T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

GÖRÜNTÜ İŞLEME VE YAPAY ZEKA YÖNTEMLERİYLE SINIR BELİRLEME

YÜKSEK LİSANS TEZİ

Hakan ÜNAL

Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Programı

AĞUSTOS, 2021

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

GÖRÜNTÜ İŞLEME VE YAPAY ZEKA YÖNTEMLERİYLE SINIR BELİRLEME

YÜKSEK LİSANS TEZİ

Hakan ÜNAL
(Y1813.010034)

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Bilim Dalı

Tez Danışmanı: Prof. Dr. Zafer ASLAN

AĞUSTOS, 2021

ONAY FORMU

ONUR SÖZÜ

Yüksek Lisans tezi olarak sunduğum “Görüntü İşleme ve Yapay Zeka Yöntemleriyle Sınır Belirleme” adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Kaynakça ’da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (26/08/2021)

Hakan ÜNAL

ÖNSÖZ

Çalışmam boyunca beni her konuda destekleyen aileme, Dr. Öğr Üyesi Ahmet Gürhanlı'ya, Dr. Öğr. Üyesi Adem Özyavaş'a, Prof. Dr. Ali Güneş'e, akademik bilgisi ve değerli yönlendirmeleri ile bu tezi bitirmemi sağlayan tez danışmanım Prof. Dr. Zafer Aslan'a, teşekkürü bir borç bilirim.

Ağustos, 2021 HAKAN ÜNAL

GÖRÜNTÜ İŞLEME VE YAPAY ZEKA YÖNTEMLERİYLE SINIR BELİRLEME

ÖZET

Bu çalışma Görüntü İşleme ve Yapay Zekâ Yöntemleriyle Sınır Belirleme problemini ele almaktadır. Sınır belirleme probleminin çözülmesi nesne tanıma, nesne sayma, görüntünün belli bir bölümünün çıkarılması ya da sınırı belli nesnenin arkasına başka bir görüntü eklenmesi gibi işlemler için kullanılmaktadır. Görünü İşleme görüntüyü işlemenin bilgisayar ile yapılması olup, bilgisayarın görüntüyü görme şekline "Bilgisayarla Görü" denilmektedir. Yapay Zekâ kısaca insan beyninin taklit edilmesidir. Bu durumda insan beynini görüntüyü işleme için kullandığı gibi bilgisayar 'da Yapay Zekâ ile Görüntüyü İşleme için kullanılabilmektedir. Makine Öğrenmesi, insanın geçmiş deneyimlerden faydalanma yetisinin taklit edilmesidir. Derin Öğrenme Yapay Sinir Ağları gibi Yapay Zekâ yöntemlerinin daha derin hali olarak kabul edilmektedir. Bu çalışmada kullanılan görseller görüntü işleme çalışmalarında sıkça kullanılan Cameraman ve Lenna görselleridir. Yöntem olarak Canny, Auto Canny, Sobel, Laplacian, Prewitt, Scharr, Holistically-Nested Edge Detection ve Richer Convolutional Features for Edge Detection yöntemleri kullanılmıştır. Yöntem uygulanırken Ortalama, Gaussian ve Medyan filtrelemeleri kullanılmış olup bu filtreler ile daha keskin sınırlar elde edilmeye çalışılmıştır. Programlama dili olarak Python dili kullanılmış olup derleyici olarak PyCharm derleyicisi tercih edilmiştir. Tezin uygulama bölümünde, Python 3.8.5 ve PyCharm 2021.1 versiyonları kullanılmıştır. Python ile beraber görüntü işleme kütüphanesi olarak OpenCV ve matematiksel işlevler için NumPy kütüphaneleri kullanılmıştır. Holistically-Nested Edge Detection yönteminde önceden eğitilmiş model için Caffe çatısı uygulanmıştır. Richer Convolutional Features for Edge Detection yönteminde PyTorch kütüphanesi kullanılmıştır. Versiyon olarak OpenCV 4.0.1 ve NumPy 1.19.2 versiyonları kullanılmıştır. Python ile yapılan uygulamanın tasarımı için PyQt5 kullanılmıştır. Tasarım için Qt Designer programı, Qt ve Qt Designer 5.6.0 versiyonu kullanılmıştır.

Tez araştırması sonucunda uygulanan yöntemlerden elde edilen bulgular karşılaştırılmıştır. Filtreleme yöntemlerinin sınır belirleme işlemlerinde büyük kolaylık sağladığı görülmüştür. Yapay Zekâ yöntemlerinin gelişmesinin Görüntü İşleme yöntemleri üzerinde etkili olduğu saptanmıştır. RCF yönteminde görüntü işleme yöntemlerine göre daha iyi sonuçlar elde edilmiş fakat görüntü işleme ve HED yöntemlerine göre daha bulanık sonuçlar elde edilmesine rağmen görüntü üzerindeki sınırların daha iyi belirlendiği görülmüş ve bu durumdan dolayı RCF yönteminin diğer yöntemlere göre daha başarılı olduğu görülmüştür.

Anahtar Kelimeler: Görüntü İşleme, Yapay Zekâ, OpenCV, Derin Öğrenme, Python

CONTOUR DETECTION WITH IMAGE PROCESSING AND ARTIFICIAL INTELLIGENCE METHODS

ABSTRACT

This study deals with the problem of Contour Detection with Image Processing and Artificial Intelligence Methods. Solving the Contour Detection problem is used for operations such as object recognition, object counting, removing a certain part of the image or adding another image behind the object with a certain contour. Image Processing methods are the processing of the image by the computer and the way the computer sees the image is called Computer Vision. Artificial Intelligence is simply an imitation of the human brain. In this study, the human brain can be used for image processing, as well as the computer can be used for image processing, as well as the computer can be used for image processing with Artificial Intelligence. In short, Machini Learning is the imitation of human ability to benefit from past experiences. Deep Learning can be briefly called as the deeper version of Artificial Intelligence methods such as Artificial Neural Networks. The images used in this study are Cameraman and Lenna images, which are frequently used in image processing studies. Canny, Auto Canny, Sobel, Laplacian, Prewitt, Holistically-Nested Edge Detection and Richer Convolutional Features for Edge Detection methods were used as methods. While applying the method, Averaging, Gaussian and Median filters were used and sharper contours were tried to be obtained with these filters. Python language was used as programming language and PyCharm compiler was preferred as compiler. Python 3.8.5 and PyCharm 2021.1 versions were used as versions. OpenCV as image processing library and NumPy libraries for mathematical functions were used together with Python. In the Holistically-Nested Edge Detection method, the Caffe framework was used for the pre-trained model. PyTorch library is used in Richer Convolutional Features for Edge Detection method. OpenCV 4.0.1 and NumPy 1.19.2 versions were used as versions. PyQt5 was used for the design of the application made with Python. The Qt Designer program was used for the design, and Qt and Qt Designer version

5.6.0 were used. As a result, these methods were compared. It has been seen that filtering methods facilitate the contour detection processes. As a conclusion, the development of Artificial Intelligence methods is effective on Image Processing methods. In the RCF method, better results were obtained than the image processing methods, but although more blurred results were obtained compared to the image processing and HED methods, it was seen that the contours on the image were determined better, and because of this, it was seen that the RCF method was more successful than other methods.

Keywords: Image Processing, Artificial Intelligence, OpenCV, Deep Learning, Python

İÇİNDEKİLER

KISALTMALAR LİSTESİ

CAT : Computed Axial Tomography

CNN : Convolutional Neural Network

CPU : Central Process Unit

ÇKA : Çok Katmanlı Algılayıcılar

DSA : Derin Sinir Ağları

GİB : Grafik İşlemci Birimi

GPU : Graphics Processing Unit

HED : Holistically-Nested Edge Detection

LVQ : Learning Vector Quantization

MİB : Merkezi İşlem Birimi

MLP : Multi Layer Perceptron

RCF : Richer Convolutional Features for Edge Detection

YSA : Yapay Sinir Ağları

ŞEKİLLER LİSTESİ

ÇİZELGELER LİSTESİ

I. GİRİŞ

Gelişen teknoloji ile birlikte insanların görme ve gördükleri üzerinden yorumlama özelliklerini bilgisayar teknolojisi ile birleştirme çalışmalarına başlanmış olup Bilgisayarla Görü (Computer Vision) olarak adlandırılmıştır. Elde edilen görüntüler üzerinde insan görsel sisteminin yapabileceği işlemlere benzer işlemleri ve bu işlemleri otomatikleştirme çalışmalarına başlanmış olup Görüntü İşleme (Image Processing) olarak adlandırılmıştır. Görüntü İşleme üzerine çeşitli teknikler geliştirilmiş olup Sınır Belirleme (Contour Detection) tekniği bu tekniklerden biri olmaktadır. Sınır Belirleme tekniği kullanılarak nesnelerin algılanması sağlanmakta olup bu nesneler yorumlanarak uzunluk, büyüklük, renk gibi özelliklerinin belirlenmesi çalışmaları yapılmaktadır.

Yapay Zekâ, insan zekasını bilgisayara aktarma çalışmaları olmaktadır. Yapay Zekâ yöntemleriyle Görüntü İşleme tekniklerini geliştirme çalışmaları yapılmaktadır. Bu çalışmalar insan zekâsından ve görme sisteminden esinlenerek yapılmaktadır. Derin Öğrenme (Deep Learning), Yapay Zekâ tekniklerinin alt konularından biri olup insanın öğrenme becerisine benzer bir sistemi ele almakta ve çok fazla veri üzerinden öğrenme işlemi gerçekleştirilmektedir. Makine Öğrenimi (Machine Learning), insanın öğrenme becerisini makinelere aktarma çalışmaları ile, bilgisayar sistemlerinin önceki deneyimlerinden öğrenmelerini ve verilen bir görev için davranışlarını geliştirmelerini sağlayan bir Yapay Zekâ tekniklerinin alt konularından biri olmaktadır.

Yapay Zekâ, bilgisayarların makine öğrenmesi, doğal dil işleme, dil sentezi, bilgisayarla görme, robotik, sensör analizi, optimizasyon ve simülasyon dahil olmak üzere insan davranışını taklit etmesini amaçlayan herhangi bir tekniktir, (Nguyen, ve diğerleri, 2019).

Makine Öğrenmesi, bilgisayar sistemlerinin önceki deneyimlerden öğrenmelerini ve verilen bir görev için davranışlarını geliştirmelerini sağlayan bir Yapay Zekâ teknikleri alt kümesi olup, Makine Öğrenmesi teknikleri arasında Destek Vektör Makineleri, Karar Ağaçları, Bayes Öğrenmesi, K-Kümelenmesi, Birlik Kuralı Öğrenmesi, Regresyon, Sinir Ağları vb. sayılabilir, (Nguyen, ve diğerleri, 2019).

Yapay Sinir Ağları, biyolojik sinir ağlarından esinlenen Makine Öğrenmesi tekniklerinin bir alt kümesi olup genellikle yapay nöronlar adı verilen ve katmanlar halinde organize edilmiş bağlı birimler topluluğu olarak tanımlanmaktadır, (Nguyen, ve diğerleri, 2019).

Derin Öğrenme, hesaplamalı çok katmanlı yapay sinir ağlarını uygulanabilir kılan yapay sinir ağı alt kümesi olup, derin sinir ağları, evrişimli sinir ağları, tekrarlayan sinir ağları, derin inanç ağları, üretken çekişmeli ağları, derin kalıntı ağları ve daha pek çok ağı kapsar, (Nguyen, ve diğerleri, 2019).

II. LİTERATÜR

(Meymandi, 2018), Kızıl Ötesi görüntüler üzerinde kenar belirleme yöntemini kullanmıştır. Bir Kümeleme algoritması ile bölümlere ayırma işleminden sonra ayrılmış bölümler arasında ilgi bölgesi çıkarmak için Sinir Ağı algoritmasını kullanmıştır. Son adımda, ilgi bölgesinde Morfolojik işletmecileri kenarları çıkarmak için kullanmıştır. K-Ortalamalar Kümeleme ve Ortalama Kaydırma yöntemleri ile görüntüyü bölmüş olup kümelerin özelliklerini Sinir Ağının girişleri olarak kullanmıştır.

(Mahmood, 2014), orijinal görüntüleri temiz, gürültülü ve gürültüden arındırılmış etkilere büründürerek oluşturulan çeşitli formları kullanarak beş kenar algoritmasını değerlendirmiş olup en iyi eşik değeri belirlemiştir.

(Oktay, 2011), çeşitli tıbbi imgeler için sınır bulma ve nesne saptama işlemlerini farklı önsel bilgilerin eklenmesi ile gerçekleştiren makine öğrenmesi tabanlı yöntemler sunmuştur. Sunulan yöntemlerde, önsel bilgi hiyerarşik bir şekilde lokal ve global olarak ayırmıştır. Böylece farklı tiplerdeki bilgileri ekleyerek daha verimli, modüler ve etkin bir şekilde sınır bulma ve nesne saptama işlemlerini gerçekleştirmeyi amaçlamıştır.

(Aktaş, 2020), gerçekleştirdiği çalışmada görme engelli bireyler için derin öğrenme yöntemleri kullanılarak dokunsal parke yüzeylerinin tespit edilmesini amaçlamıştır. Bu çalışmada derin öğrenme yöntemleri ile görüntü işleme algoritmalarını birlikte kullanmıştır. Nesne tespit etme modellerinden biri olan YOLO-V3 modeli ile DenseNet modelini birleştirilerek YOLOV3-Dense modeli oluşturmuştur. YOLO-V2, YOLO-V3 ve YOLOV3-Dense modelleri oluşturmuş ve içerisinde 4580 etiketli görsel bulunan Marmara Dokunsal Parke Yüzeyi (MDPY) veri seti üzerinde ayrı ayrı eğitilip performansları test veri seti üzerinde birbirleri ile karşılaştırmıştır. %89 F1-skor, %92 ortalama hassasiyet ve %81 IoU değerleri ile YOLOV3-Dense modelinin dokunsal parke yüzeyi tespit etmede diğer modellerden daha iyi olduğunu gözlemlemiştir.

(Çakar, 2018), akciğer bilgisayarlı tomografi kesit görüntüleri üzerinden akciğerde bulunan birimlere sınıflandırma işlemi gerçekleştirmiştir. Sistem; görüntü işleme ve yapay zekâ olmak üzere iki ana başlık altında oluşturmuştur. Görüntü işleme bölümünde ön işlemler ve aday nodüllerin tespit edilmesi gerçekleştirmiştir. Juxtapleural nodüller çekme faktörü kullanılarak tespit etmiştir. Elde edilen aday nodüllerin morfolojik özellikleri çıkarılarak yapay sinir ağını eğitmiştir. Yapay sinir ağında elde edilen katsayılar genetik algoritma ile iyileştirilerek sınıflandırma işlemini gerçekleştirmiştir. Sınıflandırma; Malign, benign, bronşlar ve damarlar, yanlışlar olarak 4 sınıfı ayırmıştır. Bu tez çalışmasında kullanılan veri seti, "Sakarya Eğitim ve Araştırma Hastane" 'sinde PET-CT bölümünden temin etmiştir. Veri setinde 36 hasta için 153 görüntüden 304 aday nodül bölgesi elde etmiştir. Yapılan deneysel çalışma sonucunda elde edilen en iyi sınıflandırma, yapay sinir ağında %92,105 doğruluk oranı elde etmiştir. Sınıflar için kullanılan özellikler için ve AUC (Area Under Curve) değeri 0,9847 elde etmiştir. Genetik algoritma ile iyileştirme yapıldığında %94,407 doğruluk oranı elde etmiştir. Sınıflar ROC (Receiver Operating Characteristic) eğrileri kullanılarak her sınıf için optimum kesim noktaları elde etmiştir. Elde edilen sonuçlar akciğer nodülleri için radyografik sınıflandırma başarımına olumlu katkı sağlamıştır. Oluşturulan sistem literatürdeki yapılan çalışmalar ile karşılaştırılabilecek düzeyde başarı sağlamıştır.

(Boztoprak, 2014), çamur hacim indeksi (ÇHİ) atıksu arıtma tesislerinin performans takibi yapılabilmesi için günlük izlenmesi gereken bir parametre olup bu parametrenin yapay zekâ ve görüntü işleme teknikleri kullanılarak tahmin edilmesini amaçlamıştır. Bunun için aktif çamur numuneleri Konya Kentsel Atıksu Arıtma Tesisi aktif çamur ünitesinin havalandırma tankından almıştır. Yöntem olarak uzaysal frekans ve hücresel sinir ağları (HSA) birlikte kullanarak yeni bir bölütleme yöntemi sunmuştur. Sonuç olarak, verileri rastgele karıştırılıp 5-kat çapraz doğrulama yöntemi ile eğitim ve test verileri elde etmiştir. Bu eğitim ve test verileri yapay sinir ağına uygulamıştır. Korelasyon katsayılarının ortalamaları LM-YSA, GA-YSA ve ARI-YSA için sırasıyla r=0.896, r=0.902 ve r=0.915 hesaplamıştır. Elde edilen sonuçlar görüntü işleme ve yapay sinir ağlarının ÇHİ tahmininde başarılı bir yöntem olarak uygulanabileceğini göstermiştir.

(Taşçı, 2013), akciğer BT (Bilgisayarlı Tomografi) kesit görüntüleri üzerinden juxtapleural nodül bölgelerinin otomatik tespitinin ve tanılamasının sağlanmasına

yönelik bir yöntem ve sistem geliştirmiştir. Tez kapsamında, akciğer BT kesit görüntülerini içeren LIDC verisetinden yararlanmıştır.

(Karakoç, 2011), bu çalışmada görüntü içinde görüntü aramayı hızlı ve verimli bir biçimde gerçekleştirme çalışmaları yapmıştır. Bu çalışma, bir görüntünün bu görüntüyü kapsayan başka bir görüntüde tespit edilmesine yönelik çalışmaları içermiştir. Bu çalışmada hızlı görüntü eşleme yöntemleri, paralel programlama ve akıllı arama algoritmaları tekniklerine dayanan bütünleşik bir yöntem önermiştir.

(Yüksel, 2010), bu çalışmanın amaçlarından biri, halkasal eksantrik borularda iki fazlı ve çok fazlı akışlarda basınç kayıplarını ve akış örüntülerini deneysel olarak gözlemiştir. İkinci amaç ise dijital görüntü işleme teknikleri ile bu akışlardaki sıvı hacimsel oranını ve kesinti yoğunluğunu tespit etmiştir. Çalışmanın son amacı da yatay halkasal ortamda iki fazlı (su ve hava) akış için geleneksel mekanistik modeller yerine yapay zekâ teknikleri kullanarak akış örüntüsü, sıvı hacimsel oranı ve basınç kaybı için tahmin modelleri geliştirmiştir. Bu çalışmada yapay zekâ tekniklerinden en yakın komşu algoritması, yapay sinir ağları ve karar ağaçları kullanmıştır. Yapay zekâ modellerinde akışın genelleştirilmesi için sıvı ve gaz fazları için yüzeysel Reynolds sayıları kullanmıştır. Sonuçlar, geri yayılımlı sinir ağının akıl özellikleri modellerinde en iyi sonucu verdiğini göstermiştir. Su ve hava akışında, 7 adet akış örüntüsü 10% hassasiyetle, sıvı hacimsel oranı ve basınç kaybı ise sırasıyla 14.4% ve 3.8% (0.0274 psi) ortalama hata payı ile tespit etmiştir.

(Baykan, 2010), yapılan tez çalışmasında kayaç yapıcı minerallerin başında gelen kuvars, muskovit, biyotit, klorit ve opak minerallerinin tespit edilmesi üzerine çalışmıştır. Mineral tespit çalışmalarında kullanılan piksel değerleri hem tek nikol, hem de çift nikol görüntülerden almıştır. Yapılan tüm çalışmalarda Çok Katmanlı Yapay Sinir Ağı (ÇKYSA) kullanmıştır. Mineral tanıma için kullanılan ilk dijital görüntüler, James Swift marka mikroskoba bağlı Videolab kamera ile Inca Programı ile elde etmiştir. Görüntüler 450x370 piksel boyutlarında RGB formatında renkli görüntülerdir. Toplam 325 adet piksel değeri RGB, HSV ve L *a*b* renk uzaylarında ayrı ayrı alınarak, mineral etiketleme çalışmaları yapmıştır. Mineral etiketlemede RGB renk uzayında ortalama %89,53; HSV'de %87,5; L*a*b*'de %89,59; RGB ve HSV renk uzayları birlikte kullanıldığında %87,45 başarı elde etmiştir. Aynı dijital görüntülerden alınan toplam 400 adet piksel değeri ile RGB renk uzayında mineral sınıflandırma çalışmaları da yapmıştır. Sınıflandırmada minerallerin tümü için

ortalama %93,86 başarı elde etmiştir. Her mineralin ayrı ayrı sınıflandırma başarılarını tespit etmek içinse, paralel çalışan ÇKYSA mimarileri kullanmıştır. Sonuçta kuvars, muskovit, biyotit, klorit, opak için elde edilen sınıflandırma başarıları sırasıyla %90,67; %96,16; %93,91; %92 ve %97,62'dir.

III. YÖNTEM

A. Görüntü İşleme

Görüntü işleme, verilerin kamera, tarayıcı vb. cihazlar yardımıyla görüntülerin alınıp değerlendirme işleminden sonra, bilgisayarların veya başka bir aracın okuyabileceği bir şekilde dönüştürülme işlemine tabi tutulması veya dijital ortamdan başka bir dijital ortama aktarılma işlemidir, (Aksoy B. , 2021; Gonzalez & Woods, 2002; Russ, 2016; Neuman, Sapirstein, Shwedyk, & Bushuk, 1989). Görüntü işleme yöntemlerinde elde edilen görüntüler işlenerek yorumlanmaktadır. Görüntü işleme için görüntü üzerinde bozulmalar varsa düzeltilmesi için işlemler yapmak gerekmekte olup bu düzeltme işleme için çeşitli yöntemler geliştirilmiştir. Görüntü üzerinde gereksiz kısımlar var ise bu kısımların yok edilmesi için çeşitli yöntemler geliştirilmiş olup örneğin gerekli kısımları sınır belirleme yöntemi kullanarak algılayıp diğer kısımları keserek gereksiz kısımlar silinebilmektedir. Görüntü işleme temelde üç adımda incelenmekte olup bu adımlar çeşitli cihazlarla görüntünün elektronik ortama aktarılması, görüntü üzerinde analiz işlemleri yapılarak işlenmesi ve işlenen görüntünün veri raporunun ve çıktısının alınmasıdır, (Neuman, Sapirstein, Shwedyk, & Bushuk, 1989; Aksoy B. , 2021). Görüntü işleme adımlarından analiz adımında görüntü üzerindeki gürültünün (görüntü bulanıklığı, netlik, kötü görüntü) azaltılması amaçlanmaktadır. Bu amaçla görüntülere giriş ve çıkış görüntülerinin gerçekliği filtreleme ile sağlanan düşük seviye, görüntülerde yer alan nesnelerin tanınması ve sınıflandırılmasında bölme ve tanıma işlemlerinin olduğu orta seviye ve görüntülerde yer alan nesneleri tanımada görüntülerin analiz edilmesini içeren yüksek seviye işlemler uygulanmaktadır (Bellaire, Talmi, Oezguer, & Koschan, 1998; Aksoy B. , 2021). Görüntü işleme analog ve dijital görüntü işleme yöntemi olmak üzere iki tür yöntem kullanmakta olup analog ve dijital görüntü işleme verinin geçmesi gereken ön işleme, geliştirme ve görüntüleme, bilgi ve çıkarımı aşamalarından oluşmaktadır (Rossum, 1999; Aksoy B. , 2021). Görüntü işlemenin resim veya videoda diyafram ayarlarının doğru ayarlanması ile uygun ışık ve renk kalitesinin ayarlanması, güvenlik taramaları için renkleri zenginleştirme, netliği artırma, görüntüyü verimli bir şekilde depolama ve aktarma, görüntü boyutu, renk, tonlama ayarları, barkod okuma, plaka

okuma, yazı karakterlerini metne dönüştürme, ortamdaki kişi sayısını bulabilme, otomatik yüz gizleme, üç boyutlu görüntülerin hazırlanması, diyafram ayarı, sanal dünya ile gerçek dünya görüntülerinin birleştirilmesi, medikal alanda 3D yazıcı ile oluşturulacak nesnenin görüntüsünün analizi ve basılması gibi birçok uygulama alanı mevcuttur, (Kuchling, 2021; Aksoy B. , 2021; Dubois, 2007). Görüntü işleme, bir görüntüyü sayısal hale dönüştürdükten sonra elde edilen verilere matematiksel işlemler ve farklı yöntemler uygulanarak çeşitli çıkarımlar oluşturma ya da yeni görüntüler üretme tekniği olup görüntü işleme tekniği, görüntünün netliğini arttırma, ışık değerlerini değiştirme ve görüntü üzerinde bulunan herhangi bir nesnenin tanımlanabilmesi gibi birçok amaçla kullanılmakta olup özellikle araç içi otomasyon sistemlerinden, robot uygulamalarında, askeri uygulamalardan tarım uygulamalarına kadar birçok alanda yaygın olarak kullanılmaktadır, (Samtaş & Gülesin, 2011; Aksoy B. , 2021).

1. Görüntünün Tanımı

Günümüzde hızla gelişmekte olan görüntü işleme, bir görüntüyü sayısal forma dönüştürerek nesne tanıma, görüntü iyileştirme gibi işlemleri gerçekleştirebilmek için kullanılan bir yöntemdir, (Gonzalez & Woods, 2002; Aksoy B. , 2021). Ayrıca görüntü işleme yöntemleri ile iyileşmiş ya da değiştirilmiş görüntüler ele alınarak görüntülerden anlamlı ve özellik bilgiler alınabilmekte olup iki boyutlu sahneyi temsilen veri olarak tanımlanabilen görüntü ultrason, elektro mikroskop ve bilgisayar sayesinde girdi olarak alınan çeşitli içeriklere sahip olabilir, (Gonzalez & Woods, 2002; Aksoy B. , 2021; Jain, 1989). Farklı içeriklere sahip olan görüntüler ile çevre, mekân, kentsel planlama ve coğrafi analizler yapılmaktadır, (Irod, 2007; Aksoy B. , 2021; Nixon & Aguado, 2019). Yapılan analiz işlemleri ile ilgili bugüne kadar birçok yaklaşım ve yöntem sunulmuş olup bu yöntemler sayesinde otoyoldan geçen araç sayısını belirlemek, aracın plakasın tespit etmek, farklı şekle sahip cisimleri ayırmak, farklı renklerdeki cisimleri algılamak, bilgisayarlı tomografi görüntüsünün netleştirilmesi, resim veya videoda doğru diyafram ayarlarının belirlenmesi, uygun renk ve ışık kalitesinin ayarlanması, güvenlik taramaları için renklerin zenginleştirilmesi, fotoğraf netliğinin artırılması, görüntünün verimli bir şekilde depolanması ve aktarılması, görüntü boyutu/renk/tonlama niteliklerinin düzenlenmesi, barkod okuma, yazı karakterinin metne dönüştürülmesi, ortamdaki kişi sayısının tespiti ve medikal görüntülerin analizi gibi birçok alanda görüntü işleme

teknolojisinden yararlanılmaktadır, (Rhody, 2021; Aksoy B. , 2021; Bayram, 2021; Jähne, 2005; Ma, Tavares, & Jorge, 2009; Samtaş & Gülesin, 2011).

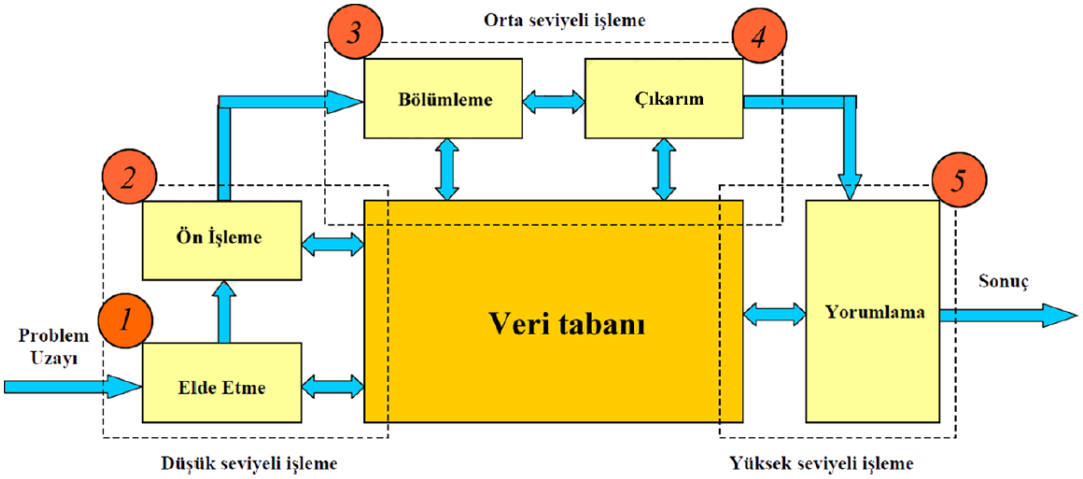
2. Görüntü İşlemenin Tarihçesi

Sayısal görüntü uygulaması ilk defa 1920'lerin başında denizaltı kablosu tarafından gönderilen gazete görsellerinin dijitalleştirilmesi ile başlamıştır, (Young, Gerbrands, & Vliet, 1998; Aksoy B. , 2021). Bir haftadan daha fazla sürede ulaştırılan görüntülerin aktarım süreci, ilerleyen dönemde kullanılan çözümleme yöntemi ile üç saate kadar düşürülmüştür, (Young, Gerbrands, & Vliet, 1998; Aksoy B. , 2021; Morse, 2000). 1960'larda ise bilgisayar teknolojisindeki gelişmeler ve uzay yarışının başlaması dijital görüntü işlemede bir çalışma dalgası yaratmış ve Ranger 7 uzay bilgisayarlardan yararlanılmıştır, (Gonzalez & Woods, 2002; Aksoy B. , 2021). Teknolojinin gelişmesiyle doğru orantılı olarak gelişmiş bilgisayarlar ve uzay programlarının da ortaya çıkmasıyla sayısal görüntü alanının potansiyeli de artmıştır, (Morse, 2000; Aksoy B. , 2021). Maryland Üniversitesi, Jet Propulsion Laboratuvarı, Bell Laboratuvarları, Massachusetts Teknoloji Enstitüsü gibi güç laboratuvarlarının da kurulmasıyla görüntü işleme teknolojisi, uzay araştırmalarında yeni bir boyut kazanmıştır, (Sugimoto, 2004; Aksoy B. , 2021). 1970'lerde dijital görüntü işleme yöntem ve teknikleri tıbbi uygulamalarda kullanılmaya başlanmış, 1979 yılında Sir Godfrey N. Hounsfield ve Prof. Allan M. Cormack, Bilgisayarlı Eksenel Tomografi (CAT) taramalarına dayanan bir teknoloji olan tomografi buluşu ile Nobel Tıp Ödülü'nü kazanmıştır, (Gonzalez & Woods, 2002; Aksoy B. , 2021). Günümüzde de uzay programlarındaki uygulamalara ek olarak, bilgisayar görüsü (Computer vision) alanında, görüntü işleme teknikleri dijital görüntülerden anlamlı sonuçlar çıkarma çalışmaları, uydu görüntüleri üzerinden nüfus yoğunluğu, hava tahmini, yerleşim yerleri ve gözlemi gibi uzaktan algılama uygulamalarında, medikal görüntüleme ve astronomi uygulamaları gibi birçok alanda kullanılmaya başlanmıştır, (Morse, 2000; Aksoy B. , 2021; Sugimoto, 2004; Bovik, 2005).

3. Görüntü İşlemenin Tanımı

Görüntü işleme, gerçek hayattan kamera, tarayıcı vb. cihazlar yardımıyla yakalanan görüntülerin bilgisayar veya başka bir cihazın okuyabileceği sayısal hale dönüştürüldükten sonra özelliklerinin ve niteliklerinin değiştirilmesi işlemidir, (Gonzalez & Woods, 2002; Aksoy B. , 2021; Irod, 2007; Russ, 2016). Görüntü işleme

teknolojisi; görüntü üzerinde bulunan herhangi bir nesnenin çıkarımının yapılabilmesi, görüntünün keskinliğini ve netliğini artırma ya da nesnelerin tanınması gibi birden fazla amaç için kullanılabilmekte olup herhangi bir resmin yazılım aracılığıyla kullanılabilmesi için sayısallaştırılması gerekmektedir, (Ding, Goshtasby, & Satter, 2001; Aksoy B. , 2021). Sayısallaştırma; resimde bulunan renkleri sayısal veriye dönüştürme işlemidir, (SHU & WU, 2010; Aksoy B. , 2021). Sayısal görüntü, her birinin değeri ve özel bir konumu olan sonlu sayıdaki resim elemanları adlandırılan alt birimlerden oluşturmaktadır. Piksel, görüntü alt birimlerinin içinde en yaygın olarak kullanılan terimdir, (Jain, 1989; Irod, 2007; Nixon & Aguado, 2019; Rhody, 2021; Aksoy B. , 2021). Görüntü işleme teknolojisi ise, kamera, video gibi araçlar ile görüntünün elektronik ortama aktarılması ile görüntünün analiz edilerek istenilen şekilde işlenmesi, görüntünün analizinden sonra oluşan veri raporu ve sonucun alınması olarak üç aşamadan oluşmakta olup görüntü işleme teknolojisinin çalışırken izlediği yol Şekil 1 'de verilmiştir, (Çayıroğlu, 2021a; Aksoy B. , 2021).



Şekil 1 Görüntü işleme genel akış şeması (Çayıroğlu, 2021a)

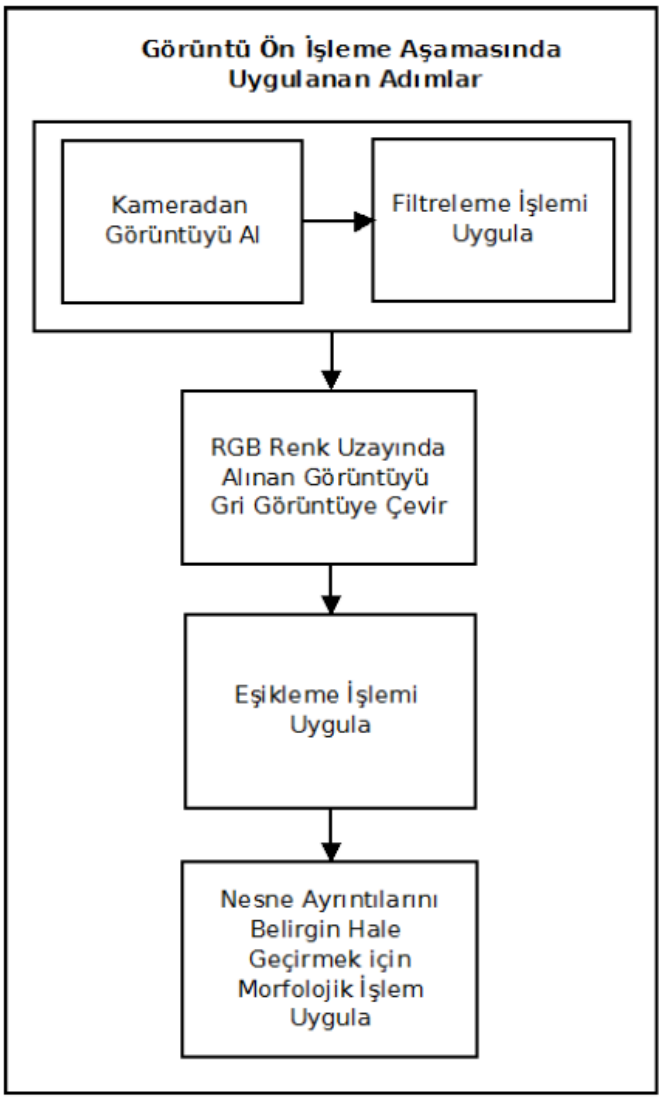
Görüntü işleme adımlarının yanı sıra görüntü işleme teknolojisi için analog ve dijital olmak üzere iki tür yöntem kullanılmaktadır, (Rhody, 2021; Aksoy B. , 2021; Bayram, 2021). Fotoğraflar ve fotokopi gibi basılı kopyalar için analog görüntü işleme teknikleri kullanılırken, görüntü üzerinde iyileştirme ya da düzeltme yapılması gerektiğinde dijital görüntü işleme tekniği tercih edilmekte olup ek olarak dijital görüntü işleme için verilerin geçmesi gereken ön işleme, geliştirme ve görüntüleme, bilgi çıkarımı gibi temel aşamalar mevcuttur, (Rhody, 2021; Aksoy B. , 2021). Bu aşamalardan sonra ilgili veriden amaca bağlı olarak istenilen sonuçlar elde

edilebilmekte olup görüntü işleme teknolojisinin kullanım amaçları beş aşamada incelenmektedir, (Aksoy B. , 2021);

- Görselleştirme: Görüntüde zor ayırt edilebilen nesneleri gözlemlemek,
- Görüntü keskinleştirme veya restorasyon: Bozuk veya gürültülü görüntüleri iyileştirmek,
- Görüntü alımı: Yüksek kaliteli görüntü arama,
- Desen Tanıma: Herhangi görüntüdeki çeşitli nesneleri tanıma,
- Görüntü Tanıma: Görüntü içerisindeki nesneleri birbirinden ayırt etme, (benimmuhendisim, 2021; Aksoy B. , 2021).

4. Görüntü Ön İşleme

Görüntü işleme aşamalarının ilki görüntü ön işleme olmakta olup kamera ya da diğer görüntüleme cihazlarıyla elde edilen görüntüleri doğrudan analiz etmek ve yorum yapmak mümkün olmamakta olup bunun nedeni kaynak olarak alınan görüntülerin gürültü, ışık yetersizliği gibi olumsuzluklar içermesi olmakta olup bu sorunları gidermek için kaynak görüntüyü iyileştirmeye yönelik filtreler uygulanarak görüntünün işlenebilmesi görüntü ön işleme aşamalarından geçirilmesi gerekmektedir, (Piskin, 2021; Aksoy B. , 2021; Çayıroğlu, 2021b). Görüntü ön işleme; görüntüleme cihazlarından elde edilen sayısal görüntüleri analiz ve yorumlama (anlamlandırma) işlemlerine geçmeden görüntü üzerinde yapılan ölçeklendirme, renk uzayındaki geçişler, eşikleme (thresholding), histogram işlemleri, filtreleme ve morfolojik dönüşüm gibi teknikler kullanılarak gerçekleştirilmektedir, (Duman, 2019; Yaman, Sarucan, Atak, & Aktürk, 2001; Rosenfeld, 1969; Karakoç, Görüntü İşleme Teknolojiler Ve Uygulamaları, 2021; Aksoy B. , 2021). Görüntü ön işleme görüntü üzerindeki nesneleri daha belirgin ve kolay işlenebilir hale getirmek için uygulanmakta olup Şekil 2 ‘de görüntü ön işlemede uygulanan temel aşamalar verilmiştir, (Solak & Altinisik, 2018).



Şekil 2 Görüntü ön işleme aşamasında uygulanan adımlar, (Solak & Altinisik, 2018)

5. Görüntü İşlemede Filtreleme

Filtreleme, orijinal bir görüntüyü mevcut durumundan gürültülerini gidererek daha iyi bir duruma getirmek için kullanılan önemli tekniklerden birisidir, (Karakoç, Görüntü İşleme Teknolojiler Ve Uygulamaları, 2021; Duman, 2019; Rao, 2007; Aksoy B. , 2021). Filtreleme işlemlerinin temel amacı görüntüdeki fiziksel özellikler arasındaki ayrımı belirginleştirerek görüntü hakkında yorum yapılabilmesini kolaylaştırmakta olup filtreleme işlemleri, görüntüyü daha açık hale getirme, koyulaştırma veya kontrastını artırma gibi işlemleri gerçekleştirmek için kullanılmaktadır, (Uğur, 2021; Aksoy B. , 2021). Temel filtreleme işlemleri genellikle mekânsal frekanslara dayanarak görüntü üzerindeki istenilen yerlerin veya nesnelerin özelliklerinden yola çıkarak aralarındaki farkların vurgulanmasını, kenar çizgilerinin belirgin hale getirilmesini ya da giderilmesini sağlamak için kullanılmakta olup

uzamsal (mekansal) filtrelemenin etki alanı, doğrudan görüntü içinde işlenmesi istenen pikseller üzerinde çalışabilmesi olup komşu pikselleri (x,y) (kernel) tanımlamak için temel yaklaşım, (x, y) merkezli kare veya dikdörtgen bir alt görüntü alanı kullanmaktır, (Keskin, Doğru, Göksel, & Balçık, 2014).

a. Ortalama (Averaging) filtreleme yöntemi

Ortalama filtre, lineer bir filtre türü olup genellikle görüntüdeki gürültüyü temizlemek için kullanılmaktadır, (Güraksın, 2018). Gürültünün temizlenmesi için görüntü üzerinde bir kernel oluşturulması gerekmekte olup bu kernel ile şablon içinde kalan piksellerin aritmetik, harmonik veya geometrik ortalaması alınıp, yeni bir piksel değeri elde edilmektedir, (Şevik, Köse, & Gençalioğlu, 2007; Aksoy B. , 2021; Librow, 2021). Elde edilen piksel değeri kernelin merkezindeki değer ile değiştirilmekte olup böylelikle görüntüdeki bütün pikseller için bu işlemler tekrarlanarak gürültü temizleme işlemi gerçekleştirilmektedir, (HIPR2, 2021c).

b. Medyan (Median) filtreleme yöntemi

Medyan filtre, doğrusal olmayan (non-lineer) bir filtre türü olup, görüntülerdeki gürültülerin azaltılmasında sıklıkla kullanılmakta olup medyan filtrele yöntemi sayesinde görüntü üzerinde kullanıcı tarafından boyutu belirlenen kernel matrisi kullanılarak filtreleme işlemi gerçekleştirilmektedir, (Uzunhisarcıklı, Göreke, & Güven, 2014; Aksoy B. , 2021). Seçilen kernel matrisi içinde kalan piksellerin yoğunluk değerleri büyükten küçüğe sıralanabileceği gibi küçükten büyüğe doğru da sıralanabilmektedir, (Kasım, 2015; Aksoy B. , 2021). Sıralamadaki ortanca piksel değeri medyan değeri olup, kernelin merkezindeki değer ile değiştirilmektedir, (Tekdemir, Arsoy, Alboyacı, & Keskin, 2013; Aksoy B. , 2021). Piksel değerlerinin büyükten küçüğe doğru sıralandığında görüntüyü temsil etmeyen uç değerler sıranın başında ya da sonunda olacağından değerler görüntüye etki etmeyecektir, (Değirmenci, Çankaya, & Demirci, 2018; Aksoy B. , 2021). Sıralama işlemi sayesinde görüntüdeki keskin kenarlar, ortalama filtreye göre daha iyi korunmaktadır, (Çayıroğlu, 2021b; Aksoy B. , 2021).

c. Gaussian filtreleme yöntemi

Gaussian filtresi görüntülerdeki gürültüleri ve ayrıntıları kaldırmak için kullanılan iki boyutlu bir bulanıklaştırma operatörüdür, (Uğur, 2021; Aksoy B. , 2021). Gauss filtresinin avantajı, filtreleme işlemini yatay ve dikey eksenlerde geçekleştirmesidir, (Aydın, 2021b; Aksoy B. , 2021).

d. Sobel filtreleme yöntemi

Sobel filtre görüntülerdeki objelerin sınırlarını tespit etmek için kullanılan bir filtreleme yöntemidir, (Aybar, 2008; Aksoy B. , 2021). Görüntülerde sınır belirlemede kullanılan en popüler filtreleme yöntemlerinden birisidir, (Çayıroğlu, 2021c; Aksoy B. , 2021). Sobel filtreleme yöntemi ile görüntü üzerinde hem yatay hem de dikey olarak sınır belirleme yapılabilmektedir, (İleri, 2021; Aksoy B. , 2021).

e. Laplacian filtreleme yöntemi

Laplacian operatörü, görüntü netleştirme, sınır algılama ve sınır hatlarını belirleme işlemlerinde sıklıkla kullanılmaktadır, (Wang, 2007; Aksoy B. , 2021). Laplacian filtre uygulama işleminde ikinci dereceden türev alınarak sınır belirleme işlemi yapılmakta olup iki boyutlu görüntülerde türev alma işlemi hem x hem de y eksenine uygulanmaktadır, (Aslan, 2018; Aksoy B. , 2021).

f. Prewitt filtreleme yöntemi

Prewitt filtreleme, görüntülerde sınır algılama işlemlerinde kullanılan yöntemlerden birisidir, (Onat, 2017; Aksoy B. , 2021). Görüntüdeki sınırları algılamak için pikseller arasındaki yoğunluk farkından yararlanılmaktadır, (Auckland, 2021; Aksoy B. , 2021). Prewitt filtreleme, sobel filtreleme gibi yatay ve dikey olmak üzere iki yönde uygulanmakta olup dikey için Gy kernel matris, yatay için Gx kernel matrisi kullanılmaktadır, (Gül, 2021; Aksoy B. , 2021). Gy ve Gx görüntü üzerine uygulandığında birinci dereceden bir türev gibi çalışarak piksek yoğunluk farkını hesaplamaktadır, (tutorialspoint, 2021; Aksoy B. , 2021; Javadzadeh, Banihashemi, & Hamidzadeh, 2015). Bu çekirdek matrisler görüntü üzerinde yalnızca yatayda veya dikeyde uygulanabildiği gibi aynı zamanda hem dikey hem de yatay olarak uygulanabilmektedir, (Aksoy B. , 2021).

g. Scharr filtreleme yöntemi

Filtre operatörlerinin optimal seçimi için yeni bir yöntem olarak sunulmuş olup Hanno Scharr tarafından doktora tezinde ortaya atılmıştır, (Scharr, 2020). Birinci dereceden türev alınarak kullanılan gradyan kenarları ve özellikleri belirlemek ve vurgulamak için kullanılan bir filtreleme yöntemi olup ayrıca kenarları veya piksel yoğunluğundaki değişiklikleri algılamak için de kullanılmaktadır, (PlantCV, 2021).

6. Morfoloji İşlemleri

Morfoloji işlemleri, görüntüdeki nesnelerin şeklini baz alarak kümeler teorisi ile oluşturulmuş önemli görüntü işleme yöntemlerinden birisidir, (Zhang, Murai, & Baltsavias, 1999; Aksoy B. , 2021). Bir dijital görüntüye morfoloji işlemi uygulanarak görüntü içinden istenilen nesnenin çıkarılması, diğer nesnelerden ayırt edilmesi, görüntüdeki gürültünün azaltılması ve bölütleme gibi işlemlerin gerçekleştirilmesinde kullanılmaktadır, (Karhan, Oktay, Karhan, & Demir, 2011; Aksoy B. , 2021). Morfolojik süzgeçleme görüntü işlemede, inceltme, erozyon gibi amaçlarla ön veya son işlem olarak sıklıkla kullanılmaktadır, (Çelik, 2011; Aksoy B. , 2021). Morfoloji işlemi, ikili görüntüler (siyah-beyaz) üzerinde uygulanabileceği gibi gri seviyeli görüntüler üzerinde de uygulanmaktadır, (Çayıroğlu, 2021d; Aksoy B. , 2021). Görüntü işlemede, genişletme (dilation) ve aşındırma (erosion) işlemleri temel morfoloji işlemleri olup morfolojik görüntü işlemede kullanılan tüm yöntemler bu iki temel işlem çerçevesinde gerçekleştirilmektedir, (Atalı, Özkan, & Karayel, 2016; Aksoy B. , 2021).

7. OpenCV nedir?

OpenCV, bilgisayarlı görü uygulamaları ve ticari ürünlerde makine öğrenme algısını hızlandırmak için hazırlanmış bir kütüphanedir, (Aksoy B. , 2021; OpenCV, 2021). OpenCV terimi (Open Source Computer Vision Library (Açık Kaynak Bilgisayar Görüntü Kütüphanesi)) kavramını ifade etmekte olup aynı zamanda makine öğrenmesini de desteklemektedir, (Aksoy A. , 2019). OpenCV, açık kaynak kodlu bir "Bilgisayalarlı Görü – Computer Vision" kütüphanesi olmakta olup görüntü işleme ile ilgili yüzlerce ileri ve temel seviyedeki fonksiyonu, optimize edilmiş halleriyle barındırmakta olup ilk olarak Intel'in Rusya'daki laboratuvarlarında, 1999 yılında geliştirilmeye başlanmıştır, (Kuyumcu, 2018). OpenCV C++, Java, Matlab/Octave, C# ve Python gibi dilleri desteklemektedir, (Kuyumcu, 2018; Aksoy B. , 2021). Bu tez de Python dili kullanılmıştır. Ayrıca OpenCV mobil uygulama geliştirme üzerine de kullanılabilmekte olup Android uygulama geliştirmede derleyici olarak Android Studio kullanılmakta ve iOS uygulama geliştirmede derleyici olarak xCode kullanılmaktadır, (Ünal, 2021). OpenCV'nin Android Studio için kullanılabilen kütüphanesi bulunmaktadır, (Polat, 2021). OpenCV'nin xCode için kullanılabilen kütüphanesi bulunmaktadır, (Ni, 2021).

8. Python nedir?

Python programlama dili 1980'lerin sonuna doğru geliştirilmeye başlanmış olup ilk implemantasyonu Guido van Rassum tarafından 1989 Aralık ayında geliştirilmeye başlanmış yüksek seviyeli, yorumlanabilir, interaktif ve nesneye yönelik bir programlama dilidir, (Aydemir, 2020; Aksoy B. , 2021). Van Rossum Python'un ana geliştiricisidir, (Aksoy A. , 2021). Python ismini ünlü komedyen Monthy Python'dan esinlenerek bulmuş olup Python açık kaynak kodlu bir programlama dili olup ücretsizdir, (Arslan, 2021). Günümüzde "Python Yazılım Vakfı" gönüllüleri tarafından geliştirilen dil, basitliği, nesne yönelimli olması, hemen hemen her ortamda çalışması (Linux, MacOs, Windows, …) ve ücretsiz dağıtımı gibi sebeplerle yazılımcılar tarafından tercih edilmektedir, (Çobanoğlu, 2021). Python program geliştirme sürecini bir hayli kısaltmakta olup ayrı bir derleyici programa ihtiyaç duymamaktadır, (Özgül, 2011). Python veri yapıları, hazır fonksiyonlar ve çeşitli algoritmaları sizin tekrar yazmanıza gerek kalmadan hazır olarak içerisinde barındırmakta olup bu sayede daha akıcı programlama yapılmasına imkân tanımaktadır, (Yıldız, 2020). Ayrıca Python PyBoard, Raspberry Pi gibi elektronik alandaki projeler içinde kullanılabilmektedir, (Taşcı, 2019).

B. Yapay Zekâ

Yapay Zekâ, bilgisayarın insana özgü özellikleri olduğu düşünülen anlam çıkarma, genelleme yapabilme, akıl yürütme ve geçmiş deneyimlerden öğrenme gibi özelliklerin bilgisayar tarafından yapılması olarak tanımlanmaktadır, (Nabiyev, 2012; Yılmaz, 2019). İnsan beyni geçmiş tecrübeleri sayesinde çok hızlı bir şekilde değerlendirme yapabilmekte olup örneğin bir şoför, yolun kayganlığını, önündeki tehlikeden uzaklığını, sayısal olarak değerlendiremese bile geçmişte edinmiş olduğu tecrübelerden faydalanarak aracın hızını azaltmaktadır, (Elmas, 2018).

1. Yapay Sinir Ağları

Yapay Sinir Ağları (YSA), yapay zekânın en bilinen yöntemlerinden biri olarak insan beyninin modellenerek taklit edilmesi olmakta olup modellemede insan beyninde bulunan biyolojik sinir hücrelerini (nöron) baz almaktadır, (Yılmaz, 2019). Yapay sinir ağları bir algoritmaya ihtiyaç duymadan programcının klasik programlama yeteneklerini gerektirmeden, kendi kendine öğrenme düzenekleri olup bu ağlar

öğrenmenin yanında ezberleme ve bilgiler arasında ilişkiler oluşturma yeteneklerine de sahiptir, (Elmas, 2018).

2. Yapay Sinir Ağlarında Öğrenme

Yapay sinir ağlarındaki öğrenme işleminde veriler işlenir ve işlenen veriler ile bir çıkış değeri üretilmekte olup bu çıkış değeri olması gereken değere ağırlıkları değiştirerek ulaşılmaya çalışılmakta olup ağırlıklar güncellenerek olması istenilen duruma ulaşılıncaya kadar ilerleyen zamana öğrenme adı verilir, (Yılmaz, 2019).

a. Danışmanlı öğrenme

Danışmanlı öğrenme yönteminde giriş değerleri ile birlikte çıkış değerlerinin de sisteme girilmesi gerekmekte olup ağda bulunan çıktı değeri ile beklenen çıktı değeri arasındaki fark ile hata hesaplanarak ağırlıkların yeniden hesaplanarak güncellenmesinde kullanılır, (Yılmaz, 2019).

b. Danışmansız öğrenme

Danışmansız öğrenme yönteminde sadece giriş değerlerinin sisteme girilmesi yeterli olmakta olup giriş değerleri kendi aralarında sınıflanarak bir sınıflama kuralı oluşturulup ağırlıklar da bu kurala göre sınıflandırma yapabilecek şekilde düzenlenmesi tamamlandığında öğrenme işlemi bitmiş olmaktadır, (Yılmaz, 2019).

c. Takviyeli öğrenme

Takviyeli öğrenme yönteminde sadece giriş değerlerinin sisteme girilmesi yeterli olmakta ancak elde edilen sonuç değerinin iyi ya da kötü olduğu konusunda ağa bilgi verilmesi gerekmekte olup bu bilgiye göre ağ kendini öğrenerek tekrar düzenlemektedir, (Yılmaz, 2019).

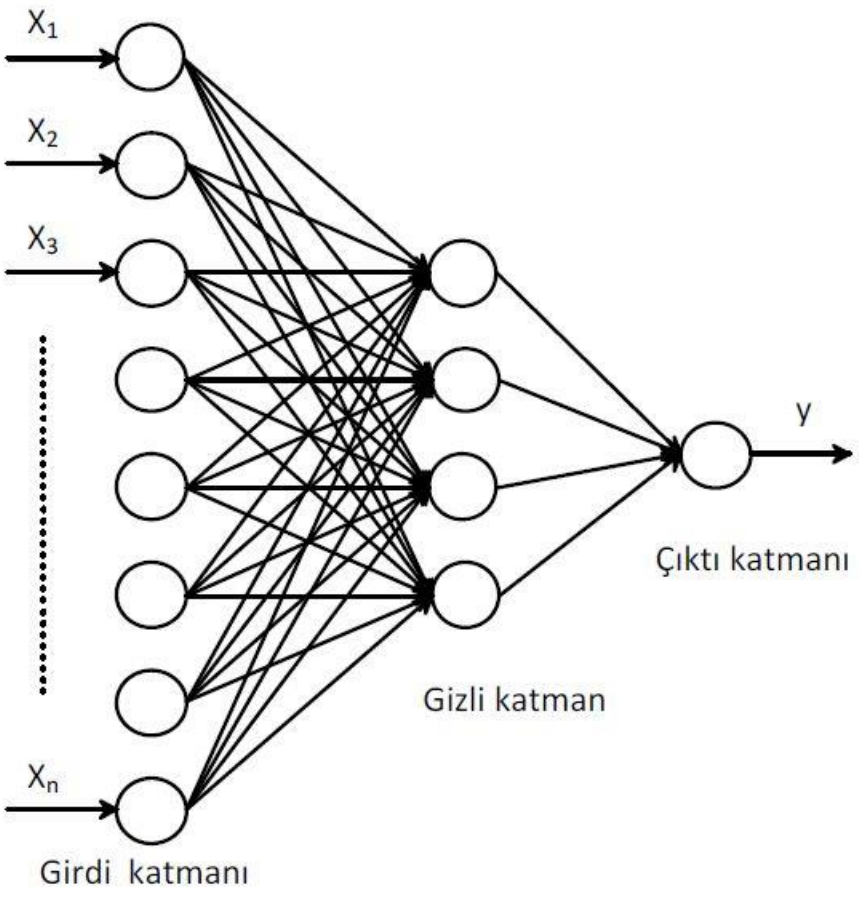
3. Yapay Sinir Ağları Modelleri ve Yapıları

Yapay sinir ağlarında, hücre elemanlarının bağlantıları değişik biçimde birbirine bağlanabilmekte olup, bağlantılarının yönlerine göre ya da ağdaki işaretlerin akış yönüne göre ağ mimarileri incelendiğinde geri beslemeli ve ileri beslemeli ağlardan bahsedilmektedir, (Yılmaz, 2019).

a. Sinir ağı

Yapısal olarak yalnız girdi ve çıktı katmanına sahip olan doğrusal olmayan ağlar, karmaşık sorunları çözebilme yeteneğinde olmamakta olup bu türde doğrusal olmayan sorunlara çözüm üretebilecek ağlar için ara katman eklenmesi gerekmektedir, (Yılmaz,

2019). Şekil 3'de 1 gizli katmana (ara katmana) sahip üç katmanlı yapay sinir ağı gösterilmiş olup katmanda bulunan bütün sinirler, sonraki katmanda bulunan her sinir ile bağlantılı olup katman içerisindeki sinirler arasında ise bağlantı olmamaktadır, (Yılmaz, 2019).



Şekil 3 Yapay Sinir Ağı Yapısı (FannTool, 2021)

b. Katmanlar

Şekil 3'de görüldüğü gibi ağ mimarisinde sinirsel katmanlar bulunmakta olup giriş katmanı başka bir ağdan ya da dışarıdan giriş alan sinirleri içermekte olup bu katman ile ağa örnekler sunulmaktadır, (Yılmaz, 2019). Giriş katmanına gelen bilgiler sonraki katmanlara giriş olacak şekilde iletilmekte olup çıkış katmanında ise ağın çıkışını başka bir ağa ya da dışarıya ileten hücreleri içermektedir, (Yılmaz, 2019). Eğer çok katmanlı bir ağ yapısı ise çıkış ve giriş katmanları arasında gizli katmanlar mevcut olup ara katman sayısı ağın en iyi çalışma durumuna getirilebilecek sayıda seçilmelidir, (Yılmaz, 2019).

Katmanlar arasında farklı bağlantılar mevcut olup bu bağlantı çeşitleri, (Yılmaz, 2019);

Tam bağlantılı: Katmandaki bütün hücreler sonraki katmandaki bütün sinirlere bağlanmış ise tam bağlantılıdır denir, (Yılmaz, 2019).

Kısmi bağlantılı: Eğer katmandaki hücreler kendilerinden sonra gelen katmandaki hücrelerin tümüyle değil de bir kısmı ile bağlantılıysa kısmi bağlantı vardır demektir, (Yılmaz, 2019).

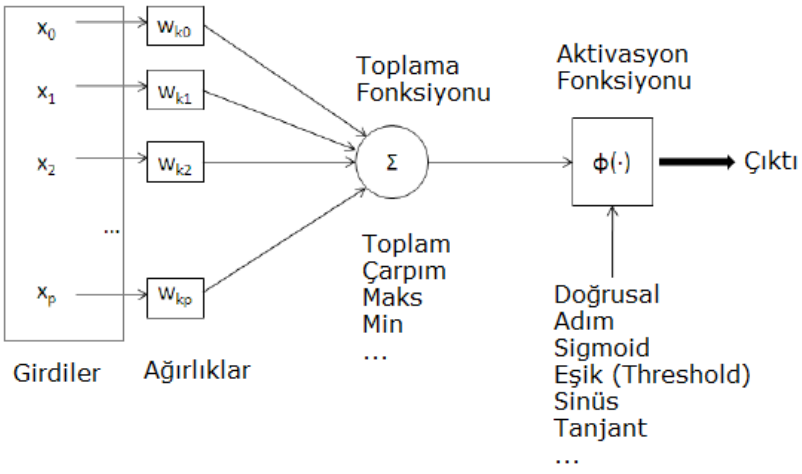
İleri besleme: Eğer katmandaki hücreler kendisinden sonra gelen katmandaki hücrelere çıkışlarını gönderip geriye giriş alamazlar ise ileri beslemeli ağ yapısından bahsedilmektedir, (Yılmaz, 2019).

Çift yönlü: Eğer katmanlardaki hücrelerin çıkışlarının bir önceki katmanlardaki hücrelere bağlayan bir yapı var ise çift yönlü bağlantı olmaktadır, (Yılmaz, 2019).

Çift yönlü ya da ileri beslemeli ağlarda kısmi bağlantı ya da tam bağlantı olabilir, (Yılmaz, 2019).

c. Tek katmanlı sinir ağları (Perceptron)

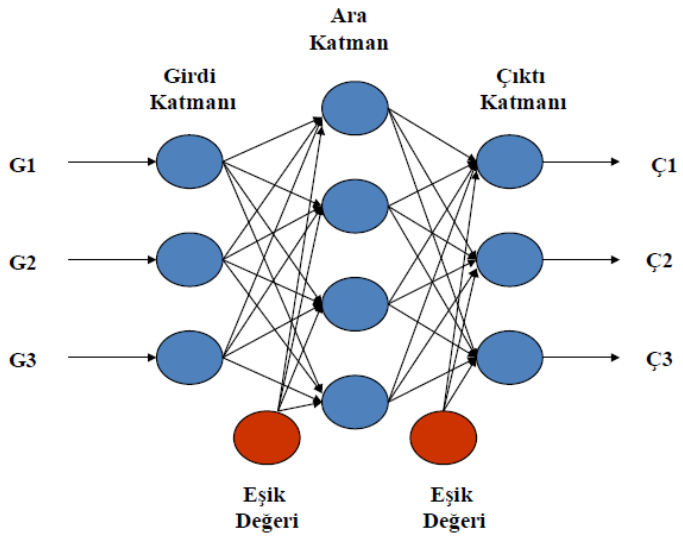
Şekil 4'deki gibi sadece çıktı ve girdi katmanı olan ağlara tek katmanlı sinir ağları denilmektedir, (Yılmaz, 2019). Zor sorunlara cevap verebilme kabiliyetine sahip olmamakta olup perceptron, Rosentblatt tarafından 1960'lı yıllarda tek katmanlı sinir ağları olarak önerilmekte olup Perceptron'lar, sorunların çözümü konusunda son derece sınırlı olmakta olup ama geri yayılma gibi kullanılan çeşitli algoritmalar ile yakınlığı bulunan önemli bir yapıdır, (Yılmaz, 2019).



Şekil 4 Tek Katmanlı Algılayıcı Mimarisi

d. Çok katmanlı algılayıcılar

Tek katmanlı algılayıcılarla doğrusal olmayan ilişkiler öğrenilmemekte olup karşılaşılan sorunların çoğu doğrusal olmamakta olup doğrusal olmayan bir sorun olan XOR sorununu çözmek amacı ile yapılan çalışmalar sonucunda çok katmanlı ağlar geliştirilmiştir, (Yılmaz, 2019). Rumelhart ve arkadaşları tarafından geliştirilen çok katmanlı algılayıcı modeline hatayı geriye yayma (backpropogation) modeli de denilmekte olup çok katmanlı algılayıcılar (ÇKA), yapay sinir ağlarının ilerleyişindeki durgunluğu sonlandırarak ilgiyi yeniden kendi üzerine toplamış olup çoğu öğrenme modeli çok katmanlı ağlar ile eğitiminde kullanılabilmektedir, (Yılmaz, 2019). Şekil 5'deki gibi çok katmanlı algılayıcılar çıkış katmanı ve girdi katmanı ile birlikte katmanlar arasında bağlantıları oluşturan ara katmanları yapısında bulundurmaktadır, (Yılmaz, 2019).



Şekil 5 Çok Katmanlı Algılayıcı Mimarisi (Umut, 2021)

i. Ara katman

Girdi katmanından iletilen bilgileri işledikten sonra sonraki katmana göndermekte olup gönderilen katman başka bir ara katman ya da çıkış katmanı olabilmekte olup bütün işlemlerin elemanları sonraki katmanlarda bulunan bütün işlemlerin elemanları ile bağlantılıdır, (Yılmaz, 2019).

ii. Çıkış katmanı

Çıkış katmanı, kendinden önce yer alan ara katmanda iletilen bilgileri işlemekte olup ağa ulaşan girdilere karşılaşıla gelen ağın hesapladığı çıkışı üreterek dışarıya sistemin sonucunu iletmekte olup birden fazla işlem elemanı bulunabilmekte ama her işlem elemanın sadece tek bir çıktısı bulunmaktadır, (Yılmaz, 2019).

e. İleri beslemeli ağlar

İleri beslemeli ağlar katmanlara ayrılmakta olup girdi katmanından çıkış katmanına tek yönlü bir bağlantı ile iletilir, (Yılmaz, 2019). Katmandaki nöronlardan sonraki katmanda yer alan bütün nöronlara bağlantı varken; aynı katman içindeki nöronlar birbirleri ile bağlantılı olmamakta olup çok katmanlı perceptron (MLP) ve LVQ (Learning Vector Quantization (Öğrenmeli Vektör Kuantulama)) ileri beslemeli ağlarına örnektir, (Yılmaz, 2019).

f. Geri beslemeli ağlar

Danışmansız öğrenme ile cevaplanan sorunlar için uygun ağlar olup ara ve çıkış katmanındaki çıkışların, önceki katmanlara geri beslenen ağ yapısı olup ağ üzerinden girişten sonraki katmanlarına doğru ileri yönde hesaplama yapılır iken geri bildirimiyle geri yönde hesaplar da yapılmaktadır, (Yılmaz, 2019).

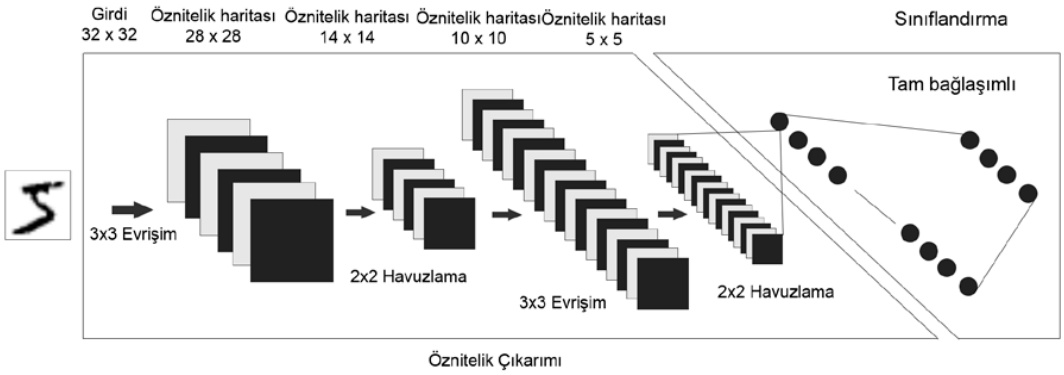
4. Derin Öğrenme

Son yıllarda sosyal medyaların ve bilişim teknolojilerinin yaygın kullanımına bağlı olarak veri miktarları büyük boyutlara ulaşmış ve çeşitlenmiş olup "Büyük Veri" (Big Data) diye adlandırılmış olup verileri işlemek ve veriyi bilgi haline çevirmek için klasik yöntemler ve araçlar yeterli olamamaya başlamıştır, (Elmas, 2018). Yapay Sinir Ağları (YSA) için büyük problem olmaya devam eden yeterli sayıda verinin bulunamıyor olması artık problem olmaktan çıkmış olup genişliğe ve derinliğe sahip veri kümeleri kolay bulunabilir hale gelmiştir, (Elmas, 2018). Gelişen teknoloji ile birlikte artan verilerin yanı sıra paralel çalışan Grafiksel İşleme Birimi (GİB, GPU) ve Merkezi İşleme Birimi (MİB, CPU) gibi donanımların gelişmesiyle birlikte daha fazla sayıda sinir ve katmana sahip YSA'nın eğitilmesi ve çalıştırılması mümkün hale gelmiştir, (Elmas, 2018). Daha fazla derinliğe ve genişliğe sahip bu ağlar "Derin Sinir Ağları" (DSA), eğitimleri de "Derin Öğrenme" (Deep Learning) olarak adlandırılmaya başlanmış olup çok kısa bir zamanda çok önemli gelişmeler yaşanmıştır, (Elmas, 2018).

5. Evrişimli Sinir Ağları (Convolutional Neural Network (CNN))

Lecun ve arkadaşları tarafından 1998 yılında öne sürülen Evrişimli Sinir Ağları görüntü gibi 2 boyutlu veriler için daha uygun olup her gizli evrişim filtresi, girdisini nöron aktivasyonlarının 3 boyutlu çıktısına dönüştürmektedir, (Yılmaz & KAYA, Derin Öğrenme, 2019). Evrişimli Sinir Ağları oluşturulurken 1962 yılında Hubel ve

Wiesel tarafından sunulan görsel korteksin nörobiyolojik yapısından esinlenilerek geliştirilmiş olup Şekil 6'da görüleceği üzere klasik bir sinir ağlarına göre daha az sayıda nöron bağlantısına gereksinim duymaktadır, (Yılmaz & KAYA, Derin Öğrenme, 2019). Evrişimli sinir ağları için çeşitli modeller söz konusu olup Krizhevsky ve arkadaşları tarafından görüntünün sınıflandırılması için oluşturulan AlexNet, Szegedy ve arkadaşları tarafından ortaya sürülen GoogleNet örnek olarak gösterilebilir, (Yılmaz & KAYA, Derin Öğrenme, 2019). Evrişimli Sinir Ağları görsel niteliklerin arasındaki bütün hiyerarşik modeli bulmak için birden fazla katmana gerek duyabilmekte olup genelde etiketli görüntülerin büyük veri kümesine gerek duymaktadır, (Yılmaz & KAYA, Derin Öğrenme, 2019).



Şekil 6 Evrişimli Sinir Ağları (Kurt, 2018)

6. Makine Öğrenmesi

Daha önceki bilgilerinden farklı olan bir bilgi ile karşılaşıldığında insan bu bilgi ile deneyim ve tecrübe elde etmektedir, (Yılmaz, 2019). Bu deneyimleme ve tecrübe edinme yeteneğinin bilgisayar üzerinde yapılmasına "Bilgisayarlı Öğrenme" ya da "Makine Öğrenmesi" denilmektedir. Örneğin, bir çocuk yaz mevsiminde kalorifere dokunur ve eli yanmaz, ancak hava şartları değişip kış mevsimi geldiğinde kalorifer yanmaya başlayıp çocuk daha önceki deneyimi ile yaz mevsiminde dokunup eli yanmadığı için kalorifere dokunduğunda, eli yanacak olup böylece daha önceki edinimlerinden farklı olarak kalorifer çalışırken dokunulmaması gerektiğini tecrübe etmiştir, (Yılmaz, 2019). Çocuk bu tecrübeden sonra kalorifer çalışırken dokunmayacak olup bilginin ve tecrübenin bu şekilde elde edilme yöntemine geri beslenmeli öğrenme yöntemi denilmektedir, (Yılmaz, 2019)

7. Bütünsel İç İçe Kenar Algılama (Holistically-Nested Edge Detection (HED))

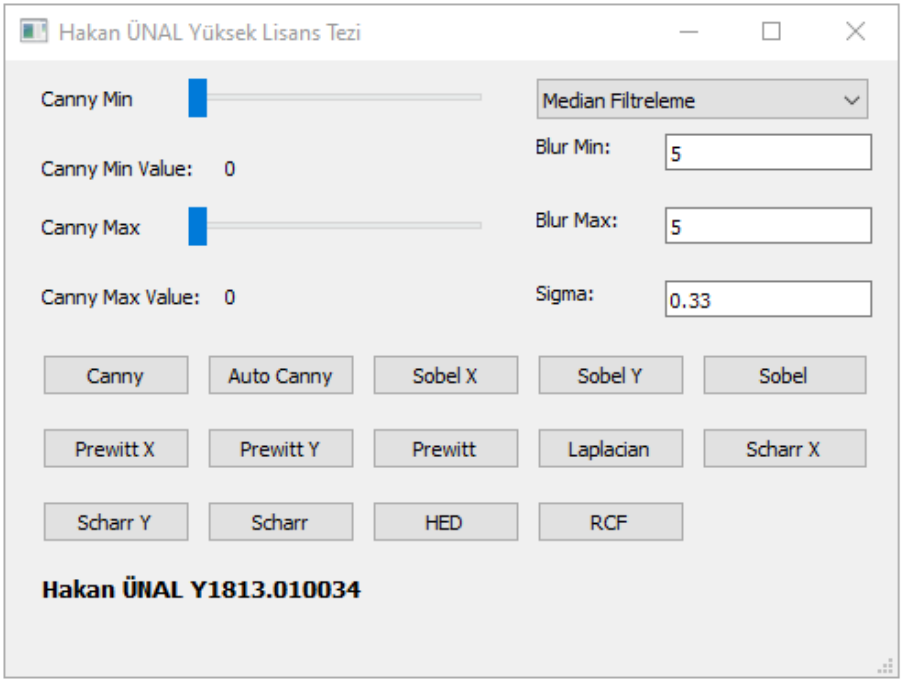
Bütünsel teriminin kullanılmasının nedeni yapılandırılmış çıktıyı açıkça modellememesine rağmen, kenarları görüntüden görüntüye bir şekilde eğitmeyi ve tahmin etmeyi amaçlamakta olup, iç içe terimi ile yan çıktılar olarak üretilen kalıtsal ve aşamalı olarak rafine edilmiş kenar haritalarını vurgulayıp her tahminin yapıldığı yolun, ardışık kenar haritalarının daha kısa olmasıyla bu kenar haritalarının her biri için ortak olduğunu göstermeyi amaçlamaktadır, (Xie & Tu, 2015). Bütünsel İç İçe Kenar Algılama tamamen evrişimli sinir ağlarından ve derinlemesine denetlenen ağlardan yararlanan bir derin öğrenme modeli aracılığıyla görüntüden görüntüye tahmin gerçekleştirir, (Xie & Tu, 2015).

8. Kenar Algılama için Daha Zengin Evrişimsel Özellikler (Richer Convolutional Features for Edge Detection)

Kenar Algılama için Daha Zengin Evrişimsel Özellikler (Richer Convolutional Features for Edge Detection (RCF)), tüm evrişimsel özellikleri, zengin özellik hiyerarşilerinden iyi bir şekilde yararlanan ve geri yayılım yoluyla eğitime uygun olan daha ayırt edici temsilde kapsüllemektedir, (Liu, ve diğerleri, 2019). RCF, görüntüden görüntüye tahminini bütünsel olarak gerçekleştirmek için nesnelerin çok ölçekli ve çok düzeyli bilgilerinden tam olarak yararlanmaktadır, (Liu, ve diğerleri, 2019). Ayrıca RCF kodlaması geliştirilirken HED kodlaması baz alınarak geliştirmeler yapılmıştır, (Liu, 2021).

IV. UYGULAMALAR

Uygulamalar geliştirilirken Python programlama dili kullanılmış ve tasarım kütüphanesi olarak PyQt5 kullanılmış ve Şekil 7'de görülmektedir.



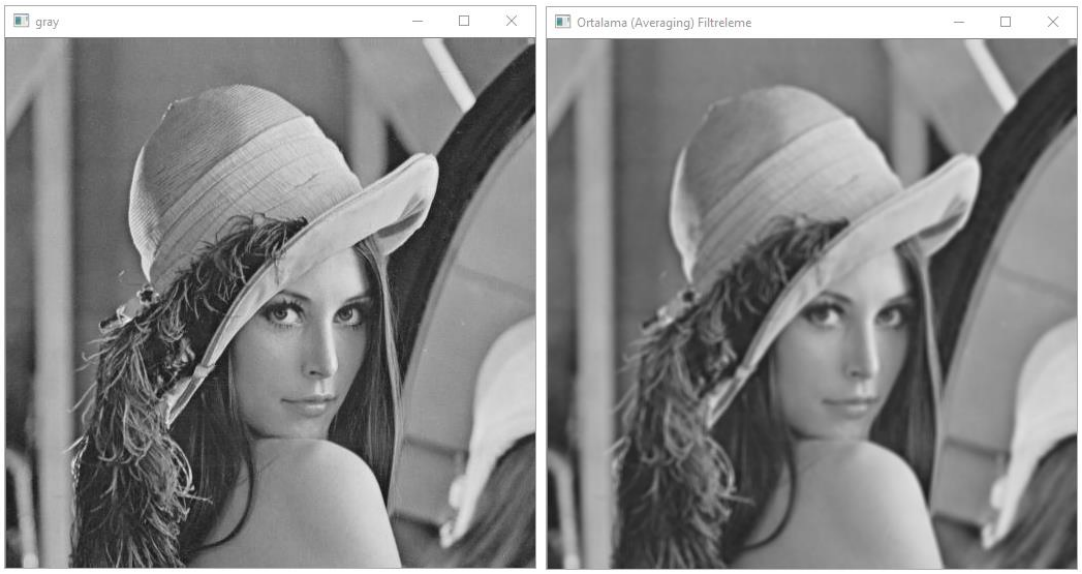
Şekil 7 Python ve PyQT 5 ile yapılan uygulamanın ekran görüntüsü

A. Görüntü İşleme Uygulamaları

Derleyici olarak PyCharm 2021.1 kullanılmıştır. Python versiyonu olarak 3.8.5 kullanılmıştır. OpenCV versiyonu olarak 4.0.1 kullanılmıştır.

1. Ortalama Filtreleme

Şekil 8 Ortalama Filtreleme yönteminin Cameraman görselindeki etkisi



Şekil 9 Ortalama Filtreleme yönteminin Lenna görselindeki etkisi

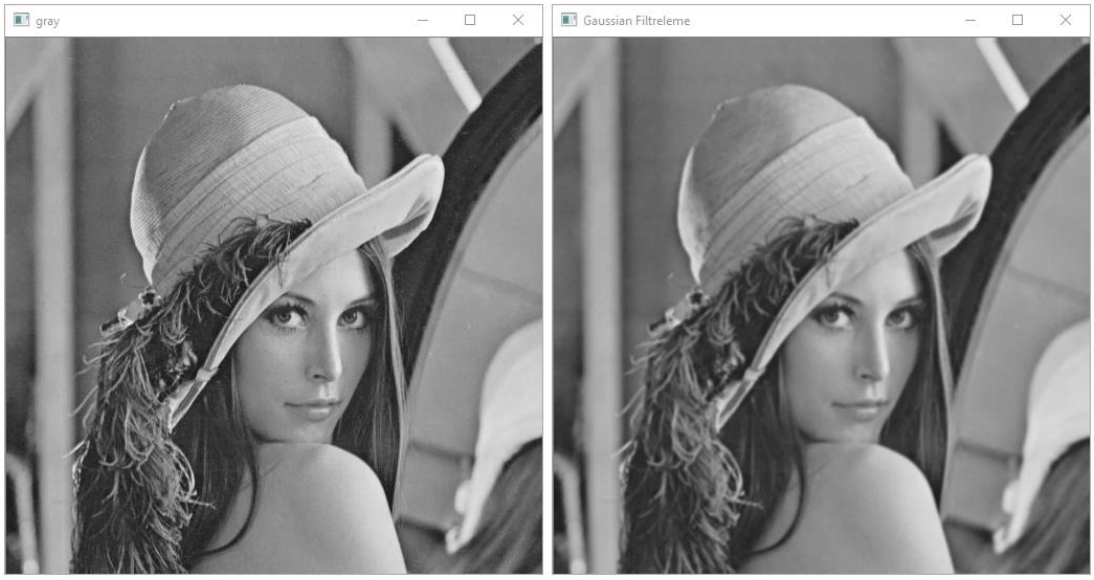
Çizelge 1 Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon

blur = cv2.blur(gray, (int(self.ui.lineEdit.text()), int(self.ui.lineEdit_2.text())), 0)
```

Parametre olarak siyah beyaz görüntüyü, çekirdek boyutunu (kernel size) örneğin 5,5 bu değeri artırmak görüntüdeki bulanıklaştırmayı artırmakta olup son parametre olarak 0 yani girdi görüntüsü ile çıktı görüntüsünün boyutlandırması aynı olmaktadır. Ortalama Filtreleme yönteminin uygulanışı Çizelge 1'de verilmekte ve Şekil 8 ve Şekil 9'da bu yöntemin etkisi görülmekte ve sol tarafta görüntünün siyah beyaz hali sağ tarafta ise filtrelenmiş hali görülmektedir.

2. Gaussian Filtreleme

Şekil 10 Gaussian Filtreleme yönteminin Cameraman görselindeki etkisi



Şekil 11 Gaussian Filtreleme yönteminin Lenna görselindeki etkisi

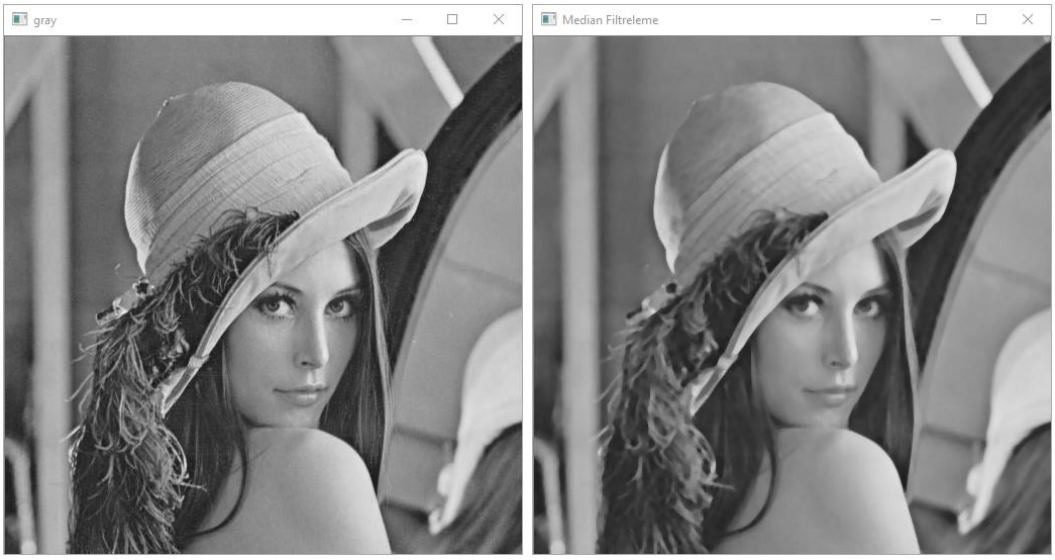
Çizelge 2 Gaussian Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon

blur = cv2.GaussianBlur(gray, (int(self.ui.lineEdit.text()), int(self.ui.lineEdit_2.text())), 0)
```

Parametre olarak siyah beyaz görüntüyü, çekirdek boyutunu (kernel size) örneğin 5,5 bu değeri artırmak görüntüdeki bulanıklaştırmayı artırmakta olup son parametre olarak 0 yani girdi görüntüsü ile çıktı görüntüsünün boyutlandırması aynı olmaktadır. Ortalama Filtreleme yönteminin uygulanışı Çizelge 2'de verilmekte ve Şekil 10 ve Şekil 11'de bu yöntemin etkisi görülmekte ve sol tarafta görüntünün siyah beyaz hali sağ tarafta ise filtrelenmiş hali görülmektedir.

3. Medyan Filtreleme

Şekil 12 Medyan Filtreleme yönteminin Cameraman görselindeki etkisi



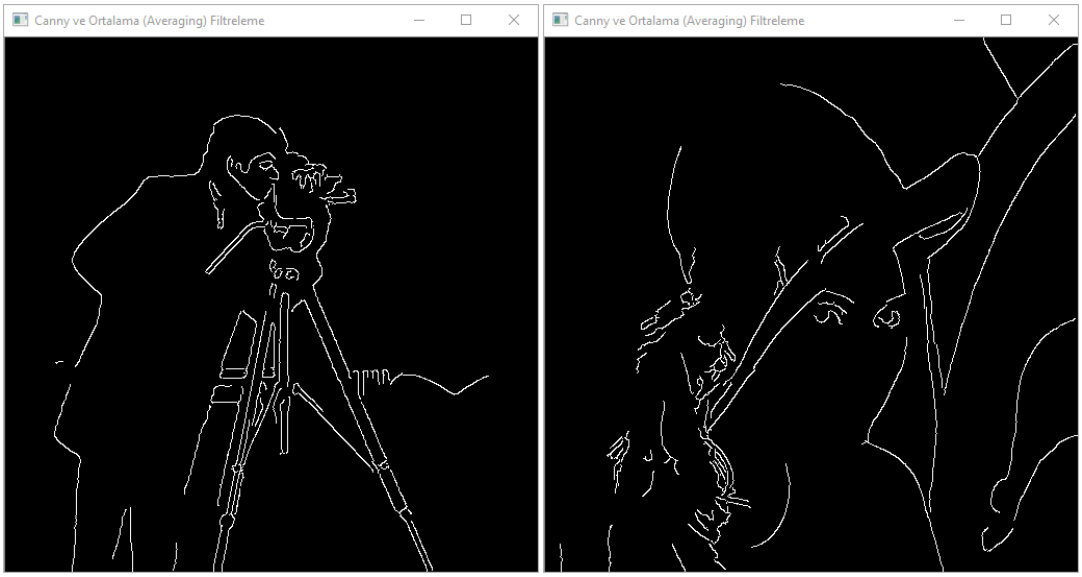
Şekil 13 Medyan Filtreleme yönteminin Lenna görselindeki etkisi

Çizelge 3 Medyan Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon

blur = cv2.medianBlur(gray, (int(self.ui.lineEdit.text())), 0)
```

Parametre olarak siyah beyaz görüntüyü, çekirdek boyutunu (kernel size) örneğin 5 bu değeri artırmak görüntüdeki bulanıklaştırmayı artırmakta olup son parametre olarak 0 yani girdi görüntüsü ile çıktı görüntüsünün boyutlandırması aynı olmaktadır. Ortalama Filtreleme yönteminin uygulanışı Çizelge 3'de verilmekte ve Şekil 12 ve Şekil 13'de bu yöntemin etkisi görülmekte ve sol tarafta görüntünün siyah beyaz hali sağ tarafta ise filtrelenmiş hali görülmektedir.

4. Canny ve Ortalama Filtreleme



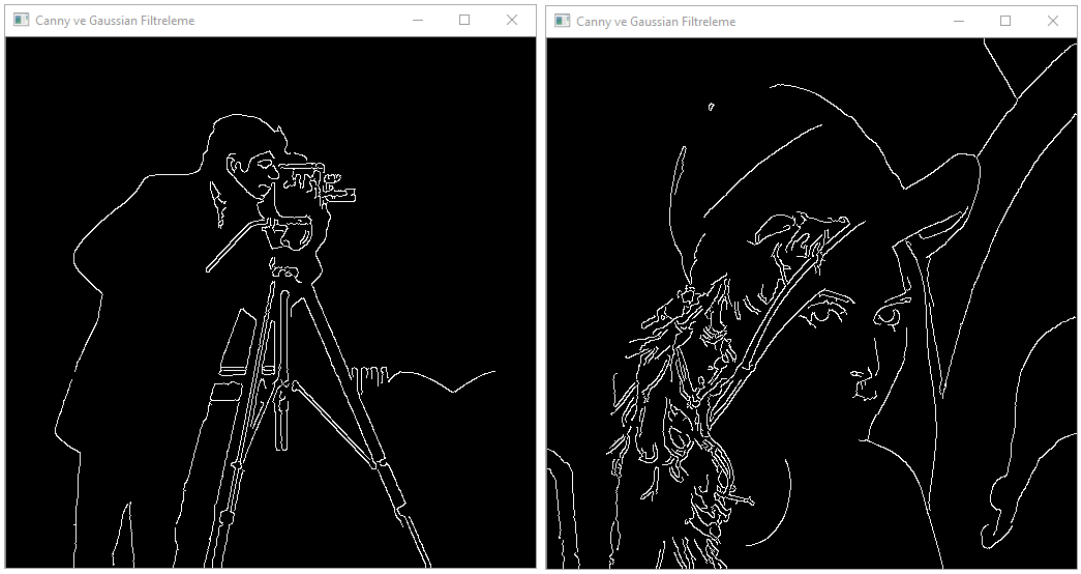
Şekil 14 Canny ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 4 Canny ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon

        blur = cv2.blur(gray, (int(self.ui.lineEdit.text()), int(self.ui.lineEdit_2.text())), 0)

        canny = cv2.Canny(blur, self.ui.horizontalSlider.value(), self.ui.horizontalSlider_2.value())

        grayRS = cv2.resize(gray, (512, 512))
blurRS = cv2.resize(blur, (512, 512))
cannyRS = cv2.resize(canny, (512, 512))
cv2.imshow("gray", grayRS)
cv2.imshow(self.ui.comboBox.currentText(), blurRS)
cv2.imshow("Canny ve " + self.ui.comboBox.currentText(), cannyRS)

        cv2.waitKey(0)
cv2.destroyAllWindows()
```

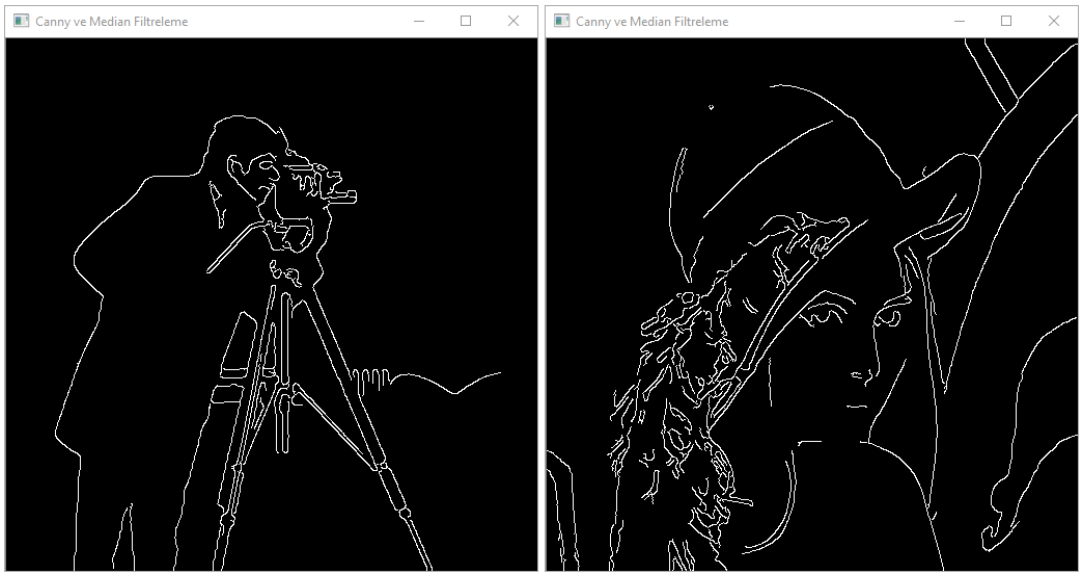
Canny fonksiyonunun aldığı parametreler bulanıklaştırılmış görüntü, birinci eşik değeri ve ikinci eşik değeri parametrelerini almaktadır. Canny ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 4'de verilmekte ve Şekil 14'de bu yöntemin etkisi görülmektedir. Canny ve Gaussian Filtreleme yönteminin ve Canny ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 15 ve Şekil 16'da görülmektedir.

5. Canny ve Gaussian Filtreleme



Şekil 15 Canny ve Gaussian Filtreleme Yönteminin Cameraman ve Lenna görselindeki etkisi

6. Canny ve Medyan Filtreleme



Şekil 16 Canny ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

7. AutoCanny ve Ortalama Filtreleme

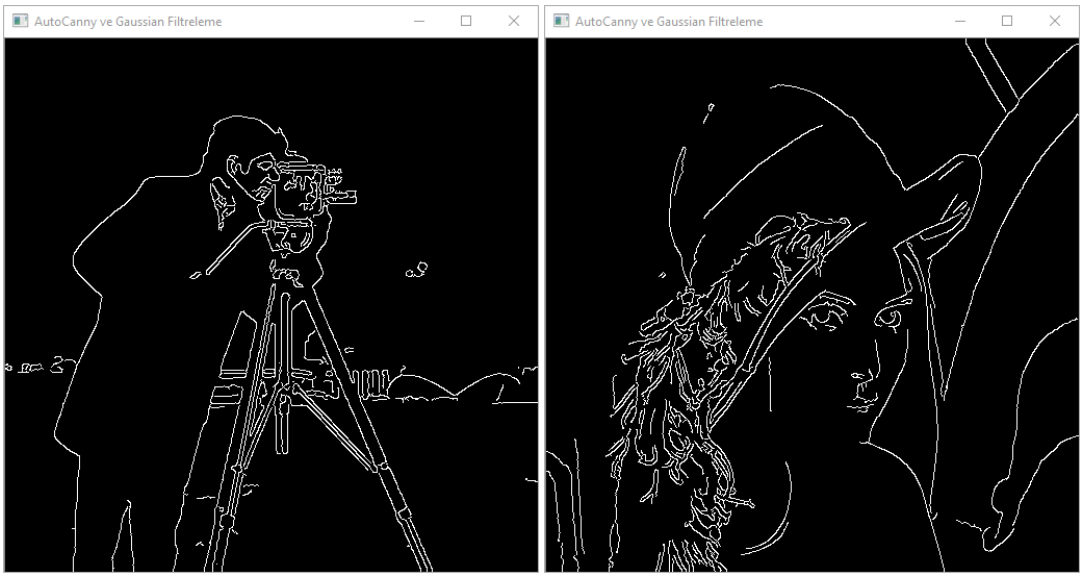
Şekil 17 AutoCanny ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 5 AutoCanny ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon

        blur = cv2.blur(gray, (int(self.ui.lineEdit.text()), int(self.ui.lineEdit_2.text())), 0)

        sigma = float(self.ui.lineEdit_3.text())
medyan = np.median(blur)
lower = int(max(0,(1.0-sigma)*medyan))
upper = int(min(255,(1.0+sigma)*medyan))
autoCanny = cv2.Canny(blur, lower, upper)
grayRS = cv2.resize(gray, (512, 512))
blurRS = cv2.resize(blur, (512, 512))
autoCannyRS = cv2.resize(autoCanny, (512, 512))
cv2.imshow("gray", grayRS)
cv2.imshow(self.ui.comboBox.currentText(), blurRS)
cv2.imshow("AutoCanny ve " + self.ui.comboBox.currentText(), autoCannyRS)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

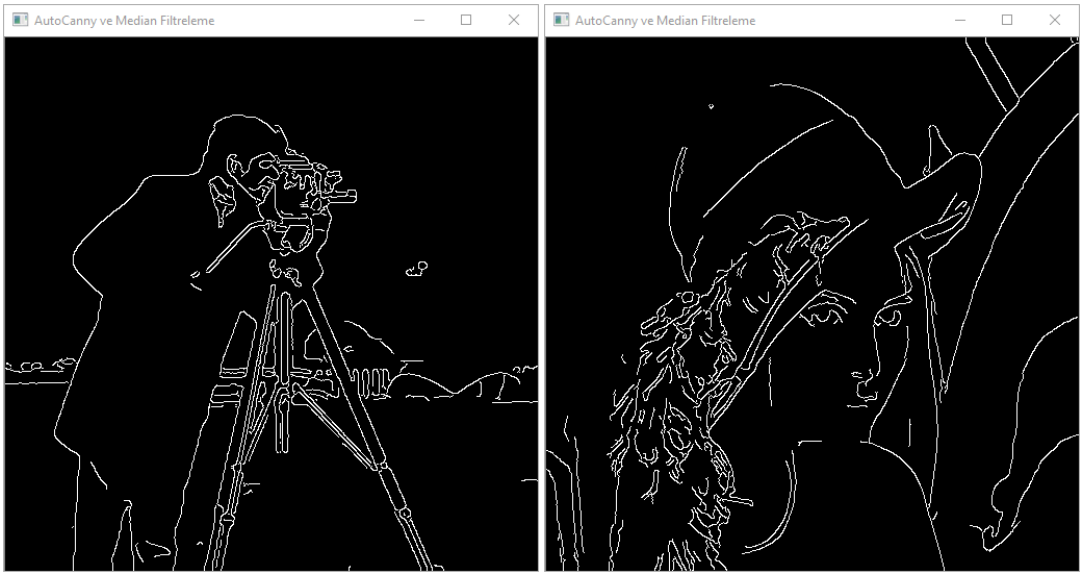
AutoCanny fonksiyonunun Canny fonksiyonundan farklı olarak eşik değerlerinin sigma, median, lower ve upper değerleri ile hesaplanması olup böylelikle Canny fonksiyonu otomatikleştirilmektedir. AutoCanny ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 5'de verilmekte ve Şekil 17'de bu yöntemin etkisi görülmektedir. AutoCanny ve Gaussian Filtreleme yönteminin ve AutoCanny ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 18 ve Şekil 19'da görülmektedir.

8. AutoCanny ve Gaussian Filtreleme



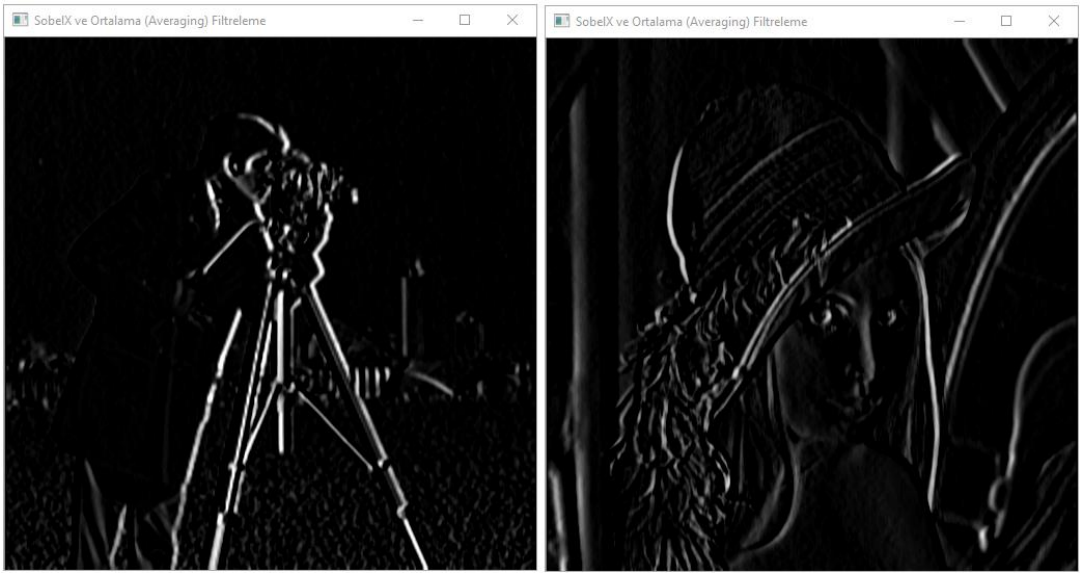
Şekil 18 AutoCanny ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

9. AutoCanny ve Medyan Filtreleme



Şekil 19 AutoCanny ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

10. SobelX ve Ortalama Filtreleme



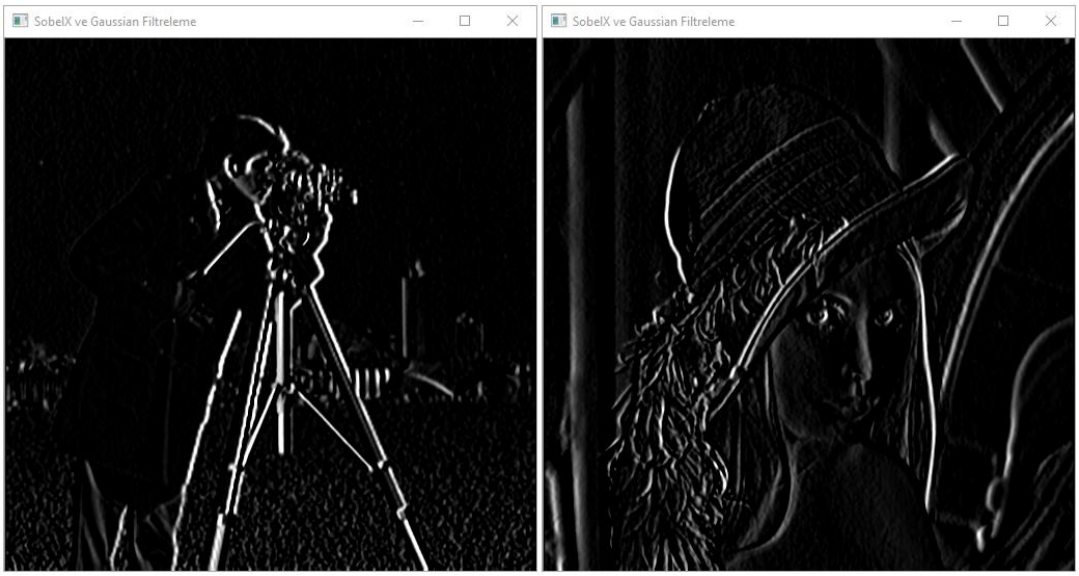
Şekil 20 SobelX ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 6 SobelX ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon

        bulanik = cv2.blur(gray, (int(self.ui.lineEdit.text()), int(self.ui.lineEdit_2.text())), 0)

        sobelXEkseni = cv2.Sobel(bulanik, cv2.CV_8U, 1, 0, ksize=3) sobelXRS = cv2.resize(sobelXEkseni, (512, 512))

        cv2.imshow("SobelX ve " + self.ui.comboBox.currentText(), sobelXRS)

        cv2.waitKey(0)
cv2.destroyAllWindows()
```

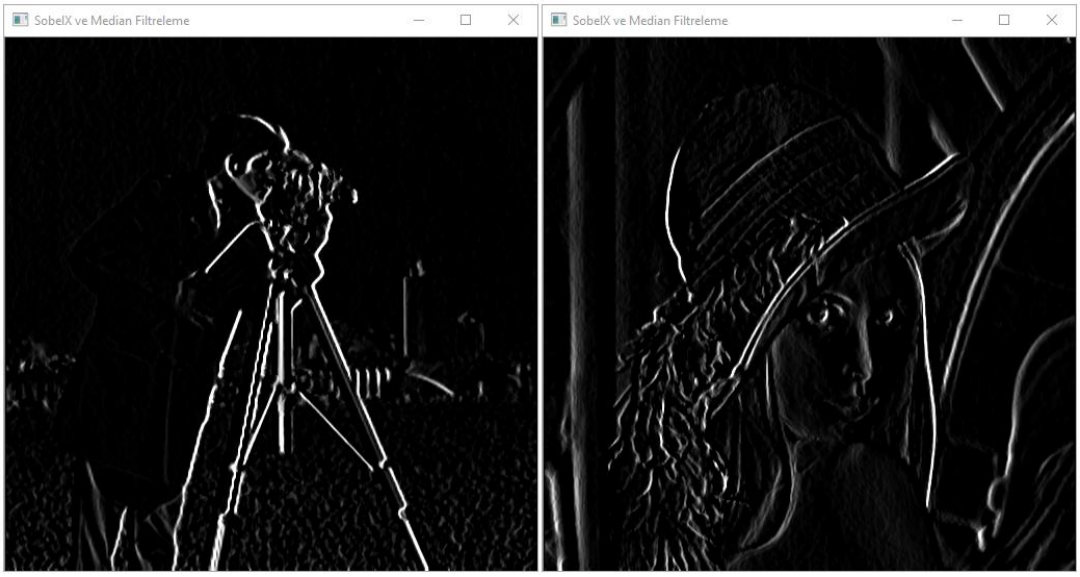
SobelX fonksiyonunun aldığı parametreler bulanıklaştırılmış görüntü, görüntü derinliği, x ekseni ya da yatay olup olmadığı yani 1 ya da 0, y ekseni ya da dikey olup olmadığı yani 1 ya da 0 ve çekirdek boyutu parametrelerini almaktadır. SobelX ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 6'de verilmekte ve Şekil 20'de bu yöntemin etkisi görülmektedir. SobelX ve Gaussian Filtreleme yönteminin ve SobelX ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 21 ve Şekil 22'de görülmektedir.

11. SobelX ve Gaussian Filtreleme



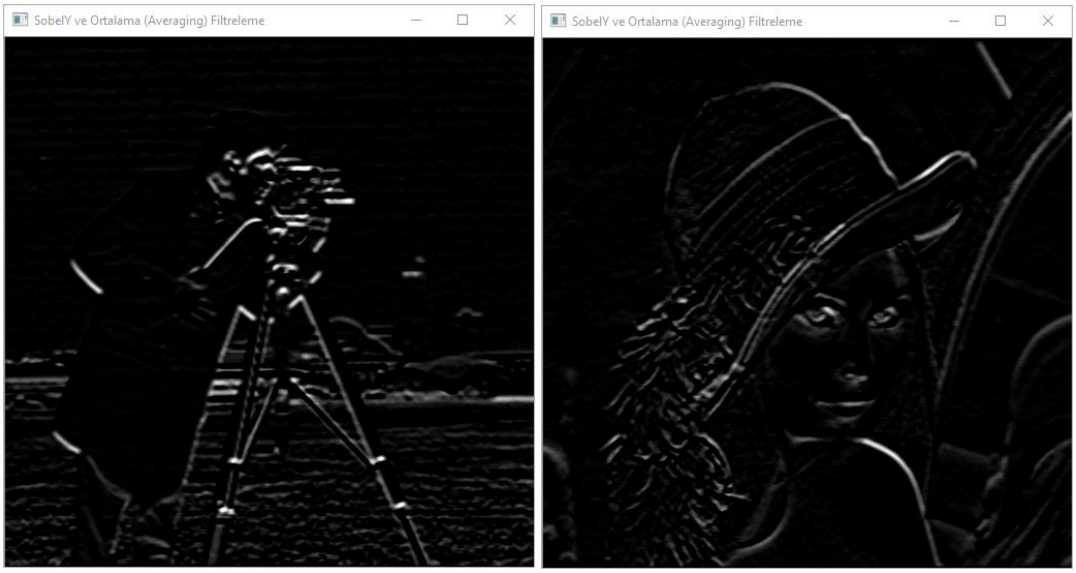
Şekil 21 SobelX ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

12. SobelX ve Medyan Filtreleme



Şekil 22 SobelX ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

13. SobelY ve Ortalama Filtreleme



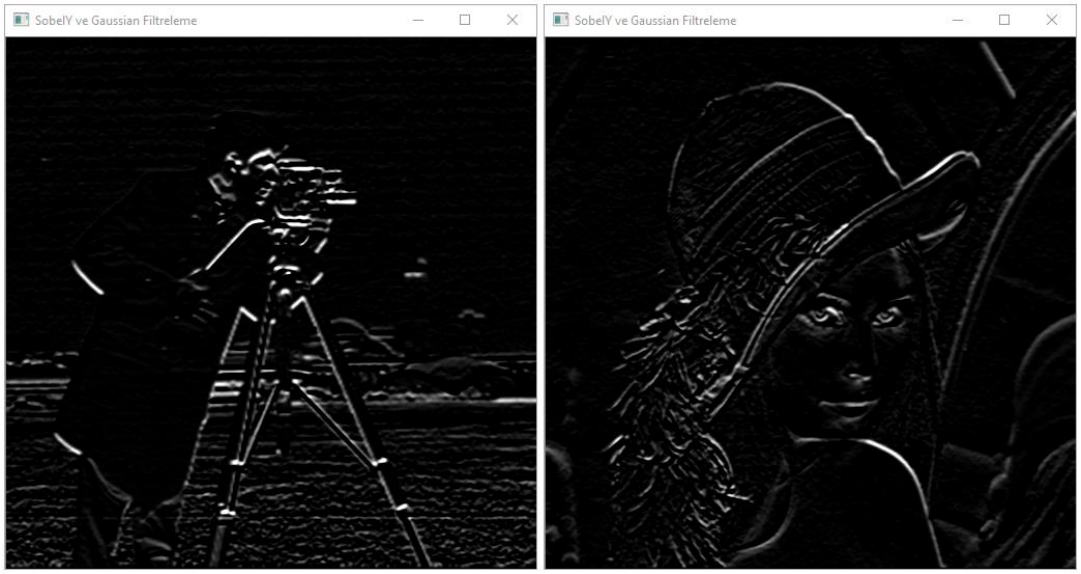
Şekil 23 SobelY ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 7 SobelY ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah
beyaz yapmak için kullanılan fonksiyon

        bulanik = cv2.blur(gray, (int(self.ui.lineEdit.text()),
int(self.ui.lineEdit_2.text())), 0)

        sobelYEkseni = cv2.Sobel(bulanik, cv2.CV_8U, 0, 1, ksize=3)
sobelYRS = cv2.resize(sobelXEkseni, (512, 512))

        cv2.imshow("SobelY ve " + self.ui.comboBox.currentText(), sobelYRS)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

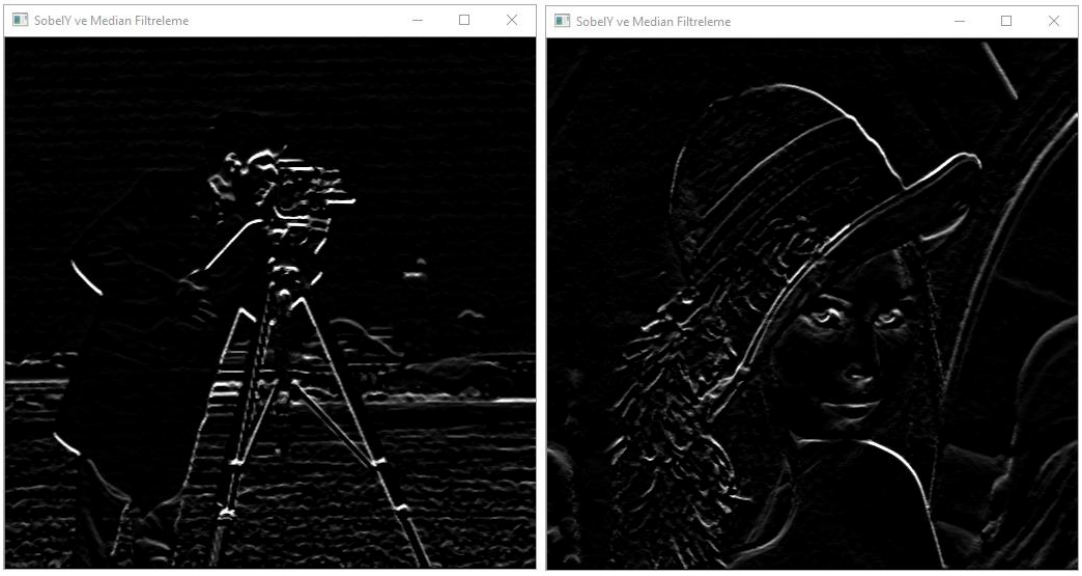
SobelY ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 7'de verilmiş ve SobelX yönteminden farkı kernel matrisidir ve Şekil 23'de bu yöntemin etkisi görülmektedir. SobelY ve Gaussian Filtreleme yönteminin ve SobelY ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 24 ve Şekil 25'de görülmektedir.

14. SobelY ve Gaussian Filtreleme



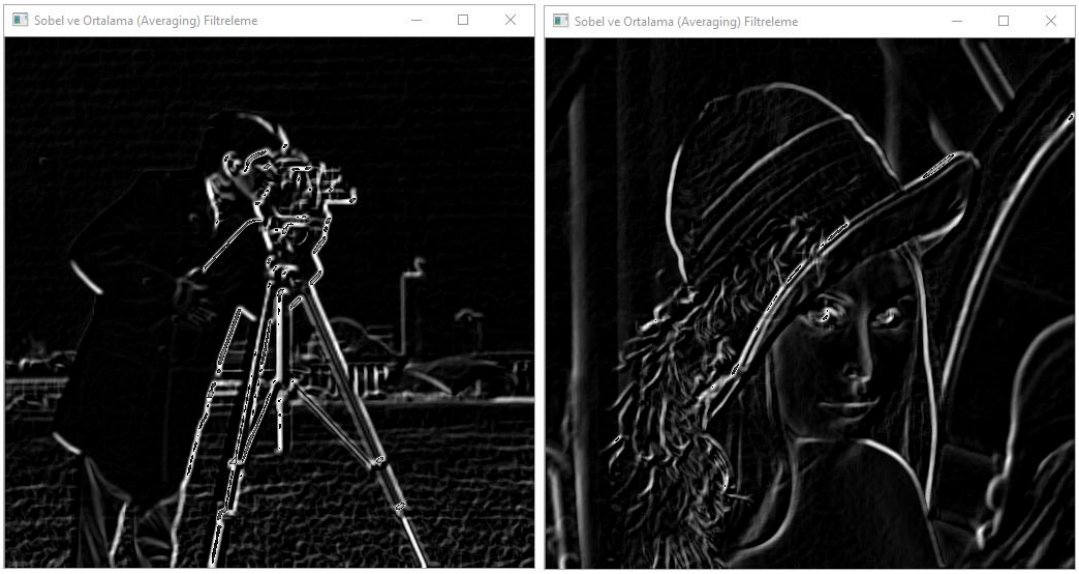
Şekil 24 SobelY ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

15. SobelY ve Medyan Filtreleme



Şekil 25 SobelY ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

16. Sobel ve Ortalama Filtreleme



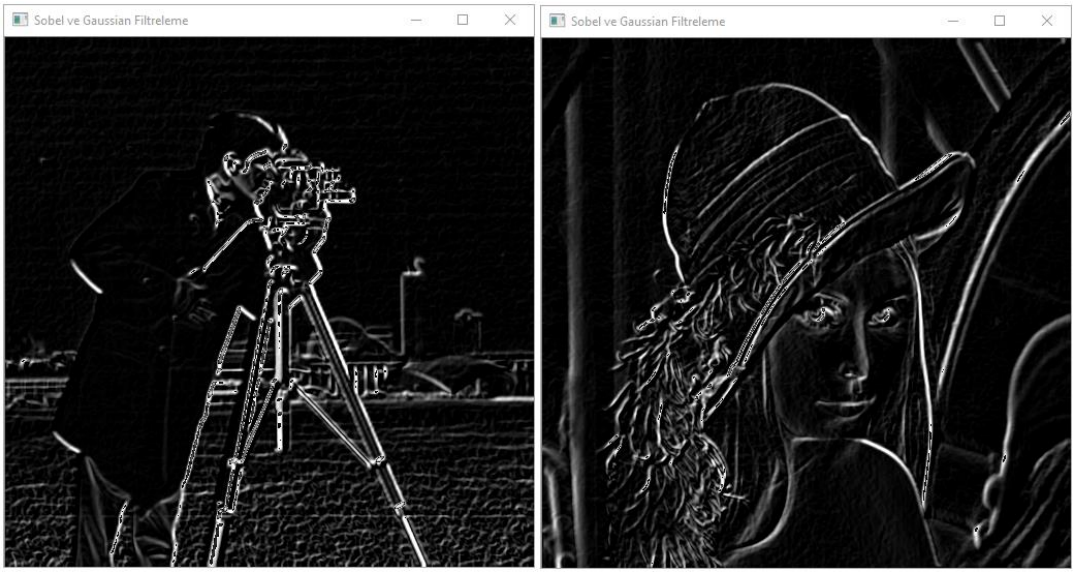
Şekil 26 Sobel ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 8 Sobel ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
cv2.imshow("Sobel ve " + self.ui.comboBox.currentText(), sobelXRS + sobelYRS)
```

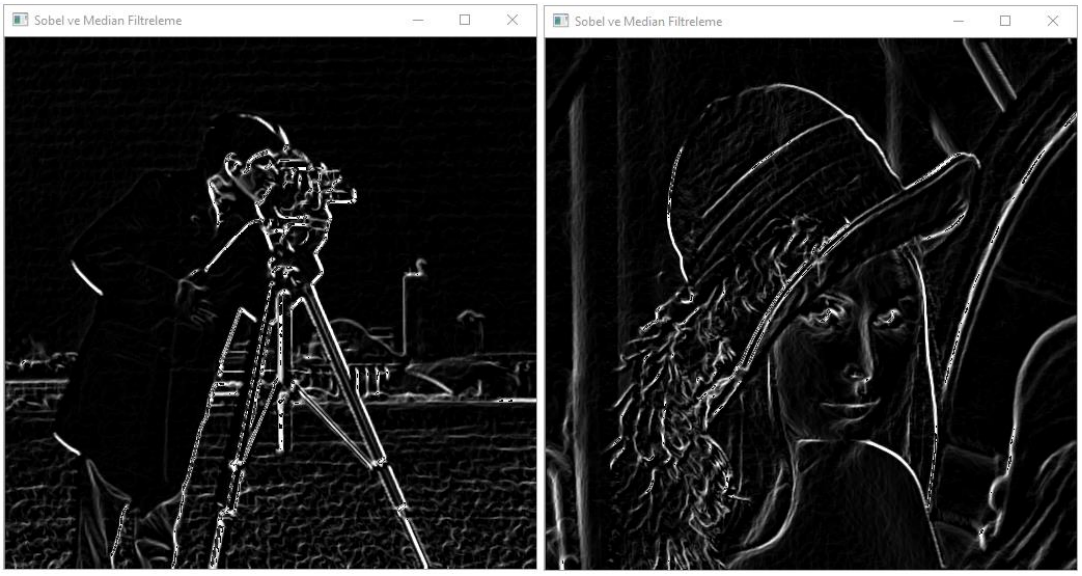
Sobel yöntemi SobelX ve SobelY yöntemlerinin birleşmiş halidir. Sobel ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 8'de verilmekte ve Şekil 26'da bu yöntemin etkisi görülmektedir. Sobel ve Gaussian Filtreleme yönteminin ve Sobel ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 27 ve Şekil 28'de görülmektedir.

17. Sobel ve Gaussian Filtreleme



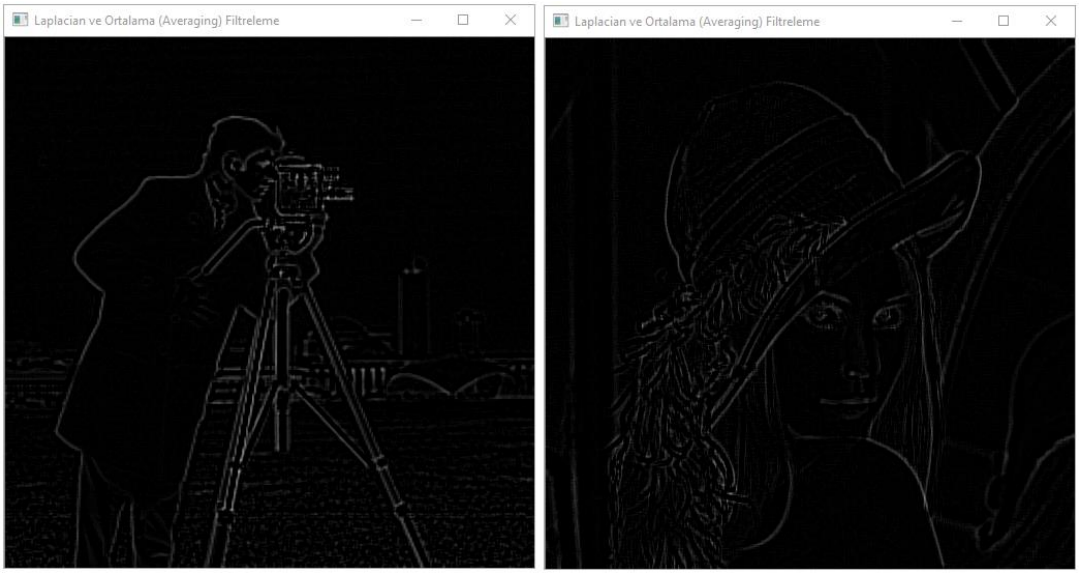
Şekil 27 Sobel ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

18. Sobel ve Medyan Filtreleme



Şekil 28 Sobel ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

19. Laplacian ve Ortalama Filtreleme



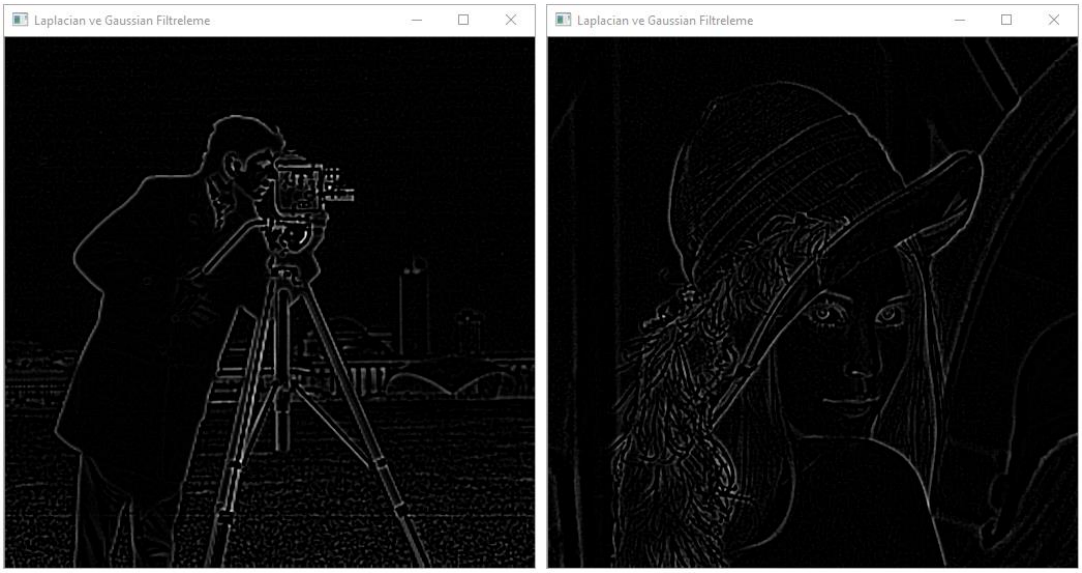
Şekil 29 Laplacian ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 9 Laplacian ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon

        blur = cv2.blur(gray, (int(self.ui.lineEdit.text()), int(self.ui.lineEdit_2.text())), 0)

        laplacian = cv2.Laplacian(blur, cv2.CV_8U, ksize=3)
laplacianRS = cv2.resize(laplacian, (512, 512))
cv2.imshow("Laplacian ve " + self.ui.comboBox.currentText(), laplacianRS)

        cv2.waitKey(0)
cv2.destroyAllWindows()
```

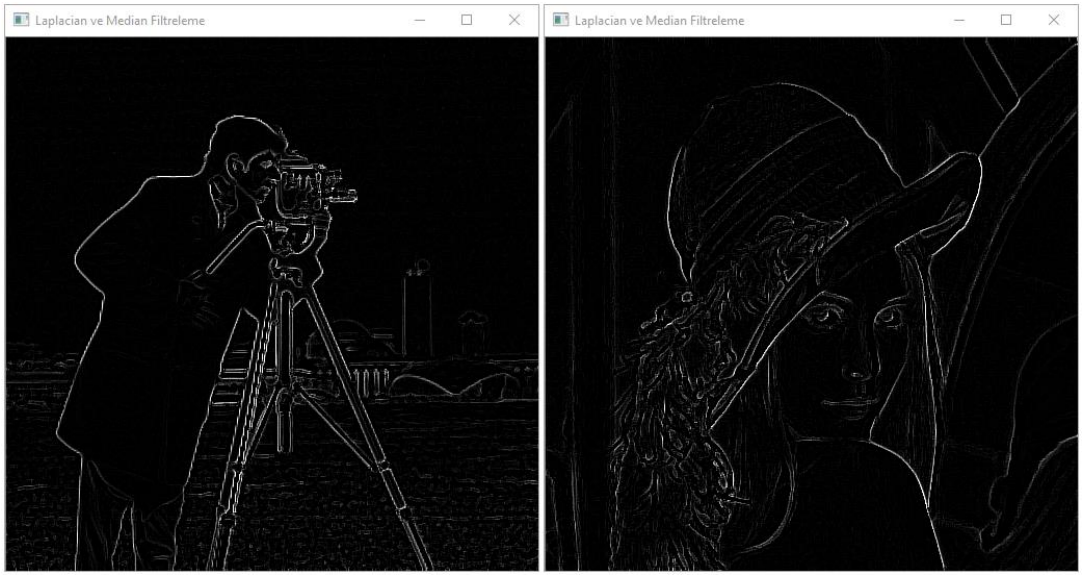
Laplacian fonksiyonunun aldığı parametreler, bulanıklaştırılmış görüntü, görüntü derinliği ve çekirdek boyutu parametrelerini almaktadır. Laplacian ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 9'da verilmekte ve Şekil 29'da bu yöntemin etkisi görülmektedir. Laplacian ve Gaussian Filtreleme yönteminin ve Laplacian ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 30 ve Şekil 31'de görülmektedir.

20. Laplacian ve Gaussian Filtreleme



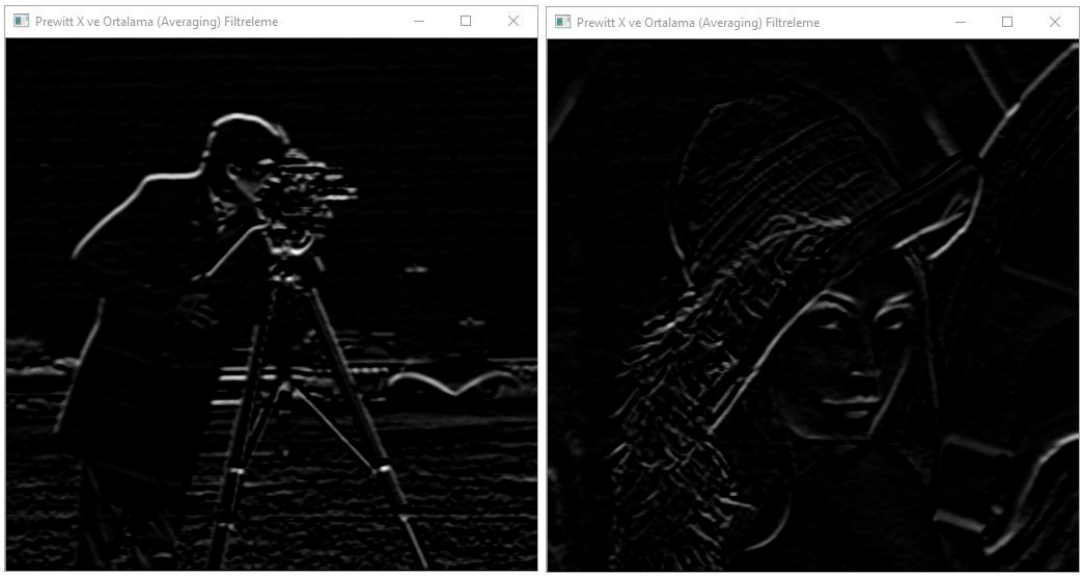
Şekil 30 Laplacian ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

21. Laplacian ve Medyan Filtreleme



Şekil 31 Laplacian ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

22. Prewitt X ve Ortalama Filtreleme



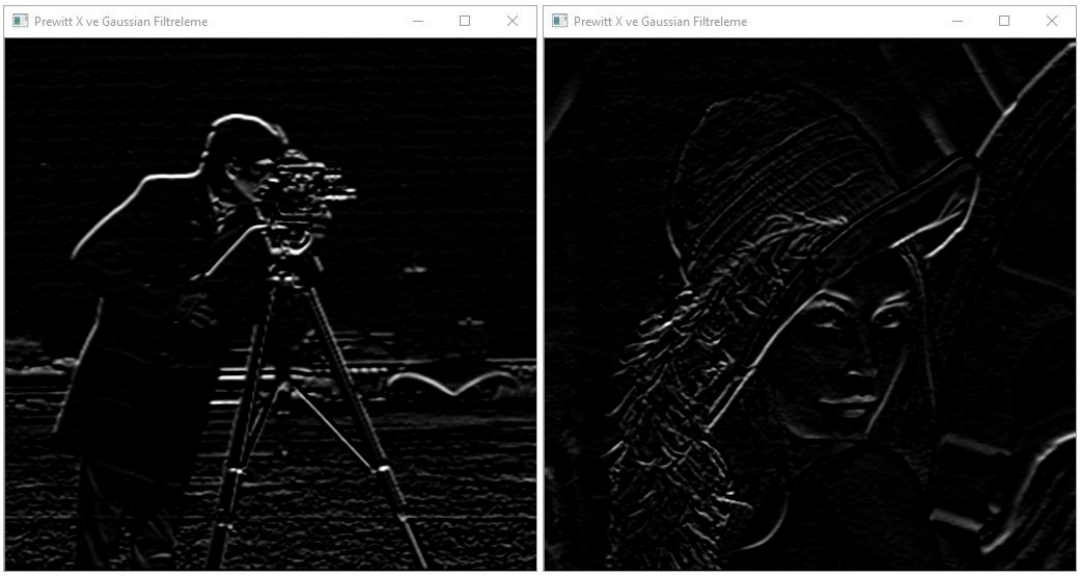
Şekil 32 Prewitt X ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 10 Prewitt X ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon

        blur = cv2.blur(gray, (int(self.ui.lineEdit.text()), int(self.ui.lineEdit_2.text())), 0)

        kernelXEkseni = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]])
prewittX = cv2.filter2D(blur, -1, kernelXEkseni)
prewittXRS = cv2.resize(prewittX, (512, 512))
cv2.imshow("Prewitt X ve " + self.ui.comboBox.currentText(), prewittXRS)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

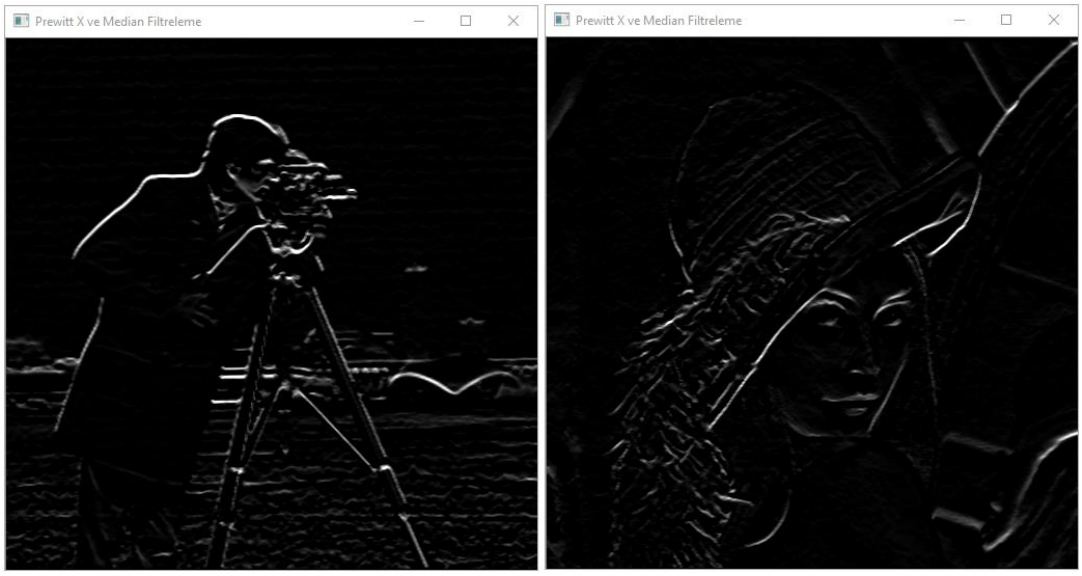
Prewitt X yöntemi OpenCV fonksiyonları arasından bulunmadığı için numpy kütüphanesinden faydalanılarak yapılmıştır. Prewitt X ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 10'da verilmekte ve Şekil 32'de bu yöntemin etkisi görülmektedir. Prewitt X ve Gaussian Filtreleme yönteminin ve Prewitt X ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 33 ve Şekil 34'de görülmektedir.

23. Prewitt X ve Gaussian Filtreleme



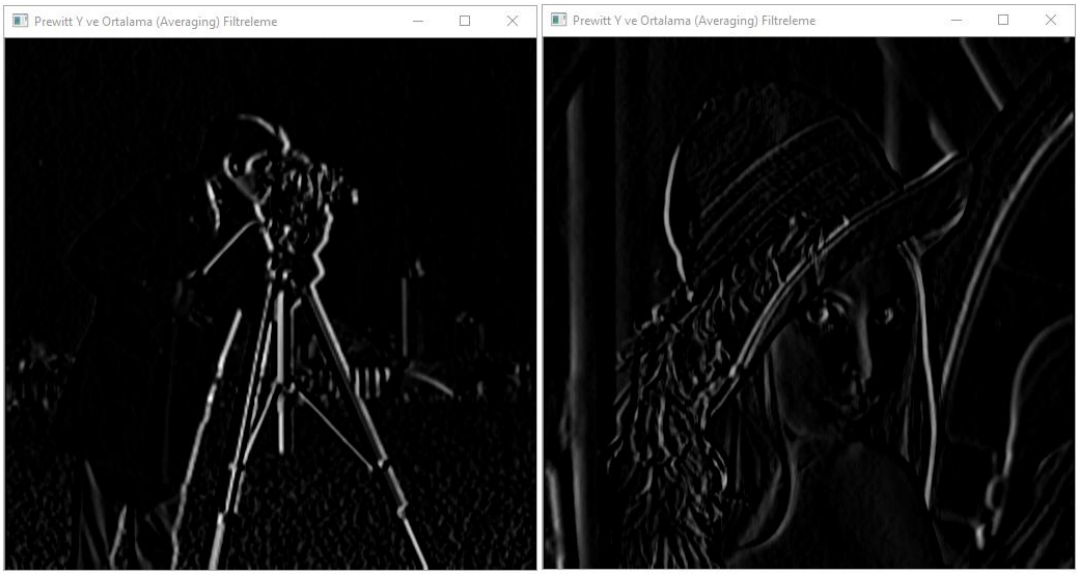
Şekil 33 Prewitt X ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

24. Prewitt X ve Medyan Filtreleme



Şekil 34 Prewitt X ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

25. Prewitt Y ve Ortalama Filtreleme



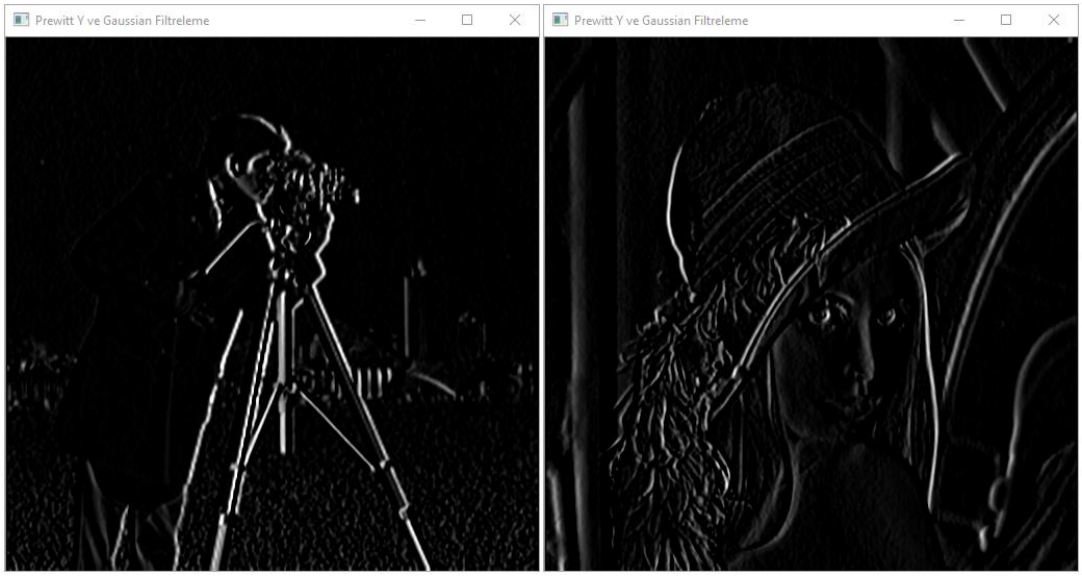
Şekil 35 Prewitt Y ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 11 Prewitt Y ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon

        blur = cv2.blur(gray, (int(self.ui.lineEdit.text()), int(self.ui.lineEdit_2.text())), 0)

        kernelY = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
prewittYEkseni = cv2.filter2D(blur, -1, kernelY)
prewittYRS = cv2.resize(prewittYEkseni, (512, 512))
cv2.imshow("Prewitt Y ve " + self.ui.comboBox.currentText(), prewittYRS)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

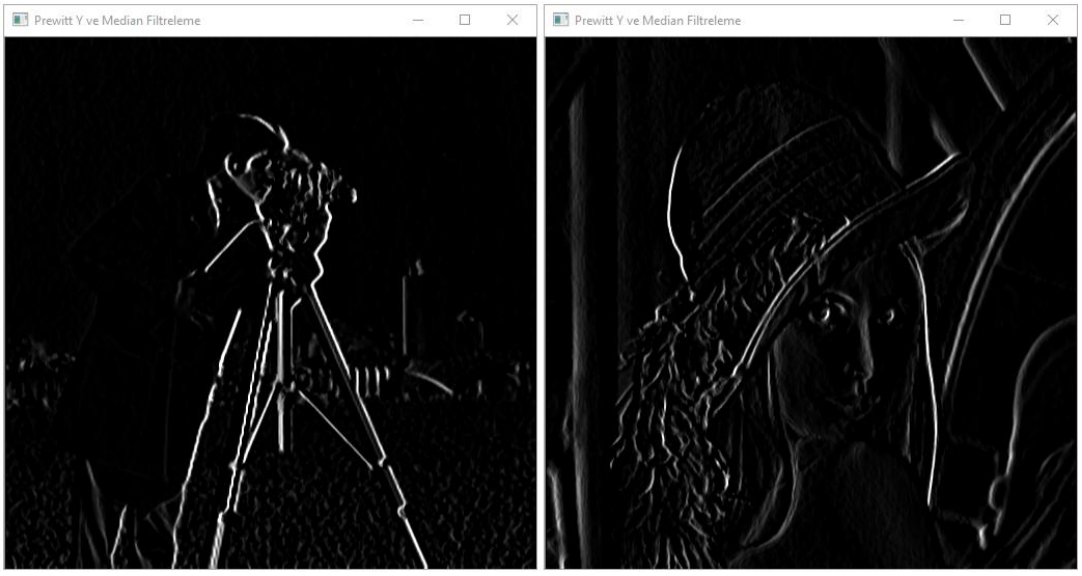
Prewitt Y yöntemi OpenCV fonksiyonları arasından bulunmadığı için numpy kütüphanesinden faydalanılarak yapılmış olup Prewitt X yönteminden farkı kernel matrisidir ve Şekil 35'de bu yöntemin etkisi görülmektedir. Prewitt Y ve Gaussian Filtreleme yönteminin ve Prewitt Y ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 36 ve Şekil 37'de görülmektedir.

26. Prewitt Y ve Gaussian Filtreleme



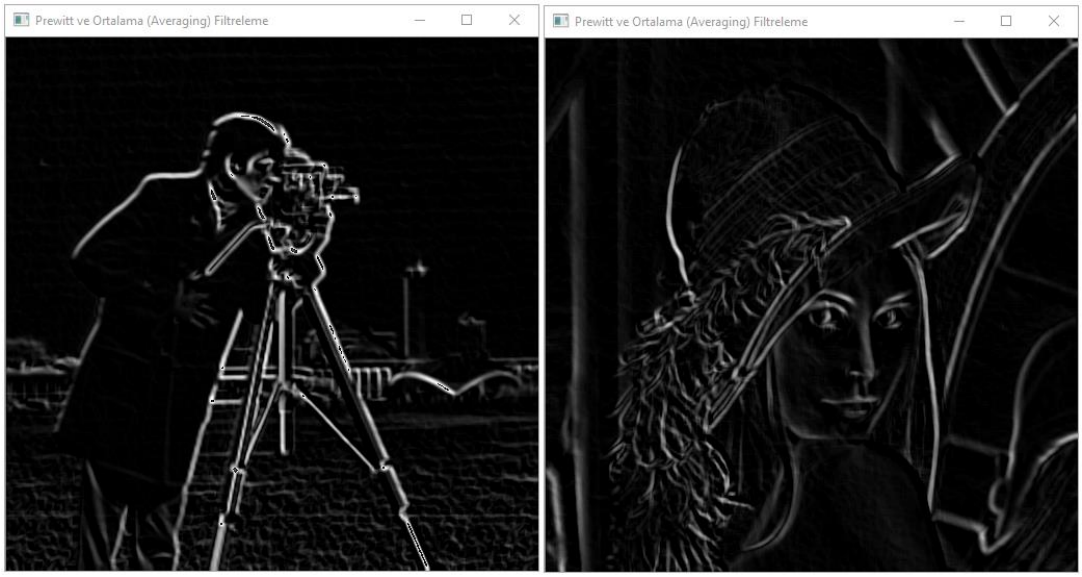
Şekil 36 Prewitt Y ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

27. Prewitt Y ve Medyan Filtreleme



Şekil 37 Prewitt Y ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

28. Prewitt ve Ortalama Filtreleme



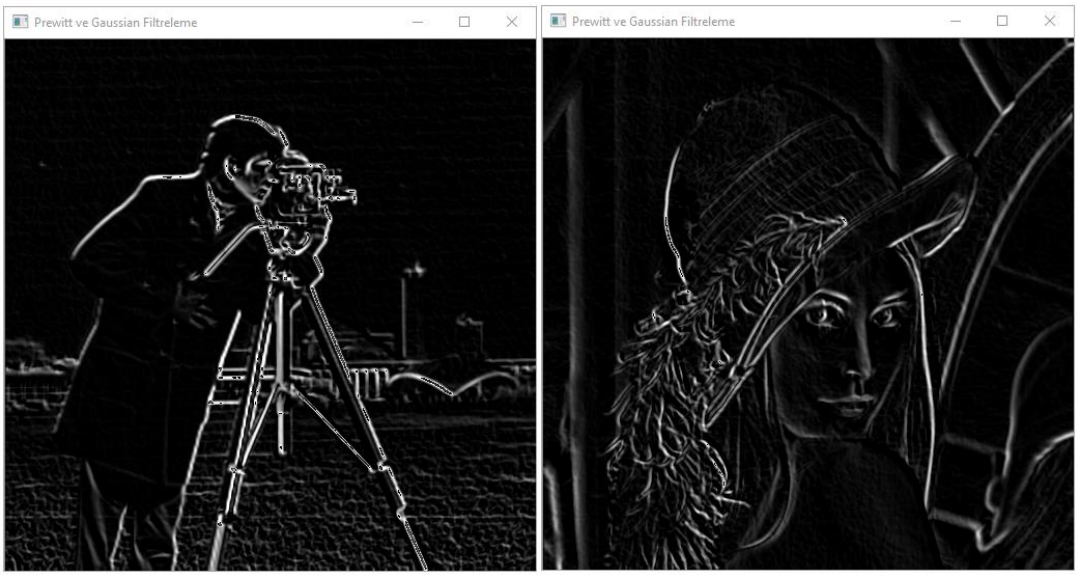
Şekil 38 Prewitt ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 12 Prewitt ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
cv2.imshow("Prewitt ve " + self.ui.comboBox.currentText(), prewittXRS + prewittYRS)
```

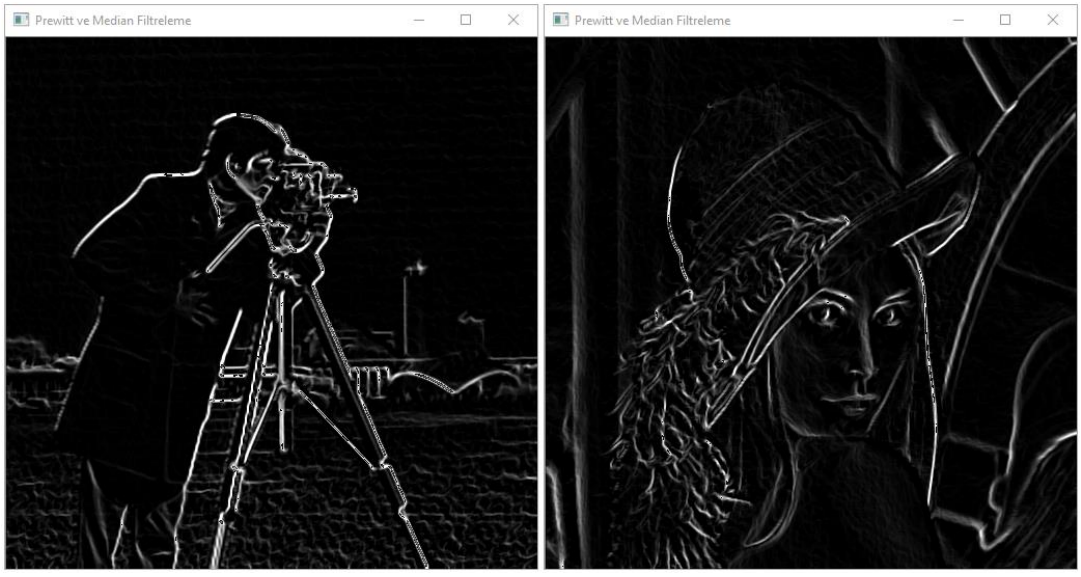
Prewitt yöntemi Prewitt X ve Prewitt Y yöntemlerinin birleşmiş halidir. Prewitt ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 12'de verilmekte ve Şekil 38'de bu yöntemin etkisi görülmektedir. Prewitt ve Gaussian Filtreleme yönteminin ve Prewitt ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 39 ve Şekil 40'da görülmektedir.

29. Prewitt ve Gaussian Filtreleme



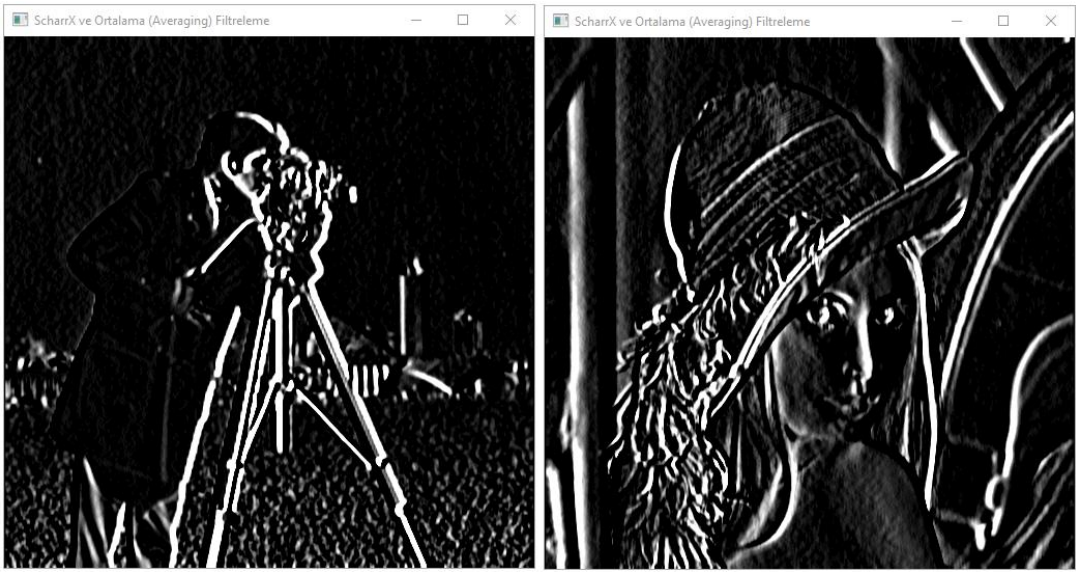
Şekil 39 Prewitt ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

30. Prewitt ve Medyan Filtreleme



Şekil 40 Prewitt ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

31. ScharrX ve Ortalama Filtreleme



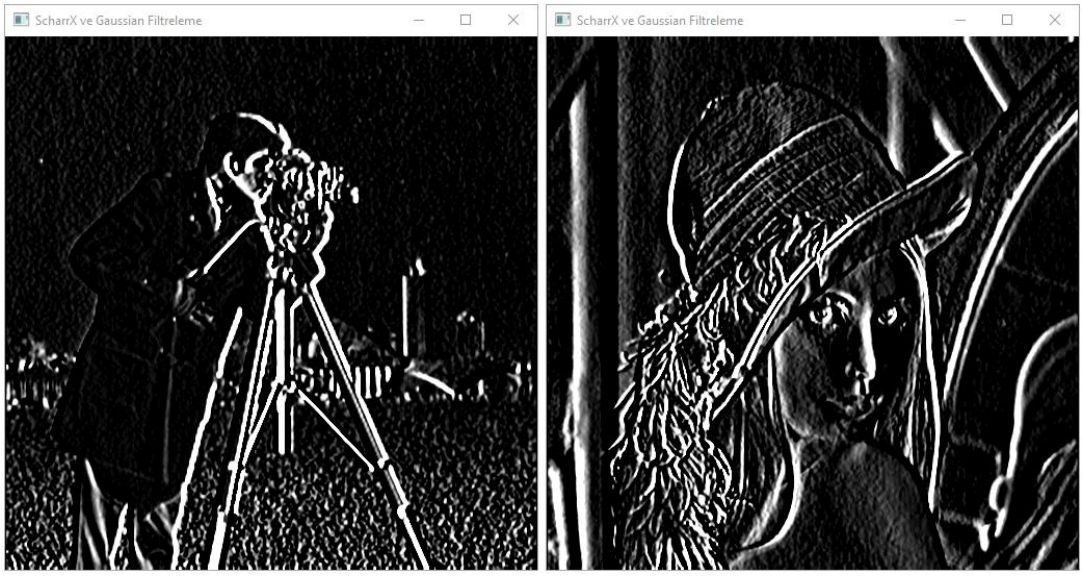
Şekil 41 ScharrX ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 13 ScharrX ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon
blur = cv2.blur(gray, (int(self.ui.lineEdit.text()), int(self.ui.lineEdit_2.text())), 0)
scharrX = cv2.Scharr(blur, cv2.CV_8U, 1, 0)
scharrXRS = cv2.resize(scharrX, (512, 512))
cv2.imshow("ScharrX ve " + self.ui.comboBox.currentText(), scharrXRS)
self.siniriBelirliGorsel = scharrXRS
cv2.waitKey(0)
cv2.destroyAllWindows()
```

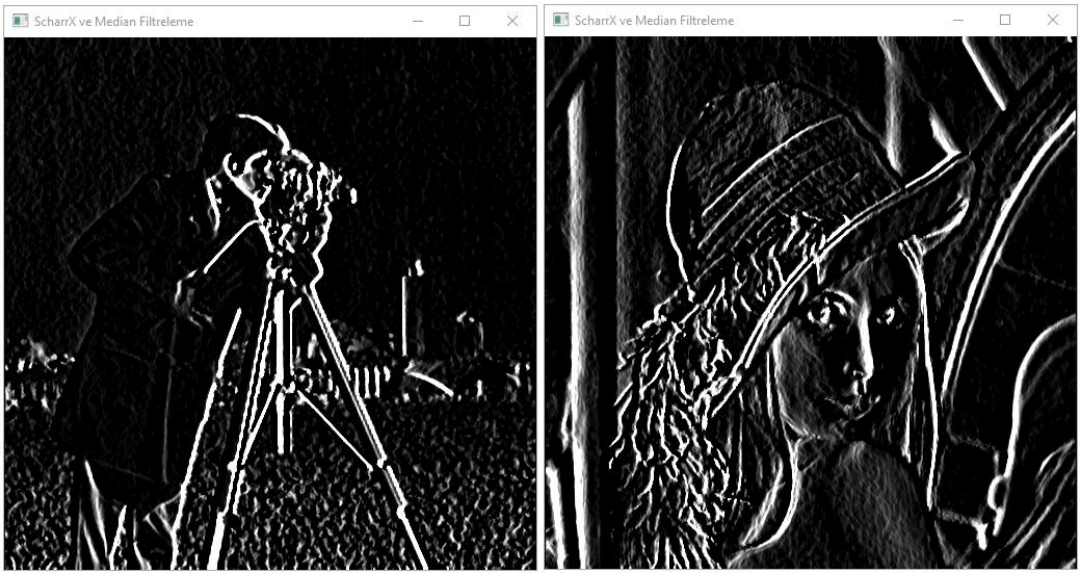
ScharrX fonksiyonunun aldığı parametreler bulanıklaştırılmış görüntü, görüntü derinliği, x ekseni ya da yatay olup olmadığı yani 1 ya da 0, y ekseni ya da dikey olup olmadığı yani 1 ya da 0 parametrelerini almaktadır. ScharrX ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 13'de verilmekte ve Şekil 41'de bu yöntemin etkisi görülmektedir. ScharrX ve Gaussian Filtreleme yönteminin ve ScharrX ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 42 ve Şekil 43'de görülmektedir.

32. ScharrX ve Gaussian Filtreleme



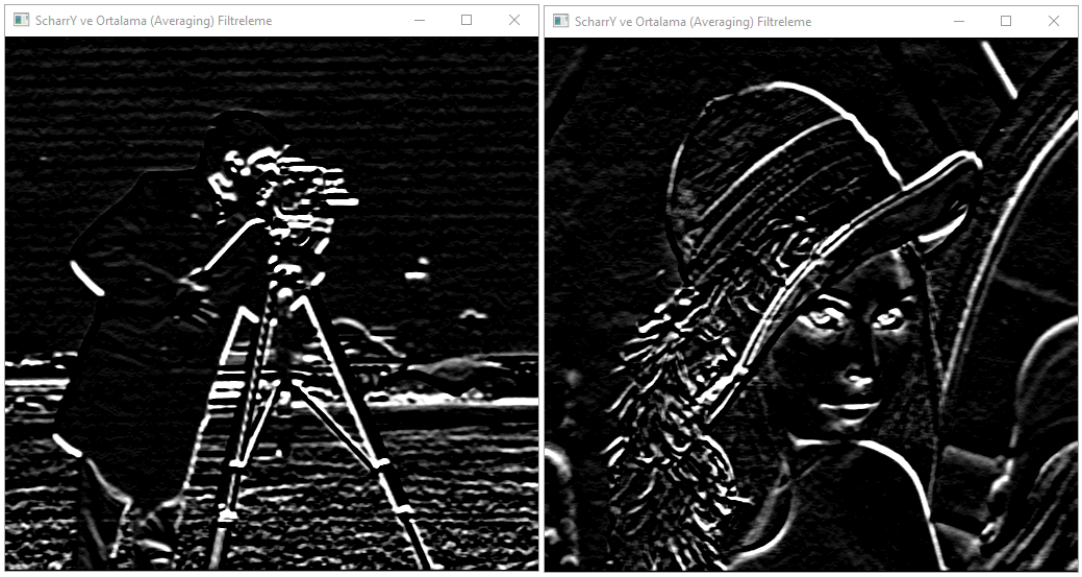
Şekil 42 ScharrX ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

33. ScharrX ve Medyan Filtreleme



Şekil 43 ScharrX ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

34. ScharrY ve Ortalama Filtreleme



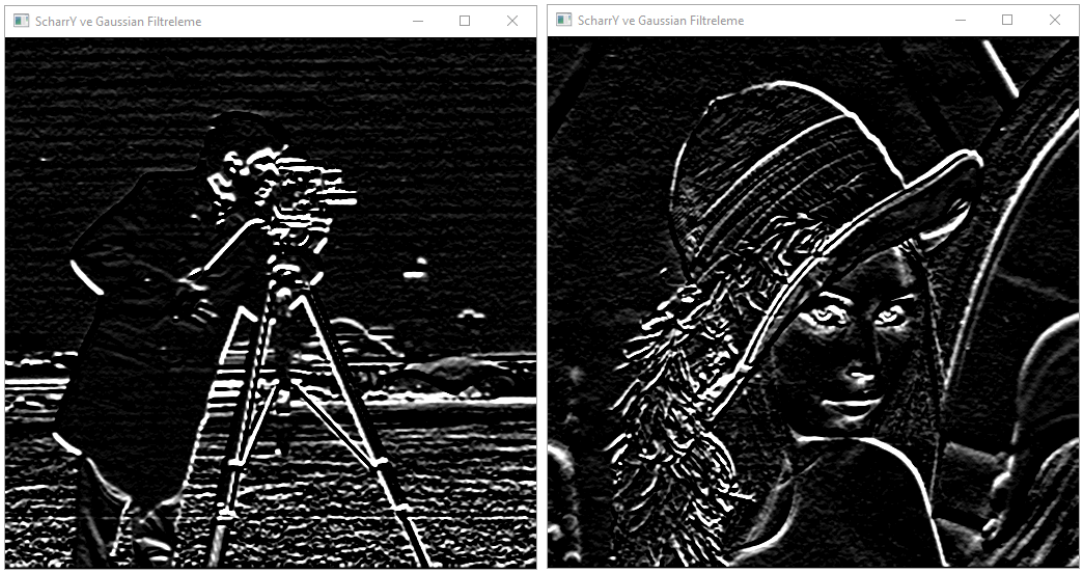
Şekil 44 ScharrY ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 14 ScharrY ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #görüntüyü siyah beyaz yapmak için kullanılan fonksiyon
blur = cv2.blur(gray, (int(self.ui.lineEdit.text()), int(self.ui.lineEdit_2.text())), 0)
scharrX = cv2.Scharr(blur, cv2.CV_8U, 1, 0)
scharrXRS = cv2.resize(scharrX, (512, 512))
cv2.imshow("ScharrX ve " + self.ui.comboBox.currentText(), scharrXRS)
self.siniriBelirliGorsel = scharrXRS
cv2.waitKey(0)
cv2.destroyAllWindows()
```

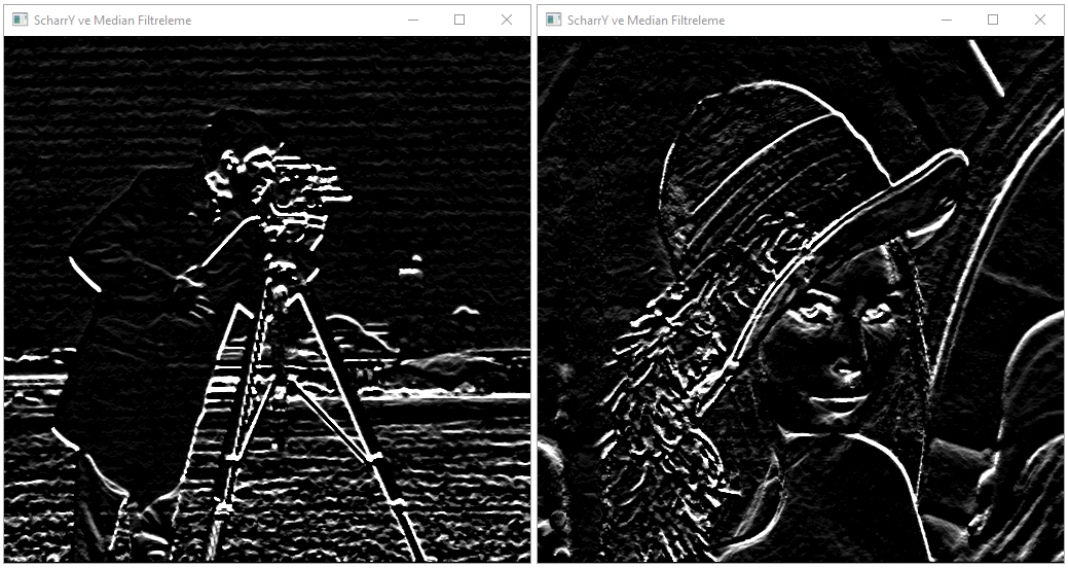
ScharrY ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 14'de verilmiş ve ScharrX yönteminden farkı kernel matrisidir ve Şekil 44'de bu yöntemin etkisi görülmektedir. ScharrY ve Gaussian Filtreleme yönteminin ve ScharrY ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 45 ve Şekil 46'da görülmektedir.

35. ScharrY ve Gaussian Filtreleme



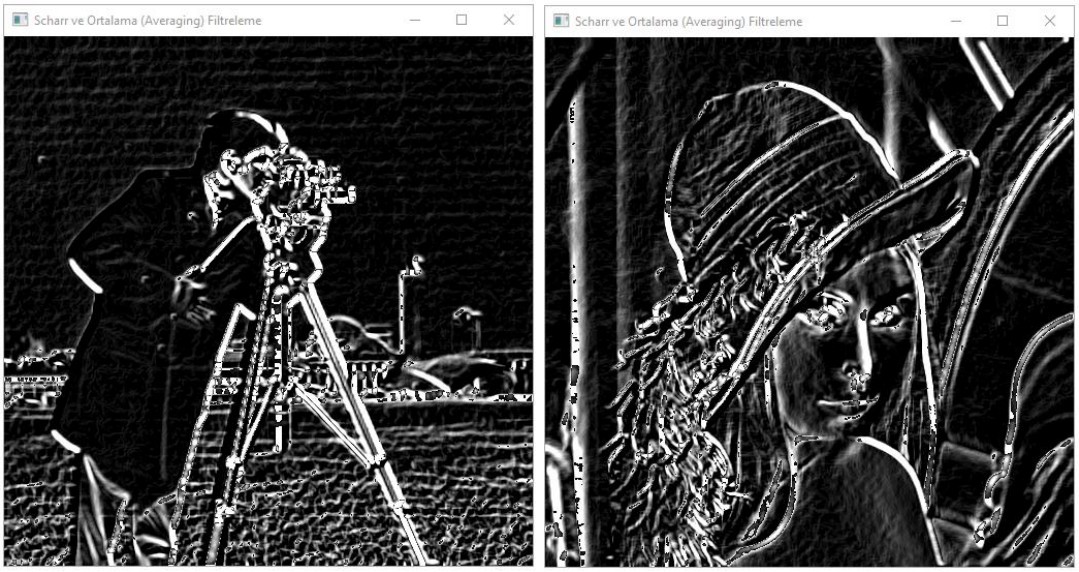
Şekil 45 ScharrY ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

36. ScharrY ve Medyan Filtreleme



Şekil 46 ScharrY ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

37. Scharr ve Ortalama Filtreleme



Şekil 47 Scharr ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 15 Scharr ve Ortalama Filtreleme yönteminin Cameraman ve Lenna görsellerine uygulanması

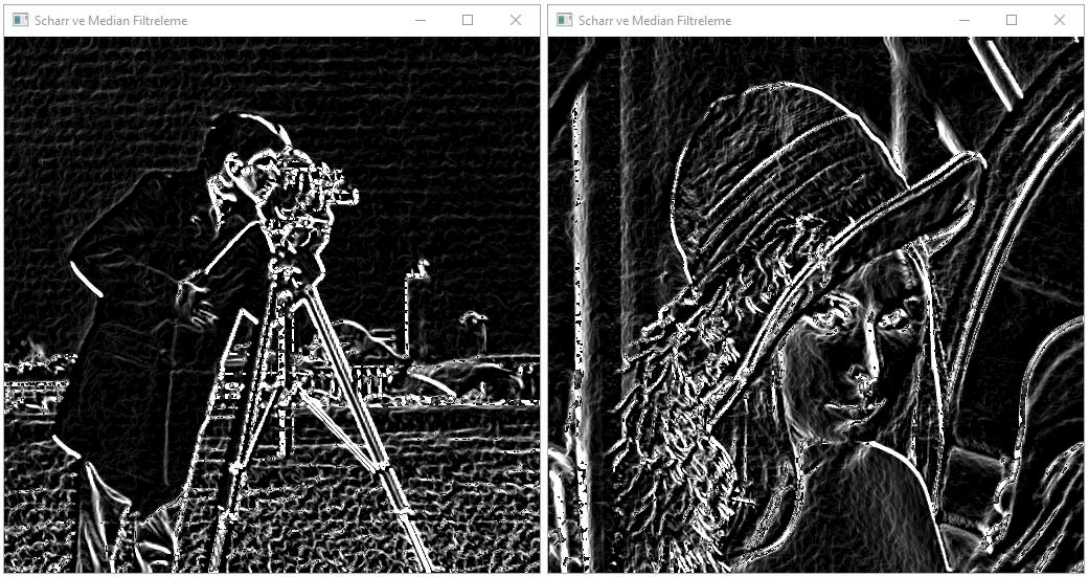
```
cv2.imshow("Scharr ve " + self.ui.comboBox.currentText(), scharrXRS + scharrYRS)
```

Scharr yöntemi ScharrX ve ScharrY yöntemlerinin birleşmiş halidir. Scharr ve Ortalama Filtreleme yönteminin uygulanışı Çizelge 15'de verilmekte ve Şekil 47'de bu yöntemin etkisi görülmektedir. Scharr ve Gaussian Filtreleme yönteminin ve Scharr ve Medyan Filtreleme yönteminin uygulanışı aynı olduğundan dolayı bu konuda çizelge verilmemiş ve bu yöntemlerin etkisi Şekil 48 ve Şekil 49'da görülmektedir.

38. Scharr ve Gaussian Filtreleme

Şekil 48 Scharr ve Gaussian Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

39. Scharr ve Medyan Filtreleme

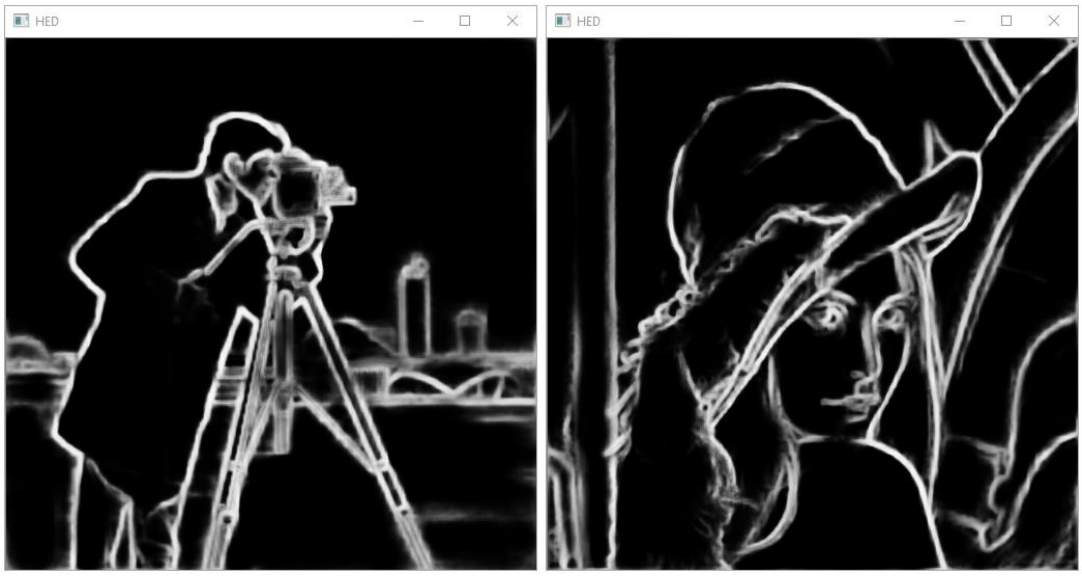


Şekil 49 Scharr ve Medyan Filtreleme yönteminin Cameraman ve Lenna görselindeki etkisi

B. Yapay Zekâ Uygulamaları

Derleyici olarak PyCharm 2021.1 kullanılmıştır. Python versiyonu olarak 3.8.5 kullanılmıştır. OpenCV versiyonu olarak 4.0.1 kullanılmıştır.

1. HED Yöntemi



Şekil 50 HED yönteminin Cameraman ve Lenna görselindeki etkisi

Çizelge 16 HED yönteminin Cameraman ve Lenna görsellerine uygulanması, (Sachan, 2021)

```
import cv2 as cv
import numpy as np
import argparse

parser = argparse.ArgumentParser(
    description=
        'Bütünsel İç İçe Kenar Algılama (https://arxiv.org/abs/1504.06375)'
        'https://github.com/s9xie/hed adresinden önceden eğitilmiş bir model kullanılmıştır.')
parser.add_argument('--input', help='Görüntü veya videonun yolu.', default='lena512color.jpg')
parser.add_argument('--write_video', help='Çıkış videosunun yazılıp yazılmayacağı', default=False)
parser.add_argument('--prototxt', help='konuşlandırma yolu.prototxt',default='deploy.prototxt', required=False)
parser.add_argument('--caffemodel', help='hed_pretrained_bsds.caffemodel e
```

Çizelge 17 HED yönteminin Cameraman ve Lenna görsellerine uygulanması Devam

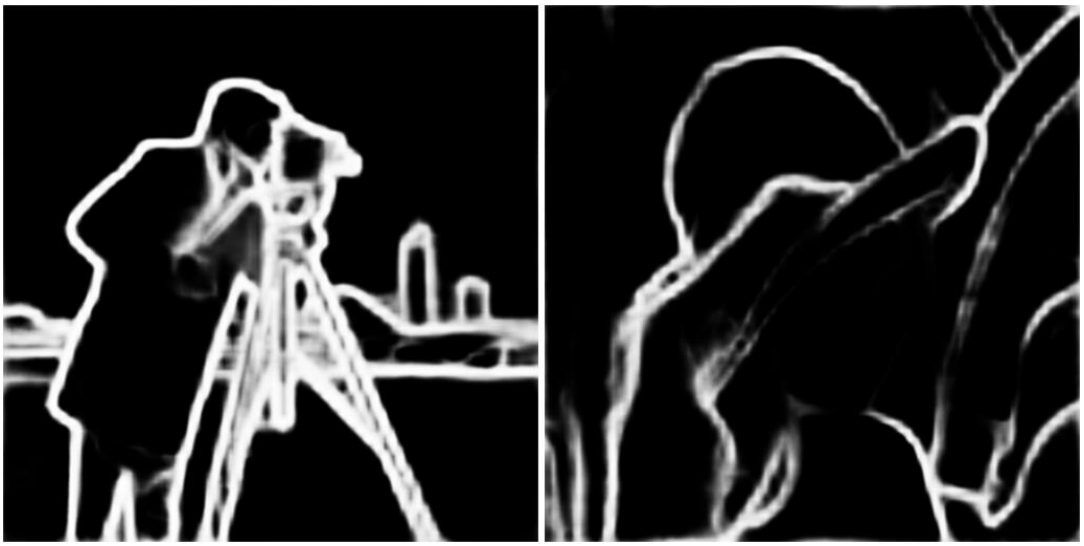
```
yol',default='hed_pretrained_bsds.caffemodel', required=False)
parser.add_argument('--width', help='Giriş görüntüsünü belirli bir genişliğe yeniden boyutlandırma', default=512, type=int)
parser.add_argument('--height', help='Giriş görüntüsünü belirli bir yüksekliğe yeniden boyutlandırma', default=512, type=int)
parser.add_argument('--savefile', help='Çıkış video yolunu belirtir', default='output.mp4', type=str)
args = parser.parse_args()

class CropLayer(object):
  def __init__(self, params, blobs):
    self.ystart = 0
    self.yend = 0
    self.xstart = 0
    self.xend = 0

  def getMemoryShapes(self, inputs):
    targetShape, inputShape = inputs[1], inputs[0]
    batchSize, numChannels = inputShape[0], inputShape[1]
    height, width = targetShape[2], targetShape[3]

    self.ystart = (inputShape[2] - targetShape[2]) // 2
    self.xstart = (inputShape[3] - targetShape[3]) // 2
    self.yend = self.ystart + height
    self.xend = self.xstart + width

    return [[batchSize, numChannels, height, width]]
  def forward(self, inputs):
    return [inputs[0][:,:,self.ystart:self.yend,self.xstart:self.xend]]
```

Çizelge 18 HED yönteminin Cameraman ve Lenna görsellerine uygulanması Devam

```
    cv.dnn_registerLayer('Crop', CropLayer)
# Modeli yükleme.
net = cv.dnn.readNet(args.prototxt, args.caffemodel)
cap = cv.VideoCapture(args.input if args.input else 0)

if args.write_video:
    h = int(cap.get(cv.CAP_PROP_FRAME_HEIGHT))
    w = int(cap.get(cv.CAP_PROP_FRAME_WIDTH))
    fourcc = cv.VideoWriter_fourcc(*'MP4V')
    writer = cv.VideoWriter(args.savefile, fourcc, 25, (w, h))
while cv.waitKey(1) < 0:
    hasFrame, frame = cap.read()
    if not hasFrame:
        cv.waitKey()
        break
    inp = cv.dnn.blobFromImage(frame, scalefactor=1.0, size=(args.width,
args.height),
                               mean=(104.00698793, 116.66876762, 122.67891434),
                               swapRB=False, crop=False)
    net.setInput(inp)
    out = net.forward()
    out = out[0, 0]
    out = cv.resize(out, (frame.shape[1], frame.shape[0]))
    out = 255 * out
    out = out.astype(np.uint8)
    out = cv.cvtColor(out, cv.COLOR_GRAY2BGR)
    con = np.concatenate((frame, out), axis=1)
    if args.write_video:
        writer.write(np.uint8(con))
    HED = cv.resize(out, (512, 512))
    cv.imshow("HED", HED)
```

HED yöntemi kullanılırken NumPy kütüphanesinden ve Caffe çatısından (framework) faydalanılmıştır. Caffe, multimedya bilimcilerine ve uygulayıcılarına son teknoloji derin öğrenme algoritmaları ile bir referans yapıları koleksiyonu için güncellenebilir ve temiz bir çerçeve sağlamakta olup genel amaçlı evrişimli sinir ağlarını ve diğer derin modelleri meta yapıları üzerinde verimli bir şekilde eğitmek ve dağıtmak için MATLAB ve Python bağlamaları bulunan BSD lisanslı bir C++ kütüphanesidir, (Jia, ve diğerleri, 2014). HED yönteminin uygulanışı Çizelge 16, Çizelge 17 ve Çizelge 18'de verilmekte ve Şekil 50'de bu yöntemin etkisi görülmektedir.

2. RCF Yöntemi



Şekil 51 RCF yönteminin Cameraman ve Lenna görselindeki etkisi

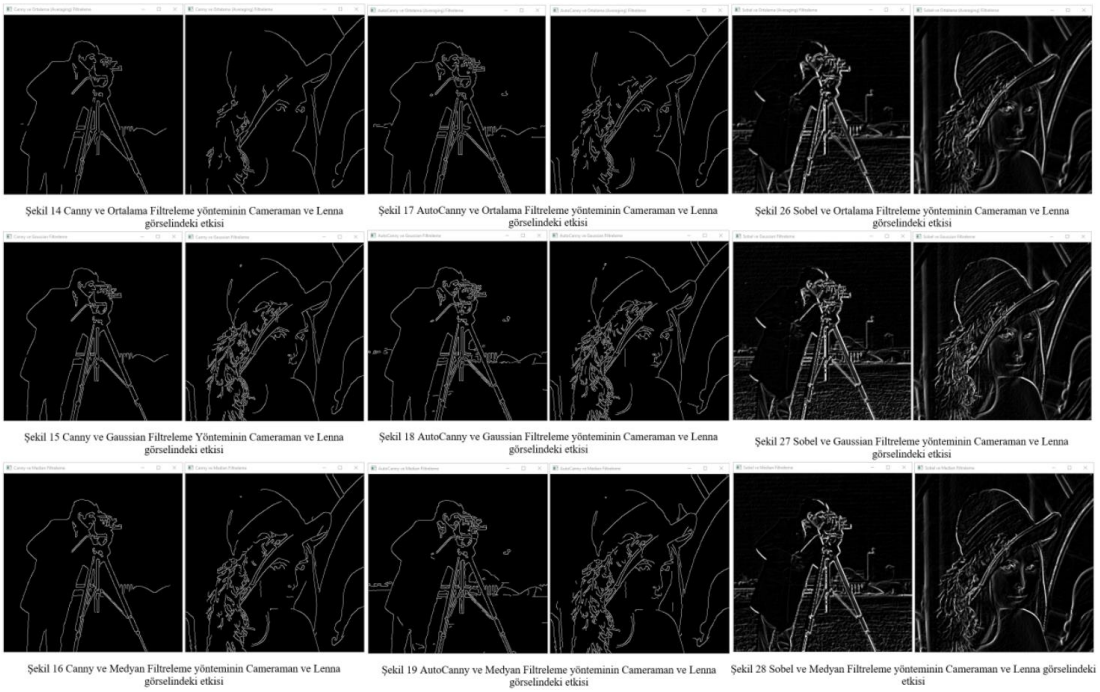
Çizelge 19 RCF yönteminin Cameraman ve Lenna görsellerine uygulanması, (Liangyu, 2021)

```
import torch
import models
import cv2
import numpy as np
from PIL import Image

from data_loader import prepare_image_cv2

resume = 'ckpt/lr-0.01-iter-490000.pth'
img_path = './examples/all_layer/lena512color.jpg'
result_path = './examples/lena512color_rcf.png'

model = models.resnet101(pretrained=False).cuda()
model.eval()

checkpoint = torch.load(resume)
model.load_state_dict(checkpoint)

original_img = np.array(cv2.imread(img_path), dtype=np.float32)
h, w, _ = original_img.shape

img = prepare_image_cv2(original_img)
img = torch.from_numpy(img).unsqueeze(0).cuda()

outs = model(img, (h, w))
result = outs[-1].squeeze().detach().cpu().numpy()

result = (result * 255).astype(np.uint8)
Image.fromarray(result).save(result_path)
```

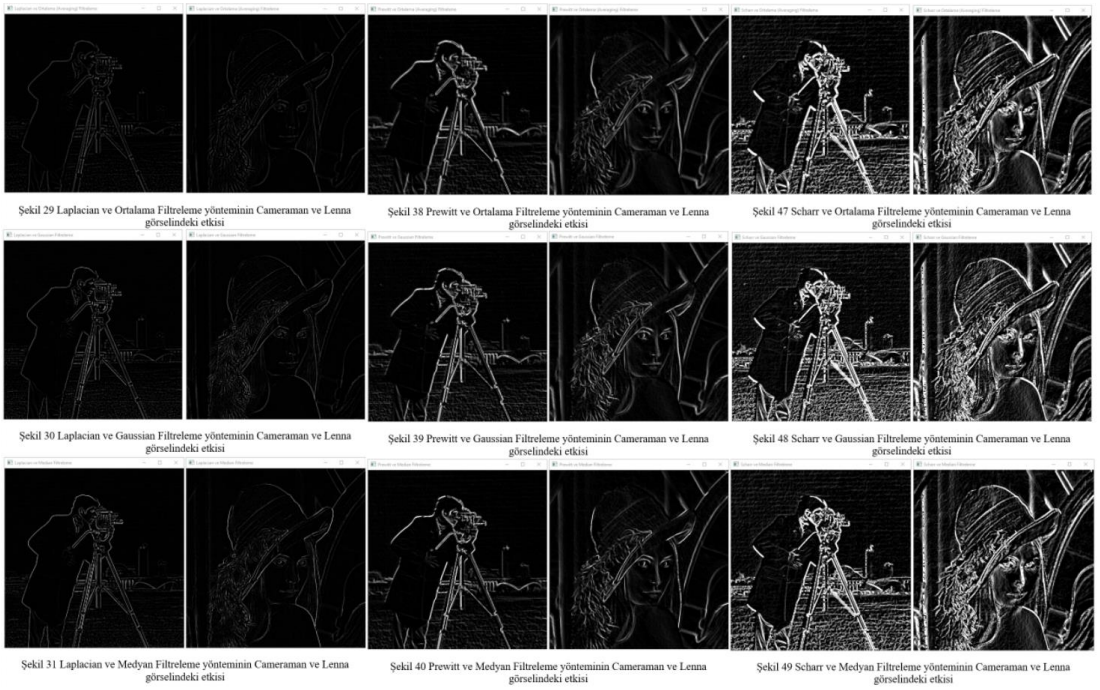
RCF yöntemi kullanılırken NumPy, PyTorch, OpenCV ve Pillow kütüphanelerinden faydalanılmıştır. Derin öğrenme çerçeveleri genellikle

kullanılabilirliğe ya da hıza odaklanır ancak PyTorch her ikisine birden odaklanarak bu iki hedefin aslında uyumlu olduğunu gösteren bir makine öğrenimi kütüphanesidir, (Paszke, ve diğerleri, 2019). Model olarak kodu destekleyen, hata ayıklamayı kolaylaştıran ve diğer popüler bilimsel bilgi işlem kitaplıklarıyla tutarlı, verimli ve destekleyici kalırken GPU'lar gibi donanım hızlandırıcıları da kullanırken Pythonic programlama stilini de beraberinde sağlamaktadır, (Paszke, ve diğerleri, 2019). RCF yönteminin uygulanışı Çizelge 19'da verilmekte ve Şekil 51'de bu yöntemin etkisi görülmektedir.

V. SONUÇ VE ÖNERİLER



Şekil 52 Canny, AutoCanny, Sobel, Ortalama, Gaussian ve Medyan Filtreleme görüntüleri



Şekil 53 Laplacian, Prewitt, Scharr, Ortalama, Gaussian, Medyan Filtreleme görüntüleri

Şekil 50 HED yönteminin Cameraman ve Lenna görselindeki etkisi

Şekil 51 RCF yönteminin Cameraman ve Lenna görselindeki etkisi

Şekil 54 HED ve RCF görüntüleri

Filtreleme yöntemlerinin sınır belirleme üzerinde etkisi olduğu görülmüştür. Bu tez çalışmasında Ortalama, Gaussian ve Medyan filtreleme yöntemleri kullanmış olup Medyan Filtrelemenin sınırları daha keskin belirlemesinden dolayı Medyan Filtreleme ile sınır belirleme için daha iyi sonuçlar elde edilmiştir. Aynı zamanda Medyan Filtreleme yöntemi ile daha az gürültülü bir sonuç ortaya çıkmış ve Şekil 52 ve Şekil 53'de görülmektedir.

Ortalama filtreleme yönteminde Gaussian ve Medyan filtreleme yöntemlerine göre kenarlar daha fazla yumuşatılarak daha bulanık bir görüntü elde edilmiştir. (Li, Wang, & Yao, 2021), su altı görüntüler üzerinde Ortalama filtreleme yöntemini kullanarak görüntü üzerindeki gürültüyü azaltma çalışmaları yapmış ve benzeri sonuç elde etmiştir.

Gaussian filtreleme yönteminde Ortalama filtreleme yöntemine göre daha az bulanık olmasına rağmen kenar keskinliği Ortalama filtreleme yöntemine göre daha fazla Medyan filtreleme yöntemine göre daha az olan bir görüntü elde edilmiştir. (Reddy & Jaya, 2021), tıbbi görüntüler üzerinde Gaussian filtreleme yöntemini kullanarak yumuşatma çalışmaları yapmış ve benzeri sonuç elde etmiştir.

Medyan filtreleme yönteminde Ortalama filtreleme yöntemine göre daha az bulanık olmasına rağmen kenar keskinliği Ortalama ve Gaussian filtreleme yöntemine göre daha fazla olan bir görüntü elde edilmiştir. (Appiah, Asante, Benjamin, & Hayfron-Acquah, 2021), Lenna, Cameraman gibi görseller üzerinde gürültü azaltma çalışmaları yapmış ve benzeri sonuç elde etmiştir.

Bu tez çalışmasında Canny, AutoCanny, Sobel, Laplacian, Prewitt ve Scharr görüntü işleme yöntemleri ile sonuç elde edilmiş olup Canny yönteminin daha iyi sonuç verdiği görülmüştür.

Canny yönteminde Medyan filtreleme yöntemi ile birlikte Ortalama ve Gaussian filtreleme yöntemlerine göre görüntü sınırlarının daha iyi belirlendiği görülmüştür. (Yang, Zhao, Huang, Wang, & Zhu, 2021), patates üzerinde Canny yöntemini kullanarak sınır belirleme çalışmaları yapmış ve benzeri sonuç elde etmiştir.

AutoCanny yönteminde Medyan filtreleme yöntemi ile birlikte Ortalama ve Gaussian filtreleme yöntemlerine göre görüntü sınırlarının daha iyi belirlendiği görülmüştür.

Canny yöntemi AutoCanny'e göre daha iyi sonuç verse de Canny yönteminin parametreleri görüntüye göre ayarlanması gerekirken AutoCanny'de böyle bir ihtiyaç olmadığından dolayı AutoCanny yöntemi Canny yöntemine göre daha hızlı çalışmaktadır.

Laplacian yönteminde Medyan filtreleme yöntemi ile birlikte Ortalama ve Gaussian filtreleme yöntemlerine göre görüntü sınırlarının daha iyi belirlendiği görülmüştür. (Tsuchimoto, ve diğerleri, 2021), Elektroensefalogram (EEG) sinyal-gürültü oranlarını azaltmak amacıyla çalışmalar yapmış ve benzeri sonuç elde etmiştir.

SobelX yönteminde Gaussian filtreleme yöntemi ile birlikte Ortalama ve Medyan filtreleme yöntemlerine göre daha net bir görüntü elde edildiği görülmüş ama görüntü üzerindeki gürültülerin Ortalama ve Medyan filtreleme yöntemindeki

gürültülere göre daha fazla olduğu görülmüştür. Bu yüzden Medyan filtreleme yöntemi daha başarılıdır.

SobelY yönteminde Gaussian filtreleme yöntemi ile birlikte Ortalama ve Medyan filtreleme yöntemlerine göre daha net bir görüntü elde edildiği görülmüş ama görüntü üzerindeki gürültülerin Ortalama ve Medyan filtreleme yöntemindeki gürültülere göre daha fazla olduğu görülmüştür. Bu yüzden Medyan filtreleme yöntemi daha başarılıdır.

Sobel yönteminde Gaussian filtreleme yöntemi ile birlikte Ortalama ve Medyan filtreleme yöntemlerine göre daha net bir görüntü elde edildiği görülmüş ama görüntü üzerindeki gürültülerin Ortalama ve Medyan filtreleme yöntemindeki gürültülere göre daha fazla olduğu görülmüştür. Bu yüzden Medyan filtreleme yöntemi daha başarılıdır. (Sharifrazi, ve diğerleri, 2021), X-ray görüntülerini kullanarak COVID-19 hastalarının doğru tespiti için Sobel filtresini sınır belirleme amacıyla kullanmış ve benzeri sonuç elde etmiştir.

Prewitt X yönteminde Gaussian filtreleme yöntemi ile birlikte Ortalama ve Medyan filtreleme yöntemlerine göre daha net bir görüntü elde edildiği görülmüş ama görüntü üzerindeki gürültülerin Ortalama ve Medyan filtreleme yöntemindeki gürültülere göre daha fazla olduğu görülmüştür. Bu yüzden Medyan filtreleme yöntemi daha başarılıdır.

Prewitt Y yönteminde Gaussian filtreleme yöntemi ile birlikte Ortalama ve Medyan filtreleme yöntemlerine göre daha net bir görüntü elde edildiği görülmüş ama görüntü üzerindeki gürültülerin Ortalama ve Medyan filtreleme yöntemindeki gürültülere göre daha fazla olduğu görülmüştür. Bu yüzden Medyan filtreleme yöntemi daha başarılıdır.

Prewitt yönteminde Gaussian filtreleme yöntemi ile birlikte Ortalama ve Medyan filtreleme yöntemlerine göre daha net bir görüntü elde edildiği görülmüş ama görüntü üzerindeki gürültülerin Ortalama ve Medyan filtreleme yöntemindeki gürültülere göre daha fazla olduğu görülmüştür. Bu yüzden Medyan filtreleme yöntemi daha başarılıdır. (Dagar & Dahiya, 2020), İkili Parçacık Sürü Algoritmasını kullanarak geliştirdikleri sınır belirleme yöntemiyle Prewitt ve Canny gibi yöntemleri kıyaslamış Prewitt yöntemini kıyaslama aşamasında kullandığı görüntülerde aldığı

sonuçlar ile bu tezdeki çalışmalar ile elde edilen sonuçlar ile benzeri sonuçlar elde etmiştir.

ScharrX yönteminde Gaussian filtreleme yöntemi ile birlikte Ortalama ve Medyan filtreleme yöntemlerine göre daha net bir görüntü elde edildiği görülmüş ama görüntü üzerindeki gürültülerin Ortalama ve Medyan filtreleme yöntemindeki gürültülere göre daha fazla olduğu görülmüştür. Bu yüzden Medyan filtreleme yöntemi daha başarılıdır.

ScharrY yönteminde Gaussian filtreleme yöntemi ile birlikte Ortalama ve Medyan filtreleme yöntemlerine göre daha net bir görüntü elde edildiği görülmüş ama görüntü üzerindeki gürültülerin Ortalama ve Medyan filtreleme yöntemindeki gürültülere göre daha fazla olduğu görülmüştür. Bu yüzden Medyan filtreleme yöntemi daha başarılıdır.

Scharr yönteminde Gaussian filtreleme yöntemi ile birlikte Ortalama ve Medyan filtreleme yöntemlerine göre daha net bir görüntü elde edildiği görülmüş ama görüntü üzerindeki gürültülerin Ortalama ve Medyan filtreleme yöntemindeki gürültülere göre daha fazla olduğu görülmüştür. Bu yüzden Medyan filtreleme yöntemi daha başarılıdır. (Alshdadi, ve diğerleri, 2020), modifiye edilmiş Scharr filtreleme kullanarak parmak izlerinin düşük seviye detaylarını çıkarma çalışmaları yapmış ve benzeri sonuçlar elde etmiştir.

Sobel, Prewitt ve Scharr yöntemlerinde X ve Y'nin birlikte kullandığı yöntemin sadece X ya da sadece Y yöntemine göre daha iyi sonuç verdiği görülmüştür.

Yapay Zekâ yöntemlerinden faydalanarak geliştirilen HED ve RCF yöntemlerinin görüntü işleme yöntemlerine göre daha iyi sonuç verdiği görülmüştür. Ancak diğer yöntemlere göre daha yavaş çalışmaktadır. Ayrıca diğer yöntemlere göre elde edilen sonuç biraz daha bulanık bir yapıya sahiptir.

HED yönteminde görüntü işleme yöntemlerine göre daha iyi sonuçlar elde edilmiş fakat görüntü işleme yöntemlerine göre daha bulanık ama RCF yöntemine göre daha net sonuç verdiği görülmüştür. (Rampun, ve diğerleri, 2019), modifiye edilmiş HED yöntemiyle mamogramlarda meme pektoral kas segmentasyonu üzerine çalışmalar yapmış ve benzeri sonuçlar elde etmiştir.

RCF yönteminde görüntü işleme yöntemlerine göre daha iyi sonuçlar elde edilmiş fakat görüntü işleme ve HED yöntemlerine göre daha bulanık sonuçlar elde

edilmesine rağmen görüntü üzerindeki sınırların daha iyi belirlendiği görülmüş ve bu durumdan dolayı RCF yönteminin diğer yöntemlere göre daha başarılı olduğu görülmüştür, (Şekil 54). (Yueming, ve diğerleri, 2019), Sanduao'da RCF yöntemine dayalı su ürünleri alanı çıkarma ve güvenlik açığı değerlendirmesi çalışmaları yapmış ve benzeri sonuçlar elde etmiştir.

HED ve RCF yöntemlerinin çalışma süreleri PyCharm derleyicisinde yer alan Profiler ile elde edilmiş olup Çizelge 20 ve Çizelge 21'de çalışma süreleri görülmektedir.

Çizelge 20 HED yönteminin çalışma süresi

Name	Call Count	Time (ms)	Own Time (ms)
HED	1	3867 %52,5	4 %0,1

Çizelge 21 RCF yönteminin çalışma süresi

Name	Call Count	Time (ms)	Own Time (ms)
RCF	1	4472 %100,0	1 %0,0

Öneriler olarak, Caffe gibi yeni bir makine öğrenmesi modeli geliştirilerek daha iyi bir öğrenme sağlanarak daha iyi sonuçlar ele edilebilir ya da HED, RCF gibi yöntemlerden esinlenerek yeni bir yöntem geliştirilerek daha iyi sonuçlar elde edilebilir. Ayrıca yönlü dalgacık dönüşümüne (directional wavelet transform) dayalı bir sınır belirleme yöntemi geliştirilerek daha iyi sonuçlar elde edilebilir, (Zhang, Ma, Liu, & Gong, 2009).

VI. KAYNAKÇA

KİTAPLAR

AKSOY, A. (2019). **OpenCV ve Python ile Eğlenceli Projeler ve Oyunlar**, İstanbul: Abaküs.

AKSOY, A. (2021). **Yeni Başlayanlar için Python**, İstanbul: Abaküs.

AKSOY B. (2021). **Python ile İmgeden Veriye Görüntü İşleme ve Uygulamaları**, Ankara: Nobel.

ARSLAN, İ. (2021). **Python ile Veri Bilimi**, İstanbul: Pusula.

AYDEMİR, M. (2020). **Python**, İstanbul: Kodlab.

ÇOBANOĞLU, B. (2021). **Herkes için Python**, İstanbul: Pusula.

ELMAS, Ç. (2018). **Yapay Zeka Uygulamaları**, Ankara: Seçkin.

JÄHNE, B. (2005). **Digital Image Processing**, Berlin Heidelberg: Springer.

JAIN, A. K. (1989). **Fundamentals of Digital Image Processing**, Englewood Cliffs: Prentice Hall Information and System Sciences Series.

KUYUMCU, B. (2018). **OpenCV Görüntü İşleme ve Yapay Öğrenme**, İstanbul: Level.

NABİYEV, P. D. (2012). **Yapay Zeka**, Ankara: Seçkin.

ÖZGÜL, F. (2011). **Python**, İstanbul: Kodlab.

RUSS, J. C. (2016). **The Image Processing Handbook**, CRC Press.

TAŞCI, V. (2019). **Python Eğitim Kitabı**, İstanbul: Dikeyeksen.

YILDIZ, B. (2020). **Python Projeler ve Popüler Kütüphaneler**, İstanbul: Kodlab.

YILMAZ, A. (2019). **YAPAY ZEKA**, İstanbul: KODLAB.

YILMAZ, A., & KAYA, U. (2019). **Derin Öğrenme**, İstanbul: Kodlab.

YOUNG, I. T., GERBRANDS, J. J., & VLIET, L. J. (1998). **Fundamentals of Image Processing**, Netherlands: Delft University of Technology.

MAKALELER

ALSHDADI, A. A., MEHBOOB, R., DAWOOD, H., ALASSAFI, M. O., ALGHAMDİ, R., & DAWOOD, H. (2020). "Exploiting Level 1 and Level 3 features of fingerprints for liveness detection", **Biomedical Signal Processing and Control.**

APPIAH, O., ASANTE, M., BENJAMIN, J., & HAYFRON-ACQUAH. (2021). "Improved approximated median filter algorithm for real-time computer vision applications", **Journal of King Saud University - Computer and Information Sciences**.

ATALI, G., ÖZKAN, S. S., & KARAYEL, D. (2016). "Morfolojik Görüntü İşleme Tekniği ile Yapay Sinir Ağlarında Görüntü Tahribat Analizi", **Academic Platform Journal of Engineering and Science,** 01-07.

AYBAR, E. (2008). "Sobel İşleci Kullanılarak Renkli Görüntülerde", **Afyon Kocatepe Üniversitesi Fen Bilimleri Dergisi**, 205-217.

BELLAIRE, G., TALMI, K., OEZGUER, E., & KOSCHAN, A. (1998). "Object Recognition Obtaining 2-D Reconstructions From Color Edges", **Proceedings IEEE Symposium on Image Analysis and Interpretation**, 192-197.

BOVIK, A. C. (2005). "Handbook of Image and Video Processing (Communications, Networking and Multimedia)", **Elsevier Academic Press**.

DAGAR, N. S., & DAHIYA, P. K. (2020). "Edge Detection Technique using Binary Particle Swarm Optimization", **Procedia Computer Science 167**, 1421-1436.

DEĞİRMENCİ, A., ÇANKAYA, İ., & DEMİRCİ, P. D. (2018). "Gradyan Anahtarlamalı Gauss Görüntü Filtresi", **Düzce Üniversitesi Bilim ve Teknoloji Dergisi**, 196-215.

DING, L., GOSHTASBY, A., & SATTER, M. (2001). "Volume image registration by template matching", **Image and vision computing**, 821-832.

DUBOIS, P. F. (2007). "Guest Editor's Introduction Python: Batteries Included", **Computing in Science and Engineering**, 9(3):7-9.

GONZALEZ, R. C., & WOODS, R. E. (2002). “Digital Image Processing Second Edition”, **New Jersey: Prentice-Hall**.

GÜRAKSIN, G. E. (2018). “Tuz Biber Gürültülerinin Giderilmesi için k-Oralama Algoritması Tabanlı Filtre Tasarımı”, **Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi**, 972-978.

HALL, M. (2007). “Smooth operator: Smoothing seismic interpretations and attributes”, **The Leading Edge**, 16-20.

HANBAY, K., TALU, M. F., & ÖZGÜVEN, Ö. F. (2017). “Fourier dönüşümü kullanılarak gerçek zamanlı kumaş hatası tespiti”, **Journal of the Faculty of Engineering and Architecture of Gazi University**, 151-158.

JAVADZADEH, R., BANIHASHEMI, E., & HAMIDZADEH, J. (2015). “Fast Vehicle Detection and Counting Using Background Subtraction Technique and Prewitt Edge Detection”, **International Journal of Computer Science and Telecommunications**, 8-12.

JIA, Y., SHELHAMER, E., DONAHUE, J., KARAYEV, S., LONG, J., GIRSHICK, R., . . . Darrell, T. (2014). “Caffe: Convolutional Architecture for Fast Feature Embedding”, **arXiv preprint arXiv:1408.5093**.

LI, T., WANG, J., & YAO, K. (2021). “Visibility enchancement of underwater images based on active polarized illumination and average filtering technology”, **Alexandria Engineering Journal**.

LIU, Y., CHENG, M.-M., HU, X., BIAN, J.-W., ZHANG, L., BAI, X., & TANG, J. (2019). “Richer Convolutional Features for Edge Detection”, **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 1939-1946.

NEUMAN, M. R., SAPIRSTEIN, H. D., SHWEDYK, E., & BUSHUK, W. (1989). “Wheat grain colour analysis by digital image processing I. Methodology”, **Journal of Cereal Science**, 175-182.

NGUYEN, G., DLUGOLINSKY, S., BOBÁK, M., TRAN, V., GARCÍA, Á. L., HEREDIA, I., . . . HLUCHÝ, L. (2019). “Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey”, **Springer**.

NIXON, M., & AGUADO, A. (2019). “Feature Extraction and Image Processing for Computer Vision”, **Elsevier Academic Press**.

RAMPUN, A., LÓPEZ-LINARES, K., MORROW, P. J., SCOTNEY, B. W., WANG, H., OCAÑA, I. G., . . . MACÍA, I. (2019). “Breast pectoral muscle segmentation in mammograms using a modified holistically-nested edge detection network”, **Medical Image Analysis**, 1-17.

RAO, D. D. (2007). “A Survey on Image Enhancement Techniques: Classical Spatial Filter,Neural Network, Cellular Neural Network, and Fuzzy Filter”, **Industrial Technology, 2006. ICIT 2006. IEEE International Conference on Industrial Technology**, 2821-2826.

REDDY, K. S., & JAYA, T. (2021). “De-noising and enhancement of MRI medical images using Gaussian filter and histogram equalization”, **Materials Today: Proceedings**.

ROSENFELD, A. (1969). “Picture Processing by Computer”, **ACM Computing Surveys**, 147-176.

XIE, S., & TU, Z. (2017). “Holistically-Nested Edge Detection”, **Springer Science+Business Media**, 3-18.

SAMTAŞ, G., & GÜLESIN, M. (2011). “Sayısal Görüntü İşleme ve Farklı Alanlardaki Uygulamaları”, **Electronic Journal of Vocational Colleges**, 85-97.

SHARIFRAZI, D., ALIZADEHSANI, R., ROSHANZAMIR, M., JOLOUDARI, J. H., SHOEIBI, A., JAFARI, M., . . . ACHARYA, U. (2021). “Fusion of convolution neural network, support vector machine and Sobel filter for accurate detection of COVID-19 patients using X-ray images”, **Biomedical Signal Processing and Control**.

SHU, X., & WU, X.-J. (2010). “A Novel Contour Descriptor for 2D Shape Matching And Its Application to Image Retrieval”, **Image and Vision Computing**, 286-294.

SOLAK, S., & ALTINISIK, U. (2018). “Görüntü İşleme Teknikleri ve Kümeleme Yöntemleri Kullanılarak Fındık Meyvesinin Tespit ve Sınıflandırılması”, **Sakarya University Journal of Science**, 56-65.

TSUCHIMOTO, S., SHIBUSAWA, S., IWAMA, S., HAYASHI, M., OKUYAMA, K., MIZUGUCHI, N., . . . USHIBA, J. (2021). "Use of common average reference and large-Laplacian spatial-filters enhances EEG signal-to-noise ratios in intrinsic sensorimotor activity", **Journal of Neuroscience Methods**.

ÜNAL, H. (2021). "Comparative Analysis of Use States of Mobile Application Development Methods", **International Journal of Advances in Engineering and Management (IJAEM)**, 1118-1130.

WANG, X. (2007). "Laplacian Operator-Based Edge Detectors", **IEEE Transactions On Pattern Analysis And Machine Intelligence**, 886-890.

XIE, S., & TU, Z. (2015). Holistically-Nested Edge Detection. **Proceedings of IEEE International Conference on Computer Vision** (1395-1403).

YAMAN, K., SARUCAN, A., ATAK, M., & AKTÜRK, N. (2001). "Dinamik Çizelgeleme İçin Görüntü İşleme Ve Arıma Modelleri Yardımıyla Veri Hazırlama", **Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi**, 19-40.

YANG, Y., ZHAO, X., HUANG, M., WANG, X., & ZHU, Q. (2021). "Multispectral image based germination detection of potato by using supervised multiple threshold segmentation model and Canny edge detector", **Computers and Electronics in Agriculture**.

YUEMING, L., XIAOMEI, Y., ZHIHUA, W., CHEN, L., ZHI, L., & FENGSHUO, Y. (2019). "Aquaculture area extraction and vulnerability assessment in Sanduao based on richer convolutional features network model", **Journal of Oceanology and Limnology**, 1941-1954.

ZHANG, Z., MA, S., LIU, H., & GONG, Y. (2009). "An edge detection approach based on directional wavelet transform", **Computers and Mathematics with Applications 57**, 1265-1271.

ANSİKLOPEDİLER

ELEKTRONİK KAYNAKLAR

AUCKLAND, T. U., "Prewitt Filter Ders Notları", cs.auckland.ac.nz: https://www.cs.auckland.ac.nz/courses%0B/compsci373s1c/PatricesLectures/Prewitt_2up.pdf adresinden alındı, (Erişim Tarihi: 19.06.2021)

AYDIN, İ., “BMU-357 Sayısal Görüntü İşleme Ders Notları Bölüm 4”, web.firat.edu.tr: http://web.firat.edu.tr/iaydin/bmu357/bmu_357_bolum4.pdf adresinden alındı, (Erişim Tarihi: 19.06.2021a)

AYDIN, İ., “BMU-357 Sayısal Görüntü İşleme”, Fırat Üniversitesi, Bilgisayar Mühendisliği Bölümü: http://web.firat.edu.tr/iaydin/bmu357/bmu_357_bolum5.pdf adresinden alındı, (Erişim Tarihi: 19.06.2021b)

BAYRAM, B., “Akademik Veri Yönetim Sistemi”, Yıldız Teknik Üniversitesi: https://avesis.yildiz.edu.tr/bayram adresinden alındı, (Erişim Tarihi: 16.06.2021)

BENİMMUHENDİSİM., “Görüntü İşleme Temel Adımları”, benim mühendisim: https://www.benimuhendisim.com/goruntu-isleme/ adresinden alındı, (Erişim Tarihi: 17.06.2021)

BOURKE, P., “Image Filtering in the Frequency Domain”, http://paulbourke.net/: http://paulbourke.net/miscellaneous/imagefilter/ adresinden alındı, (Erişim Tarihi: 19.06.2021)

ÇAYIROĞLU, İ., “Görüntü İşleme Ders Notları 7. Hafta”, ibrahimcayiroglu: http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-7.Hafta.pdf adresinden alındı, (Erişim Tarihi: 19.06.2021c)

ÇAYIROĞLU, İ., “Görüntü İşleme Ders Notları 8. Hafta”, ibrahimcayitoglu.com: http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-8.Hafta.pdf adresinden alındı, (Erişim Tarihi: 20.06.2021d)

ÇAYIROĞLU, İ., “Görüntü İşleme Ders Notları 5. Hafta”, IbrahimCayiroglu.Com: http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-5.Hafta.pdf adresinden alındı, (Erişim Tarihi: 17.06.2021b)

ÇAYIROĞLU, İ., “Görüntü İşleme Ders Notları 1. Hafta”, IbrahimCayiroglu.Com: http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-1.Hafta.pdf adresinden alındı, (Erişim Tarihi: 17.06.2021a)

FANNTOOL., “YSA Nedir”, fanntool.blogspot.com: http://fanntool.blogspot.com/p/ysa-nedir.html adresinden alındı, (Erişim Tarihi: 21.06.2021)

GÜL, Ç., “Görüntünün Sınır Eğrisini Çıkaran Filtreler (Edge Detection)”, caglargul: http://caglargul.blogspot.com/2015/09/goruntunun-snr-egrisini-ckaran.html adresinden alındı, (Erişim Tarihi: 19.06.2021)

HIPR2., “Conservative Smoothing”, homepages.inf.ed.ac.uk: https://homepages.inf.ed.ac.uk/rbf/HIPR2/csmooth.htm adresinden alındı, (Erişim Tarihi: 19.06.2021a)

HIPR2., “Fourier Transform”, homepages.inf.ed.ac.uk: https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm adresinden alındı, (Erişim Tarihi: 19.06.2021b)

HIPR2., “Mean Filter”, HIPR2: https://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm adresinden alındı, (Erişim Tarihi: 18.06.2021c)

İLERİ, A., “Görüntü Filtrelerini Uygulama ve Kenarları Algılama”, medium: https://abdulsamet-ileri.medium.com/g%C3%B6r%C3%BCnt%C3%BC-filtrelerini-uygulama-ve-kenarlar%C4%B1-alg%C4%B1lama-21d42f194db4 adresinden alındı, (Erişim Tarihi: 19.06.2021)

KARAKOÇ, M., “Görüntü İşleme Teknolojiler Ve Uygulamaları”, Akademik Bilişim 1-3 Şubat 2012 Uşak Üniversitesi: https://ab.org.tr/ab12/sunum/21-goruntu_isleme-Karakoc.pdf adresinden alındı, (Erişim Tarihi: 17.06.2021)

KUCHLING, A. M., “Python Documentation”, Functional Programming HOWTO: https://docs.python.org/3/howto/functional.html adresinden alındı, (Erişim Tarihi: 16.06.2021)

LIANGYU, C., “rcf-edge-detection”, Github: https://github.com/mayorx/rcf-edge-detection adresinden alındı, (Erişim Tarihi: 26.07.2021)

LIBROW., “Mean filter, or average filter”, Librow: http://www.librow.com/articles/article-5 adresinden alındı, (Erişim Tarihi: 18.06.2021)

LIU, Y., “RCF”, github.com: https://github.com/yun-liu/RCF adresinden alındı, (Erişim Tarihi: 26.07.2021)

NI, Y., “OpenCV with Swift - step by step”, medium.com: https://medium.com/@yiweini/opencv-with-swift-step-by-step-c3cc1d1ee5f1 adresinden alındı, (Erişim Tarihi: 04.08.2021)

OPENCV., “OpenCV About”, OpenCV: https://opencv.org/about/ adresinden alındı, (Erişim Tarihi: 16.06.2021)

PİSKİN, M., “Görüntü İşleme Kitabı”, Mesut Piskin: https://mesutpiskin.com/blog/opencv-ile-goruntu-isleme-kitabi.html adresinden alındı, (Erişim Tarihi: 17.06.2021)

PLANTCV., “plantcv.readthedocs.io”, readthedocs.io: https://plantcv.readthedocs.io/en/v3.11.0/scharr_filter/ adresinden alındı, (Erişim Tarihi: 25.07.2021)

POLAT, E., “OpenCV’nin Android Studio Ortamında Kullanılması”, medium.com: https://medium.com/@enespolat/opencvnin-android-studio-ortam%C4%B1nda-kullan%C4%B1lmas%C4%B1-38f38dfd38e adresinden alındı, (Erişim Tarihi: 04.08.2021)

RHODY, H., “SIMG-782 Introduction to Digital Image Processing”, Rochester Institute of Technology: https://www.cis.rit.edu/class/simg782/ adresinden alındı, (Erişim Tarihi: 16.06.2021)

SACHAN, A., “Deep Learning based Edge Detection in OpenCV”, CV-Tricks.com: https://cv-tricks.com/opencv-dnn/edge-detection-hed/ adresinden alındı, (Erişim Tarihi: 22.05.2021)

SEKHON, M., “Image Filters in Python”, towardsdatascience: https://towardsdatascience.com/image-filters-in-python-26ee938e57d2 adresinden alındı, (Erişim Tarihi: 19.06.2021)

SONUGÜR, G., “2019_2020 Güz – Gİ Ders 8 Notları”, guraysonugur.aku.edu.tr: https://guraysonugur.aku.edu.tr/2019/11/20/2019_2020-guz-gi-ders-8-notlari/ adresinden alındı, (Erişim Tarihi: 19.06.2021)

TUTORIALSPOINT., “Prewitt Operator”, tutorialspoint: https://www.tutorialspoint.com/dip/prewitt_operator.htm adresinden alındı, (Erişim Tarihi: 19.06.2021)

UĞUR, A., “Görüntü İşlemeye Giriş Introduction to Image Processing”, Doç. Dr. Aybars UĞUR, DOCPLAYER: https://docplayer.biz.tr/1151263-Goruntu-islemeye-giris-introduction-to-image-processing-doc-dr-aybars-ugur.html adresinden alındı, (Erişim Tarihi: 18.06.2021)

UMUT, İ., "Çok Katmanlı Algılayıcı", ilhanumut.trakya.: http://ilhanumut.trakya.edu.tr/mlp.pdf adresinden alındı, (Erişim Tarihi: 21.06.2021)

TEZLER

ABAS, A. İ. (2011). "Çok Spektrallı Görüntü Füzyonu", (Yüksek Lisans Tezi), Bilgisayar Mühendisliği, Selçuk Üniversitesi.

AKTAŞ, A. (2020). "Derin Öğrenme Yöntemleri İle Dokunsal Parke Yüzeyi Tespiti", (Yüksek Lisans Tezi), Bilgisayar Mühendisliği, Marmara Üniversitesi.

ASLAN, M. F. (2018). "Opencl Ortamında Görüntü İyileştirme İşlemlerinin", (Yüksek Lisans Tezi), Elektrik Elektronik Mühendisliği, Süleyman Demirel Üniversitesi.

BAYKAN, N. (2010). "Robotik Bir Mikroskop Sisteminden Elde Edilen Görüntülerin Görüntü İşleme Ve Yapay Zeka Yöntemleri İle Analizi", (Doktora Tezi), Bilgisayar Mühendisliği, Selçuk Üniversitesi.

BOZTOPRAK, H. (2014). "Görüntü İşleme Teknikleri Ve Yapay Zeka Yöntemleri Kullanarak Atıksu Arıtmada Performans Analizlerinin İncelenmesi", (Doktora Tezi), Elektrik Elektronik Mühendisliği, Selçuk Üniversitesi.

ÇAKAR, E. (2018). "Akciğer Tomografisi Görüntülerinde Görüntü İşleme Teknikleri Kullanılarak Nodül Tespiti Ve Yapay Zeka İle Nodüllerin Nitelendirilmesi", (Yüksek Lisans Tezi) Biyomedikal Mühendisliği, Sakarya Üniversitesi.

ÇELİK, E. (2011). "Görüntü İşlemeye Dayalı Avuç İçi İzinin Yapay Sinir Ağı İle Tanınması", (Yüksek Lisans Tezi), Elektronik-Bilgisayar Eğitimi, Marmara Üniversitesi.

KARAKOÇ, M. (2011). "Görüntü İşleme Teknikleri Ve Yapay Zeka Yöntemleri Kullanarak Görüntü İçinde Görüntü Arama", (Yüksek Lisans Tezi), Bilgisayar Mühendisliği, Pamukkale Üniversitesi.

KASIM, Ö. (2015). "Kanserli lökosit hücrelerinin tespit ve sınıflandırılmasında dinamik bölütleme", (Doktora Tezi), Elektronik-Bilgisayar Eğitimi, Marmara Üniversitesi.

KURT, F. (2018). “Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi”, (Yüksek Lisans Tezi), Bilgisayar Mühendisliği, Hacettepe Üniversitesi.

MAHMOOD, A. (2014). “Effect Of Noıse On Edge Detectıon Technıques”, (Yüksek Lisans Tezi), Bilgisayar Mühendisliği, Çanyaka Üniversitesi.

MEYMANDI, T. M. (2018). “Ir Image Edge Detection Using Neural Network And Clustering”, (Yüksek Lisans Tezi), Bahçeşehir Üniversitesi.

OKTAY, A. B. (2011). “Önsel Bilgi Kullanılarak Tıbbi Görüntülerde Makine Öğrenmesi Tabanlı Kontur Bulma Ve Nesne Konumlandırma”, (Doktora Tezi), Bilgisayar Mühendisliği, Gebze Yüksek Teknoloji Enstitüsü.

SCHARR, H. (2000). “Optimal Operators In Digital Image Processing”, (Doktora Tezi), Combined Faculties for the Natural Sciences and for Mathematics, Rupertus Carola University of Heidelberg, Germany.

TAŞÇI, A. E. (2013). “Akciğer Tomografileri Kullanılarak Yapay Zeka Ve Görüntü İşleme Tekniklerine Dayalı Otomatik Nodül Bölge Tespit Yöntemi Geliştirilmesi”, (Yüksek Lisans Tezi), Bilgisayar Mühendisliği, Ege Üniversitesi.

YÜKSEL, H. E. (2010). “Halkasal Eksantrik Borularda Akış Özelliklerinin Dijital Görüntü İşleme Ve Yapay Zeka Teknikleri Kullanarak Tespiti”, (Yüksek Lisans Tezi), Tobb Ekonomi Ve Teknoloji Üniversitesi.

DİĞER KAYNAKLAR

DUMAN, B. (2019). “Görüntü İşleme Tekniklerinin Eklemeli İmalatta Kullanımı”, 4th International Congress on 3D Printing (Additive Manufacturing) Technologies And Digital Industry (s. 11-14), Antalya: International Congress on 3D Printing (Additive Manufacturing) Technologies And Digital Industry.

IROD, B. (2007). “Digital Image Processing”, Ders Notları, Kaliforniya: Stanford Üniversitesi Elektrik Mühendisliği Bölümü.

KARHAN, M., OKTAY, M. O., KARHAN, Z., & DEMİR, H. (2011). “Morfolojik Görüntü İşleme Yöntemleri ile Kayısılarda Yaprak Delen (Çil) Hastalığı Sonucu Oluşan Lekelerin Tespiti”, 6th International Advanced Technologies Symposium (IATS’11) (s. 172-176), Elazığ: Fırat Üniversitesi.

KESKİN, M., DOĞRU, A. Ö., GÖKSEL, Ç., & BALÇIK, F. B. (2014). "Çok Spektrumlu Verilerden Bilgi Çıkarımında Mekansal Filtreleme Etkisinin İncelenmesi", 5. Uzaktan Algılama-CBS Sempozyumu (UZAL-CBS 2014), İstanbul: UZAL-CBS 2014).

LEI, P. (2004). "Adaptive Median Filtering", Machine Vision 140.429 Digital Image Processing.

MA, Z., TAVARES, J. M., & JORGE, R. M. (2009). "A Review on the Current Segmentation Algorithms for Medical Images", IMAGAPP 2009 - Proceedings of the First International Conference on Computer Imaging Theory and Applications, (s. 135-140), Lisboa, Portugal.

MORSE, B. S. (2000). "Digital Image Processing", Ders Notları, Utah: Brigham Young Üniversitesi Bilgisayar Mühendisliği Bölümü.

ONAT, E. (2017). "Sobel, Robert, Prewitt ve Laplace Filtreleri Kullanarak Gerçek Zamanlı Video Sinyallerinin øúlenmesinin FPGA'da Gerçeklenmesi", 2017 25th Signal Processing and Communications Applications Conference (SIU) (s. 1-4), Antalya: IEEE.

PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., . . . CHINTALA, S. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library", 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Canada: arXiv.

ROSSUM, G. v. (1999). "Computer Programming for Everybody (Revised Proposal): A Scouting Expedition for the Programmers of Tomorrow", Technical Report, Corporation for National Research Initiatives.

SUGIMOTO, S. (2004). U. S. Patent No. 6,750,890. Washington, DC: U.S. Patent and Trademark Office.

ŞEVİK, U., KÖSE, C., & GENÇALİOĞLU, O. (2007). "Retina Görüntülerinde Yaşa Bağlı Makula Dejenerasyonunun Bölge Büyütme Yöntemiyle Segmentasyonu", EMO 12. Ulusal kongresi (s. 635-638), EMO 12. Ulusal kongresi.

TEKDEMİR, İ. G., ARSOY, A. B., ALBOYACI, P. D., & KESKİN, A. G. (2013). "Uyarlamalı Medyan Filtresi ve Birikimli Toplam Yöntemleri ile Bir Güç

Sisteminde Arıza Tespitinin İncelenmesi", 2013 Enerji Verimliliği ve Kalitesi Sempozyumu, Kocaeli: 2013 Enerji Verimliliği ve Kalitesi Sempozyumu.

UZUNHİSARCIKLI, E., GÖREKE, V., & GÜVEN, A. (2014). "Gri Seviyeli Eş-oluşum Matrisleri Kullanılarak Sayısal Mamogram Görüntüsünden Doku Özniteliklerinin Çıkarılması ve Yapay Sinir Ağı ile Kitle Tespiti", Tıptekno 14 Tıp Teknolojileri Ulusal Kongresi (s. 95-98), Nevşehir: Tıptekno 14 Tıp Teknolojileri Ulusal Kongresi.

ZHANG, C., MURAI, S., & BALTSAVIAS, E. (1999). "Road Network Detection by Mathematical Morphology", 3D Geospatial Data Production: Meeting (s. 185-200), Switzerland: Institute of Geodesy and Photogrammetry.

ÖZGEÇMİŞ

Ad-Soyad **:** Hakan ÜNAL

ÖĞRENİM DURUMU:

- **Lisans** **:** 2017, Süleyman Demirel Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü

DİĞER YAYINLAR:

- ÜNAL H., 2021. “Comparative Analysis of Use States of Mobile Application Development Methods”, International Journal of Advances in Engineering and Management (IJAEM), Volume 3, Issue 5 May 2021, pp: 1118-1130, ISSN: 2395-5252