T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KUADRATİK GÖRÜNTÜ FİLTRELERİNİN HIZLANDIRILMIŞ EĞİTİMİ İÇİN GPU TABANLI YENİ BİR ALGORİTMA TASARIMI

DOKTORATEZİ

Süleyman UZUN

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Doç. Dr. Devrim AKGÜN**

Ağustos 2018

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KUADRATİK GÖRÜNTÜ FİLTRELERİNİN HIZLANDIRILMIŞ EĞİTİMİ İÇİN GPU TABANLI YENİ BİR ALGORİTMA TASARIMI

DOKTORA TEZİ

Süleyman UZUN

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ

Bu tez 01 / 08 / 2018 tarihinde aşağıdaki jüri tarafından oybirliği/oyçokluğu ile kabul edilmiştir.

Prof. Dr.
İlyas ÇANKAYA
Jüri Başkanı

Doç. Dr.
Cüneyt BAYILMIŞ
Üye

Doç. Dr.
Devrim AKGÜN
Üye

Dr. Öğr. Üyesi
Gökçen ÇETİNEL
Üye

Dr. Öğr. Üyesi
Barış BORU
Üye

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Süleyman UZUN
01.08.2018

TEŞEKKÜR

Doktora eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Doç. Dr. Devrim AKGÜN'e sonsuz teşekkürlerimi sunarım.

Doktora eğitimim boyunca yanımda olan ve beni destekleyen sevgili eşime, hayatıma anlam katan ve canımdan çok sevdiğim kızlarım Melike Su ve Bilge Ece'ye, maddi ve manevi olarak sürekli desteklerini arkamda hissettiğim annem, babam, ablam ve kardeşime çok teşekkür ederim. Ayrıca eğitimim boyunca küçük - büyük beni destekleyen mesai arkadaşlarıma da teşekkür ederim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

CPU	: Central Process Unit
CUDA®	: Compute Unified Device Architecture
RDU	: Rastgele Değişken Uygunluk
DU	: Değişken Uygunluk
GA	: Genetik Algoritmalar
GRDU	: GPU Rastgele Değişken Uygunluk
GPU	: Graphichs Processing Unit
GSU	: GPU Sabit Uygunluk
ID	: Identification
MAE	: Mean Absolute Error
MSE	: Mean Squared Error
NVCC	: NVidia’s Cuda Complier
PSNR	: Peak Signal to Noise Ration
PSO	: Parçacık Sürü Optimizasyon Algoritmaları
PTX	: Paralel İş Parçacığı Çalıştırma
SSIM	: Structural Similarity Index Measurement
SU	: Sabit Uygunluk
UCID	: Uncompressed Image Dataset

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Görüntü filtresi, GPU, Kuadratik görüntü filtresi, sezgisel algoritmalar, genetik algoritmalar, parçacık sürü optimizasyonu

Kuadratik görüntü filtreleri, doğrusal olmayan gürültü karakteristiklerini ele almada genellikle doğrusal filtrelere göre daha başarılıdır. Ancak, ikinci dereceden filtrelerin başarısı için uygun ağırlıkların belirlenmesi, doğrusal filtrelere ağırlıklarının belirlenmesi gibi kolay değildir. Bununla birlikte referans ve gürültülü görüntülerin bulunduğu durumlarda filtre ağırlıkları optimizasyon yöntemleri ile belirlenebilir. Sezgisel algoritmalar kullanılarak ağırlıkların belirlenmesi için, referans görüntü ile filtrelenmiş görüntüye dayanan bir uygunluk değeri (Fitness Value) hesaplanması gerekir. Hesaplanan bu uygunluk değerine göre maske ağırlıkları güncellenmektedir. Bu durum durdurma kriterine kadar yinelemeli olarak tekrarlanır ve maske ağırlıklarının uygun değerleri hesaplanır. Bu süreçler çarpma işlemleri yoğun olan Kuadratik filtrelerde çok zaman almaktadır. Bu tez çalışması kapsamında Kuadratik görüntü filtrelerinin maske eğitimlerini hızlandırmak için algoritmik hızlandırma, donanımsal hızlandırma ve bunların birlikte kullanıldığı yöntemler geliştirilmiştir. Kuadratik görüntü filtrelerin maske ağırlıklarının belirlenmesi için literatürde sıklıkla başvurulan sezgisel algoritmalardan, GA (Genetik Algoritmalar) ve PSO (Parçacık Sürü Optimizasyon) Algoritmaları kullanılmıştır. Deneysel çalışmalarda, referans görüntü üzerine Gaussian gürültüsü eklenerek gürültülü test görüntüleri elde edilmiştir. Uygunluk değeri, referans görüntü ve filtrelenmiş görüntü kullanılarak belirlenen hata fonksiyonundan hesaplanmıştır. Yapılan deneysel çalışmalardan elde edilen sonuçlara göre hem algoritmik hem de donanımsal hızlandırma için önerilen yöntemlerin eğitim sürelerini kısaltmada başarılı olduğu görülmüştür. Elde edilen karşılaştırmalı sayısal sonuçlara göre, GPU hızlandırma ile birlikte kullanılan algoritmik hızlandırmada eğitim süreleri ortalama 300 kat daha fazla kısalmıştır. Ayrıca görüntü kaliteleri de sıralı gerçekleştirme ile elde edilen görüntü kalitelerine benzer oranda elde edilmiştir.

A NEW GPU-BASED ALGORITHM DESIGN FOR ACCELERATED TRAINING OF QUADRATIC IMAGE FILTERS

SUMMARY

Keywords: image filter, GPU, Quadratic image filter, heuristic algorithms, genetic algorithms, particle swarm optimization

Quadratic image filters are usually more successful than linear filters in handling nonlinear noise characteristics. However, determining the appropriate weights for the success of the second order filters is not as straightforward as determining the weights of the linear filters. On the other hand, when there are reference and noisy images, the filter weights can be determined by optimization methods. To determine the weights using heuristic algorithms, a fitness value based on the reference image and the filtered image must be calculated. The mask weights are updated according to this calculated fitness value. This is iteratively repeated until the stopping criterion and the appropriate values of the mask weights are calculated. These processes take a lot of time in Quadratic filters, which are intensive processes of multiplication. Within the scope of this thesis, algorithmic acceleration, hardware acceleration and methods used together to develop mask training of quadratic image filters have been developed. GA (Genetic Algorithms) and PSO (Particle Swarm Optimization) Algorithms have been used for the heuristic algorithms frequently used in the literature for determining mask weights of quadratic image filters. In experimental studies, noisy test images were obtained by adding a Gaussian noise to the reference image. The fitness value is calculated from the error function determined using the reference image and the filtered image. According to the results obtained from the experimental studies, it is seen that the methods proposed for both algorithmic and hardware acceleration are successful in shortening the training periods. According to the comparative numerical results obtained, the training times were shortened by an average of 300 times in the algorithmic acceleration used with GPU acceleration. In addition, image qualities are similar to those obtained by sequential realization.

BÖLÜM 1. GİRİŞ

Görüntü işleme günümüzde popüler çalışma alanlarından biri haline gelmiştir. Çeşitli mühendislik alanlarında ve bilimsel araştırmalarda görüntü işleme teknikleri yoğun bir şekilde kullanılmaktadır. Görüntü işleme teknikleriyle, görüntülerin anlamlandırılması, yorumlanması, görüntü üzerinde nesne bulma vb. gibi çok daha çeşitli amaçlar için kullanılmaktadır. Görüntü iyileştirme, gürültü yok etme, kenar belirleme, özellik çıkarım gibi birçok işlem görüntü filtreleme teknikleriyle gerçekleştirilebilir [1], [2], [11]–[13], [3]–[10]. Temelde görüntü filtreleme yöntemi, bir görüntü filtre maskesinin giriş görüntüsü üzerinde gezdirilmesi yaklaşımına dayanmaktadır. Bu gezdirme işlemleri sırasında, giriş görüntü piksel değerlerinin, görüntü filtresi değerlerini kullanarak matematiksel denklemlerle yeniden hesaplanmaktadır. Görüntünün filtrelenmesi esnasında görüntü maske ağırlıklarının sonuca doğrudan etkisi olduğundan bu değerlerin iyi şekilde belirlenmesi çok önem arz etmektedir. Görüntü maske katsayılarının belirlenmesi için analitik yöntemler kullanılabilir. Alternatif olarak eğitim görüntüleri ile maske katsayıları eğitilebilir. Bu amaçla GA, PSO gibi sezgisel (Heuristic) yöntemler kullanılabilir. Bu yöntemler hesaplama bakımında oldukça ağır yöntemler oldukları için geleneksel CPU (Central Process Unit) üzerinde gerçekleştirildiklerinde çalışma süreleri oldukça uzun ve performansları ise düşük olabilmektedir.

Bu tez çalışmasının amacı, Kuadratik görüntü filtrelerinin GA ve PSO gibi sezgisel algoritmalar ile eğitim sürelerinin kısaltılması amacıyla algoritmik, donanımsal ve hem algoritmik hem de donanımsal olarak yeni algoritmaların gerçekleştirilmesidir. Donanımsal hızlandırma için GPU (Graphichs Processing Unit)'ların hesaplama gücünden faydalanılarak NVIDIA'nın geliştirdiği CUDA® (Compute Unified Device Architecture) hesaplama platformu kullanılacaktır. Çalışmada doğrusal ve Kuadratik filtreler gibi yaygın kullanılan görüntü maskelerinin katsayılarının belirlenmesi için

GA ve PSO tabanlı eğitim kullanılacaktır. Bu amaçla CUDA mimarisini temel alan yeni bir hızlandırılmış algoritma tasarımı gerçekleştirilecektir.

Bu tez çalışmasının sonraki bölümleri şu şekilde düzenlenmiştir;

Bölüm 2'de tez çalışmasıyla ilgili kaynak araştırması verilmiştir.

Bölüm 3'de doğrusal görüntü filtreleri, Gaussian görüntü filtreleri, Kuadratik filtreler, çalışmalarda kullanılan sezgisel optimizasyon algoritmaları, deneylerde kullanılan GPU kart ve CUDA mimarisi hakkında detaylı bilgiler verilmiştir.

Bölüm 4'de Kuadratik görüntü filtrelerin görüntü kalitelerinden ödün vermeden eğitim sürelerini kısaltmak için tasarlanan algoritmik hızlandırma ve donanımsal hızlandırma yöntemleri hakkında detaylı bilgiler verilmiştir. Ayrıca tez çalışması boyunca kullanılan görüntü kalitesi ölçüm teknikleri de detaylı bir şekilde anlatılmıştır.

Bölüm 5'de önerilen tüm yöntemler için yapılan deneysel çalışmalar ve sonuçları sunulmuştur.

Bölüm 6'da Kuadratik görüntü filtrelerin maske ağırlıklarının eğitimleri ve bu eğitim sürelerinin düşürülmesi için geliştirilen yöntemlerin sonuçları, genel değerlendirmeleri ve gelecekte yapılacak olan çalışma önerileri anlatılmıştır.

BÖLÜM 2. KAYNAK ARAŞTIRMASI

Bu bölümde Kuadratik görüntü filtreleri ve GPU tabanlı paralel programlama ile ilgili yapılmış bazı kaynak araştırması çalışmalarına yer verilmiştir. Kuadratik görüntü filtreleri içerisinde hem doğrusal hemde doğrusal olamayan filtreleri barındırdığından dolayı görüntü iyileştirme çalışmalarında da oldukça çok tercih edilmektedir. Fakat görüntü iyileştirme çalışmalarının performansını arttırmak için araştırmacılar farklı yöntemler kullanmaktadır. Bu amaçla, Bernstein ve arkadaşları görüntü iyileştirme için eşleme temelli doğrusal olmayan ikinci derece Volterra filtrelerini sunmaktadırlar [14]. Temelde giriş sinyalleri ikinci derece Volterra filtrelerine uygulanmadan önce normalizasyon işlemlerine ve haritalama işlemlerine tabi tutulmaktadır. Haritalama işlemlerinin olduğu blokta teager filtresi kullanılmaktadır. Normalizasyon işlemlerini ise teager filtresinin giriş ve çıkış aralıklarının aynı olması için kullanılmaktadır. İkinci derece Volterra filtrelerinden sonra ise çıkış haritalama işlemleri ve denormalizasyon işlemleri yapılmaktadır. Yapılan çalışmada görüntünün koyu kısımlarında başarı oranları kullanılan bir fonksiyondan dolayı sınırlanmaktadır. Genelde araştırmacılar Kuadratik görüntü filtrelerinin maske ağırlıklarını belirleme, maske ağırlıkları sayısını indirgeme vb. çalışmalara yoğunlaşmışlardır.

2.1. Kuadratik Görüntü Filtrelerinin Maske Ağırlıkları Üzerinde Yapılan Çalışmalar

Kuadratik görüntü filtreleri doğrusal olmayan filtreler sınıfının bir üyesidir. Bu tip filtrelerin yapısında maske ağırlıkları sayısı çok fazla olduğundan bu maske ağırlıklarının belirlenmesi oldukça zor olmaktadır. Bu sebeple, araştırmacılar maske ağırlıklarını belirlemek için farklı yöntem arayışları içerisine girmişlerdir. Bu amaçla, Ramponi [15] kenar yönünden bağımsız olan filtre katsayılarını simetri koşullarını kullanarak belirlemektedir. Bu şekilde filtre maske ağırlık sayısı 81 olan Volterra

görüntü filtresinin eleman sayısını 11'e indirerek Volterra filtre aileleri için yeni bir optimizasyon yaklaşımı ortaya koymuştur. Çalışmasında referans görüntü üzerine Gaussian ve Gaussian darbe gürültüsü ekleyerek gürültülü referans görüntüleri elde ederek kullanmıştır. Sonuçta elde ettiği filtreyi kenar belirleme işlemlerinde test etmiş ve doğrusal filtrelere göre daha fazla performans elde etmiştir. Volterra filtrelerinin filtre yapısını daha etkin kullanmak ve simetrik şartlarını daha da basitleştirmek amacıyla, Ramponi ve Sicuranza [16] matris tanımlamaları yapmışlardır. Tanımladıkları matrislerle daha az filtre katsayılarını hesaplayarak görüntü iyileştirme ve gürültülü resmi temizlemeyi başarmışlardır. Benzer şekilde bu matris gösterimini Volterra serisi temelli doğrusal olmaya 2-D filtreler için Ramponi ve arkadaşları sunmuştur [17]. Matrisin tasarımını sadeliği ve verimliliğinden dolayı Powell'ın doğrudan arama yöntemine dayalı optimizasyon yaklaşımı ile yapmışlardır. Böyle bir algoritma doğrudan arama tipindedir ve objektif fonksiyonun türevlerini kullanmaz. Giriş görüntüsü olarak önceden belirlenmiş statik özelliklere sahip sabit bir gürültü grubu setine sentetik bir dikey kenra eklenerek elde edilmiş olan görüntüler kullanılmıştır. Uygunluk fonksiyonunu, filtrelenmiş görüntü ile referans görüntünün arasındaki farkın karesini alarak elde edilmiştir. Sonuç olarak gürültüyü yumşatırken kenarları koruyan bir filtre ortaya çıkmıştır. Kuadratik görüntü filtrelerin maske ağırlıklarının azaltılmasına yönelik olarak Zhou ve arkadaşları [18] alfa ağırlıklı Kuadratik filtre adında yeni, güçlü ve doğrusal olmayan bir filtre tanıtmışlardır. Alfa ağırlıklı Kuadratik filtreleri giriş görüntüsünü, görüntü nesnelerinin ya da özelliklerinin yönlendirilmesinden bağımsız hale getirmek için izotropik görüntü operatörü olarak tasarlamışlardır. Böylece filtrenin maske ağırlıklarının sayısı azaltılabilir. Alfa ağırlıklı Kuadratik filtrelerin giriş görüntüsünün iki pikseli arasındaki mesafeye göre 3 tipe ayırmışlardır;

1. *Tip sıfır alfa ağırlıklı Kuadratik filtre:* Bu tip filtre ikinci dereceden terimlerin tamamından meydana gelir ve iki piksel arasındaki mesafenin sıfır olduğunu kabul eder. Yani iki piksel aynı konumdadır.
2. *Tip bir alfa ağırlıklı Kuadratik filtre:* Bu tip filtre ikinci dereceden terimlerin tamamını içerir ve iki piksel arasındaki mesafenin bir olduğunu kabul eder. Yani iki piksel bir birine bitişiktir.

3. *Tip iki alfa ağırlıklı Kuadratik filtre:* Bu tip filtre ikinci dereceden terimlerinden oluşur ve iki piksel arasındaki mesafenin iki olduğunu kabul eder.

Önerilen algoritma aşağıdaki kurallara göre oluşturulmuştur;

1. Maske boyutu 3×3 ise, giriş görüntüsü doğrudan alfa ağırlıklı Kuadratik filtre tarafından filtrelenir,
2. Maske boyutu 5×5 ise, yeni bir 3×3 filtre maskesi oluşturulur ve 5×5 filtre maskesinden 9 adet görüntü pikseli seçilir ve ardından alfa ağırlıklı Kuadratik filtreye uygulanır.
3. Maske boyutu 5×5 filtre maskesinden büyükse, filtre maskesinin dört köşesinde alfa ağırlıklı Kuadratik filtreyi 3×3 alt filtre maskesine uygulanır. Alfa ağırlıklı Kuadratik filtrenin son çıkışı bu dört alt filtre maskesinden elde edilen sonuçların ortalama değeridir.

Bu şekilde önerilen yeni algoritma ile kullanıcıların sadece 3×3 boyutlarında bir filtre maskesine indirgenerek alfa ağırlıklı Kuadratik filtrelerin katsayıları azaltmıştır. Elde edilen yeni algoritmayı mamogram görüntüleri üzerinde uygulayarak genel kontrastı etkili bir şekilde arttırmış ve mamogramlardaki yerel ince ayrıntıları ve karanlık bölgeleri iyileştirdiği gözlemlenmiştir. Kanamadi ve arkadaşları da doğrusal olmayan alfa ağırlıklı Kuadratik filtre çalışması sunmaktadırlar [19]. Zhou ve arkadaşlarının yaptığı çalışmadan farklı olarak filtre görüntü detaylarını korurken yüksek bozuk gri skala görüntülerindeki gürültüyü temizleyen maskeleme tekniği kullanılmışlardır. Kuadratik görüntü filtrelerin maske ağırlıklarının sayısını azaltmak için alfa ağırlıklı Kuadratik filtreleri tercih etmişlerdir. Çalışmalarının amacı, mamogram görüntülerinin görsel kalitesini arttırarak yüksek çözünürlüklü ve yüksek kontrastlı görüntüler elde etmektir. LodAMEE görüntü iyileştirme derecesini ölçmek için kullanılmışlardır. Çalışma sonunda alfa ağırlıklı Kuadratik filtrelerin tiplerini (tip sıfır, tip bir ve tip iki) birbirleriyle karşılaştırmışlardır. Önerilen yöntemin mamogram görüntülerinin genel kontrastını arttırırken gürültüleri temizlediği ve yerel ince ayrıntılar ile karanlık bölgeleri geliştirdiği sonucuna ulaşılmıştır. Sicuranza ise yaptığı çalışmada [20], Volterra serilerinin doğrusal olmayan kısmı için Kronecker üretimine dayanan matris gösterimi üretmiştir. Matris gösterimi doğrusal sayısal filtreler için önerilen dağıtılmış aritmetiğin iyi bilindiğini doğrusal olmayan operatörlerin gerçeklemesi için nasıl kabul

edildiğini göstermektedir. Kombinasyonların karakteristikleri ve yönlü hafıza yapıları karşılaştırılarak karmaşık azalmanın sorununu tartışmıştır.

2.2. Kuadratik Görüntü Filtrelerinin Diğer Filtrelerle Karşılaştırlıması

Kuadratik görüntü filtrelerinin doğrusal olmayan gürültüleri temizlemede doğrusal filtrelere göre oldukça başarı sağlamaktadır. Doğrusal filtrelerde filtreleme işlemleri sonucunda görüntü üzerindeki detaylar kaybolmakta ve görüntü bulanıklaşmaktadır. Kuadratik görüntü filtrelerinin yapısında bulunan doğrusal kısmı alçak geçiren filtre, Kuadratik kısmı ise yüksek geçiren filtrelere gibi çalıştığı için görüntü filtreleme işlemleri sonucunda görüntü detayları korunurken herhangi bir bulanıklaşma olmamaktadır [21]. Bu konu ile ilgili araştırmacılar tarafından yapılan bir çok çalışma bulunmaktadır. Bu çalışmalardan birini, Johilakshmi ve arkadaşları yapmışlardır [22]. Yaptıkları çalışmada, Gaussian, zehir ve beyaz gürültülerden etkilenen mamogram görüntülerinin kontrast iyileştirilmesi için uyarlanabilir Volterra filtresini kullanmışlardır. Mamogram görüntüler, X-ışını donanım sistemlerinden dolayı düşük kontrast ve kötü radyografik çözünürlüğe sahiptirler. Bu durum lezyon detaylarının yanlış görülmesine yol açmaktadır. Bundan dolayı doğrusal olmayan bu gürültüleri temizlerken arka plan gürültüsünü bastıracak ve aynı zamanda kenarları koruyarak daha iyi filtreleme sonucu elde etmek için doğrusal olmayan Volterra filtrelerini tercih etmişlerdir. Volterra filtresinin performansını uzaysal düzlemde alçak, medyan, minimum ve maksimum gibi doğrusal olmayan filtreler ile karşılaştırmışlardır. Çalışmalarını daha da geliştirmek için gürültülü mamogram görüntüyü frekans düzleminde de Volterra, medyan, minimum, maksimum ve ortalama olmak üzere beş farklı filtre kullanılarak filtrelemişlerdir. Performans ölçümlerinde MSE (Mean Square Error) ve PSNR (Peak Signal to Noise Ration) kullanılmıştır. Volterra filtresinin PSNR değeri uzaysal düzlemde diğer filtrelerden yüksek olduğu görülmüştür. Frekans düzleminde ise Volterra filtresinin çıkışı diğer filtre çıkışlarından daha iyi olduğu gözlemlenmiştir. Volterra filtresinin PSNR ve MSE değerleri diğer filtrelerden çok daha yüksek olduğu sonucuna ulaşmışlardır. Meenavathi ve Rajesh ise Gaussian ve Gaussian darbe gürültülü görüntülerin iyileştirilmesi ve onarımı için Volterra serisi temelli filtre kullanmışlardır [21]. Volterra filtrelerinin maske ağırlıklarını FIR

algoritmasını kullanarak belirlemişlerdir. Volterra filtrelerinin maske ağırlıklarını azaltmak için leksikografik (lexicographic) matris tekniği ve simetrik koşulları kullanarak azaltmışlardır. Önerdikleri çalışma aşağıdaki aşamalardan oluşmaktadır;

a. *1. Adım:* Verilen görüntü için uygun kesme frekansını seçmek için ortalam yoğunluk bulunur,
b. *2. Adım:* Verilen görüntü üzerine Gaussian ve Karmaşık Gaussian darbe gürültüsü eklenerek gürültülü görüntü elde edilir,
c. *3. Adım:* Uygun pencere yöntemiyle FIR algoritması kullanılarak doğrusal ve doğrusal olmayan filtre katsayıları hesaplanır,
d. *4. Adım:* Blok leksikografik matrisi ve simetrik koşulları kullanılarak filtre maskesi katsayılarının sayısı azaltılır,
e. *5. Adım:* Görüntü filtrelenir,
f. *6. Adım:* Filtrenin performansı farklı parametreler kullanılarak test edilir.

Yapılan testler sonucunda görüntü iyileştirme ve onarmada kullanılan Volterra filtresinin alçak geçiren filtre, yüksek geçiren filtre ve medyan filtre gibi standart doğrusal ve doğrusal olmayan filtrelere göre çok daha başarılı olduğunu tespit etmişlerdir. Mitra ise yaptığı çalışmada [23], kontrast iyileştirme, darbe gürültüsünü yok etme ve görüntü yakınlaştırma uygulamalarında Kuadratik görüntü filtrelerinin başarımından bahsetmektedir. Yaptığı çalışma sonucunda elde ettiği sonuçlar aşağıda belirtildiği gibidir;

a. Doğrusal olmayan gürültülerin görüntüden temizlenmesinde doğrusal filtrelerin başarımı düşük olmaktadır,
b. Doğrusal filtreler bu gürültüleri temizlerken görüntü kenarlarını bulanıklaştırır ve görüntü detaylarını kaybetmektedir,
c. Bu gibi gürültüleri temizlemek için doğrusal olamayan filtreler kullanılması gerektiğini bildirmektedir,

Hari ve arkadaşaları ise yaptıkları çalışmada [24], mamogramlardaki kalifikasyonların iyileştirilmesi için Volterra serileri temelli Kuadratik kenar belirleme filtre tasarımı ve uygulaması yapmışlardır. Bu filtre ile meme kanserine yol açan mikro kalifikasyonların erken belirlenmesi hedeflenmektedir. Görüntü kalitesi ölçüm tekniklerinden SNR (Signal to Noise Ratio) ve PSNR kullanmışlardır. Kuadratik

görüntü filtrelerini Laplasyan, Gabor, Canny, LoG (Laplasyan of Gaussian) ve Prewitt filtreleri gibi filtrelerle karşılaştırmışlardır. Görüntü filtreleme sonucunda; Gabor filtresi ile kalifikasyonları iyileştirilmiştir fakat kalifikasyonların kenarların belirsiz olduğu gözlemlemiştir. LoG filtresinin arka plan dokusunu gizleyemediği görülmüştür. Prewitt filtresi ise görüntüyü hiç geliştiremediği sonucuna ulaşılmıştır. Canny kenar dedektöründe kalifikasyonlar farkedilememektedir. Kuadratik filtresi ile arka plan dokusunun bastırıldığı ve kalifikasyonlar ile kalifikasyonların kenarları açıkca görülmektedir. Thomas ve arkadaşları da yaptıkları çalışmada [25], çok gürültülü koşullarda elde edilen MRI görüntülerindeki dürtüsel gürültünün temizlenmesi için Kuadratik filtreler üzerinde durmuşlardır. Kuadratik filtreleri, medyan, ortalama ve Gaussian filtreler ile karşılaştırmışlardır. Kuadratik filtrelerin dürtüsel gürültünün temizlenmesinde daha üstün olduğunu görmüşlerdir. Pandaya ve arkadaşları yaptıkları çalışmada [26], Volterra filtreleri kullanılarak sayısal mamogramların iyileştirilmesinde kullanmışlardır. Volterra filtrenin mamogram lezyonlarını iyileştirmede en iyi başarımı sağladığını görmüşlerdir.

2.3. GPU Tabanlı Hızlandırma

Kuadratik görüntü filtreleri içerisindeki ikinci dereceden çarpım işlemlerinin işlemlerin yoğunluğundan dolayı doğrusal filtrelere göre oldukça yüksek işlem gücü gerektirir. Buna benzer yüksek hesaplama gücü gerektiren problemleri çözmek için araştırmacılar CPU, GPU ve FPGA gibi donanımları kullanarak çok farklı yöntemler denemektedirler. Bu amaçla, Asano ve arkadaşları yaptıkları çalışmada [27], FPGA, GPU ve CPU kullanarak iki boyutlu filtre, üç boyutlu görme (stereo-vision) ve K-ortalamalar kümesi uygulamaları gerçekleştirmişler ve performans karşılaştırmalarını yapmışlardır. Çalışma sonucunda iki boyutlu filtre uygulamasında en iyi performansı GPU verirken küçük filtrelerde CPU'nun FPGA'ya göre daha iyi sonuç verdiği gözlemlenmiştir. Fakat filtre boyutları arttıkça CPU'nun performansı düşerken FPGA performansının sabit olduğu gözlemlenmiştir. Üç boyutlu görme uygulamasında FPGA'nın diğerlerine göre çok daha iyi performans verdiği gözlemlenmiştir. CPU performansının GPU performansından daha iyi olduğu sonucu ortaya çıkmıştır. K-ortalamalar kümesi uygulamasında ise en iyi sonucun FPGA tarafından verildiği

gözlemlenmiştir. 4 iş parçacıklı (Thread) CPU'nun ise GPU'dan çok daha iyi performans sağladığı gözlemlenmiştir. Fialka ve Cadik [28], GPU üzerinde konvolüsyon filtre ve hızlı Fourier dönüşüm uygulamalarını gerçekleştirmişlerdir. GPU uygulamalarında bu metotları gerçek zamanlı uygulamalar ve kullanımını performans açısından değerlendirmektedirler. Değerlendirme sonucunda, GPU üzerinde basit ve küçük filtreler uygulandığında konvolüsyon işleminin hızlı fourier dönüşüm işleminden çok iyi performans sağladığı gözlemlenmiştir. Shi ve arkadaşları [29], Hesaplanmış Tomografi (Computed tomography-CT) görüntülerinin yeniden oluşturulmasında CUDA program modelini kullanarak hızlandırmayı amaçlamışlardır. Aynı zamanda konvolüsyon filtresinin de hızlandırılmasında CUDA temelli model kullanılmıştır. Çalışma kapsamında yapılan deneysel sonuçlara göre çekirdekte çalıştırılan algoritmanın 80 kat daha hızlı ve 11 kat daha performanslı olduğu gözlemlenmiştir. Keçeli ve Can [30], Nvidia CUDA platformunda K-means kümeleme algoritmalarını kullanarak otomatik ve gözetimsiz beyin bölütleme yöntemi sunmaktadırlar. Geliştirdikleri yöntemi CPU ve GPU üzerinde gerçekleştirerek karşılaştırmaları yapılmıştır. Çalışma sonucunda GPU üzerinde gerçekleştirilen yöntem CPU'ya göre 25-30 kata kadar hızlandırma sağladıkları gözlemlenmiştir. Ono ve arkadaşları [31], görüntü ve video işleme için daha çok sezgisel programlama dili ve bu dil için çevirici (translator) önermektedirler. Bu çeviriciler tasarlanırken yüksek seviyeli video işleme kütüphanesi olan RaVioli kullanılmıştır. Tasarlanan bu çevirici kullanılarak programcılar, CUDA API'leri ve GPU mimarilerini her ikisini hakkında bilgi sahibi olmaksızın GPU kullanarak performans kazançları elde etmişleridir. Badem [32], kamera yardımıyla alınan biber görüntülerinin tanımlanarak yön tespitlerini yapmıştır. Bu amaç doğrultusunda sezgisel bulanık mantık, kenar çıkartım, otsu algoritması ve çok katmanlı algılayıcı yapay sinir ağı modeli kullanmıştır. Geliştirilen uygulama hem CPU hem de GPU üzerinde gerçekleştirilmiştir. GPU ile gerçekleştirilen uygulamanın CPU'ya göre otsu algoritmasında 3 kat, sezgisel bulanık mantık algoritmasında 442 kat ve sistemin toplam cevap verme süresinde 79 kat hızlanma elde edilmiştir. Iandola ve arkadaşları [33], 2D görüntü konvolüsyon işlemini hızlandırmak için GPU üzerindeki bellek iletişimini azaltarak görüntü bölgesini ön belleğe kaydeden yeni bir algoritma tasarlamıştır. Bu tasarım ile daha az iş parçacığı kullanılmış ve iş parçacığı başına düşen iş miktarı arttırılmıştır. Yapılan

çalışmada küçük filtreler kullanılmış ve geliştirilen algoritmayla o tarihteki son versiyon GPU Kütüphanesi üzerinde 1,2 ile 4,5 arasında hızlanma elde edilmiştir. Chen ve arkadaşları [34], uzaysal (spatial) düzleminde 2D (2- boyutlu) resimlerin konvolüsyon filtre uygulamaları yapılmıştır. Filtreleme işlemlerini hem CPU hem de GPU üzerinde gerçekleştirmişlerdir. Filtreleme işlemlerinin GPU üzerinde CPU'ya göre 10 kat daha fazla hız elde etmişlerdir. Çekmez [35], GA ve Karınca Kolonisi optimizasyon tekniklerini grafik kart üzerinde paralel çalışacak şekilde kodlayarak insansız hava aracının rota planlamasını gerçekleştirmiştir. Çalışmalarını hem GPU hem de CPU üzerinde test ederek bir kıyaslama yapmıştır. GPU üzerinde üretilen çözümün CPU'ya göre çok daha kısa sürede olduğu gözlemlenmiştir. Koçak [36], gerçek zamanlı insan sayma uygulaması gerçekleştirmiştir. Bu uygulama hem Matlab® ortamında hem de CUDA mimarisi kullanılarak GPU üzerinde gerçekleştirilmiş ve her iki ortamın problem çözümündeki performansları karşılaştırılmıştır. GPU üzerinde gerçekleştirilen uygulamanın önemli derecede hızlı olduğu gözlemlenmiştir. Fakat Matlab ortamında gerçekleştirilen uygulamayı test etmenin GPU ortamına göre çok daha kolay olduğu sonucuna varılmıştır. Koca [37], gölge eşleme algoritması ile gerçek zamanlı görüntü elde etme yöntemini incelemiş ve gölge algoritmalarını gerçekleştirmiştir. Gerçekleştirilen uygulamaların hem CPU hem de GPU üzerinde ayrı ayrı performans analizleri yapılmıştır. Sonuçta GPU da gerçekleştirilen uygulama CPU'da gerçekleştirilene göre %45,3 oranında performans sağladığı gözlemlenmiştir. Rahmani [38], entropi kodlama algoritması Huffman kodlayıcının GPU üzerinde paralel uygulamasını gerçekleştirmiştir. Aynı uygulamayı CPU üzerinde de test ederek performans karşılaştırması yapmıştır. GPU üzerinde gerçekleştirilen uygulama CPU üzerinde gerçekleştirilene göre 22 kat daha hızlandığı gözlemlenmiştir. Akgün ve Erdoğmuş [39], GA kullanarak maskeleme ağırlıklarının eğitimini hızlandırmak için GPU temelli alt görüntü bloklarını çağıran yeni bir algoritma geliştirmişlerdir. Yöntem, doğrudan yöntem, popülasyon temelli yöntem, blok temelli yöntem ve alt görüntü temelli yöntem olan diğer alternatif tasarımlar dikkate alınarak geliştirilmiştir. Bu tasarımlar arasında alt görüntü temelli yöntem en iyi performansı vermiştir. Çalışma süresi ve hızlandırmalı ivme grafiklerine göre uygulanan yöntem 3.5GHz işlemciye göre GeForce™ GTX600 üzerinde 55-90 kat daha fazla hızlanma sağladığı görülmüştür. Patel yaptığı çalışmada [40], kutupsal

imgeleme işlemlerini gerçekleştirmek için Focal Plane Array (FPA) içerisindeki her bir pikselin stokes parametrelerini ve doğrusal polarizasyon derecesini (DoLP) Matlab üzerinde hesaplamıştır. Bu hesaplamada çerçeve başına 30 saniye düşmektedir. GPU kullanarak bu süreyi 50 mili saniyelere kadar düşürerek 600 kat kadar hızlanma elde etmiştir.

Yukarıda özetlenen çalışmalarda Kuadratik filtre birçok alanda görüntü iyileştirme ve gürültü temizleme gibi uygulamalarda kullanılmıştır. Ancak kuadratik görüntü filtrelerinin yapısıdan dolayı maske ağırlıklarının belirlenmesi doğrusal görüntü filtrelerindeki kadar açık olmadığı için doğrusal filtreler kadar tercih edilmemektedir. Bundan dolayı Kuadratik görüntü filtrelerinin katsayılarının belirlenmesi için pratik bir yaklaşımın geliştirilmesi bu noktadaki açığın giderilmesi açısından literatüre katkıda bulunacaktır. Bu maske ağırlıklarının belirlenmesi için sezgisel algoritmalar kullanılmaktadır. Buradaki en büyük sorun, bu maske ağırlıklarının belirlenmesi süresinin oldukça uzun sürmesidir. Örneğin 3×3 maske boyutu için Kuadratik görüntü filtrelerinin bünyesinde maske ağırlıkları olarak doğrusal kısımdan 9 doğrusal olmayan kısımdan 45 olmak üzere toplamda 54 adet maske ağırlığına sahiptir. Bu amaçla GPU tabanlı hesaplamanın kullanılması yukarıda bahsedilen literatür çalışmalarında olduğu gibi görüntü filtreleme işlemini hızlandırmada kullanılabilir. Ayrıca, sezgisel optimizasyon algoritmalarının popülasyon tabanlı çalışma yapısı uygunluk fonksiyonunu yoğun bir şekilde kullanımını gerektirmesi, görüntü filtreleme işleminin hızlandırılmasının gerekliliğini artırmaktadır.

Bu tez çalışmasında, Kuadratik görüntü filtrelerinin hızlandırılmış eğitimi için hem donanımsal hem de algoritmik hızlandırma sağlayacak yeni bir yöntem geliştirilmiştir. Bu amaçla literatüre yapılan katkılar;

1. Kuadratik görüntü filtrelerin maske ağırlıklarının farklı optimizasyon algoritmalarla eğitiminin sağlanması,
2. Tasarlanan algoritmaların burada kullanılmayan diğer sezgisel algoritmalarına kolayca uyarlanabilmesi,
3. Kuadratik görüntü filtrelerinin algoritmik hızlandırılması,
4. Kuadratik görüntü filtrelerinin donanım tabanlı hızlandırılması,

5. Kuadratik görüntü filtrelerinin hem donanımsal hem de algoritmik hızlandırılmasının gerçekleştirilmesi,
6. Kuadratik görüntü filtreleri gibi zor problemlerin başarılı bir şekilde çok kısa sürelerde çözülmesinin diğer zor problemlerin çözümlerine ışık tutması,
7. Kullanımı kısıtlı olan başarılı bir filtrenin bu sayede kullanım oranlarının arttırılması.

şeklinde özetlenebilir.

BÖLÜM 3. MATERYAL VE YÖNTEM

3.1. Giriş

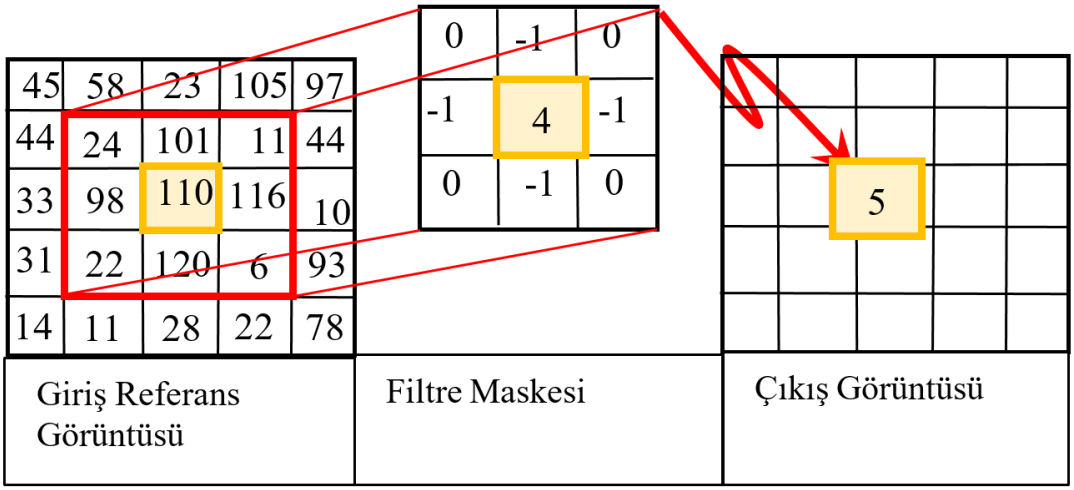
Bu bölümde, Kuadratik görüntü filtreleri, bu filtrelerin maske ağırlıklarını eğitmek için kullanılan optimizasyon algoritmaları, eğitim sürelerini kısaltmak için geliştirilen donanımsal hızlandırma yöntemlerinin çalıştırılacağı GPU ve CUDA mimarileri hakkında detaylı bilgiler sunulmaktadır. Donanımsal hızlandırmada kullanılacak olan donanım Nvidia'nın Maxwell mimarisi ile geliştirdiği grafik kartı kullanılmaktadır. Bu kart üzerinde çalışacak algoritmalar CUDA paralel hesaplama platformu ve programlama modeli kullanılarak geliştirilmiştir.

3.2. Doğrusal Görüntü Filtreleri

Konvolüsyon ya da korelasyon doğrusal görüntü filtreleme tekniklerinde, filtre maskesi matrisi gürültülü referans görüntü üzerindeki bütün piksellere uygulanıp çıkış piksellerinin hesaplanmasıyla görüntü filtreleme işlemleri gerçekleştirilmiş olur. Bu gezdirme işlemleri yapılırken genelde aşağıda gösterildiği gibi (Denklem 3.1) kullanılır [2], [41];

$$y(i, j) = \sum_{m=-N}^{N} \sum_{n=-N}^{N} w_{m,n} x(i-m, j-n) \tag{3.1}$$

y çıkış görüntüsünü, x, giriş görüntüsünü, i ile j piksel konumunu ve $w_{m,n}$ ise maske ağırlıklarını göstermektedir. Filtre maskesi matrisinin merkezi, giriş referans görüntüsü üzerinde filtrelenecek olan pikselle üst üste gelecek şekilde Şekil 3.1.'de gösterildiği gibi konumlandırılır. Sonrasında filtre maskesi matrisinin karşılığında gelen giriş referans görüntüsündeki piksel değerleri bir birleriyle çarpılır.



Şekil 3.1. Doğrusal filtreleme örneği

Çarpma işlemi sonucunda bütün değerler toplanır ve maske matrisinin merkezinin denk geldiği giriş referans görüntüsü pikselinin değeri bu toplama işlemi sonucu elde edilen değer olmaktadır. Şekil 3.1.'de örnek olarak laplasyan filtre maske matrisi kullanılarak doğrusal filtreleme işlemi gerçekleştirilmektedir. Şekildeki örneğe göre çıkış görüntü piksel değeri aşağıdaki (Denklem 3.2) gibi hesaplanır.

$$\begin{aligned} y &= (24\times 0)+(101\times(-1))+(11\times 0)+(98\times(-1))+(110\times 4) \\ &\quad +(116\times(-1))+(22\times 0)+(120\times(-1))+(6\times 0) \\ y &= -(101+98+116+120)+440 \\ y &= 5 \end{aligned} \tag{3.2}$$

Bu örnekte kenar belirleme filtresi kullanılmıştır. Kenar belirleme filtre maske matrisi giriş görüntüsü üzerindeki her bir piksel üzerinde dolaştırılarak filtreleme işlemi tamamlanır. Bu tez çalışmasında Kuadratik görüntü filtrelerinin doğrusal filtrelerle karşılaştırılmasında Gaussian filtre maskesi kullanılmıştır.

3.2.1. Gaussian Görüntü Filtreleri

Gaussian filtreler genelde gürültü temizleme, yumuşatma ve kenar koruma gibi işlemlerde kullanılan bir filtredir [42]–[44]. Gaussian filtrelerin ortalama değeri sıfır

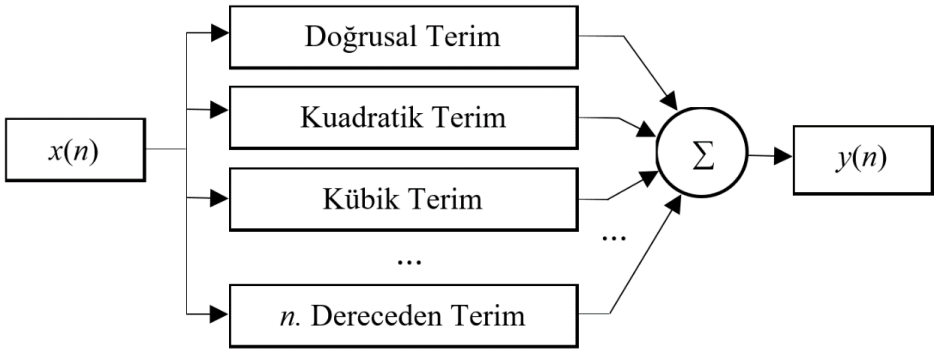
olarak alındığında iki boyutlu görüntüler için genel olarak kullanılan denklem aşağıda (Denklem 3.3) gösterilmiştir [45].

$$G(i,j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{3.3}$$

Yukarıda gösterilen eşitlikte (Denklem 3.3) *i* satır, *j* sütun, ve *σ* ise Gaussian dağılımının standart sapmasını ifade etmektedir. Gaussian filtreler alçak geçiren filtrelerin bir sınıfı olduklarından dolayı gürültü temizleme işlemlerinde yüksek frekans bileşeni içeren kenarları yok etmektedir [46]–[49]. Bu filtre ile görüntü filtreleme işlemleri doğrusal görüntü filtreleri başlığı altında anlatıldığı gibi olmaktadır. Tek fark filtre maske ağırlıkları değişmektedir. Filtre maske ağırlıkları Gauss katsayısı ile farklı değerler almaktadır.

3.3. Kuadratik Görüntü Filtreleri

Doğrusal sistem teorisi oldukça iyi ve kullanışlı olmasına rağmen, uygulamada birçok alan ve sistemler doğrusal değildir [24], [50]. Volterra serileri, polinom filtreler olarak bilinen doğrusal olmayan filtrelerin diğer bir sınıfıdır ve doğrusal olmayan davranışları ele almada daha güçlüdür [51]–[53]. Volterra serileri doğrusal olmayan sistemlerin sentezi, analizi ve gösterimi için çok önemli bir modeldir [19]. Doğrusal olmayan 2D filtrenin sınıfı matris notasyonu ile kesikli ayrık Volterra serilerini temel alır [51]. Volterra serileri, açıldıklarında doğrusal, kuadratik, kübik, vb. isimlendirilen farklı formlara dönüşmektedir. Genel olarak sistem modeli Şekil 3.2.'de gösterilmektedir.



Şekil 3.2. Volterra serilerinin sistem modeli[54]

Volterra fltrelerinin en genel olarak denklemi aşağıdaki eşitlikte (Denklem 3.4) gösterildiği gibi ifade edilebilir;

$$y(n) = y_0 + \sum_{p=1}^{\infty} \hbar_p[x(n)] \tag{3.4}$$

Yukarıda gösterilen eşitlikte (Denklem 3.4) y_0 sabiti temsil etmektedir ve $\hbar_p$ ise filtrenin p'inci dereceden doğrusal olmayan davranışını karakterize eden genelleştirilmiş bir yanıtıdır. $\hbar_p$ aşağıda (Denklem 3.5) verilen eşitlikte gösterildiği gibi açık bir şekilde ifade edilebilir;

$$\hbar_p[x(n)] = \sum_{i_1=0}^{N-1} \dots \sum_{i_p=0}^{N-1} \hbar_p(i_1, i_2, \dots i_p) x(n - i_1) x(n - i_2) \dots x(n - i_p) \tag{3.5}$$

Yukarıda gösterilen eşitlik (Denklem 3.5) ikinci dereceye kadar açıldığında Kuadratik görüntü filtrelerinin denklemi elde edilir. Aşağıdaki eşitliklerde (Denklem 3.7 ve Denklem 3.8) gösterilmiştir. Kuadratik görüntü filtrelerinin denklemini en genel haliyle aşağıdaki eşitlikte (Denklem 3.6) gösterildiği gibi ifade edilebilir;

$$y(m, n) = y_0 + y_1(m, n) + y_2(m, n) \tag{3.6}$$

Buradaki y_0 sabiti temsil etmektedir ve bu çalışmada sıfır alınmıştır, $y_1(m, n)$ Kuadratik görüntü filtresinin doğrusal kısmını, $y_2(m, n)$ kuadratik kısmını temsil etmektedir [55], [56]. $y_1(m, n)$ ve $y_2(m, n)$ aşağıdaki eşitliklerde (Denklem 3.7 ve Denklem 3.8) gösterildiği gibi ayrı ayrı ifade edilebilir [57];

$$y_1(m, n) = \sum_{i=-(N-1)/2}^{(N-1)/2} \sum_{j=-N(N-1)/2}^{(N-1)/2} w_{i,j}^1 \, x_{m+i, n+j} \tag{3.7}$$

$$y_2(m,n) = \sum_{i=-(N-1)/2}^{(N-1)/2} \sum_{j=-N}^{(N-1)/2} \sum_{k=-(N-1)/2}^{(N-1)/2} \sum_{l=-(N-1)/2}^{(N-1)/2} w_{i,j,k,l}^2 \, x_{m+i,\, n+j} x_{k+i,\, l+j} \tag{3.8}$$

Yukarıda gösterilen eşitliklerde (Denklem 3.7 ve Denklem 3.8) bir dizi toplam operatörünü içermektedir. Bunların anlaşılmasını kolaylaştırmak için bu eşitlik (Denklem 3.7) aşağıda (Denklem 3.9) gösterildiği gibi genel olarak ifade edilebilir;

$$y_1(m,n) = \sum_{i=0}^{N\times N} W_i^1 X_{m,n}^1(i) \tag{3.9}$$

Yukarıda gösterilen eşitlik (Denklem 3.9) aşağıda (Denklem 3.10) gösterildiği gibi vektörel notasyonda ifade edilebilir;

$$\left.\begin{aligned} W_1 &= [w_1(-N,-N), w_1(-N,-N+1), \dots, w_1(N,N)] \\ X_{m,n}^1 &= [x(m-N,n-N), \dots, x(m+N,n+N)]^T \end{aligned}\right\} \tag{3.10}$$

Yukarıdaki eşitlikte (Denklem 3.10) vektörel notasyonda ifade edilen denklem genel olarak aşağıdaki eşitlikte (Denklem 3.11) gösterildiği gibi ifade edilebilir;

$$y_1(m,n) = W_1 {X_{m,n}^1}^T \tag{3.11}$$

Kuadratik kısmı iki boyutlu (2-D) biçimde aşağıdaki eşitliklerde (Denklem 3.12 ve Denklem 3.13) gösterildiği gibi ifade edilebilir;

$$y_2(m,n) = \sum_{i=0}^{N\times N} \sum_{j=0}^{N\times N} w_2(i,j) x_1(m+i,n+j) x_1(m+i,n+j) \tag{3.12}$$

$$y_2(m,n) = \sum_{i=0}^{N\times N} \sum_{j=0}^{N\times N} w_{i,j}^2 X_{m,n}^1(i) X_{m,n}^1(j) \tag{3.13}$$

Yapılan çalışmada, Kuadratik filtrelerin maske boyutları pratikte en çok başvurulan 3×3 olarak alınmıştır. Ancak elde edilen sonuçlar 5×5 ve 7×7 gibi daha yüksek boyutlara kolayca genişletilebilir. Kuadratik filtrelerin maske boyutunu 3×3 olacak şekilde alındığında denklemlerde verilen *N* değeri 3 olacaktır. Bu durumda yukarıdaki eşitlik (Denklem 3.10) için verilen ağırlık ve girdi vektörü aşağıdaki eşitlikte (Denklem 3.14) gösterildiği gibi ifade edilebilir;

$$\left.\begin{aligned} W_1 &= [w_0\ w_1\ w_2\ w_3\ w_4\ \ w_5\ w_6\ w_7\ w_8] \\ X_{m,n}^1 &= [x_0\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8]^T \end{aligned}\right\} \tag{3.14}$$

Yukarıdaki eşitlikte (Denklem 3.14) $W_1(-N,-N)$ ile $W_1(N,N)$ ağırlıkları W_0 ile W_8 değiştirilir ve giriş vektörü de aynı şekilde basitleştirilir. Yukarıda gösterilen eşitlikte (Denklem 3.14) giriş sinyalinin çarpımlarının *N*(*N-1*)*/2* adedinin simetrik olduğu görülmektedir. Aşağıdaki eşitlikte (Denklem 3.15) 3×3 durumu için kuadratik giriş terimleri 9×9 büyüklüğündeki bir matris formuna dönüştürülmüştür. Burada matris elemanlarının köşegene göre simetrik olduğu görülmektedir.

$$X_{m,n}^{1}X_{m,n}^{1} = \begin{bmatrix} x_0x_0 & x_0x_1 & \dots & x_0x_8 \\ x_1x_0 & x_1x_1 & \dots & x_1x_8 \\ \vdots & \vdots & \vdots & \vdots \\ x_8x_0 & x_8x_1 & \dots & x_8x_8 \end{bmatrix} \tag{3.15}$$

Yukarıda gösterilen eşitlik (Denklem 3.15) simetrik terimler elenerek yeniden yazılırsa, ikinci toplam operatörü olan *j*'nin indeksinin değeri aşağıdaki eşitlikte (Denklem 3.16) gösterildiği gibi *i*'nin indeks değerinden başlar.

$$y_2(m,n) = \sum_{i=0}^{N\times N}\sum_{j=i}^{N\times N} w_{i,j}^{2}X_{m,n}^{1}(i)X_{m,n}^{1}(j) \tag{3.16}$$

Yukarıdaki eşitliğe (Denklem 3.15) göre, indisler 0'dan 8'e olacak şekilde alındığında ve simetrik terimler sıfır yapılırsa matris aşağıdaki (Denklem 3.17) gibi sadeleştirilmiş olur;

$$X_{m,n}^{1}X_{m,n}^{1} = \begin{bmatrix} x_0x_0 & x_0x_1 & \dots & x_0x_8 \\ 0 & x_1x_1 & \dots & x_1x_8 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & x_8x_8 \end{bmatrix} \tag{3.17}$$

Yukarıdaki eşitlikte (Denklem 3.17) gösterilen matris formun içerisinde simetrik çarpımlar bulunmamaktadır. Sıfırları hariç tutarak matrisin satırlarını aşağıda gösterilen eşitlikte (Denklem 3.18) vektör formunda ifade edilebilir;

$$X_{m,n}^{2} = [x_0x_0 \;\; x_0x_1 \;\cdots\; x_8x_8]^T \tag{3.18}$$

Dolayısıyla kuadratik kısım ise aşağıda gösterilen eşitlikteki (Denklem 3.19) gibi vektörlerin çarpımı şeklinde ifade edilebilir;

$$y_2(m,n) = W_2 {X^2_{m,n}}^T \tag{3.19}$$

Yukarıdaki eşitlikte (Denklem 3.19) gösterilen ağırlık vektörü aşağıdaki eşitlikte (Denklem 3.20) gösterildiği gibi vektör formunda ifade edilebilir;

$$W_2 = [w_0^2 \ w_1^2 \ w_2^2 \ \dots \ w_{45}^2] \tag{3.20}$$

W_2 ve $X^2_{m,n}$ öğelerinin eleman sayısı filtre maskesinin boyutlarına bağlıdır. $N \times N$ boyutlu maske için simetrik parçalar hariç olmak üzere kuadratik vektöründeki terimlerin sayısı aşağıdaki eşitlikte (Denklem 3.21) gösterildiği gibi hesaplanabilir.

$$N_{kuadratik} = N \times (N+1)/2 \tag{3.21}$$

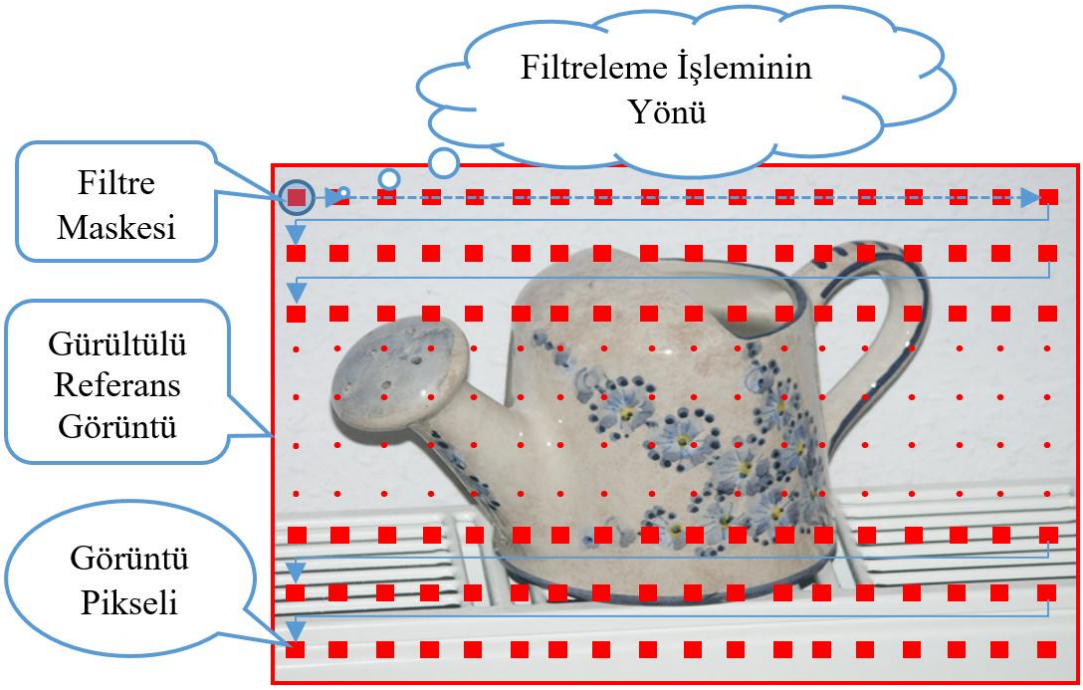
Yukarıda eşitlikler (Denklem 3.11 ve Denklem 3.19) ile verilen doğrusal ve kuadratik kısımların toplamı sabit terimi hariç tutulduğunda aşağıdaki eşitlikte (Denklem 3.22) gösterildiği gibi ifade edilebilir;

$$y(m,n) = W_1 {X^1_{m,n}}^T + W_2 {X^2_{m,n}}^T \tag{3.22}$$

Aynı açılımlar, Kuadratik görüntü filtrelerinin 5×5, 7×7 gibi daha büyük boyutlarda ifade etmek için de aynı şekilde genişletilebilir. Bu durumda yukarıda gösterilen eşitlikte (Denklem 3.22) kullanılan vektörün eleman sayısı yukarıda gösterilen eşitliğe (Denklem 3.21) göre belirlenir.

Yukarıda denklemlerle anlatılan Kuadratik görüntü filtreleri uygulamada filtreleme işlemlerini Şekil 3.3.'de görüldüğü sırada piksellerde işlem yapılarak gerçekleştirmektedir. *i* indisi filtrenin satır konumunu belirlerken, *j* indisi ise sütun konumunu belirlemektedir. Bununla birlikte yapılan çalışmada, filtre yapısından dolayı piksellerin filtrelenme sırasının görüntü kalitesi açısından önemi

bulunmamaktadır. Ayrıca GPU tabanlı görüntü filtrelemede birçok piksel aynı anda filtrelenmektedir.



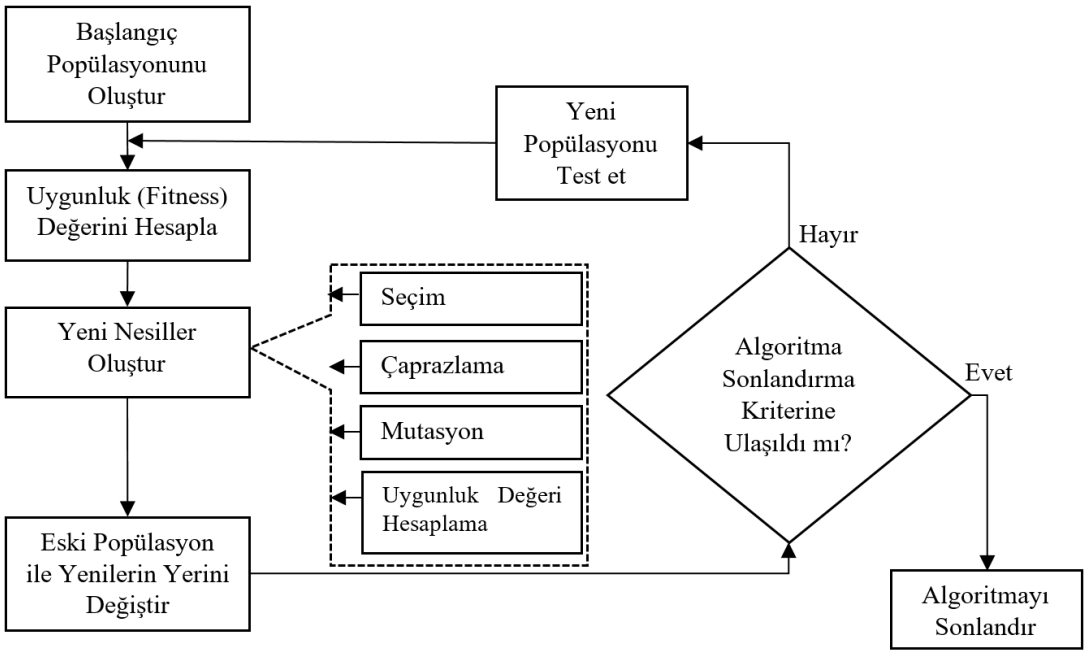
Şekil 3.3. Kuadratik görüntü filtrelerinin filtreleme işlem modeli

3.4. Sezgisel Algoritmalar ile Optimizasyon

Genel olarak optimizasyon işlemi, mümkün olan alternatifler içerisinden belirlenen koşullar ile en iyiyi seçme işlemidir [58]. Sezgisel algoritmalar ise optimizasyon problemleri için en yakın çözümleri verebilen algoritmalardır. Genel amaçlı sezgisel optimizasyon algoritmaları, biyoloji, fizik, kimya, sürü, sosyal ve müzik tabanlı olmak üzere altı farklı grupta değerlendirilmektedir [58]. Yapılan tez çalışmasında sürü tabanlı PSO algoritması ve biyoloji tabanlı GA tercih edilmiştir. GA ve PSO optimizasyon algoritmaları, yapılan kaynak araştırmalarında yaygın olarak kullanılan temel algoritmalar olmasından dolayı tez çalışması bu iki algoritma ile sınırlandırılmıştır. Tez çalışması kapsamında uygulanan algoritmalar benzer mantık ile çalışan diğer sezgisel optimizasyon yöntemlerine de genişletilebilir.

3.4.1. Genetik algoritmalar

GA, doğal genetik yapı ve seçime sahip, arama ve optimizasyon yöntemidir [59]–[62]. Bu tip bir algoritma global çözümü geleneksel yöntemlerle bulunamayan problemlerin çözümünde başarı sağlamaktadır [59], [60], [69], [61]–[68]. Buna benzer algoritmalar çözümü elde etmek için çaprazlama ve mutasyon adında farklı iki operatör kullanırlar. GA'da elde edilen her bir çözüme kromozom ya da birey adı verilmektedir [64]. GA'da elde edilen çözümlerin en iyileri belirlenerek bir sonraki nesillere aktarılmaktadır ve bu işlem seçme işlemi olarak adlandırılmaktadır. GA sonucunda rastgele birden çok çözüm elde edilebilir. Elde edilen bu çözümler arasından en iyileri seçilerek bir sonraki nesle aktarılır ve kötü sonuca sahip olanlar ise elenirler. Şekil 3.4.'de GA'ın çalışma şeması görülmektedir. Sistem girişinde renkli görüntü üzerine Gaussian gürültüsü eklenerek GA'a gönderilmektedir. Başlangıç popülasyonu rastgele oluşturulur ve bu oluşturulan popülasyona göre uygunluk değeri hesaplanır. Bu değerlere göre seçim, çaprazlama ve mutasyon operatörlerinden geçirilerek en iyi çözüm kümeleri oluşturulur.



Şekil 3.4. GA akış şeması

Yapılan çalışma için GA sonlandırma kriteri olarak maksimum tekrarlama sayısı kullanılmıştır. Sonuçta elde edilen en iyi uygunluk değerine göre filtre maske katsayıları bulunmaktadır. Bu katsayılar kullanılarak gürültülü görüntü son olarak filtrelenip hesaplanan uygunluk değeri gösterilmektedir.

Algoritma 1: GENETİK ALGORİTMALAR (GA)

Giriş: Referans Görüntü, Gürültü Eklenmiş Referans Görüntü, Görüntü Özellikleri, Durdurma Kriteri, Popülasyon Sayısı
Çıkış: En iyi Bireyler(En iyi Filtre maske ağırlıkları)

```
Başlangıç Popülasyonu Oluşturulur.
Uygunluk Değeri Hesaplanır.
Fitness();
Populasyon icerisindeki butun bireyler için tekrarlanır.
while (tekrarlama < maksimumtekrarlama) do
    Bütün Popülasyonlar için Uygunluk Değeri Hesaplanır.
    Fitness();
    Seçim Metodu Kullanılarak Bireyler Seçilir.
    Selection();
    Seçilen Bireyler Çaprazlanır.
    Crossover();
    Seçilen Bireyler Rastgele Mutasyona Uğratılır.
    Mutation();
    En iyi Adaylar Sonraki Nesle Aktarılır.
    Elitism();
return En iyi Uygunluk Değeri;
```

Şekil 3.5. GA sözde kodu

Şekil 3.5.'de GA için sözde kod verilmiştir. "*while*" ile belirtilen şart içerisinde tekrarlama sayısının belirlenen maksimum tekrarlama sayısı olduğu görülmektedir. Bu sayıya ulaşıncaya kadar uygunluk değeri hesaplamaları ve filtre ağırlıkları eğitimi sürekli olarak devam etmektedir.

3.4.2. Parçacık sürü optimizasyon algoritmaları

PSO, kuş ve balık sürülerinin davranışlarından esinlenerek geliştirilmiş hızlı yakınsama özelliğine sahip ve doğrusal olmayan problemlerin çözümüne yönelik tasarlanmış bir algoritmadır [70], [71]. Bu algoritma nispeten diğer optimizasyon algoritmalarına göre çok daha hızlı ve parametre sayısının az olması sebebiyle kolaydır [72], [73]. Sebebi ise içerisinde türev bilgisine ihtiyaç duyulmamasından dolayı problem çözmedeki karmaşık işlemlerin olmamasıdır [74]. Hızlı yakınsama özelliğinden dolayı GA ve Benzetim Tavlama algoritmalarının ardından en çok

çalışılan algoritmadır [75]. Algoritma rastgele çözümlerin olduğu bir sürüyle başlatılır. Diğer optimizasyon algoritmalarında sürü ismi yerine popülasyon terimi kullanılır. Sürüyü oluşturan her bir kuş bir parçacık olarak adlandırılır ve her bir parçacık bir aday çözümdür [75]. Örneğin D boyutlu X adet parçacık olduğunu varsayalım. Bu durumda parçacık matrisi aşağıda (Denklem 3.23) gösterildiği gibidir;

$$y = \begin{bmatrix} p_{11} & p_{12} & \vdots & p_{1D} \\ p_{21} & p_{22} & \vdots & p_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ p_{z1} & p_{z2} & \vdots & p_{zD} \end{bmatrix} \tag{3.23}$$

Parçacık matrisindeki *i*' inci parçacık aşağıda (Denklem 3.24) gösterildiği gibi ifade edilir.

$$x_i = [x_{i1} \quad x_{i2} \quad \cdots \quad x_{iD}] \tag{3.24}$$

Önceki tekrarlamalarda uygunluk değerini veren *i*' inci parçacığın pozisyonu aşağıda (Denklem 3.25) gösterildiği gibidir ve bunlar yerel en iyi parçacık olarak adlandırılmaktadır.

$$p_{eniyi_i} = [p_{eniyi_1 1} \quad p_{eniyi_1 2} \quad \cdots \quad p_{eniyi_1 D}] \tag{3.25}$$

Her tekrarlamada bütün parçacıklar içerisindeki en iyisi ise aşağıda (Denklem 3.26) gösterildiği gibidir ve bunlar global en iyi parçacık olarak adlandırılmaktadır.

$$g_{eniyi_i} = [g_{eniyi_1 1} \quad g_{eniyi_1 2} \quad \cdots \quad g_{eniyi_1 D}] \tag{3.26}$$

i' inci parçacığın hızı aşağıda (Denklem 3.27) gösterildiği gibidir.

$$v_i = [v_{i1} \quad v_{i2} \quad \cdots \quad v_{iD}] \tag{3.27}$$

Yerel en iyi ve global en iyi değerlerinin belirlenmesinin ardından her bir parçacığın hızları aşağıda (Denklem 3.28) gösterildiği gibi güncellenmektedir [76].

$$v_i^{k+1} = v_i^k + c_1 r_1^k \left(p_{eniyi_i}^k - p_i^k\right) + c_2 r_2^k \left(g_{eniyi}^k - p_i^k\right) \quad (3.28)$$

Yerel en iyi ve global en iyi değerlerinin belirlenmesinin ardından her bir parçacığın konumları aşağıda (Denklem 3.29) gösterildiği gibi güncellenmektedir [76].

$$p_i^{k+1} = p_i^k + v_i^{k+1} \quad (3.29)$$

Yukarıda verilen (Denklem 3.28) c_1 ve c_2 öğrenme faktörlerini ifade etmektedir. Diğer bir deyişle parçacığı yerel en iyiye ve global en iyiye yönlendiren sabitleri ifade etmektedir. Yukarıda verilen (Denklem 3.28) r_1 ve r_2 (0-1) arasında değişen rastgele değerler alan değişkenleridir. k ise tekrarlama sayısını göstermektedir. Şekil 3.6.'da PSO için genel sözde kod verilmiştir.

Algoritma 2: PARÇACIK SÜRÜ OPTİMİZASYON (PSO) ALGORİTMASI

Giriş: Verilen Problem için Uygunluk Fonksiyonu, Algoritma Parametreleri
Çıkış: Uygunluk Fonksiyonunu En Aza İndirmek için (veya Maksimize Etmek için) En İyi Birey

```
Rastgele Aday Çözümlerini Kullanarak Başlanır.
Her Bir Adayın Uygunluk Değeri Değerlendirilir.
while (tekrarlama sayısı < maksimum tekrarlama sayısı) do
    Yeni Bir Popülasyon Oluşturmak için Bireyler Yeniden Oluşturulur.
    Her Bir Çözümün Uygunluğu Değerlendirilir.
    Daha Iyi Uygunluk Değerleri ile Sahip Çözümler Seçilir.
    Baskın Olmayan Çözümler Güncellenir.
```

Return En İyi Bireyler;

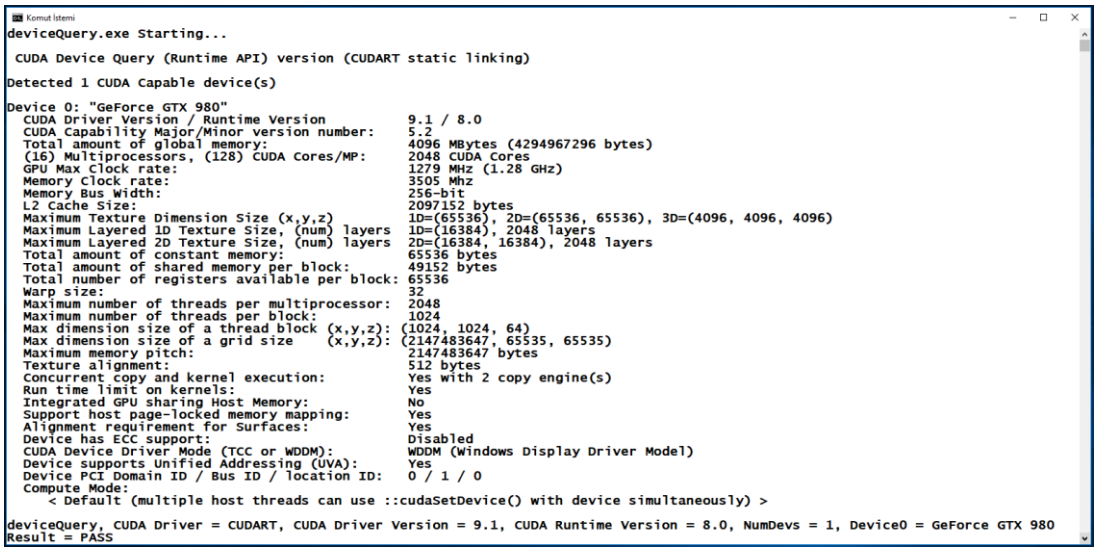
Şekil 3.6. PSO sözde kod

3.5. GPU Tabanlı Hesaplama

CPU mimarisi hızla gelişerek tek çekirdekli ve buradan çok çekirdekliye geçiş ile hesaplama başarımlarında ciddi artışlar sağlanmıştır. Fakat mikroişlemcilerin enerji tüketimleri ve ısı üretimlerinin artması CPU'ların gelişimini sınırlamıştır [77], [78]. Buna bağlı olarak ilk başlarda elde edilen performans artışları artık sağlanamamaya başlanmıştır. Bunun sonucunda özellikle paralel işlem yapan ve yüksek performans isteyen uygulamalar için günümüzde GPU kartlar yaygın bir şekilde kullanmaya

başlanmıştır. Ayrıca GPU kartların CPU'larla uyum içerisinde çalışması da GPU kartların kullanımını arttırmıştır [78], [79]. Yapılan tez çalışmasında GPU kart, CPU'nun işlem yükünü hafifletmek amacıyla çok fazla işlem gücü gerektiren işlemleri gerçekleştirmek için kullanılmıştır. Diğer seri işlemlerin hepsi CPU üzerinde yürütülmüştür. Günümüzde çok çeşitli ve çok sayıda GPU kartlar bulunmaktadır. Yapılan tez çalışmasında Kuadratik görüntü filtrelerinin maske ağırlıklarını eğitmek için GPU tabanlı algoritmalar geliştirilmiş ve bunun için Nvidia firması tarafından geliştirilen CUDA çatısı kullanılmıştır. CUDA, GPU üzerinde genel hesaplamalar için Nvidia tarafından geliştirilen bir programlama modeli ve paralel hesaplama platformudur [80]. CUDA ile geliştiriciler GPU gücünü kullanarak bilgisayar uygulamalarını önemli ölçüde hızlandırmaktadırlar [80]. GPU ile hızlandırılmış uygulamalarda iş yükünün ardışık kısımları genelde CPU üzerinde çalışırken uygulamanın hesaplama yoğunluğunun olduğu paralelleştirilebilen kısımlar ise paralel olarak binlerce GPU çekirdeği üzerinde çalıştırılır [80]. Bu grafik kartları farklı mimariler kullanılarak üretilmektedir. Ticari kaygılardan dolayı kullanılan mimariler hakkında çok fazla bilgi bulunmamaktadır [81]. Çalışmalarda kullanılan GPU mimarisi Maxwell mimarisine sahiptir. Maxwell mimarisi Nvidia'nın, GeForce 700 serisi, 800M serisi, GeForce 900 serisi ve Quadro Mxxx serilerinde kullanılmaktadır. Bu mimariyle üretilen GPU kartların L2 ön bellekleri 2MB yapılarak bellek bant genişlikleri arttırılmıştır. Buna göre bellek veri yolları 128 bit'e indirgenerek önceki mimarilere göre oldukça güç tasarrufu sağlanmıştır [82]. Ayrıca GPU için kullanılan ve çok önemli olan bir parametre ise hesaplama kapasitesidir (computing capability). Bu hesaplama kapasitesi GPU'dan GPU'ya ve GPU'nun oluşturuldukları mimariye göre değişmektedir. Nvidia CUDA C derleyicisi olarak Nvcc (NVidia's Cuda Complier)'yi kullanmaktadır. Nvcc, GPU üzerinde bulunan her bir çekirdek için hem mimariye özgü cubin dosyaları hem de ileri uyumluluk PTX (Parallel Thread Execution) sürümlerini üreterek kullanmaktadır. Her bir cubin dosyası, belirli bir hesaplama yeteneği versiyonunu kullanır ve sadece aynı sürüm ve türevleriyle uyumlu olarak çalışır. Örneğin hesaplama yeteneği 3.0'ı hedefleyen bir cubin dosyası, hesaplama yeteneği 3.x olan sürümlerin tamamını desteklerken, hesaplama yeteneği 5.x (Maxwell) olan cihazlarla uyumlu olarak çalışmaz [83].

Şekil 3.7.'de CUDA'nın "CUDA Device Query" adında oluşturduğu örnek projesinin çalıştırılması sonucunda elde edilmiş ekran görüntüsüdür. Bu ekran görüntüsünde GPU için hem donanımsal hem de yazılımsal özellikleri görülmektedir.



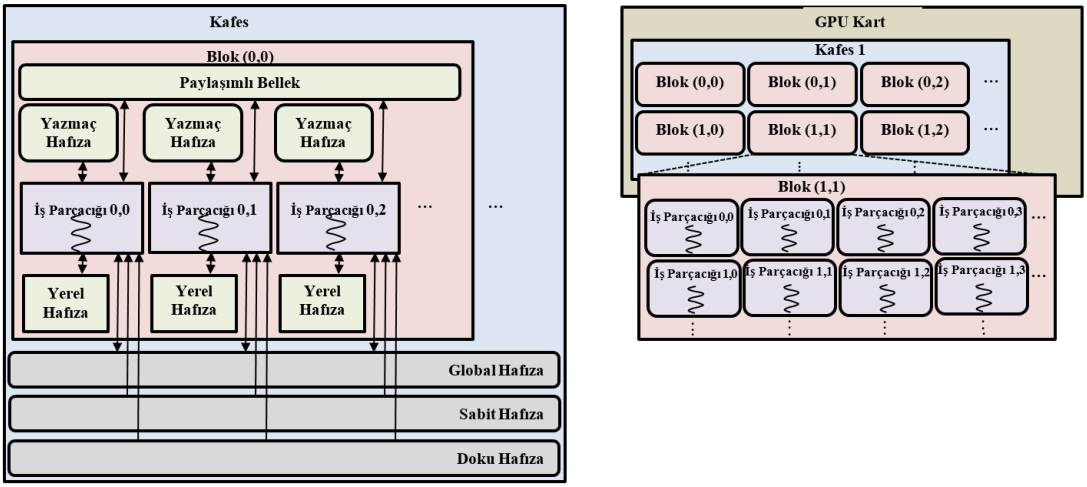
Şekil 3.7. Deneylerde kullanılan GPU'nun donanımsal ve yazılımsal kapasiteleri

Çalışmada kullanılan GPU kart 2048 adet CUDA çekirdeğe ve 4GB DDR5 Ram belleğe sahiptir. Her bir blok için 1024 adet iş parçacığı bulunmaktadır. Toplam oluşturulan blok parçacığı sayısı ise 1024 tanedir. Bunun anlamı GPU kart üzerinde yazılımsal olarak toplamda 1024×1024 tane iş parçacığı oluşturulmaktadır. GPU kartların hesaplama kapasiteleri bir birlerinden farklıdır ve Nvidia'nın GPU kartları için hesaplama kapasitesi versiyonları çok önemli bir kriterdir. Tez çalışmasında kullanılan GPU kartın hesaplama kapasitesi CUDA 5.2'dir ve OpenGL 4.4'ü de desteklemektedir. 165W güç tüketimine sahip ve maksimum güç tüketimi 500W'dır. Buradaki özelliklerinin dışında her bir iş parçacığı blokları 48KB paylaşımlı belleğe (Shared-memory), her bir iş parçacığı 16KB yazmaç belleğe (register memory) sahiptir.

3.5.1. CUDA mimarisi

Günümüzde hızla gelişen teknolojiyle, donanımlar üzerlerinde çalışan yazılımların ihtiyaçlarını karşılamada giderek zorlanmaktadır. Özellikle görüntü işleme

konularında, her bir piksel değeri bir filtre maskesi ile defalarca kez çarpılıp toplanmakta, ayrıca bu filtre maskesi ve işlenen görüntünün boyutları arttıkça yapılan işlemler daha fazla uzamaktadır. Bundan dolayı bu tarz iş yükü gerektiren işlemlerde çok fazla çekirdeğe ihtiyaç duyulmaktadır. GPU paralel ifade edilebilen ve işlem yükü fazla olan uygulamaları çalıştırmada, içerisindeki birçok çekirdekten dolayı oldukça başarılıdırlar. CUDA, grafik işlemciler üzerine yazılmış ilk paralel hesaplama platformu ve programlama modelidir [80]. Yazılım dili olarak C temelli bir dil olan CUDA C kullanılmaktadır. GPU içerisinde düşük güçlü iş parçacıklarının kullanılması bazı paralel algoritmaların, sıralı çalışan CPU algoritmalarına göre oldukça hızlı hesaplanmasını sağlamaktadır. Nvidia tarafından geliştirilen CUDA ile Nvidia grafik kartları genel amaçlı hesaplama uygulamalarında kullanılabilmektedir. GPU kartlar üzerinde çok sayıda CUDA çekirdekleri (CUDA cores) bulunmaktadır [84]. GPU üzerindeki tüm bu çekirdeklere yükler eşit dağıtıldığında zaman performansı açsından ne kadar iyi olduğu görülebilmektedir. CUDA mimarisi temelde grafik işleme yoğunluğu olan, ağır hesaplamaların olduğu uygulamalar ve çok fazla paralellik gerektiren uygulamalara yönelik olarak geliştirilmiştir. GPU üzerinde birçok veri işlemeye yönelik aritmetik ve lojik birimlerden (ALU) oluşmuş bir mimariye sahiptir. Bu mimari ile aritmetiğin yoğun olduğu görüntü işleme, 3D derleme, sinyal işleme vb. uygulamalarda oldukça başarılı olabilmektedir. CUDA mimarisi tasarımına eklediği kanal yapısıyla kullanıcılarının GPU kart içerisindeki çekirdeklere iş yükü paylaştırması, süreç kontrollerinin yapılması vb. işlemlerle uğraştırmayacak şekilde sağlamaktadır. Şekil 3.8.'de gösterildiği gibi her bir kafes içerisinde GPU kart donanımın özelliğine göre yüzlerce blok bulunmaktadır. Bu blokların her biri içerisinde hesaplamaları yapan yine donanımın özelliğine göre değişen sayıda iş parçacıkları bulunmaktadır. Bir kafes içerisinde çalışan iş parçacıklarının sayısı her blok içerisindeki iş parçacıklarının kafes içerisindeki blok sayısı ile çarpımı kadardır.



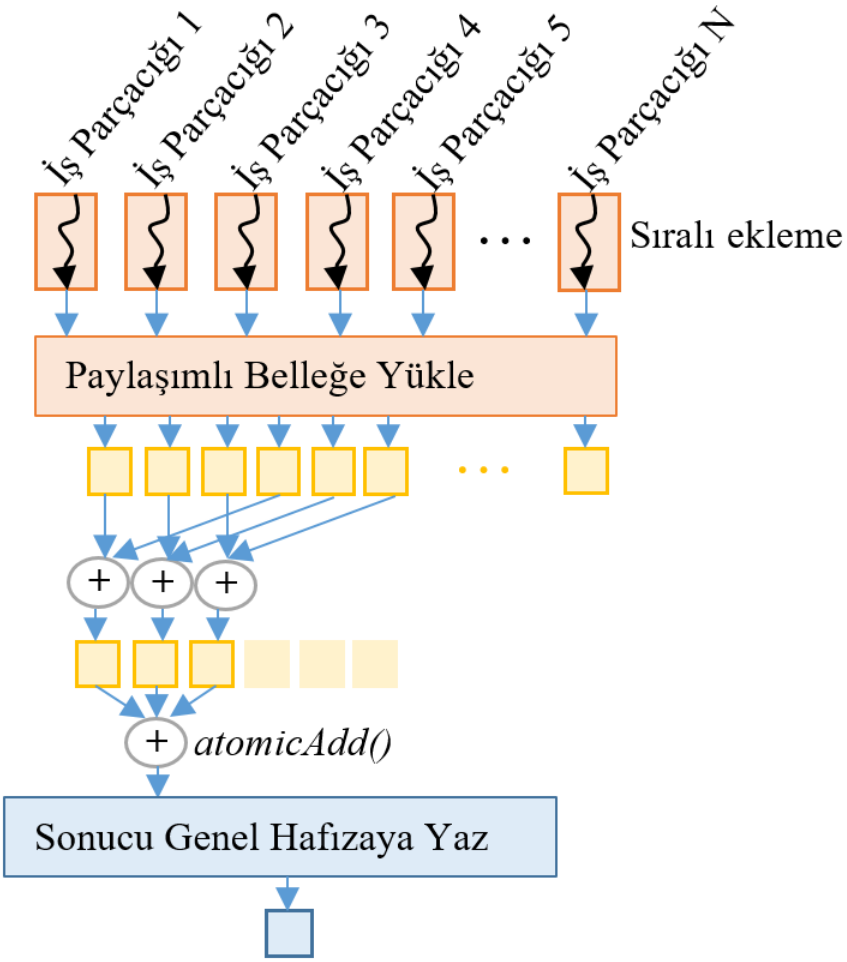
Şekil 3.8. CUDA mimarisinde kafes (grid) ve blok yapısı [85]

CUDA C ile yazılmış bir program aslında çekirdek (kernel) adı verilen seri bir programdır. CUDA mimarisi bu seri programdan binlerce kopya oluşturarak paralel bir şekilde işlenmesini sağlar. Çekirdek fonksiyonları, normal C fonksiyonlarından farklı olarak birkaç kere çağrıldığında birkaç tane ayrı iş parçacığını paralel olarak çalıştırırlar [86], [87]. Çekirdek fonksiyonları *__global__* ifadesi ile tanımlanmaktadır. Çekirdek fonksiyonunu çalıştıracak olan iş parçacığı sayısı <<<...>>> ifadesi ile belirtilir. Çekirdek fonksiyonunu çalıştıran her bir iş parçacığına bir *ID* (Identification) değeri atanır. Bu değere *threadIDx* değişkeni üzerinden ulaşılmaktadır [88]. CUDA mimarisi, paylaşımlı bellek, ön bellek, Yerel bellek (local memory), Genel bellek (Global memory), Sabit bellek (constant memory), Doku bellek(texture memory) olmak üzere farklı bellek yapılarına sahiptir. İş parçacıkları çalışma sırasında birden fazla bellek alanındaki verilere erişmektedirler [34]. İş parçacıklarının en hızlı ulaşabildikleri bellekler sırasıyla, her bir iş parçacığının kendine ait ön bellek ve yerel belleği, blokların hepsinin erişebildiği paylaşımlı bellek ve bütün iş parçacıklarının erişebildikleri genel bellek, sabit bellek ve doku bellektir [34], [40], [89], [90]. Yerel bellek alanı iş parçacıklarının bulunduğu yerde olduğundan erişim süresi ve gecikme çok kısa ve band genişliği düşüktür. Paylaşımlı bellek iş parçacığı bloklarına genel bellekten daha yakın olduğu için band genişliği yüksek ve hızı genel belleğe göre yüksek, yerel belleğe göre daha düşüktür. Genel bellek ise çok geniş band genişliğine sahip fakat GPU içerisindeki bütün iş parçacıklarının ulaşabileceği bir konumda olduğundan erişim süreleri diğer belleklere göre oldukça fazladır[34]. Sadece iş

parçacıklarının kullandığı bellek tipi ön bellek ve yerel bellek tipidir. Bir blok içerisinde bulunan bütün iş parçacıklarının ortak olarak eriştikleri bellek tipi paylaşımlı bellek tipidir. Bu bellek tipi CPU üzerinde biriktirme ara belleği gibi kullanılır. Amacı ise genel bellek erişim gecikmelerini azaltmaktır. GPU üzerinde ise bant genişliğini arttırmak için kullanılır. Bloklar dışında kafes yapısında bulunan Genel belleğe ise bütün iş parçacıkları ulaşabilmektedir. Sadece Sabit bellek ve Doku bellek tipleri ile iş parçacıkları arasında tek yönlü bir kullanım vardır. Sabit bellek GPU kartın hesaplama kapasitesine göre değişirken bu belleğin efektif kullanımı zordur. Doku bellek ise yüksek performanslı bir bellek olup GPU kart tarafından donanım olarak desteklenmektedir. Bu belleği uygulama ara yüzlerinde kullanıldığında sadece okuma işlemleri yapılabilir. Genel bellek üzerindeki işlemlere göre çok daha performanslı çalışır.

3.5.2. AtomicAdd() fonksiyonu

Atomik işlemler, herhangi bir iş parçacığından etkilenmeden gerçekleştirilen işlemlerdir. Genelde çoklu iş parçacığı uygulamalarında çok yaygın olan yarış koşulları (race condition) problemini önlemek için kullanılmaktadır. CUDA'daki atomik işlemler genellikle paylaşımlı hafıza ve genel hafıza için çalışmaktadır. Paylaşılan hafızadaki atomik işlemler genellikle aynı iş parçacığı bloğu içerisindeki farklı iş parçacıkları arasındaki yarış koşullarını önlemek için kullanılmaktadır. Genel hafızadaki atomik işlemler, hangi iş parçacığının bloğuna bakılmaksızın iki farklı iş parçacığı arasındaki yarış koşullarını önlemek için kullanılmaktadır [91]. GPU kart mimarisinde hesaplama yeteneği 6.x'den daha düşük olanlarda GPU kart üzerinde yapılan atomik işlemler yalnızca GPU'ya göre atomiktir. Şekil 3.9.'da *atomicAdd()* fonksiyonuna örnek bir çalışma şekli görülmektedir.



Şekil 3.9. AtomicAdd() fonksiyonunun çalışma şekli[92]

İş parçacıklarında yapılan hesaplamalar *atomicAdd()* fonksiyonu tarafından toplanmaktadır ve toplam sonucu yazana kadar işini bitiren bütün iş parçacıkları diğer iş parçacıklarını beklemektedirler. *atomicAdd()* fonksiyonuna ait örnek kod parçası şekil 3.10.'da gösterilmektedir [91], [93].

```
int atomicAdd(int *dosya, int degisken){
int eski;
eski = *dosya; //hafızadan yükle
*dosya=eski+degisken;// degisken ile topladıktan sonra hafızaya yükle
return eski;
}
```

Şekil 3.10. AtomicAdd() fonksiyonu kod bloğu

3.5.3. __Syncthreads() fonksiyonu

Yarış koşullarından kaçınmak için iş parçacıkları tarafından paylaşılan alanlara erişimlerin doğru şekilde senkronize edilmesi gerekmektedir. Paralel işlemlerin yapıldığı uygulamalarda ortak depolama alanlarına işlemlerini bitiren iş parçacıklarının yazma ya da okuma yapmaları kaçınılmazdır. Bu gibi durumlarda iş parçacıklarının o depolama alanlarına ulaşımları senkron edilmeli aksi takdirde elde edilen sonuçlar yanlış çıkacaktır. CUDA yarış koşullarından kaçınmak için atomik işlemler gibi bazı önlemler almış olsa bile zaman zaman iş parçacıklarının senkronize edilmeye ihtiyaçları olmaktadır. *__syncthreads()* fonksiyonu bir iş parçacığının ulaşması gereken bir depolama alanına ulaştıktan sonra hiç bir işlem yapmadan diğer iş parçacıklarını beklemesini sağlamaktadır. Böylece o depolama alanına aynı iş parçacıklarının diğer iş parçacıklarına göre daha fazla kullanmasının önüne geçilmiş olacaktır. Örneğin Şekil 3.11.'de görülen kod bloğunda, *Thread_1*, *Thread_0* tarafından tanımlanan bir değeri okur.

```
__global__void update_race(int* x, int* y){
int i = threadIdx.x + blockDim.x * blockIdx.x;
if ( i == 0 ) *x=1;
if ( i == 1 ) *y = *x;
}
```

Şekil 3.11. Senkronize edilmemiş kod bloğu [94]

Programın *Thread_1*'in yazılan konumundan sonra konumunu *Thread_1* değerini okumasını sağlaması gerekmektedir. Şekil 3.12.'de ise aynı blok içerisindeki iş parçacıklarının senkronizasyon yerleri belirlenir ve *__syncthreads();* öğesi kullanılarak *Thread_1* yazma işini bitirdikten sonra diğer iş parçacıklarını bekleyecek ve sonrasında okuma işini yapacaktır.

```
__global__void update_race(int* x, int* y){
int i = threadIdx.x + blockDim.x * blockIdx.x;
if ( i == 0 ) *x=1;
__syncthreads();
if ( i == 1 ) *y = *x;
}
```

Şekil 3.12. Senkronize edilmiş kod bloğu [94]

3.6. Sonuç

Bu bölümde, Kuadratik görüntü filtreleri detaylı bir şekilde ele alınarak matematiksel temelleri anlatılmıştır. Ayrıca bu filtrelerin maske ağırlıklarını eğitmek için kullanılan sezgisel algoritmalar da ayrı ayrı ele alınmıştır. Tez çalışması kapsamında Kuadratik filtrelerin maske ağırlıklarının eğitim sürelerini azaltmak için geliştirilen yöntemlerden biri olan donanımsal hızlandırmanın tasarlanacağı ortam ve çalışacağı donanım hakkında detaylı açıklamalar yapılmıştır. Ayrıca CUDA çerçevesi hakkında temel bilgiler de verilmiştir. Bu bölümde bahsedilen teknolojiler kullanılarak Kuadratik görüntü filtrelerinin maske ağırlıkları eğitim sürelerinin ciddi anlamda düşürülmesi hedeflenmektedir.

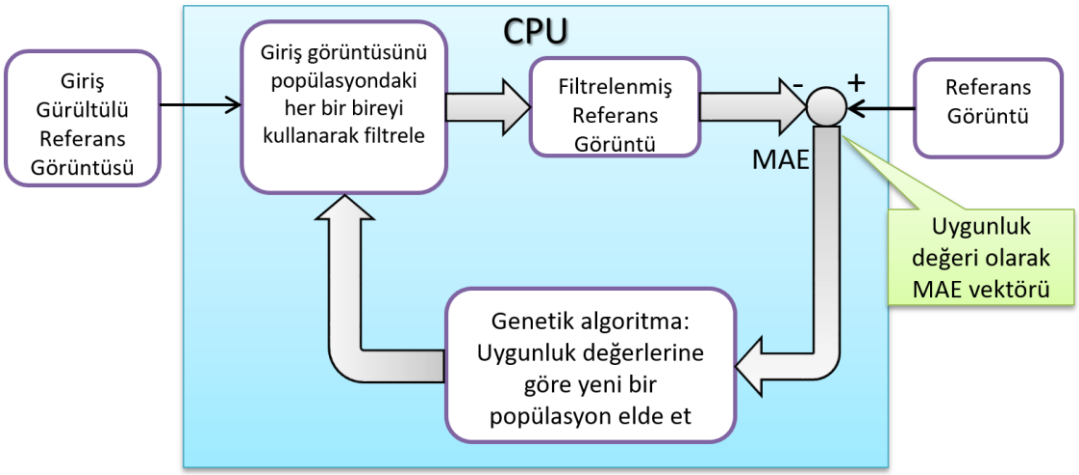
BÖLÜM 4. ÖNERİLEN YÖNTEM

4.1. Giriş

Sezgisel algoritmalar, problemlerin türüne bağlı olarak bir sorunun çözümü için yoğun hesaplama gücü isteyebilir. GA, PSO ve benzer sezgisel algoritmalar başlangıçta rastgele bir popülasyon oluşturur ve tekrarlama sayısı ilerledikçe hesaplanan uygunluk değerine göre bu popülasyonlar değişikliğe uğratılır. Gürültülü görüntüyü filtrelemek için kullanılan görüntü filtre katsayılarının en uygun olanını belirlemek gerekir. Bunun için gürültülü referans görüntüsünün uygunluk değerinin referans görüntüsünün uygunluk değerine yaklaştırılarak kıyaslanması gerekir. Uygunluk fonksiyonunun referans görüntüye yakınlığını belirlemek için MSE, MAE (Mean Absolute Error), PSNR yada SSIM (Structural Similarity Index Measurement) gibi görüntü kalitesi ölçüm tekniklerinden biri kullanılabilir. Mevcut yaklaşımlarda da, popülasyonun bir bireyi için uygunluk değeri, filtrelenmiş referans görüntü ve referans görüntü kullanılarak MAE veya MSE değerlerinin hesaplanması ile belirlenir. Filtrelenmiş görüntü, bir bireyin maske ağırlıkları olarak görüntünün filtrelenmesi ile elde edilir. Bu işlemler popülasyondaki her bir birey için her bir algoritma tekrarlama sayısında tekrarlanır ve böylece popülasyon gelişir. Bütün popülasyon için uygunluk hesaplanması genellikle yoğun hesaplama gücü gerektirir. Dolayısıyla uygunluk fonksiyonundaki (Fitness Function) algoritmik bir hızlandırma, hesaplama süresini önemli derecede azaltabilir. Bu amaçla, GA ve PSO kullanılarak Kuadratik filtrelerin eğitim süresini azaltmak için değişken bir yaklaşım ve bu yaklaşımın bir varyasyonu sunulmuştur. Ayrıca uygunluk fonksiyonunun GPU üzerine taşınarak hesaplanmasını temeline dayanan ve önerilen donanımsal hızlandırma yöntemleri de bu bölümde ayrıntılı olarak anlatılmıştır.

4.2. CPU Üzerinde Sıralı Gerçekleştirme

CPU üzerinde sıralı gerçekleştirme, hem algoritmik hızlandırma hem de GPU kart üzerinde gerçekleştirilen donanımsal hızlandırmayı ölçmek için referans almak amacıyla geliştirilmiştir. Kuadratik filtrelerin uygunluk değeri hesaplamaları CPU içerisinde sıralı bir şekilde gerçekleşmektedir. Bu da hesaplamaları oldukça yavaşlatmaktadır. Şekil 4.1.'de Sabit Uygunluk (SU) yönteminin çalışma şeması görülmektedir.



Şekil 4.1. SU yöntemi çalışma şeması

Genel olarak SU yöntemi şu şekilde çalışır; giriş gürültülü referans görüntüsü, GA tarafından başlangıçta rastgele olacak şekilde üretilen Kuadratik filtrelerin maske ağırlıkları kullanılarak filtrelenmiş referans görüntü elde edilir. Uygunluk değerinin hesaplanması için MAE kullanılmıştır. Yani filtrelenen referans görüntü ile referans görüntüsü arasındaki farkların mutlak değerleri alınarak hesaplanır. Daha sonra bu uygunluk değeri GA tarafından kullanılarak yeni maske ağırlıkları hesaplanır. Bu süreç maksimum tekrarlama sayısına ulaşıncaya kadar devam eder.

4.3. Önerilen Algoritmik Hızlandırma Yöntemleri

Kuadratik filtrelerin GA ve PSO ile eğitiminde, popülasyonların tamamı için uygunluk değeri hesaplamaları için çok yoğun hesaplama gücüne ihtiyaç duyulur. Bu hesaplama

gücünü elde etmek amacıyla bu tez çalışması kapsamında farklı algoritmik hızlandırma yöntemleri önerilmiştir. Önerilen bu yöntemlerin temel prensibi GA ve PSO'nun eğitime giriş referans görüntüsünün önceden belirlenen bir bölgesinden başlamasının sağlanmasıdır. Sezgisel algoritmaların eğitime giriş referans görüntünün bir kısmından başlaması eğitim süresini kısaltacak ve çok daha kısa sürelerde en iyi çözümleri bulması sağlanacaktır. Ayrıca eğitim sonlanmadan önce örneğin, tekrarlama sayısının %80'nine ulaşıldığında GA ve PSO maske ağırlıkları eğitimlerine giriş referans görüntüsünün tamamında yapması sağlanmıştır. Önerilen bu yöntemler, Değişken Uygunluk (DU) yöntemi ve Rastgele Değişken Uygunluk (RDU) yöntemleridir. DU, GA'a entegre edilmiş olup uygunluk fonksiyonunda kullanılacak olan piksel sayısını Şekil 4.2a.'da gösterildiği gibi belirli oranlarda değiştirmektedir. Bu yaklaşımda, görüntünün boyutu GA'ın mevcut tekrarlamalarına göre belirlenir ve önceden belirlenmiş bir sayıdaki pikseller kullanılarak uygunluk fonksiyonu hesaplanır. Aşağıdaki eşitlikte (Denklem 4.1) k. tekrarlamadaki görüntünün genişliğini hesaplamada kullanılmaktadır.

$$W_K = W_{RI} \times B_{CR} + \frac{W_{RI} \times k}{M_I} - \frac{B_{CR} \times W_{RI} \times k}{M_I} \tag{4.1}$$

Denklemde kullnılan; W_K k. tekrarlamadaki görüntünün genişliğini, W_{RI} referans görüntünün genişliğini, B_{CR} sabit oranını, M_I maksimum tekrarlama sayısını ve k tekrarlama sayısını temsil etmektedir. B_{CR} değişkenini ortak paranteze alınırsa denklem aşağıdaki eşitlikte (Denklem 4.2) gösterildiği gibi elde edilir;

$$W_K = W_{RI} \times B_{CR}\left(1 - \frac{k}{M_I}\right) + \frac{W_{RI} \times k}{M_I} \tag{4.2}$$

Yukarıdaki eşitliği (Denklem 4.2) daha anlaşılır hale getirmek için $\frac{k}{M_I}$ 'yı I_{NI} şeklinde gösterirsek eşitlik aşağıda (Denklem 4.3) gösterildiği gibi elde edilir;

$$W_K = W_{RI} \times B_{CR}\left(1 - I_{NI}\right) + W_{RI} \times I_{NI} \tag{4.3}$$

Görüntünün genişliğini yukarıdaki eşitliklerle (Denklem 4.1, Denklem 4.2 ve Denklem 4.3) hesaplandığı gibi görüntünün yüksekliği de aşağıdaki eşitlikte (Denklem 4.4) gösterildiği gibi hesaplanır.

$$H_K = H_{RI} \times B_{CR} + \frac{H_{RI} \times k}{M_I} - \frac{B_{CR} \times H_{RI} \times k}{M_I} \tag{4.4}$$

Yukarıdaki eşitlikte (Denklem 4.4) kullanılan, H_K k. tekrarlamadaki görüntünün yüksekliğini, H_{RI} referans görüntünün yüksekliğini ifade eder. Denklemde geçen B_{CR} değişkenini ortak paranteze alınırsa eşitlik aşağıda (Denklem 4.5) gösterildiği gibi elde edilir.

$$H_K = H_{RI} \times B_{CR}\left(1 - \frac{k}{M_I}\right) + \frac{H_{RI} \times k}{M_I} \tag{4.5}$$

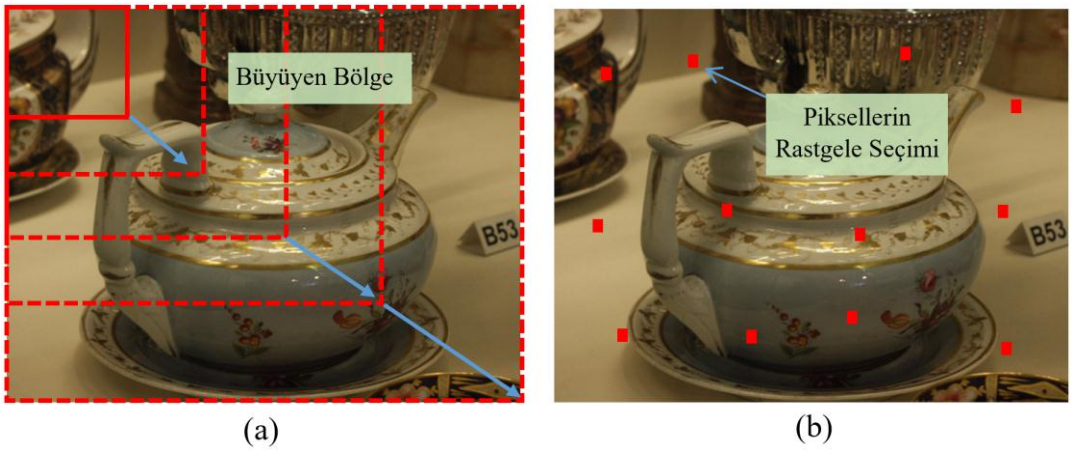
Yukarıdaki eşitliği (Denklem 4.5) daha anlaşılır hale getirmek için $\frac{k}{M_I}$ 'yı I_{NI} şeklinde gösterirsek eşitlik aşağıda (Denklem 4.6) gösterildiği gibi elde edilir;

$$H_K = H_{RI} \times B_{CR}\left(1 - I_{NI}\right) + H_{RI} \times I_{NI} \tag{4.6}$$

Yukarıdaki eşitliklerde (Denklem 4.3 ve Denklem 4.6) görüntünün her bir tekrarlamadaki genişliğini ve yüksekliğini hesaplamaktadır. B_{CR} sabit oranı ise $\left(B_{CR} = \frac{1}{2^n}\right)$ şeklinde kullanılmıştır.

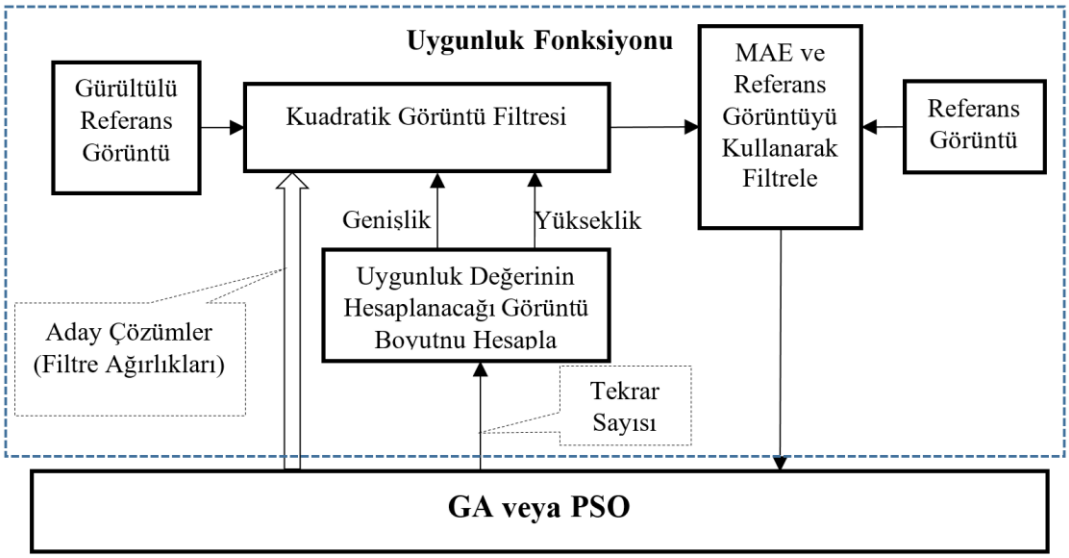
Diğer önerilen algoritmik hızlandırma yöntemi olan RDU'de GA ve PSO'ya entegre edilmiş olup uygunluk fonksiyonunda kullanılacak piksel sayısını değiştirmektedir.

DU yönteminden farklı olarak bu yöntemde GA ve PSO'nun eğitim yaptığı görüntü parçalarının pikselleri Şekil 4.2b.'de görüldüğü gibi bütün görüntü üzerinden rastgele bir şekilde seçilmektedir. GA ve PSO'nun eğitim yaptığı görüntünün boyutları da yukarıda gösterilen eşitliklerle (Denklem 4.3 ve Denklem 4.6) hesaplanmaktadır.



Şekil 4.2 Uygunluk fonksiyonunu hesaplamak için yöntem a) Bölgenin aşamalı büyümesi b) Piksellerin rastgele seçilmesi [95].

Önerilen yöntemlerin temel amacı, GA ve PSO'nun eğitime, görüntünün çok daha küçük bir alanında başlaması ve başlangıçtan itibaren çok daha iyi sonuçlar vermesini sağlamaktır. Bu yöntemlerin blok diyagramı Şekil 4.3.'de gösterilmiştir. Optimizasyon algoritmaları başlangıçta bütün görüntüyü kullanmak yerine yukarıda anlatılan matematiksel denklemlerle hesaplanan ve çok daha küçük bir alanı kullanarak uygunluk değerini hesaplarlar. Uygunluk değerinin hesaplandığı bu alan bütün popülasyonlar tarafından kullanılır. Bu şekilde uygunluk fonksiyonunun getirdiği hesaplama yoğunluğu azaltılmış olur. Hesaplanan bu uygunluk değerine göre yeni ağırlıklar belirlenmektedir. Bir sonraki tekrarlamada hem bu yeni ağırlıklar hem de eğitimin yapılacağı görüntünün boyutları yeniden hesaplanır. Uygunluk değeri ise hesaplanan yeni görüntü boyutu değişmeden bütün popülasyon tarafından kullanılarak hesaplanır.



Şekil 4.3 Önerilen algoritmanın blok diyagramı

Önerilen yöntemlerin genel olarak algoritmalarına bakıldığında Şekil 4.4.'de DU yöntemine ait algoritmanın sözde kodu görülmektedir. GA ve PSO, Kuadratik filtrelere ağırlık olarak aday bir çözüm uygular ve daha sonra gürültü eklenmiş olan referans görüntüsü filtrelenir.

Algorithm 3: Değişken Uygunluk (DU) Yöntemi

Giriş: Uygunluk Fonksiyonu, Popülasyon Sayısı, Maksimum Tekrarlama Sayısı, Başlangıç Oranı, GENİŞLİK, YÜKSEKLİK
Çıkış: En iyi Birey

```
Bir Popülasyon Oluşturmak için Belirli Sayıda Aday Çözüm Üretilir.
Popülasyon=Başlangıç Popülasyonu(Popülasyon Sayısı) ;
Başlangıç Uyugunluk Değerleri için Yeni Genişlik ve Yükseklik Hesaplanır.
Genişlik= GENİŞLİK*Başlangıç Oranı ;
Yükseklik= YÜKSEKLİK*Başlangıç Oranı ;
Bütün Popülasyon Üyeleri için Uygunluk Değeri Hesaplanır.
Uygunluk Değeri=Uygunluk Fonksiyonu(Popülasyon,Genişlik,Yükseklik) ;
Değişken Uygunluk Değeri için Başlangıç Oranı Belirlenir.
for (k=1 to MaksimumTekrarlamaSayısı) do
    Bir Seçim Yöntemi Kullanarak Ebeveynler Seçilir.
    Selection(Uygunluk Değeri,Popülasyon) ;
    Seçilen Ebeveynler Çaprazlanır.
    Crossover(Uygunluk Değeri,Popülasyon) ;
    Seçilen Üyeler Rastgele Mutasyona Uğratılır.
    Mutation(Uygunluk Değeri,Popülasyon) ;
    k ıncı Tekrarlama için Yeni Genişlik ve Yükseklik Değerleri Hesaplanır.
    Genişlik= GENİŞLİK*Başlangıç Oranı + (1- Başlangıç Oranı)*GENİŞLİK*k / Maksimum Tekrarlama;
    Yükseklik= YÜKSEKLİK*Başlangıç Oranı + (1- Başlangıç Oranı)*YÜKSEKLİK*k / Maksimum Tekrarlama;
    Görüntünün Genişlik×Yükseklik Kadar Kısmı için Uygunluk Değeri Hesaplanır.
    Uygunluk Fonksiyonu(Popülasyon,Genişlik,Yükseklik) ;
    En Iyi Adayların Sonraki Kuşaklara Aktarılır.
    Elitism(Uygunluk Değeri,Popülasyon) ;
Return En İyi Birey ;
```

Şekil 4.4. DU yönteminin sözde kodu

GA ve PSO ayrıca, uygunluk fonksiyonu hesaplamak için piksel sayısını genişlik ve yükseklik değerlerini belirleyerek sınırlamaktadır. Seçilen sayıda piksel için filtrelenmiş ve referans görüntülerin MAE değeri hesaplandıktan sonra, GA ve PSO'ya uygunluk değeri olarak sağlanmaktadır. Bir sonraki tekrarlama için, yeni genişlik ve yükseklik değerlerini belirleyerek uygunluk fonksiyonunun hesaplanmasında kullanılan piksel sayıları arttırılmıştır. Bu işlemler son tekrarlama sayısına kadar sürdürülür. Giriş referans görüntüsünün tamamının kullanımı tekrarlama sayısının %80'nine ulaşıldığında kullanılmaya başlanır. Bu değerin daha düşük olması eğitim süresini arttırmaktadır. Yüksek olması ise elde edilen maske ağırlıklarının bütün görüntü üzerindeki eğitimlerinin sayısı az olacağından görüntü kalitesi daha düşük çıkmaktadır. Uygunluk değeri hesaplamaları bütün tekrarlamalar boyunca görüntünün tamamında değil de görüntünün belirli bir bölümünde yapıldığı için eğitim süreleri kısalmakta, GA ve PSO ise daha hızlı sonuç vermektedir.

Algoritma 4: Rastgele Değişken Uygunluk (RDU) Yöntemi

Giriş: uygunluk Fonksiyonu, Popülasyon Sayısı, Maksimum Tekrarlama Sayısı, Başlangıç Oranı, Genişlik, Yükseklik
Çıkış: En iyi Bireyler

```
Bir Popülasyon Oluşturmak İçin Belirli Sayıda Aday Çözüm Üretilir.
Popülasyon = Başlangıç Popülasyonu (Popülasyon Sayısı)
İlk Uygunluk Değeri için Yeni Genişlik ve Yükseklik Hesaplanır.
v_rows= rows*initialRatio
v_cols=cols*initialRatio
Yeni Görüntü Boyutu için Rastgele Satırlar ve Sütunlar belirlenir.
randRowVec=fillRand(v_rows)
randColVec=fillRand(v_cols))
Satırlar×Sütunlar Piksel Sayısı Kullanılarak Uygunluk Değeri Hesaplanır.
Uygunluk Vektörü = Uygunluk Fonksiyonu(Popülasyon,v_rows,v_cols, randRowVec, randColVec)
Değişken Uygunluk Değeri için Başlangıç Oranı Belirlenir.
for (k=1 to MaksimumTekrarlamaSayısı) do
    Yeni Popülasyon Oluşturulur.
    Yeniden Oluşturma(Uygunluk Vektörü,Popülasyon)
    İlgili Tekrarlama için Yeni Yükseklik ve Genişlik Değeri Hesaplanır.
    v_rows= rows× initialRatio + (1- initialRatio)× rows × k / maxIter)
    v_cols=cols× initialRatio + (1- initialRatio) × cols × k / maxIter
    Yeni Görüntü Boyutu için Rastgele Yükseklik ve Genişlik Değerleri
    Hesaplanır.
    randRowVec=fillRand(v_rows)
    randColVec=fillRand(v_cols)
    Satırlar×Sütunlar Piksel Sayısı Kullanarak Uygunluk Değeri Hesaplanır.
    fitnessVec=fitnessFunc(Pop,v_rows,v_cols, randRowVec, randColVec)
    En iyi Aday veya Adaylar Seçilir ve Güncellenir.
    Seç ve Güncelle(Uygunluk Değeri Vektörü,Popülasyon)
Return En İyi Birey
```

Şekil 4.5. RDU yönteminin sözde kodu

RDU yöntemine ait algoritmanın sözde kodu Şekil 4.5.'de görülmektedir. Bu algoritmada ise eğitimin yapılacağı bölgenin pikselleri bütün görüntü üzerinden rastgele olacak şekilde seçilmektedir. Böylece görüntü özelliklerinin uygunluk fonksiyonuna daha iyi yansıtılması sağlanmıştır. Önerilen algoritmik hızlandırmalarda başlangıçta oluşturulan popülasyon üzerinden uygunluk değerinin hesaplanması işleminin de oldukça uzun sürdüğü gözlemlendiğinden bu süreninde kısaltılması için başlangıçta kullanılan görüntünün de boyutları belirli oranda küçültülmeye ihtiyaç duyulmuştur. Yukarıda anılan uygunluk değeri hesaplama işlemine ait algoritmanın sözde kodu Şekil 4.6.'da görülmektedir. Önce uygunluk değerinin hesaplandığı alan belirlenmektedir. Bu alan bir kez belirlendikten sonra bütün popülasyonlar aynı alanı kullanmaktadırlar. Sonrasında bütün popülasyon için uygunluk değerleri hesaplanır.

Algoritma 5: Uygunluk Fonksiyonu

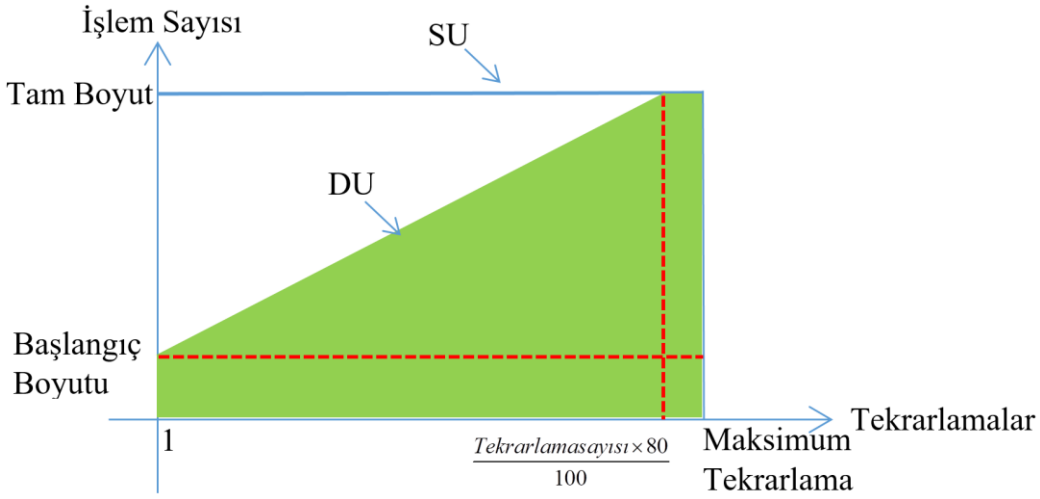
Giriş: Referans Görüntü, Popülasyon, v_rows, v_cols, randRowVec, randColVec
Çıkış: Popülasyon için Uygunluk Değeri Olarak MAE vektörü

```
Popülasyondaki Tüm Bireyler Üzerinde Tekrarlanır.
for (k=1 to length(Pop)) do
    Ağırlıklar Olarak k ıncı Bireyi Kullanarak Görüntü Filtrelenir.
    for (m=1 to v_rows) do
        for (n=1 to v_cols) do
            Rastgele İndeks Üretilir.
            x = randRowVec(m)
            y = randColVec(n)
            Aday Çözümü Kullanarak Piksel Hesaplanır.
            FiltImage(x,y) = filterPixel(x,y,Pop(k))
    k ıncı Bireyin Uygunluk Değeri Hesaplanır.
    fitnessArray(k) = Error(FiltImage, RefImage)
```

Return Uygunluk Değeri Dizisi

Şekil 4.6. Uygunluk değeri hesaplama algoritması

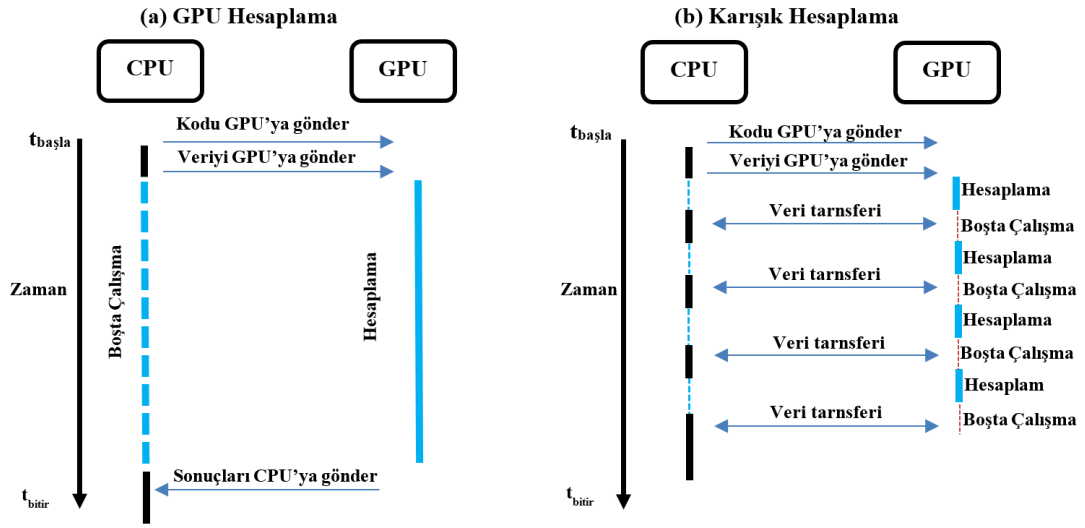
Şekil 4.7.'de önerilen yöntemlerde tekrarlamaların görüntü boyutuna oranları verilmektedir. Başlangıçta sezgisel algoritmalar tarafından rastgele olacak şekilde üretilen popülasyonlar için uygunluk değeri hesaplamaları Şekil 4.7.'de görüldüğü gibi çok küçük bir parçadan başlamaktadır. Görüntü önceden belirlenen toplam tekrarlama sayısının %80'ine ulaştığında tam boyuta ulaşır ve bir süre eğitim bu tam görüntü üzerinde devam ederek biter.



Şekil 4.7. SU ve DU için işlemlerin sayısı

4.4. Önerilen Donanımsal Hızlandırma Yöntemleri

CPU ve GPU birlikte güçlü bir hesaplama donanımı oluşturur. Burada seri işlemlerde daha verimli çalıştığı için CPU sıralı (seri) çalışan algoritmaları, seri işlemlerde çok performans göstermeyen GPU ise paralel algoritmaları çalıştıracak şekilde programlanabilir [81]. Sebebi ise CPU'ların çalışma frekansı çok daha yüksek ve daha güçlü çekirdeklere sahip olduklarından seri işlemlerde, GPU'ya göre daha performanslı çalışmaktadır. Bundan dolayı GPU'lar CPU'ların bir alternatifi olan teknolojiler değil birleriyle koordinasyonlu bir şekilde çalışan teknolojilerdir [81]. GPU ile hesaplama genelde Şekil 4.8.'de görüldüğü gibi iki yaklaşım ile kullanılır; birisi hesaplamayı tamamen GPU üzerinde gerçekleştirme, diğeri CPU ve GPU'nun paylaşımlı kullanıldığı hibrit yaklaşım. Her iki yaklaşımda da hesaplama başlamadan önce algoritmanın hesaplanması için gerekli bazı veriler için GPU kart üzerinde yer ayrılır ve ardından veriler transfer edilir. Bu işlemlerin hibrit yaklaşımda hesaplama anında da gerçekleştirilmesi gerekebilir. Yapılan tez çalışmasında, GPU kullanımı hibrit yaklaşımda olduğu gibi gerçekleştirilmesi planlanmıştır. GA ve PSO'nun çalışması sırasında uygunluk fonksiyonunun GPU üzerinde hesaplanması ve diğer kısımların CPU üzerinde yürütülmesi planlanmıştır. GA ve PSO tekrarlı olarak her yenilenen popülasyon için uygunluk fonksiyonlarını yeniden hesaplaması gerektiğinden, GPU çalışma anında Şekil 4.8b.'de görüldüğü gibi CPU ile paylaşımlı olarak görevleri gerçekleştirecektir.

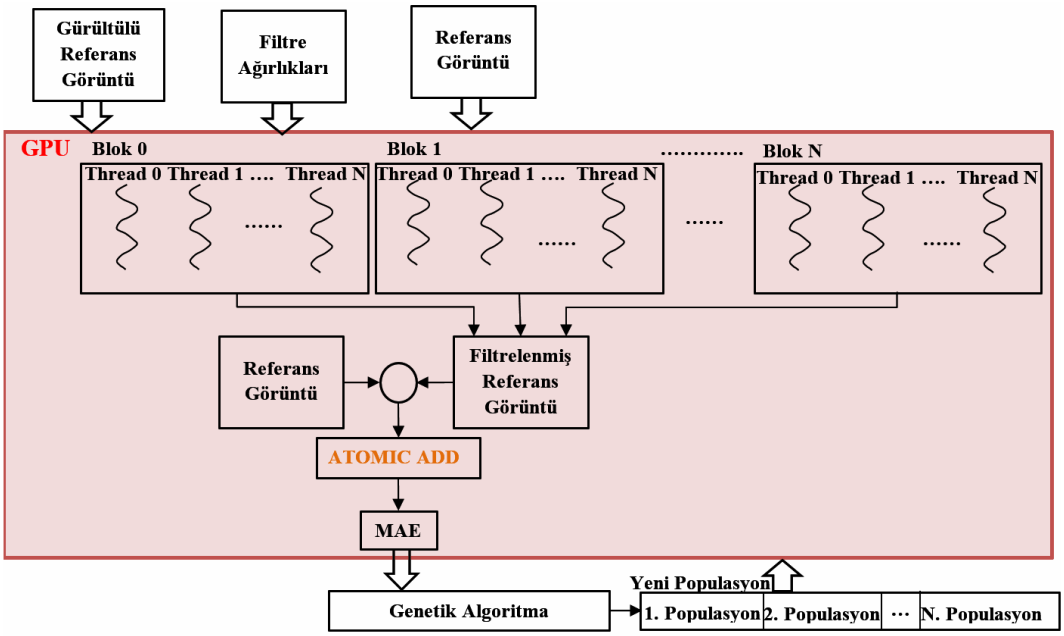


Şekil 4.8. GPU ve hibrit kullanım tabanlı GPU hesaplama yaklaşımı [85]

Algoritmanın hesaplama yükü bakımından en ağır olduğu kısım görüntü filtreleme işlemleri sonucu hata hesabının gerçekleştirildiği uygunluk fonksiyonu olduğu için özellikle uygunluk fonksiyonu GPU üzerinde hesaplanmasına odaklanılmıştır. Bu bölümde, GPU kartlar üzerinde çalışacak şekilde tasarlanan ve önerilen algoritmalar detaylı bir şekilde anlatılmıştır. Bu önerilen yöntemlerle Kuadratik görüntü filtrelerinin eğitim süreleri ciddi anlamda azalması planlanmaktadır.

4.4.1. GPU Sabit Uygunluk Yöntemi

GPU Sabit Uygunluk (GSU) yöntemi, SU yönteminde anlatıldığı gibi gerçekleşmektedir. Bu iki yöntem arasındaki tek fark, GSU yönteminde, uygunluk fonksiyonu paralel çalışacak şekilde CUDA C dili ile gerçekleştirilmiştir. Şekil 4.9.'da gösterildiği gibi gürültü eklenmiş referans görüntü, referans görüntü, maske ağırlıkları ve referans görüntü bilgileri GPU içerisinde çalıştırılmak üzere CPU'dan GPU'ya kopyalanmaktadır.



Şekil 4.9. GSU yöntemi çalışma şeması

GPU içerisindeki birçok iş parçacığı sayısı piksel sayısına göre belirlendiği için uygunluk değeri hesaplamaları CPU'ya göre oldukça kısa sürmektedir. GA tekrarlamalarında, uygunluk değerlerini ayrı ayrı hesaplamak için maske ağırlıklarını temsil eden her bir birey için ayrı GPU çekirdeği çalıştırılır. Şekil 4.10.'da görüldüğü gibi tasarlanan CUDA çekirdek koduyla SU yönteminin GPU'ya uygulanması son derece basit olmaktadır.

Algoritma 6: GPU SABİT UYGUNLUK (GSU) YÖNTEMİ

Giriş: Referans Görüntü, Gürültü Eklenmiş Referans Görüntü, Görüntü Maskesi Bilgileri, Diğer Parametreler
Çıkış: MAE (Ortalama Mutlak Hata)

```
    Piksel ID'lerini Belirlemek için İş Parçacığı ID'leri Alınır.
PixelID = BlockDim.x * BlockIDx.x + ThreadIDx.x ;
    Geçerli Blok için Paylaşılan Bir Hata Dizisi Tanımlanır.
_Shared_ Float BlockShared[256 ];
    Maske Verilerini Kullanarak Filtreleme İşlemi Piksellere Uygulanır.
Pikselleri Filtrele = Filtrele(Gürültü Eklenmiş Referans Görüntüsü, Maske Verileri, Piksel ID);
    Referans Görüntüyü Kullanarak MAE Hesaplanır.
BlockShared = abs(Filtrelenmiş Referans Görüntüsü - Referans Görüntü);
    Hesaplanan MAE'ler için İş Parçacıkları Senkronize Edilir.
_Syncthreads();
    Blok Paylaşılan Hata Dizisinin MAE'si Hesaplanır.
BlockSharedMAE = ParalellSum(BlockShared);
    Bütün Görüntünün MAE'sini Hesaplamak için Tüm Paylaşılan Blok Toplanır.
AtomicAdd( δ MAE, BlockShared /(256*3));
Return MAE ;
```

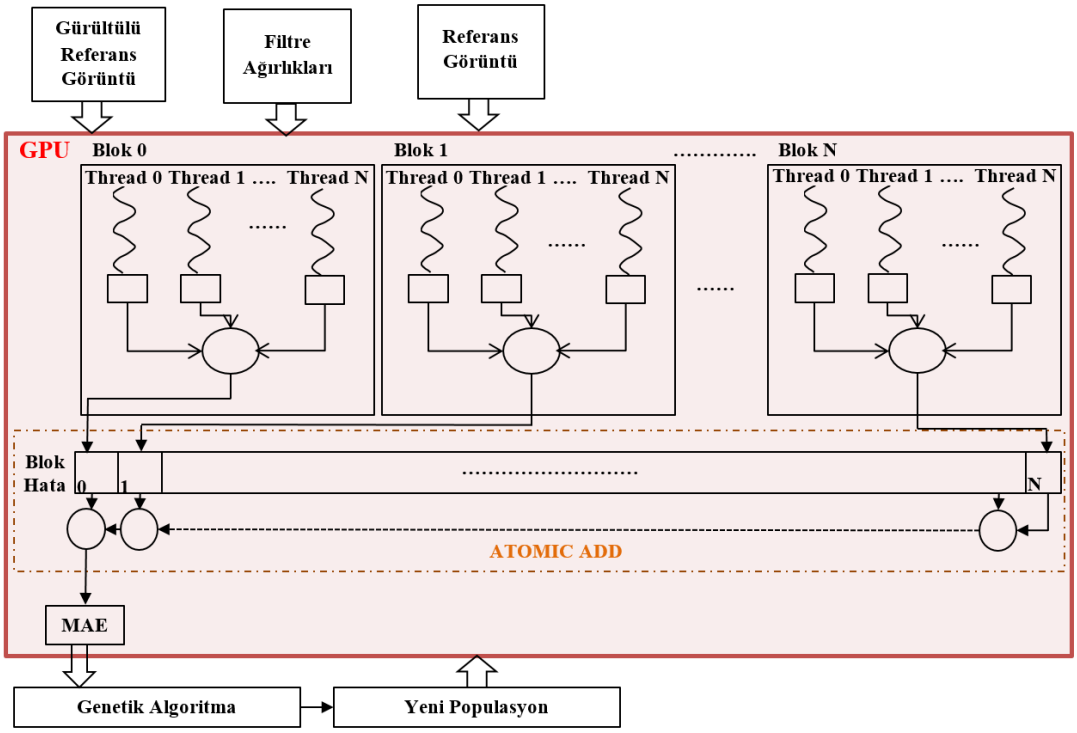
Şekil 4.10. GSU yöntemine ait algoritmanın sözde kodu

Bir CUDA iş parçacığı kullanılarak filtrelenecek olan piksel, *ThreadIDx.x* ve *BlockIDx.x* tarafından belirlenir ve popülasyon içerisindeki maske ağırlıklarına göre gürültü eklenmiş referans görüntü filtrelenir. Filtreleme işlemlerinden sonra hesaplanan MAE değerleri blok paylaşımlı bir dizi kullanılarak toplanır. Bu toplama işlemi CUDA C'ye ait *atomicAdd()* fonksiyonu ile gerçekleştirilmektedir. Bu toplama işlemi öncesinde hesaplama yapan bütün iş parçacıkları senkronize edilir. Eğer burada senkronize edilmezse iş parçacıkları ortak paylaşılan diziye rastgele şekilde değerler ekleyecekler ve sonuçta elde edilen MAE değerleri farklı olacaktır. Oldukça basit olan bu yöntemde iş parçacıkları çok daha hızlı olan yazmaç belleğini kullanmaz. Şekil 4.9.'dan da anlaşılacağa üzere popülasyonlar GPU içerisine tek tek gönderilmektedir. Bunun anlamı, bir popülasyon için filtreleme işlemi yapılarak uygunluk değeri hesaplanıyor, sonra diğer popülasyon GPU içerisine alınmaktadır. Örneğin test aşamasında popülasyon sayıları 200, 400 ve 800 olacak şekilde tasarlanmıştır. 200 popülasyon GPU içerisine tek tek alınmaktadır. Bunun anlamı her defasında ve sürekli CPU ile GPU arasında bir kopyalama işlemleri sürüp gitmektedir. Uygunluk fonksiyonu yüzlerce iş parçacığı üzerinde çok kısa sürede hesaplandığından CPU üzerinde gerçekleştirilen algoritmaya göre yine de çok hızlı çalışmaktadır. CPU ile GPU arasında yapılan sürekli kopyalama işlemleri de oldukça zaman almaktadır. İşte GPU içerisinde gerçekleştirilen farklı tasarımlar bu gibi zaman alan iş ve işlemlerin kısaltılmasıyla algoritmanın daha fazla hızlanması sağlanmaktadır. Görüntü kalitesi bakımında CPU üzerinde çalışan algoritma ile arasında çok büyük farklar bulunmamaktadır, ortaya çıkan küçük farkların sebebi ise, algoritma her çalışmasında gürültülü referans görüntüsünü rastgele oluşturması ve GA'ın başlangıçta rastgele popülasyonları oluşturmasından kaynaklanmaktadır.

4.4.2. GPU Popülasyon Temelli Yöntem

GPU Popülasyon Temelli (GPT) yönteminde, GA tarafından oluşturulan popülasyon toplu halde GPU içerisinde genel hafızaya alınmaktadır. Bu şekilde bir önceki anlatılan algoritmadan biraz daha hızlı çalışması planlanmıştır. Çünkü CPU ile GPU arasında sık yapılan bir işlemi doğrudan GPU içerisine taşınması hesaplama zamanını oldukça düşürmüştür. Şekil 4.11.'de GPT yöntemin çalışma şeması görülmektedir. Burada

gerçekleştirilen işlemler GSU yönteminde anlatıldığı gibi olmaktadır. GPT yönteminin GSU yönteminden farkı, hesaplama zamanını hızlandırmak amacıyla GA'ın oluşturduğu popülasyon toplu halde GPU içerisinde bulunan genel hafızaya alınmasıdır. Bu şekilde bir uygulamayla uygunluk değerinin hesaplanması sırasında sürekli olarak CPU'dan GPU'ya ve GPU'dan CPU'ya doğru yapılan kopyalama işlemleri azaltılarak deneysel sonuçlarla da ifade edildiği gibi, ciddi zaman kazançları elde edilmiştir.



Şekil 4.11. GPT yöntemi çalışma şeması

Bu yaklaşım, Şekil 4.12.'de GPT yönteminin sözde kodunda görüldüğü gibi daha önce anılan SU yönteminin GPU üzerinde gerçekleştirilmesinin üzerine popülasyon toplu halde genel hafızaya alınarak geliştirilmiştir. Her bir blok içerisinde blok uygunluk değeri hesaplanmaktadır ve bu hesaplanan değerler sonrasında GPU'ya ait olan *atomicAdd()* fonksiyonu ile toplanarak GA'a gönderilmektedir. GA bu uygunluk değerlerine göre yeni popülasyonlar oluşturmakta ve bu popülasyonlar tekrar toplu halde GPU içerisine alınarak bu iş ve işlemler maksimum tekrarlama sayısına ulaşıncaya kadar devam etmektedir. Yine bu hesaplama sonucunda görüntü kalitesini

etkileyen herhangi bir iş veya işlem yapılmadığından elde edilen görüntü kalitesi diğer yöntemlerde elde edilenler ile aynı seviyede elde edilir.

Algoritma 7: GPU POPÜLASYON TEMELLİ (GPT) YÖNTEM

Giriş: Referans Görüntü, Gürültü Eklenmiş Referans Görüntü, Popülasyon Bilgileri (Bütün Popülasyonun), Diğer Parametreler
Çıkış: MAE (Ortalama Mutlak Hata)

```
    Piksel ID'lerini Belirlemek için İş Parçacığı ID'leri Alınır.
PixelID = BlockDim.x * BlockIDx.x + ThreadIDx.x ;
    Geçerli Blok için Paylaşılan Bir Hata Dizisi Tanımlanır.
_Shared_ Float BlockShared[256 ];
    Maske ağırlığı kadar Görüntü Pikseli Yerel Belleğe Alınır.
R=Kopyala-R(Görüntü Piksel ID, N ;
G=Kopyala-G(Görüntü Piksel ID, N ;
B=Kopyala-B(Görüntü Piksel ID, N ;
    for (i = 0 to populasyonsayisi) do
      Seçilen Maske Ağırlıklarını Kullanarak Filtreleme Yapılır.
       Piksel Filtrele =  Filtre(Gürültü Eklenmiş Referans Görüntüsü, Maske Verileri, Pixel ID);
      Referans Görüntüyü Kullanarak MAE Hesaplanır.
       BlockShared(i) = abs(Filtrelenmiş Referans Görüntüsü - Referans Görüntü)/(3*Popülasyon Sayısı;
      Hesaplanan MAE'ler için İş Parçacıkları Senkronize Edilir.
       _Syncthreads();
      Blok Paylaşılan Hata Dizisinin MAE'leri Hesaplanır.
       BlockSum =  ParalellSum(BlockShared);
      Bütün Gürüntünün MAE'sini Hesaplamak için Paylaşılan Bloklar Toplanır.
       AtomicAdd( δ MAE (i), BlockShared /(256*3));
      Hesaplanan MAE'ler için İş Parçacıkları Senkronize Edilir.
       _Syncthreads();
Return MAE ;
```

Şekil 4.12. GPT yöntemi sözde kodu

4.5. Görüntü Kalitesi Performans Metrikleri

Bu çalışmada, görüntü filtreleme işlemlerinde dört farklı görüntü kalitesi ölçüm tekniği kullanılmıştır. Bu teknikler; MAE, MSE, PSNR ve SSIM şeklindedir [96]–[98]. MAE ve MSE değerleri ne kadar küçük olursa görüntü filtreleme işlemlerinin o derece başarılı olduğu kabul edilmektedir. Aşağıdaki denklemlerde x filtrelenmiş referans görüntüyü y referans görüntüyü, N ve M ise görüntülere ait boyut bilgilerini göstermektedir. Genel olarak MAE değerinin hesaplanmasında aşağıdaki eşitlik (Denklem 4.7) kullanılmaktadır.

$$MAE_{xy} = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|x_{ij} - y_{ij}\right| \tag{4.7}$$

Yukarıdaki eşitlikte (Denklem 4.7) x_{ij} referans görüntüyü, y_{ij} ise filtrelenmiş görüntüyü temsil etmektedir. Aşağıda (Denklem 4.8) MSE değerinin hesaplanmasında kullanılan denklem görülmektedir.

$$MSE_{xy} = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[x_{ij} - y_{ij}\right]^2 \tag{4.8}$$

PSNR' in yüksek değerleri görüntünün iyi filtrelenmiş olduğunu ve gürültü seviyesinin çok daha düşük olduğunu göstermektedir [99]. Aşağıda (Denklem 4.9) PSNR değerinin hesaplanmasında kullanılan denklem görülmektedir [100].

$$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{4.9}$$

Denklemdeki $\left(\frac{MAX^2}{MSE}\right)$'deki kare ifadesi yok edildiğinde denklem aşağıda (Denklem 4.10) gösterildiği gibi olmaktadır;

$$PSNR = 20\log_{10}\left(\frac{MAX}{\sqrt{MSE}}\right) \tag{4.10}$$

Yukarıdaki denklemde (Denklem 4.10) gösterilen $\left(\frac{MAX}{\sqrt{MSE}}\right)$ içerisine logaritma dağıtılırsa denklem aşağıda (Denklem 4.11) gösterildiği gibi olmaktadır;

$$PSNR = 20\log_{10}\left(MAX\right) - 10\log_{10}(MSE) \tag{4.11}$$

Yapılan deneysel çalışmalarda kullanılan renkli görüntüleri oluşturan R, G ve B bileşenleri 8 bit ile ifade edildiği için *MAX* değeri 255 olarak alınmıştır. Genel olarak PSNR ve MSE, MAE arasında ters orantı bulunmaktadır. Görüntü üzerindeki gürültü

miktarı azaldıkça MAE ve MSE değerleri düşerken PSNR değerleri de o denli yükselmektedir [99]. SSIM parametresi, yerel görüntü yapısını, parlaklığı ve kontrastı tek bir yerel kalite puanıyla birleştirir. Yapısal benzerlik indeksi olan SSIM parametresi, sayısal görüntüleri veya videoların kalitesini tahmin etmek için görüntü işleme alanında sıklıkla kullanılan bir parametredir [101]. Aşağıdaki denklemlerde (Denklem 4.12 ve Denklem 4.13) yapısal benzerlik indeksi SSIM parametresinin hesaplanmasında kullanılan denklemler görülmektedir. Bu denklemlerde belirtilen, I_{xy} parlaklık, C_{xy} karşıtlık ve S_{xy} yapı değeri parametresini göstermektedir. Parlaklık, karşıtlık ve yapı değeri parametrelerinin ağırlıklarını ise α, γ, ve β göstermektedir. Referans görüntü ve filtrelenmiş görüntülere ait kovaryans ise σ_{xy} ile gösterilmektedir. Referans görüntüsüne ait standart sapma σ_x ile filtrelenmiş görüntüsüne ait standart sapma ise σ_y ile gösterilmiştir. Bölme işlemlerinde kullanılan c_1, c_2 ve c_3 ise zayıf parametreleri dengelemek için kullanılmaktadır [102]. Zayıf parametreleri dengelemek için kullanılan $c_1=(K_1L)^2$, $c_2=(K_2L)^2$ ve $c_3= c_2/2$ şeklinde tanımlanmıştır [102], [103].

$$I_{xy} = \frac{2\mu_x\mu_y+c_1}{\mu_x{}^2+\mu_y{}^2+c_1} \quad C_{xy} = \frac{2\sigma_x\sigma_y+c_2}{\sigma_x{}^2+\sigma_y{}^2+c_2} \quad S_{xy} = \frac{\sigma_{xy}+c_3}{\sigma_x+\sigma_y+c_3} \tag{4.12}$$

$$SSIM_{xy} = \left[I_{xy}{}^{\alpha} \cdot C_{xy}{}^{\beta} \cdot S_{xy}{}^{\gamma}\right] \quad SSIM_{xy} = \frac{(2\mu_x \ \ y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x{}^2+\mu_y{}^2+c_1)(\sigma_x{}^2+\sigma_y{}^2+c_2)} \tag{4.13}$$

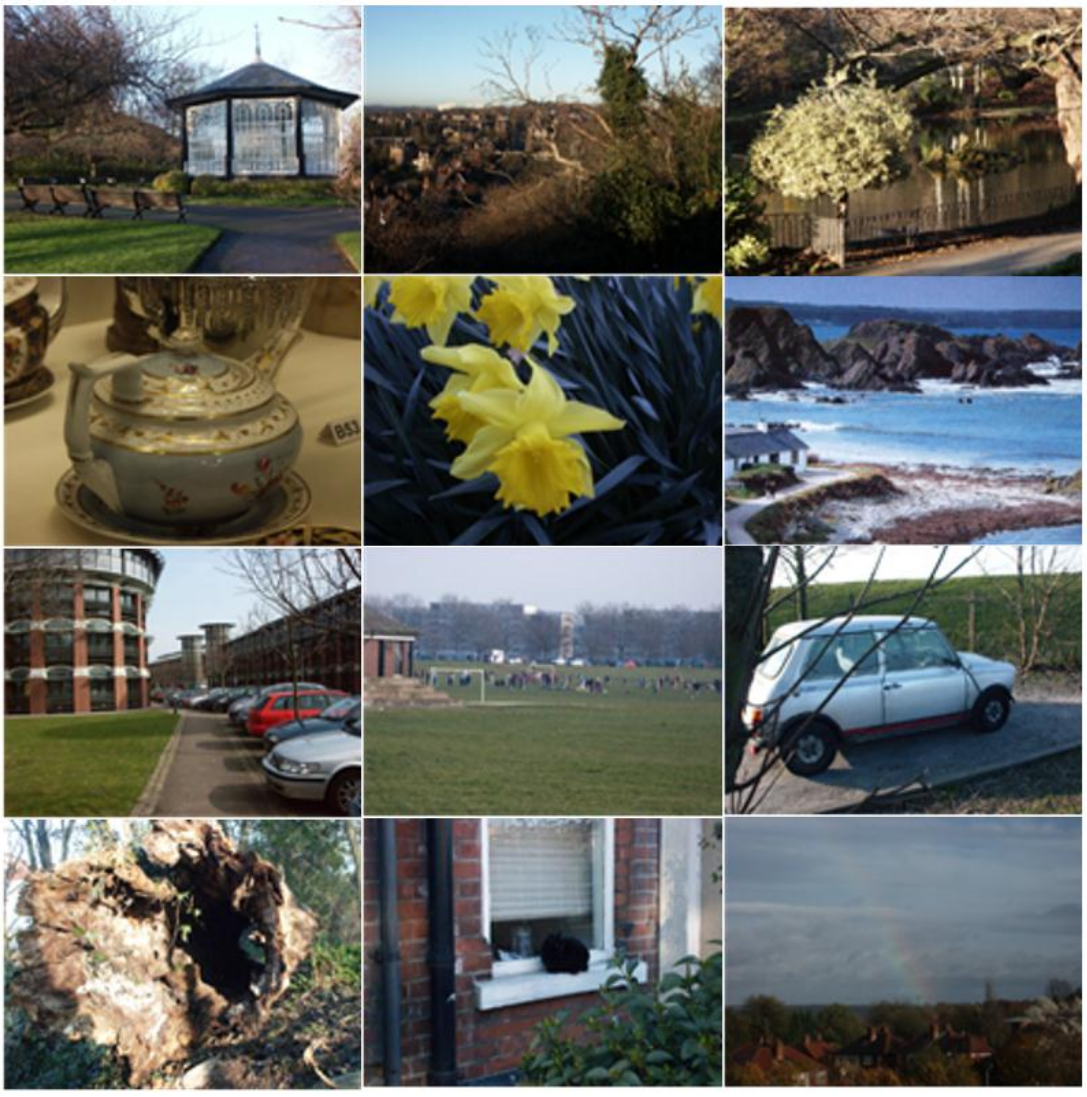
4.6. Sonuç

Bu bölümde, deneysel çalışmalarda önerilen yöntemlerin hızlandırma performanslarını ölçmek için referans almak amacıyla kullanılacak SU yöntemi geliştirilmiştir. Bu yöntem CPU üzerinde sıralı bir şekilde çalışmaktadır. Ayrıca Kuadratik görüntü filtrelerin maske ağırlıklarının eğitim sürelerini kısaltmak amacıyla DU ve RDU algoritmik hızlandırma yöntemleri ile GSU ve GPT donanımsal hızlandırma yöntemleri detaylı bir şekilde anlatılmıştır. Donanımsal hızlandırma yöntemlerinde, uygunluk fonksiyonu GPU kart üzerinde binlerce CUDA çekirdeği tarafından paralel olarak çalışacak şekilde tasarlanmıştır. Bu bölümde önerilen

yöntemlerin ciddi anlamda hızlandırma yapması beklenmektedir. Deneylerde kullanılan ve görüntü kalitesi ölçüm teknikleri bu bölümde anlatılmıştır.

BÖLÜM 5. DENEYSEL SONUÇLAR

Yapılan deneylerde, Şekil 5.1.'de gösterilen 512×384 piksel boyutlarına sahip 12 farklı test görüntüsü kullanılmıştır [95].



Şekil 5.1. Deneysel çalışmalarda kullanılan referans görüntüler

Her bir görüntü için geliştirilen algoritma istatistikte de önerilen minimum sayı [104] olarak 30 kez çalıştırılmış ve elde edilen sonuçların ortalaması kaydedilmiştir. Deneysel çalışmalarda MAE ve MSE görüntü kalitesi ölçüm teknikleri kullanılmıştır. Ayrıca eğitim için kullanılan GA ve PSO'da farklı popülasyon sayısı ve tekrarlama sayıları kullanılmış ve eğitim süreleri ölçülerek kaydedilmiştir. GA'ın parametreleri; mutasyon oranı 0,1, çaprazlama oranı 0,5, Kuadratik filtrelerin doğrusal kısımlarının üst ve alt sınırları 1,0 ile -1,0 arasında, Kuadratik filtrenin kuadratik kısımlarının üst ve alt sınırları 0,5 ile -0,5 arasında belirlenmiştir. PSO'nun öğrenme katsayıları olan c_1 ve c_2 2,0, r_1 değişkeni 0,0, r_2 değişkeni 1,0, maksimum hız değeri (Vmax) 0,02, Kuadratik filtrelerin doğrusal kısımlarının üst ve alt sınırları 1,0 ile -1,0 arasında, Kuadratik filtrenin kuadratik kısımlarının üst ve alt sınırları 0,5 ile -0,5 arasında belirlenmiştir.

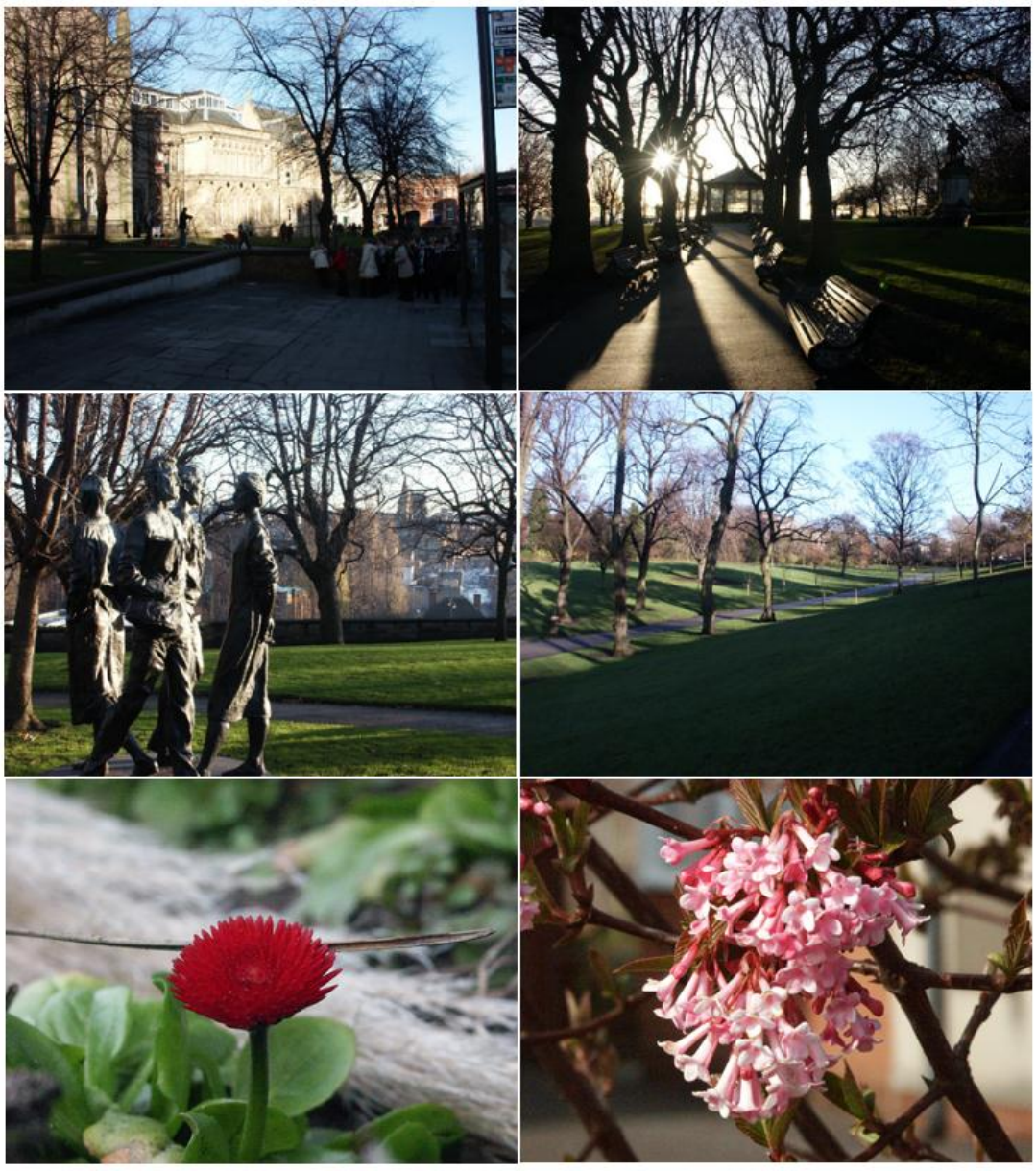
Tez çalışması kapsamında deneysel çalışmalara başlamadan önce doğrusal olmayan Kuadratik görüntü filtrelerinin, doğrusal olmayan gürültülü görüntülerin filtrelenmesi konusunda Gaussian filtrelerle karşılaştırması yapılmıştır. Bu karşılaştırma işlemlerinde kullanılacak olan Gaussian filtre maske ağırlıkları, literatürde tavsiye edilen Gauss katsayıları da içerisinde olacak şekilde 0,7; 0,8; 0,9; 1,0; 1,1 ve 1,2 seçilmiş ve Tablo 5.1.'de bu katsayılar kullanıldığında elde edilen maske ağırlıkları verilmiştir.

Tablo 5.1. Gauss katsayılarına göre elde edilen Gausssian filtre maske ağırlıkları

Sıra No	**Gauss Katsayıları**							
	0,5	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**	**1,1**	**1,2**
1	0,011	0,027	0,043	0,057	0,067	0,075	0,081	0,085
2	0,083	0,111	0,121	0,124	0,124	0,123	0,122	0,121
3	0,011	0,027	0,043	0,057	0,067	0,075	0,081	0,085
4	0,083	0,111	0,121	0,124	0,124	0,123	0,122	0,121
5	0,619	0,445	0,337	0,272	0,231	0,204	0,185	0,171
6	0,083	0,111	0,121	0,124	0,124	0,123	0,122	0,121
7	0,011	0,027	0,043	0,057	0,067	0,075	0,081	0,085
8	0,083	0,111	0,121	0,124	0,124	0,123	0,122	0,121
9	0,011	0,027	0,043	0,057	0,067	0,075	0,081	0,085

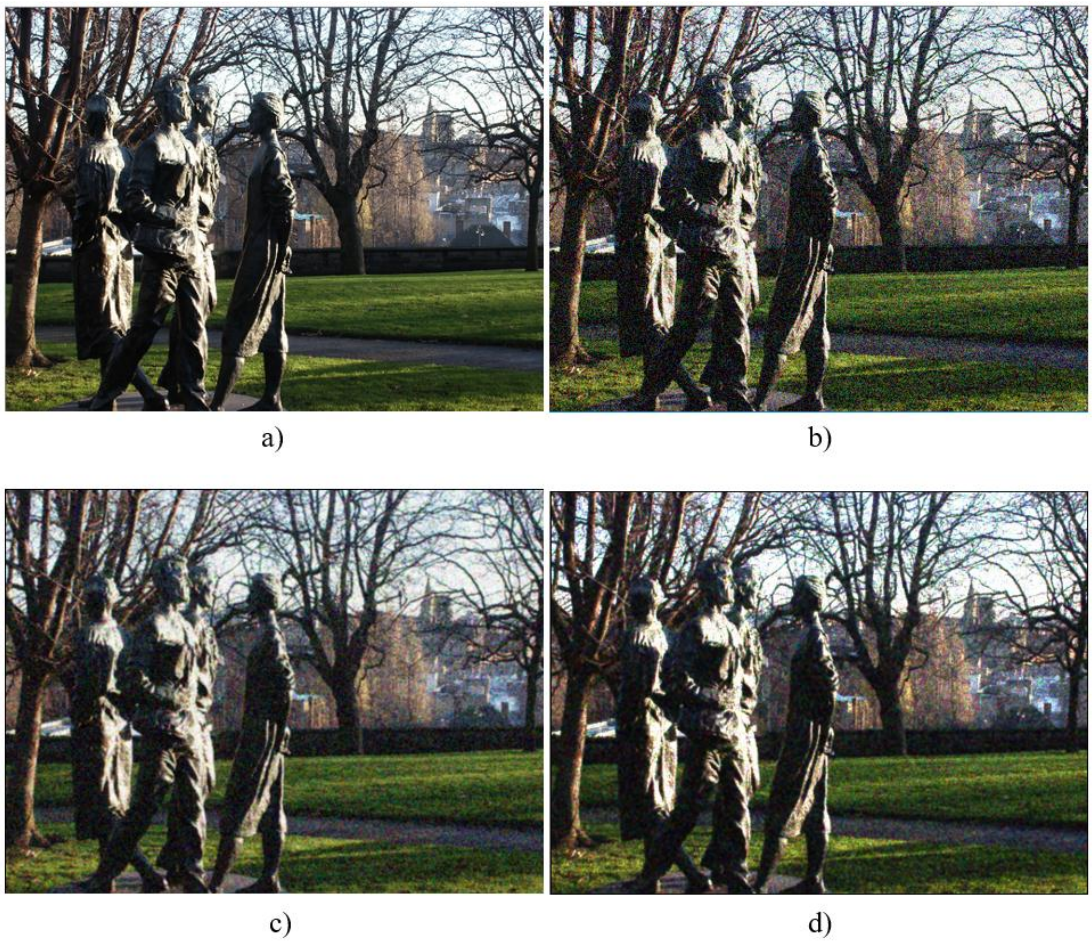
0,1 – 0,4 aralığı ile 1,3 – 2,0 aralıklarındaki katsayılar ile elde edilen Gaussian filtre maske ağırlıkları burada kullanılan maske ağırlıklarına göre çok daha kötü sonuçlar

verdikleri için kullanılmamıştır. Karşılaştırmada kullanılan görüntüler 512×384 piksel boyutlarına sahip Şekil 5.2.'de gösterilen 6 farklı referans görüntü kullanılmıştır [95]. Bu referans görüntüler üzerine Gauss gürültüsü eklenerek gürültülü referans görüntüler elde edilmiştir. Bu gürültülü referans görüntüler hem Gaussian filtresi ile hem de Kuadratik görüntü filtresi ile filtrelenerek MAE görüntü kalite değerleri kaydedilmiştir.



Şekil 5.2. Kuadratik görüntü filtreleri ile Gaussian filtrelerin karşılaştırılmasında kullanılan referans görüntüler.

Örnek olarak Şekil 5.3.'de gösterilen bir görüntünün Kuadratik görüntü filtreleri ve Gaussian filtre kullanılarak elde edilen filtreleme sonucundaki görüntü kalitesi başarım sonuçları görülmektedir.



Şekil 5.3. Kuadratik görüntü filtresi ve Gaussian filtre kullanılarak filtrelenmiş örnek görüntüler a) Referans görüntü, b) Gürültü eklenmiş referans görüntü (MAE: 23,15), c) Gaussian filtre ile filtrelenmiş görüntü (MAE: 15,49), d) Kuadratik görüntü filtresi ile filtrelenmiş görüntü (MAE: 13,38).

Şekil 5.3a.'da referans görüntü, Şekil 5.3b.'de referans görüntüye Gauss gürültüsü eklenmiş görüntü, Şekil 5.3c.'de Gaussian filtre ile filtrelenmiş görüntü, Şekil 5.3d.'de Kuadratik görüntü filtresi ile filtrelenmiş görüntü görülmektedir. Gaussian filtresi kullanılarak MAE değeri 23,15 olan gürültü eklenmiş referans görüntüsünün filtrelenmesi sonucunda MAE görüntü kalite değerini 15,49'a düşürmüştür. Kuadratik görüntü filtresi kullanılarak aynı MAE değerine sahip gürültü eklenmiş referans görüntüsünün filtrelenmesi sonucunda MAE görüntü kalite değerinin 13,38'e düşürüldüğü görülmektedir.

Tablo 5.2.'de Gaussian filtresi ile farklı Gauss katsayılarıyla elde edilmiş maske ağırlıkları kullanılarak elde edilen MAE görüntü kalitesi sonuçları görülmektedir. Tablodaki değerlere bakıldığında farklı görüntülerde farklı Gauss ağırlığı ile elde edilen maske ağırlıkları daha iyi sonuçlar vermektedir. Örneğin test görüntü-1, test görüntü-3 ve test görüntü-4 için 0,7 Gauss ağırlığı ile elde edilen maske ağırlıkları ile filtreleme sonucu en iyi çıkarken test görüntü-2 için 0,6 Gauss ağırlığı ile elde edilen maske ağırlıkları ile filtreleme sonucu en iyi çıkmakta, test görüntü-5 ve test görüntü-6 için ise 1,1 Gauss ağırlığı ile elde edilen maske ağırlıkları ile filtreleme sonucu en iyi çıkmaktadır. Bunun sebebi ise kullanılan referans görüntünün özelliği ile ilgilidir.

Tablo 5.2. Test görüntülerinin Gaussian filtre ile filtrelendikten sonra elde edilen MAE değerleri

Gauss Katsayıları	Test Görüntü-1	Test Görüntü-2	Test Görüntü-3	Test Görüntü-4	Test Görüntü-5	Test Görüntü-6
0,5	16,11	15,91	16,94	15,80	16,55	15,98
0,6	14,43	**15,35**	15,53	13,69	12,90	12,84
0,7	**14,03**	15,62	**15,40**	**12,99**	11,07	11,38
0,8	14,09	16,02	15,64	12,83	10,20	10,77
0,9	14,24	16,37	15,92	12,84	9,79	10,52
1,0	14,39	16,65	16,16	12,90	9,57	10,41
1,1	14,53	16,87	16,36	12,96	**9,31**	**10,32**
1,2	14,67	17,05	16,54	13,04	9,39	10,35

Gaussian filtrelerde elde edilen maske ağırlıkları belirli oldukları için farklı özelliklere sahip görüntüler üzerinden gürültü temizleme işlemlerinde başarılı daha düşük başarım sağlayabilirler. Maske ağırlıklarının sezgisel algoritmalarla eğitilerek belirlendiği durumlarda görüntünün özelliğine göre maske ağırlıkları hesaplandığından çok daha iyi sonuçlar verebilmektedir. Bunu göstermek için aynı referans görüntüler ve aynı gürültülü referans görüntüler kullanılarak Kuadratik görüntü filtreleri ile de filtrelenmiştir.

Tablo 5.3.'de Gaussian filtre ile elde edilen en iyi MAE değerleri ile Kuadratik görüntü filtrelerinin MAE ve MSE ile eğitimleri sonucunda elde edilen MAE değerlerinin karşılaştırması görülmektedir. Kuadratik görüntü filtrelerinin filtre katsayıları GA ile elde edilmiştir. Ayrıca Kuadratik görüntü filtrelerin GA ile eğitimi esnasında hem MAE hem de MSE kullanılarak MAE değerleri üzerinde görüntü kalitesi ölçüm

tekniklerinin etkisi de gözlemlenmiştir. Genel olarak Kuadratik görüntü filtrelerinin Gaussian filtrelerine göre daha iyi sonuç verdiği görülmektedir. Test görüntü-1, test görüntü-2, test görüntü-3 ve test görüntü-4'deki görüntülere dikkat edilecek olursa görüntü üzerinde ayrıntılar çok daha fazladır. Bundan dolayı Kuadratik görüntü filtresinin görüntü filtre başarımı sırasıyla 13,10; 12,64; 13,30 ve 11,41 olurken Gaussian filtrenin başarımı ise sırasıyla 14,03; 15,35; 15,40 ve 12,99 olmaktadır. Test görüntü-5 ve test görüntü-6'da görülen görüntülerde ayrıntılar biraz daha az olduğundan Kuadratik görüntü filtresi ile Gaussian filtresinin başarımları bir birlerine çok daha yakın oldukları görülmektedir.

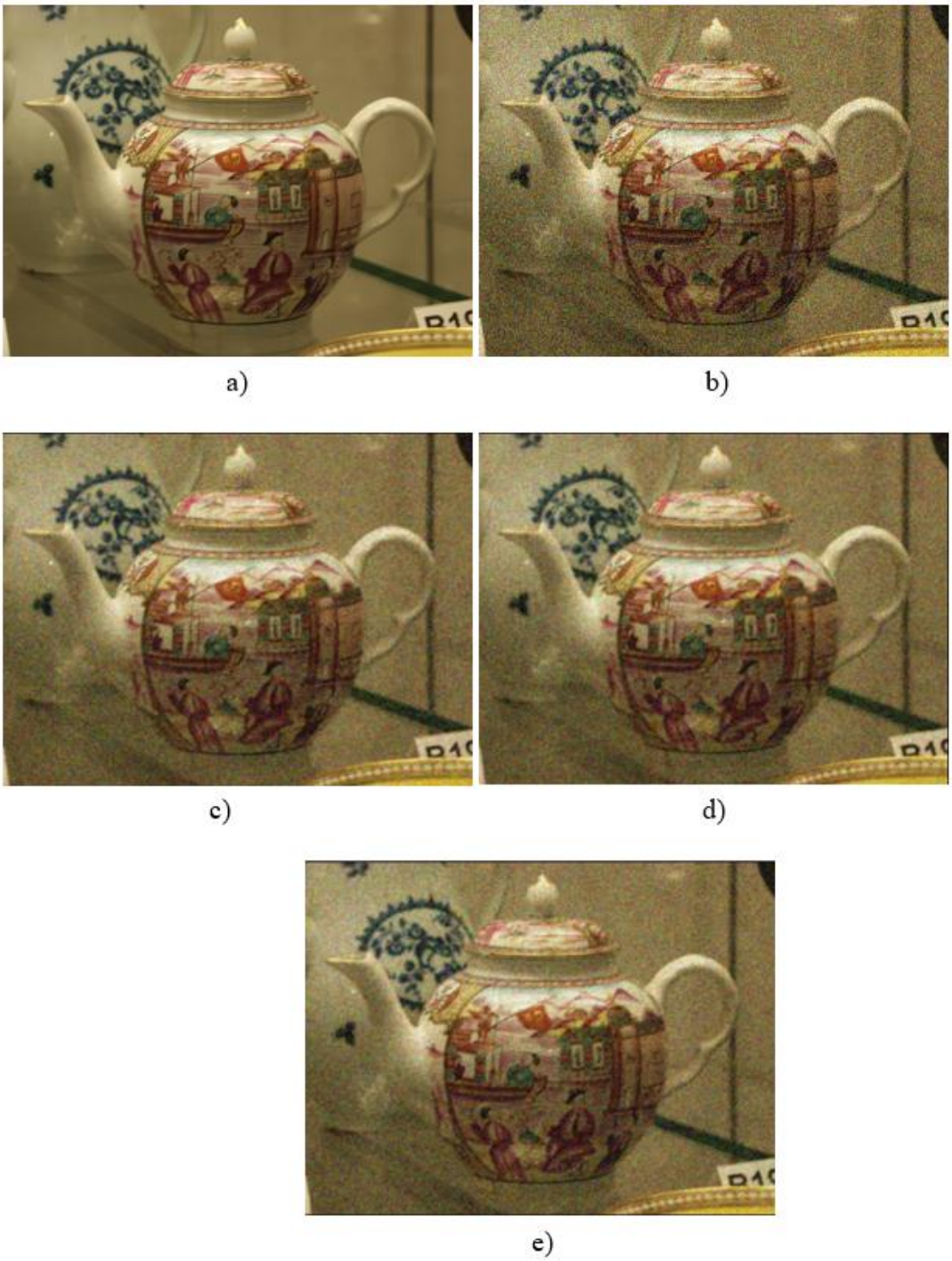
Tablo 5.3. Kuadratik görüntü filtreleri ile Gaussian filtrelerinin MAE görüntü kalitesi bakımından karşılaştırılması

Test Görüntüleri	**Gaussian Filtre**	**Kuadratik Filtre**	
		MAE Uygunluk Değeri	**MSE Uygunluk Değeri**
Test Görüntü-1	14,03	**13,10**	13,34
Test Görüntü-2	15,35	**12,64**	12,96
Test Görüntü-3	15,40	**13,30**	13,42
Test Görüntü-4	12,99	**11,41**	11,53
Test Görüntü-5	9,31	**9,29**	9,41
Test Görüntü-6	10,32	**10,17**	10,34

Kuadratik görüntü filtrelerinin GA eğitimlerinde uygunluk değeri MAE ile elde edildiğinde MSE'ye göre daha iyi sonuçlar verdiği de görülmektedir. Örneğin test görüntü-1 için eğitimin MAE ile yapılması sonucunda hesaplanan MAE değeri 13,10 iken eğitimin MSE ile yapılması sonucu hesaplanan MAE değeri 13,34 olmaktadır. MAE dışında MSE, PSNR ve SSIM görüntü kalitesi ölçüm teknikleri de eğitimlerde kullanılmış olup genel olarak MAE ile daha iyi sonuçlar elde edilmiştir. Bu sebeplerden dolayı bu tez çalışması kapsamında yapılan GA ve PSO eğitimlerinde MAE görüntü kalitesi kullanılmıştır.

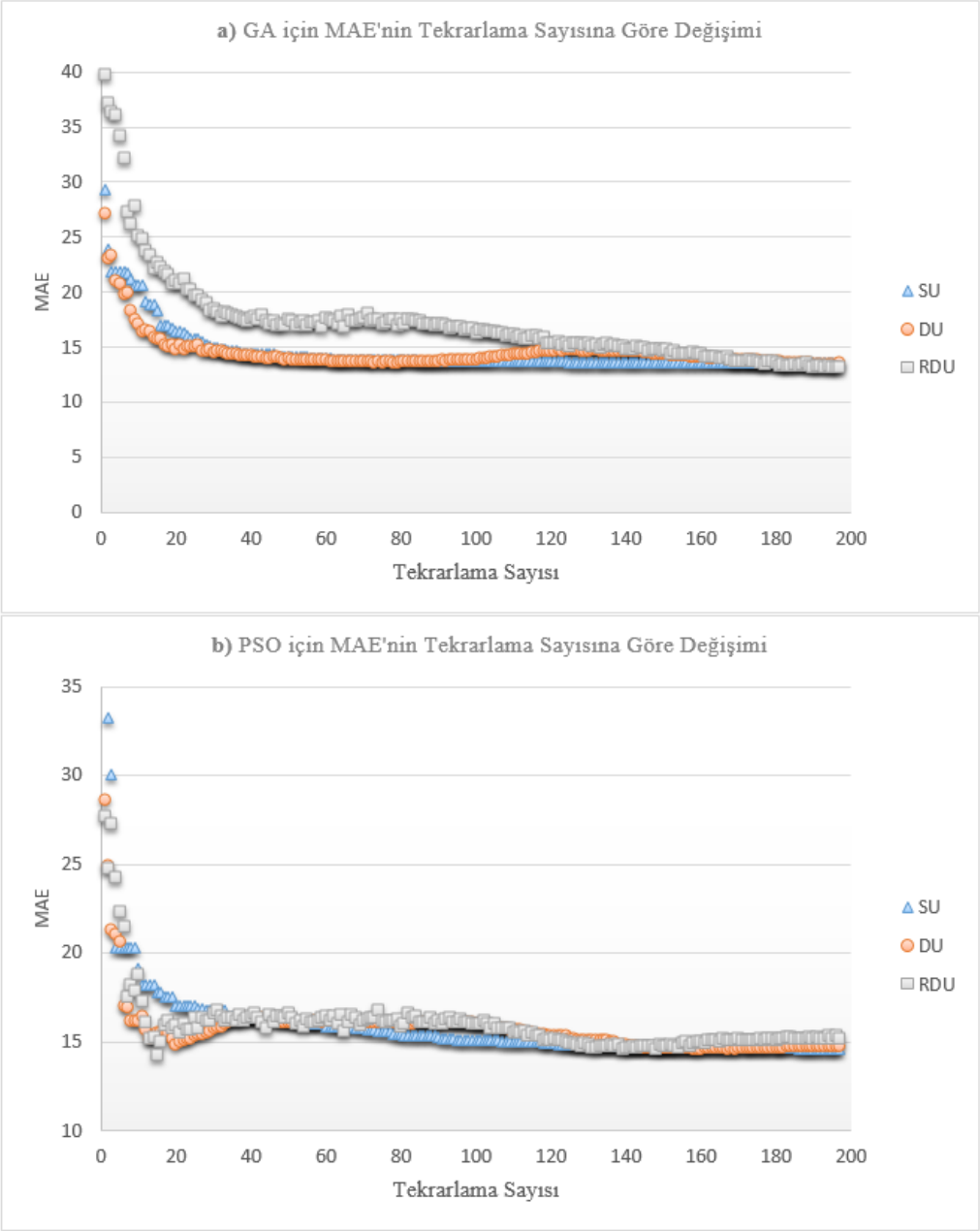
Yapılan ilk deneysel çalışmada algoritmik hızlandırma için geliştirilen DU ve RDU yöntemlerinin performans analizleri incelenecektir. Bu deneysel çalışma, Quad-Core AMD Opteron™ 2378 2.40GHz çift işlemci, 18GB Ram bellek ve Windows Server 2012 Essential™ 64-bit işletim sistemine sahip bir sunucuda elde edilmiştir. Algoritmalar Visual Studio 2012 ™ platformunda C++ dili kullanılarak geliştirilmiştir. Örnek olarak Şekil 5.4.'de gösterilen bir görüntünün SU, DU ve RDU

yöntemleri kullanılarak elde edilen filtreleme sonucundaki görüntü kalitesi başarım sonuçları görülmektedir.



Şekil 5.4. SU, DU ve RDU yöntemleri kullanılarak filtrelenmiş örnek görüntüler (Popülasyon sayısı 200, tekrarlama sayısı 1000 olarak belirlenmiştir) a) Referans görüntü, b) Gürültü eklenmiş referans görüntü (MAE: 23,02), c) SU yöntemi ile filtrelenmiş görüntü (MAE: 8,61), d) DU yöntemi ile filtrelenmiş görüntü (MAE: 8,69), e) RDU yöntemi ile filtrelenmiş görüntü (MAE: 8,72)

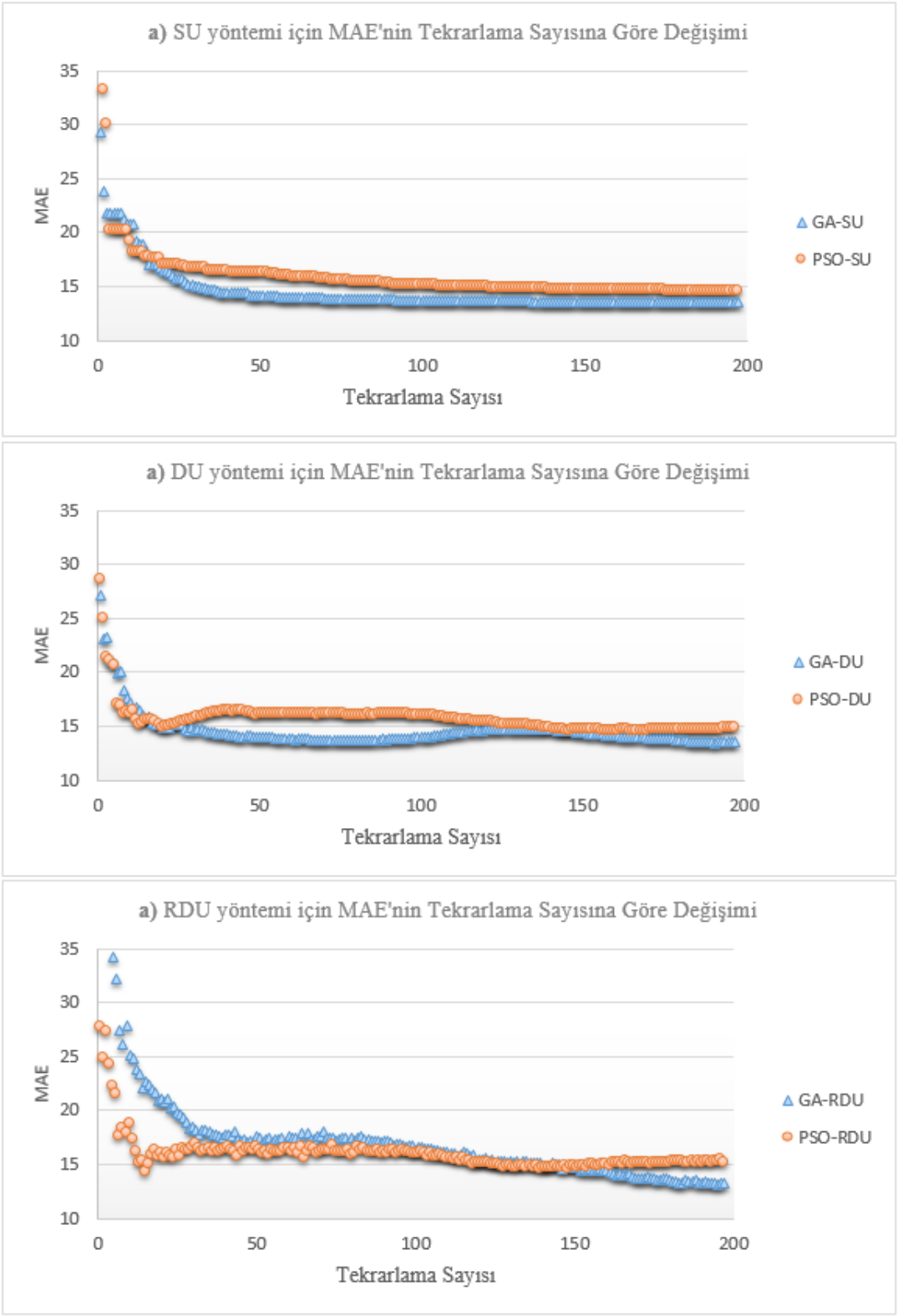
Şekil 5.4a.'da referans görüntü, Şekil 5.4b.'de referans görüntüye Gaussian gürültüsü eklenmiş görüntü, Şekil 5.4c.'de SU yöntemi ile filtrelenmiş görüntü, Şekil 5.4d.'de DU yöntemi ile filtrelenmiş görüntü ve Şekil 5.4e.'de RDU yöntemi ile filtrelenmiş görüntü görülmektedir. Elde edilen görüntü kalite sonuçları birbirlerine oldukça yakın olmakla birlikte DU ve RDU ile filtrelenmiş görüntülerin eğitim süreleri SU'ya göre oldukça düşük olmaktadır.



Şekil 5.5. GA ve PSO kullanılarak SU, DU ve RDU için tekrarlama sayılarına karşılık MAE değerleri grafiği

Şekil 5.5.'de GA ve PSO sezgisel algoritmaları kullanılarak SU, DU ve RDU yöntemlerinin tekrarlama sayısına bağlı MAE değerleri grafikleri görülmektedir. Test edilen SU, DU ve RDU yöntemleri için MAE değerlerinin yakınsaması grafiğin ölçeklenmesi nedeniyle yakın görülmektedir. Şekil 5.5a.'da GA'ın Şekil 5.5b.'de PSO'nun MAE değerine yakınsama grafikleri görülmektedir. Grafiklerde görülen değerler sezgisel algoritmaların popülasyon sayısı ve tekrarlama sayısı parametre değerleri 200 olacak şekilde belirlenerek elde edilmiştir. GA kullanıldığında SU ve DU yöntemleri RDU yöntemine göre daha erken iyi MAE değerlerine ulaşmaktadır. PSO kullanıldığında ise DU ve RDU yöntemleri SU yöntemine göre daha erken iyi MAE değerlerine ulaştıkları görülmektedir. RDU yöntemi PSO kullanıldığında çok daha erken iyi MAE değerlerine ulaştığı görülmektedir.

Şekil 5.6.'da SU, DU ve RDU yöntemlerinin farklı sezgisel algoritmalar kullanıldığında tekrarlama sayılarına göre MAE değerleri değişim grafiği görülmektedir. Şekil 5.6a.'da SU yönteminin farklı sezgisel algoritmalar kullanıldığında tekrarlama sayısına bağlı MAE değerleri grafiği görülmektedir. SU yönteminde GA'ın kullanılmasıyla daha iyi MAE değerlerine PSO'nun kullanılmasına göre çok daha erken ulaşmaktadır. Şekil 5.6b.'de DU yönteminin farklı sezgisel algoritmalar kullanıldığında tekrarlama sayısına bağlı MAE değerleri grafiği görülmektedir. DU yönteminin GA ile kullanılmasının PSO'ya göre çok daha iyi sonuçlar verdiği görülmektedir. Şekil 5.6c.'de RDU yönteminin farklı sezgisel algoritmalar kullanıldığında tekrarlama sayısına bağlı MAE değerleri grafiği görülmektedir. RDU yönteminde diğer yöntemlerden farklı olarak PSO ile kullanılmasıyla daha iyi MAE değerlerine GA ile kullanıma göre çok daha erken ulaştığı görülmektedir. Bu grafiklere göre SU ve DU yöntemlerinin GA ile kullanımlarında daha iyi MAE değerlerine erken ulaşılırken RDU yönteminde ise PSO ile kullanımda daha iyi MAE değerlerine erken ulaşılmıştır.



Şekil 5.6. SU, DU ve RDU yöntemlerinin GA ve PSO kullanılarak elde edilen tekrarlama sayılarına karşılık MAE değerleri grafiği

Tablo 5.4.'de SU, DU ve RDU yöntemleri için ardışık çalışmaların görüntü kalite değişimlerini daha yakından kontrol etmek için örnek 10 çalışmanın sayısal değerlerini göstermektedir. Tabloda genelde sonuçlar bir birine oldukça yakındır.

Tablo 5.4. GA ve PSO'dan test çalışmalarından alınan MAE sonuçları

	GA			**PSO**		
	SU	**DU**	**RDU**	**SU**	**DU**	**RDU**
Çalışma 1	**8,54**	8,57	8,90	**8,87**	9,38	9,94
Çalışma 2	**8,52**	8,53	8,73	**8,67**	9,30	10,31
Çalışma 3	8,55	**8,52**	8,59	**8,62**	10,03	9,43
Çalışma 4	**8,51**	8,53	8,63	**9,00**	9,15	10,47
Çalışma 5	**8,54**	8,56	8,57	**8,70**	9,09	9,10
Çalışma 6	**8,52**	**8,52**	8,61	**8,76**	9,37	9,15
Çalışma 7	**8,54**	**8,54**	8,92	**8,64**	9,25	9,16
Çalışma 8	**8,53**	8,55	8,80	**8,73**	9,48	9,17
Çalışma 9	**8,53**	8,60	8,58	**8,81**	9,09	9,56
Çalışma 10	**8,51**	8,53	8,62	**8,81**	9,65	9,73
Ortalama	**8,53**	8,55	8,70	**8,76**	9,38	9,60
Ortalama Sapma	**0,01**	0,02	0,11	**0,09**	0,20	0,41
Ortalama Sapma %	1,13	**0,23**	1,31	**1,01**	2,18	4,5
Minimum	**8,51**	8,52	8,57	**8,62**	9,09	9,10
Maksimum	**8,54**	8,60	8,90	**9,00**	10,03	10,47

GA ile elde edilen ortalama görüntü kalitesi farklı çalışmalarda çok tutarlı sonuçlar göstermektedir. Ayrıca SU, DU ve RDU yöntemleri için elde edilen MAE görüntü kalitesi değerleri de birbirlerine çok yakındır. GA için ortalama sapma en fazla %1,13'dür ve bu da mevcut durum için uygun bir değerdir. Ancak bu davranış, kullanılan referans görüntüsüne, tekrarlama sayısına, popülasyon sayısına ve algoritmada kullanılan parametrelerin değerine göre değişiklik gösterebilir. Ardışık çalışmalar için PSO davranışı SU ile yapılan test çalışmaları DU ve RDU ile karşılaştırıldığında oldukça tutarlı olduğu görülmektedir. Genelde DU ve RDU yöntemlerinin PSO ile kullanımlarında minimum ve maksimum değerler arasındaki boşluk araştırıldığında SU'ya yakın sonuçlar üretmediği görülmektedir.

Tablo 5.5., Tablo 5.6. ve Tablo 5.7.'de sırasıyla SU, DU ve RDU kullanılarak hesaplanan görüntü kalite ölçümleri gösterilmektedir. Ayrıca her tabloda PSO ve GA için SU, DU ve RDU karşılaştırmaları verilmiştir. Sonuçlar tekrarlama sayısı 100, 200 ve 500 olacak şekilde belirlenmiştir. Her bir ölçüm için popülasyon sayısı 500'e

sabitlenmiştir. 100 tekrarlama sayısı için MAE değerleri karşılaştırıldığında SU ve DU sonuçlarının biraz daha iyi sonuç verdikleri görülmektedir. Aynı davranış 200 ve 500 tekrarlama sayıları için de gözlemlenmiştir. Bunun sebebi ise uygunluk değeri hesaplamaları için daha küçük bölgeleri seçen DU'nun hesaplama yaklaşımından kaynaklanmaktadır. Sayısal sonuçlara göre, tekrarlama sayısının arttırılması, test edilen tüm yaklaşımların sonuçlarını iyileştirdiği gözlemlenmiştir. Tekrarlama sayısı 500'e çıkartıldığında elde edilen MAE sonuçlarının birbirlerine yaklaştığı görülmüştür. Sonuçların seçilen algoritma parametrelerine çok duyarlı olduğu ve uygulanan algoritmaların popülasyon sayısına ve tekrarlama sayısının önemli ölçüde değişebileceğine dikkat edilmelidir. Uygulanan yöntemlerin hesaplama zamanları ise Tablo 5.8., Tablo 5.9. ve Tablo 5.10.'da sırasıyla SU, DU ve RDU yöntemleri GA ve PSO ile eğitildiklerinde elde edilen eğitim sürelerini göstermektedir. Tekrarlama sayılarının artışı eğitim sürelerini de bütün algoritmalar için arttırdığı görülmektedir. Popülasyon sayısı 500 olacak şekilde sabit tutulduğunda 100 tekrarlama sayısı için SU yönteminde ortalama eğitim süresi 44,87 dakika olurken, DU yönteminde bu süre 16,50 dakika olmakta, RDU yönteminde ise bu süre 17,23 dakika olmaktadır. Tekrarlama sayısı 200'e çıkartıldığında SU yönteminde ortalama eğitim süresi 93,04 dakika olurken, DU yönteminde bu süre 34,16 dakika olmakta, RDU yönteminde ise 34,90 dakika olmaktadır. Tekrarlama sayısı 500'e çıkartıldığında SU yönteminde ortalama eğitim süresi 226,07 dakika olurken, DU yönteminde bu süre 86,43 dakika olmakta, RDU yönteminde ise 87,48 dakika olmaktadır. Bu sonuçlara göre önerilen algoritmaların SU yöntemine göre çok daha kısa sürelerde eğitimlerini bitirdikleri görülmektedir. Kuadratik görüntü filtreleri gibi ağır filtrelerin, filtreleme eğitimleri sonunda görüntü kalitesi korunurken eğitim sürelerinin azaltılması hedefine ulaşıldığı görülmektedir.

Tablo 5.5. SU yöntemi için MAE ve MSE kalite değerleri (Popülasyon sayısı: 500)

Test Görüntüsü	Tekrarlama Sayısı =100				Tekrarlama Sayısı =200				Tekrarlama Sayısı =500			
	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO
1	**13,05**	13,24	**311,73**	321,73	**12,97**	13,04	**306,91**	310,35	**12,88**	12,95	**302,04**	305,23
2	**12,44**	12,68	**280,14**	294,54	**12,37**	12,50	**276,88**	283,90	**12,34**	12,43	**274,69**	280,96
3	**14,77**	14,88	**378,42**	384,57	**14,64**	14,70	**370,36**	374,54	**14,59**	14,62	**368,59**	370,35
4	**8,87**	9,11	**125,95**	132,83	**8,85**	8,96	**125,29**	128,41	**8,78**	8,88	**123,25**	126,24
5	**8,65**	8,83	**122,48**	128,17	**8,65**	8,76	**122,59**	125,61	**8,62**	8,66	**121,62**	122,76
6	**13,73**	14,14	**313,72**	334,94	**13,70**	13,86	**312,30**	321,03	**13,62**	13,75	**308,95**	314,84
7	**12,30**	12,54	**260,25**	273,63	**12,27**	12,40	**258,54**	265,77	**12,23**	12,30	**256,79**	261,13
8	**10,10**	10,62	**167,27**	184,05	**10,12**	10,35	**168,00**	175,49	**10,05**	10,22	**165,58**	171,16
9	**11,56**	11,72	**242,53**	250,75	**11,49**	11,60	**239,52**	244,66	**11,46**	11,51	**237,97**	241,01
10	**14,25**	14,58	**348,38**	367,56	**14,17**	14,31	**344,35**	352,60	**14,13**	14,22	**342,16**	347,08
11	**10,27**	10,53	**170,61**	179,65	**10,25**	10,41	**169,72**	175,67	**10,21**	10,29	**168,76**	171,47
12	**8,53**	8,96	**118,69**	130,00	**8,48**	8,67	**116,98**	122,15	**8,49**	8,57	**117,20**	119,58

Tablo 5.6. DU yöntemi için MAE ve MSE kalite değerleri (Popülasyon sayısı: 500)

Test Görüntüsü	Tekrarlama Sayısı =100				Tekrarlama Sayısı =200				Tekrarlama Sayısı =500			
	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO
1	**13,12**	13,15	**310,50**	314,84	**13,01**	12,95	**307,12**	304,98	12,94	**12,88**	303,92	**302,33**
2	**12,51**	13,52	**283,04**	332,45	**12,44**	14,73	**280,64**	395,40	**12,35**	14,65	**277,08**	393,65
3	14,80	**14,72**	**374,35**	374,83	**14,70**	15,00	**369,64**	391,07	**14,60**	14,76	**367,36**	375,79
4	**8,87**	9,10	**126,05**	132,31	**8,84**	9,16	**124,92**	133,96	**8,79**	9,00	**123,43**	129,69
5	**8,71**	8,76	**123,22**	126,07	**8,67**	8,71	**122,29**	124,17	**8,63**	8,72	**121,68**	124,33
6	**13,72**	14,13	**313,30**	335,15	**13,67**	14,59	**311,75**	363,54	**13,66**	14,22	**310,40**	346,98
7	**12,28**	12,67	**260,24**	277,88	**12,28**	12,47	**259,35**	268,99	**12,25**	12,38	**257,54**	264,16
8	**10,16**	11,12	**169,25**	190,60	**10,15**	10,78	**168,73**	189,24	**10,09**	11,95	**167,05**	232,26
9	**11,57**	11,83	**243,34**	256,00	**11,53**	11,75	**240,83**	251,12	**11,53**	11,89	**240,53**	254,10
10	**14,29**	14,70	**351,36**	370,63	**14,22**	14,44	**346,87**	359,44	**14,19**	15,16	**345,55**	395,33
11	**10,27**	10,56	**170,57**	180,54	**10,24**	10,68	**169,81**	184,86	**10,21**	10,67	**168,79**	184,51
12	**8,59**	9,74	**120,77**	157,27	**8,61**	9,91	**121,02**	167,33	**8,52**	9,90	**118,35**	167,41

Tablo 5.7. RDU yöntemi MAE ve MSE kalite değerleri (Popülasyon sayısı: 500)

Test Görüntüsü	Tekrarlama Sayısı =100				Tekrarlama Sayısı =200				Tekrarlama Sayısı =500			
	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO
1	**13,07**	15,30	**311,55**	416,11	**13,02**	14,98	**308,95**	394,33	**12,96**	15,37	**306,43**	401,46
2	**12,51**	13,82	**285,34**	342,90	**12,46**	14,62	**285,08**	389,50	**12,45**	14,70	**284,97**	396,33
3	**14,75**	15,34	**374,62**	398,73	**14,71**	15,71	**374,35**	418,28	**14,63**	16,48	**371,96**	467,08
4	**8,89**	9,19	**126,48**	134,84	**8,86**	9,20	**125,52**	135,09	**8,82**	9,05	**124,69**	130,70
5	**8,71**	9,47	**124,50**	144,97	**8,68**	9,00	**124,01**	131,23	**8,66**	8,82	**124,16**	125,87
6	**13,70**	14,46	**311,75**	351,69	**13,67**	14,50	**308,87**	360,61	**13,63**	14,54	**306,59**	364,74
7	**12,34**	14,00	**264,30**	335,19	**12,31**	13,11	**262,91**	296,72	**12,26**	13,64	**260,61**	325,30
8	**10,15**	10,89	**168,12**	193,35	**10,10**	10,94	**167,27**	195,30	**10,07**	11,63	**166,64**	219,76
9	**11,58**	12,12	**244,55**	268,90	**11,51**	11,90	**240,53**	257,47	**11,52**	11,98	**240,79**	258,20
10	**14,34**	15,49	**355,10**	409,60	**14,27**	15,17	**349,03**	393,59	**14,17**	15,39	**343,05**	403,34
11	**10,29**	10,91	**171,48**	192,66	**10,24**	10,91	**169,86**	192,04	**10,23**	11,36	**169,55**	210,15
12	**8,70**	9,57	**122,81**	150,87	**8,66**	10,14	**121,86**	175,81	**8,63**	10,05	**120,89**	171,99

Tablo 5.8. SU yöntemi eğitim süreleri (Popülasyon Sayısı: 500)

Test Görüntüsü	Tekrarlama Sayısı =100		Tekrarlama Sayısı =200		Tekrarlama Sayısı =500	
	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)
1	**44,00**	46,93	**92,85**	94,19	**224,73**	229,72
2	**44,48**	47,35	**93,63**	94,94	**226,49**	231,42
3	**44,66**	47,05	**92,87**	94,35	**224,85**	229,99
4	**44,69**	47,05	**92,93**	94,34	**225,05**	229,84
5	**45,07**	47,48	**94,09**	95,40	**227,23**	232,45
6	**44,69**	47,05	**93,11**	94,43	**225,16**	229,99
7	**45,08**	45,39	**92,80**	93,52	**226,37**	237,96
8	**45,13**	45,38	**92,81**	93,63	**226,54**	238,42
9	**45,14**	45,41	**92,83**	93,65	**226,47**	238,44
10	**45,17**	45,42	**92,93**	93,72	**226,41**	238,46
11	**45,15**	45,36	**92,63**	93,65	**226,49**	238,43
12	**45,20**	45,50	**93,07**	93,83	**227,13**	239,13

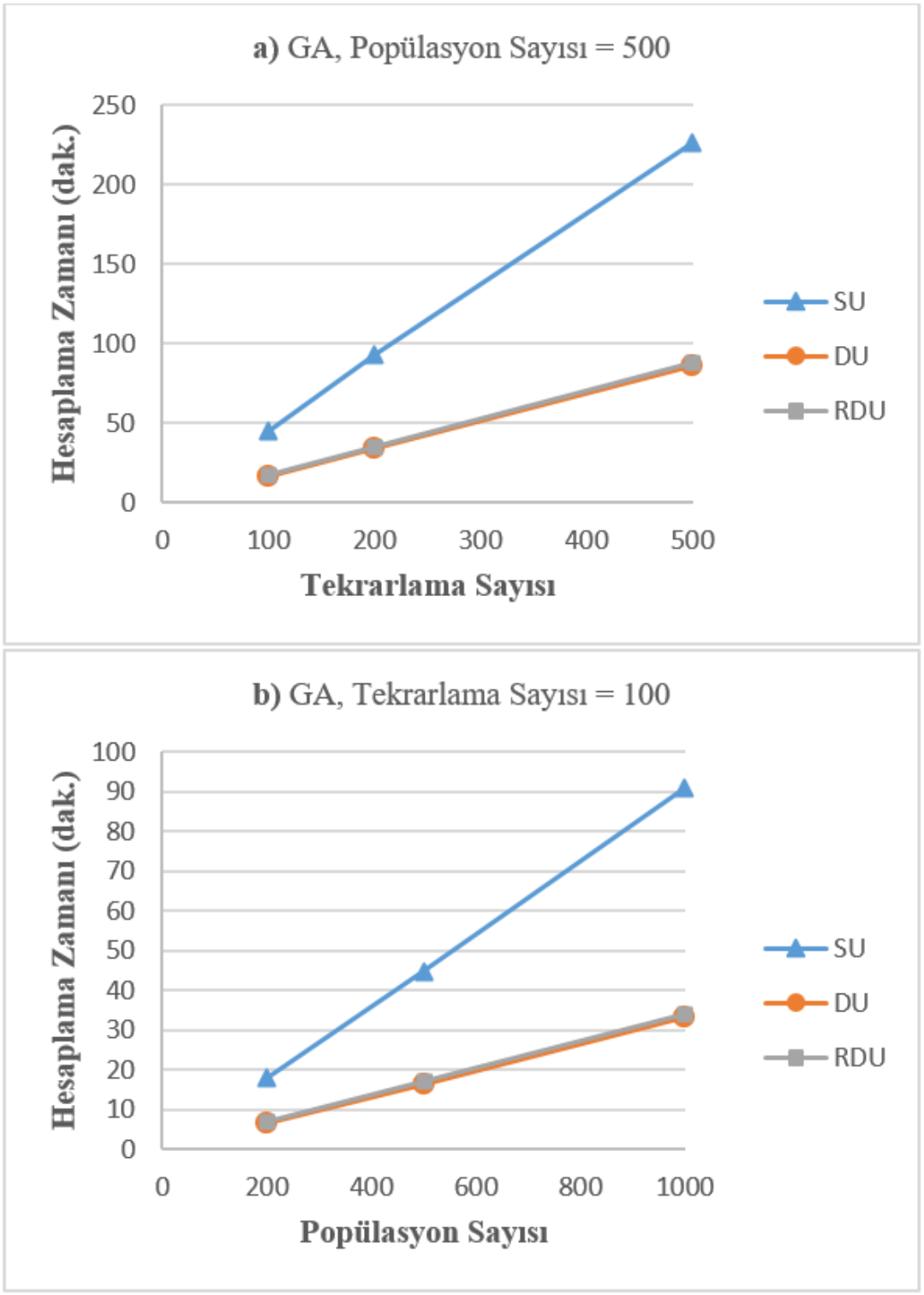
Tablo 5.9. DU yöntemi eğitim süreleri (Popülasyon Sayısı: 500)

Test Görüntüsü	Tekrarlama Sayısı =100		Tekrarlama Sayısı =200		Tekrarlama Sayısı =500	
	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)
1	**16,47**	17,57	**34,10**	35,95	**86,65**	89,68
2	**16,53**	17,66	**34,24**	36,08	**87,04**	90,13
3	**16,49**	17,60	**34,14**	35,86	**86,78**	89,80
4	**16,44**	17,59	**34,16**	35,97	**86,77**	89,86
5	**16,57**	17,69	**34,30**	36,15	**87,21**	90,38
6	**16,49**	17,61	**35,66**	35,98	**86,72**	89,87
7	**16,45**	17,41	**33,87**	35,59	**85,95**	89,44
8	**16,50**	17,43	**33,88**	35,64	**85,97**	89,58
9	**16,54**	17,45	**33,94**	35,68	**86,07**	89,64
10	**16,57**	17,50	**33,97**	35,71	**86,20**	89,81
11	**16,53**	17,47	**33,82**	35,65	**85,77**	89,67
12	**16,51**	17,45	**33,92**	35,65	**86,06**	89,63

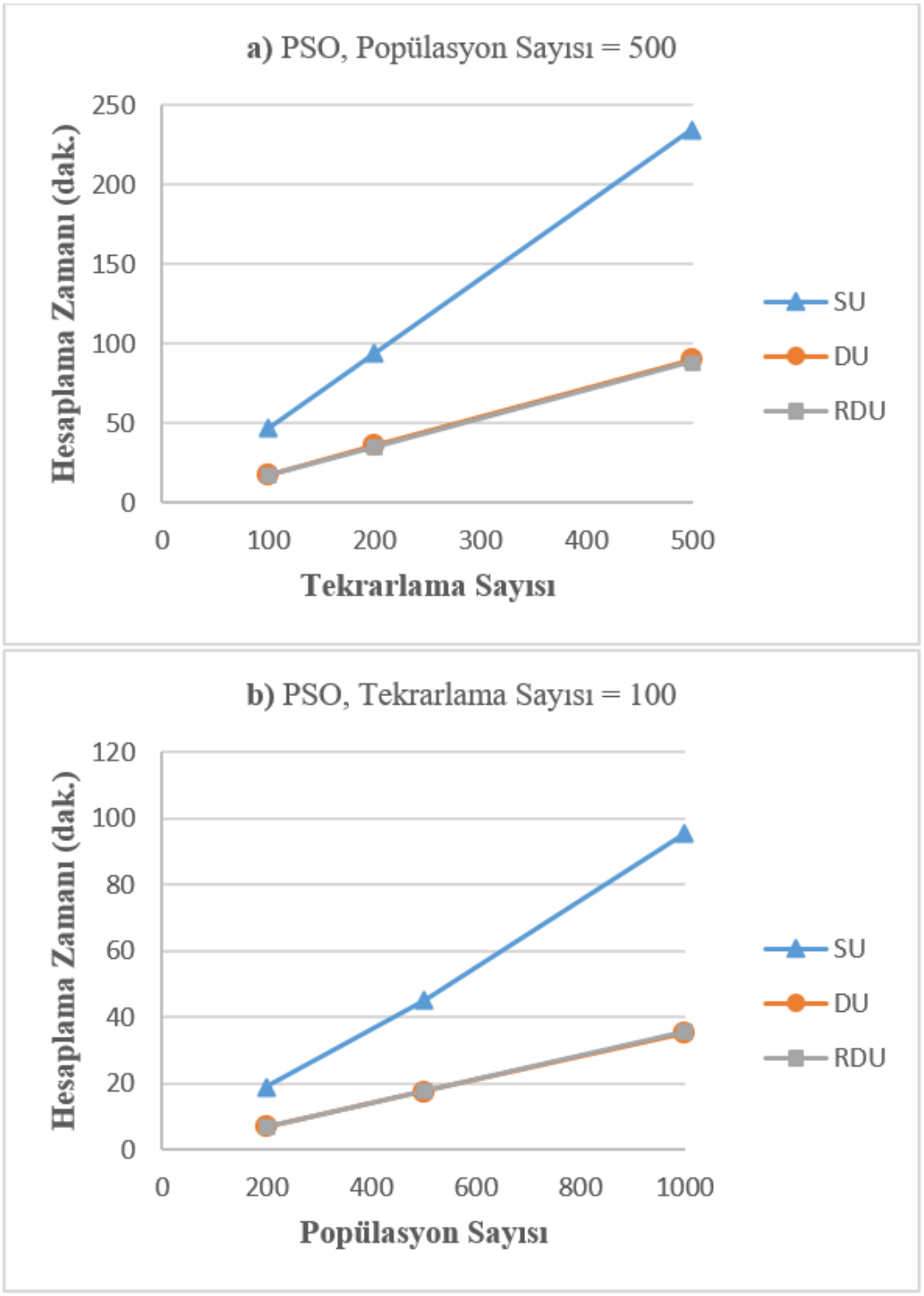
Tablo 5.10. RDU yöntemi eğitim süreleri (Popülasyon Sayısı: 500)

Test Görüntüsü	Tekrarlama Sayısı =100		Tekrarlama Sayısı =200		Tekrarlama Sayısı =500	
	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)
1	**17,18**	17,70	**34,75**	35,35	**87,36**	88,53
2	**17,29**	17,78	**34,91**	35,47	**87,73**	88,93
3	**17,21**	17,72	**34,72**	35,33	**87,26**	88,62
4	**17,25**	17,73	**34,78**	35,35	**87,46**	88,66
5	**17,24**	17,81	**34,84**	35,50	**87,54**	89,08
6	**17,21**	17,72	**34,82**	35,31	**87,57**	88,62
7	**17,18**	17,50	**34,83**	35,23	**87,36**	88,54
8	**17,28**	17,51	**35,08**	35,30	**87,67**	88,59
9	**17,24**	17,54	**34,95**	35,30	**87,40**	88,66
10	**17,23**	17,56	**34,97**	35,39	**87,41**	88,84
11	**17,21**	17,53	**34,99**	35,33	**87,36**	88,72
12	**17,26**	17,53	**35,23**	35,32	**87,66**	88,76

Uygulanan yöntemlerin GA ile eğitim süreleri Şekil 5.7.'de daha açık bir şekilde görülmektedir. Şekil 5.7a.'da GA ile eğitimde popülasyon sayısı 500'e sabitlendiğinde tekrarlama sayısı 100, 200 ve 500 olarak belirlendiğinde tekrarlama sayısına bağlı eğitim süreleri görülmektedir. Şekil 5.7b.'de yine GA ile eğitimde tekrarlama sayısı 100 olacak şekilde sabitlendiğinde popülasyon sayısı 200, 500 ve 1000 olarak belirlendiğinde popülasyon sayısına bağlı eğitim süreleri görülmektedir. SU yönteminde hesaplama zamanı yaklaşık 225 dakika iken DU ve RDU yöntemlerinde bu süre ortalama 95 dakika civarındadır. Önerilen yöntemler SU yöntemine göre ortalama 2,3 kat daha hızlı olduğu görülmektedir. Uygulanan yöntemlerin PSO ile eğitim süreleri Şekil 5.8.'da daha açık bir şekilde görülmektedir. Şekil 5.8a.'da PSO ile eğitimde popülasyon sayısı 500'e sabitlendiğinde tekrarlama sayısı 100, 200 ve 500 olarak belirlendiğinde tekrarlama sayısına bağlı eğitim süreleri görülmektedir. Şekil 5.8b.'de yine PSO ile eğitimde tekrarlama sayısı 100 olacak şekilde sabitlendiğinde popülasyon sayısı 200, 500 ve 1000 olarak belirlendiğinde popülasyon sayısına bağlı eğitim süreleri görülmektedir. Bütün şekillerde DU ve RDU eğitim sürelerinin çok yakın SU sürelerinin çok daha fazla olduğu görülmektedir. Tekrarlama sayılarının iki katına çıkartılması her bir yöntemin eğitim süresini de ortalama iki katına çıkardığı görülmektedir. Tekrarlama sayılarının 200'den 500'e çıkartılmasıyla eğitim sürelerinin de ortalama 2,5 katına çıktığı görülmektedir. Hesaplama zamanları karşılaştırıldığında DU yönteminin SU yöntemine göre yaklaşık 2,5 kat oranında eğitim sürelerini hızlandırdığı görülmektedir.



Şekil 5.7. SU, DU ve RDU yöntemlerinin GA ile eğitim sürelerinin tekrarlama sayısına göre grafiği



Şekil 5.8. SU, DU ve RDU yöntemlerinin PSO ile eğitim sürelerinin tekrarlama sayısına göre grafiği

Yapılan tez çalışmasında, algoritmik hızlandırmanın yanı sıra donanımsal olarak da hızlandırma yapılmıştır. Donanımsal hızlandırma için Nvidia GeForce ekran kartı

kullanılmıştır. Kuadratik görüntü filtrelerinin eğitiminde en çok zaman uygunluk değeri hesaplama aşamasında kaybedilmektedir. Bundan dolayı uygunluk değeri hesaplama fonksiyonu CUDA C dili kullanılarak GPU üzerinde çalışacak şekilde yeniden yazılmıştır. Donanımsal hızlandırma deneysel çalışması, Intel® Core ™ i7-3770 CPU, 3.40 GHz çift çekirdek, 16GB Ram, 2048 CUDA çekirdeğine sahip ve hesaplama kapasitesi 5.2 olan NVidia GeForce GTX980 ekran kartına sahip Windows 10 pro 64-bit işletim sistemine sahip bilgisayar üzerinde gerçekleştirilmiştir. Kodlar C++ ve CUDA C programlama dilinde Visual Studio 2015™ platformu üzerinde yazılmıştır. Deneylerde, Şekil 5.1.'de gösterilen referans görüntüler kullanılmıştır. Bu deneysel çalışmada hem CPU hem de GPU hibrit bir şekilde kullanılmıştır. Sebebi ise, Kuadratik görüntü filtrelerin bir görüntüyü filtrelemesi diğerlerine göre zor bir işlem olduğundan dolayı sadece uygunluk fonksiyonu hesabı GPU üzerinde yapılmıştır. Bunun dışındaki diğer kodlar CPU üzerinde yürütülmektedir. Ayrıca CPU üzerinde yürütülen kodların uygunluk fonksiyonu çalışma zamanına göre ihmal edilebilecek seviyelerde olduğu da gözlemlenmiştir. Bu deneysel çalışmada popülasyon sayısı 200, 400 ve 800 olacak şekilde farklı sayılarda belirlenirken algoritma tekrarlama sayısı 100 olarak belirlenmiştir. GPU üzerinde çalışması için geliştirilen iki farklı yöntem kullanılmıştır. İlk kullanılan yöntemde GPU üzerine popülasyonlar CPU'dan tek tek kopyalanmaktadır. Yani bir popülasyon GPU üzerine alınarak uygunluk değeri hesaplanmakta ve sonuçlar CPU'ya kopyalanmaktadır. Sonrasında diğer popülasyon GPU içerisine kopyalanmakta ve bu kez onun için uygunluk değeri hesaplanmaktadır. İşlemler bu şekilde belirlenen popülasyonun hepsi için tek tek yapılmaktadır. Diğer yöntemde ise popülasyon bir kerede GPU içerisine kopyalanmakta ve en sonunda uygunluk değeri CPU'ya kopyalanmaktadır. Böylece GPU içerisine bir kerede alınan bütün popülasyon için uygunluk değeri hesaplaması bir kerede yapılmaktadır. Bu tekniğe ise GPU Popülasyon Temelli (GPT) Yöntem adı verilmiştir. Bu her iki tekniğin kendi aralarında avantaj ve dezavantajları olduğu gibi eğitim sürelerini kısaltmak adına CPU'ya göre çok fazla avantaj sağladığı görülmüştür.

Tablo 5.11., Tablo 5.12. ve Tablo 5.13.'de Kuadratik görüntü filtrelerin maske ağırlıklarının eğitiminde donanımsal hızlandırma sağlamak için önerilen yöntemlerin kalite ölçüm sonuçları görülmektedir. Ayrıca, donanımsal hızlandırmayı kıyaslamak

amacıyla aynı eğitimin CPU üzerinde gerçekleştirilmesi sonucu elde edilen görüntü kalite ölçüm sonuçları da görülmektedir. Ayrıca her tabloda PSO ve GA için SU, GSU ve GPT yöntemlerinin karşılaştırmaları verilmiştir. Elde edilen deneysel sonuçlar popülasyon sayısı 200, 400 ve 800 olacak şekilde belirlenmiştir. Her bir ölçüm için algoritma tekrarlama sayısı 100'e sabitlenmiştir. 200 popülasyon sayısı için MAE karşılaştırıldığında hem CPU üzerinde yapılan çalışma hem de GPU üzerinde yapılan çalışma sonuçları çok yakın çıkmaktadır. Aynı davranış 400 ve 800 popülasyon sayıları için de gözlemlenmiştir. Bu durum, GPU üzerinde yapılan donanımsal hızlandırma için yazılan algoritmanın da doğru bir şekilde çalıştığını göstermektedir. Sayısal sonuçlara göre, popülasyon sayısının arttırılması, SU, GSU ve GPT yöntemlerinin görüntü kalite sonuçlarını iyileştirdiği gözlemlenmiştir. Popülasyon sayısı 800'e çıkartıldığında elde edilen MAE sonuçlarının bir birlerine yaklaştığı görülmüştür. Sonuçların seçilen algoritma parametrelerine çok duyarlı olduğu ve uygulanan algoritmaların popülasyon sayısına ve tekrarlama sayısının önemli ölçüde değişebileceğine dikkat edilmelidir. SU, GSU ve GPT yöntemlerinin eğitim süreleri Tablo 5.14., Tablo 5.15. ve Tablo 5.16.'da sırasıyla SU, GSU ve GPT yöntemleri, GA ve PSO ile eğitildiklerinde elde edilen eğitim sürelerini de gösterilmektedir. Popülasyon sayılarının artışı eğitim sürelerini bütün algoritmalar için arttırdığı görülmektedir. Tekrarlama sayısı 100 olacak şekilde sabit tutulduğunda 200 popülasyon sayısı için SU yönteminde ortalama eğitim süresi 18,04 dakika olurken, GSU yönteminde bu süre 0,43 dakika olmakta, GPT yönteminde ise bu süre 0,13 dakika olmaktadır. Popülasyon sayısı 400'e çıkartıldığında SU yönteminde ortalama eğitim süresi 36,40 dakika olurken, GSU yönteminde bu süre 0,83 dakika olmakta, GPT yönteminde ise 0,23 dakika olmaktadır. Popülasyon sayısı 800'e çıkartıldığında SU yönteminde ortalama eğitim süresi 72,68 dakika olurken, GSU yönteminde bu süre 1,65 dakika olmakta, GPT yönteminde ise 0,46 dakika olmaktadır. Bu sonuçlara göre önerilen donanımsal hızlandırma algoritmalarının CPU'ya göre çok daha kısa sürelerde eğitimlerinin bittiği görülmektedir. Kuadratik görüntü filtreleri gibi ağır filtrelerin, filtreleme eğitimleri sonunda görüntü kalitesi korunurken eğitim sürelerinin donanımsal olarak azaltılması hedefine ulaşıldığı görülmektedir.

Tablo 5.11. CPU kullanılarak SU yöntemi ile elde edilen MAE ve MSE kalite değerleri (Tekrarlama sayısı: 100)

Test	Popülasyon Sayısı =200				Popülasyon Sayısı =400				Popülasyon Sayısı =800			
Görüntüsü	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO
1	**13,35**	13,39	**326,46**	328,25	**13,11**	13,21	**312,80**	320,60	**12,97**	13,12	**308,08**	315,03
2	**12,64**	12,82	**286,65**	299,89	**12,49**	12,58	**282,49**	293,52	**12,37**	12,61	**277,61**	290,80
3	**15,03**	15,13	**390,85**	397,50	**14,78**	14,84	**377,70**	383,41	**14,66**	14,82	**373,27**	381,92
4	**9,01**	9,24	**129,75**	136,37	**8,91**	9,11	**126,85**	132,60	**8,85**	9,14	**125,21**	133,33
5	**8,85**	8,87	**128,65**	129,58	**8,64**	8,82	**122,33**	127,77	**8,63**	8,79	**122,13**	127,29
6	**14,04**	14,40	**327,90**	346,35	**13,75**	14,09	**315,53**	332,34	**13,64**	14,09	**310,47**	332,40
7	**12,55**	12,67	**270,96**	277,03	**12,37**	12,67	**264,71**	279,15	**12,27**	12,53	**259,30**	273,20
8	**10,38**	10,90	**176,16**	193,26	**10,13**	10,66	**168,07**	185,86	**10,08**	10,45	**166,75**	178,76
9	**11,71**	11,96	**250,72**	261,94	**11,57**	11,81	**243,22**	254,03	**11,53**	11,73	**241,37**	251,53
10	**14,48**	14,57	**361,04**	366,42	**14,23**	14,57	**347,14**	366,96	**14,18**	14,39	**343,96**	355,34
11	**10,48**	10,74	**177,28**	187,28	**10,31**	10,50	**171,84**	178,52	**10,24**	10,54	**169,51**	180,11
12	**8,69**	9,18	**123,20**	137,22	**8,55**	8,87	**118,60**	128,01	**8,51**	8,77	**117,78**	125,11

Tablo 5.12. GSU yöntemi kullanılarak elde edilen MAE ve MSE kalite değerleri (Tekrarlama sayısı: 100)

Test	Popülasyon Sayısı =200				Popülasyon Sayısı =400				Popülasyon Sayısı =800			
Görüntüsü	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO
1	**13,30**	13,40	**325,46**	331,30	**13,08**	13,17	**312,62**	318,01	**12,99**	13,13	**308,43**	315,61
2	**12,75**	12,80	**292,28**	300,60	**12,43**	12,72	**280,53**	294,71	**12,39**	12,60	**276,84**	288,97
3	**15,01**	15,03	**390,13**	395,26	**14,80**	14,91	**378,83**	386,88	**14,69**	14,76	**373,39**	378,39
4	**8,89**	9,25	**129,29**	136,56	**8,88**	9,13	**126,05**	132,90	**8,85**	9,09	**125,21**	131,99
5	**8,82**	8,93	**127,28**	131,03	**8,72**	8,85	**124,65**	128,72	**8,63**	8,78	**122,21**	126,74
6	**13,95**	14,35	**324,52**	343,56	**13,73**	14,21	**314,14**	339,70	**13,68**	14,03	**311,61**	329,49
7	**12,53**	12,69	**269,90**	279,59	**12,35**	12,58	**261,58**	274,34	**12,26**	12,49	**258,91**	268,62
8	**10,28**	10,91	**172,85**	194,26	**10,15**	10,73	**168,95**	187,94	**10,04**	10,46	**165,68**	179,15
9	**11,73**	11,90	**251,19**	257,80	**11,60**	11,72	**243,77**	251,11	**11,51**	11,71	**240,26**	249,28
10	**14,52**	14,83	**361,45**	378,59	**14,24**	14,57	**346,13**	366,57	**14,20**	14,50	**345,72**	362,61
11	**10,50**	10,70	**177,74**	185,37	**10,30**	10,51	**171,62**	179,25	**10,22**	10,41	**168,99**	175,98
12	**8,69**	9,20	**122,56**	137,77	**8,55**	8,95	**119,19**	129,91	**8,54**	8,73	**118,58**	123,45

Tablo 5.13. GPT yöntemi kullanılarak elde edilen MAE ve MSE kalite değerleri (Tekrarlama sayısı: 100)

Test Görüntüsü	Popülasyon Sayısı =200				Popülasyon Sayısı =400				Popülasyon Sayısı =800			
	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO
1	**13,30**	13,39	**322,28**	328,70	**13,03**	13,24	**310,82**	321,54	**12,98**	13,16	**307,67**	315,97
2	**12,78**	12,75	**292,45**	301,21	**12,48**	12,66	**281,86**	293,52	**12,36**	12,59	**276,03**	287,69
3	**14,93**	15,02	**385,90**	393,52	**14,77**	14,93	**377,25**	385,50	**14,64**	14,80	**371,89**	379,33
4	**9,05**	9,22	**130,62**	135,90	**8,88**	9,12	**126,18**	133,06	**8,83**	9,09	**124,95**	131,98
5	8,92	**8,87**	130,60	**129,58**	**8,71**	8,82	**124,71**	127,75	**8,63**	8,77	**121,97**	126,64
6	**13,96**	14,59	**325,25**	357,72	**13,70**	14,26	**313,48**	341,50	**13,67**	14,03	**311,55**	327,75
7	**12,65**	12,81	**274,79**	284,22	**12,32**	12,57	**261,33**	274,40	**12,31**	12,48	**260,18**	269,04
8	**10,38**	10,84	**175,75**	190,95	**10,16**	10,67	**168,98**	186,07	**10,07**	10,43	**166,46**	177,94
9	**11,73**	11,86	**250,46**	255,25	**11,64**	11,80	**246,16**	254,22	**11,50**	11,72	**240,20**	250,92
10	**14,42**	14,64	**356,78**	369,09	**14,26**	14,65	**348,93**	368,57	**14,19**	14,39	**344,75**	356,73
11	**10,53**	10,71	**179,26**	186,65	**10,27**	10,55	**170,96**	180,60	**10,25**	10,52	**170,03**	179,73
12	**8,84**	9,03	**126,78**	132,46	**8,56**	9,01	**119,52**	132,06	**8,51**	8,85	**117,94**	126,93

Tablo 5.14. CPU kullanılarak SU yöntemi ile elde edilen eğitim süreleri (Tekrarlama sayısı: 100)

Test Görüntüsü	Popülasyon Sayısı =200		Popülasyon Sayısı =400		Popülasyon Sayısı =800	
	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)
1	**17,90**	19,09	**36,55**	36,60	**72,99**	76,91
2	**18,11**	19,24	**36,83**	36,93	**73,83**	77,54
3	**18,00**	19,11	**36,59**	36,71	**73,02**	77,08
4	**18,06**	19,11	**36,59**	36,70	**73,06**	77,19
5	**18,27**	19,18	**36,79**	37,05	**74,16**	77,84
6	**18,07**	19,13	**36,59**	36,70	**73,13**	77,21
7	**18,01**	18,83	**36,11**	36,52	**71,74**	75,94
8	**18,01**	18,85	**36,15**	36,53	**72,13**	75,67
9	**18,02**	18,86	**36,13**	36,53	**71,80**	75,56
10	**17,98**	18,87	**36,16**	36,54	**71,88**	75,72
11	**18,00**	18,85	**36,12**	36,56	**72,16**	75,62
12	**18,05**	18,87	**36,22**	36,63	**72,28**	75,92

Tablo 5.15. GSU yöntemi kullanılarak elde edilen eğitim süreleri (Tekrarlama sayısı: 100)

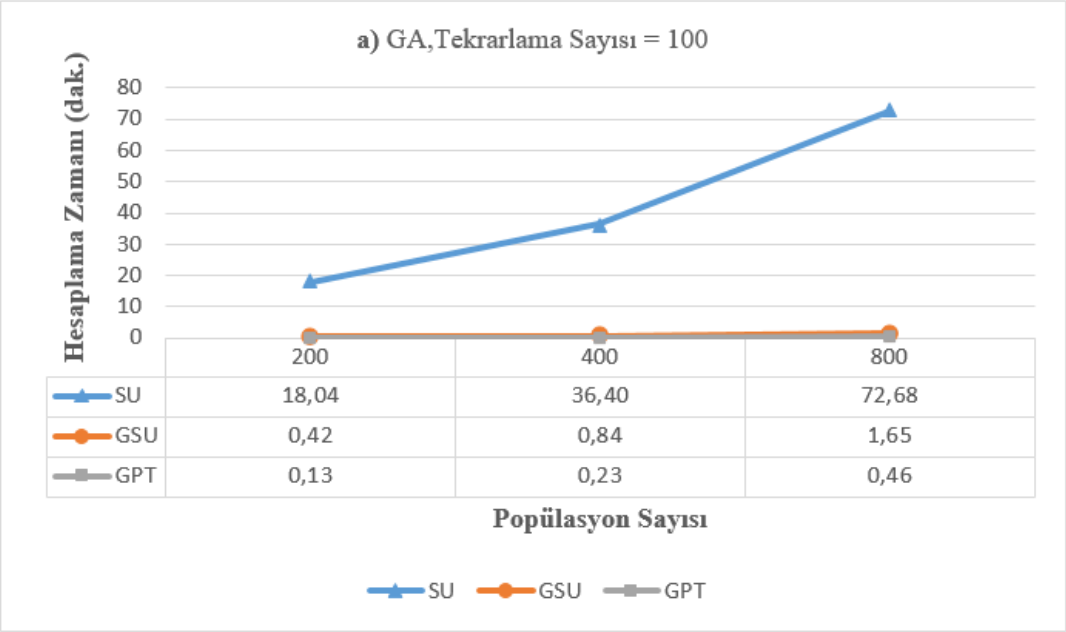
Test Görüntüsü	Popülasyon Sayısı =200		Popülasyon Sayısı =400		Popülasyon Sayısı =800	
	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)
1	**0,42**	0,55	**0,84**	1,10	**1,67**	2,16
2	**0,42**	0,54	**0,84**	1,10	**1,65**	2,14
3	**0,43**	0,55	**0,84**	1,10	**1,65**	2,15
4	**0,42**	0,55	**0,84**	1,08	**1,69**	2,15
5	**0,42**	0,54	**0,84**	1,07	**1,66**	2,19
6	**0,43**	0,55	**0,84**	1,08	**1,64**	2,17
7	**0,42**	0,55	**0,84**	1,08	**1,64**	2,19
8	**0,42**	0,55	**0,85**	1,09	**1,65**	2,18
9	**0,42**	0,55	**0,84**	1,09	**1,64**	2,17
10	**0,42**	0,55	**0,83**	1,08	**1,64**	2,16
11	**0,43**	0,55	**0,84**	1,08	**1,63**	2,16
12	**0,42**	0,55	**0,83**	1,07	**1,64**	2,16

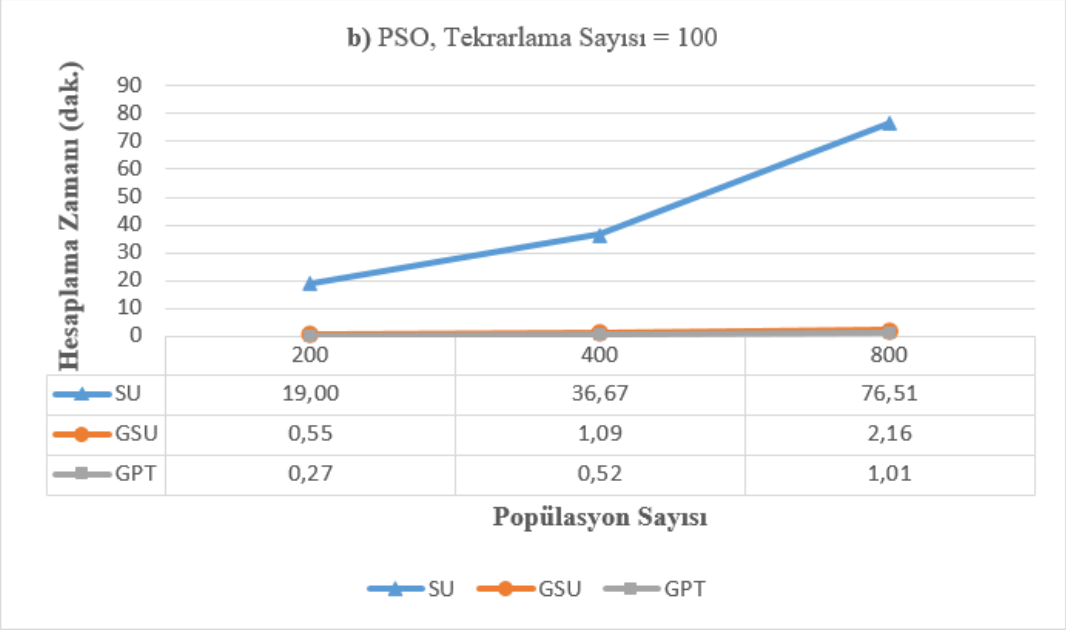
Tablo 5.16. GPT yöntemi kullanılarak elde edilen eğitim süreleri (Tekrarlama sayısı: 100)

Test Görüntüsü	Popülasyon Sayısı =200		Popülasyon Sayısı =400		Popülasyon Sayısı =800	
	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)
1	**0,13**	0,27	**0,23**	0,52	**0,51**	1,06
2	**0,13**	0,26	**0,23**	0,51	**0,51**	1,05
3	**0,13**	0,26	**0,23**	0,52	**0,51**	1,06
4	**0,13**	0,27	**0,23**	0,52	**0,51**	1,07
5	**0,13**	0,26	**0,23**	0,52	**0,51**	1,06
6	**0,13**	0,27	**0,23**	0,53	**0,44**	1,03
7	**0,13**	0,27	**0,23**	0,52	**0,44**	0,95
8	**0,13**	0,27	**0,23**	0,53	**0,44**	0,98
9	**0,13**	0,27	**0,23**	0,53	**0,43**	0,97
10	**0,13**	0,27	**0,23**	0,53	**0,42**	0,97
11	**0,13**	0,27	**0,23**	0,52	**0,42**	0,97
12	**0,13**	0,28	**0,26**	0,53	**0,42**	0,97

Donanımsal hızlandırmada uygulanan yöntemlerin eğitim süreleri Şekil 5.9.‘de karşılaştırmalı grafikler ile daha açık bir şekilde gösterilmiştir. Şekil 5.9a.’da GA ile eğitimde tekrarlama sayısı 100’e sabitlendiğinde popülasyon sayısı 200, 400 ve 800 olarak belirlendiğinde popülasyon sayısına bağlı eğitim süreleri görülmektedir. Şekil 5.9b.’de PSO ile eğitimde tekrarlama sayısı 100’e sabitlendiğinde popülasyon sayısı 200, 400 ve 800 olarak belirlendiğinde popülasyon sayısına bağlı eğitim süreleri görülmektedir. GSU yöntemi ve GPT yöntemindeki eğitim sürelerinin CPU üzerinde gerçekleştirilen SU yöntemi süresine göre çok daha az olduğu ve çok daha performanslı olduğu görülmektedir. GPT yöntemindeki eğitim süresi GSU yöntemine göre yaklaşık 3,23 kat daha hızlı olduğu görülmektedir. Bunun sebebi ise CPU’dan GPU’ya veri kopyalama işi oldukça zaman aldığından ve GSU yönteminde ise her bir popülasyon tek tek GPU’ya kopyalanmasından kaynaklanmaktadır. SU yöntemi eğitim sürelerinin ise hem GSU hem de GPT yöntemine göre çok daha fazla olduğu görülmektedir. Bunun sebebi ise GPU içerisinde hesaplama yapan çekirdek sayısının CPU’ya oranla çok daha fazla olmasıdır. Önerilen yöntemlerin CPU üzerinde çalışan SU yöntemine göre önemli derecede zaman kazancı sağladığı açık bir şekilde görülmektedir. Popülasyon sayılarının iki katına çıkartılması her bir yöntemin eğitim süresini de ortalama iki katına çıkartıldığı görülmektedir. Popülasyon sayılarının 400’den 800’e çıkartılmasıyla eğitim sürelerinin de ortalama 2,02 katına çıktığı görülmektedir.

RDU ve GPT yöntemlerinin bir arada kullanıldığı hem algoritmik hem de donanımsal hızlandırma sağlayan yeni bir algoritma tasarlanmıştır. Bunun sebebi, GPT ve RDU yöntemlerinin deneysel sonuçlarında elde edilen MAE değerlerinin ve eğitim sürelerinin çok daha iyi olmasıdır. Her iki yöntemin birleştirilmesiyle görüntü kalitesinden ödün verilmeden çok daha kısa eğitim süreleri elde edilmiştir. Tablo 5.17.’de GPT-RDU yönteminin maske katsayı eğitimleri hem GA hem de PSO ile yapılmış ve görüntü kalitesi sonuçları verilmiştir. Tablodan da anlaşılacağı üzere MAE görüntü kalitesi hem GA hem de PSO ile yapılan eğitimlerde korunmaktadır.





Şekil 5.9. CPU üzerinde çalışan SU, GSU ve GPT yöntemlerinin GA ve PSO için eğitim sürelerinin tekrarlama sayısına göre grafiği

Ayrıca diğer yöntemlerde elde edilen sonuçlarda olduğu gibi popülasyon sayısının artması MAE görüntü kalitesinin daha da iyileşmesini sağladığı görülmektedir. Tablo 5.18.'de ise GPT-RDU yönteminin hem GA hem de PSO ile eğitim süreleri görülmektedir. Tablo 5.18.'deki sonuçlara göre tekrarlama sayısı 100 olacak şekilde sabit tutulduğunda 200 popülasyon sayısı için GPT-RDU yönteminde ortalama eğitim süresi 0,05 dakika olmaktadır. Popülasyon sayısı 400'e çıkartıldığında GPT-RDU

yönteminde ortalama eğitim süresi 0,10 dakika olmaktadır. Popülasyon sayısı 800'e çıkartıldığında GPT-RDU yönteminde ortalama eğitim süresi 0,19 dakika olmaktadır. Bu sonuçlara göre önerilen hem donanımsal hem de algoritmik hızlandırma algoritması hem CPU'ya göre hem de önerilen diğer algoritmik ve donanımsal hızlandırma yöntemlerine göre çok daha kısa sürelerde eğitimlerinin bittikleri görülmektedir. Kuadratik görüntü filtrelerinin GPT-RDU yöntemi ile filtreleme eğitimleri sonunda görüntü kalitesi korunurken eğitim sürelerinin azaltılması hedefine ulaşıldığı görülmektedir.

Tablo 5.19.'da CPU üzerinde çalışan SU yöntemi, donanımsal hızlandırma için önerilen GSU, ve GPT yöntemleri ile hem algoritmik hem de donanımsal hızlandırma için önerilen GPT-RDU yöntemlerinin GA ile eğitim sürelerinin ortalamaları görülmektedir. Eğitim süreleri PSO içinde çok yakın çıktığı için sadece GA için eğitim süreleri verilmiştir. Tablo 5.19.'da verilen süreler, Tablo 5.14., Tablo 5.15., Tablo 5.16. ve Tablo 5.18.'deki 12 görüntünün ortalama eğitim süreleridir. Bu sürelere bakıldığında tekrarlama sayısı 100 olacak şekilde sabit tutulduğunda 200, 400 ve 800 popülasyon sayıları kullanıldığında eğitim süresi en kısa olan yöntem GPT-RDU yöntemi olduğu görülmektedir.

Tablo 5.17. GPT ve RDU yöntemlerinin birlikte kullanımı ile elde edilen MAE ve MSE kalite değerleri (Tekrarlama sayısı: 100)

Test Görüntüsü	Popülasyon Sayısı =200				Popülasyon Sayısı =400				Popülasyon Sayısı =800			
	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO
1	**13,48**	13,60	**330,62**	334,00	**13,13**	13,68	**313,16**	336,21	**13,01**	13,51	**306,01**	326,76
2	**12,71**	13,01	**295,41**	313,42	**12,52**	12,97	**287,20**	312,61	**12,41**	13,03	**282,22**	313,42
3	**15,08**	15,27	**390,41**	395,13	**14,89**	15,40	**381,04**	403,30	**14,80**	15,07	**375,75**	387,24
4	**9,13**	9,32	**133,13**	138,28	**8,90**	9,55	**126,72**	145,57	**8,84**	9,31	**124,86**	138,15
5	**8,95**	9,08	**131,11**	133,98	**8,73**	8,99	**124,66**	131,63	**8,71**	8,91	**124,11**	129,59
6	**13,87**	14,77	**320,26**	364,99	**13,75**	15,06	**315,32**	377,78	**13,67**	14,63	**311,17**	359,15
7	**12,57**	12,77	**274,27**	285,17	**12,43**	12,82	**266,77**	288,51	**12,30**	12,59	**262,24**	277,20
8	**10,35**	11,20	**174,87**	204,28	**10,12**	11,40	**167,83**	210,86	**10,07**	11,04	**166,39**	198,48
9	**12,20**	12,32	**274,86**	277,23	**11,64**	12,05	**246,71**	264,37	**11,54**	11,90	**241,95**	258,21
10	**14,62**	15,43	**367,44**	408,46	**14,30**	15,09	**352,68**	391,30	**14,24**	14,91	**349,92**	382,97
11	**10,42**	11,46	**175,78**	212,59	**10,31**	10,95	**171,91**	193,97	**10,27**	10,76	**170,62**	187,58
12	**8,70**	9,29	**123,01**	141,16	**8,54**	9,50	**119,04**	148,00	**8,50**	9,58	**117,87**	152,16

Tablo 5.18. GPT ve RDU yöntemlerinin birlikte kullanımı ile elde edilen eğitim süreleri (Tekrarlama sayısı: 100)

Test Görüntüsü	Popülasyon Sayısı =200		Popülasyon Sayısı =400		Popülasyon Sayısı =800	
	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)	GA Eğitim Süresi(dak.)	PSO Eğitim Süresi(dak.)
1	**0,06**	0,09	**0,10**	0,14	**0,20**	0,24
2	**0,05**	0,09	**0,10**	0,14	**0,20**	0,24
3	**0,05**	0,09	**0,10**	0,14	**0,19**	0,24
4	**0,05**	0,09	**0,10**	0,14	**0,19**	0,24
5	**0,05**	0,09	**0,10**	0,14	**0,19**	0,24
6	**0,05**	0,09	**0,10**	0,14	**0,19**	0,24
7	**0,05**	0,09	**0,10**	0,14	**0,19**	0,24
8	**0,05**	0,09	**0,10**	0,14	**0,19**	0,24
9	**0,05**	0,09	**0,10**	0,14	**0,19**	0,24
10	**0,05**	0,09	**0,10**	0,14	**0,21**	0,24
11	**0,05**	0,09	**0,10**	0,14	**0,21**	0,24
12	**0,05**	0,09	**0,10**	0,14	**0,21**	0,24

Tablo 5.19. SU, GSU ve GPT yöntemlerinin GA için eğitim süreleri (dak.)

	Popülasyon Sayısı		
Yöntem	**200**	**400**	**800**
SU	18,04	36,40	72,68
GSU	0,43	0,83	1,65
GPT	0,13	0,23	0,46
GPT-RDU	**0,05**	**0,10**	**0,19**

Tablo 5.20.'de önerilen donanımsal hızlandırma yöntemlerinin kendi aralarındaki hızlandırma oranları ile hem algoritmik hem de donanımsal hızlandırma yöntemi arasındaki hızlandırma oranları görülmektedir. Tablo 5.20.'ye göre tekrarlama sayısı 100 olacak şekilde sabit tutulduğunda popülasyon sayısı 200 olarak belirlendiğinde GSU yönteminin SU yöntemine göre 42,95 kat, GPT yönteminin SU yöntemine göre 138,76 kat ve GPT-RDU yöntemninin SU yöntemine göre ise 257,71 kat daha fazla hızlandırma elde edildiği görülmektedir. Ayrıca GPT-RDU yönteminin GSU yöntemine göre 6 kat ve GPT yöntemine göre 1,85 kat daha fazla hızlanma elde edilmiştir. GPT yöntemi ise GSU yöntemine göre de 3,23 kat daha fazla hızlanma elde edildiği görülmektedir. Aynı şekilde popülasyon sayısı 400 ve 800'e çıkartıldığında ise en fazla hızlandırma oranının GPT-RDU yöntemi ile elde edildiği görülmektedir.

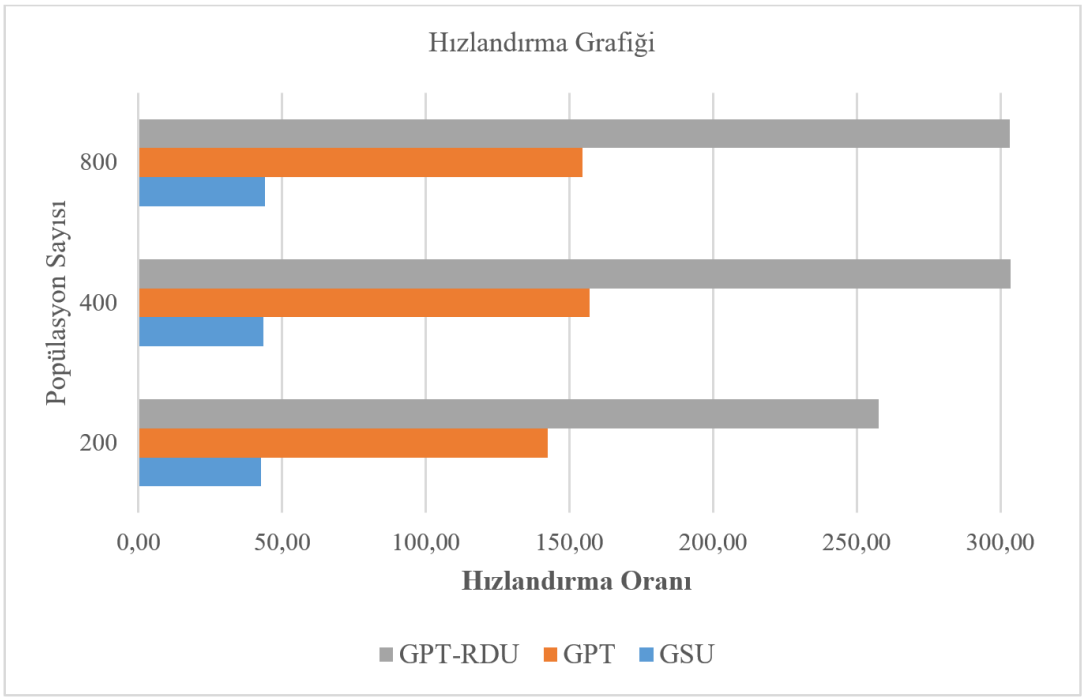
Tablo 5.20. SU, GSU ve GPT yöntemlerinin GA için hızlandırma tablosu

	Popülasyon Sayısı		
Yöntem	**200**	**400**	**800**
SU/GSU	42,95	43,38	44,10
SU/GPT	138,76	158,43	158,19
GSU / GPT	3,23	3,65	3,58
SU / GPT-RDU	**257,71**	**303,66**	**303,20**
GSU / GPT-RDU	6,00	7,00	6,87
GPT / GPT-RDU	1,85	1,91	1,91

Şekil 5.10.'da Tablo 5.20.'de gösterilen hızlandırma oranlarının grafik olarak gösterimi görülmektedir. Şekil 5.10.'a bakıldığında, 200 popülasyon sayısı için GSU yönteminin CPU'ya göre yaklaşık 42,95 kat, GPT yönteminin CPU'ya göre 138,76 kat oranında eğitim sürelerini azalttığı, 400 popülasyon sayısı için GSU yöntemi 43,38

kat, GPT yöntemi 158,43 kat oranında eğitim süresini azalttığı, 800 popülasyon sayısı için GSU yöntemi 44,10 kat, GPT 158,19 kat oranında eğitim süresini azalttığı görülmektedir. GSU ve GPT yöntemlerinin kendi aralarındaki hızlandırma oranları ise 200 popülasyon için 3,23 kat, 400 popülasyon sayısı için 3,65 kat ve 800 popülasyon için 3,58 kat olduğu görülmektedir. Buradaki hızlanma sebebi ise CPU'dan CPU'ya ve CPU'dan GPU'ya doğru yapılan veri kopyalama işlemleridir. GPT-RDU yönteminin SU yöntemine göre 200 popülasyon için 257,71 kat hızlandırma sağladığı görülmektedir. Popülasyon sayısı iki katına çıkartıldığında hızlanma oranı 303,66 kat olmaktadır. Popülasyon sayısı 4 katına çıkartıldığında ise hızlanma oranı 303,20 kat olmaktadır. GPT-RDU yöntemi GSU yöntemi ile karşılaştırıldığında ise 200 popülasyon sayısı için 6 kat, 400 popülasyon sayısı için 7 kat ve 800 popülasyon sayısı için 6,87 kat hızlanma sağladığı görülmektedir. Bu her iki yöntemde GPU üzerinde çalışmasına rağmen GPT-RDU yönteminin eğitim süreleri oldukça düşük çıkmaktadır. GPT-RDU yöntemi GPT ile karşılaştırıldığında ise, 200 popülasyon için 1,85 kat, 400 popülasyon için 1,91 ve 800 popülasyon için 1,91 kat hızlandırma sağladığı görülmektedir.

Sonuç olarak önerilen yöntemlerin tamamında görüntü kalitesinden ödün vermeden hızlandırma elde edilmiştir. En fazla hızlandırma hem algoritmik hem de donanımsal hızlandırma yöntemi olan GPT-RDU yönteminden elde edilmiştir. Daha sonra donanımsal hızlandırma yöntemlerinden GPT ve GSU yöntemlerinde elde edilmiştir. En az hızlandırma ise algoritmik hızlandırma yöntemlerinden RDU ve DU yöntemlerinden elde edilmiştir.



Şekil 5.10. SU yöntemine göre, GSU, GPT, GPT-RDU yöntemlerinin GA için hızlandırma grafiği

BÖLÜM 6. SONUÇ VE ÖNERİLER

Sunulan çalışmalarda, GA ve PSO sezgisel algoritmalar kullanılarak Kuadratik görüntü filtrelerin maske ağırlıklarının belirlenmesi ayrıntılı olarak incelenmiştir. Bu amaçla Kuadratik görüntü filtrelerinin maske ağırlıklarının eğitim sorun tek bir hedef optimizasyon problemi olarak formüle edilmiştir. Basit bir yaklaşım olarak SU, hem GA hem de PSO kullanılarak en uygun maske ağırlıklarının elde edilmesi için doğrudan kullanılmıştır. Deneysel sonuçlar her iki algoritmanın da MAE ve MSE görüntü kalitesi değerlerinin birbirlerine yakın olduğunu göstermiştir. Görüntü filtreleme işlemlerinin maliyeti nedeniyle uygunluk fonksiyonunun hesaplanması oldukça uzun zaman almaktadır. Bu nedenle hem GA hem de PSO optimizasyon algoritmalarının eğitim süreleri birbirine yakın elde edilmiştir. Hesaplama sürelerini kısaltmak için; algoritmik hızlandırma, donanımsal hızlanma ve hem algoritmik hem de donanımsal hızlanma yöntemleri geliştirilmiştir.

Algoritmik hızlanma olarak DU yöntemi ve onun rastgele versiyonu olan RDU yöntemleri sezgisel algoritmalarla birlikte kullanılmıştır. Bu her iki yöntemde görüntü filtreleme işlemlerinin başlangıcında görüntünün çok küçük bir bölümünü kullanarak arama yapmaya başlar ve tekrarlama sayıları arttıkça bu bölge büyür ve bütün görüntüye yakınlaşır. Belirli bir bölge yerine filtreleme esnasında filtrelemenin yapılacağı görüntü piksellerinin bütün görüntü üzerinden rastgele olacak şekilde seçilmesinin, görüntü filtreleme kalitesini arttırdığı görülmüştür. Elde edilen deneysel sonuçlara göre SU ve RDU yöntemlerinin GA ve PSO ile eğitimlerinin sonucunda elde edilen görüntülerin görüntü kalite değerleri benzer çıkmaktadır. Sonuçlar ayrıca DU veya RDU yönteminin SU yöntemine göre yaklaşık 2,5 ile 3,0 kat algoritmik hızlanma sağladığı görülmüştür. Önerilen algoritmalar diğer sezgisel algoritmaların çoğu ile birleştirilebilir.

Donanımsal hızlanma yöntemi için önerilen yöntemler Nvidia GeForce GTX980 ekran kartı üzerinde uygunluk fonksiyonlarının çalışacağı şekilde yeniden yazılmıştır. GSU ve GPT adı verilen bu yöntemler sezgisel algoritmalarla birlikte kullanılmıştır. Her iki yöntemin arasındaki en temel fark GSU yönteminde popülasyonlar tek tek GPU içerisine kopyalanmakta ve her bir popülasyon için hesaplanan uygunluk değeri GPU'dan CPU'ya kopyalanırken, GPT yönteminde popülasyonlar toplu halde bir kerede GPU içerisine alınmakta ve sadece bütün popülasyonun ortak uygunluk değeri hesaplanıp CPU'ya kopyalanmaktadır. Elde edilen görüntü kalitesi deneysel sonuçlarına göre hem GSU hem de GPT yöntemlerinde GA ve PSO için çoğu sonuçlar benzer çıkmaktadır. Fakat eğitim sürelerine bakıldığında GA ile PSO arasında ortalama 2 kat zaman farkı görülmektedir. Buradan donanımsal hızlandırma için geliştirilen algoritmaların GA ile eğitimlerinin PSO ile eğitimlerine göre ortalama 1,5 kat daha hızlı çalıştığı görülmektedir.

Hem donanımsal hem de algoritmik hızlanma olarak, algoritmik hızlandırma yöntemlerinden en iyi performans sağlayan RDU yöntemi ile donanımsal en iyi performans sağlayan GPT yöntemleri birleştirilerek GPU kart üzerinde çalışması sağlanmıştır. GPT-RDU yöntemi hem GA hem de PSO ile birlikte kullanılmıştır. Bu yeni yöntemle GPT ve RDU yöntemlerinde elde edilen görüntü kalitelerinden ödün vermeden eğitim sürelerinin daha fazla düşürülmesi sağlanmıştır. Yeni yöntemin en önemli avantajlarından biri başlangıçta eğitimi yapılacak olan bütün popülasyonun bir kerede GPU kart içerisine alınması ve eğitime görüntünün küçük bir kısımında başlaması ile hem GA'ın hem de PSO'ların çok daha hızlı bir şekilde doğru sonuca ulaşmalarının sağlanmasıdır.

Bu tez çalışması kapsamında önerilen yöntemler ile Kuadratik görüntü filtrelerin maske ağırlıkları başarılı bir şekilde eğitilmiş ve görüntü kalitesinden ödün vermeden eğitim süreleri ciddi anlamda azaltılmıştır. Bu çalışma Kuadratik görüntü filtreler gibi zor problemlerin sezgisel algoritmalarla hızlı bir şekilde çözüldüğünü göstermekte ve bu gibi çalışmalara ışık tutacağı düşünülmektedir. Ayrıca geliştirilen yöntemlerin tamamı diğer sezgisel algoritmalara da çok kolay bir şekilde entegre edilebilecek şekilde tasarlandığından birçok çalışmaya yol göstereceği düşünülmektedir.

Bu tez çalışmasında, tek bir görüntü kullanılarak maske ağırlıkları eğitilmiştir. Bir görüntü yerine bir görüntü grubu kullanılarak maske eğitimleri yapılabilir. Sebebi ise görüntüler bir birinden farklıdır, kimisinde detaylar çok fazladır, kimisinde daha azdır, kimisinde ise detaylar belirli bölgelerde yoğun belirli bölgelerde çok daha az olabilir. Tek görüntü ile yapılan eğitimler sonucunda elde edilen maske ağırlıkları bazı görüntüleri başarılı bir şekilde filtrelerken aynı başarıyı başka görüntülerde gösteremeyebilir. Görüntü grubunda yapılan eğitimlerde elde edilen maske ağırlıkları birçok görüntünün filtrelenmesinde daha fazla başarım gösterebilir.

Ayrıca bu tez çalışması boyunca donanımsal hızlandırma amacıyla tek bir grafik kart kullanılmıştır. Bu grafik kartların sayısı arttırılarak eğitim süreleri çok daha kısaltılabilir. Hatta eğitim süreleri CPU üzerinde çalışan seri programın çalışma sürelerine ulaştırıldığında artık eğitimin daha fazla hızlandırılmasının bir anlamı kalmaz. Bu durumda CPU üzerinde çalışan seri programların da bir kısmı GPU üzerine taşınarak daha fazla hızlandırma elde edilebilir.

Uygunluk değerinin hesaplanma yöntemi kullanılan optimizasyon algoritmasının başarımını doğrudan etkilediği için yeni önerilecek uygunluk hesaplama yaklaşımları ile daha yüksek gürültü temizleme başarımları elde edilebilir.

KAYNAKLAR

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. 2007.

[2] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, "Digital Image Processing Using Matlab," *Education*, vol. 624, no. 2. p. 609, 2004.

[3] V. Hlavac, "Fundamentals of Image Processing," in *Optical and Digital Image Processing: Fundamentals and Applications*, pp. 71–96, 2011.

[4] R. Jennane, A. Almhdie-Imjabber, R. Hambli, O. N. Ucan, and C. L. Benhamou, "Genetic algorithm and image processing for osteoporosis diagnosis," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 5597–5600, 2010.

[5] A. Nandal, V. Bhaskar, and A. Dhaka, "Contrast-based image enhancement algorithm using grey-scale and colour space," *IET Signal Process.*, vol. 12, no. 4, pp. 514–521, Jun. 2018.

[6] Y. Liu, H. Yan, S. Gao, and K. Yang, "Criteria to evaluate the fidelity of image enhancement by MSRCR," *IET Image Process.*, vol. 12, no. 6, pp. 880–887, Jun. 2018.

[7] S. Lee, D. Kim, and C. Kim, "Ramp Distribution-Based Image Enhancement Techniques for Infrared Images," *IEEE Signal Process. Lett.*, vol. 25, no. 7, pp. 931–935, Jul. 2018.

[8] K. Kim, S. Kim, and K.-S. Kim, "Effective image enhancement techniques for fog-affected indoor and outdoor images," *IEEE 17th Int. Conf. Parallel Distrib. Syst.*, vol. 12, no. 4, pp. 465–471, Apr. 2018.

[9] S. Tan and Z. Shen, "Hybrid Problem-Based Learning in Digital Image Processing: A Case Study," *IEEE Trans. Educ.*, vol. 61, no. 2, pp. 1–9, May 2017.

[10] E. Kalali and I. Hamzaoglu, "Low complexity 2D adaptive image processing algorithm and its hardware implementation," *IEEE Trans. Consum. Electron.*, vol. 63, no. 3, pp. 277–284, Aug. 2017.

[11] Y. Pan, H. Liao, J. Li, X. Liu, and W. Zhu, "Improved Image Processing Algorithms for Microprobe Final Test," *IEEE Trans. Components, Packag. Manuf. Technol.*, vol. 8, no. 3, pp. 499–505, 2018.

[12] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng, “Graph Spectral Image Processing,” *Proceedings of the IEEE*, vol. 106, no. 5. pp. 907–930, May-2018.

[13] Z. Qin, J. Weng, Y. Cui, and K. Ren, “Privacy-Preserving Image Processing in the Cloud,” *IEEE Cloud Comput.*, vol. 5, no. 2, pp. 48–57, Mar. 2018.

[14] R. Bernstein, M. Moore, and S. Mitra, “Adjustable quadratic filters for image enhancement,” in *Image Processing, 1997. Proceedings., International Conference on*, pp. 287–290, 1997.

[15] G. Ramponi, “Edge extraction by a class of second-order nonlinear filters,” *Electron. Lett.*, vol. 9, no. 22, pp. 482–484, 1986.

[16] G. Ramponi and G. L. Sicuranza, “Quadratic Digital Filters for Image Processing,” *IEEE Trans. Acoust.*, vol. 36, no. 6, pp. 937–939, Jun. 1988.

[17] N. V. Filters, “A Computational Method for the Design of,” *IEEE Trans.*, vol. 35, no. 9, pp. 1095–1102, 1988.

[18] Y. Zhou, K. Panetta, and S. Agaian, “Mammogram enhancement using alpha weighted quadratic filter,” in *Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society: Engineering the Future of Biomedicine, EMBC 2009*, pp. 3681–3684, 2009.

[19] M. Kanamadi, V. Waghamode, and S. Bandekar, “Alpha Weighted Quadratic Filter Based Enhancement for Mammogram,” in *Proceedings of International conference on “Emerging Research in Computing, Information, Communication and Applications” (ERCICA)*, pp. 68–74, 2013.

[20] G. L. Sicuranza, “Nonlinear Digital Filter Realization by Distributed Arithmetic,” *IEEE Trans. Acoust.*, vol. 33, no. 4, pp. 939–945, 1985.

[21] M. B. Meenavathi and K. Rajesh, “Volterra Filtering Techniques for Removal of Gaussian and Mixed Gaussian-Impulse Noise,” *Int. J. Electr. Robot.*, vol. 1, no. 2, pp. 1–7, 2007.

[22] G. Jothilakshmi and E. Gopinathan, “Mammogram Enhancement Using Quadratic Adaptive Volterra Filter A Comparative Analysis In Spatial And Frequency Domain,” 2006.

[23] S. K. Mitra, “Image processing using quadratic volterra filters,” in *Computers and Devices for Communication (CODEC*, pp. 1–2, 2012.

[24] V. S. Hari, R. V. P. Jagathy, and R. Gopikakumari, “Enhancement of calcifications in mammograms using volterra series based quadratic filter,” in *2012 International Conference on Data Science & Engineering (ICDSE)*, pp. 85–89, 2012.

[25] L. Thomas, G. Krishnan, R. A. Mol, and A. Roy, “Removal of Impulsive Noise from MRI Images using Quadratic Filter,” *Int. J. Eng. Res. Technol.*, vol. 3, no. 4, pp. 2220–2223, 2014.

[26] A. Pandey, A. Yadav, and V. Bhateja, “Design of new volterra filter for mammogram enhancement,” in *Advances in Intelligent Systems and Computing*, vol. 199 AISC, pp. 143–151, 2013.

[27] S. Asano, “Performance Comparision of FPGA,GPU and CPU in image Processing,” *Image Process.*, pp. 126–131, 2009.

[28] O. Fialka and M. Čadik, “FFT and convolution performance in image filtering on GPU,” in *Information Visualization, 2006. IV 2006. Tenth International Conference on*, pp. 609–614, 2006.

[29] B. Shi, S. Chen, F. Huang, C. Wang, and K. Bi, “The parallel processing based on CUDA for convolution filter FDK reconstruction of CT,” in *Proceedings - 3rd International Symposium on Parallel Architectures, Algorithms and Programming, PAAP 2010*, pp. 149–153, 2010.

[30] A. S. Keceli and A. B. Can, “GPU based brain segmentation method,” in *2011 IEEE 19th Signal Processing and Communications Applications Conference (SIU)*, 2011, pp. 258–261.

[31] A. Ono, K. Kondo, T. Inaba, T. Tsumura, and H. Matsuo, “A GPU-supported high-level programming language for image processing,” in *Proceedings - 7th International Conference on Signal Image Technology and Internet-Based Systems, SITIS 2011*, pp. 245–252, 2011.

[32] H. Badem, “Gpu ile Hızlandırılmış Bulanık Mantık Algoritmalarının Görüntü İşlemede Kullanılması,” Kahramanmaraş Sütçü İmam Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar ve Öğretim Teknolojileri Eğitimi, Yüksek Lisans Tezi, 2012.

[33] F. N. Iandola, D. Sheffield, M. J. Anderson, P. M. Phothilimthana, and K. Keutzer, “Communication-minimizing 2D convolution in GPU registers,” in *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings*, pp. 2116–2120, 2013.

[34] X. Chen, Y. Qiu, and H. Yi, “Implementation and performance of image filtering on gpu,” in *Intelligent Control and Information*, pp. 514–517, 2013.

[35] U. Çekmez, “İnsansız Hava Araçlarında Büyük Ölçekli Yol Planlama Problemlerinin GPU Üzerinde CUDA Yardımı İle Çözümü,” Hava Harp Okulu, Havacılık ve Uzay Teknolojileri Enstitüsü, Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi, 2014.

[36] P. Y. Koçak, “GPU Programlama ile Yüksek Performanslı Görüntü İşleme Uygulamaları,” İstanbul Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi, 2014.

[37] M. Koca, “Gerçek Zamanlı Görüntü Oluşumu için Kullanılan Gölgeleme Algoritmalarının CUDA Kütüphanesi ile Gerçekleştirilmesi,” Karabük Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği, Yüksek Lisans Tezi, 2014.

[38] H. Rahmani, “A Parallel HuffmanCoder on The CUDA Architecture,” Anadolu Üniversitesi, Fen Bilimleri Enstitütsü, Bilgisayar Mühendisliği, Yüksek Lisans Tezi, 2014.

[39] D. Akgün and P. Erdoğmuş, “GPU accelerated training of image convolution filter weights using genetic algorithms,” *Appl. Soft Comput.*, vol. 30, pp. 585–594, May 2015.

[40] H. Patel, “GPU accelerated real time polarimetric image processing through the use of CUDA,” in *Proceedings of the IEEE 2010 National Aerospace and Electronics Conference, NAECON 2010*, pp. 177–180, 2010.

[41] D. Akgün, “Paralel Görüntü Filtreleme Için Çok Çekirdekli Bilgisayar Üzerinde Başarim Analizi,” *İleri Teknol. Bilim. Derg.*, vol. 2, no. 1, pp. 76–83, 2013.

[42] F. Cabello, J. Leon, Y. Iano, and R. Arthur, “Implementation of a fixed-point 2D Gaussian Filter for Image Processing based on FPGA,” in *Signal Processing - Algorithms, Architectures, Arrangements, and Applications Conference Proceedings, SPA*, vol. 2015–Decem, pp. 28–33, 2015.

[43] C. Liu, Z. Shang, and Q. Chen, “An Adaptive Tone Mapping Algorithm Based on Gaussian Filter,” in *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pp. 374–379, 2016.

[44] P. Y. Hsiao, S. S. Chou, and F. C. Huang, “Generic 2-D gaussian smoothing filter for noisy image processing,” in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2007.

[45] J. J. Hwang and K. H. Rhee, “Gaussian filtering detection based on features of residuals in image forensics,” in *2016 IEEE RIVF International Conference on Computing and Communication Technologies: Research, Innovation, and Vision for the Future, RIVF 2016 - Proceedings*, pp. 153–157, 2016.

[46] D. Yıldırım, B. Dizdaroğlu, H. M. Bölümü, B. Mühendisliği, B. Karadeniz, and T. Üniversitesi, “Yönbağımsız ve Yönbağımlı Gauss Süzgeçleme Isotropic and Anisotropic Gaussian Filtering,” in *Eleco 2014 Elektrik-Elektronik-Bilgisayar ve Biyomedikal Mühendisliği Sempozyumu*, pp. 568–572, 2014.

[47] R. O. Julio, L. B. Soares, E. A. C. Costa, and S. Bampi, “Energy-efficient Gaussian filter for image processing using approximate adder circuits,” in *2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, pp. 450–453, 2015.

[48] K. Ito, “Gaussian filter for nonlinear filtering problems,” *Proc. 39th IEEE Conf. Decis. Control (Cat. No.00CH37187)*, vol. 2, no. 5, pp. 910–927, 2000.

[49] S. Liu, J. Zhang, and J. Chen, “Multi-focus image fusion using Gaussian filter and dynamic programming,” in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1182–1185, 2017.

[50] D. M. Weber and D. P. Casasent, “Quadratic Gabor filters for object detection,” *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 218–230, 2001.

[51] T. Jinshan, E. Peli, and S. Acton, “Image enhancement using a contrast measure in the compressed domain,” *Signal Process. Lett. IEEE*, vol. 10, no. 10, pp. 289–292, 2003.

[52] R. J. P. deFigueiredo and S. Matz, “Exponential nonlinear Volterra filters for contrast sharpening in noisy images,” in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 4, pp. 2263–2266, 1996.

[53] J. Mohammadi, K. Mirzaie, and V. Derhami, “Parallel genetic algorithm based on GPU for solving quadratic assignment problem,” in *Conference Proceedings of 2015 2nd International Conference on Knowledge-Based Engineering and Innovation, KBEI 2015*, 2016, pp. 569–572, 2015.

[54] K. de Jong, “Learning with Genetic Algorithms: An Overview,” *Mach. Learn.*, vol. 3, no. 2, pp. 121–138, 1988.

[55] S. Y. Fakhouri, “Identification of the Volterra kernels of nonlinear systems,” *IEE Proc.*, vol. 127–D, no. 6, pp. 296–304, 1980.

[56] R. D. Nowak and B. D. Van Veen, “Random and Pseudorandom Inputs for Volterra Filter Identification,” *IEEE Trans. Signal Process.*, vol. 42, no. 8, pp. 2124–2135, 1994.

[57] P. Venkatappareddy and B. Lall, “Linear in the parameter model for Homomorphic filter: Volterra series based approach,” in *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, pp. 650–654, 2016.

[58] S. Akyol and B. Alataş, “Güncel Sürü Zekâsı Optimizasyon Algoritmaları,” *Nevşehir Üniversitesi Fen Bilim. Enstitü Derg.*, vol. 1, no. 1, pp. 36–50, 2012.

[59] G. G. Emel and Ç. Taşkın, “Genetik Algoritmalar ve Uygulama Alanları,” *Uludağ Üniversitesi İktisadi ve İdari Bilim. Fakültesi*, vol. 21, no. 1, pp. 129–152, 2002.

[60] H. Yıldız, “Paralel Genetik Algoritma İle Sayısal Filtre Optimizasyonunun Karşılaştırmalı Analizi,” Kahramanmaraş Sütçü İmam Üniversitesi, Fen Bilimleri Enstitüsü, Elektrik Eğitimi Anabilim Dalı, Yüksek Lisans Tezi, 2013.

[61] A. Ouarda, “A Comparison of Evolutionary Algorithms : PSO , DE and GA for Fuzzy C-Partition,” *Int. J. Comput. Appl.*, vol. 91, no. 10, pp. 32–39, 2014.

[62] A. Tuncer, "Otonom araçlar için Yol Bulma Probleminin Genetik algoritmlar ve FPGA ile Çözüm ve Gerçekleştirilmesi," Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Elektronik ve Bilgisayar Eğitimi Anabilim Dalı, Doktora Tezi, 2013.

[63] B. Bolat, K. Erol, and C. Imrak, "Genetic algorithms in engineering applications and the Function of operators," *J. Eng. Nat. Sci.*, pp. 264–271, 2004.

[64] E. U. Ergül, "Çok amaçlı Genetik Algoritmalar: Temelleri ve Uygulamaları," Ondokuz Mayıs Üniversitesi, Fen Bilimleri Enstitüsü, Elektrik - Elektronik Mühendisliği Anabilim Dalı, Doktora Tezi, 2010.

[65] Ş. Doğan, G. Koca, H. Y.-K. F. ve Mühendislik, and U. 2017, "Arama Kurtarma Faaliyetlerinde Optimal Takım Dağıtımının Sağlanması İçin 3 Boyutlu Yüzeylere Genetik Algoritma Yönteminin Uygulanması," *Karaelmas Fen ve Mühendislik Derg.*, vol. 7, no. 2, pp. 577–585, 2017.

[66] O. Kalaycı, "Parça Toplayan Hareketli Robotlar İçin Genetik Algoritmalarla Yol Planlaması," Ege Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Doktora Tezi, 2009.

[67] M. Üstündağ, "Zayıf Radar Sinyallerinin Genetik Algoritmalar Kullanılarak Gürültüden Arındırılması," Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Elektrik - Elektronik Mühendisliği Anabilim Dalı, Doktora Tezi, 2012.

[68] V. Kachitvichyanukul, "Comparison of Three Evolutionary Algorithms: GA, PSO, and DE," *Ind. Eng. Manag. Syst.*, vol. 11, no. 3, pp. 215–223, 2012.

[69] D. Snyers and Y. Pétillot, "Image processing optimization by genetic algorithm with a new coding scheme," *Pattern Recognit. Lett.*, vol. 16, no. 8, pp. 843–848, 1995.

[70] D. J. Krusienski and W. K. Jenkins, "Particle swarm optimization for adaptive IIR filter structures," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, 2004, vol. 1, pp. 965–970.

[71] E. Assareh, M. A. Behrang, M. R. Assari, and A. Ghanbarzadeh, "Application of PSO (particle swarm optimization) and GA (genetic algorithm) techniques on demand estimation of oil in Iran," *Energy*, vol. 35, no. 12, pp. 5223–5229, 2010.

[72] M. A. Çavuşlu, C. Karakuzu, and S. Şahin, "Parçacık Sürü Optimizasyonu Algoritması ile Yapay Sinir Ağı Eğitiminin FPGA Üzerinde Donanımsal Gerçeklenmesi," *Politek. Derg.*, vol. 13, no. 2, pp. 83–92, 2010.

[73] S. K. Saha, S. Mukherjee, D. Mandal, R. Kar, and S. P. Ghoshal, "Gravitational search algorithm in digital FIR low pass filter design," in *2012 Third International Conference on Emerging Applications of Information Technology*, no. 1, pp. 52–55, 2012.

[74] K. K. Çevik and H. E. Koçer, "Parçacık Sürü Optimizasyonu ile Yapay Sinir Ağları Eğitimine Dayalı Bir Esnek Hesaplama Uygulaması," *Süleyman Demirel Üniversitesi Fen Bilim. Enstitüsü Derg.*, vol. 17, no. 2, 2013.

[75] P. Erdoğmuş, E. Y.-İ. T. B. Dergisi, and U. 2015, "Parçacık Sürü Optimizasyonu ile Kısıtsız Optimizasyon Test Problemlerinin Çözümü," *J. Adv. Technol. Sci.*, vol. 4, no. 1, pp. 14–22, 2015.

[76] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pp. 69–73, 1998.

[77] J. Owens and M. Houston, "GPU computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[78] S. Asano, T. Maruyama, and Y. Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," in *2009 International Conference on Field Programmable Logic and Applications*, 2009, pp. 126–131.

[79] C. S. Lin, C. W. Hsieh, H. Y. Chang, and P. A. Hsiung, "Efficient workload balancing on heterogeneous gpus using mixedinteger non-linear programming," *J. Appl. Res. Technol.*, vol. 12, no. 6, pp. 1176–1186, 2014.

[80] N. Corporation, "CUDA Zone | NVIDIA Developer," *CUDA toolkit & sdk*, 2018. [Online]. Available: https://developer.nvidia.com/cuda-zone. [Accessed: 20-Jun-2018].

[81] A. Bakırtaş, "GPU Üzerinde Yazılım Tabanlı anten Gerçeklenmesi," Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Telekomünikasyon Mühendilsiği Bölümü, Yüksek Lisans Tezi, 2015.

[82] NVIDIA, "NVIDIA Maxwell architecture," 2014. [Online]. Available: https://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_980_Whitepaper_FINAL.PDF. [Accessed: 14-Jun-2018].

[83] C. Toolkit, "CUDA Toolkit Documentation," *NVIDIA Dev. Zo. See https//docs.nvidia.com/cuda/maxwell-compatibility-guide/index.html (accessed 12/06/2018)*, 2018.

[84] S. Cook, *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*. 2013.

[85] "NVIDIA CUDA Compute Unified Device Architecture," 2007. [Online]. Available: http://web.cse.ohio-state.edu/~agrawal.28/788-su08/Papers/week2/Cuda.pdf. [Accessed: 27-May-2018].

[86] J. Ghorpade, "GPGPU Processing in CUDA Architecture," *Adv. Comput. An Int. J.*, vol. 3, no. 1, pp. 105–120, 2012.

[87] D. Kirk, “NVIDIA cuda software and gpu parallel computing architecture,” in *Proceedings of the 6th international symposium on Memory management - ISMM ’07*, pp. 103–104, 2007.

[88] H. Güler, “GPU ile Kumaş Hata Tespiti ve sınıflandırılması,” Kahramanmaraş Sütçü İmam Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar ve Öğretim Teknolojileri Eğitimi, Yüksek Lisans Tezi, 2013.

[89] A. S. Al-Hamoudi and A. Ahmed Biyabani, “Accelerating data mining with CUDA and OpenMP,” in *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, pp. 528–535, 2014.

[90] M. Broxvall, K. Emilsson, and P. Thunberg, “Fast GPU based adaptive filtering of 4D echocardiography,” *IEEE Trans. Med. Imaging*, vol. 31, no. 6, pp. 1165–1172, 2012.

[91] NVIDIA, “Cuda C Programming Guide,” *Programming Guides. https://docs.nvidia.com/cuda/cuda-c-programming-guide/*, 2017. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/. [Accessed: 14-Jun-2018].

[92] M. Osama Mahmoud Muhammad, “An Efficient Exploration Algorithm for Digital VLSI Systems and Its Implementation on CPU-GPU Heterogeneous Systems,” Minia University, Computer and Systems Engineering Department Faculty of Engineering, Master of Science, 2015.

[93] L. Nyland and S. Johns, “Understanding and using atomic memory operations,” in *4th GPU Technology Conf.(GTC’13), March*, pp. 1–61, 2013.

[94] J. Balfour, “CUDA threads and atomics,” 2011.

[95] G. Schaefer and M. Stich, “UCID - An Uncompressed Colour Image Database,” *SPIE, Storage Retr. Methods Appl. Multimed.*, vol. 5307, pp. 472–480, 2003.

[96] A. Tanchenko, “Visual-PSNR measure of image quality,” *J. Vis. Commun. Image Represent.*, vol. 25, no. 5, pp. 874–878, 2014.

[97] G. J. Wang, C. Xie, S. Chen, J. J. Yang, and M. Y. Yang, “Random matrix theory analysis of cross-correlations in the US stock market: Evidence from Pearson’s correlation coefficient and detrended cross-correlation coefficient,” *Phys. A Stat. Mech. its Appl.*, vol. 392, no. 17, pp. 3715–3730, 2013.

[98] D. Brunet, E. R. Vrscay, and Z. Wang, “On the mathematical properties of the structural similarity index,” *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1488–1499, 2012.

[99] Ü. Kaş and E. Tanyıldızı, “Performance Analysis of Eulerian Colour and Motion Magnification Techniques,” *Afyon Kocatepe Univ. J. Sci. Eng.*, vol. 17, no. 2, pp. 506–515, Aug. 2017.

[100] Barış Demirci, "Görüntü Steganografi Medotları ve Performanslarının Karşılaştırılması," Selçuk Üniversitesi, Fen Bilimleri Enstitüsü, Bilişim Teknolojileri Mühendisliği, Yüksek Lisans Tezi, 2016.

[101] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[102] M. C. Çatalbaş and A. Gülten, "Bilgisayar Tomografi İmgeleri için Geliştirilmiş Ters Mesafe Ağırlıklandırma Yöntemi Tabanlı Süper Çözünürlük," *Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Derg.*, vol. 2018, no. 2018, Apr. 2018.

[103] Sara Behjat Jamal, "Bulanık Mantık Uyarlamalı İki Yanlı (Bilateral) Görüntü Filtresi Tasarımı," Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği, Yüksek Lisans Tezi, 2016.

[104] A. Arcuri and L. Briand, "A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Softw. Test. Verif. Reliab.*, vol. 24, no. 3, pp. 219–250, May 2014.

ÖZGEÇMİŞ

Süleyman UZUN, ilk, orta ve lise eğitimini Mersin'de tamamladı. 2002 yılında Mersin Endüstri Lisesi'nden mezun oldu. 2003 yılında başladığı Gazi Üniversitesi Elektronik Öğretmenliği Bölümü'nü 2008 yılında bitirdi. 2008 yılında Gazi Üniversitesi Bilişim Enstitüsünde Bilgisayar ve Elektronik Eğitimi Bölümü'nde yüksek lisans eğitimine başladı. 2009 yılında Erzincan Üniversitesi'nde Öğretim Görevlisi, 2013 yılında Bilecik Şeyh Edebali Üniversitesi Bilgi İşlem Daire Başkanlığı Ağ ve Sistem Yönetimi Şube Müdürlüğü'nde uzman, 2017 yılında Bilecik Şeyh Edebali Üniversitesi Açık ve Uzaktan Öğrenme Uygulama ve Araştırma Merkezi'nde Sistem Sorumlusu olarak çalışmaya başladı. Yüksek Lisans eğitimini 2011 yılında tamamladı. 2014 yılında Sakarya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar ve Bilişim Mühendisliği Bölümü'nde doktora eğitimine başladı. Halen Bilecik Şeyh Edebali Üniversitesinde Öğretim Görevlisi olarak çalışmaktadır.