



ONDOKUZ MAYIS ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

RESİM DÜZENLEYİCİ MOBİL ANDROID UYGULAMA

PROJE RAPORU

MURAT DEMİR

Danışman

Dr. Öğretim Üyesi Sercan DEMİRCİ

MAYIS, 2020

ÖZET

Dijital dünyada her bir resim karesi farklı renklere sahip küçük küçük noktaların bir araya gelmesiyle oluşur. Yani resimlerin yapıtaşları piksel adını verdığımız bu noktalardır. Eğer ekran çözünürlüğünü biraz azaltıp ekrana yaklaşırsanız pikselleri rahatlıkla gözlemlayabilirsiniz.

Bahsedilen pikselleri değerler olarak düşünürsek resmi de bu değerlerden oluşan matris olarak düşünebiliriz. Bu matrislerin her bir değeri resimdeki her bir pikselin konumunu ve rengini tutmaktadır. Eğer bu matrislerin değerlerini değiştirirsek haliyle resimdeki piksellerin renk ve konum bilgileri değişecek ve resmimiz bu değişimin kapsamı oranında farklı bir görünüme sahip olacaktır. Matrisler üzerinde yapılan bu işlemlere literatürde Görüntü İşleme (Image Processing) ismi verilmektedir

Bu kapsamda kullanıcının seçmiş olduğu bir resmin üzerindeki matrlslere görüntü işleme (image processing) yöntemleri uygulanılarak resim istenilen oranda özelleştirilmektedir. Bu görüntü işleme yöntemleri RESİM DÜZENLEYİCİ MOBİL ANDROİD UYGULAMA PROJESİ 'inde kullanılarak kullanıcı hiç bir matris işlemleriyle uğraşmaya gerek duymadan uygulama üzerindeki butonlar yardımıyla resme otomatik olarak matris işlemlerinin uygulanması anlatılmıştır.

RESİM DÜZENLEYİCİ MOBİL ANDROİD UYGULAMA PROJESİ ile müşteri kendi android cihazında bulunan resimlere veya o anda uygulama yardım ile çekeceği yeni fotoğrafa çeşitli görüntü işleme algoritmalarının yardımıyla müşterinin tamamen kendisinin özelleştirebileceği şekilde resmi istediği hale getirebilmesini sağlayacak bir uygulamadır.

İÇİNDEKİLER

I GİRİŞ

1 AMAÇ	2
2 LİTERATÜR ÖZETİ	3

II MATERİYAL

2.1 KIRPMA	5
2.2 PAYLAŞ	6
2.3 EFEKTLER	7
2.3.1 Sepya	7
2.3.2 Televizyon	8
2.3.3 Eskiz	9
2.3.4 Neon	10
2.3.5 Lomo	11
2.3.6 Gri Tonlama	12
2.3.7 Negatif	13
2.4 FİLTRELER	14
2.4.1 Gauss Bulanıklığı	14
2.4.2 Kabartma	15
2.4.3 Gravür	16
2.4.4 Keskinlik	17

2.4.5 LR Motion	18
2.4.6 Ortalama	19
2.4.7 Pürüzsüz	20
2.4.8 Parıltı	21
2.4.9 Roman Kabartma	22
2.5 ÇEVİR	23
2.5.1 Yatay	23
2.5.2 Dikey	24
2.6 Parlaklık	25
2.6.1 Parlaklık Artırma	25
2.7 Doyma	26
2.7.1 Doyma Artırma	26
2.8 Gama	27
2.8.1 Gama Artırma	27
2.9 Kontrast	28
2.9.1 Kontrast Artırma	28
2.10 Renk Açıkllaştır	29
2.10.1 Mavi	29
2.10.2 Yeşil	30
2.11 Uygula	31
2.11.1 Histogram	31
2.11.2 PseudeHDR	32
III YÖNTEM	
3 YÖNTEM	34

3.1 Image Processesing	34
3.2 Android Studio	35
3.3 Java	36

IV SONUÇ

4 SONUÇ	38
---------	----

KAYNAKÇA	39
----------	----

5 EKLER	40
---------	----

5.1 ÖZGEÇMİŞ	40
------------------------	----

5.2 KODLANMASI	41
--------------------------	----

5.2.1 EFEKTLER	41
--------------------------	----

5.2.2 FİLTRELER	48
---------------------------	----

5.2.3 ÇEVİR	58
-----------------------	----

5.2.4 Parlaklık	60
---------------------------	----

5.2.5 Doyma	61
-----------------------	----

5.2.6 Gama	62
----------------------	----

5.2.7 Kontrast	63
--------------------------	----

5.2.8 Renk Açıklasır	64
--------------------------------	----

5.2.9 Uygula	66
------------------------	----

ŞEKİLLER LİSTESİ

1	Caption	5
2	Caption	6
3	Sepya Efekti	7
4	Televizyon Efekti	8
5	Eskiz Efekti	9
6	Neon Efekti	10
7	Lomo Efekti	11
8	Gri Tonlama Efekti	12
9	Negatif Efekti	13
10	Gauss Bulanıklığı Filtresi	14
11	Kabartma Filtresi	15
12	Gravür Filtresi	16
13	Keskinlik Filtresi	17
14	Lr Motion Filtresi	18
15	Ortalama Filtresi	19
16	Pürüzsüz Filtresi	20
17	Parıltı Filtresi	21
18	Roman Kabartma Filtresi	22
19	Yatay Çevirme	23

20	Dikey Çevirme	24
21	Parlaklılık Artırma	25
22	Doyma Artırma	26
23	Gama Artırma	27
24	Kontrast Artırma	28
25	Mavi Artırma	29
26	Yeşil Artırma	30
27	Histogram	31
28	PseudeHDR	32
29	Sepya Efekti	41
30	Televizyon Efekti	42
31	Eskiz Efekti	43
32	Neon Efekti	44
33	Lomo Efekti	45
34	Gri Tonlama Efekti	46
35	Negatif Efekti	47
36	Gauss Bulanıklığı Filtresi	48
37	Kabartma Filtresi	49
38	Roman Kabartma Filtresi	50
39	Gravür Filtresi	51
40	Keskinlik Filtresi	52
41	Lr Motion Filtresi	53
42	Ortalama Filtresi	54
43	Pürüzsüz Filtresi	55

44	Parıltı Filtresi	56
45	Roman Kabartma Filtresi	57
46	Yatay Çevirme	58
47	Dikey Çevirme	59
48	Parlaklık Artırma	60
49	Doyma Artırma	61
50	Gama Artırma	62
51	Kontrast Artırma	63
52	Mavi Artırma	64
53	Yeşil Artırma	65
54	Histogram	66

BÖLÜM: I

GİRİŞ

1

A M A Ç

RESİM DÜZENLEYİCİ MOBİL ANDROİD PROJESİ'nin Amacı, Android cihazları tarafından çekilen veya var olan fotoğrafların Kullanıcının istediği yönde, özel algoritmalar yardımıyla göze en güzel görünecek hale getirilmesidir.

Akıllı telefonların en çok kullanılan ve en önemli özelliği fotoğraf çekmektir . Öyle ki, artık akıllı telefon üreticileri, yeni modeller çıkardıklarında en çok vurguladıkları konu, en yeni ürünlerinin kamera özellikleri oluyor. Kullanıcı açısından baktığımızda ise, akıllı telefonların kamera özelliklerinin, her geçen gün gelişmesinin etkisiyle, fotoğraf makinası kullanımının giderek azaldığını ve mobil fotoğrafçılık dediğimiz olsunun yükselen bir trend haline gelmekte olduğunu görüyoruz.

Bunlara paralel olarak, fotoğrafçılık eğitimi veren merkezlerde, mobil fotoğrafçılık derslerinin de gündeme alındığı görülüyor. Mobil fotoğrafçılık ayrı bir uzmanlık olmakla birlikte, başlangıç olarak, akıllı telefonlarıyla fotoğraf çekenlerin yararlanabileceği Resim Düzenleyici uygulamasının en temel amacı müşterilere en güzel görünümde resimler sunabilmektir.

Gelişen Teknolojiyle birlikte insanların göze hoş gelicek fotoğrafların bulunma ihtiyacı artmış durumdadır. Bu ihtiyacı gidermek amacıyla geliştirilen bu uygulama projesi gelişime son derece açık ve şeffaf bir yapıya sahiptir. Çeşitli Argeler ve uygulama üzerindeki dinamizmin verdiği güç ile sürekli büyüebilen bir proje olabilecektir.

2

LİTERATÜR ÖZETİ

Bu Kısmnda, görüntü işleme yöntemlerini profesyonel manada günümüze taşıyan ilk uygulama olan PHOTOSHOP'un nasıl ortaya çıktığı anlatılmaktadır.

Photoshop tarihi ilk olarak 1987 yılında o dönemde Amerika Birleşik Devletleri, Michigan Üniversitesi'nde doktora öğrencisi olan Thomas Knoll ve kardeşi John Knoll ile başlar. O dönemde yazılımı birlikte geliştirmeye başlayan abi kardeşin yolları 1988 yıllarda Adobe ile kesişir. Kardeşler yazılımın tüm haklarını Adobe'a satarlar. 1990 yılına gelindiğinde ise yazılımın 1.0 sürümü ilk olarak Mac bilgisayarlar için satışa sunulur. Geliştirildiği ilk yıllarda ismi kısa bir süre ImagePro olduktan daha sonra halen günümüzde kullanılan Photoshop adını alır.

Photoshop 1992 yılına gelinceye kadar Mac sürümü ile piyasada bulunur. Bu tarihten itibaren ise hem Mac hem PC sürümü satışa sunulur. Fakat özellikle Ülkemizde baskı endüstrisi uzun yıllar Mac hakimiyetinde olduğu için neredeyse 2000'li yılların başlarına kadar ağırlıklı olarak Photoshop Mac bilgisayarlarda kullanılır. Windows'lu bilgisayarların yaygınlaşması ve artmasıyla beraber PC sürümü de yaygın olarak kullanılmaya başlandı. Günümüzde özellikle bilgisayarlar üzerinde halen kullanılmaya devam etmektedir.

BÖLÜM: II

MATERIAL

2.1 KIRPMA

Varsayalım ki üç günlük bir yürüyüş gezisinden döndünüz. Medeniyete dönüş yapar yapmaz tüm dijital fotoğraflarınızı bilgisayarınıza attınız; bazı fotoğrafların etiketlenmesi gerekebilir, bazlarının da rötuşa ihtiyacı olabilir ama biraz kırpma işlemi ile hepsi çok daha güzel görünecektir.

Fotoğraflarınızı kolayca kırpın! Çektığınız fotoğraflar hatırı sayılır bir sayıya ulaştığında, kırpma aracımızla fotoğraflarınızı sonraki projenizde mükemmel görünecek biçimde dönüştürebilir ve doğru boyuta sahip olacak şekilde kırpabilirsiniz. Fotoğrafları macera dolu bir fotoğraf kolajı oluşturacak şekilde bir araya getirirken veya bir davetiye ya da broşüre eklerken, görsel kırpma aracımız tasarımlarınızın net, düzenli ve etkileyici görünmesini sağlar. Üstelik bunu doğrudan telefon veya tabletinizden de yapabilirsiniz!

Kırpma işlemi için Java kodlarıyla resmin matris boyutları kullanıcının seçtiği oranda kesilmiştir.



Şekil 1: Caption

2.2 PAYLAŞ

Resim Düzeneyici Uygulaması ile fotoğrafınızı çeker çekmez veya düzenledikten sonra uygulama üzerinden anında arkadaşlarınız ile yardımcı bir uygulama (örneğin: Gmail, Whatsap ...) ile paylaşabilirsiniz.

Paylaş işlemi için java kodlarıyla implicit intent kullanılarak fotoğrafın paylaşılması sağlanmıştır.



Şekil 2: Caption

2.3 EFEKTLER

Fotoğraf çekerken yenilikler peşindeyseniz, gerçeği manipüle ederek elde edebileceğiniz farklı farklı efektler ile ilginç sonuçlar elde edebilirsiniz. Bunun için çekimden önce veya çekim sonrasında dijital olarak uygulayabileceğiniz çeşitli efektler bulunmaktadır. Bu bölümde çekilmiş fotoğraftaki ışık kaynağıyla oynayarak konunuzu farklı yazılımlar kullanarak basit bir görüntüyü nasıl dramatik bir fotoğrafa çevirebileceğinizin yollarını bulacaksınız.

EFEKTLER kısmında uygulamada resmi çektiğinden sonra veya var olan resmi seçtiğinde Efektler butonuna dokunmasıyla açılan Sepya, Televizyon, Eskiz, Neon, Lomo, Gri Tonlama ve Negatif efektleri gösterilmektedir.

2.3.1 Sepya

Fotoğrafın sararmış görüntüsünü elde etmek için uygulanan işlemidir eskiden fotoğrafçılar bu işlemi karanlık odada baskından sonra hazırlamış oldukları nescafe veya kahve sularının içinde bir müddet bekletirlerdi ve yapılan sepia tonlaması farklılık gösterebilirdi.

Şuan dijital ortamda yapılan sepia'nın tonlaması, efekti yapan makineye (fotoğraf makinesi, çekim aşaması, fotoğrafın tonu) ihtiyaç duyulmadan uygulamamızdaki bu buton ile bu efekti resminize uygulayabilirsiniz.

Sepya efektiyle resim üzerindeki kırmızı, mavi, yeşil renklerin ton doygunluğunun değiştirilmesiyle resme bu efekt kazandırılmıştır.



Şekil 3: Sepya Efekti

2.3.2 *Televizyon*

Televizyon efektiyle resminizin eski tip televizyonların ekranındaymışcasına görünmesini sağlayabilir ve böylece kendinizi geçmişe götürerek maziyi bu güne taşıyabilirsiniz..

Bu Efekt ile Resim üzerindeki matrislerdeki belirli piksellere beyaz, mavi gibi renklerin uygulanmasıyla orjinal resmin sabit kalması sağlanılarak sanki bir TV ekranındaymışsına bir görünüm elde edilmiştir.

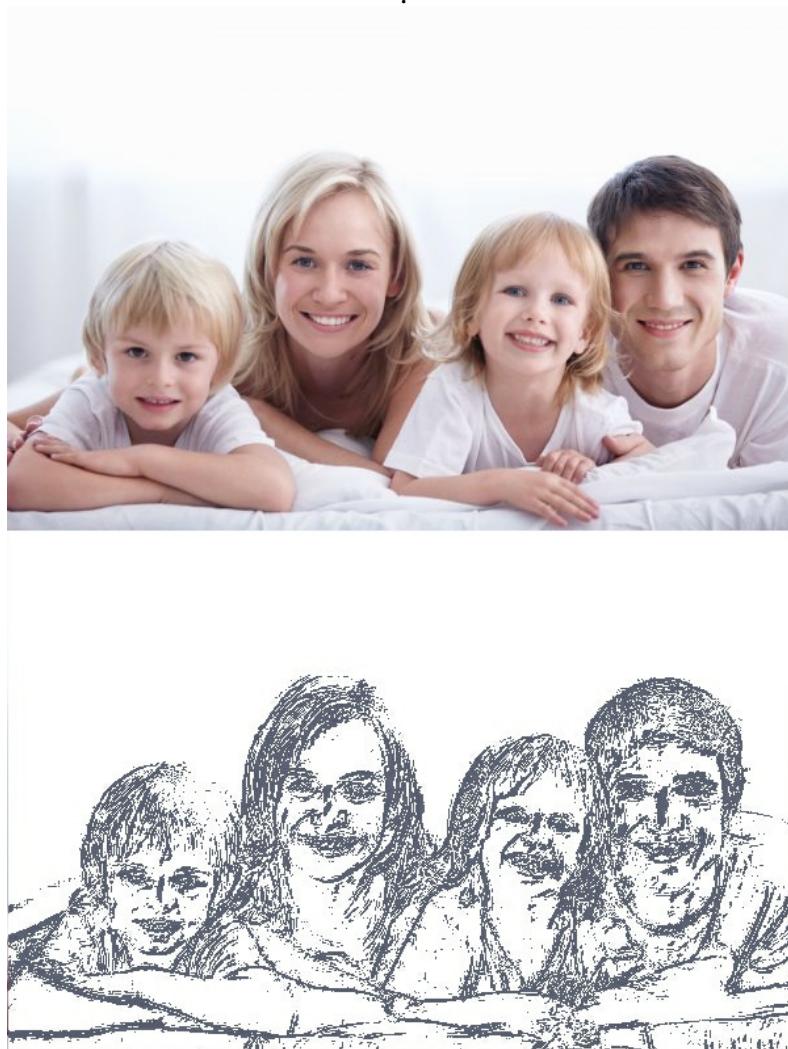


Şekil 4: Televizyon Efekti

2.3.3 *Eskiz*

Eskiz ile resminizin sanki profesyonel ressamların elinden çıkışmış bir kara kalem çalışmasındaki gibi görünmesini sağlayabilirsiniz. Resimleriniz bu sayede daha farklı bir ortama sahip olarak resminize farklılık katacaktır.

Eskiz Efekti ile resim önce 3 boyutlu matristen 2 boyuta indirgenerek üzerinde görünen nesneler belirgin ton farkları yardımıyla seçiliip kırpılmasıyla ve renklerin siyah, beyaz olacak şekilde yeniden düzenlenmesiyle elde edilmiştir.



Şekil 5: Eskiz Efekti

2.3.4 Neon

Resminize Bu Neon Efektini uygulamak size daha çok eğlence katabilir. Resminizdeki nesnelerin bulunduğu kısımları yeşil renk ile çizen bu efekt özel algoritmalar kullanılarak tasarlanmıştır.

Neon efekti ile resim 3 boyutlu matristen 2 boyutlu matrise indirgenerek siyah beyaz görünüm elde edilmiştir. Kalan kısımda siyah tonlarının bulunduğu bölgeler belirli yüzdeliklerle indirgenerek nesne değişimleri saptanmıştır. Bunlarla beraber bu neslerin sadece dış katmanına yeşil renk uygulanarak Neon efekti elde edilmiştir.



Şekil 6: Neon Efekti

2.3.5 *Lomo*

Doksanlı yılların başında iki tane Viyanalı gencin Rusya da keşfettikleri Loma Kompakt makinesinin üretim ve dağıtım sürecine dahil olmalarıyla beraber Lomo efekti ortaya çıkmıştır. Bu efekt ile birlikte birinci şahıs açısından bakarcasına resminize birinci şahıs görünümü katabilirsiniz. Sanki bir bölmeden görüntülenmesini sağlayarak ayrı bir boyuta çıkaracak bu efekt ile arkadaşlarınızla birlikte eğlenebilirsiniz.

Lomo efekti resmin köşelerinden başlayarak belirli bir ovallıkte siyah rengin gittikçe resmin kendi rengine dönecek şekilde programlanarak hazırlanmıştır

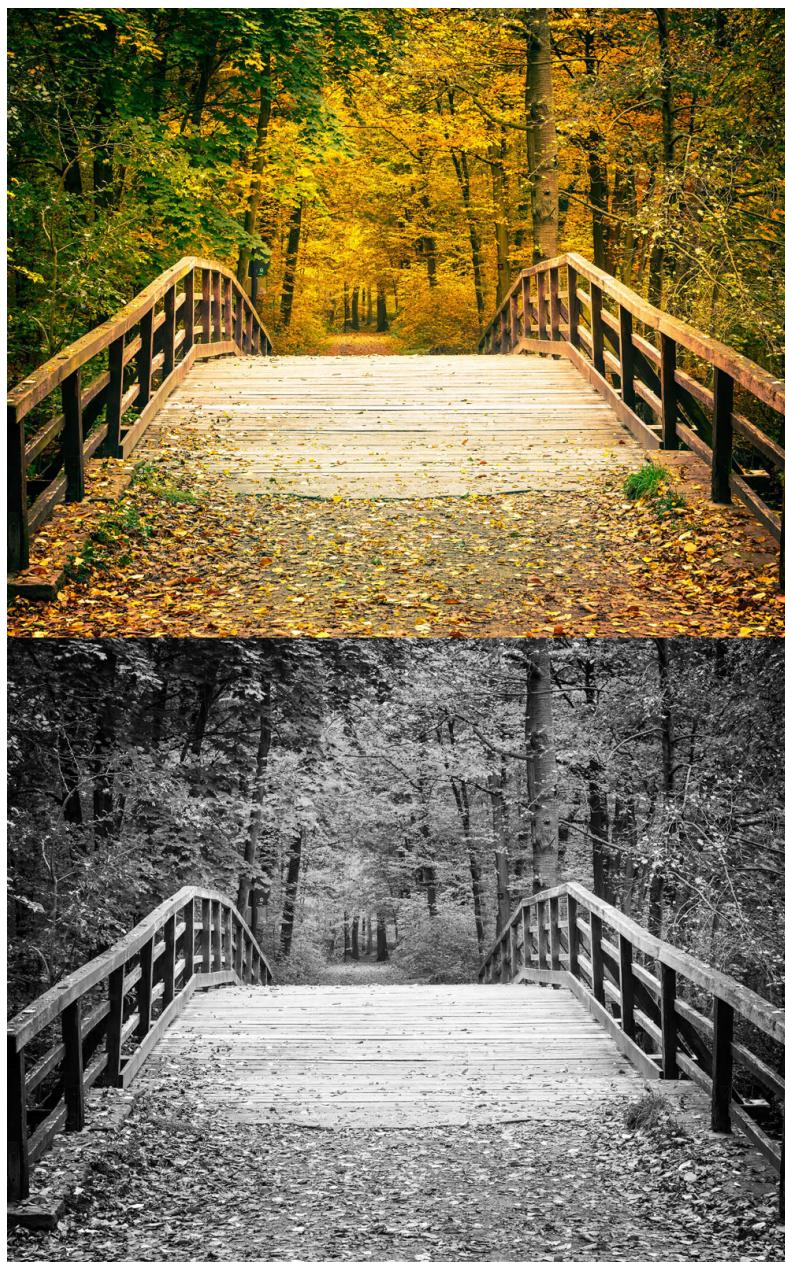


Şekil 7: Lomo Efekti

2.3.6 *Gri Tonlama*

Resminizin siyah beyaz görünümünde olmasını sağlayabilirsiniz ve böylece resminize klasik bir görünüm katıp, sizi geçmişe yolculuk ettirecektir.

Gri tonlamada Resim 3 boyutlu matristen 2 boyutlu matris indirgenmiştir. Böylece resimden renk bilgisi çıkarılmış klasik görünüm korunmuştur. Ayrıca siyah ve beyaz tonların gri yüzdesi artırılmıştır.



Şekil 8: Gri Tonlama Efekti

2.3.7 Negatif

Grafik tasarım açısından negatif alan veya diğer adıyla beyaz alan, bir görsel çalışma içindeki nesnenin etrafındaki alanı tanımlamak içim kullanılıyor. Bu alan tasarımin başarısı açısından en az nesnenin kendisi kadar önem taşır. Pozitif alanın sınırlarını belirleyen ve görsel kompozisyon da denge getiren unsur, negatif alandır. Ve bu alan ile odak noktası negatif alana çekilmektedir

Negatif efekti ile resimdeki renklerin zıt renkleri oluşturacak şekilde programlanıp resimdeki 3 boyutlu matris yeniden düzenlenmiştir.



Şekil 9: Negatif Efekti

2.4 *FİLTRELER*

Android cihazınızın kamerasıyla oynamadan veya önüne film koyulmasına hiç bir gereksinim duyulmadan yazılımsal olarak gerçek hayatı çok uğraş verilerek yapılacak bir çok işlemi bir butona dokunarak anında en güzel resim görüntüleri elde edebilirsiniz.

FİLTRELER kısmında uygulamada resmi çektiğinden sonra veya var olan resmi seçtiğinizde Filtreler butonuna dokunmasıyla açılan Gauss Bulanıklığı, Kabartma, roman Kabartma, Gravür, Keskinlik, LR Motion, Ortalama, Pürüzsüz, Parıltı, Roman Kabartma Filtreleri gösterilmektedir.

2.4.1 *Gauss Bulanıklığı*

Gauss Bulanıklığı butonuna basıldığında resminiz özel gauss algoritması kullanılarak belirli bölgelerin bulanık görünerek resminizin dikkat çekici özelliği artırılır.

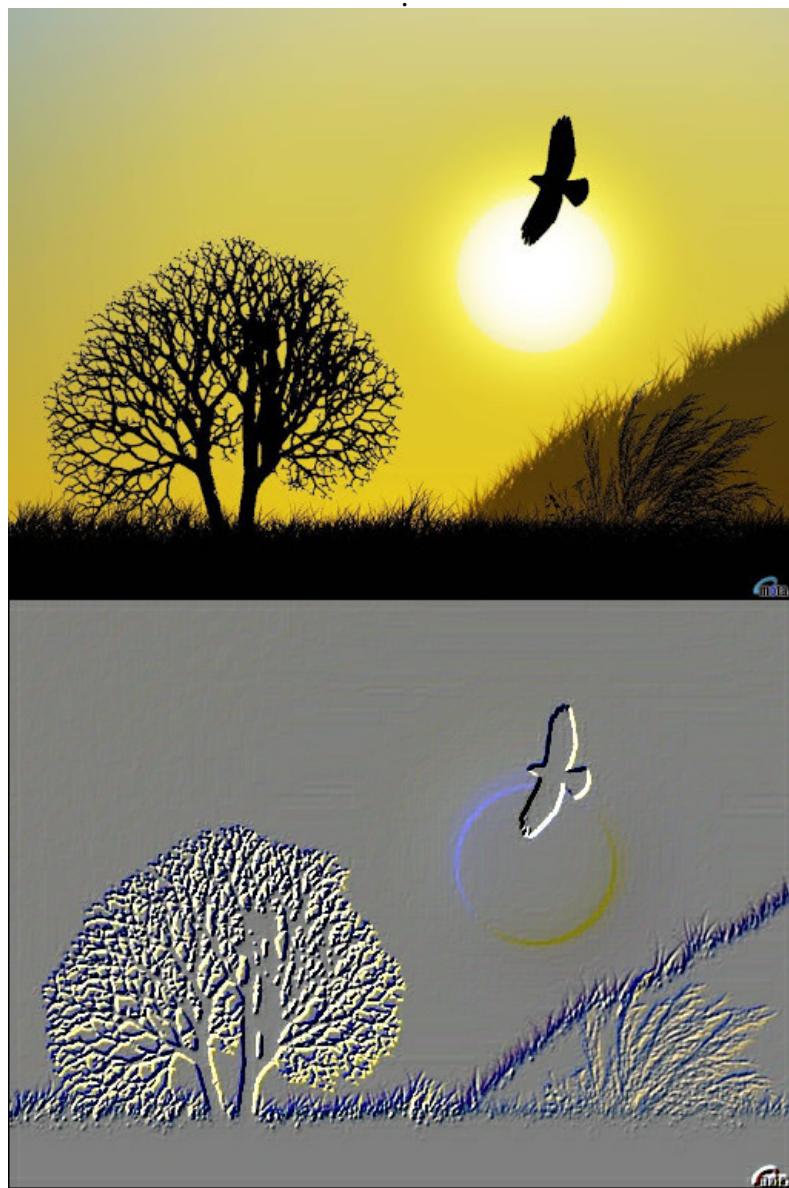
Gauss Bulanıklığı filtresi ile resmin 3. boyutundaki renklerin matris değerlerinin döngülerle diğer piksellere dağıtılması sağlanmıştır.



2.4.2 *Kabartma*

Kabartma Filtresi ile fotoğrafınız bir kabartma ustasının elinden çıkışmış gibi görünüp gayet havalı bir boyuta ulaşacaktır.

Kabartma滤镜 ile resim 3 boyuttan 2 boyuta indirgenip nesneler ton farklılıklarını yardımıyla seçilmiştir. Tekrar 3 boyuta çıkarılıp seçilen bu nesnelerin çevresel piksel değerlerine renk tonlanması yapılmıştır.



Şekil 11: Kabartma Filtresi

2.4.3 *Graviür*

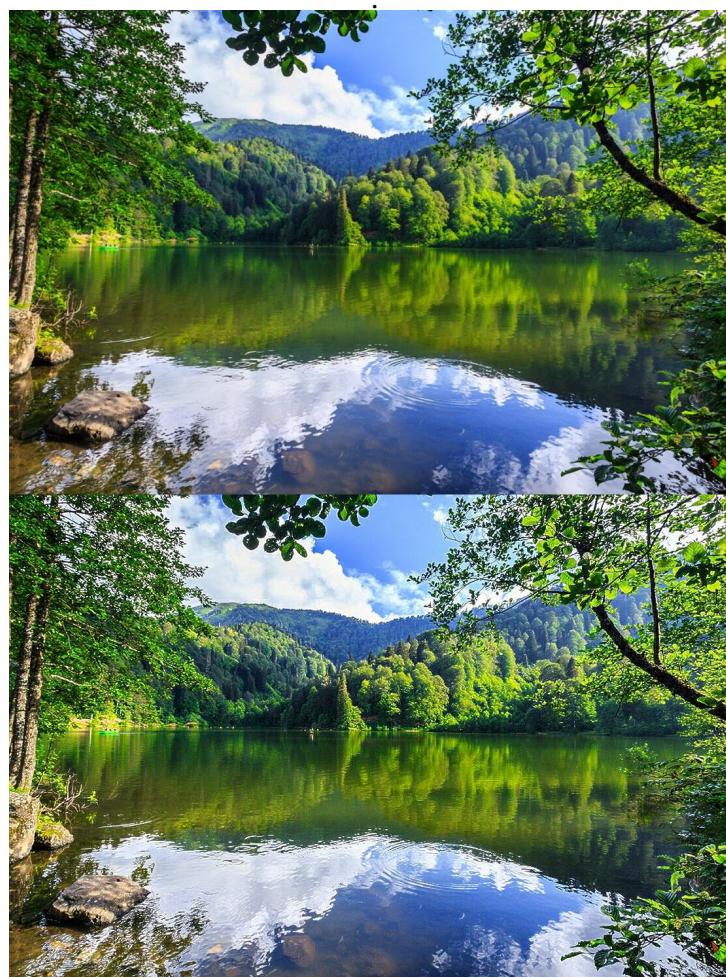
Gravür sanat türlerinden biri olmakta ve geçmişi çok eski dönemlere dayanmaktadır. Metal, taş, tahta üzerine oyma tekniğini kullanarak yapılan ve kâğıda aktarılarak sanat, yaygın bir şekilde yapılmaktadır. En İnce detaylar kullanılarak ve titizlik isteyen gravür, matbaacılık alanında da kullanılmaktadır. Muşamba gibi çeşitli malzemeler üzerine de uygulanmaktadır. Uygulamamız Resminizin Gravür Sanatı ile yapılmış gibi görünmesini sağlayabilir. Gravür滤resi kabartma滤resine ek olarak resim 3. boyuta çıkarılmadan önce matrislerin piksel değişimlerinin belirli değerlere atanmasıyla 3. boyutta renklere bu değişimlerin eklenmesiyle elde edilmiştir.



Şekil 12: Gravür Filtresi

2.4.4 Keskinlik

Bir fotoğraf çekerken dikkat edilmesi gereken konuların başında keskinlik gelir. Fotoğrafın keskinliği ise bazı çevresel şartlardan kullanılan donanıma kadar birçok etkene bağlıdır. Fotoğrafçının kontrol ettiği konular içinde en önemlilerinden biri de keskinliktir. Fotoğrafta net olarak görülecek detay olarak tanımlanabilecek keskinliğin birçok farklı ifade şekli vardır. Keskinlik ile fotoğrafçı dikkat çekmek istediği odak noktasını öne çıkarabilir. Keskinlik filtresi resimdeki nesnelerin ton farklılıklarıyla değişimleri saptanıp bu değişimlerin gerçekleştiği bölgelerdeki renklerin beyaz dengesi ve renk tonunun aynı oranda artırılmasıyla elde edilmiştir.

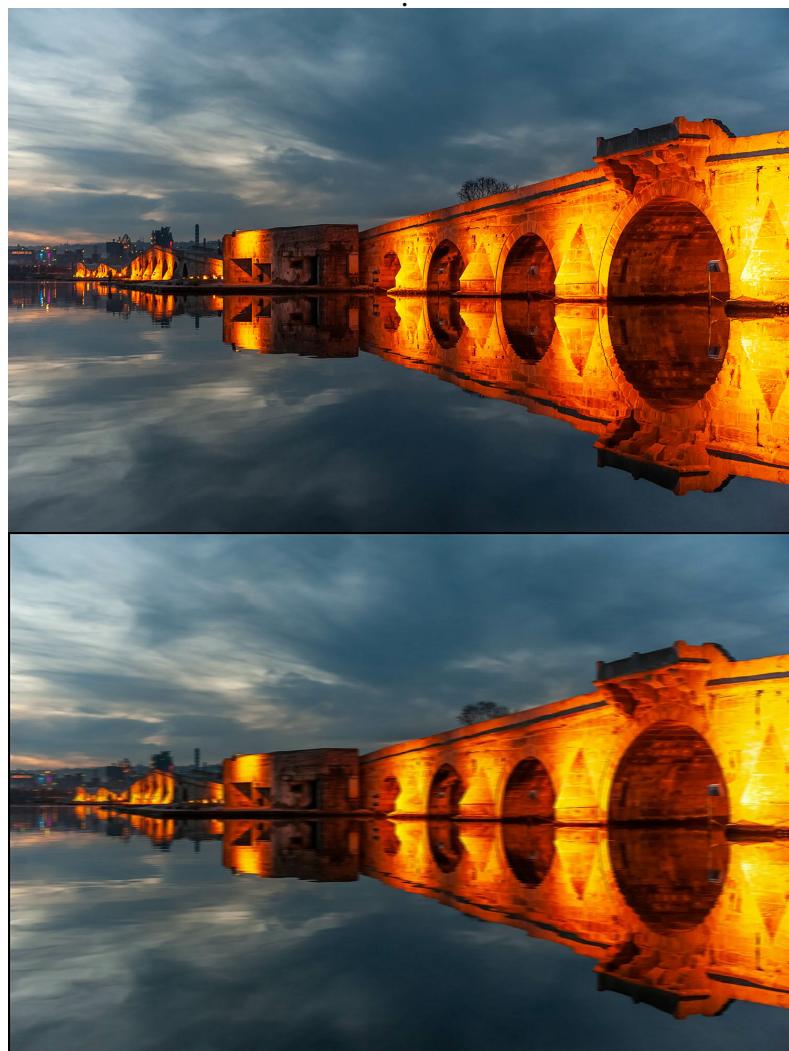


Şekil 13: Keskinlik Filtresi

2.4.5 *LR Motion*

Türkçesi Sol Sağ Sahne olan bu filtre resminizin en net görünümü sahip olmasını sağlar. Özel algoritması ile resmin sol kısmına ve sağ kısmına uygulanan filtre sahneye odaklanma kolaylığı sağlayacaktır.

LR Motion filtresi resimdeki matrislerin 3.boyutuyla solundaki ve sağındaki değerlerin belirli oranda değiştirilerek elde edilmiştir.



Şekil 14: Lr Motion Filtresi

2.4.6 *Ortalama*

Ortalamafiltresi resminizdeki renk yoğunluğunun tüm resme dağılmmasını sağlayabilir.

Ortlamafiltresi resmin 3. boyutundaki renk bilgisindeki yüksek olan değerlerinin tüm matriste artırılmasıyla elde edilmiştir.



Şekil 15: Ortalama Filtresi

2.4.7 *Pürüzsüz*

Pürüzsüz filtresi ile özellikle cildinizdeki istenmeyen görüntülerden kurtulmanızı sağlamaktadır. Pürüzsüz bir görünüm sahip olmanızı sağlayabilir.

Pürüzsüz filtresi resimdeki birbirinden zıt değerlerdeki renk bilgilerinin ve beyaz dengesinin birbirlerine yakınsamaları sağlanarak elde edilmiştir.



Şekil 16: Pürüzsüz Filtresi

2.4.8 *Parıltı*

Parıltı butonu ile resminize uygulama tarafından otomatik belirlenmiş bir parlama efekti ekleyebilirsiniz. Karanlık resimleriniz parıltı efekti ile aydınlanacaktır..

Parıltıfiltresi resminizin genel ve özellikle siyah tonlarının arttığı bölgelerdeki piksellerin beyaz oranının artırılmasıyla elde edilmiştir.

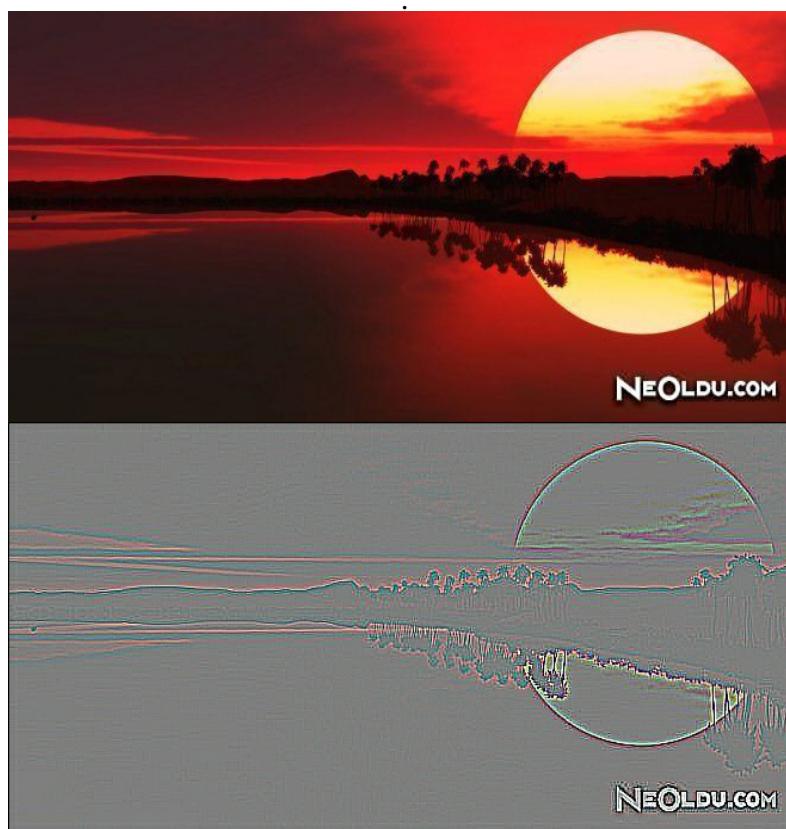


Şekil 17: Parıltı Filtresi

2.4.9 *Roman Kabartma*

Roman kabartma resminize evrensel Roman kabartma modunda görüntülenmesini sağlayabilir ve böylece resminizi profesyonel elliinden çıkışmış bir kabartma görünümüne sahip olur.

Resim 2 boyuta indirgenerek ton farklılıklarını yardımıyla resimdeki nesneler seçilmiş ve seçilen bu nesnelerin kenarlarına resim 3. boyuta çıkarıldığında örijinal resimdeki renk tonlarının tekrar uygulanmasıyla elde edilmiştir.



Şekil 18: Roman Kabartma Filtresi

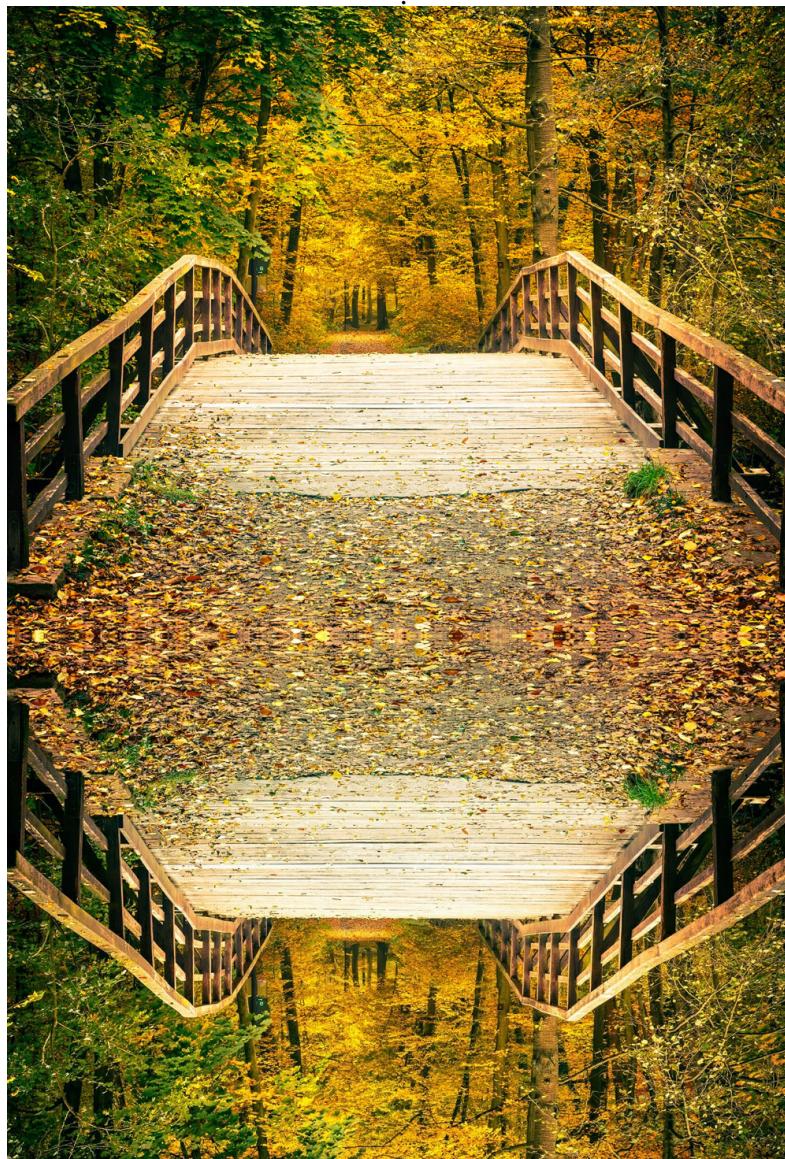
2.5 ÇEVİR

Çevir Bölümünde Yatay ve Dikey Olmak Üzere İki Butonla Resmin Yatay veya Dikey olarak çevrilmesi sağlanmıştır. Böylece Ters görünen resimler kullanıcı İsteğine bırakılıp ,kullanıcının resmi uygun gördüğü yönde sabit kılmasını sağlayabilmektedir.

2.5.1 *Yatay*

Resminizi Yatay olarak tersinin alınmasını sağlayabilirsiniz. Böylece resminiz yatay olarak simetrik bir görünüme sahip olup,simetrisi ile bütünleşip devamı niteliğinde olucaktır.

Yatay çevirme, resmin en alt kısmındaki matris değerleriyle en üst kısmındaki matris değerlerinin simetriksel ve 3 boyutsal olarak yer değiştirilmesiyle elde edilmiştir.



2.5.2 *Dikey*

Resminizi Dikey olarak tersini alabilirsiniz. Yani aynadaki görüntüsünü dikey olarak görüntüleyebilirsiniz.

Dikey çevirme, resmin en sol kısımdaki matris değerleriyle en sağ kısımdaki matris değerlerinin simetrikSEL ve 3 boyutsal olarak yer değiştirilmesiyle elde edilmiştir.



Şekil 20: Dikey Çevirme

2.6 Parlaklık

Parlaklık kısmında Bir hareketli çubuk yardımıyla resmin parlaklığını istenilen oranda artırıp veya azaltabilme imkanına sahipsiniz. Böylece karanlık resimleri aydınlatır veya aşırı parlak resimleri daha az parlaklığa ulaştırabilirsiniz.

2.6.1 Parlaklık Artırma

Göze hoş gelecek fotoğrafın en önemli kısmı harika bir pozlamadır. Fotoğraf çekerken ne kadar dikkatli olsak da bazı resimlerin karanlık çıkışmasını engelleyemeyiz.

Sensör doğru ışık miktarını emdiğinde, görünümü güzel bir fotoğraf elde ederiz. Özellikle karanlık mekanlarda çekilen fotoğraflarda sonuç memnun edici değildir. Mobil fotoğrafçılık günümüzde yaygın kullandığımız bir yöntem. Kaliteli telefonlar dışında bir çok cep telefonunun çektığı resimler malesef karanlık olabiliyor. Böyle bir durumda ne yapabilirim? Karanlık bir fotoğrafı aydınlatmak mümkün mü? Bunu nasıl yapabilirim, soruları karşımıza çıkıyor.

Bu durumda uygulamamızın Parlaklık butonu kullanılarak resmimize istediğimiz oranda parlaklık verebilebilir.

Parlaklık, Parıltı efektinin hareketli çubuk yardımıyla kullanıcının istediği oranda resme canlı olarak uygulaması sağlanmıştır.



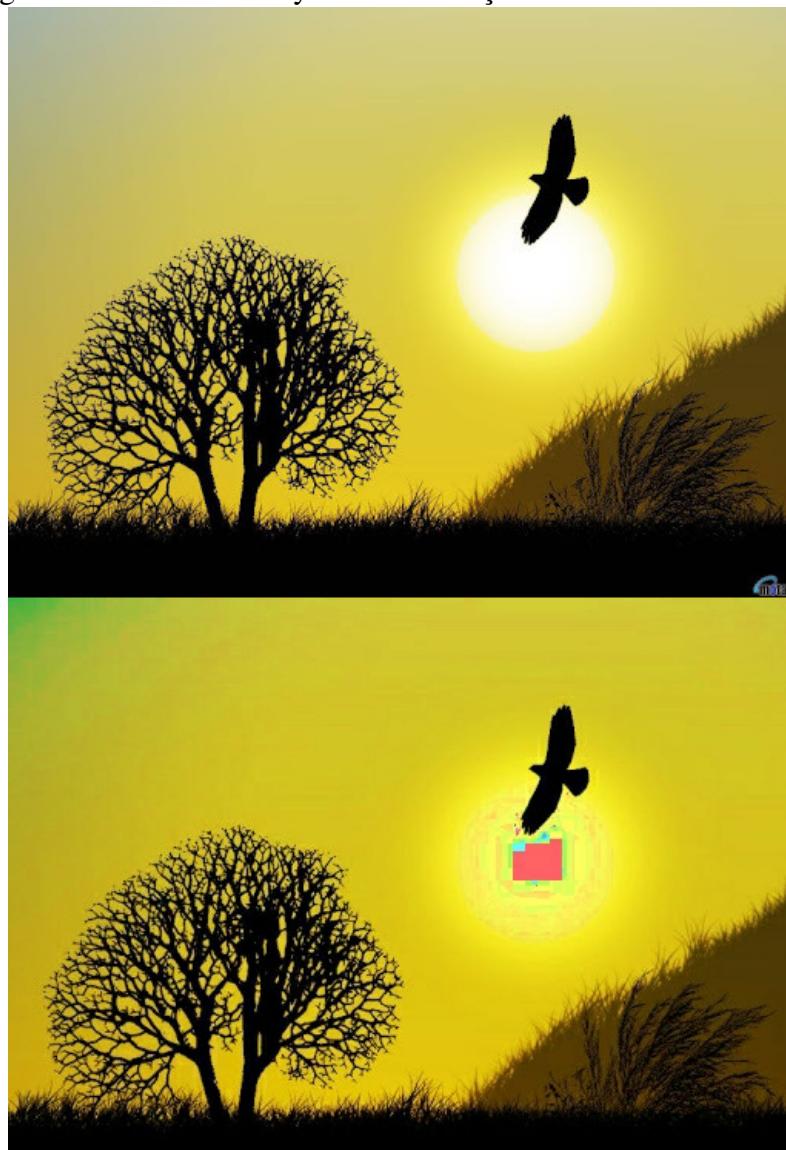
2.7 *Doyma*

Doyma; Kısaca resmin renk doygunluğudur. Renk konusunda daha az renklere sahip resimlerin daha yoğun renkler ile güzel görünmesini sağlayabilirisiniz.

2.7.1 *Doyma Artırma*

Renklerin şiddetini tanımlamak için doygunluk kelimesi kullanılır.Doygun durumdaki bir renk derin, zengin ve canlı görünür.Doygun olmayan halinde ise solgun, yumuşak ve zayıf bir görünüm bırakır. Bir rengin gösterdiği doygunluk seviyesiyle fotoğraf üzerindeki bu renge ait pigmentlerin yoğunluğu arasında fiziki birer ilişki mevcuttur.

Doyma artırma, resminizin hareketli çubuk yardımıyla 3. boyuttaki matrislerin renk oranlarının doğru orantıda artırılmasıyla elde edilmiştir.



2.8 *Gama*

Gama ; Resmin yüzeysel parlaklığını artırmak için kullanılır. Kullanıcı İstediği oranda resmin yüzeysel parlaklığını artırabilir.

2.8.1 *Gama Artırma*

Gamma değerini ayarlayarak resminizdeki orta değerli tonların parlaklığını değiştirebilirsiniz. Gamma değerini düşürdüğünüzde, resminizdeki orta değerli tonlar koyulaşır. Gamma değerini artttırdığınızda, orta değerli tonların rengi açılır. Her iki durumda da, parlak beyazlar ve çok koyu siyahlar etkilenmez.

Gama Artırma, resminizin matrislerinin ortalamasına yakın değerlikte değerlerin kullanıcının istediği oranda hareketli çubuk yardımıyla canlı olarak artırılabilmesi veya azaltılmasıyla elde edilmiştir.



Şekil 23: Gama Artırma

2.9 Kontrast

Kontrast resimdeki en aydınlık bölüm ve en karanlık bölüm arasındaki farka denir. Kontrast ayarı ile resimdeki bu farkı istediğiniz oranda artırabilirsiniz.

2.9.1 Kontrast Artırma

Kontrasti nasıl kullanacağınızı bilmeniz, birbirinden etkileyici fotoğraflar çekebilmenizi sağlar, bu yüzden nasıl kullanılacağının bilinmesi çok önemlidir. Kontrast, en genel manasıyla izleyicilerin dikkatini fotoğraftaki konuya yönlendirmek için kullanılan bir araçtır. İki çeşidi vardır: Ton Kontrasti ve Renk Kontrasti. Ton kontrasti, en açık tondan en koyu tonlara kadar olan tonlardaki farkı, başka bir deyişle, beyazdan griye kadar tonlardaki ton farkını ifade eder. Renk kontrasti ise renklerin birbiriyle nasıl etkileşim kurduğunu ifade eder.

Kontrast Artırma, resmin matrisel olarak birbirinden yüksek değerlikteki farklılıkların aynı oranda kullanıcının hareketli çubuk yardımıyla artırılıp veya azaltılmasıyla elde edilmiştir



2.10 Renk Açıklaştır

Renk Açıklaştırmayı Resme Kırmızı, Mavi ve Yeşil Renk tonlarının yansımaları sağlanabilir. Böylece Resmi maximum seviyede özelleştirme imkanını elinizde bulundurabilirsiniz.

2.10.1 Mavi

Mavi butonununa dokunulmasıyla gelen hareketli çubuğu kullanıcının resimdeki mavi renkleri istediği oranda artırmasını sağlamaktadır. Böylece mavinin eşsiz klasik görünümünden yararlanılabilir.

Mavi, matrislerin 3. boyutundaki mavilik oranının kullanıcının hareketli çubuk yardımıyla artırılarak elde edilmiştir.



2.10.2 *Yeşil*

Yeşil Butonuna basıldığında çıkan hareketli çubukla resminize Yeşil renk efektini ekleyebilirsiniz. Böylece resminiz gece görünümü özelliğine sahip olabilecektir. Ayrıca bu efekt ile doğa manzaralarında yeşil rengi ön plana çıkarmak içinde kullanılabilir...

Yeşil, matrislerin 3. boyutundaki Yeşillik oranının kullanıcının hareketli çubuk yardımıyla artırmasıyla elde edilmiştir.



Şekil 26: Yeşil Artırma

2.11 *Uygula*

Uygula Butonu İle Resmin Histogram veya Pseoude HDR modunda görüntülenmesini sağlayabilirsiniz .

2.11.1 *Histogram*

Histogram, fotoğraf çekimi veya sonrasında fotoğrafı detaylı bir şekilde inceleyebilmenizi, anlayabilmenizi sağlayan en önemli değerlendirme grafiklerinden birisidir.

Aslında matematiksel bir terim olarak karşımıza çıkan histogram, fotoğrafçılık alanında piksellerin ışık değerleri ile oluşturulmuş grafiğinden başka bir şey değildir. Bu sayede fotoğraflarınıza ait hızlı değerlendirmeler yapabilir ve buna göre makine ayarlarıyla oynayarak ışık değerlerini tekrar gözden geçirebilirsiniz.



Şekil 27: Histogram

2.11.2 *PseudeHDR*

HDR modu neredeyse yeni nesil bütün akıllı telefonlarda, fotoğraf makinelerinde ve televizyonlarda bulunan bir yazılım örneğidir. Örneğin izlediğiniz bir video HDR olarak çekildiyse bütün detaylara hakim olmanız daha olasıdır. HDR ne demek sorusu, kontrast farkı ve zengin renk seçeneği olarak açıklanabilir. Aslında bu tanım HDR modu nedir sorusunun da cevabı.

PseudeHDR, matristeki her bir piksellerdeki değerlerin bir yandaki piksele kopyalanıp ve bu pikselin diğer yandaki pikselin orantısına yakınsanıp çoğaltımasıyla elde edilen eski matrise oranla daha yüksek boyutlu olan yeni matrisin elde edilmesidir. Yani Çözünürlük artırmadır.



Şekil 28: PseudeHDR

BÖLÜM: III

YÖNTEM

3

YÖNTEM

3.1 *Image Processesing*

Görüntü İşleme, görüntüyü dijital form haline getirmek, bazı işlemleri gerçekleştirmek için geliştirilmiş, spesific görüntü elde etmek veya ondan bazı yararlı bilgiler çıkarmak için kullanılan bir yöntemdir. Bu yöntemin girdisi video kesiti veya fotoğraf gibi bir görüntüdür. Çıktısı ise görüntünün istenilen yada dikkat edilmesi gereken durumlara karşılık gelmektedir. Genellikle Görüntü İşleme sistemi, önceden belirlenmiş sinyal işleme(Signal Processing) yöntemlerini uygularken görüntüleri iki boyutlu sinyaller olarak ele almaktadır.

Günümüzde işletmelerin çeşitli yönleriyle kullandıkları görüntü işleme sistemleri hızla büyüyen teknolojiler arasında yer alır. Görüntü İşleme, mühendislik ve bilgisayar bilimleri disiplinlerinde temel araştırma alanını oluşturmaktadır.

Görüntü İşleme temel olarak aşağıdaki üç adımı içerir.

Görüntünün optik tarayıcı ile veya dijital fotoğraflarla alınması

Veri sıkıştırma, görüntü iyileştirme ve uygu fotoğrafları gibi insan gözü olmayan lekelenme kalıplarını içeren görüntüleri analiz etmede kullanma.

Çıktı, sonuçlarını görüntü analizine dayalı olarak değiştirip kullanıma hazır hale getirme.

Image Processing kısmında en etkili yöntem evrensel olarak kabul görmüş olan matlab uygulamasıdır. Kendine has diliyle matematiksel işlemleri en iyi derecede yapabilmesiyle maximum düzeyde işlevsellik sağlar.

Matlab Uygulamasından edinilen bu bilgiler RESİM DÜZENLEYİCİ MOBİL ANDROİD PROJESİ 'nde kullanılarak uygulamanın en üst düzeyde kullanılması sağlanmıştır.

[1]

3.2 *Android Studio*

Android Studio, Android uygulamalarının geliştirildiği, üst seviye özelliklere sahip ve Google tarafından da önerilen resmi programlama aracıdır.

Android Studio'nun kod geliştiricilere sunduğu temel özellikler şunlardır:

Gradle tabanlı, esnek proje inşa sistemi.

Hızlı ve zengin özellikli bir emülatör

Farklı özellik ve sürümlere göre çoklu APK çıktısı.

Genel uygulama özelliklerini oluşturmanıza ve örnek kodu içe aktarmanıza yardımcı olacak kod şablonları ve GitHub entegrasyonu

C ++ ve NDK desteği

Ekran tasarımlarını kolaylaştıran sürükle-bırak özellikli zengin editör.

Uygulamanın performansı, Kullanılabilirliği, farklı sürümlerde çalışılabilirliğinin kontrol edileceği test araçları ve frameworkler

Kolay ve güvenli APK imzalanması.

Ek uğraşa gerek kalmadan Google hizmetlerini uygulamaya ekleyebilme.

[2]

3.3 Java

Java teknolojisi nedir ve neden kullanmam gereklidir?

Java ilk olarak Sun Microsystems tarafından 1995 yılında piyasaya sürülen bir programlama dili ve bilgi işlem platformudur. Java yüklemeyinizde çalıştırılamayacağınız bir çok uygulama ve web sitesi mevcut olup her geçen gün sayıları artmaktadır. Java hızlı, güvenli ve güvenilirdir. Dizüstü bilgisayarlardan veri depolama merkezlerine, oyun konsollarından bilimsel süper bilgisayarlara, cep telefonlarından Internete kadar Java her yerde!

Java Dili Android yazılımların vazgeçilmez bir parçası olduğu için ek bir dile ihtiyaç duymadan uygulamamın stabilizasyonunu maximum seviyeye çıkarmak için kullanmış olduğum ana yapımdır.

Java Dili Özellikle Image Processing yöntemlerinde oldukça üst bir seviyeye sahiptir. Bu Projenin Ana Hattı Görüntü İşleme yöntemlerini kullanmak olduğu için Java Dilini RESİM DÜZENLEYİCİ MOBİL ANDROİD PROJESİ ’nde bir yöntem olarak kullanmış bulunmaktayım.

Bunlara Binanen Uygulamamda bulunan çeşitli efekt, filtre, çevirme, histogram çıkarma gibi işlemlerin yapılması için Matris işlemlerine ihtiyaç vardı. Bu İhtiyacı karşılamak adına bu dilden maximum seviyede faydalanilmıştır.

EKLER kısmında Java ’ da yazılmış kaynak kodlarının bir kısmını görebilirsiniz.

[3]

BÖLÜM: IV

S O N U Ç

4

S O N U Ç

Sonuç olarak RESİM DÜZENLEYİCİ MOBİL ANDROID UYGULAMA PROJESİ ile yapılan tüm işlemler kaynaklardan alınan eğitimlerin üzerine kodlaması, kod düzeni, oluşumu vs. Murat DEMİR'e aittir. Bu proje AR-GE konusunda son derece elverişlidir. Bundan sonraki dönemde AR-GE ' si yapılacaktır.

Bu proje ile java, android programlama, görüntü işleme, proje yönetimi gibi bir çok kazanıma sahip olunmuştur. Ayrıca proje için destekleri adına Danışmanım Dr. Öğr. Üyesi Sercan DEMİRCİ hocama çok teşekkür ederim.

K A Y N A K Ç A

- [1] MATLAB. Görüntü İşleme. <https://www.mathworks.com/support/learn-with-matlab-tutorials.html>, Mayıs 2020.
- [2] STUDIO, A. Android studio. <https://developer.android.com/training/basics/firstapp>, Mayıs 2020.
- [3] TUTORIALSPPOINT. Java. <https://www.tutorialspoint.com/java/index.htm>, Mayıs 2020.

5

E K L E R

5.1 ÖZGEÇMİŞ

Adı Soyadı: Murat DEMİR

Doğum Yeri ve Tarihi: BAFRA / 17.02.1995

İlköğretim: Cumhuriyet İlköğretim Okulu

Orta Öğretim: Bafra Anadolu Teknik Lisesi - Web Programlama Bölümü

Lisans Üniversite: Ondokuz Mayıs Üniversitesi, Bilgisayar Mühendisliği Özel Öğrencisi

Projeler: Adobe Flash Player Quiz Uygulaması - Bafra A.T.L Grafik Anismasyon projesi

, C# ile Web Tasarım Uygulaması ve Programlanması- Bafra A.T.L Web Tasarımı ve

Programlama projesi, Silverlight Web Uygulaması - Bafra A.T.L Gelişmiş İnternet Uygulamaları

projesi..., Matlab üzerinde görüntü işleme ile yapay zeka yardımı olmadan nesnelerin

kırılması - OMÜ Bilg. Müh. Özel Konular projesi, E-Ticaret Web sitesi tasarımı ve

programlanması OMÜ Web Programlama projesi, Hasta Kayıt Veritabanı ve Web Sitesi

Tasarımı ve Programlanması- OMÜ Veritabanı Yönetim Sistemleri projesi...

Başarılar: Bafra Cumhuriyet İlköğretim Okulu- *Sınıf Birinciği, Bafra Anadolu Teknik Lisesi - *Okul Birinciliği

Elektronik posta: murat.demir@bil.omu.edu.tr

Adres: Emirefendi mahallesi, Şehit Murat Yıldız sokak, No:28, BAFRA/SAMSUN

5.2 KODLANMASI

5.2.1 EFEKTLER

5.2.1.1 Sepya

```

for (i = this.windowHalfSize; i < h; i++) {
    for (j = this.windowHalfSize; j < w; j++) {
        processPixelPos = i * w + j;
        A = (colors[processPixelPos] >> 24) & 0xFF;
        avgR = 0;
        avgG = 0;
        avgB = 0;

        //Window
        for (k = -windowHalfSize; k <= windowHalfSize; k++) {
            for (l = -windowHalfSize; l <= windowHalfSize; l++) {
                pos = (i + k) * w + (h + l);
                avgR += (colors[pos] >> 16) & 0xFF;
                avgG = (colors[pos] >> 8) & 0xFF;
                avgB = colors[pos] & 0xFF;
            }
        }

        avgR = (int) ((float) avgR / totalPixel);
        avgG = (int) ((float) avgG / totalPixel);
        avgB = (int) ((float) avgB / totalPixel);
        nvalue = (A << 24) | (avgR << 16) | (avgG << 8) | avgB;
        colors[processPixelPos] = nvalue;
    }
}

```

Şekil 29: Sepya Efekti

5.2.1.2 Televizyon

```

for (i = 0; i < w; i++) {
    for (j = 0; j < h; j += gap) {
        R = G = B = 0;
        for (k = 0; k < gap; k++) {
            index = (j + k) * w + i;
            if (index < pixels.length) {
                R += ((pixels[index] >> 16) & 0xFF) / gap;
                G += ((pixels[index] >> 8) & 0xFF) / gap;
                B += (pixels[index] & 0xFF) / gap;
            }
        }
        R = normalize(R);
        G = normalize(G);
        B = normalize(B);

        for (k = 0; k < gap; k++) {
            index = (j + k) * w + i;
            if (index < pixels.length) {
                if (k == 0) {
                    pixels[index] = Color.rgb(R, green: 0, blue: 0);
                }
                if (k == 1) {
                    pixels[index] = Color.rgb(red: 0, G, blue: 0);
                }
                if (k == 2) {
                    pixels[index] = Color.rgb(red: 0, green: 0, B);
                }
            }
        }
    }
}

```

Sekil 30: Televizyon Efekti

5.2.1.3 Eskiz

```
int sketchColor = Color.parseColor( colorString: "#5c6274");
Log.v( tag: "Sketch Color", msg: sketchColor + "");
int centerGray;
int rightBottomIndex;
int rightBottomGray;
Bitmap bmOut = Bitmap.createBitmap(w, h, inp.getConfig());
for (i = 1; i < h - 1; i++) {
    for (j = 1; j < w - 1; j++) {
        pos = i * w + j;
        centerGray = getGray(originPixels[pos]);
        rightBottomIndex = (i + 1) * w + j + 1;
        if (rightBottomIndex < pixels.length) {
            rightBottomGray = getGray(originPixels[rightBottomIndex]);
            if (Math.abs(centerGray - rightBottomGray) >= threshold) {
                pixels[pos] = sketchColor;
            } else {
                pixels[pos] = Color.rgb( red: 255, green: 255, blue: 255);
            }
        }
    }
}
bmOut.setPixels(pixels, offset: 0, w, x: 0, y: 0, w, h);
return bmOut;
}
```

Sekil 31: Eskiz Efekti

5.2.1.4 Neon

```

for (i = halfMaskSize; i < height - halfMaskSize; i++) {
    for (j = halfMaskSize; j < width - halfMaskSize; j++) {
        index = 0;
        xVal = yVal = 0;
        for (m = -halfMaskSize; m <= halfMaskSize; m++) {
            for (n = -halfMaskSize; n <= halfMaskSize; n++) {
                oriPos = (i + m) * width + j + n;
                oriGray = grayscale(oriPixels[oriPos]);
                xVal += oriGray * xSobel[index];
                yVal += oriGray * ySobel[index];
                index++;
            }
        }

        afterGray = Math.abs(xVal) + Math.abs(yVal);
        if (afterGray > 255)
            afterGray = 255;
        if (afterGray < 0)
            afterGray = 0;

        pos = i * width + j;
        if (afterGray > threshold) {
            pixels[pos] = Color.rgb(neonR, neonG, neonB);
        } else {
            pixels[pos] = Color.rgb(red: 1, green: 1, blue: 1);
        }
    }
}
bmOut.setPixels(pixels, offset: 0, width, x: 0, y: 0, width, height);

```

Şekil 32: Neon Efekti

5.2.1.5 Lomo

```

double scaler;
int[] pixels = new int[w * h];
inp.getPixels(pixels, offset: 0, w, x: 0, y: 0, w, h);
for (i = 0; i < h; i++) {
    for (j = 0; j < w; j++) {
        distance = Math.sqrt(Math.pow(centerX - j, 2) + Math.pow(centerY - i, 2));
        pos = i * w + j;
        if (distance > roundRadius) {
            R = (pixels[pos] >> 16) & 0xFF;
            G = (pixels[pos] >> 8) & 0xFF;
            B = pixels[pos] & 0xFF;

            scaler = scale(distance);
            scaler = Math.abs(scaler);
            R = normalize( v: R - scaler);
            G = normalize( v: G - scaler);
            B = normalize( v: B - scaler);

            pixels[pos] = Color.rgb(R, G, B);
        }
    }
}
Bitmap bmOut = Bitmap.createBitmap(w, h, inp.getConfig());
bmOut.setPixels(pixels, offset: 0, w, x: 0, y: 0, w, h);

return bmOut;
}

```

Şekil 33: Lomo Efekti

5.2.1.6 Gri Tonlama

```

public Bitmap perform(Bitmap inp) {
    Bitmap bmOut = Bitmap.createBitmap(inp.getWidth(), inp.getHeight(),
        inp.getConfig());
    int A, R, G, B;
    int w = inp.getWidth();
    int h = inp.getHeight();
    int[] colors = new int[w * h];
    inp.getPixels(colors, offset: 0, w, x: 0, y: 0, w, h);
    int i = 0;
    int j = 0;
    int pos;
    int val;
    for (i = 0; i < h; i++) {
        for (j = 0; j < w; j++) {
            pos = i * w + j;
            A = (colors[pos] >> 24) & 0xFF;
            R = (colors[pos] >> 16) & 0xFF;
            G = (colors[pos] >> 8) & 0xFF;
            B = colors[pos] & 0xFF;
            if (!isBostColor) {
                R = G = B = (int) (0.299 * R + 0.587 * G + 0.114 * B);
            } else {
                val = intensityBost
                    + (int) (0.299 * R + 0.587 * G + 0.114 * B);
                if (val > 255) {
                    val = 255;
                }
                R = G = B = val;
            }
        }
    }
}

```

Sekil 34: Gri Tonlama Efekti

5.2.1.7 Negatif

```
    ...  
    int i = - w,  
    int j = 0;  
    int pos;  
    int nvalue;  
    for (i = 0; i < h; i++) {  
        for (j = 0; j < w; j++) {  
            pos = i * w + j;  
            A = (colors[pos] >> 24) & 0xFF;  
            R = 255 - (colors[pos] >> 16) & 0xFF;  
            G = 255 - (colors[pos] >> 8) & 0xFF;  
            B = 255 - colors[pos] & 0xFF;  
            nvalue = (A << 24) | (R << 16) | (G << 8) | B;  
            colors[pos] = nvalue;  
        }  
    }  
    bmOut.setPixels(colors, offset: 0, w, x: 0, y: 0, w, h);  
    return bmOut;  
}
```

Sekil 35: Negatif Efekti

5.2.2 FILTRELER

5.2.2.1 Gauss Bulanıklığı

```

double sumR = 0, sumG = 0, sumB = 0;
int index = 0;
int bound = maskSize / 2;

int row, col, i, j;
int pixel_index;
for (row = bound; row < height - bound; row++) {
    for (col = bound; col < width - bound; col++) {
        index = 0;
        sumR = sumG = sumB = 0;
        for (i = -bound; i <= bound; i++) {
            for (j = -bound; j < bound; j++) {
                pixel_index = (row + i) * width + col + j;
                if (pixel_index < pixels.length) {
                    sumR += ((tempPixels[pixel_index] >> 16) & 0xFF) * kernel[index];
                    sumG += ((tempPixels[pixel_index] >> 8) & 0xFF) * kernel[index];
                    sumB += (tempPixels[pixel_index] & 0xFF) * kernel[index];
                    index++;
                }
            }
        }
        pixels[row * width + col] = Color.rgb((int) sumR, (int) sumG, (int) sumB);
    }
}
bmOut.setPixels(pixels, offset: 0, width, x: 0, y: 0, width, height);
return bmOut;

```

Şekil 36: Gauss Bulanıklığı Filtresi

5.2.2.2 Kabartma

```
public class EmbossTransformation extends AbstractEffectTransformation {  
  
    private ConvolutionMatrix embossMatrix;  
  
    public EmbossTransformation() {  
        int[][] config = new int[][]{  
            {-1, -1, 0},  
            {-1, 0, 1},  
            {0, 1, 1}  
        };  
  
        embossMatrix = new ConvolutionMatrix(config, factor: 1, offset: 128);  
    }  
  
    @Override  
    public void setThumbImage(Bitmap bmThumb) {  
  
    }  
  
    @Override  
    public Bitmap getThumbnail() { return null; }  
  
    @Override  
    public Bitmap perform(Bitmap inp) { return embossMatrix.convolute(inp); }  
}
```

Şekil 37: Kabartma Filtresi

5.2.2.3 Roman Kabartma

```
public class RomanticEmbossTransformation extends AbstractEffectTransformation {
    private ConvolutionMatrix embossMatrix;

    public RomanticEmbossTransformation() {
        int[][] config = new int[][]{
            {-1, 0, -1},
            { 0, 4,  0},
            {-1, 0, -1}
        };
        embossMatrix = new ConvolutionMatrix(config, factor: 1, offset: 127);
    }

    @Override
    public void setThumbImage(Bitmap bmThumb) {

    }

    @Override
    public Bitmap getThumbnail() { return null; }

    @Override
    public Bitmap perform(Bitmap inp) { return embossMatrix.convolute(inp); }
}
```

Şekil 38: Roman Kabartma Filtresi

5.2.2.4 Graviür

```
public class EngravingTransformation extends AbstractEffectTransformation {  
  
    private ConvolutionMatrix engravingMatrix;  
  
    public EngravingTransformation() {  
        int[][] config = new int[][]{  
            {-2, 0, 0},  
            {0, 2, 0},  
            {0, 0, 0}  
        };  
  
        engravingMatrix = new ConvolutionMatrix(config, factor: 1, offset: 95);  
    }  
  
    @Override  
    public void setThumbImage(Bitmap bmThumb) {  
    }  
  
    @Override  
    public Bitmap getThumbnail() { return null; }  
  
    @Override  
    public Bitmap perform(Bitmap inp) { return engravingMatrix.convolute(inp); }  
}
```

Şekil 39: Gravür Filtresi

5.2.2.5 Keskinlik

```

for (i = WINDOW_H_SIZE; i < h - WINDOW_H_SIZE; i++) {
    for (j = WINDOW_H_SIZE; j < w - WINDOW_H_SIZE; j++) {
        laplacian_index = 0;
        R = G = B = 0;
        pixel_index = i * w + j;
        A = (pixels[pixel_index] >> 24) & 0xFF;

        for (k = -WINDOW_H_SIZE; k <= WINDOW_H_SIZE; k++) {
            for (l = -WINDOW_H_SIZE; l <= WINDOW_H_SIZE; l++) {
                pixel_index = (i + l) * w + j + k;
                R += ((pixels[pixel_index] >> 16) & 0xFF) * laplacian[laplacian_index];
                G += ((pixels[pixel_index] >> 8) & 0xFF) * laplacian[laplacian_index];
                B += (pixels[pixel_index] & 0xFF) * laplacian[laplacian_index];
                laplacian_index++;
            }
        }

        R = Math.min(255, Math.max(0, R));
        G = Math.min(255, Math.max(0, G));
        B = Math.min(255, Math.max(0, B));
        edgePixels[i * w + j] = (A << 24) | (R << 16) | (G << 8) | B;
    }
}

pixel_index = 1;

```

Şekil 40: Keskinlik Filtresi

5.2.2.6 LR Motion

```
public class LeftRightMotionBlurTransformation extends AbstractEffectTransformation {  
  
    private ConvolutionMatrix LRMBMatrix;  
  
    public LeftRightMotionBlurTransformation() {  
        int[][] config = new int[][]{  
            {0, 0, 0, 0, 0, 0, 0, 0, 0},  
            {0, 0, 0, 0, 0, 0, 0, 0, 0},  
            {0, 0, 0, 0, 0, 0, 0, 0, 0},  
            {0, 0, 0, 0, 0, 0, 0, 0, 0},  
            {0, 0, 0, 0, 0, 0, 0, 0, 0},  
            {2, 2, 2, 2, 2, 2, 2, 2, 2},  
            {0, 0, 0, 0, 0, 0, 0, 0, 0},  
            {0, 0, 0, 0, 0, 0, 0, 0, 0},  
            {0, 0, 0, 0, 0, 0, 0, 0, 0}  
        };  
        LRMBMatrix = new ConvolutionMatrix(config, factor: 18, offset: 0);  
    }  
}
```

Şekil 41: Lr Motion Filtresi

5.2.2.7 Ortalama

```

int[] pixels = new int[w * h];
inp.getPixels(pixels, offset: 0, w, x: 0, y: 0, w, h);
int pos;
int cPos;
int sumR, sumG, sumB;
int totalPix = WINDOW_H_SIZE * 2 + 1;
totalPix *= totalPix;
for (i = WINDOW_H_SIZE; i < h - WINDOW_H_SIZE; i++) {
    for (j = WINDOW_H_SIZE; j < w - WINDOW_H_SIZE; j++) {
        sumR = sumG = sumB = 0;
        pos = i * w + j;
        A = (pixels[cPos] >> 24) & 0xFF;
        for (k = -WINDOW_H_SIZE; k <= WINDOW_H_SIZE; k++) {
            for (l = -WINDOW_H_SIZE; l <= WINDOW_H_SIZE; l++) {
                pos = (i + l) * w + j + k;
                sumR += (pixels[pos] >> 16) & 0xFF;
                sumG += (pixels[pos] >> 8) & 0xFF;
                sumB += pixels[pos] & 0xFF;
            }
        }
        pixels[cPos] = (A << 24) | ((sumR / totalPix) << 16) | ((sumG / totalPix) << 8) | (sumB / totalPix);
    }
}
bmOut.setPixels(pixels, offset: 0, w, x: 0, y: 0, w, h);

return bmOut;
}

```

Şekil 42: Ortalama Filtresi

5.2.2.8 Pürüzsüz

```
public class SmoothTransformation extends AbstractEffectTransformation {  
  
    private ConvolutionMatrix smMatrix;  
  
    public SmoothTransformation() {  
        int[][] config = new int[][]{  
            {1, 1, 1},  
            {1, 7, 1},  
            {1, 1, 1}  
        };  
  
        smMatrix = new ConvolutionMatrix(config, factor: 15, offset: 1);  
    }  
  
    @Override  
    public void setThumbImage(Bitmap bmThumb) {  
  
    }  
  
    @Override  
    public Bitmap getThumbnail() { return null; }  
  
    @Override  
    public Bitmap perform(Bitmap inp) { return smMatrix.convolute(inp); }  
}
```

Şekil 43: Pürüzsüz Filtresi

5.2.2.9 Parıltı

```
public class HighlightTransformation extends AbstractEffectTransformation {  
  
    @Override  
    public Bitmap perform(Bitmap inp) {  
        Bitmap bmOut = Bitmap.createBitmap( width: inp.getWidth() + 96,  
                                         height: inp.getHeight() + 96, Bitmap.Config.ARGB_8888);  
        Canvas canvas = new Canvas(bmOut);  
        canvas.drawColor( color: 0, PorterDuff.Mode.CLEAR);  
        Paint ptBlur = new Paint();  
        ptBlur.setMaskFilter(new BlurMaskFilter( radius: 15, Blur.NORMAL));  
        int[] offsetXY = new int[2];  
        Bitmap bmAlpha = inp.extractAlpha(ptBlur, offsetXY);  
        Paint ptAlphaColor = new Paint();  
        ptAlphaColor.setColor(0xFFFFFFFF);  
        canvas.drawBitmap(bmAlpha, offsetXY[0], offsetXY[1], ptAlphaColor);  
        bmAlpha.recycle();  
        canvas.drawBitmap(inp, left: 0, top: 0, paint: null);  
        return bmOut;  
    }  
  
    @Override  
    public void setThumbImage(Bitmap bmThumb) { this.thumbImage = bmThumb; }  
  
    @Override  
    public Bitmap getThumbnail() { return perform(this.thumbImage); }  
}  
HighlightTransformation
```

Şekil 44: Parıltı Filtresi

5.2.2.10 Roman Kabartma

```
public class RomanticEmbossTransformation extends AbstractEffectTransformation {
    private ConvolutionMatrix embossMatrix;

    public RomanticEmbossTransformation() {
        int[][] config = new int[][]{
            {-1, 0, -1},
            { 0, 4,  0},
            {-1, 0, -1}
        };
        embossMatrix = new ConvolutionMatrix(config, factor: 1, offset: 127);
    }

    @Override
    public void setThumbImage(Bitmap bmThumb) {

    }

    @Override
    public Bitmap getThumbnail() { return null; }

    @Override
    public Bitmap perform(Bitmap inp) { return embossMatrix.convolute(inp); }
}
```

Şekil 45: Roman Kabartma Filtresi

5.2.3 ÇEVİR

5.2.3.1 Yatay

```
public class FlippingTransformation extends AbstractEffectTransformation {
    public static final int FLIP_VERTICAL = 1;
    public static final int FLIP_HORIZONTAL = 2;

    private int type;

    public FlippingTransformation(int type) { this.type = type; }

    @Override
    public void setThumbImage(Bitmap bmThumb) {

    }

    @Override
    public Bitmap getThumbnail() { return null; }

    @Override
    public Bitmap perform(Bitmap inp) {
        Matrix matrix = new Matrix();
        if (type == FLIP_HORIZONTAL) {
            // y = y * -1
            matrix.preScale( sx: 1.0f, sy: -1.0f);

        } else if (type == FLIP_VERTICAL) {
            matrix.preScale( sx: -1.0f, sy: 1.0f);
        }
        return Bitmap.createBitmap(inp, x: 0, y: 0, inp.getWidth(), inp.getHeight(), matrix, filter: true);
    }
}
```

Sekil 46: Yatay Çevirme

5.2.3.2 Dikey

```
public class FlippingTransformation extends AbstractEffectTransformation {
    public static final int FLIP_VERTICAL = 1;
    public static final int FLIP_HORIZONTAL = 2;

    private int type;

    public FlippingTransformation(int type) { this.type = type; }

    @Override
    public void setThumbImage(Bitmap bmThumb) {

    }

    @Override
    public Bitmap getThumbnail() { return null; }

    @Override
    public Bitmap perform(Bitmap inp) {
        Matrix matrix = new Matrix();
        if (type == FLIP_HORIZONTAL) {
            // y = y * -1
            matrix.preScale( sx: 1.0f, sy: -1.0f);

        } else if (type == FLIP_VERTICAL) {
            matrix.preScale( sx: -1.0f, sy: 1.0f);
        }
        return Bitmap.createBitmap(inp, x: 0, y: 0, inp.getWidth(), inp.getHeight(), matrix, filter: true);
    }
}
```

Şekil 47: Dikey Çevirme

5.2.4 Parlaklık

5.2.4.1 Parlaklık Artırma

```

for (i = 0; i < h; i++) {
    for (j = 0; j < w; j++) {
        pos = i * w + j;
        A = (colors[pos] >> 24) & 0xFF;
        R = (colors[pos] >> 16) & 0xFF;
        R += this.currentValue;
        if (R > 255) {
            R = 255;
        } else if (R < 0) {
            R = 0;
        }
        G = (colors[pos] >> 8) & 0xFF;
        G += this.currentValue;
        if (G > 255) {
            G = 255;
        } else if (G < 0) {
            G = 0;
        }
        B = colors[pos] & 0xFF;
        B += this.currentValue;
        if (B > 255) {
            B = 255;
        } else if (B < 0) {
            B = 0;
        }
        nvalue = (A << 24) | (R << 16) | (G << 8) | B;
        colors[pos] = nvalue;
    }
}

```

Şekil 48: Parlaklık Artırma

5.2.5 Doyma

Doyma Artırma

```

public Bitmap perform(Bitmap inp) {
    Bitmap bmOut = Bitmap.createBitmap(inp.getWidth(), inp.getHeight(), inp.getConfig());
    int A, R, G, B;
    int i, j;
    int w = inp.getWidth();
    int h = inp.getHeight();
    int[] pixels = new int[w * h];
    inp.getPixels(pixels, offset: 0, w, x: 0, y: 0, w, h);
    float[] HSV = new float[3];
    float remainSat;
    int pos;
    for (i = 0; i < h; i++) {
        for (j = 0; j < w; j++) {
            pos = i * w + j;
            Color.colorToHSV(pixels[pos], HSV);
            remainSat = 1.0f - HSV[1];
            HSV[1] += remainSat * adjustAmount;
            HSV[1] = (float) Math.max(0.0, Math.min(HSV[1], 1.0));
            pixels[pos] = Color.HSVtoColor(HSV);
        }
    }
    bmOut.setPixels(pixels, offset: 0, w, x: 0, y: 0, w, h);
    return bmOut;
}

```

Şekil 49: Doyma Artırma

5.2.6 Gama

5.2.6.1 Gama Artırma

```

int[] gammaR = new int[MAX_SIZE];
int[] gammaG = new int[MAX_SIZE];
int[] gammaB = new int[MAX_SIZE];

int i, j;
for (i = 0; i < MAX_SIZE; i++) {
    gammaR[i] = (int) Math.min(
        MAX_INT_VALUE,
        MAX_DBL_VALUE * Math.pow(i / MAX_DBL_VALUE, REVERSE / red));
    gammaG[i] = (int) Math.min(
        MAX_INT_VALUE,
        MAX_DBL_VALUE* Math.pow(i / MAX_DBL_VALUE, REVERSE / green));
    gammaB[i] = (int) Math.min(
        MAX_INT_VALUE,
        MAX_DBL_VALUE* Math.pow(i / MAX_DBL_VALUE, REVERSE/ blue));
}
inp.getPixels(pixels, offset: 0, w, x: 0, y: 0, w, h);
int pos;
for (i = 0; i < h; i++) {
    for (j = 0; j < w; j++) {
        pos = i * w + j;
        A = (pixels[pos] >> 24) & 0xFF;
        R = gammaR[(pixels[pos] >> 16) & 0xFF];
        G = gammaG[(pixels[pos] >> 8) & 0xFF];
        B = gammaB[pixels[pos] & 0xFF];
        pixels[pos] = (A << 24) | (R << 16) | (G << 8) | B;
    }
}

```

Şekil 50: Gama Artırma

5.2.7 Kontrast

Kontrast Artırma

```
Bitmap bmOut = Bitmap.createBitmap(w, h, inp.getConfig());
double contrast = ((maxValue + currentValue) / (double) maxValue);
contrast *= contrast;
int[] contrastLT = new int[256];
for (i = 0; i < 256; i++) {
    contrastLT[i] = (int) (((((i / 255.0) - 0.5) * contrast) + 0.5) * 255.0);
    if (contrastLT[i] < 0)
        contrastLT[i] = 0;
    if (contrastLT[i] > 255)
        contrastLT[i] = 255;
}
int tmpPx;
for (i = 0; i < h; i++) {
    for (j = 0; j < w; j++) {
        pos = i * w + j;

        A = (pixels[pos] >> 24) & 0xFF;
        R = contrastLT[(pixels[pos] >> 16) & 0xFF];
        G = contrastLT[(pixels[pos] >> 8) & 0xFF];
        B = contrastLT[pixels[pos] & 0xFF];

        pixels[pos] = (A << 24) | (R << 16) | (G << 8) | B;
    }
}
```

Şekil 51: Kontrast Artırma

5.2.8 Renk Açıklıyor

5.2.8.1 Mavi

```

for (i = 0; i < h; i++) {
    for (j = 0; j < w; j++) {
        pos = i * w + j;
        A = (pixels[pos] >> 24) & 0xFF;

        R = (pixels[pos] >> 16) & 0xFF;
        if (this.type == RED) {
            R = (int) (R * (1 + percent));
            if (R > 255)
                R = 255;
        }
        G = (pixels[pos] >> 8) & 0xFF;
        if (this.type == GREEN) {
            G = (int) (G * (1 + percent));
            if (G > 255)
                G = 255;
        }

        B = pixels[pos] & 0xFF;
        if (this.type == BLUE) {
            B = (int) (B * (1 + percent));
            if (B > 255)
                B = 255;
        }
    }
}

```

Şekil 52: Mavi Artırma

5.2.8.2 Yeşil

```

for (i = 0; i < h; i++) {
    for (j = 0; j < w; j++) {
        pos = i * w + j;
        A = (pixels[pos] >> 24) & 0xFF;

        R = (pixels[pos] >> 16) & 0xFF;
        if (this.type == RED) {
            R = (int) (R * (1 + percent));
            if (R > 255)
                R = 255;
        }
        G = (pixels[pos] >> 8) & 0xFF;
        if (this.type == GREEN) {
            G = (int) (G * (1 + percent));
            if (G > 255)
                G = 255;
        }

        B = pixels[pos] & 0xFF;
        if (this.type == BLUE) {
            B = (int) (B * (1 + percent));
            if (B > 255)
                B = 255;
        }
    }
}

```

Şekil 53: Yeşil Artırma

5.2.9 Uygula

Histogram

```

for (row = 0; row < height; row++) {
    for (col = 0; col < width; col++) {
        index = row * width + col;
        smoothR = (smoothPixels[index] >> 16) & 0xFF;
        oriR = (pixels[index] >> 16) & 0xFF;
        if (smoothR / 255.0 <= 0.5) {
            newR = 2 * (smoothR / 255.0) * (oriR / 255.0);
        } else {
            newR = 1 - 2 * (1 - oriR / 255.0) * (1 - smoothR / 255.0);
        }

        smoothG = (smoothPixels[index] >> 8) & 0xFF;
        oriG = (pixels[index] >> 8) & 0xFF;
        if (smoothG / 255.0 <= 0.5) {
            newG = 2 * (smoothG / 255.0) * (oriG / 255.0);
        } else {
            newG = 1 - 2 * (1 - oriG / 255.0) * (1 - smoothG / 255.0);
        }

        smoothB = smoothPixels[index] & 0xFF;
        oriB = pixels[index] & 0xFF;
        if (smoothB / 255.0 <= 0.5) {
            newB = 2 * (smoothB / 255.0) * (oriB / 255.0);
        } else {
            newB = 1 - 2 * (1 - oriB / 255.0) * (1 - smoothB / 255.0);
        }
        blurA = (smoothPixels[index] >> 24) & 0xFF;
        newR *= 255;
        newG *= 255;
    }
}

```

Şekil 54: Histogram