 (1) : 77-85 (2004).

[6] Yang, W., Li, D., Zhu, L., Kang, Y., Li, F., A new approach for image processing in foreign fiber detection. *Computers and Electronics in Agriculture*, 68 (1) : 68-77 (2009).

[7] Boshra, D.F., Measurements of Trash Contents and Grades in Cotton Using Digital Image Analysis, Beijing, China. 3th International Conference On Signal Processing, 1996, Beijing s.1545-1548 (1996).

[8] Qu, X., Ding, T.H., A Fast Feature Extraction Algorithm for Detection of Foreign Fiber in Lint Cotton within a Complex Background. *Acta Automatica Sinica*, 36 (6): 785-790 (2010).

[9] Ji, R., Li, D., Chen, L., Yang, W., Classification and identification of foreign fibers in cotton on the basis of a support vector machine. *Mathematical and Computer Modelling*, 51 (11-12) : 1433-1437 (2010).

[10] Zhang, Y., Smith, P.W., Robust and Efficient Detection of Non-Lint Material in Cotton Fiber Samples, 6th IEEE Workshop on Applications of Computer Vision, 2002, s. 335-336. (2002)

[11] Bigand, A., Colot, Olivier, Wenzhu, Fuzzy filter based on interval-valued fuzzy sets for image filtering. *Fuzzy Sets and Systems*, 161 (1): 96-117 (2010).

[12] Chaira, T., Ray, A.K., A new measure using intuitionistic fuzzy set theory and its application to edge detection. *Applied Soft Computing*, 8 (2) : 919–927 (2008).

[13] Mushrif, M.M., Ray, A.K., A-IFS Histon Based Multithresholding Algorithm for Color Image Segmentation. *IEEE Signal Processing Letters*, 16 (3) : 168- 171 (2009).

[14] Powell, K., Biomedical imaging ecosystem and the role of the GPU, International Symposium on Biomedical Imaging, 2009, Santa Clara, USA. s.1291-1292 (2009).

[15] Patel, H., GPU Accelerated Real Time Polarimetric Image Processing through the use of CUDA, Proceedings of the IEEE 2010 National Aerospace and Electronics Conference (NAECON), 2010, s.177-180 (2010).

[16] Mielikainen, J., Huang, B., Huang, H.L.A., Goldberg, M.D., GPU Acceleration of the Updated Goddard Shortwave Radiation Scheme in the Weather Research and Radiation Scheme in the Weather Research. *IEEE Journal Of Selected Topıcs In Applıed Earth Observatıons And Remote Sensıng*, 5 (2) : 555-562 (2012).

[17] Mielikainen, J., Huang, B., Huang, H.L.A., Goldberg, M.D., GPU Implementation of Stony Brook University 5-Class Cloud Microphysics Scheme in the WRF. *IEEE Journal Of Selected Topics In Applied Earth Observations And Remote Sensing*, 5 (2) : 625-633 (2012).

[18] Ashwin, A. M, Zhang L., Feng, W., GPU-RMAP: Accelerating Short-Read Mapping on Graphics Processors, 13th IEEE International Conference on Computational Science and Engineering, 2010, s.169-175 (2010).

[19] Panasonic. Panasonic Resmi Web Sitesi, URL: (erişim tarihi:10/05/2012) http://www.panasonic.com.tr/html/tr_TR/%C3%9Cr%C3%BCnler/Video+Kameralar/SD+Video+Kameralar/HDC-SD80/overview/7401950/index.html?trackInfo=true (2012).

[20] IEEE 1394b Frame Grabbers. (2012) FIW64

[21] Microsoft Corp. Visual Studio Resmi Sitesi, URL: (erişim tarihi:16/05/2012) http://www.microsoft.com/visualstudio/tr-tr (2012)

[22] Nvidia. Nvidia Developer Zone, Toolkit, URL: (erişim tarihi:18/05/2012) http://developer.nvidia.com/cuda-toolkit-40 (2012)

[23] OpenCV. OpenCV Resmi Sitesi, URL: (erişim tarihi:13/04/2012) http://code.opencv.org/projects/opencv/wiki (2012)

[24] Cmake.Cmake Resmi Sitesi, URL: (erişim tarihi:19/04/2012) http://www.cmake.org/ (2012)

[25] BrookGPU Graphics Lab, Stanford University, URL: (erişim tarihi:19/02/2012) http://graphics.stanford.edu/projects/brookgpu/ (2012)

[26] Stein M., CUDA Basics. s: 13 URL: (erişim tarihi:13/05/2012) http://www.cs.nyu.edu/manycores/cuda_many_cores.pdf (2012)

[27] LostCircuits, URL: (erişim tarihi:18/05/2012) http://www.lostcircuits.com/graphics/asus_engtx480/fermi1.gif (2012)

[28] Anandtech, Intel's Pentium D dual core architecture URL: (erişim tarihi:13/05/2012) http://www.anandtech.com/show/1665/2 (2012)

[29] Amdahl, G. M., Validity of the single processor approach to achieving large scale computing capabilities, AFIPS spring joint computer conference, s.1-4 (1967).

[30] IBM (erişim tarihi:17/04/2012) http://www.ibm.com/developerworks/linux/library/l-cluster1/figure3.gif (2012)

[31] Programming Massively Parallel Processors with CUDA, Stanford University, (erişim tarihi: 19/01/2012) http://code.google.com/p/stanford-cs193g-sp2010/ (2012)

[32] CUDA Best Practice Guide, NVIDIA Corp. s: 31 URL :(erişim tarihi: 13/15/2012) http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Best_Practices_Guide.pdf (2012)

[33] Kindratenko. V URL :(erişim tarihi: 16/05/2012) www.ncsa.illinois.edu/~kindr/ (2012)

[34] Wikipedia, URL: (erişim tarihi: 17/05/2012) http://en.wikipedia.org/wiki/Grayscale (2012)

[35] Cuda C Programming Guide, NVIDIA Corp., (erişim tarihi:19/05/2012) http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf

[36] Sanders, Jason, Kandrot, Edward, 2010. CUDA by Example: An Introduction to General-Purpose GPU Programming, 1st ed., Addison-Wesley Professional, 2010.

[37] Zadeh, L.A, Fuzzy sets. *Information and Control*, 8 (3) : 338–353 (1965).

[38] Mamdani, E.H, Application of fuzzy algorithms for control of simple dynamic plant. *Proceedings of the Institution of Electrical Engineers*, 122 (12) : 1585 - 1588 (1974).

[39] Dağdelen Ulvi, Bulanık Mantık ile Adım Motor Kontrolü. Yüksek Lisans Tezi. Erciyes Üniversitesi. Kayseri.SS: 58 (1996).

[40] Atanossov, K.T., Intuitionistic Fuzzy Sets. *Fuzzy Sets and Systems*, 20 (1) : 87-96 (1986).

[41] Atanassov K.T.,, Intuitionistic fuzzy sets: past, present and future, EUSFLAT Conf'03, (2003).

[42] Atanassov K,1999. Intuitionistic Fuzzy Sets: Theory and Applications.: Springer-Verlag, ISBN-13: 978-3790812282 , ss :341 (1999).

[43] Atanassov K.T.,, S., Stoeva, Intuitionistic fuzzy set, Polish Symposium on Interval and Fuzzy Mathematics, Poznan, 1993, s.23-26 (1993).

[44] Chaira, T., Ray, A.K. ,Segmentation using fuzzy divergence. *Pattern Recognition Letters*, 24 (12) : 1837–1844 (2003).

[45] Shannon, C. E, A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27 (1) : 379–423 ; 623–656 (1948).

[46] Kullback, S., Liebler, R.A., On Information and Sufficiency, *The Annal Mathematical Statistics*, 22 : 79-86 (1951).

[47] Hamming, R.W., Error Detecting and Correcting Codes, *The Bell System Technical Journal*, 29 (2) : 2-7 (1950).

[48] Danielsson, P.E., Euclidean Distance Mapping, *Computer Graphics and Image Processing*, 14 : 227-240 (1980).

[49] Szmidt, E., Kacprzyk, J., Distance between intuitionistic fuzzy set, *Fuzzy Sets System*, 14 : 505-518 (2000).

[50] Szmidt, E., Kacprzyk, J., Entropy for intuitionistic fuzzy set, *Fuzzy Sets System*, 118 : 467–477 (2001).

[51] Montes, S., Couso, I., Gil, P., Bertoluzza, C., Divergence measure between fuzzy sets, *International Journal of Approximate Reasoning*, 3 : 91-105 (2002).

[52] Otsu, N., A Threshold Selection Method from Gray-level Histograms, *IEEE Transactions on Systems, Man and Cybernetic*, 9 : 62-66 (1979).

[53] Varyans, Wikipedia. URL:(erişim tarihi:22/05/2012) http://tr.wikipedia.org/wiki/Varyans

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı, soyadı : Eyüp YALÇIN

Uyruğu : T.C.

Doğum tarihi ve yeri : 27.01.1986 Kahramanmaraş / MERKEZ

Medeni hali : Bekar

Telefon : (+90) 532 468 77 32

e-posta : eyupyalcinn@gmail.com.

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	KSÜ/ Elektrik-Elektrik Mühendisliği	2009
Lise	Fatih Yabancı Dil Ağırlıklı Lisesi	2004

Yabancı Dil

İngilizce

Hobiler

Sudoku, Go, Origami, Futbol.