



دانشگاه صنعتی امیرکبیر
(بی تکنیک تهران)

پاسخ تمرین اول تصویر پردازی رقمی

استاد درس جناب آقای دکتر رحمتی

نیمسال دوم سال تحصیلی ۱۴۰۰-۱۴۰۱

محسن عبادپور

شماره دانشجویی: ۴۰۰۱۳۱۰۸۰

ایمیل: m.ebadpour@aut.ac.ir

فهرست پاسخ ها

۱	مسئله ۱ قسمت (a)
۲	مسئله ۱ قسمت (b)
۲	مسئله ۱ قسمت (c)
۴	مسئله ۲ قسمت (a)
۴	مسئله ۲ قسمت (b) و (c)
۶	مسئله ۳ قسمت (a)
۷	مسئله ۳ قسمت (b) و (c)
۸	مسئله ۳ قسمت (d)
۹	مسئله ۳ قسمت (e)
۱۱	مسئله ۴ قسمت (a)
۱۲	مسئله ۴ قسمت (b)
۱۳	مسئله ۴ قسمت (c)
۱۴	مسئله ۵ قسمت (a)
۱۵	مسئله ۵ قسمت (b)
۱۶	مسئله ۵ قسمت (c)
۱۷	مسئله ۵ قسمت (d)
۱۷	مسئله ۵ قسمت (e)

مسئله اول

مسئله ۱ قسمت (a)

در پاسخ به این قسمت از سوال، تابع خواسته شده به همان فرمت پیاده سازی شده و در سورس فایل قرار دارد. (قسمت مربوطه در داخل فایل ژوپیتر title شده است). طبق خواسته شده از توابع آماده برای عملیات های ماتریسی آماده استفاده نشده و سطر های تصاویر یک به یک داخل حلقه پیمایش شده و پیکسل های هر کدام از راست به چپ جا به جا شده (راست ترین با چپ ترین) و تصویر آینه شده حاصل می شود.

مسئله ۱ قسمت (b)

برای حل این قسمت، یک تابع با نام MirrorHandler پیاده سازی شده است که وظیفه آن گرفتن یک تصویر و محاسبه سه حالت ممکن برای آینه تصاویر (ذکر شده در متن سوال) می باشد. ابتدا از ستون وسط تصویر آن را به دو نیم کرده و آینه هر کدام را جداگانه محاسبه می کنیم (left_mirror, right_mirror); سپس با استفاده از آینه های به وجود آمده، سه حالت خواسته شده را تولید کرده و آنها را به عنوان نتیجه برمی گردانیم. تصاویر حاصل علاوه بر نمایش در صفحه بعد، همگی در فolder outputs قرار داده شده است.

مسئله ۱ قسمت (c)

دو متغیر خواسته شده برای ارزیابی تصاویر حاصل به صورت کامل و از پایه و در فرمت خواسته شده پیاده سازی شده است که در سورس پیوستی قابل مشاهده است. برای صحت سنجی پیاده سازی انجام شده از PSNR موجود در Open-CV بهره گرفته و نتایج دقیقا یکسان بود. منطق و رابطه PSNR ساده بود و با یک جست و جوی ساده بصورت کامل متوجه شده و اقدام به پیاده سازی کردم اما برای SSIM مورد کمی فرق داشت چرا که متوجه شدن نیاز به مطالعه بیشتر داشت. برای مطالعه و متوجه شدن تئوری از این [لینک](#) و برای پیاده سازی نیز از این [لینک](#) بهره گرفتم. از جایی که با در نظر گرفتن مقادیر متفاوت برای ضرایب ثابت موجود در SSIM می توان به نتایج عددی گوناگون رسید، بنده ضرایب پیشنهاد شده در پیاده سازی اشاره شده در لینک بالا را تغییر نداده و به آن پسندید کردم چرا که در هر صورت هدف ما مقایسه نسبت به تصاویر دیگر است و تغییر ضرایب این نسبت را دجالش چالش نمی کند. در قسمت عنوان تصاویر صفحه بعد مقادیر هر کدام گزارش شده است.



Erling Haaland | LL: PSNR=20.76108, SSIM=0.87641 | RR: PSNR=20.76108, SSIM=0.87639 | RL: PSNR=17.75078, SSIM=0.75258



Harry Kane | LL: PSNR=22.90733, SSIM=0.90165 | RR: PSNR=22.90733, SSIM=0.90165 | RL: PSNR=19.89703, SSIM=0.80293



Jose Mourinho | LL: PSNR=15.60517, SSIM=0.8347 | RR: PSNR=15.60517, SSIM=0.83468 | RL: PSNR=12.59487, SSIM=0.66909



Jurgen Klopp | LL: PSNR=19.80453, SSIM=0.84976 | RR: PSNR=19.80453, SSIM=0.84974 | RL: PSNR=16.79423, SSIM=0.69905



Kevin De Bruyne | LL: PSNR=23.82639, SSIM=0.87753 | RR: PSNR=23.82639, SSIM=0.87752 | RL: PSNR=20.81609, SSIM=0.75478



Lionel Messi | LL: PSNR=19.93349, SSIM=0.8708 | RR: PSNR=19.93349, SSIM=0.87082 | RL: PSNR=16.92319, SSIM=0.74127



Ngolo Kante | LL: PSNR=20.39897, SSIM=0.87588 | RR: PSNR=20.39897, SSIM=0.87591 | RL: PSNR=17.38867, SSIM=0.75147



Pep Guardiola | LL: PSNR=20.32388, SSIM=0.87855 | RR: PSNR=20.32388, SSIM=0.87856 | RL: PSNR=17.31358, SSIM=0.75671



Zlatan Ibrahimovic | LL: PSNR=19.70987, SSIM=0.88266 | RR: PSNR=19.70987, SSIM=0.88265 | RL: PSNR=16.69957, SSIM=0.76511



شکل ۱ - خروجی مربوط به سوال اول - قرینگی چهره

تصویر مقابل خروجی مربوط به کل این سوال می باشد. به ترتیب از چپ به راست برای هر فرد تصویر اورجینال(به همراه مرز قرینه)، قرینه L (قرینه چپ)، قرینه R (قرینه راست) و قرینه L (تمام قرینه) قرار گرفته است.

به ازای هر کدام از این قرینه ها و به ازای هر یک از معیار های سنجش PSNR و SSIM می توان قرینه ترین فرد را انتخاب کرد؛ از این رو، جدول زیر برای مقایسه بهتر آماده شده است:

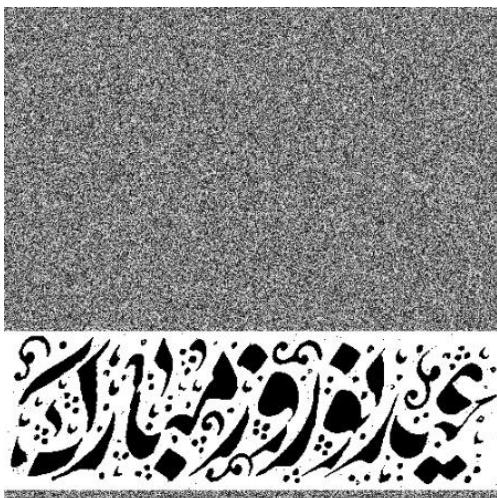
تمام قرینه	قرینه چپ	قرینه راست	Kevin
SSIM			Kevin
Harry	Harry	Harry	Harry

با توجه به نتایج بدست آمده می توان گفت Kevin De Bruyne متقارن ترین چهره با توجه به معیار PSNR را داراست و همچنین نیز با توجه به معیار SSIM متقارن ترین چهره را دارد. Harry Kane

مسئله ۲ قسمت دوم

مسئله ۲ قسمت a)

برای حل این قسمت یک تابع با نام OffSetGenerator پیاده سازی شد که دو تصویر ورودی را دریافت کرده و آن دو را از مرز افقی و درجهت عمود روی هم عبور داده و تفاوت پیکسل های مشترک را محاسبه کرده و تصویر حاصل را ذخیره می کند. عرض تصاویر داده شده ۵۰۰ پیکسل می باشد پس در کل به تعداد $500 * 2 = 1000$ حالت ممکن برای غلطاندن تصاویر روی هم وجود خواهد داشت. بنده تمام های ممکن را در سیستم خود ذخیره کرده و دنبال پیام مخفی شده جست و جو کردم که نتیجه این شد که در یکی از حالت ها یک



شکل ۲ - خروجی مربوط به قسمت ۵ سوال دوم - پیام مخفی شده

پیغام "عید نوروز مبارک" گنجانده شده بود که در مقابل آمده است.(در سایر حالت ها تصاویر کاملاً نوبزی بود)

از جایی که حجم همه‌ی ۱۰۰۰ تصویر تولیدی بسیار بالا بود، در فolder outputs صرفا ۵ حالت قبل و ۵ حالت بعد از پیام "عید نوروز مبارک" قرار داده شده است تا سیر غلطاندن قابل نمایش باشد.(در صورتی که شرط موجود برای ذخیره عکس در سورس کد حذف شود، همه‌ی ۱۰۰۰ تصویر مجدد در همان فolder تولید و قابل مشاهده خواهد شد)

مسئله ۲ قسمت b و c)

برای پاسخ گویی به این دو بخش از سوال، دو تابع با عناوین Encode و Decode پیاده سازی شده است. برای مقادیر باینری مورد استفاده برای روشن یا خاموش کردن پیکسل ها شامل از مقدار ۲۵۵ و ۰ استفاده شده است. همچنین از جایی که تصویر پیام با تصویر پایه(تصویری که می‌خواهیم در آن پیام را مخفی کنیم) هم سایز نیستند، ابتدا تصویر پیام را Resize کرده ایم. نهایی به صورت زیر آمده شده است:



شکل ۳ - قسمت ۳ سوال سوم - تصویر آمده برای پنهان سازی

تابع Decode پیاده سازی معادل با الگوریتم نوشته شده در متن سوال می باشد و هیچ نکته ای اضافه یا کسر نشده است؛ تابع Encode نیز برای پنهان نمودن تصویر پیام در تصویر پایه می باشد که الگوریتم آن تقریباً معکوس الگوریتم Decode می باشد. هدف الگوریتم این بوده است که تصویر تک کاناله که حاصل تفاوت کانال قرمز و آبی (img_mid) می باشد ذخیره نماید. از جایی که تصویر پیام ما نیز باینری (پیام) را در یک تصویر تک کاناله که حاصل تفاوت کانال قرمز و آبی (img_mid) می باشد decode ابتدا پس از قدر مطلق گرفتن از تفاوت دو کانال قرمز و آبی، از باقیمانده بر دو برای باینری کردن مقادیر و استخراج پیام استفاده می کند.

حال برای Encode نیز معکوس آن را پیاده سازی کرده ایم؛ ابتدا کانال img_mid را بدست آورده و سپس شروع به پردازش پیکسل ها می کنیم. اگر شرط باقیماندگی بر دو پیکسل از تصویر پیام و تصویر پایه درست بود (طبق الگوریتم) به این معناست که در زمان استخراج پیام، پیکسل به درستی بازگردانده می شود پس نیاز به تغییر در زمان Encode نیست. حال اگر شرط باقیماندگی درست نباشد، بایستی یک واحد در کanal img_mid تغییر یابد تا آن پیکسل در زمان Decode بدرستی یافت شود. (اضافه یا کسر کردن یک واحد، باقیماندگی دو را تغییر می دهد و تفاوتی ندارد کدام باشد)

برای تغییر دادن یک واحدی مقدار img_mid که حاصل قدر مطلق تفاوت دو کانال آبی و قرمز است، می توان با تغییر یک واحدی یکی از این کانال ها در آن پیکسل به این هدف رسید که در پیاده سازی انجام شده آن تغییر در کانال قرمز اتفاق می افتد (می تواند آبی یا حتی تصادفی باشد).

تصویر Decode و Encode شده (Decode بر اساس تصویر Encode) بصورت زیر بدست آمده است (در پوشه outputs با سایز اصلی قرار دارد) که قابل مشاهده است فرآیند پنهان سازی و آشکار سازی پیام در تصویر پایه بصورت کامل و دقیقاً انجام شده است:



شکل ۴ - قسمت b,c سوال دوم - تصویر پیام Decode شده از تصویر Encode

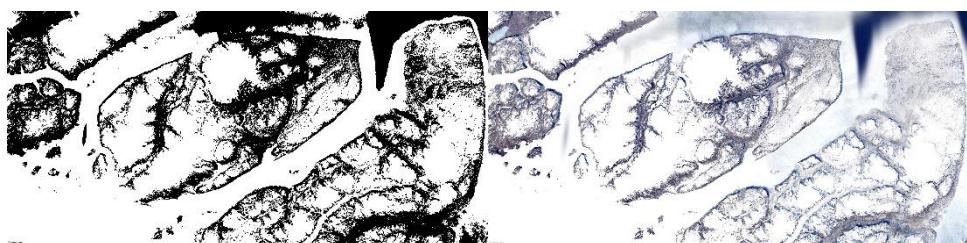


شکل ۵ - قسمت c سوال دوم - تصویر شده پیام در تصویر

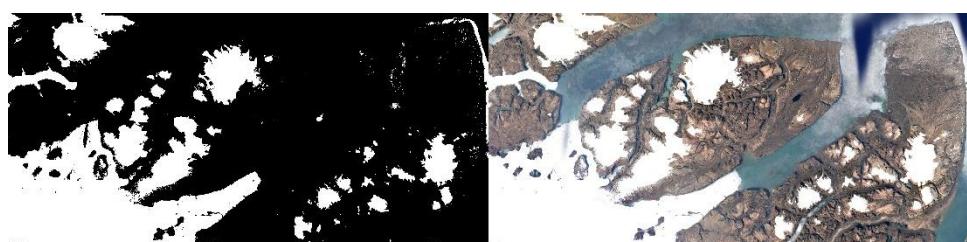
مسئله سوم

مسئله ۳ قسمت (a)

برای استخراج ناحیه های یخی در تصاویر جزیره Mylius-Erichsen بایستی پیکسل های یخی را استخراج نمود و سپس درصد تغییرات یخ ذوب شده محاسبه نمود. scale و سایز تصویر برداری شده یکسان و برابر با ۱۰ کیلومتر است و عمق یخ ها هم یکسان در نظر گرفته شده است(در متن سوال اشاره ای به متغیر بودن عمق یخ نشده است؛ از این رو پیکسل های یخی را باید شمارش کرد و نسبت درصد به یکدیگر را محاسبه نمود. برای استخراج پیکسل های یخی که رنگ سفیدی دارند باید پیکسل های با مقادیر نزدیک به ۲۵۵ در سه کانال RGB را فیلتر کرده و تعداد آنها را محاسبه کرد. Threshold برای تعیین پیکسل های یخی مقدار ۲۱۵ در نظر گرفته شده است. پیکسل های یخی استخراج شده برای هر کدام از سال های ۲۰۲۰ و ۲۰۰۰ در زیر قابل مشاهده است:



شکل ۶- قسمت ۵ سوال ۳ - یخ استخراجی سال ۲۰۰۰



شکل ۷- قسمت ۵ سوال ۳ - یخ استخراجی سال ۲۰۲۰

طبق نسبت پیکسل های یخی در سال ۲۰۲۰ نسبت به ۲۰۰۰ تقریباً معادل با $60/52\%$ یخ ها ذوب شده است.

مسئله ۳ قسمت b و c

برای پاسخ گویی به این قسمت، ابتدا باید پیکسل های مربوط به ناحیه آب را استخراج کرده سپس هر یک پیکسل را ضرب در سطح معادل با scale کرده و مقدار حاصل را نیز در عمق آب بدست آید. بعد تصاویر در هر سه حالت یکی بوده و مقیاس خطی تصویر برداری نیز ۲۰ کیلومتر است بدین معنا که تعداد پیکسل های خط افقی معین شده در تصویر معادل با ۲۰ کیلومتر است. برای استخراج این هدف، این ناحیه را بصورت دستی Crop کرده و مشخص می شود که ۳۱۴ پیکسل نماینده ۲۰ کیلومتر است پس طول هر پیکسل برابر با $\frac{۳۱۴}{۲۰۰۰۰} = ۰.۰۱۵7$ متر بوده و توان دوم آن نیز نشانگر سطح هر پیکسل می باشد که در محاسبات استفاده است.

در تصاویر سال ۲۰۱۸ و ۲۰۲۰ پیکسل های مربوط به دریاچه شباهت بسیاری با ناحیه پیرامون دارد و این باعث ایجاد خطای بسیاری می شود. از جایی که در تصویر سال ۲۰۰۰ ناحیه دریاچه به صورت کامل قابل استخراج است، آن را بدست آورده و از آن به عنوان Mask در سایر تصاویر استفاده شده است.

تابعی تحت عنوان WaterMaskExtractor پیاده سازی شده است که ناحیه های مربوط به آب که در یک قطعه و پیوسته بیشترین اتصال قرار داشته باشند را استخراج کرده و در قالب marker معادل با آن تصویر بر میگرداند(با استفاده از توابع کتابخانه Open-CV).

برای هر سال ابتدا ناحیه دریاچه بر اساس mask سال ۲۰۰۰ استخراج کرده و سپس ناحیه های دارای بیشترین آب را با تابع بالا بدست می آوریم؛ سپس در آن نواحی از پیکسل های در هر سه کanal میانگین گرفته(میانگین هر کanal) و با در نظر گرفتن یک ترانس بالا و پایین، در کل محیط دریاچه پیکسل های آبی را استخراج می کنیم. در نهایت تفاوت هر یک از آن را با سال قبل محاسبه کرده و نمایش می دهیم. این خروجی ها برای هر سال به ترتیب از چپ به راست در تصویر صفحه بعد قابل ملاحظه است.

برای محاسبه حجم آب تبخیر شده در سال ۲۰۱۸ نسبت به ۲۰۰۰ تعداد پیکسل های آبی هر کدام را در مساحت هر پیکسل(مقیاس بدست آمده طبق توضیحات بالا) ضرب و مجدد در عمق داده شده برای هر سال ضرب می کنیم تا حجم بدست آید و سپس تفاضل آن را برای حجم تبخیری محاسبه می کنیم که تقریبا با مقدار زیر برابر می شود؛ همچنین برای محاسبه درصد آب های احیا شده نیز، درصد حجم آب در سال ۲۰۱۸ و ۲۰۲۰ را نسبت به ۲۰۰۰ بدست می آوریم که برابر با مقدار های زیر است. مشاهده می شود که تقریبا ۱۸٪ (۲۰-۲۰) آب

ها با مرجع قرار دادن سال ۲۰۰۰ احیا شده است:

```
area_in_meter = (20000/314)**2
vol_2000 = _2000_water_count*area_in_meter*(2.8)
vol_2018 = _2018_water_count*area_in_meter*(0.6)
vol_2020 = _2020_water_count*area_in_meter*(0.9)
print("Vol of Evaporated from 2000 to 2018:",vol_2000-vol_2018)
```

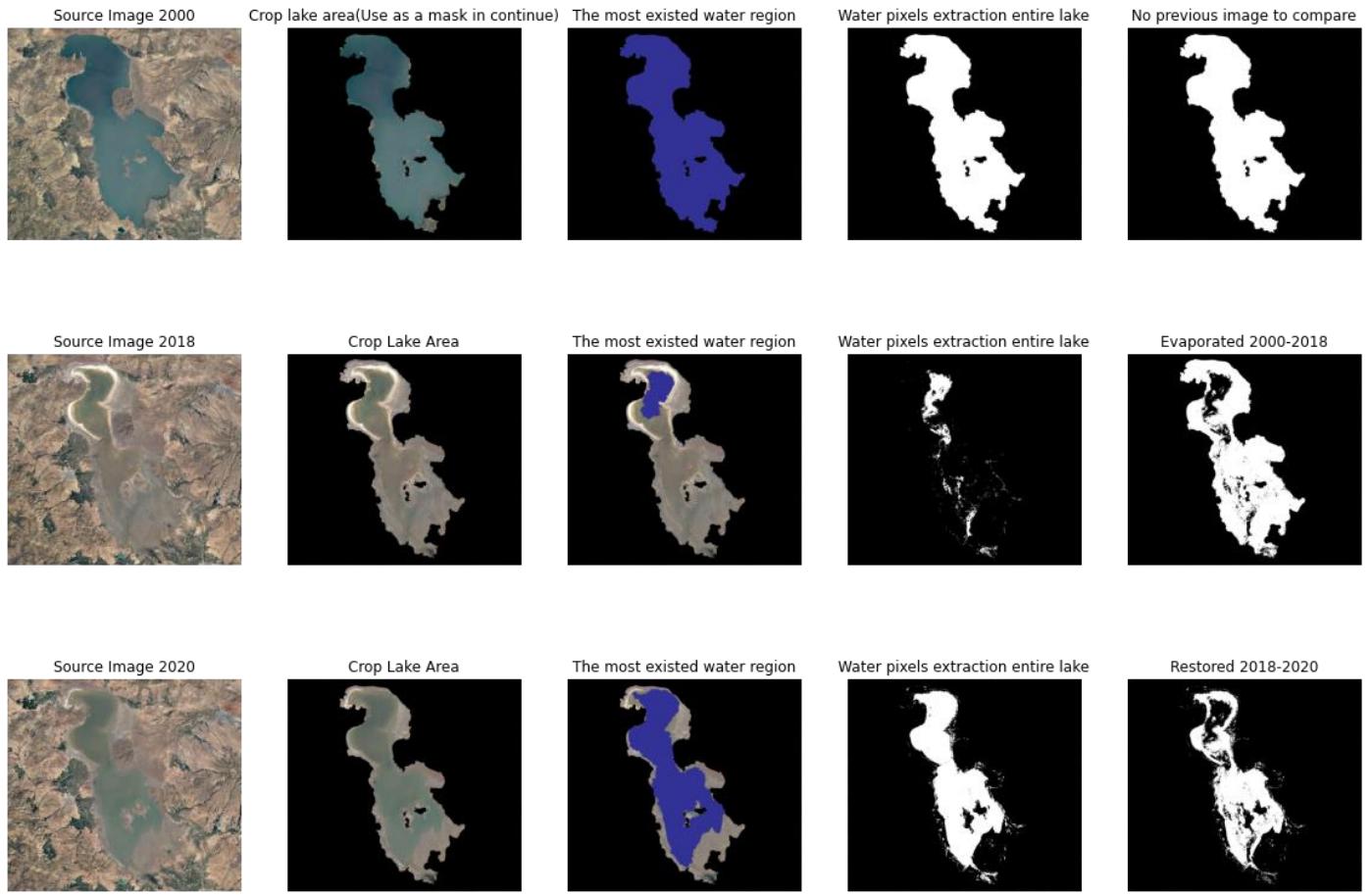
Vol of Evaporated from 2000 to 2018: 14756634346.22094

شکل ۱- قسمت b سوال سوم - حجم آب تبخیر شده

```
print("Percentage of water existed in 2018 to 2000:",round(vol_2018/vol_2000*100,3),"%")
print("Percentage of water existed in 2020 to 2000:",round(vol_2020/vol_2000*100,3),"%")
✓ 0.1s
```

Percentage of water existed in 2018 to 2000: 2.434 %
Percentage of water existed in 2020 to 2000: 20.433 %

شکل ۹- قسمت b سوال سوم - درصد احیای دریاچه



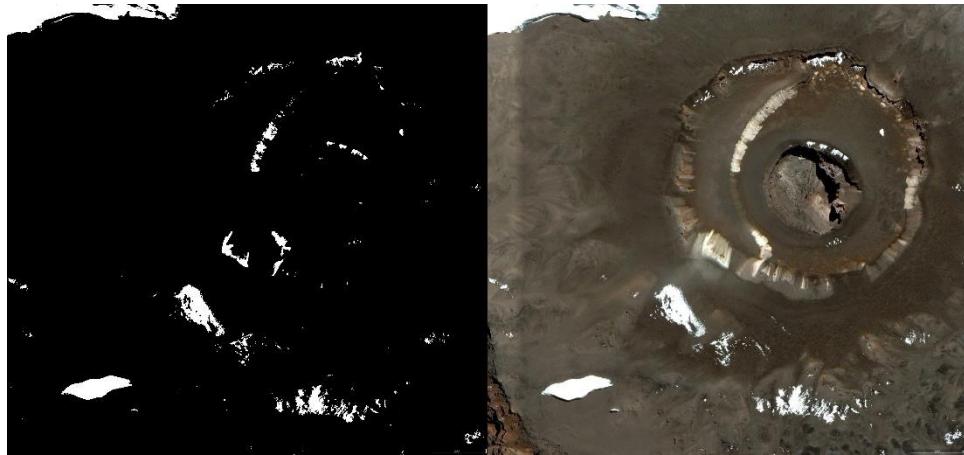
شکل ۱۰ - قسمت c , b سوال سوم - اقدامات انجام شدهی مرحله به مرحله برای تصاویر هر سال

مسئله ۳ قسمت d

برای پاسخ گویی به این قسمت نیز تقریبا مشابه با قسمت a اقدام می کنیم؛ ناحیه یخی پیکسل هایی هستند که سطوح روشانی بسیار بالایی دارند لذا کافی است با قرار دادن یک threshold روی کanal های سه گانه تصاویر پیکسل های یخی شناسایی شده در هر سه کanal را استخراج کرده و با اعمال and روی آنها، ناحیه های تمام یخی را mask نمود که نتایج برای هر سال بصورت زیر قابل مشاهده است:



شکل ۱۱ - قسمت d سوال سوم - یخ تشخیص داده شده سال ۲۰۰۳



شکل ۱۲ - قسمت d سوال سوم - بخش تشخیص داده شده سال ۱۷۰۲

بعد تصاویر در هر دو حالت یکی بوده و مقیاس خطی تصویر برداری نیز ۲۰۰ متر است بدین معنا که تعداد پیکسل های خط افقی معین شده در تصویر معادل با ۲۰۰ متر است. برای استخراج این هدف، این ناحیه را بصورت دستی Crop کرده و مشخص می شود که پیکسل نماینده ۲۰۰ متر است؛ پس طول هر پیکسل برابر با $\frac{293}{200}$ متر بوده و توان دوم آن نیز نشانگر سطح هر پیکسل می باشد که در محاسبات می توان استفاده نمود؛ اما از جایی که نسبت تغییرات خواسته شده است، میتوان این مقدار را از صورت و مخرج ساده نمود.

درصد بخ های ذوب شده تقریباً برابر با $11/61\%$ تخمین زده می شود:

```
# calculate how much ice is melted
str(round((1-((icy_2017)*4.8)/((icy_2003)*6.2))*100,2))+"%"
✓ 0.8s
'61.11%'
```

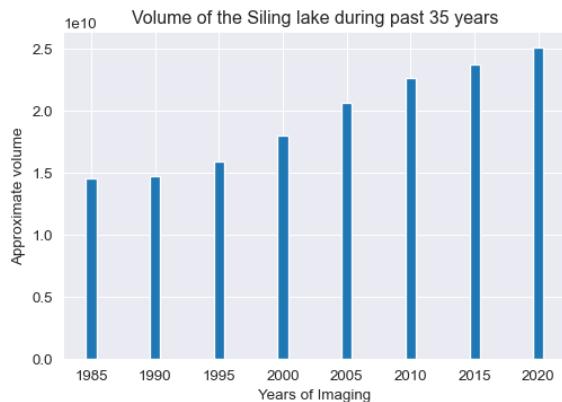
شکل ۱۳ - قسمت d سوال سوم - نسبت بخ ذوب شده

مسئله ۳ قسمت e)

برای پاسخ گویی به این قسمت نیز تقریباً مشابه با قسمت a و d اقدام می کنیم؛ باید threshold هایی روی کanal های سه گانه اعمال نموده و آنها را به باینری (۰ و ۲۵۵) تبدیل نموده و ناحیه آبی را تشخیص داد. باید threshold کanal ها از هم متفاوت باشد تا ناحیه تیره تر و آبی-سبز تر که نشانگر آب ها می باشد شناسایی شود. به ترتیب سطح ۸۵ و ۹۰ و ۱۰۰ برای کanal های RGB به عنوان مرز روشنایی در نظر گرفته می شود. سپس پیکسل هایی که خاموش بوده یا پیکسل هایی که کanal آبی و آبی-سبز روشن داشته باشند به عنوان آب در نظر گرفته می شود. تصاویر به ترتیب عملیات در صفحه بعد قابل ملاحظه است.

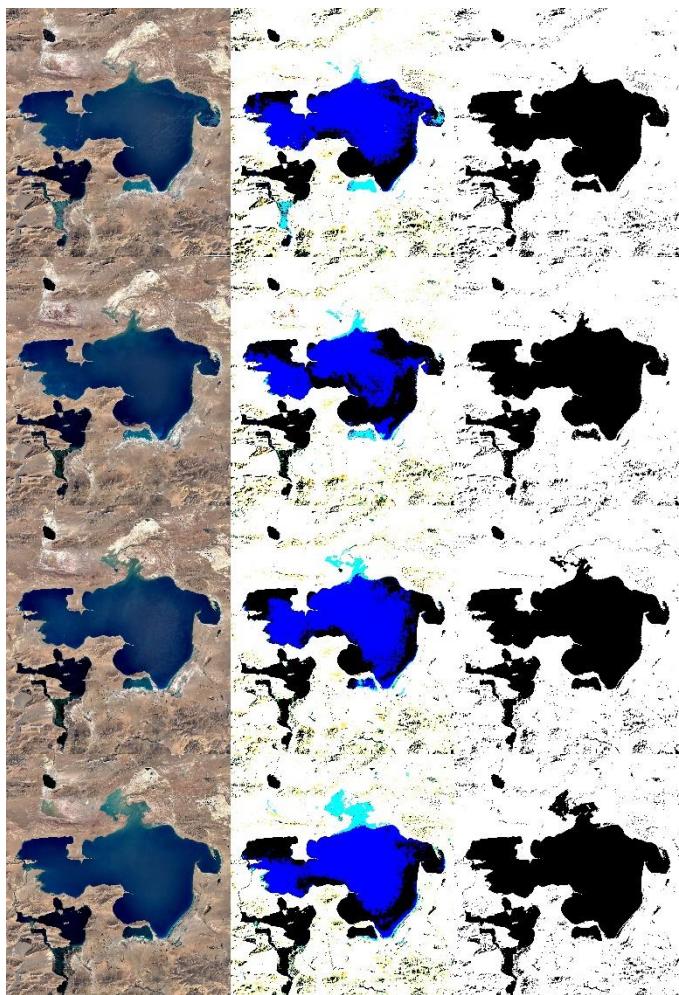
بعد تصاویر در همه حالت ها یکی بوده و مقیاس خطی تصویر برداری نیز ۱۰ کیلومتر است بدین معنا که تعداد پیکسل های خط افقی معین شده در تصویر معادل با ۱۰ کیلومتر است. برای استخراج این هدف، این ناحیه را بصورت دستی Crop کرده و مشخص می شود که ۲۵۱

پیکسل نماینده ۱۰ کیلومتر است؛ پس طول هر پیکسل برابر با $10000/251$ متر بوده و توان دوم آن نیز نشانگر سطح هر پیکسل می باشد که در محاسبات حجم می توان استفاده نمود. به عنوان عمق ۶.۴ متر را در نظر گرفته و در هر عکس آن را 0.3 افزایش داده و در تعداد



شکل ۱۴- قسمت ۲ سوال سوم - نمودار تغییرات حجم آب از سال ۱۹۸۵ تا ۲۰۲۰

پیکسل های ناحیه آب ضرب کرده و مقدار حاصل را در سطح معادل $(10000/251)$ ضرب میکنیم تا حجم آب به تفکیک هر سال بدست آید که نتیجه آن در قالب نمودار میله ای بصورت مقابله حاصل شده است:



شکل ۱۵- قسمت ۲ سوال سوم - پردازش های انجام شده برای تصاویر سال ۱۹۸۵ تا ۲۰۰۰ - شکل ۱۶- قسمت ۲ سوال سوم - تصاویر سال ۲۰۰۰ تا ۲۰۲۰

مسئله چهارم

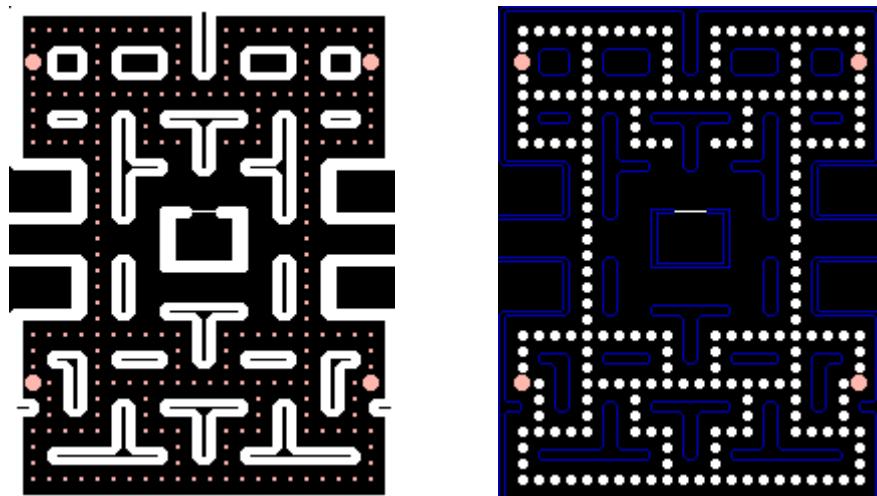
مسئله ۴ قسمت (a)

برای حل این قسمت، دقیقاً تابع خواسته شده به همان فرمت پیاده سازی شد. مختصات نقاط نقوشه(Dot) را که همان مسیر های نقطه ای نارنجی رنگ ما برای شروع هستند با استفاده از تابع `matchTemplate` از کتابخانه `open-cv` استخراج کرده و ۵ مورد را تصادفی انتخاب می کنیم. سپس روی یکی از نقاط `pacman` و روی چهار نقطه دیگر روح ها را قرار می دهیم. برای هر کدام از روح ها، پا و جهت حرکت را نیز تصادفی انتخاب می کنیم.

برای قرار دادن المان ها روی نقوشه، یک تابع به نام `PosWriter` پیاده سازی شده است که وسط آبجکت را دقیقاً روی نقطه تصادفی قرار می دهد و خروجی نقوشه را در قالب عکس بر می گرداند. یک نمونه خروجی اولیه در تصویر زیر مشخص آمده است:



همچنین در زیر نتایجی از تشخیص مانع ها و نقطه ها در تصویر نقوشه با استفاده از تابع `matchTemplate` آورده شده است:



مسئله ۴ قسمت b

برای حل این قسمت، تابع خواسته شده با همان فرمت ورودی و خروجی پیاده سازی شد. اما برای نظام مندی و یکپارچگی کد ها، در بدنه این تابع دو سری تابع مختلف فراخوانی می شود. سری اول مربوط به راه داده و تغییر حالات روح ها می باشد که در زیر به اختصار هر یک توضیح داده می شود:

❖ تابع GetNewPos : این تابع بر اساس جهت حرکت المان ها، پوزیشن جدید را بر اساس پوزیشن فعلی بدست آورده و بر می گرداند.

❖ تابع Access : این تابع اجازه حرکت از یک نقطه به جهت مورد نظر را بررسی می کند که مانع یا دیوار نباشد.

❖ تابع GhostMove : این تابع روح مورد نظر، نقطه ای روح در فریم قبلی در آن حضور داشته و نقشه بازی را دریافت می کند و این روح را با استفاده از دو تابع مدنظر بالا، در نقشه قرار می دهد.

سری دوم توابع پیاده سازی شده شامل حرکت دادن پکمن می باشد که در زیر توضیحات مختصر آن آورده شده است:

❖ تابع IsNotAte : این تابع یک نقطه از نقشه به همراه جهت مورد بررسی و عمق دید را دریافت می کند و بررسی می کند که آیا در عمق خواسته شده(فاصله از نقطه کنونی) در جهت مورد نظر، dot برای خوردن پکمن وجود دارد یا خیر. و در صورتی که آیتمی برای خوردن وجود داشته باشد، مقدار true بر می گردد.

❖ تابع EatPos : این تابع موقعیت موقعیت فعلی و جهت پکمن را دریافت می کند و نقطه ای در حال گذر را به عنوان نقطه خورده شده ذخیره می نماید. این تابع زمانی فراخوانی می شود که مانع در جهت حرکت پکمن وجود نداشته و dot در مسیرش وجود دارد.

❖ تابع PacMove : این تابع پک من را با استفاده از توابع مورد نیاز پیاده سازی شده در فوق حرکت می دهد. پارامتر های ورودی این تابع عبارت است از نقشه ای که پکمن در آن جاگذاری شود، موقعیت فعلی پکمن و جهت پک من.

همه توابع فوق از دل تابع گفته شده یعنی gen_demo فراخوانی می شود. گفتنی است با توجه به این طبق ویدیوی بازی که در متن سوال قرار داشت، روح ها میتوانند از روی هم عبور کنند. در زمانی که روح ها از روی هم عبور می کنند تشخیص روح توسط matchTemplate و یا هر گونه کار های مقایسه و تطبیق عکس ها و استخراج آن امکان پذیر نیست.(اگر یک روح روی دیگری راه برود امکان تشخیص خود روح و پا هایش وجود ندارد) از این رو وضعیت برخی المان ها در کنار تشخیص توسط تصویر نگه داری می شود.

خروجی همه عکس های تولیدی در پوشه ضمیمه شده قرار دارد.

مسئله ۴ قسمت (c)

ویدیوی خواسته شده با سرعت ۸ فریم در ثانیه و در فرمت mp4 ذخیره شده و با نام Video Output-۰۰.mp4 در پوشه خروجی این سوال قرار داده شده است. ویدیوی خواسته شده با استفاده از عکس های خروجی قسمت قبل و توسط تابع VideoWriter از کتابخانه openCV خروجی گرفته شده است.

مسئله پنجم

مسئله ۵ قسمت (a)

چشم ما و شبکیه اش که میتوان آن را یک دوربین تصویر برداری تلقی نمود، یک عملکرد فوق العاده ای در بحث پردازش تصویر و انتقال آن به مغز ما دارد. عمدۀ پردازش های انجام شامل پردازش های Enhancement می باشد؛ یکی از این پردازش ها deblurring و blurring می باشد که در آن به عبارتی چشم فوکوس می کند. وقتی چشم به یک شی نزدیک نگاه می کند، اشیا و محیط پشت آن را که در میدان دید وجود دارد (blur) می کند تا جزئیات شی نزدیک بهتر قابل درک شود و در حال مقابله نیز وقتی یک شی دور را نگاه می کند، اشیا و محیط نزدیک در میدان دید را تار می کند تا جزئیات آن بهتر قابل درک شود.

یک پردازش دیگر تقریبا شامل contrast stretching و تنظیم سطح روشنایی می باشد. در آن چشم وقتی در محیط تاریک قرار می گیرد، نور دریافتی از محیط را تقویت می کند تا جزئیات بیشتری قابل درک باشد (مانند آن که شب وقتی در اتاق چراغ خاموش مطلق می شود، همچنان اجزایی از اتاق قابل درک می باشد). در نقطه مقابله نیز وقتی چشم در ناحیه روشن و پر نور قرار می گیرد، نور دریافتی از محیط را محدود می کند تا بتواند اجزا و اشیای موجود در محیط را از دست نداده و آنها را درک و مشاهده کند.

یک پردازش دیگر شامل denoising می باشد که در آن چشم می تواند نویز های حاصل از محیط را محدود کند که تقریبا همگام با پردازش blurring رخ می دهد. مانند زمانی که چشم انسان در یک محیط گرد و خاکی یا مه و باران قرار می گیرد، اجزای وابسته به آن محیط (ریزگرد ها یا قطرات باران) را در میدان دید محدود کرده و کم تاثیر می کند.

یک پردازش دیگر که آن را تا حدودی می توان نزدیک به calibration یا registration دانست، چشم انسان جهت نور و زاویه تصویر را اصلاح می کند و تصویر را در صورت نیاز دوران می دهد تا جهت مستقیم به مغز تحويل داده شود (فارغ از اصلاح جهت نور و روودی به ناحیه شبکیه ای چشم که یک پردازش دیگر است)؛ مثلاً حالتی که اگر چشم به یک شی مستقیم نگاه کند و ما زاویه سر و راستای دید خود را ضمن نگاه کردن به آن شی تغییر دهیم (نه چندان اساسی و شدید)، راستای تصویر دریافتی از چشم ثابت می ماند.

(نکته بالا را یک متخصص شبکیه ای چشم به بنده گفته است؛ لذا ممکن است در انتقال مطلب مد نظر او در چارچوب مباحث پردازش تصویر دچار خطأ باشیم یا منظور او کلا مورد دیگری بوده باشد؛ به همین جهت ممکن است این پیش پردازش گفته شده جای بحث داشته باشد)

مسئله ۵ قسمت b

دو دوربین گفته شده را ابتدا بر اساس حجم ذخیره سازی یا همان storage مقایسه می کنیم؛ با افزایش چشم گیر سایز تصویر خروجی و در نتیجه افزایش ذخیره سازی تعداد پیکسل های بیشتر، حجم ذخیره سازی مورد نیاز نیز افزایش می یابد. در دوربین گوشی Samsung Galaxy S22 که سنسور isocell-gn5 را دارد، تصاویر خروجی دارای تعداد پیکسل های بالای بوده و بایستی پیکسل های بیشتری نسبت به 13 iphone 50MP (12MP) پس حجم ذخیره سازی بیشتری برای تصاویر خروجی از S22 مورد نیاز است. به بیان دیگری، از جایی که سایز تصویر خروجی از S22 بزرگتر از 13 iphone می باشد؛ حجم مورد نیاز برای ذخیره سازی نیز بیشتر خواهد بود.

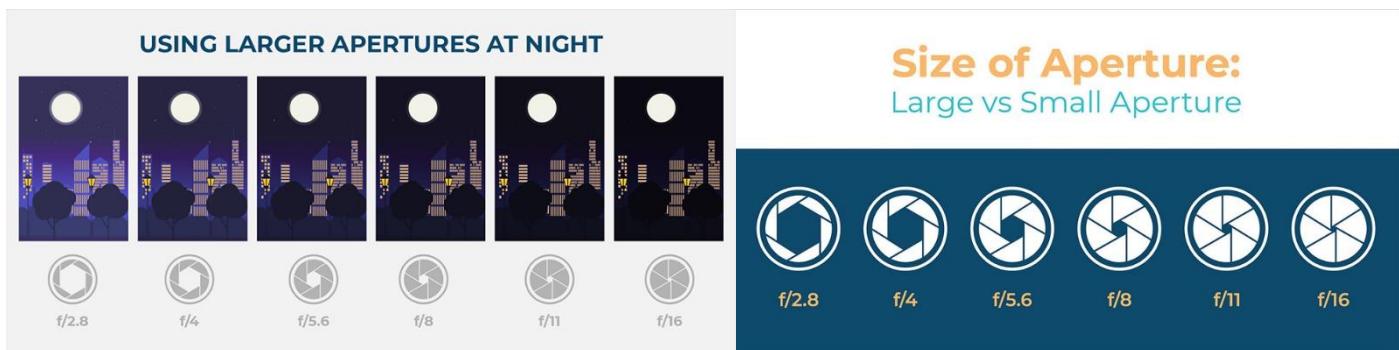
البته باید توجه داشت سنسور مذکور مورد استفاده در S22 به جهت ذخیره جزئیات حداکثری با هدف کاهش پیکسل های مورد نیاز برای ذخیره سازی و بهبود شفافیت، هر چهار پیکسل همسایه را با هم ادغام کرده و در قالب یک پیکسل ذخیره می کند لذا تقریبا 12.5MP ذخیره می شود که همچنان بیشتر از iphone می باشد. البته حجم مورد نیاز برای ذخیره عکس های گرفته شده به عواملی دیگری نظیر تنوع رنگ های همسایگی در محیط، نویز و ... نیز بستگی دارد.

حال در خصوص مقایسه resolution نیز طبق اسلاید درس (intro3) می دانیم که هر چه چقدر رزولوشن یک تصویر بالاتر باشد، یعنی تعداد پیکسل های ذخیره شده آن از محیط نیز بالاتر بوده و پیکسل های بیشتری برای جزئیات ریز در تصویر تعلق و ذخیره می شود که این نیز باعث بهبود کیفیت تصویر می باشد. (البته با فرض یکسان بودن سایز فریم و دیافراگم و شدت نور محیط و...). لذا طبق اطلاعات صورت سوال رزولوشن دوربین 13 iphone معادل با 12MP کمتر از دوربین S22 (50MP~>12.5MP) می باشد لذا تاحدودی انتظار جزئیات بیشتری از دوربین S22 می رود.

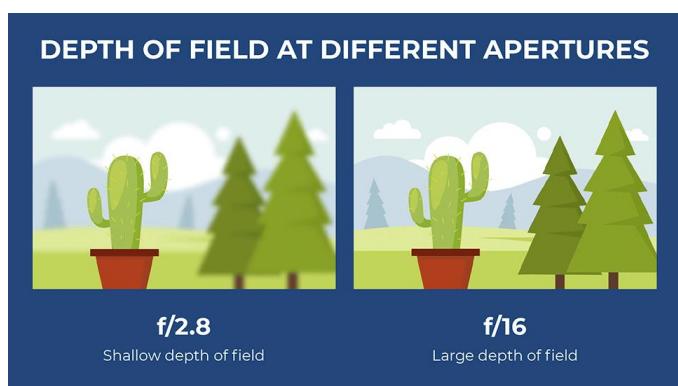
طبق مقایسه قبل انتظار می رود به واسطه رزولوشن دوربین S22، کیفیت آن نیز فوق العاده بهتر باشد اما چنین نتیجه گیری درست نیست چرا که دیافراگم دوربین نیز یک عامل بسیار مهم در ثبت تصاویر با کیفیت می باشد. حال مورد بعدی برای مقایسه، کیفیت تصاویر یا quality می باشد. تعیین یک معیار واحد برای ارزیابی کیفیت یک تصویر جزو چالش های حوزه پردازش تصویر بوده است و توصیف های گوناگونی برای آن نظیر (۱) عملکرد خوب در انواع محیط های روشن و تاریک نظری شب و روز، (۲) ثبت بهتر جزئیات صحنه، (۳) نزدیکی رنگ های ثبت شده نسبت به واقعیت، (۴) مقایسه در شرایط یکسان و... وجود دارد.

در مقایسه میدانی علاوه بر مشاهده نمونه عکس های گرفته شده از هر دو گوشی، ویدیوی مقایسه دوربین آن دو را در youtube نیز نگاه کردم. به عنوان چشم ناظر به این نتیجه رسیدم کیفیت دوربین 13 iphone بهتر از S22 است چرا که رنگ ها طبیعی و واقعی تر بود در حالی که در S22 رنگ ها و جزئیات بسیار تیز و تصنویعی ثبت می شد.

پس از مقایسه میدانی کیفیت، نوبت مقایسه توسط داده های آماری رسیده است: دیافراگم دوربین که دهانه‌ی ورودی نور به سنسور محسوب می شود یک مورد اساسی در کیفیت تصاویر ثبت شده می باشد. اگر دیافراگم دوربین بتواند نور بیشتری از محیط دریافت نماید، خواهد توانست عملکرد خوبی در شرایط های نوری مختلف خصوصاً ناحیه های کم نور و تاریک ایفا نماید و باعث افزایش کیفیت خواهد شد. همچنین اگر دیافراگم دوربین بزرگ باشد، باعث خواهد شد که تصویر حاصل عمق میدان نوری وسیع تری را ارائه کند که این نیز باعث ایجاد blurring در ناحیه خارج از مرکز(فوکوس) شده و جزئیات در ناحیه مرکز(فوکوس) بهتر و با کیفیت تر نمایان و ثبت باشد.(منبع)



با توجه به توضیحات بالا، میتوان انتظار داشت که دوربین iPhone 13 بواسطه داشتن دهانه دیافراگم بزرگتر (1.8 در مقابل 1.6)، قدرت مانور بیشتری در عکس برداری داشته و کیفیت بهتری در ارائه جزئیات و همچنین محیط روشن و تاریک داشته باشد؛ که این نیز تا حد اندکی بر عکس نتیجه مقایسه رزولوشن است. لذا گفتنی است صرفاً مقایسه آماری و بر اساس ساختار فیزیکی دوربین یا تعداد پیکسل های



تصویر خروجی مرجع معتبری برای مقایسه کیفیت تصاویر نمی باشد چرا که موتور پردازشگر گرافیکی و بهبود دهنده تصویر موجود در دوربین نیز تأثیر بسیار بسیاری در کیفیت تصاویر ثبت شده دارد و می تواند در برخی موارد این مقایسه‌ی انجام شده را به چالش بکشد؛ همان گونه که شرکت Apple از دیرباز تاکیدش در بهبود کیفیت تصاویر خروجی و نه شاخص های آماری دوربین نظری سایز بوده است.

مسئله ۵ قسمت (c)

می دانیم که تصاویر دیجیتال رنگی توسط ماتریس هایی دو بعدی و در سه کanal RGB توصیف می شوند که در آیه های این ماتریس ها اعدادی صحیح و نامنفی بوده و با توجه تعداد bit های مدنظر می تواند مقادیر مختلفی به خود بگیرد که معمول ترین آن در بحث تصویر برداری و ذخیره سازی هشت بیتی و شامل ۰ الی ۲۵۵ می باشد. زمانی که هر یک از پیکسل های رنگی توسط یک تبدیلی نظری رابطه زیر

به یک کanal grayscale تبدیل می شود، ممکن است عدد به دست آمده به جهت اعمال ریاضی صحیح نباشد؛ لذا بایستی یک عملیات تقریب زنی اعمال شود که این باعث می شود مقدار بدست دقیقاً برابر با رابطه نبوده و امکان اعمال معکوس آن از بین برود:

NTSC formula: $(0.299 \cdot \text{Red} + 0.587 \cdot \text{Green} + 0.114 \cdot \text{Blue}) * 255 = \text{grayscale}$

همچنین از طرفی طبق رابطه جبری موجود، اگر بخواهیم از RGB تبدیل انجام دهیم بایستی یک معادله با سه مجھول را حل کنیم که این جواب یکتا ندارد لذا معکوس این تبدیل معنی دار نیست.

مسئله ۵ قسمت (d)

عنوان re-sampling به عملیاتی اطلاق می شود که طی آن نمونه(پیکسل) های حاضر در تصویر شامل کاهش(downsampling) یا افزایش(upsampling) شود که معمولاً resize نیز خطاب می شود؛ البته معمولاً افزایش نمونه ها در استفاده از عبارت بیشتر مد نظر قرار می گیرد. برای این امر، الگوریتم ها و روابط ریاضیاتی متعددی پیشنهاد می شود که یکی از مهم ترین آنها درون یابی عددی بین پیکسل ها برای تولید پیکسل جدید می باشد که تولید جزئیات در تصویر را تا حدودی در پی دارد. به طور خلاصه، با تولید پیکسل بین پیکسل های موجود، سعی می شود رزولوشن تصویر افزایش یابد.

عنوان sub-sampling نیز به عملیاتی اطلاق می شود که طی آن تعدادی از نمونه(پیکسل) های تصویر حذف شده و ما صرفاً یک زیر مجموعه از پیکسل های تصویر را انتخاب و با آن کار می کنیم. برای این امر نیز الگوریتم ها و روابط متعددی پیشنهاد می شود که یکی از آنها حذف periodic می باشد که مثلاً یک در میان پیکسل ها حذف می شود. به طور خلاصه، با حذف تعدادی از پیکسل ها سعی می شود ضمن حفظ جزئیات، ابعاد تصویر کاهش پیدا کرده و حجم ذخیره سازی کمتری مورد نیاز باشد.

مسئله ۵ قسمت (e)

بله قطعاً تاثیر گذار بوده و میتواند در تشخیص شباهت رنگی بین تصاویر رنگی تفاوت بسیار ایجاد کند. برای مقایسه شباهت رنگی قطعاً gray انتخاب خوبی نیست چرا که در بخش c همین سوال دیدیم که ممکن است دو رنگ کاملاً متفاوت در RGB در فضای grayscale یک مقدار یکسان بگیرد که این باعث می شود عملاً امکان مقایسه از بین برود.

از جایی که هدف مقایسه رنگ های شبیه به هم می باشد لذا اگر رنگ های شبیه به هم در فضای رنگی نزدیک هم قرار گیرند، مقایسه و ارزیابی عددی شباهت کار صحیحی خواهد بود لذا RGB و CMYK نیز تا حدودی فضای رنگی خوبی برای مقایسه شباهت دو رنگ نمی باشد چرا که رنگ های شبیه بهم در یک محدوده عددی و کنار هم بصورت منسجم قرار ندارند و این باعث می شود تفاضل گیری کanal

ها معیاری برای سنجش شباهت نباشد. در مقابل فضاهای رنگی HSL و YUV و خصوصاً HSV فضاهای رنگی مناسب تری برای سنجش شباهت رنگ‌ها می‌باشد چرا که در هر سه، به جهت عددی رنگ‌های شبیه به هم در همه کانال‌های مختص خود نزدیک هم قرار دارند. با توجه به ساختار قطبی، به نظر بnde HSV فضای رنگی توصیف پذیر تری برای سنجش شباهت دو رنگ را ارائه می‌کند و انتخاب بهتری است.

