بهبود کنتراست

ترانه قندی

اطلاعات گزارش

تاریخ: 13980816

واژگان کلیدی:
بهبود کنتراست
هیستوگرام
تصویر رنگی
همسان سازی هیستوگرام
همسان‌سازی محلی هیستوگرام
همسان سازی پویای هیستوگرام
بهبود کنتراست به شیوه ی فازی

چکیده

در بین روشهای بهبود کیفیت تصاویر، می‌توان به بهبود کنتراست تصویر اشاره کرد. کنتراست به معنی تمایز بین سطوح روشنایی و یا رنگهاست که باعث بهتر تشخیص دادن جزییاتی از تصویر خواهد شد. بهبود کنتراست به عنوان پیش پردازشی پر کاربرد، در بسیاری از پردازش ها پیش از پردازش اصلی به کار می‌رود.

1-مقدمه

بهبود کنتراست، از پیش پردازش‌های مهمی به شمار می‌رود که کاربرد فراوانی دارد. بهبود کنتراست به ویژه در تصاویر پزشکی به دلیل نمایان کردن جزییات تصویر که پیش از این پردازش قابل رویت نبوده اند، حائز اهمیت است.همچنین بهبود کنتراست در تصاویر غیر پزشکی، از نظر بصری تصویر را بهبود می‌بخشند.

همسان سازی هیستوگرام تصویر، از روشهای معروف بهبود کنتراست می‌باشد که برخی از الگوریتم های ارائه شده در این گزارش برپایه‌ی این شیوه هستند. از دیگر روشهای بهبود کنتراست می‌توان به روش فازی نیز اشاره کرد.

در این گزارش به شرح و توضیح بهبود کنتراست تصاویر پزشکی تهیه شده از شبکیه با چندین الگوریتم می‌پردازیم.

2-شرح تکنیکال

- **هیستوگرام**

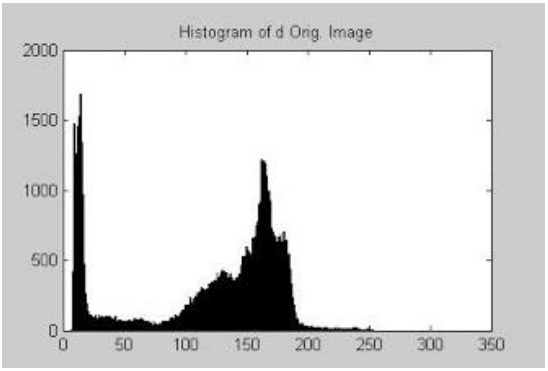
هیستوگرام یک تصویر، تابعی گسسته است که اطلاعات تصویر دو بعدی را به برداری یک بعدی تبدیل می‌کند و به صورت زیر تعریف می‌شود:

$$h_r(k)=n_k$$

که در آن k نشان دهنده‌ی سطح خاکستری، r_k نشان دهنده ی k اُمین سطح خاکستری و n_k تعداد پیکسل های تصویر است که دارای سطح خاکستری r_k هستند. در هیستوگرام تصویر، اطلاعات مکانی از بین می‌رود و برگشت پذیر نیست؛ اما هیستوگرام تصویر اطلاعات ارزشمندی در مورد تصویر در اختیار می‌گذارد. هیستوگرامی مطلوب است که یکنواخت باشد و دارای واریانس بالایی داشته باشد؛ به این معنی

که سطوح خاکستری موجود در تصویر زیاد باشند. تصویر CameraMan در ادامه به عنوان مثال آورده شده است.

تصویر 1-تصویر CameraMan



تصویر 2- هیستوگرام مربوط به تصویر CameraMan

- **هیستوگرام نرمالیزه شده**

هیستوگرام نرمالیزه شده، با تقسیم هرمقدار هیستوگرام بر تعداد کل پیکسل های تصویر (n) به دست می‌آید:

$$P_r(k) = n_k/n$$

هیستوگرام نرمالیزه شده از آنجا که احتمال وقوع هر سطح خاکستری را در یک تصویر بیان می‌کند،اطلاعات آماری مفیدی را در اختیار می‌گذارد.

اکنون به بررسی روشهای بهبود کنتراست که پیاده سازی کرده ایم، می‌پردازیم.

2.1-الگوریتم‌ها

- **همسان سازی هیستوگرام**

همسان سازی هیستوگرام روشی است که در آن هیستوگرام تصویر اصلی گسترده می‌شود. درنتیجه ی این گستردگی، هیستوگرام یکنواخت تر خواهد شد و تعداد سطوح خاکستری موجود در تصویر بیشتر خواهد بود.

$$s = T(r),\ 0 \le r \le 1$$

که در آن s و r مقادیر نرمالیزه شده ی شدت روشنایی پیکسل ها هستند.

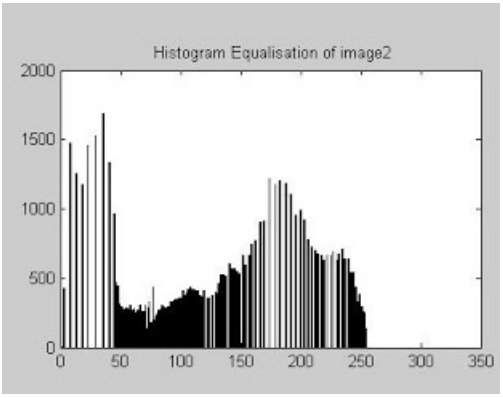
- **شروط تبدیل**
  - $T(r)$ در بازه‌ی $0 \le r \le 1$ صعودی باشد.
  - برای $0 \le r \le 1$، باید $0 \le T(r) \le 1$ باشد.

در این روش پس از محاسبه ی هیستوگرام نرمالیزه شده، CDF (Cumulative Distribution function) محاسبه می‌شود. اگر احتمال وقوع یک سطح خاکستری را p درنظر بگیریم، CDF به صورت زیر محاسبه می‌شود:

$$CDF(i) = \sum_{j=0}^{i} p(j)$$

پس از محاسبه ی CDF برای هیستوگرام نرمالیزه شده، برای هر سطح خاکستری، تعداد پیکسل ها را در مقدار متناظر آن سطح در CDF ضرب می‌کنیم. جزء صحیح مقدار به دست آمده، مقدار جدید در هیستوگرام همسان سازی شده است. در نهایت به یک Lookup table خواهیم رسید که مقادیر سطوح خاکستری را پس از همسان سازی شدن مشخص می‌کند.

تصویر 3-تصویر CameraMan پس از همسان سازی هیستوگرام



تصویر 4-هیستوگرام همسان سازی شده ی تصویر CameraMan

- **همسان سازی محلی هیستوگرام**

همسان سازی هیستوگرام که پیش تر شرح داده شد برای بهبود کلی تصویر مناسب است؛ اما ویزگی های مربوط به روشنایی محلی تصویر را درنظر نمی‌گیرد. اگر برخی نواحی در تصویر دارای روشنایی بالایی باشند، در بقیه ی نواحی تاثیر گذاشته و بر آنها غالب خواهند شد.

روشی دیگر برای همسان سازی هیستوگرام، همسان سازی محلی است که این مشکل را برطرف می‌کند.

در این روش پنجره ای با ابعاد مشخص شده در نظر گرفته می‌شود و تصویر به قطعاتی به ابعاد این پنجره تقسیم شده، همسان سازی هیستوگرام بر روی این قطعات صورت می‌گیرد.

ازطرفی این روش بار محاسباتی بالاتری خواهد داشت و گاه برخی نواحی از تصویر را بیش از حد تغییر می‌دهد. همچنین نویز موجود در تصویر نیز همراه با تصویر شدید تر می‌شود. مشکل دیگر این روش ایجاد افکت بلاکی است که موجب کاهش کیفیت تصویر از نظر بصری می‌شود.

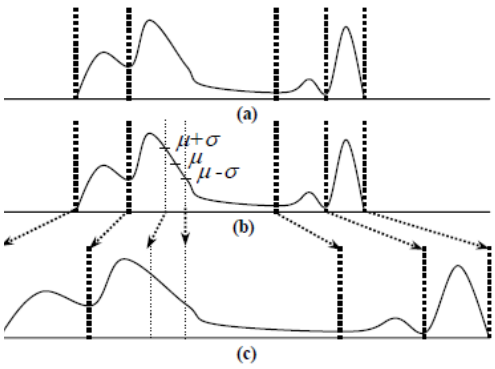
- **همسان سازی پویای هیستوگرام**

در روش های پیشین، این امکان وجود دارد که بخشهایی از هیستوگرام که مقادیر بزرگتری دارند روی نواحی دیگر تصویر تاثیر بگذارند.

روش دیگر برای بهبود کنتراست، استفاده از همسان سازی پویای هیستوگرام است که در آن هیستوگرام تصویر اصلی به بخشهایی تقسیم می‌شود، به گونه ای که در هیچ زیر-هیستوگرامی، مقدار غالبی وجود نداشته باشد. سپس برای هر زیر-هیستوگرام، تابع تبدیل با توجه به همسان سازی استاندارد که پیش تر شرح داده شد تولید می‌شود.

زیرهیستوگرام ها با توجه به local minima های هیستوگرام تصویر اصلی تعیین می‌شوند. ابتدا یک فیلتر smoothing روی هیستوگرام اعمال می‌شود تا minima های ناچیز حذف شوند. سپس بخشهای بین دو minima به عنوان یک زیر-هیستوگرام جدا می‌شوند. به عبارتی، اگر $m_0, m_1, \ldots, m_n$ ، $(n+1)$ سطح خاکستری باشند، اولین زیر-

هیستوگرام بین m_0 و m_1 قرار می‌گیرد. این روش تا حدودی باعث می‌شود بخشهایی که درواقع peak هیستوگرام تصویر اصلی می‌باشند، روی باقی نواحی تاثیر بگذارند.
اما حتی این عمل تضمین نمی‌کند که برخی بخشها بر دیگر نواحی غالب نشوند.
برای اینکه از این موضوع اطمینان حاصل کنیم، mean(μ) و standard deviation(σ) را برای هر زیر-هیستوگرام محاسبه می‌کنیم. اگر در یک زیر-هیستوگرام تعداد سطوح خاکستری که مقادیری بین (μ-σ) تا (μ+σ) داشته باشند از 63.5 درصد تعداد کل مقادیر زیر-هیستوگرام شود،آن زیرهیستوگرام توزیعی نرمال دارد و بخش غالبی در آن وجود ندارد. در غیر این صورت، آن زیر-هیستوگرام به سه بخش دیگر درمکانهایی که سطح خاکستری آنها (μ- σ) و (μ+ σ) باشند، تقسیم می‌شود و بخشهای اول و سوم دوباره بررسی خواهند شد. اما بخش میانی قطعا نرمال است[1].



تصویر 5-(a) تقسیم هیستوگرام اصلی به زیر-هیستوگرام(b)تقسیم دوباره ی یک زیر-هیستوگرام به دلیل نرمال نبودن توزیع(c)تخصیص سطوح خاکستری به زیر-هیستوگرام ها

- **بهبود کنتراست تصویر به روش فازی**

این روش از عملیات نقطه ای استفاده می‌کند. در این الگوریتم:

  - تصویر به صورت لگاریتمی بازنمایی می‌شود و مقادیر تصویر اجزائی از فضای دیگر(اقلیدسی) هستند و نه حقیقی.
  - تصویر توسط قطعه های فازی پردازش می‌شود.

متدهای ساده اما قدرتمندی برای بهبود کیفیت تصویر توسط تبدیلات affine در قالب مدل لگاریتمی قابل استفاده هستند. درصورتی که قطعه های مورد پردازش در تصویر از نظر آماری یکنواخت باشند، نتایج بهتری به دست خواهد آمد.
در روشهایی همچون همسان سازی محلی، با افکت بلاکی(block-effect) روبرو خواهیم شد. الگوریتم فازی به جای قطعه های معمولی در که همسان سازی محلی مورد استفاده بودند، از قطعات فازی (fuzzy window) استفاده می‌کند.
در هر یک از پنجره های فازی، یک تبدیل affine (در فضای لگاریتمی) با استفاده از میانه ی فازی و واریانس فازی که برای هریک از پیکسل های پنجره محاسبه می‌شود، تعریف می‌شود.
تصویر نهایی از مجموع وزن دار هر پنجره ی فازی به دست می‌آید. وزنهای مورد استفاده همان درجه‌ی عضویت هستند که در منطق فازی تعریف می‌شوند و قطعات فازی را تعریف می‌کنند.

- **الگوریتم انتخابی**

الگوریتم انتخاب شده از بین الگوریتم هایی که شرح داده شد، الگوریتم فازی است، زیرا

از اطلاعات آماری تصویر برای بهبود تصویر استفاده می‌کند که باعث می‌شود مشکلات روشهای دیگر، از جمله تشدید بیش از حد، افکت بلاکی، و تاثیر گذاشتن مقادیر بزرگ هیستوگرام بر روی باقی نواحی و به وجود آمدن اثر شسته شده(washed-out effect) رخ ندهد.

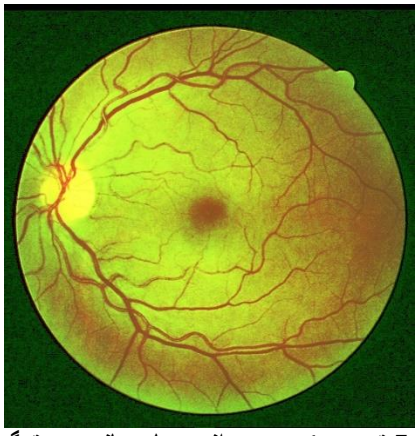
3-شرح نتایج

الگوریتم های شرح داده شده بر روی تصاویر eye1 تا eye8 اجرا شده اند.پردازش هم برروی تصویر اصلی (سه بانده) و هم برروی باند سبز تصویر صورت گرفته است زیرا در آن بیشترین کنتراست بین زمینه و عروق و ضایعات وجود دارد.

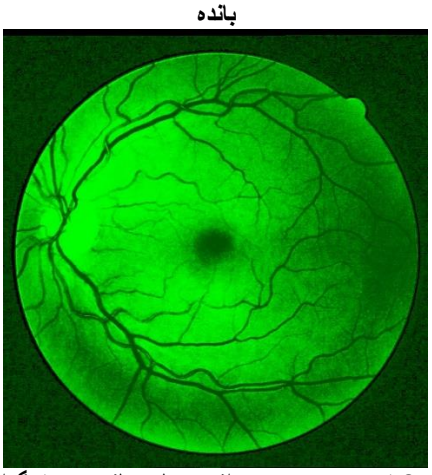
- **نتایج حاصل از همسان سازی هیستوگرام**

نتایج حاصل از این الگوریتم، ظاهرا فاقد باند آبی می‌باشند؛ درحالیکه این پدیده بر اثر بارزتر شدن باند سبز رخ داده است.همانطور که پیشتر اشاره شد، یکی از مشکلات روش همسان سازی هیستوگرام، تشدید بیش از حد برخی ویژگیهاست. این پدیده در مورد عکس eye3 و eye4 کمتر رخ داده است.

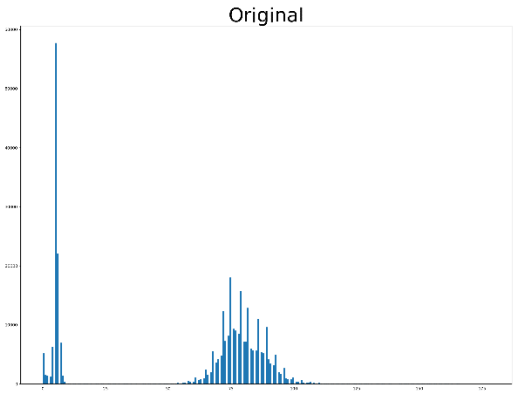
تصویر eye1-6



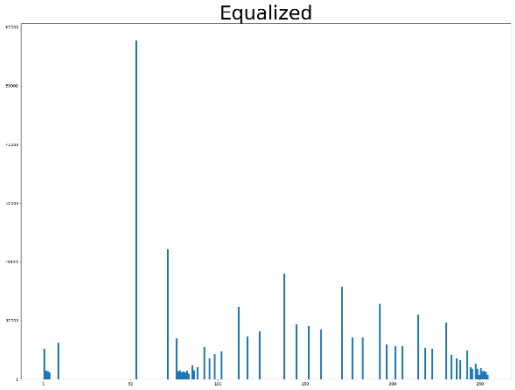
تصویر 7-تصویر eye1 پس از همسان سازی هیستوگرام-سه بانده



تصویر 8-تصویر eye1 پس از همسان سازی هیستوگرام-تک بانده

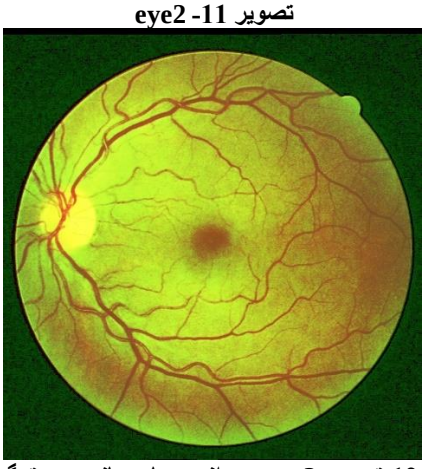


تصویر 9-هیستوگرام اصلی مربوط به eye1

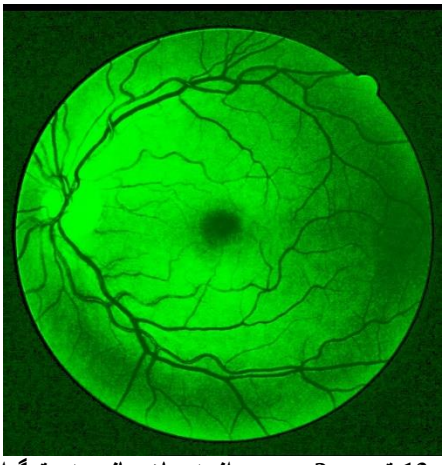


تصویر 10-هیستوگرام همسان سازی شده برای eye1

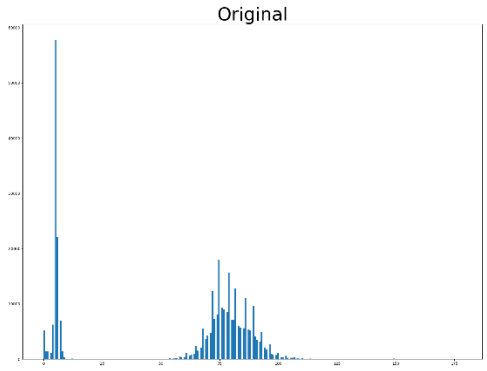
تصویر 11- eye2



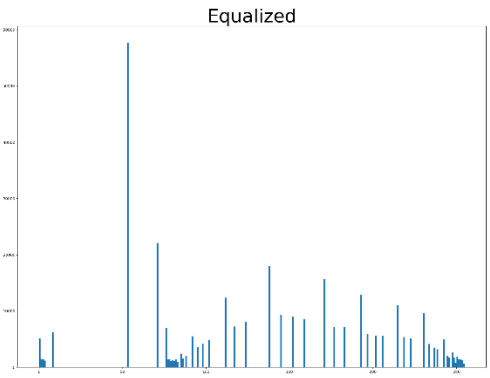
تصویر 12-تصویر eye2 پس از همسان سازی هیستوگرام-سه باند



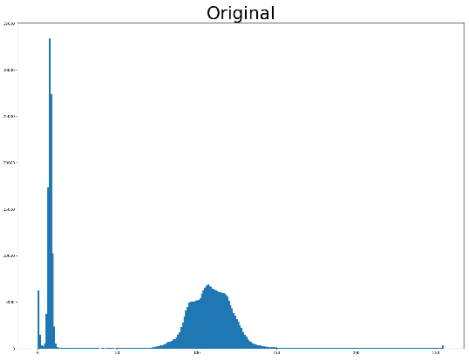
تصویر 13-تصویر eye2 پس از همسان سازی هیستوگرام-تک باند



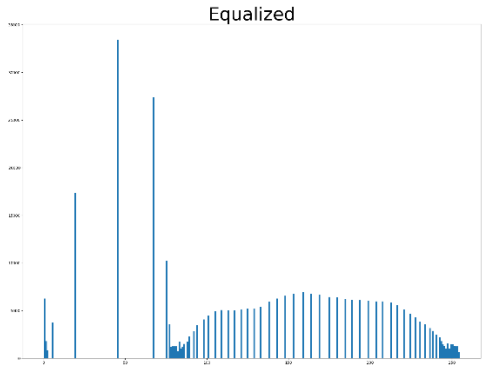
تصویر 14-هیستوگرام اصلی eye2



تصویر 15-هیستوگرام همسان سازی شده eye2



تصویر 19-هیستوگرام اصلی eye3



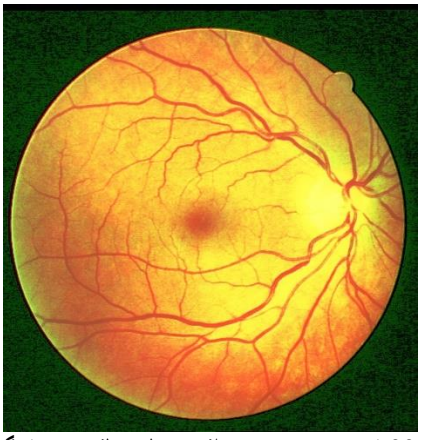
تصویر 20-هیستوگرام همسان سازی شده eye3

تصویر eye4-21

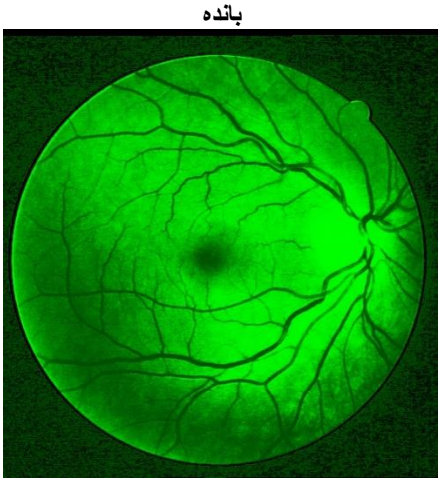
تصویر eye3-16

تصویر 17-تصویر eye3 پس از همسان سازی هیستوگرام-سه باند

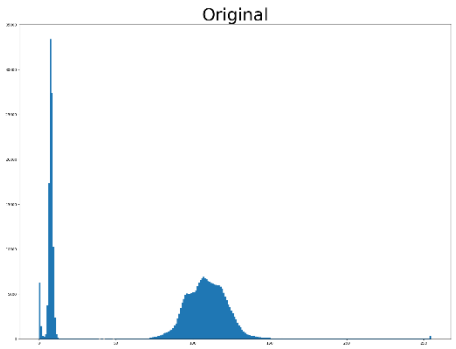
تصویر 18-تصویر eye3 پس از همسان سازی هیستوگرام-تک باند



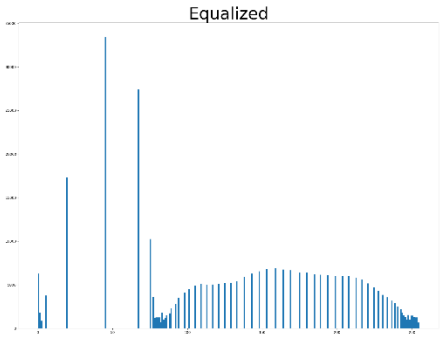
تصویر 22-تصویر eye4 پس از همسان سازی هیستوگرام-سه باند



تصویر 23- تصویر eye4 پس از همسان سازی هیستوگرام-تک باند

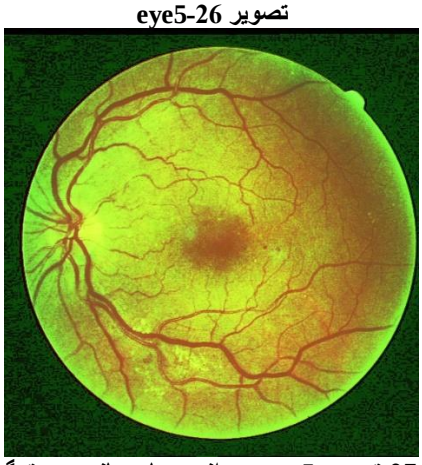


تصویر 24-هیستوگرام اصلی eye4

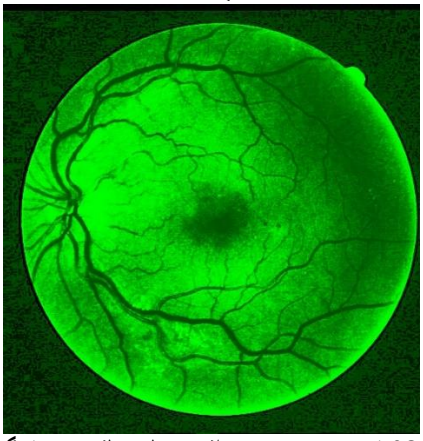


تصویر 25-هیستوگرام همسان سازی شده eye4

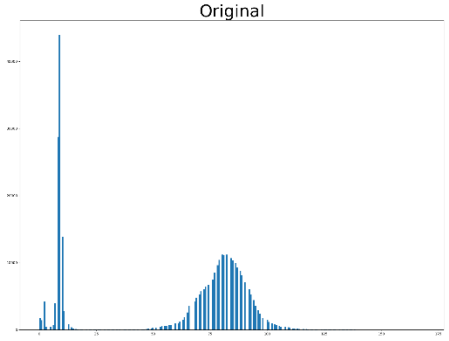
تصویر eye5-26



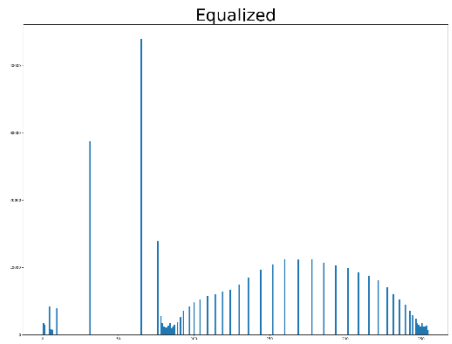
تصویر 27-تصویر eye5 پس از همسان سازی هیستوگرام-سه باند



تصویر 28-تصویر eye5 پس از همسان سازی هیستوگرام-تک باند



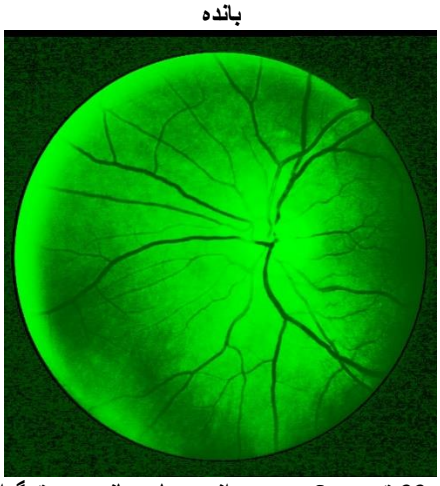
تصویر 29-هیستوگرام اصلی تصویر eye5



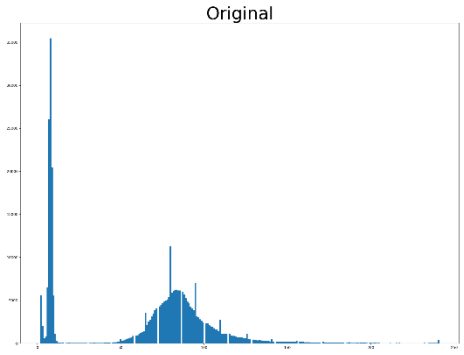
تصویر 30-هیستوگرام همسان سازی شده eye5

تصویر eye6-31

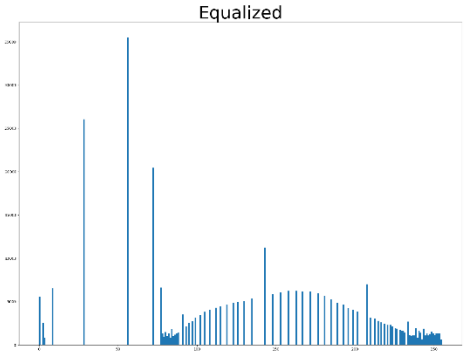
تصویر 32-تصویر eye6 پس از همسان سازی هیستوگرام-سه بانده



تصویر 33-تصویر eye6 پس از همسان سازی هیستوگرام-تک بانده

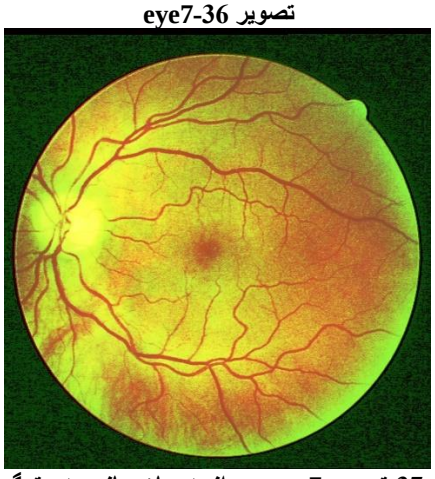


تصویر 34-هیستوگرام اصلی eye6

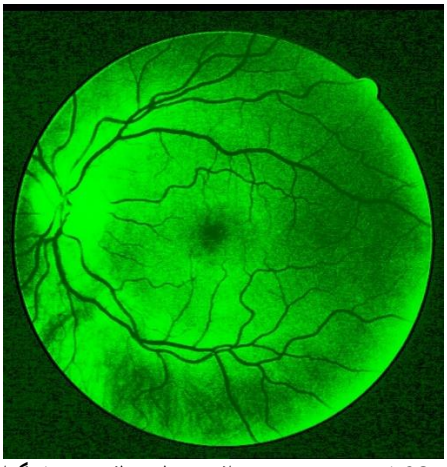


تصویر 35-هیستوگرام همسان سازی شدهeye6

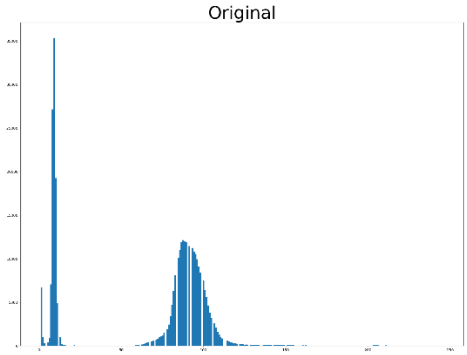
تصویر eye7-36



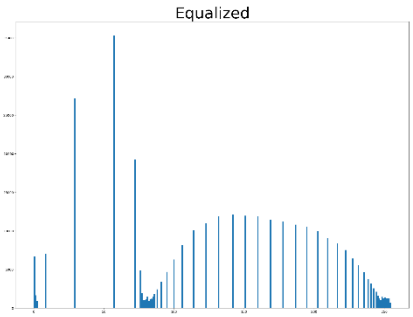
تصویر 37-تصویر eye7 پس از همسان سازی هیستوگرام-سه باند



تصویر 38-تصویر eye7 پس از همسان سازی هیستوگرام-تک باند



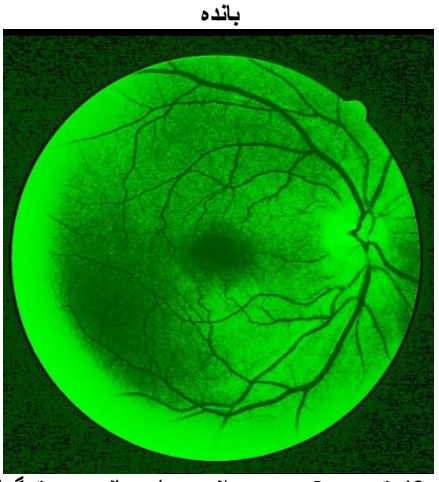
تصویر 39-هیستوگرام اصلی eye7



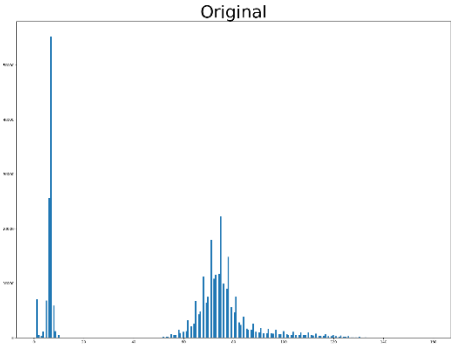
تصویر 40-هیستوگرام همسان سازی شده eye7

تصویر 41-eye8

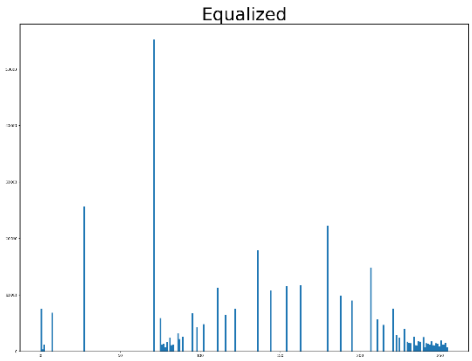
تصویر 42-تصویر eye8 پس از همسان سازی هیستوگرام-سه باند



تصویر 43-تصویر eye8 پس از همسان سازی هیستوگرام-تک باند



تصویر 44-هیستوگرام اصلی eye8

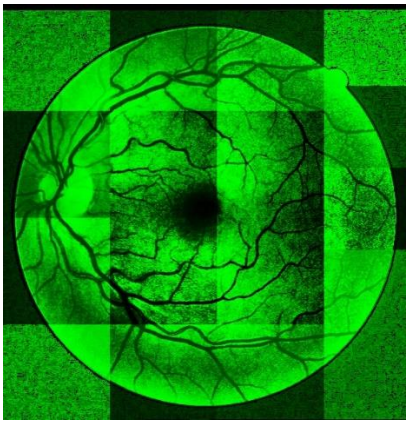


تصویر 45-هیستوگرام همسان سازی شده eye8

- **همسان سازی محلی هیستوگرام**

در این روش نیز افکت بلاکی، از کیفیت تصویر از نظر بصری می‌کاهد؛ اما ویژگیهای محلی تصویر را حفظ می‌کند.

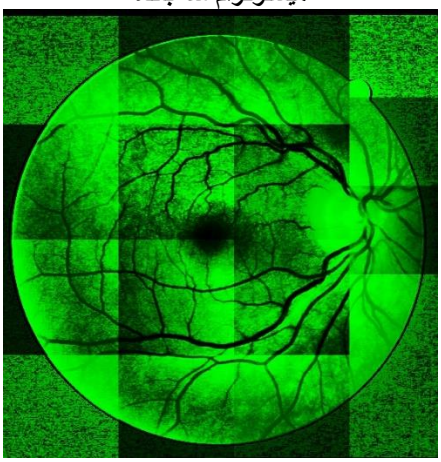
تصویر 46-تصویر eye1 پس از همسان سازی محلی هیستوگرام-سه باند



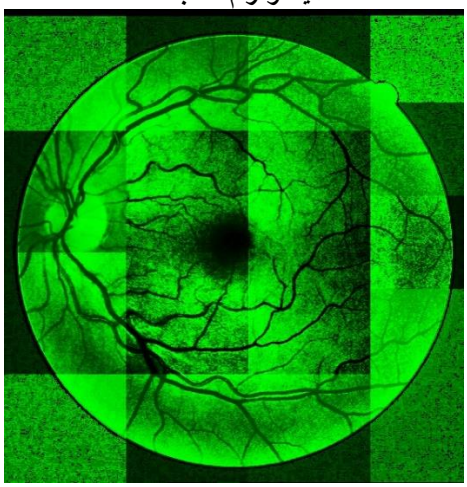
تصویر 47-تصویر eye1 پس از همسان سازی محلی هیستوگرام-تک بانده

تصویر 50-تصویر eye3 پس از همسان سازی محلی هیستوگرام-سه بانده

تصویر 48-تصویر eye2 پس از همسان سازی محلی هیستوگرام-سه بانده

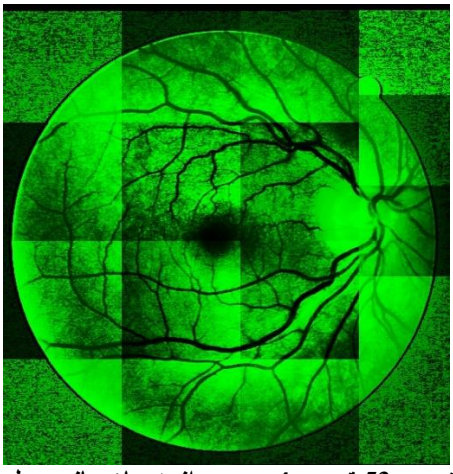


تصویر 51-تصویر eye3 پس از همسان سازی محلی هیستوگرام-تک بانده

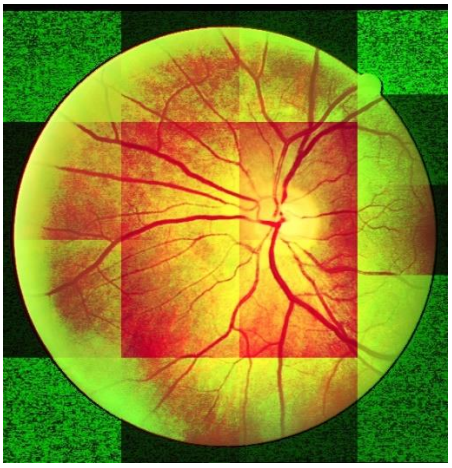


تصویر 49-تصویر eye2 پس از همسان سازی محلی هیستوگرام-تک بانده

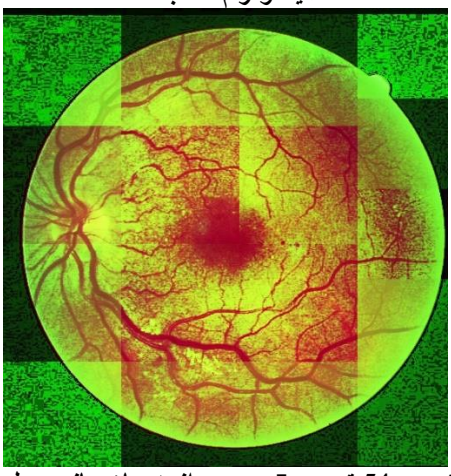
تصویر 52-تصویر eye4 پس از همسان سازی محلی هیستوگرام-سه بانده



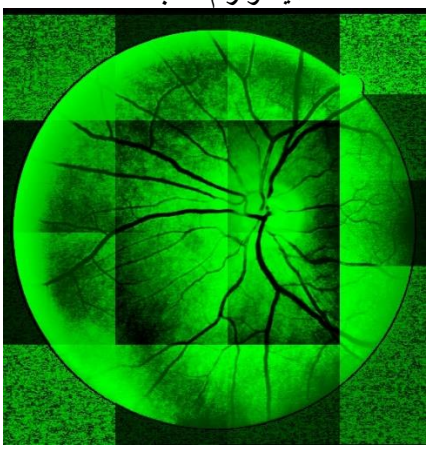
تصویر 53-تصویر eye4 پس از همسان سازی محلی هیستوگرام-تک بانده



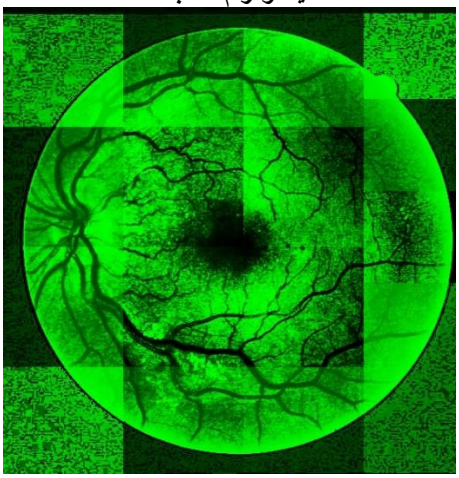
تصویر 56-تصویر eye6 پس از همسان سازی محلی هیستوگرام-سه بانده



تصویر 54-تصویر eye5 پس از همسان سازی محلی هیستوگرام-سه بانده

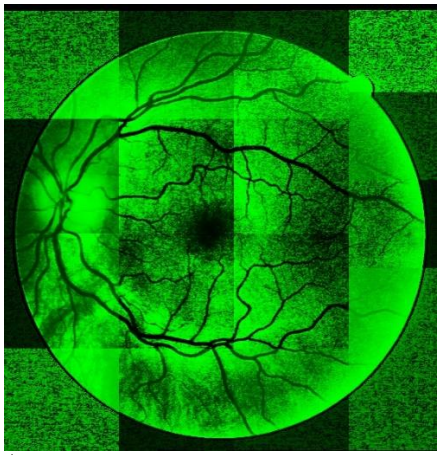


تصویر 57-تصویر eye6 پس از همسان سازی محلی هیستوگرام-تک بانده

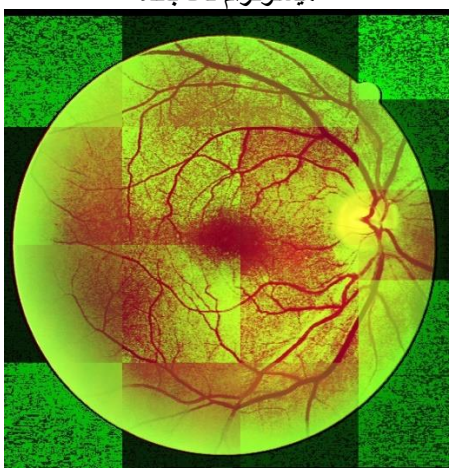


تصویر 55-تصویرeye5 پس از همسان سازی محلی هیستوگرام-تک بانده

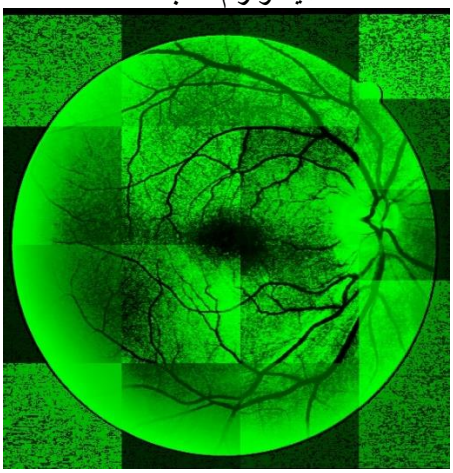
تصویر 58-تصویر eye7 پس از همسان سازی محلی هیستوگرام-سه بانده



تصویر 59-تصویر eye7 پس از همسان سازی محلی هیستوگرام-تک بانده

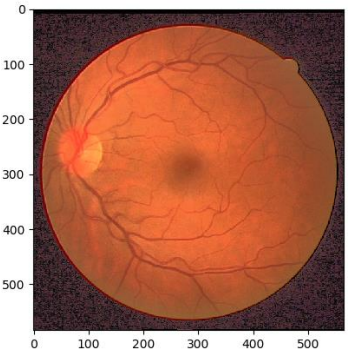


تصویر 60-تصویر eye8 پس از همسان سازی محلی هیستوگرام-سه بانده

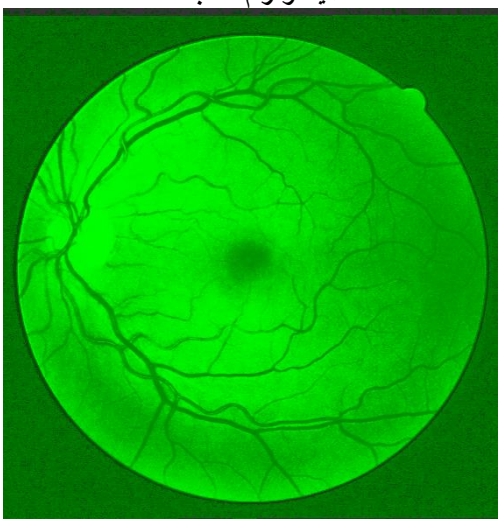


تصویر 61-تصویر eye8 پس از همسان سازی محلی هیستوگرام-تک بانده

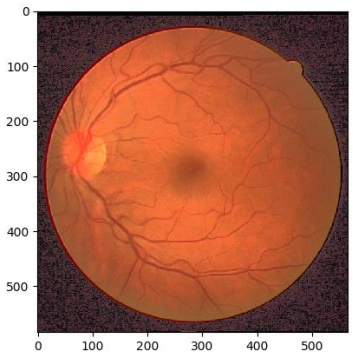
- همسان سازی پویای هیستوگرام



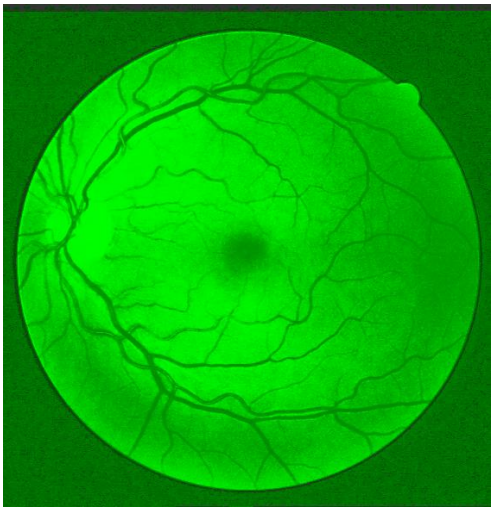
تصویر 62--تصویر eye1- پس از همسان سازی پویای هیستوگرام-سه بانده



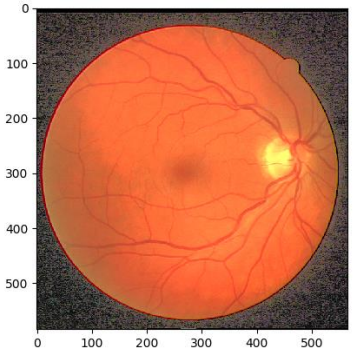
تصویر 63-تصویر eye1- پس از همسان سازی پویای هیستوگرام-تک بانده



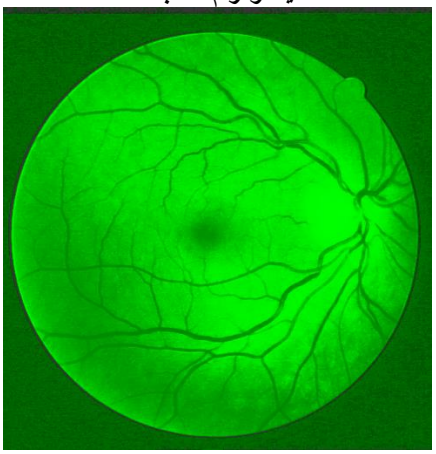
تصویر 64تصویر eye2- پس از همسان سازی پویای هیستوگرام-سه بانده



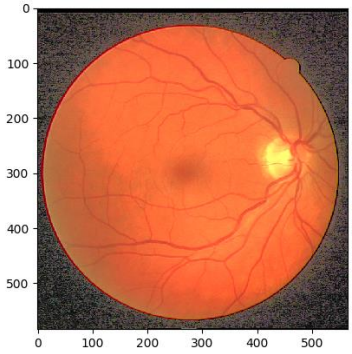
تصویر 65تصویر eye2- پس از همسان سازی پویای هیستوگرام-تک بانده



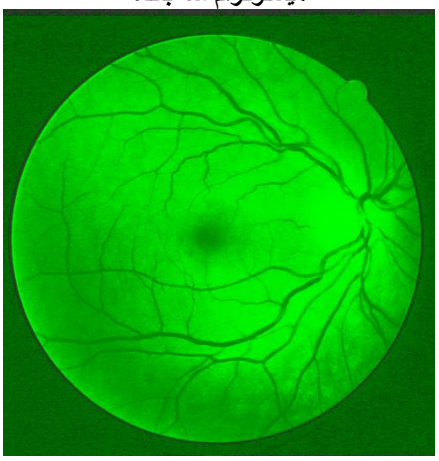
تصویر 66تصویر 3eye- پس از همسان سازی پویای هیستوگرام-سه بانده



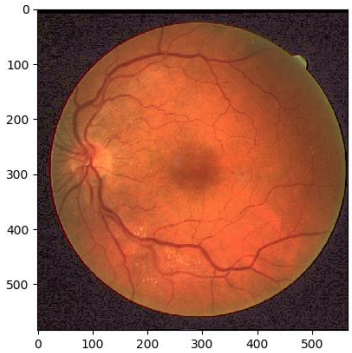
تصویر 67-تصویر eye3- پس از همسان سازی پویای هیستوگرام-تک بانده



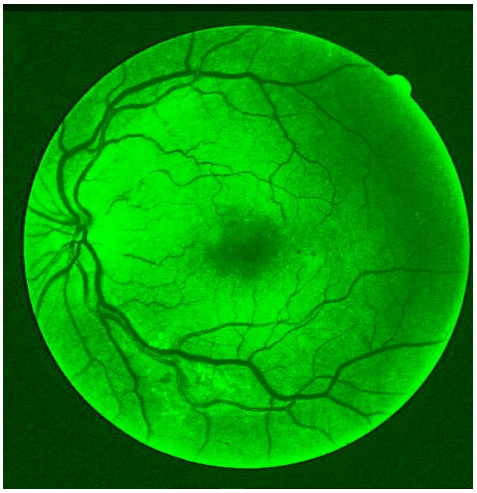
تصویر 68تصویر eye4- پس از همسان سازی پویای هیستوگرام-سه بانده



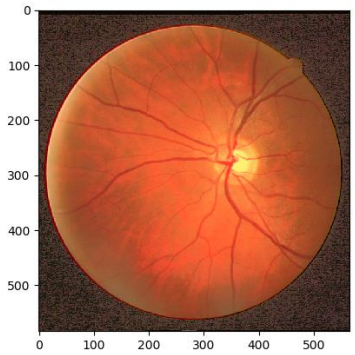
تصویر 69تصویر eye4- پس از همسان سازی پویای هیستوگرام-تک بانده



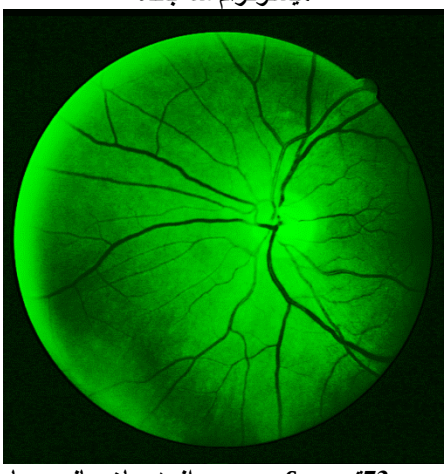
تصویر 70تصویر eye5- پس از همسان سازی پویای هیستوگرام-سه بانده



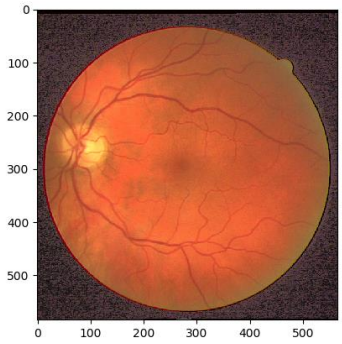
تصویر 71تصویر eye5- پس از همسان سازی پویای هیستوگرام-تک باندە



تصویر 72تصویر eye6- پس از همسان سازی پویای هیستوگرام-سه باندە

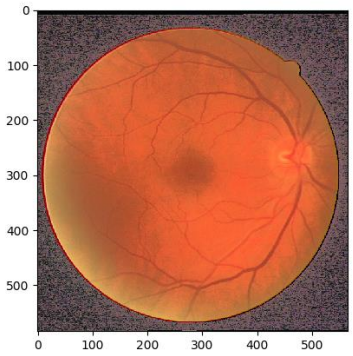


تصویر 73تصویر eye6- پس از همسان سازی پویای هیستوگرام-تک باندە

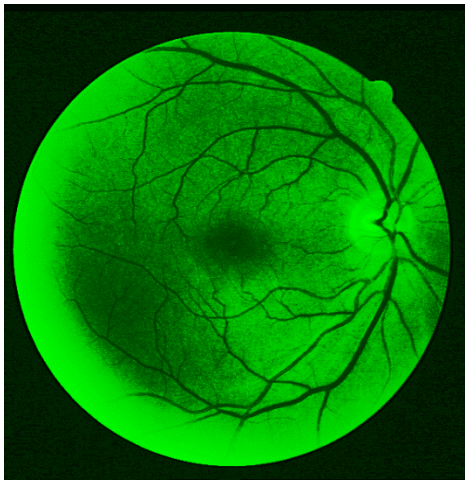


تصویر 74تصویر eye7- پس از همسان سازی پویای هیستوگرام-سه باندە

تصویر 75تصویر eye7- پس از همسان سازی پویای هیستوگرام-تک باندە



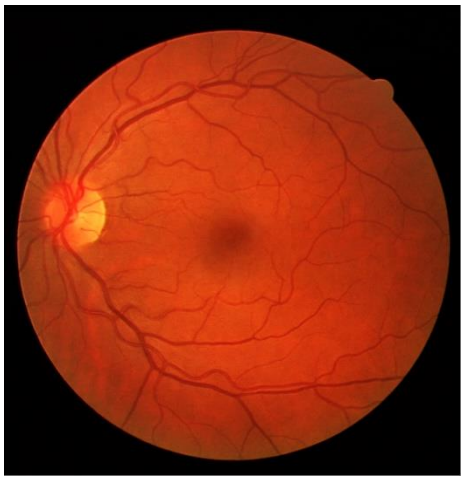
تصویر 76تصویر eye8- پس از همسان سازی پویای هیستوگرام-سه باندە



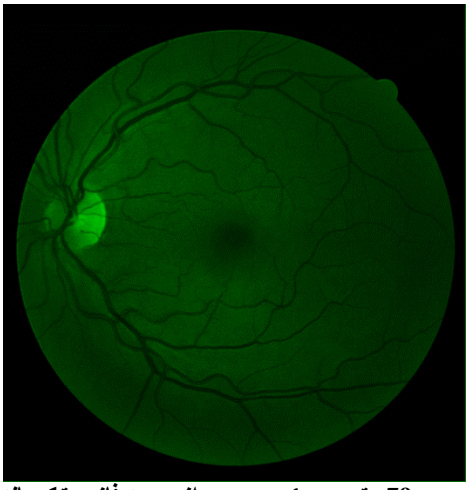
تصویر 77تصویر eye8- پس از همسان سازی پویای هیستوگرام-تک بانده

- **بهبود کنتراست به شیوه ی فازی**

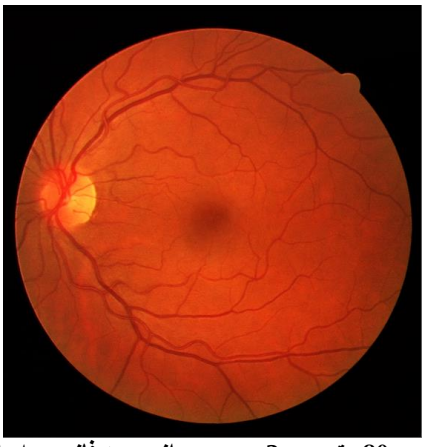
نتایج حاصل از این الگوریتم تا حد زیادی به پارامتر های الگوریتم بستگی دارد. با افزایش گاما، میانه افزایش می‌یابد و روشنایی تصویر نیز افزایش خواهد یافت. با افزایش ideal_variance، واریانس نیز افزایش می‌یابد. درنتیجه پارامتر لاندا نیز بیشتر خواهد شد کنتراست افزایش خواهد یافت.



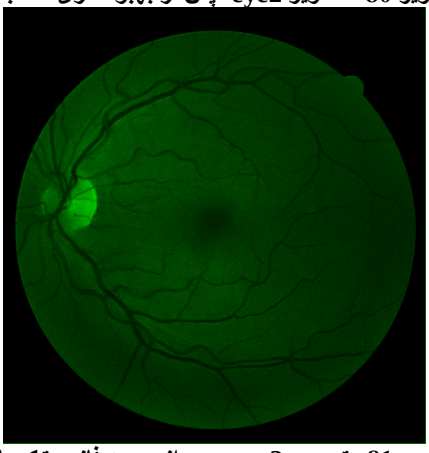
تصویر 78-تصویر eye1- پس از بهبود فازی-سه بانده



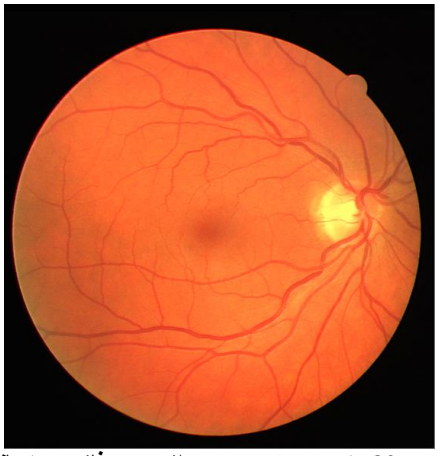
تصویر 79--تصویر eye1- پس از بهبود فازی-تک بانده



تصویر 80--تصویر eye2- پس از بهبود فازی-سه بانده



تصویر 81--تصویر eye2- پس از بهبود فازی-تک بانده



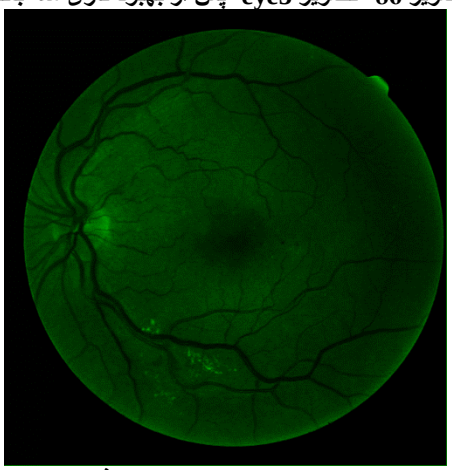
تصویر 82--تصویر eye3- پس از بهبود فازی-سه بانده

تصویر 85--تصویر eye4- پس از بهبود فازی-تک بانده

تصویر 83--تصویر eye3- پس از بهبود فازی-تک بانده

تصویر 86--تصویر eye5- پس از بهبود فازی-سه بانده

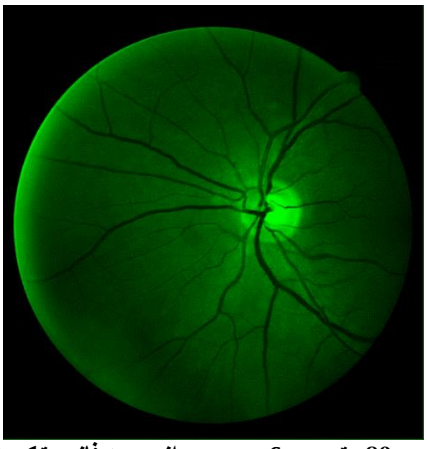
تصویر 84--تصویر eye4- پس از بهبود فازی-سه بانده



تصویر 87--تصویر eye5- پس از بهبود فازی-تک بانده

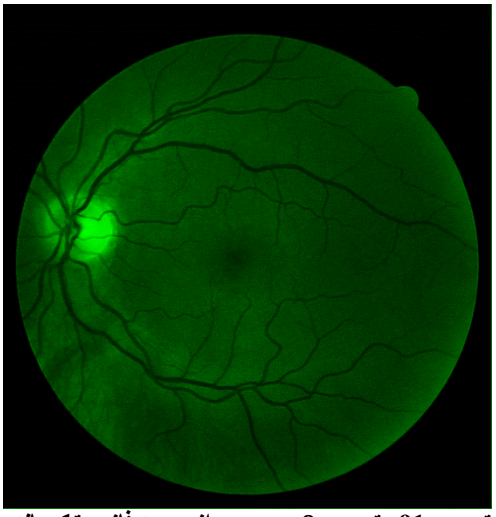
تصویر 88--تصویر eye6- پس از بهبود فازی-سه بانده

وزن دار بودن پیکسل ها در محاسبه‌ی پنجره ی فازی، در گوشه ی سمت راست و بالای شبکیه مشاهده می‌شود.



تصویر 89--تصویر eye6- پس از بهبود فازی-تک بانده

تصویر 90--تصویر eye7- پس از بهبود فازی-سه بانده



تصویر 91--تصویر eye8- پس از بهبود فازی-تک بانده

4- پیوست (کد برنامه)

Histogram Equalization

```
from __future__ import division
from PIL import Image
import cv2
from pylab import *

def
enhance_image_using_histogram_equaliz
ation(img,name):
    myim = img
    myim=cv2.cvtColor(myim,
cv2.COLOR_RGB2BGR) #convert the color
system
    blue,g,red = cv2.split(myim)
#split channels
    # cv2.imshow('RGB', myim) #show
the original image
    I = g.flatten()



    # calculate pdf
    h = dict()
    for p in I:
        # if not h.has_key(p):
        if p not in h:
            h[p]=1
        else:
            h[p]+=1

    # figure(figsize=(18, 10))
    # subplot()
    # plot(list(h.keys()),
list(h.values()), '.')
    #
plt.savefig('results/figures/'+str(na
me)+'-pdf.png')
```

```
nal')
    #
cv2.imshow('RGB',im_inhanced_3bands)
    #
cv2.imwrite('results/'+str(name)+'-
image_balanced_histogram_equalization
_3bands.jpg',im_inhanced_3bands)
    #
cv2.imwrite('results/'+str(name)+'-
image_balanced_histogram_equalization
_green.jpg',im_inhanced_green)
    # cv2.imshow('RBG',myim)


    # cv2.imshow('G-
RBG',im_inhanced_3scales)
    # cv2.imshow('G-
RGB',im_original_3scales)
    # figure(figsize=(48,16))
    #
subplot(121),hist(im_inhanced.flatten
(),256),title('Equalized', size=48)
    #
subplot(122),hist(I,256),title('Origi
nal', size=48)
    # # plt.show()
    #
plt.savefig('results/figures/'+str(na
me)+'-equalized-histogram-
standard.png')

    # waitforbuttonpress(0)
    return
im_inhanced_3bands,im_inhanced_green



if __name__ == '__main__':

img=array(Image.open("images/eye8.bmp
")) #replace the argument with the
desired image to process
    name="eye8"

enhance_image_using_histogram_equaliz
ation(img,name)
```

Local Histogram Equalization

```
from __future__ import division
from HistogramEqualization import
enhance_image_using_histogram_equaliz
ation
from PIL import Image
import cv2
from pylab import *

if __name__ == '__main__':
    img =
```

```
    h_sorted = sorted(h.items())

    #calculate cdf
    r=0
    cdf = list()
    for p in h_sorted:
        r += p[1]
        cdf.append([p[0], r])




    #calculate equalized pdf
    scale=255
    cdf_min = cdf[0][1]
    cdf_max = cdf[-1][1]
    d = cdf_max - cdf_min

    eq_v = list()
    eq_v_toPlot=list()
    for p in cdf:
        cdf_v = p[1]
        n = cdf_v - cdf_min
        h_v = round((n/d) * scale)
        eq_v.append([p[0], h_v])
        eq_v_toPlot.append(h_v)

    hv = dict(eq_v) #dictionaries are
easy in searching and indexing

    im_inhanced = g.copy()

    n_rows = len(im_inhanced) # the
image height
    row_width = len(im_inhanced[0])

    for row in range(n_rows):
        for vi in range(row_width):
            v = im_inhanced[row][vi]
            im_inhanced[row][vi] =
hv[v]
    I0 = np.zeros(myim.shape[:2],
np.uint8)
    im_inhanced_3bands =
np.dstack([blue,im_inhanced,red])
    im_inhanced_green =
np.dstack([I0,im_inhanced,I0])
    # im_original_3scales =
np.dstack([I0,myim,I0])
    #
im_inhanced_3scales=cv2.merge((red,im
_inhanced,blue))
    # fig, (ax1, ax2) =
subplots(1,2,figsize=(16,16))
    #
ax1.imshow(im_inhanced_3scales),ax1.s
et_title('Equalized')
    #
ax2.imshow(myim),ax2.set_title('Origi
```

```
        cv2.imshow('RGB',
result_image_3bands)
        cv2.imshow('RGB',
result_image_1band)

start_height=start_height+window_size
        start_width=0
    window_size=myim.shape[1]-
start_height
    for i in range(start_height,
myim.shape[1], window_size):
        for j in range(start_width,
height_limit * window_size,
window_size):
            slice =
myim[start_width:start_width +
window_size,
start_height:start_height +
window_size, ]
            slice3bands,slice1band =
enhance_image_using_histogram_equaliz
ation(slice,name)

result_image_3bands[start_width:start
_width + window_size,
start_height:start_height +
window_size, ] = slice3bands

result_image_1band[start_width:start_
width + window_size,
start_height:start_height +
window_size, ] = slice1band
            start_width = start_width
+ window_size
            cv2.imshow('RGB',
result_image_3bands)
            cv2.imshow('RGB',
result_image_1band)
        print(start_width)
        slice =
myim[start_width:myim.shape[0],
start_height:start_height +
window_size, ]
        slice3bands,slice1band =
enhance_image_using_histogram_equaliz
ation(slice,name)

result_image_3bands[start_width:myim.
shape[0], start_height:start_height +
window_size, ] = slice3bands

result_image_1band[start_width:myim.s
hape[0], start_height:start_height +
window_size, ] = slice1band
        cv2.imshow('RGB',
result_image_3bands)
        cv2.imshow('RGB',
result_image_1band)
```

```
array(Image.open("images/eye8.bmp"))
    name="eye8"
    # myim = cv2.cvtColor(img,
cv2.COLOR_RGB2BGR)
    myim=img
    result_image_3bands=myim.copy()
    result_image_1band=myim.copy()
    window_size=150
    start_height=0
    start_width=0
    end=window_size


width_limit=int(img.shape[0]/window_s
ize)

height_limit=int(img.shape[1]/window_
size)


    for i in
range(start_height,width_limit*window
_size,window_size):
        for j in
range(start_width,height_limit*window
_size,window_size):

slice=myim[start_width:start_width+wi
ndow_size,start_height:start_height+w
indow_size,]
            slice3bands,
slice1band=enhance_image_using_histog
ram_equalization(slice,name)

result_image_3bands[start_width:start
_width+window_size,start_height:start
_height+window_size,]=slice3bands

result_image_1band[start_width:start_
width+window_size,start_height:start_
height+window_size,]=slice1band

start_width=start_width+window_size
        print(start_width)
        slice =
myim[start_width:myim.shape[0],
start_height:start_height +
window_size, ]
        slice3bands,slice1band =
enhance_image_using_histogram_equaliz
ation(slice,name)

result_image_3bands[start_width:myim.
shape[0], start_height:start_height +
window_size, ] = slice3bands

result_image_1band[start_width:myim.s
hape[0], start_height:start_height +
window_size, ] = slice1band
```

```
    dIv[dIv == 0] = 0.00001
    dI = np.sqrt(dIh ** 2 + dIv **
2).astype(np.uint32)
    di = dI[2:hei + 2, 2:wid + 2]

    dSh = scipy.signal.convolve2d(S,
np.rot90(fh, 2), mode='same')
    dSv = scipy.signal.convolve2d(S,
np.rot90(fv, 2), mode='same')
    dSh[dSh == 0] = 0.00001
    dSv[dSv == 0] = 0.00001
    dS = np.sqrt(dSh ** 2 + dSv **
2).astype(np.uint32)
    ds = dS[2:hei + 2, 2:wid + 2]

    h = H[2:hei + 2, 2:wid + 2]
    s = S[2:hei + 2, 2:wid + 2]
    i = I[2:hei + 2, 2:wid +
2].astype(np.uint8)

    Imean =
scipy.signal.convolve2d(I,
np.ones((5, 5)) / 25, mode='same')
    Smean =
scipy.signal.convolve2d(S,
np.ones((5, 5)) / 25, mode='same')

    Rho = np.zeros((hei + 4, wid +
4))
    for p in range(2, hei + 2):
        for q in range(2, wid + 2):
            tmpi = I[p - 2:p + 3, q -
2:q + 3]
            tmps = S[p - 2:p + 3, q -
2:q + 3]
            corre =
np.corrcoef(tmpi.flatten('F'),
tmps.flatten('F'))
            Rho[p, q] = corre[0, 1]

    rho = np.abs(Rho[2:hei + 2, 2:wid
+ 2])
    rho[np.isnan(rho)] = 0
    rd = (rho * ds).astype(np.uint32)
    Hist_I = np.zeros((256, 1))
    Hist_S = np.zeros((256, 1))

    for n in range(0, 255):
        temp = np.zeros(di.shape)
        temp[i == n] = di[i == n]
        Hist_I[n + 1] =
np.sum(temp.flatten('F'))
        temp = np.zeros(di.shape)
        temp[i == n] = rd[i == n]
        Hist_S[n + 1] =
np.sum(temp.flatten('F'))

    return Hist_I, Hist_S
```

```
cv2.imshow('RGB',result_image_3bands)

cv2.imshow('RGB',result_image_1band)

cv2.imwrite('results/'+str(name)+'-
local-histogram-equalization-
3bands.jpg',result_image_3bands)

cv2.imwrite('results/'+str(name)+'-
local-histogram-equalization-
1band.jpg',result_image_1band)
    waitforbuttonpress(0)
```

Dynamic Histogram Equalization

```
from __future__ import division
import matplotlib.colors
import scipy, scipy.misc,
scipy.signal
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import cv2
from pylab import *


def build_is_hist(img):
    hei = img.shape[0]
    wid = img.shape[1]
    ch = img.shape[2]
    Img = np.zeros((hei + 4, wid + 4,
ch))
    for i in range(ch):
        Img[:, :, i] = np.pad(img[:,
:, i], (2, 2), 'edge')
    hsv =
(matplotlib.colors.rgb_to_hsv(Img))
    hsv[:, :, 0] = hsv[:, :, 0] * 255
    hsv[:, :, 1] = hsv[:, :, 1] * 255
    hsv[hsv > 255] = 255
    hsv[hsv < 0] = 0
    hsv =
hsv.astype(np.uint8).astype(np.float6
4)
    fh = np.array([[-1.0, 0.0, 1.0],
[-2.0, 0.0, 2.0], [-1.0, 0.0, 1.0]])
    fv = fh.conj().T

    H = hsv[:, :, 0]
    S = hsv[:, :, 1]
    I = hsv[:, :, 2]

    dIh = scipy.signal.convolve2d(I,
np.rot90(fh, 2), mode='same')
    dIv = scipy.signal.convolve2d(I,
np.rot90(fv, 2), mode='same')
    dIh[dIh == 0] = 0.00001
```

```
    plt.show()


if __name__ == '__main__':
    main()
```

Fuzzy Contrast Enhancement

```
import cv2 as cv
import numpy as np
import math
import time
from matplotlib import pyplot as plt
n = 3# number of rows (windows on
columns)
m = 3 # number of colomns (windows on
rows)
EPSILON = 0.00001
#GAMMA, IDEAL_VARIANCE 'maybe' have
to changed from image to another
GAMMA = 0.3 # Big GAMMA >> Big mean
>> More Brightness
IDEAL_VARIANCE = 0.95 #Big value >>
Big variance >> Big lamda >> more
contrast
img_name = 'test.jpg'
img = cv.imread(img_name)
#img = cv.resize(img, (200, 200))
layer = cv.cvtColor(img,
cv.COLOR_BGR2GRAY)
WIDTH = layer.shape[1]
HEIGHT = layer.shape[0]
x0, x1, y0, y1 = 0, WIDTH - 1, 0,
HEIGHT - 1


# split the image to windows
def phy(value):  # phy: E --> R
    # if ((1+value)/((1-
value)+0.0001)) < 0:
    # print(value)
    return 0.5 * np.log((1 + value) /
((1 - value) + EPSILON))


def multiplication(value1, value2):
# ExE --> R
    return phy(value1) * phy(value2)


def norm(value):
    return abs(phy(value))


def scalar_multiplication(scalar,
value):  # value in E ([-1,1])
    s = (1 + value) ** scalar
    z = (1 - value) ** scalar
    res = (s - z) / (s + z + EPSILON)
```

```
def dhe(img, alpha=0.5):
    hist_i, hist_s =
build_is_hist(img)
    hist_c = alpha * hist_s + (1 -
alpha) * hist_i
    hist_sum = np.sum(hist_c)
    hist_cum = hist_c.cumsum(axis=0)

    hsv =
matplotlib.colors.rgb_to_hsv(img)
    h = hsv[:, :, 0]
    s = hsv[:, :, 1]
    i = hsv[:, :, 2].astype(np.uint8)

    c = hist_cum / hist_sum
    s_r = (c * 255)
    i_s = np.zeros(i.shape)
    for n in range(0, 255):
        i_s[i == n] = s_r[n + 1] /
255.0
    i_s[i == 255] = 1
    hsi_o = np.stack((h, s, i_s),
axis=2)
    result =
matplotlib.colors.hsv_to_rgb(hsi_o)

    result = result * 255
    result[result > 255] = 255
    result[result < 0] = 0
    return result.astype(np.uint8)


def main():
    name="eye1"
    # img = cv2.imread(img_name)

img=array(Image.open("images/eye1.bmp
"))
    # myim = cv2.cvtColor(img,
cv2.COLOR_RGB2BGR)
    blue, g, red = cv2.split(img)
    I0 = np.zeros(img.shape[:2],
np.uint8)
    img_1band=cv2.merge((I0,g,I0))
    result = dhe(img)
    result_1band=dhe(img_1band)
    #
cv2.imwrite('results/'+str(name)+'-
dynamic-histogram-equalization-
3bands.png',result)

cv2.imwrite('results/'+str(name)+'-
dynamic-histogram-equalization-
1band.png',result_1band)
    plt.imshow(result)
    plt.savefig('results/' +
str(name) + '-dynamic-histogram-
equalization-3bands.png')
```

```
def mapping(img, source, dest):
    return (dest[1] - dest[0]) *
((img - source[0]) / (source[1] -
source[0])) + dest[0]



e_layer_gray = mapping(layer, (0,
255), (-1, 1))

def cal_ps_ws(m, n, w, h, gamma):
    ps = np.zeros((m, n, w, h))
    for i in range(m):
        for j in range(n):
            for k in range(w):
                for l in range(h):
                    ps[i, j, k, l] =
p(i, j, k, l)

    ws = np.zeros((m, n, w, h))
    for i in range(m):
        for j in range(n):
            ps_power_gamma =
np.power(ps[i, j], gamma)
            for k in range(w):
                for l in range(h):
                    ws[i, j, k, l] =
ps_power_gamma[k, l] / (np.sum(ps[:,
:, k, l])+EPSILON)
    return ps, ws
print('Ps and Ws calculation is in
progress...')
start = time.time()
ps, ws = cal_ps_ws(m, n, WIDTH,
HEIGHT, GAMMA)
end = time.time()
print('Ps and Ws calculation has
completed successfully in '+str(end-
start)+' s')


def cal_means_variances_lamdas(w,
e_layer):
    means = np.zeros((m, n))
    variances = np.zeros((m, n))
    lamdas = np.zeros((m, n))
    taos = np.zeros((m, n))

    def window_card(w):
        return np.sum(w)

    def window_mean(w, i, j):
        mean = 0
        for k in range(HEIGHT):
            for l in range(WIDTH):
                mean = addition(mean,
scalar_multiplication(w[i, j, l, k],
    return res


def addition(value1, value2):  #
value1,value2 are in E ([-1,1])
    res = (value1 + value2) / (1 +
(value1 * value2) + EPSILON)
    return res


def subtract(value1, value2):  #
value1,value2 are in E ([-1,1])
    res = (value1 - value2) / (1 -
(value1 * value2) + EPSILON)
    return res


def C(m, i):
    return math.factorial(m) /
((math.factorial(i) *
math.factorial(m - i)) + EPSILON)


def qx(i, x):  # i: window index in
rows, x: number of current pixel on
x-axis
    if (x == WIDTH - 1):
        return 0
    return C(m, i) * (np.power((x -
x0) / (x1 - x), i) * np.power((x1 -
x) / (x1 - x0),

m))  # This is the seconf
implementation
    # return C(m,i)*((np.power(x-
x0,i) * np.power(x1-x,m-i)) /
(np.power(x1-x0,m)+EPSILON))


def qy(j, y):
    '''
    The second implementation for the
formula does not go into overflow.
    '''
    if (y == HEIGHT - 1):
        return 0
    return C(n, j) * (np.power((y -
y0) / (y1 - y), j) * np.power((y1 -
y) / (y1 - y0),

n))  # This is the seconf
implementation
    # return C(n,j)*((np.power((y-
y0),j) * np.power((y1-y),n-j))/
(np.power(y1-y0,n)+EPSILON))


def p(i, j, x, y):
    return qx(i, x) * qy(j, y)
```

```
        for j in range(n):
            win = window_enh(w, i, j,
e_layer)
            w1 = w[i, j].T.copy()
            for k in range(width):
                for l in
range(height):
                    new_image[l, k] =
addition(new_image[l, k],
scalar_multiplication(w1[l, k],
win[l, k]))
    return new_image
def one_layer_enhacement(e_layer):
    #card_image =
layer.shape[0]*layer.shape[1]
    new_E_image = image_enh(ws,
e_layer)
    res_image = mapping(new_E_image,
(-1, 1), (0, 255))
    res_image = np.round(res_image)
    res_image =
res_image.astype(np.uint8)
    return res_image

res_img =
one_layer_enhacement(e_layer_gray)

plt.subplot(1, 2, 1)
plt.imshow(img, cmap = 'gray')
plt.subplot(1, 2, 2)
plt.imshow(res_img, cmap = 'gray')
plt.title('Fuzzy Grayscale image
enhacement.')
plt.show()



#Constants
n = 2 # number of rows (windows)
m = 2 # number of colomns (windows)
GAMMA = 1
EPSILON = 0.00001
IDEAL_VARIANCE = 3

#Call image
img_name = 'images/eye8.bmp'
name="eye8"
img = cv.imread(img_name)
WIDTH = img.shape[1]
HEIGHT = img.shape[0]
x0, x1, y0, y1 = 0, WIDTH - 1, 0,
HEIGHT - 1

#Image fuzzification
layer_b, layer_g, layer_r =
cv.split(img)
e_layer_b, e_layer_g, e_layer_r =
mapping(layer_b, (0, 255), (-1, 1)),
```

```
e_layer[k, l]))
        mean /= window_card(w[i, j])
        return mean

    def window_variance(w, i, j):
        variance = 0
        for k in range(HEIGHT):
            for l in range(WIDTH):
                variance += w[i, j,
l, k] *
np.power(norm(subtract(e_layer[k, l],
means[i, j])), 2)
        variance /= window_card(w[i,
j])
        return variance

    def window_lamda(w, i, j):
        return
np.sqrt(IDEAL_VARIANCE) /
(np.sqrt(variances[i, j]) + EPSILON)

    def window_tao(w, i, j):
        return window_mean(w, i, j)

    for i in range(m):
        for j in range(n):
            means[i, j] =
window_mean(ws, i, j)
            variances[i, j] =
window_variance(ws, i, j)
            lamdas[i, j] =
window_lamda(ws, i, j)
    taos = means.copy()

    return means, variances, lamdas,
taos


print('means, variances, lamdas and
taos calculation is in progress...')
start = time.time()
means, variances, lamdas, taos =
cal_means_variances_lamdas(ws,
e_layer_gray)
end = time.time()
print('means, variances, lamdas and
taos calculation is finished in ' +
str(end - start) + ' s')
def window_enh(w, i, j, e_layer):
    return
scalar_multiplication(lamdas[i, j],
subtract(e_layer, taos[i, j]))

def image_enh(w, e_layer):
    new_image =
np.zeros(e_layer.shape)
    width = e_layer.shape[1]
    height = e_layer.shape[0]
    for i in range(m):
```

```
mapping(layer_g, (0, 255), (-1, 1)),
mapping(layer_r, (0, 255), (-1, 1))
e_layer_rgb =
scalar_multiplication(1/3,
addition(addition(e_layer_b,
e_layer_g), e_layer_r)) #Mean of the
three layers

#Cal Ps, Ws
print("calculate ps and ws for color
image")
ps, ws = cal_ps_ws(m, n, WIDTH,
HEIGHT, GAMMA)

#Cal means, variances, lamdas, taos
print("calculate
means,variance,lamdas,taos for color
image")
means, variances, lamdas, taos =
cal_means_variances_lamdas(ws,
e_layer_rgb)

#Layers enhacement
print("enhancing bands")
res_r =
one_layer_enhacement(e_layer_r)
res_g =
one_layer_enhacement(e_layer_g)
res_b =
one_layer_enhacement(e_layer_b)
I0 = np.zeros(img.shape[:2],
np.uint8)
res_img_3bands = cv.merge([res_b,
res_g, res_r])
res_img_1band = cv.merge([I0, res_g,
I0])

# print("plotting results")
# plt.subplot(1, 2, 1)
# plt.imshow(cv.cvtColor(img,
cv.COLOR_BGR2RGB))
# plt.subplot(1, 2, 2)
# plt.imshow(cv.cvtColor(res_img,
cv.COLOR_BGR2RGB))
# plt.title('Fuzzy RGB image
enhacement.')
# plt.show()
cv.imwrite('results/'+name+'-fuzzy-
3band.png',res_img_3bands)
cv.imwrite('results/'+name+'-fuzzy-
1band.png',res_img_1band)
```