**ـ بسم الله الرحمن الرحیم ـ**

**تمرین تحویلی شماره 3 - Multimedia**

**زهرا لطیفی 9923069**

---

**i. تشخیص صورت با استفاده از Haar Features**

**با استفاده از مطالبی که مطالعه کرده اید و همچنین به کارگیری ویژگی های Haar در OpenCv، تعداد صورت ها را در تصویر faces.jpg گزارش کنید. در قدم بعد، با استفاده از یک webcam، تصویر صورت خود را به صورت real-time با استفاده از مدل از پیش آموزش دیده OpenCv که از ویژگی های Haar استفاده میکند، پردازش کرده و مقدار FPS را گزارش کنید. چه مواردی درمقدار FPS دست یافته دخیل هستند؟ بیشترین مقدار آن را گزارش کنید.**

برای انجام بخش اول این سوال، ابتدا تابع imshow را مطابق با تمارین قبل تعریف کردیم تا برای نمایش تصاویر در هر بخش از قالب یکسانی استفاده کنیم.

سپس تصویر اصلی را با کمک تابع cv2.imread خوانده و با تابع imshow نمایش دادیم.

در گام بعد جهت انجام عملیات مورد نظر، تصویر را Grayscale کردیم.

برای اعمال face detection برروی این تصویر، لازم هست ابتدا فایل classifier را Load کنیم. HAAR Cascade Classifier یک روش تشخیص شی است که ویژگی های HAAR را برای شناسایی اشیاء در یک تصویر، وارد مجموعه‌ای از طبقه‌بندی کننده‌ها (Cascade) می‌کند. این روش برای شناسایی یک نوع شی آموزش دیده؛ با این حال، می‌توان چندین مورد از آنها را به صورت موازی استفاده کرد. (تشخیص چشم ها و چهره ها با هم) در واقع در این روش به جای استفاده از تمام فیچرها، تنها آنهایی که informative تر هستند استفاده می‌شوند تا ابتدا بررسی کنیم که آیا هر منطقه به طور بالقوه می‌تواند چهره داشته باشد یا خیر. بنابراین نیاز به محاسبه همه فیچرها به طور همزمان از بین خواهد رفت.

سپس با استفاده از تابع cascade.detectMultiScale چهره‌ها را در تصویر grayscale دیتکت کردیم.

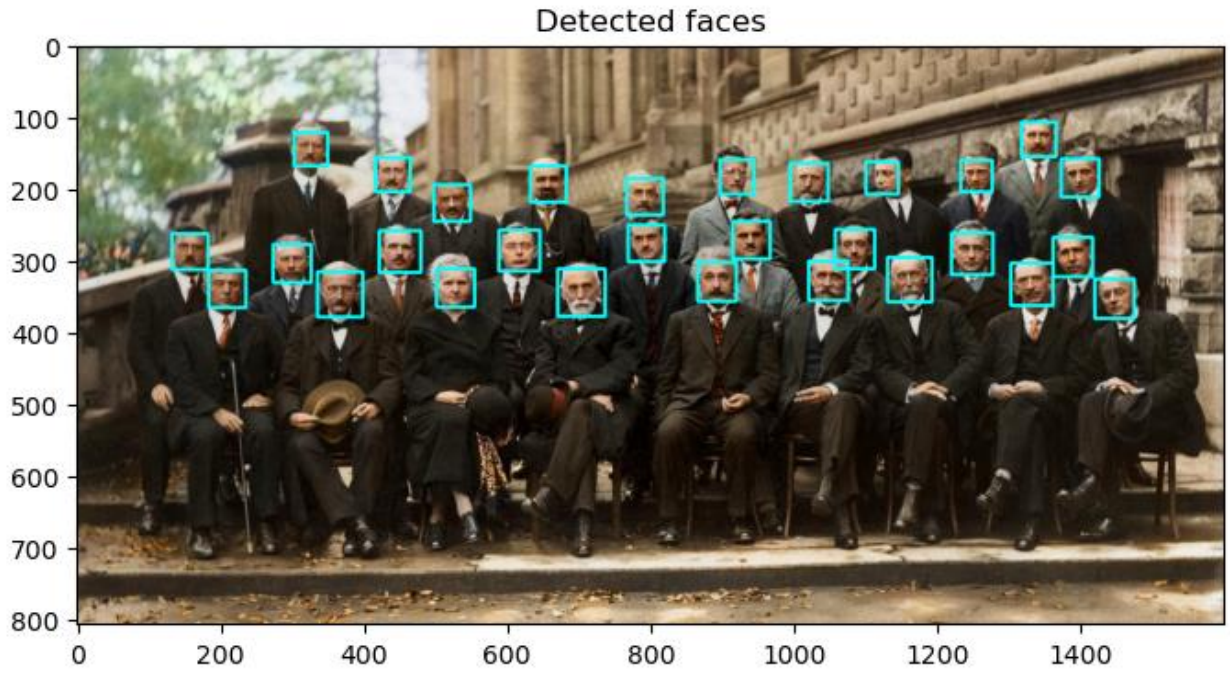
این تابع دو فاکتور قابل تنظیم دارد. یکی Scale Factor و دیگری Min Neighbors.

✓ Scale Factor : مشخص می‌کند که هر بار که مقیاس بندی می‌کنیم، چقدر اندازه تصویر را کاهش می‌دهیم. در تشخیص چهره ما معمولاً از 1.3 استفاده می‌کنیم. این به این معنی است که هر بار که مقیاس تصویر را 30٪ کاهش می‌دهیم. مقادیر کوچکتر مانند 1.05 زمان بیشتری برای محاسبه نیاز دارند، اما سرعت تشخیص را افزایش می دهند.

✓ Min Neighbors : تعداد همسایه‌هایی را که هر پنجره باید داشته باشد مشخص می‌کند و به طور معمول بین 3-6 تنظیم می شود. این فاکتور به عنوان تنظیم حساسیت عمل می‌کند. مقادیر پایین گاهی اوقات چند چهره را روی یک چهره تشخیص می‌دهند. مقادیر بالا باعث می‌شود که موارد مثبت کاذب کمتری تضمین شود، اما ممکن است برخی از چهره‌ها را از دست بدهیم. مشخصا در حل این سوال با تغییر این شاخص، با افزایش آن، به اشتباه یک کراوات دیتکت می‌شد و با کاهش آن، یکی از چهره‌های نیم‌رخ را از دست می‌دادیم.

این تابع مختصات نقاط دیتکت شده را به ما می‌دهد پس بعد از آن یک for نوشتیم تا از هر نقطه مستطیلی رسم کند.

در نهایت تصویری که چهره‌ها روی آن مشخص شده‌اند، نمایش داده شده است و تعداد چهره‌ها هم برابر با 29 گزارش شده است.

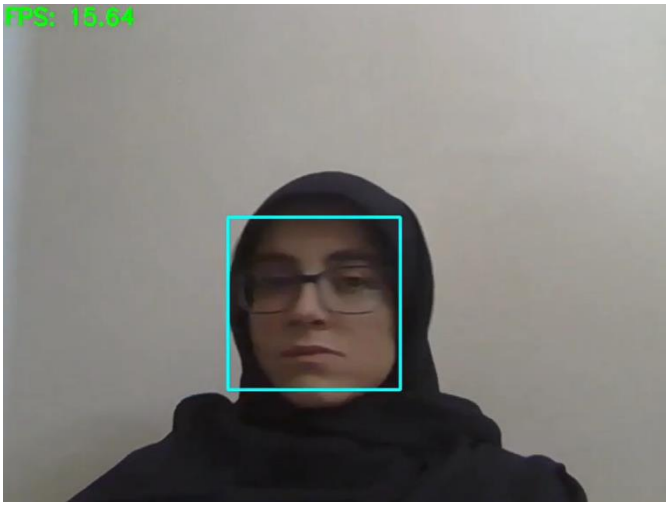



```
Number of detected faces: 29
```

برای انجام بخش دوم با دستور cv2.VideoCapture(0) وبکم سیستم را باز کرده و فریم‌ها را ضبط می‌کنیم.

سپس مقادیری برای محاسبه FPS تعریف شده اند و یک while تعریف شده که تا هنگامی که کاربر نخواسته، فریم‌های ضبط شده را در متغیر img ذخیره می‌کند. سپس FPS محاسبه شده، همانند فرآیند توضیح داده شده در بخش اول سوال، در هر فریم چهره‌های موجود دیتکت شده و مستطیل‌ها رسم می‌شوند.

سپس با استفاده از تابع cv2.putText مقدار FPS برروی هر فریم نوشته می‌شود.




درحین ضبط ویدیو، هرجا کاربر دکمه q را بفشارد، ضبط ویدیو متوقف شده و ذخیره می‌شود.

نتایج، در فایل آپلود شده و کدها در فایل تحت عنوان Q1_HW3_9923069 قابل مشاهده هستند.

**ii. Face Landmarks cropping**

**ابتده وارد سایت https://www.fotor.com/images/create شده و یک تصویر چهره با ویژگی های دلخواه بسازید. با استفاده از کتابخانه Dlib نقاط Landmark تشکیل دهنده صورت را پیدا کرده و آن ها را گزارش کنید. درمحله بعد این کار را با یک تصویر از چهره خودتان انجام دهید.**

**آیا میتوانید این دو چهره را با یکدیگر جابه جا کنید؟**

**مراحل کار را توضیح دهید.**

برای انجام این سوال هم ابتدا تصویر ساخته شده را لود کرده و نمایش دادیم. سپس آن را grayscale کرده و همانند سوال قبل با استفاده از تابع cascade.detectMultiScale چهره را در تصویر دیتکت کردیم.

سپس با یک for مختصات چهارطرف صورت را مشخص کردیم. فایل predictor مناسب را لود کرده و با استفاده از آن شکل صورت را استخراج کرده و نقاط landmark را در چهره با دستور predictor(gray, face) که در آن predictor با استفاده از کتابخانه dlib و بهره بردن از دستور dlib.shape_predictor("shape_predictor_68_face_landmarks.dat") ایجاد شده، مشخص کردیم. مختصات هر نقطه landmark پیدا شده را به تابع Circle داده و دوایر را برروی نقاط مربوطه رسم کردیم. ( در کد، ابتدا تک دایره اول مشخصا رسم شده و سپس سایر دوایر با یک for رسم شده اند.)

مشابه با همین کار را بار دیگر با تصویر خود انجام دادیم.

جابه‌جایی چهره‌ها به طور کلی دارای مراحل زیر است:

1. شناسایی فیچرهای صورت
2. تاب برداشتن تصویر به تناسب حالت جدید و متفاوت صورت
3. تطبیق رنگ
4. ایجاد حاشیه های بدون درز در لبه‌های صورت تعویض شده جدید

ابتدا ترتیب شماره‌های نقاط هر بخش از صورت مشخص شده‌اند. (Facial Landmarks Number Order)

باید بر روی چهره‌ها convex hull ای که در نتیجه اتصال نقاط landmark ها ایجاد شده قرار بگیرد؛ پس با range های جدا شده با توجه به این موضوع، Align_points و Overlay_Points را مشخص کردیم. سپس چندتابع تعریف کردیم تا فرآیند جایجایی چهره ها گویا تر انجام شود. اولی تابع get_landmark است که هرتصویر را گرفته و ماتریسی که در آن مختصات نقاط landmark آن تصویر قرار گرفته را به عنوان خروجی

می‌دهد. تابع draw_convex_hull تصویر و نقاط و یک رنگ را گرفته و convexhull مربوطه را بر روی هر چهره رسم می‌کند. در تابع get_face_mask، ماسک چهره موجود در تصویر را با استفاده از landmark ها مشخص می‌کنیم و سپس برای نرم شدن لبه ها دوبار از GaussianBlur استفاده می‌کنیم.

در تابع transformation_from_points ترازکردن نقاط باتوجه به متفاوت بودن حالت چهره ها در دو تصویر صورت می‌گیرد تا نتیجه طبیعی تری داشته باشیم.

تابع resize_and_landmarks تغییر سایزهای لازم را اعمال کرده و landmark ها و تصویر تغییر سایز یافته را به عنوان خروجی می‌دهد.

در تابع warp_im چهره ها با هم جا به جا می‌شوند.

در تابع correct_colours هم رنگ های چهره ها با هم تطبیق پیدا می‌کنند و همینطور فیلترهایی برای Blur کردن اعمال شده است.

پس از تعریف توابع، می‌بینیم که هردو تصویر resize شده و landmark هایشان مشخص شده اند. سپس نقاط نرمالایز شده، ماسک برروی تصویر دوم قرار گرفته، چهره جدا شده و روی تصویر اول قرار گرفته است.

درنهایت اصلاح رنگ ها صورت گرفته و تصویر خروجی به جهت اندازه نرمالایز شده و نمایش داده شده است.

نتایج هر مرحله در فایل Q2_HW3_9923069 آپلود شده قابل مشاهده هستند.