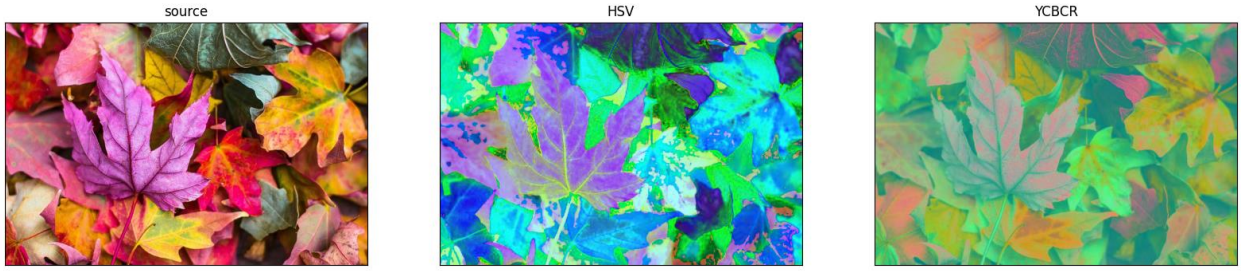
1) الف) با توجه به فرمول پایین پیاده سازی میکنیم کد را که البته باید رنگ های R , G , B را به اسکیل 0 تا 1 ببریم چون CMYK در مقیاس درصد است. برای این کار آن ها را تقسیم بر 255 میکنیم. در ادامه هم دقیقا فرمول زیر را پیاده سازی میکنیم. پاسخ :[0.31372549, 0.23529412, 0. , 0.49019608]

$$K = 1 - max(R, G, B)$$

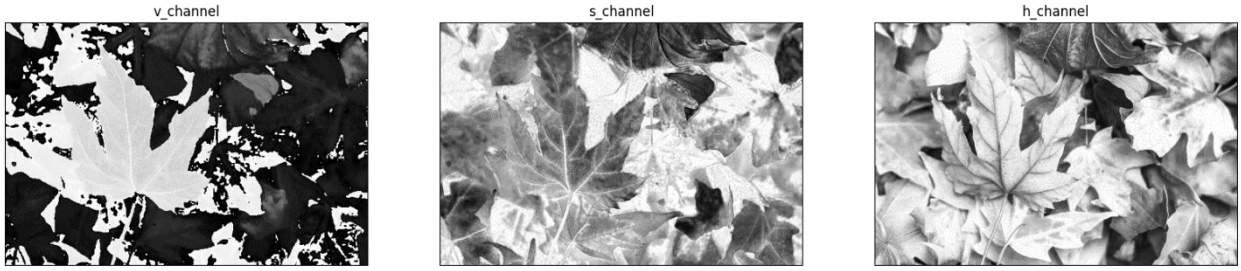
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1-K \\ 1-K \\ 1-K \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

ب) برای تبدیل فضای عکس که BGR است (چون با کتابخانه opencv خوانده ایم) به YCbCr آرگومان دوم تابع cvtColor را BGR2YCbCr قرار میدهیم.

برای تبدیل فضای عکس که BGR است (چون با کتابخانه opencv خوانده ایم) به HSV آرگومان دوم تابع cvtColor را BGR2HSV قرار میدهیم.



ج) برای اینکار البته به فضای HSV میبریم عکس را بعد از آن با استفاده از تابع split سه فضا را جدا میکنیم خروجی:



د) در این قسمت کانال رنگی 0 را مقادیر تصویر اول ، کانال رنگی 1 را مقادیر تصویردوم و کانال رنگی 2 را مقادیر تصویر اول قرار میدهیم با این کار اگر در هر دو تصویر یکسان باشد باشد پیکسل خاصمان، رنگش چون در هر سه کانال یکسان میشود، پس در grayscale است. در غیر این صورت یکی از دو کانال اول و دوم مقادیرشان برابر نیستند و رنگ آن نقاط متفاوت میشود. خروجی:

ه) سازگاری با دستگاه های مختلف : دستگاه ها قادر به نمایش تصویر در فضای خاصی هستند.

پوشش بیشتر رنگ ها: هر فضای رنگی دارای محدودیت هایی در مورد پوشش رنگ ها است.

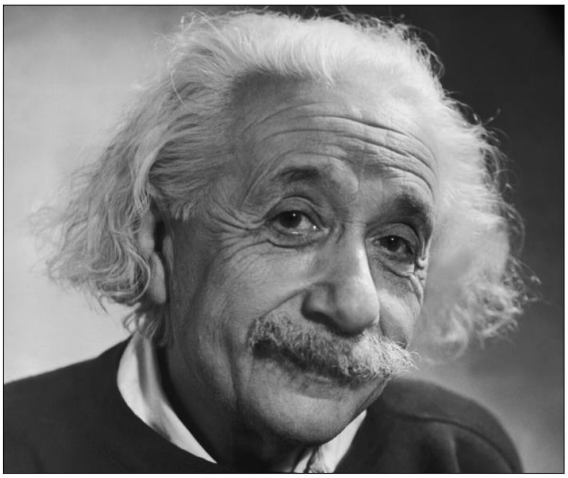
امنیت: در برخی موارد، استفاده از چند فضای رنگی به عنوان یک روش امنیتی مورد استفاده قرار می گیرد.

2) در این قسمت با استفاده از تابع Plotter ای که در سوال 1 تعریف شده عکس ها را در یک سطر نشان میدهیم.

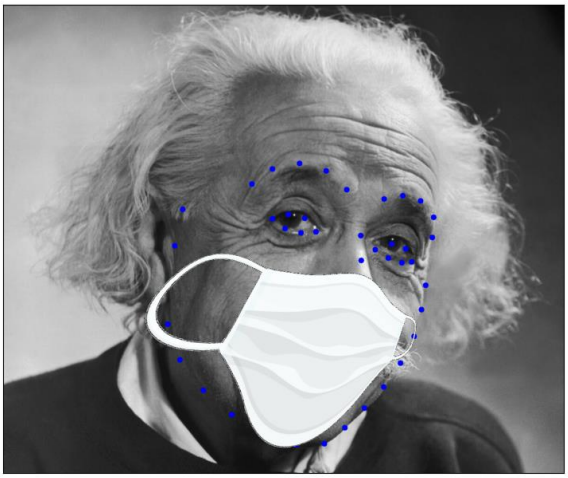
سپس با استفاده از stichy.stich که ورودی لیستی از عکس میگیرد عکس ها را به هم وصل میکنیم. خروجی:

3) در این سوال ابتدا مدل تشخیص چهره و بعد مدل pretrained شده که 68 نقطه را تشخیص میدهد را لود میکنیم. سپس عکس را به grayscale میبریم و بعد با استفاده از detector هر چهره ای که داریم را اطراف آن یک مستطیل در نظر میگیرد . در ادامه با استفاده از predictor هر 68 نقطه را به صورت لیستی از تاپل میگیریم و به صورت یک آرایه numpy نگه میداریم. بعد با استفاده از تابع getPerspectiveTransform برای محاسبه ماتریس تبدیل بین نقاط خاصی که از ماسک در پینت انتخاب کردیم را به نقاط مشخص شده در predictor استفاده میشود و بعد با استفاده از تابع warpPerspective ماسک را تبدیل می کنیم. بعد چون درعکس نقاطی که شامل ماسک است 1 است و نقاط دیگر 0 اند، در یک فور روی تمام عکس جاهایی که 1 باشند را روی عکس صورت می اندازیم.

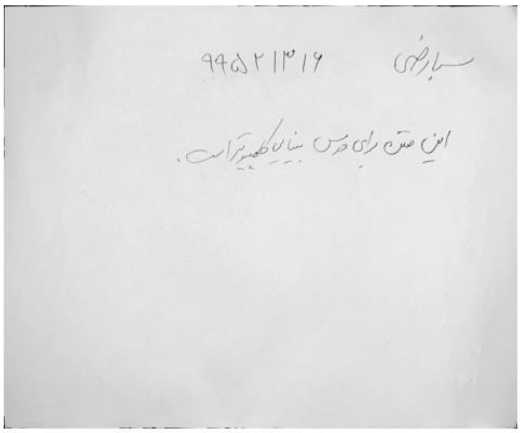
face



result



4) ابتدا عکس را خوانده و به grayscale می بریم. بعد با فیلتر گوسی نویز عکس را از بین می بریم.در ادامه با استفاده از canny (چون خطا اش کم است و برای تشخیص اشیا بهتر است) خط های عکس را تشخیص می دهیم. سپس با استفاده از findContours همه ی کانتور ها را پیدا می کنیم. و با drawContours خط ها مشخص میکنیم. بعد ماکسیمم کانتور را در نظر گرفته و با مقداری درصد خطا کانتور هایی که نزدیک به این ماکسیمم اند را به عنوان گوشه در نظر میگیریم. در ادامه مانند سوال قبل نقاط گوشه ی عکسی که محاسبه کرده ایم را به نقاط یک صفحه عادی مپ می کنیم خروجی:



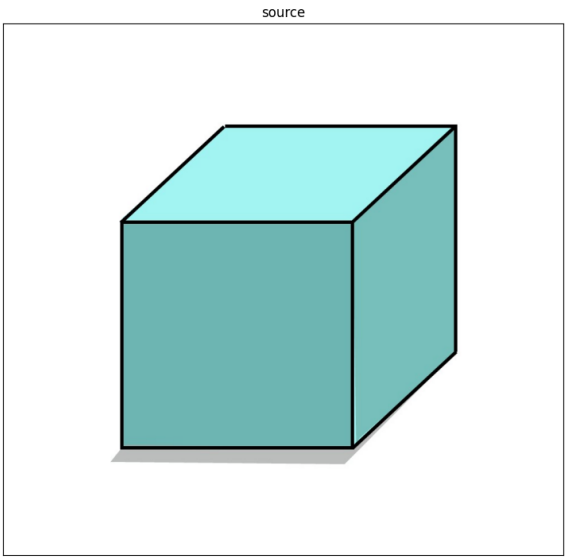
از

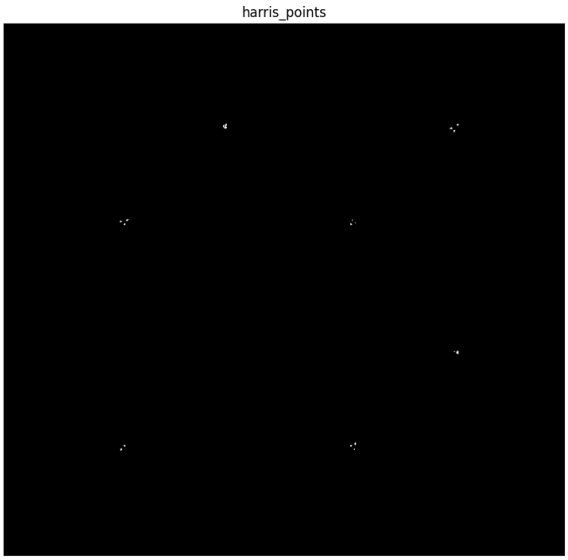
5) ابتدا با استفاده از دو فرمول زیر ماتریس را ساخته و هریس را محاسبه میکنیم.

$$\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

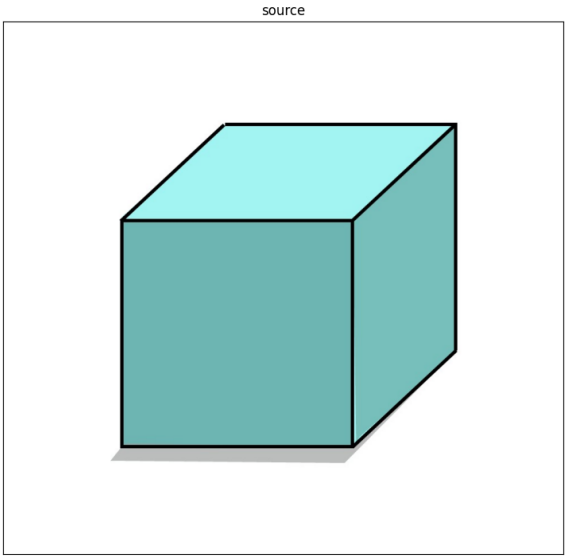
$$R = det(M) - k\big(trace(M)\big)^2$$

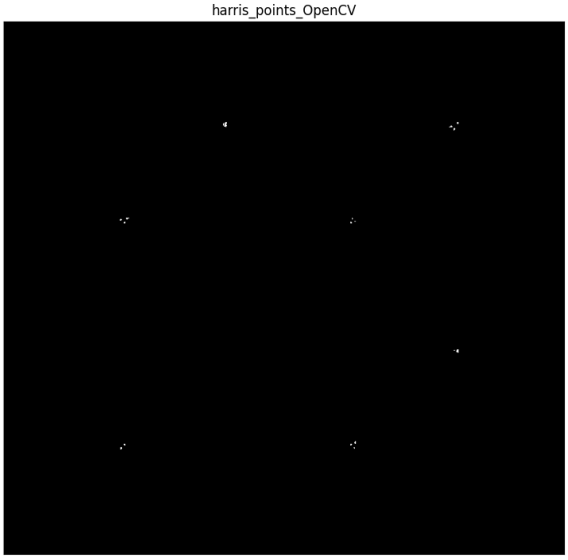
برا مشتق در راستای x , y از تابع Sobel استفاده میکنیم و از فیلتر گوسی برای سه تا Ix^2 , $IxIy$, Iy^2 استفاده میکنیم. بعد R را محاسبه کرده . و نقاطی که مقدار R آن ها ازماکسیمم R اشان (با trteshold 0.01) بیشتر بود را 255 قرار میدهیم. خروجی :





خروجی با کتابخانه opencv :





خروجی دو تصویر تفاوتی ندارند فقط محاسبات در حالت دوم بسیار راحت تر است.

6) SIFT: یک روش است که ویژگی های تصاویر را در مقیاس های گوناگون شناسایی می کند، این امکان را برای ما فراهم می کند تا تصاویری را که از مقیاس، چرخش یا ترجیحات ترجمه شده اند را مطابقت دهیم. این روش به دلیل دقت بالای خود شناخته شده است اما محاسبات آن هزینه بالایی دارد .

SURF : نسخه سریعتری از SIFT است که از تقریب فیلتر جعبه برای برآورد لاپلاسیان گوسی به جای محاسبه آن به صورت صریح استفاده می کند. این روش نیز برای مقیاس‌ناپذیر بودن و توانایی مقابله با تغییرات در شرایط نوری شناخته شده است.

ORB: الگوریتمی سریعتر و کم هزینه‌تر از SIFT و SURF است. این الگوریتم از ترکیب روش‌های FAST (Features from Accelerated Segment Test)و BRIEF (Binary Robust Independent Elementary Features) برای استخراج

ویژگی‌های تصاویر استفاده می کند. همچنین ORB مقاوم در برابر تغییرات در شرایط نوری و قابلیت مقابله با تغییرات در چرخش را داراست

بر اساس سرعت : SURF سریع تر از SIFT عمل می کند و ORB از هر دو سریع تر است.

بر اساس نقاط کلیدی شناسایی شده : ORB از هر دو بهتر عمل میکند و تقریبا 3 برابر SURF است .
بر اساس درصد نقاط کلیدی مطابقت دار:در تصویر روشن شده ORB بسیار بهتر از دو روش دیگر است در حالی که تفاوت دو روش دیگر 1 درصد است. در تصویر چرخیده ORBوSURF 100 درصد نقاط را تطبیق داده اند ولی SIFT 96 درصد و ضعیف تر عمل کرده است.
تشخیص نقاط کلیدی و تطبیق آنها : SURF , ORB از SIFT بهتر عمل میکنند.