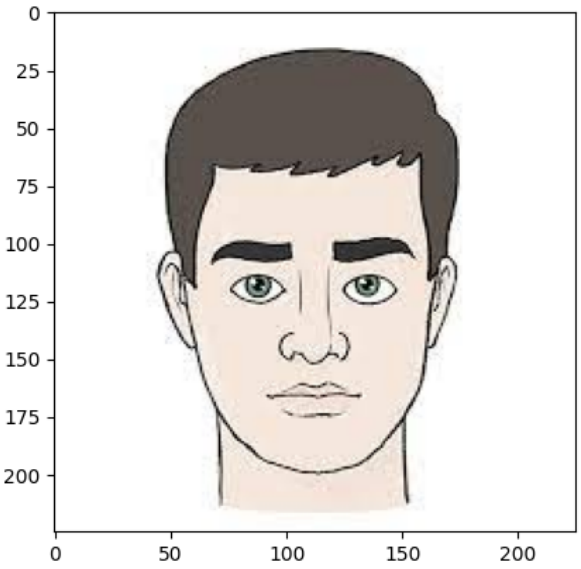
به نام خدا

گزارش تمرین سری چهارم - درس مبانی بینایی کامپیوتر

سید محمد علی نخاری - شماره دانشجویی : 99521496

سوال سوم) ابتدا باید تصویر گفته شده را با کتابخانه opencv بخوانیم. برای این کار نیاز به مسیر قرار گرفتن تصویر داریم که چون برای حل تمرین از محیط colab استفاده کردم، این مسیر را درون متغیر path ذخیره کردم. علاوه بر این چون کتابخانه opencv تصویر را به فرمت BGR میخواند باید کانال های آن را برعکس کنیم تا زمانی که با استفاده از کتابخانه matplotlib آن را میخوانیم تصویر به درستی نمایش داده شود. تصویر خوانده شده به صورت زیر است:



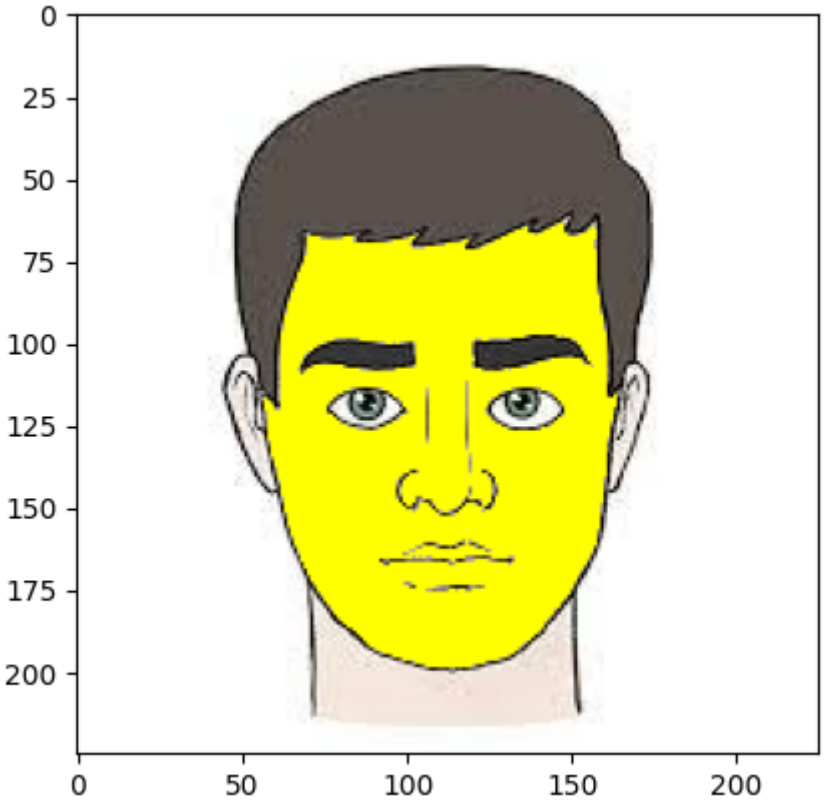
پس از اینکه تصویر را خواندیم با استفاده از تابع segment و یک نقطه شروع که اصطلاحا به آن seed هم گفته میشود شروع به پیدا کردن ناحیه هایی با شباهت نقطه شروع میکنیم. در این سوال با دادن نقطه شروع روی گونه سمت راست فرد، به دنبال پیدا کردن ناحیه صورت او هستیم.
برای اینکار ابتدا بر روی تصویر پیمایش کرده و اختلاف هر نقطه از تصویر با رنگ سفید([255, 255, 255]) بدست می آوریم. با گرفتن میانگین از نتیجه به دست آمده چک میکنیم اگر مقدار میانگین از عدد 70 کوچکتر باشد مقدار متناظر با آن نقطه را 1 و در غیر اینصورت برابر با صفر

قرار میدهیم. هدف از این کار باینری کردن تصویر ورودی میباشد.(با این کار نقاطی که رنگ مشابه دارند مقدار یک و رنگ هایی که اختلاف زیاد دارند مقدار صفر میگیرند)

پس از باینری کردن تصویر، یک داده ساختار صف ایجاد میکنیم و نقطه آغازین را درون آن میریزیم. سپس حلقه ای می نویسیم و تا زمانی که این صف خالی نشده است، پیمایش را ادامه میدهیم. در هر پیمایش نقطه اول صف را برداشته و دو شرط را چک میکنیم. شرط اول اینکه نقطه مورد نظر در محدوده تصویر داده شده باشد و شرط دوم اینکه نقطه را از قبل ندیده باشیم.

اگر نقطه مورد نظر شرایط گفته شده را داشته باشد در صورتی که مقدار متناظر با این نقطه در تصویر باینری شده مقدار یک داشته باشد مقدار آن در آرایه mask برابر با 255 قرار میدهیم.(این آرایه برای نگهداری نقاط مشاهده شده استفاده میشود) سپس رنگ متناظر با این نقطه در تصویر اصلی را برابر با رنگ زرد قرار میدهیم([0, 255, 255]) و در آخر چهار نقطه متصل به این نقطه را به صف اضافه میکنیم(نقاط بالا-پایین-چپ و راست).

نتیجه نهایی به صورت زیر میباشد:

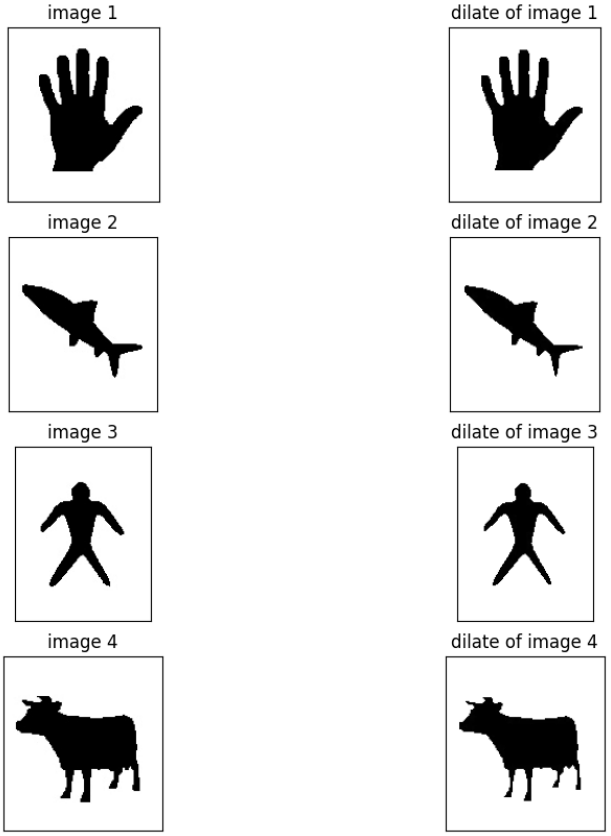


سوال ششم)

الف) یکی از توابع کمکی نوشته شده برای این سوال، تابع padding است که کار آن اضافه کردن حاشیه از نوع zero-padding است.

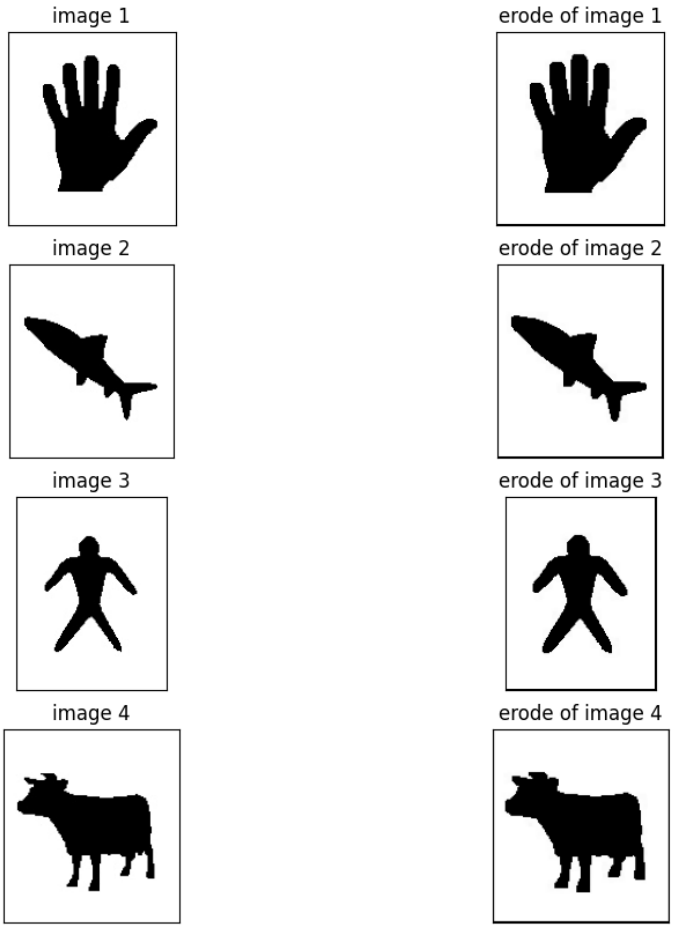
عملگر گسترش:

برای پیاده سازی تابع dilation ابتدا کرنل داده شده را به اندازه 180 درجه می چرخانیم تا انعکاس آن به دست بیاید. سپس بر روی تصویر داده شده پیمایش میکنیم و به ازای هر پیکسل به اندازه کرنل یک مربع با مرکزیت پیکسل مورد نظر جدا میکنیم. پس از آن بررسی میکنیم که کرنل داده شده اشتراکی با پیکسل های جدا شده از تصویر دارد یا نه. برای این کار دو ماتریس را در هم ضرب میکنیم و اگر حاصل عددی بزرگتر از صفر داشت، بزرگترین مقدار موجود در مربع جدا شده از تصویر را جایگزین همان پیکسل از تصویر میکنیم. نتیجه به صورت زیر خواهد بود:



همانطور که مشاهده میشود نواحی تاریک در تصاویر نهایی نازک تر شده و این یعنی نواحی با رنگ روشن گسترش(dilate) یافته است.

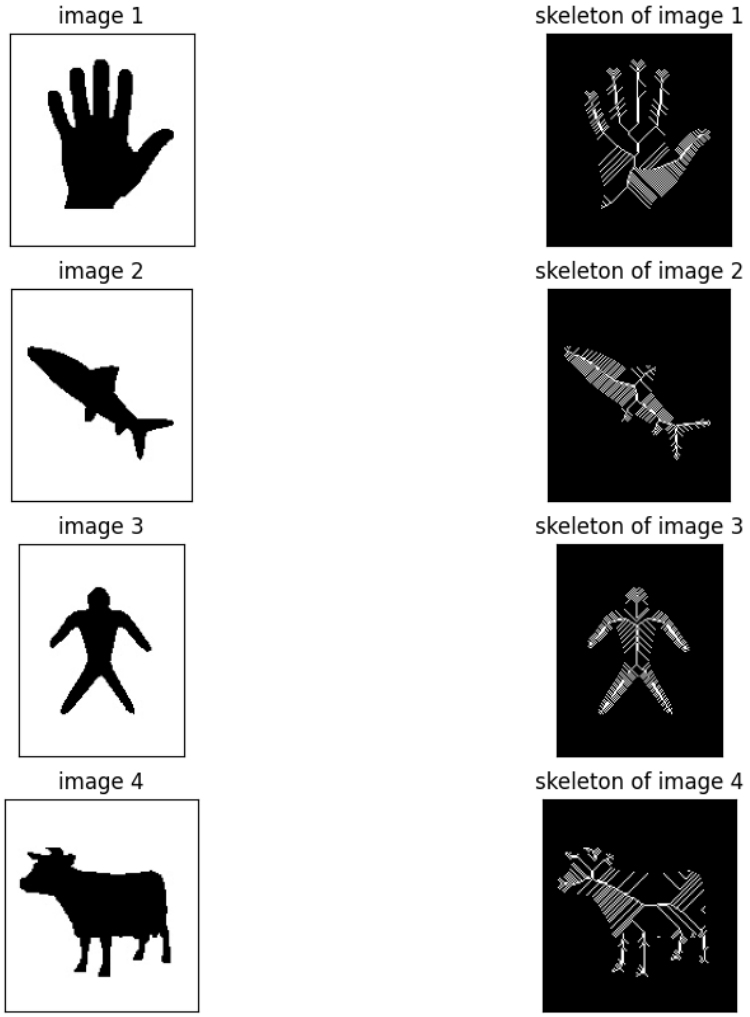
عملگر سایش: در اینجا هم مثل قسمت قبل عمل میکنیم با چند تفاوت. اولین تفاوت این است که نیازی به انعکاس عنصر ساختاری نداریم. دومین تفاوت این است که برای انجام این عمل باید کرنل داده شده زیر مجموعه ماتریس جدا شده از تصویر اصلی باشد. برای بررسی این شرط کرنل داده شده در ماتریس جدا شده از تصویر ضرب کرده و چک میکنیم که آیا حاصل جمع عناصر حاصل ضرب از حاصل جمع عناصر موجود در ماتریس جدا شده از تصویر هست یا نه. اگر بود مقدار خانه مورد نظر از تصویر اصلی را برابر با کوچکترین مقدار موجود در ماتریس جدا شده از تصویر قرار میدهیم.



همانگونه که واضح است نتیجه حاصل از سایش در نواحی تیره ضخیم تر شده است و این یعنی نواحی روشن با سایش مواجه شده اند.

برای انجام باقی عملگر ها مورفولوژی یعنی عملگر های باز و بسته، از توابع نوشته شده در بالا یعنی توابع dilate و erode استفاده میکنیم. برای عملگر باز ابتدا تصویر داده شده را با کرنل erode و سپس نتیجه به دست آمده را با همان کرنل dilate میکنیم.
برای عملگر بسته هم ابتدا تابع dilate و سپس عملگر erode را بر روی نتیجه به دست آمده صدا میزنیم.
ب) برای انجام این قسمت فقط کافی است توابع موجود در کتابخانه opencv را صدا زده و تصویر و کرنل داده شده را به عنوان پارامتر های ورودی به آنها پاس دهیم.
برای عملگر های باز و بسته از تابع morfologyEx استفاده کردم که یکی از پارامتر های ورودی آن مشخص میکند که عملگر باز بر روی تصویر انجام شود یا عملگر بسته.(cv2.MORPH_OPEN و cv2.MORPH_CLOSE)
نتایج حاصل از توابع آماده کتابخانه opencv با نتایج به دست آمده از توابع پیاده سازی شده، یکسان بودند.
ج) منظور از اسکلت تصویر، فرایندی است که در آن سعی میکنیم تا پیش زمینه تصویر یا همان foreground را تا حد ممکن از بین ببریم تا تنها اجزا اصلی (اسکلت) از آن باقی بمانند.
برای اینکار ابتدا نیاز است تا تصویر داده شده را باینری کنیم. یعنی رنگ های روشن 255 و رنگ های تیره را با 0 مقدار دهی کنیم. علاوه بر این نیاز است تا جای پیکسل های روشن و تیره تصویر را جابجا کنیم چرا که در تصاویر داده شده اشیاه رنگ تیره دارند و پس زمینه رنگ سفید. برای اینکه بهتر بتوانیم نتیجه نهایی را مشاهده کنیم نیاز است که اسکلت تصویر را به رنگ سفید و پس زمینه به رنگ مشکی باشد.
پس از باینری کردن تصویر، بر روی آن پیمایش میکنیم تا زمانی که پیکسل های با مقادیر مخالف صفر در آن به صفر برسند(یعنی foreground را آنقدر سایش میدهیم که تنها اجزای اصلی آن باقی بمانند) در هر پیمایش بر روی تصویر باینری شده و یک کرنل عملگر باز را اعمال میکنیم تا جزئیات ریز و نویز های موجود در تصویر را تا حد خوبی از بین ببریم. پس از آن مقدار به دست آمده از عملگر باز را از تصویر باینری کم کرده و آن را با متغیر skeleton که آرایه ای دو بعدی به اندازه

تصویر ورودی و با مقادیر 0 است جمع میکنیم و در آخر تصویر باینری را برابر با نتیجه حاصل از همان تصویر باینری با کرنل پس از اعمال عملگر سایش، قرار میدهیم. نتیجه حاصل از پیاده سازی به صورت زیر میباشد:



منابع استفاده شده:

- اسلاید ها و فیلم های کلاس درس
- سایت رسمی کتابخانه opencv و numpy
- chatGPT