به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

مبانی بینایی کامپیوتر

استاد سیفی پور

نام و نام خانوادگی	سیده دیبا روانشید شیرازی ، کامیار رحمانی
شماره دانشجویی	810199431 ، 810199422
تاریخ ارسال گزارش	1403,4,16

فهرست گزارش سوالات

Age Detection 4

توضیح فایل‌ها و کتابخانه‌های موردنیاز 4

توضیح الگوریتم استفاده شده 4

نتایج : 5

Cartoonify 8

توضیح فایل‌ها و کتابخانه‌های موردنیاز 8

توضیح الگوریتم استفاده شده: 8

نتایج : 9

Colors 10

توضیح فایل‌ها و کتابخانه‌های موردنیاز 10

توضیح در مورد الگوریتم استفاده شده: 10

نتایج : 11

Detect Face and Blur 12

توضیح فایل‌ها و کتابخانه‌های موردنیاز: 12

توضیح در مورد الگوریتم استفاده شده: 12

نتایج : 13

Face recognition 14

توضیح فایل‌ها و کتابخانه‌های موردنیاز: 14

توضیح در مورد الگوریتم استفاده شده: 14

نتایج : 15

Image to Sketch 16

توضیح فایل‌ها و کتابخانه‌های موردنیاز: 16

توضیح در مورد الگوریتم استفاده شده: 16

نتایج : 16

QR code 17
توضیح فایل‌ها و کتابخانه‌های موردنیاز: 17
توضیح در مورد الگوریتم استفاده شده: 17
برای استفاده از وبکم: 17
نتایج : 18
Remove Background 19
توضیح فایل‌ها و کتابخانه‌های موردنیاز: 19
توضیح در مورد الگوریتم استفاده شده: 19
نتایج : 20
Remove Shadow 21
توضیح فایل‌ها و کتابخانه‌های موردنیاز: 21
توضیح در مورد الگوریتم استفاده شده: 21
نتایج : 22

Age Detection

توضیح فایل‌ها و کتابخانه‌های موردنیاز

این پروژه از کتابخانه OpenCV و الگوریتم های یادگیری عمیق برای انجام تشخیص چهره و تشخیص سن و جنسیت استفاده می‌کند. پس در ابتدا باید کتابخانه‌های cv2، math و argparse را در محیط Python نصب داشته باشیم یا نصب کنیم.

فایل‌های مدل مورد نیاز شامل opencv_face_detector.pbtxt و opencv_face_detector_uint8.pb برای تشخیص چهره، age_deploy.prototxt و age_net.caffemodel برای تشخیص سن، و gender_deploy.prototxt و gender_net.caffemodel برای تشخیص جنسیت هستند. این فایل‌ها باید در پوشه پروژه قرار داده شوند. سپس، با اجرای اسکریپت، اگر ورودی تصویر از طریق آرگومان `--image`داده شود، تصویر ورودی پردازش می‌شود و در غیر این صورت، دوربین وبکم به عنوان ورودی استفاده می‌شود و به صورت real-time تشخیص را انجام میدهد.

برای اینکه راحتتر بتوانیم عکس ها را ورودی بدهیم یک کد پایتون دیگر نیز نوشتم به نام gad2.py که ابتدا فایل هارا باز میکند تا بتوانیم عکس مورد نظر را انتخاب کنیم و سپس روی عکس الگورتیم تشخیص را اجرا میکند.

توضیح الگوریتم استفاده شده

الگوریتم استفاده شده در این پروژه شامل دو بخش اصلی است: تشخیص چهره و تشخیص سن و جنسیت. ابتدا با استفاده از مدل‌های از پیش آموزش‌دیده شده تشخیص چهره، چهره‌های موجود در تصویر شناسایی می‌شوند. این کار با استفاده از شبکه‌های عصبی عمیق و روش تبدیل تصویر به بلوک (blob) انجام می‌شود. سپس برای هر چهره شناسایی‌شده، یک بلوک جدید ساخته می‌شود و به مدل‌های تشخیص سن و جنسیت ارسال می‌شود. این مدل‌ها پس از دریافت ورودی، احتمال‌های مربوط به هر دسته از سن و جنسیت را برمی‌گردانند و با استفاده از این احتمال‌ها، سن و جنسیت مربوط به هر چهره پیش‌بینی می‌شود. در نهایت، نتایج روی تصویر نمایش داده می‌شود.

نتایج :

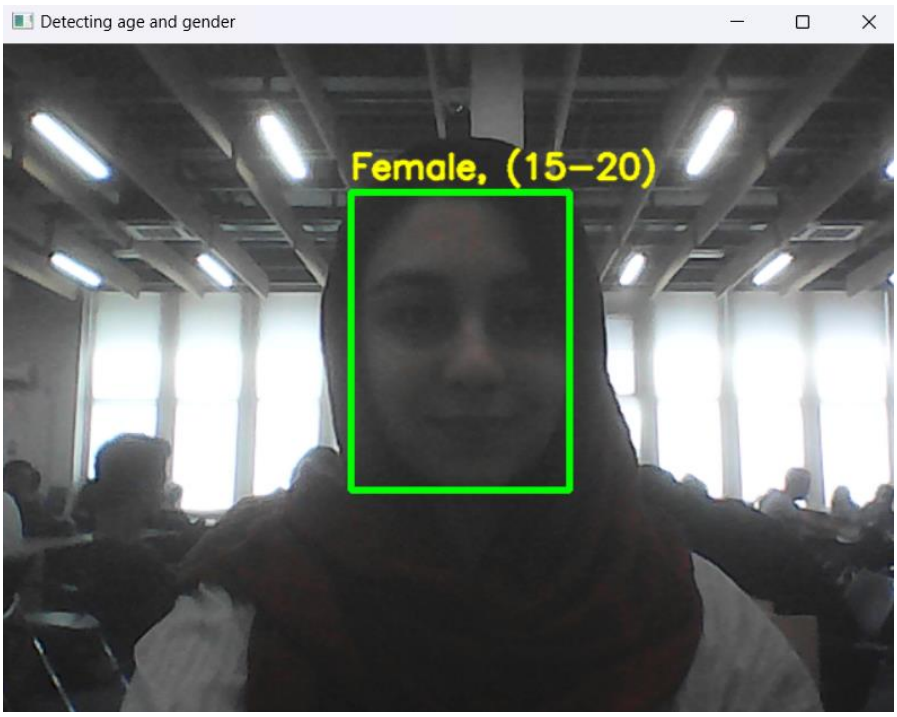


Figure 1 تصویر دریافت شده از وبکم

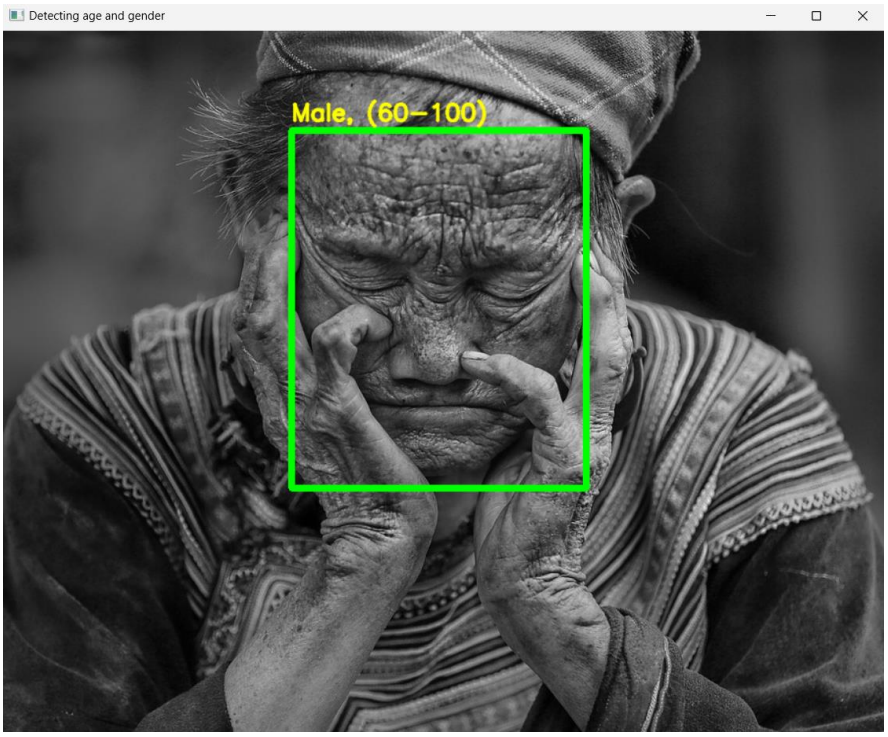


Figure 2 خروجی تصویر **man1**

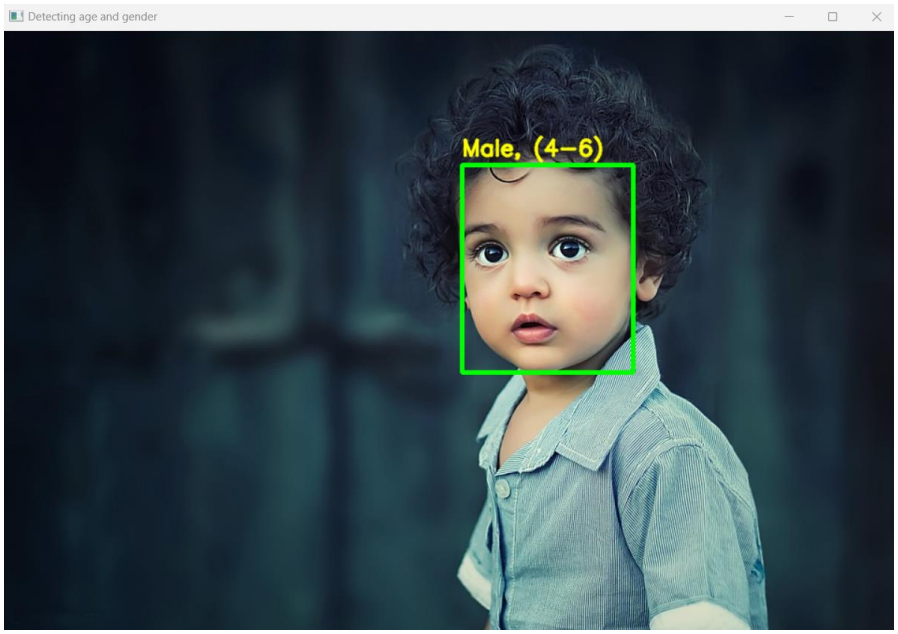


Figure 3 تصویر خروجی **kid1**

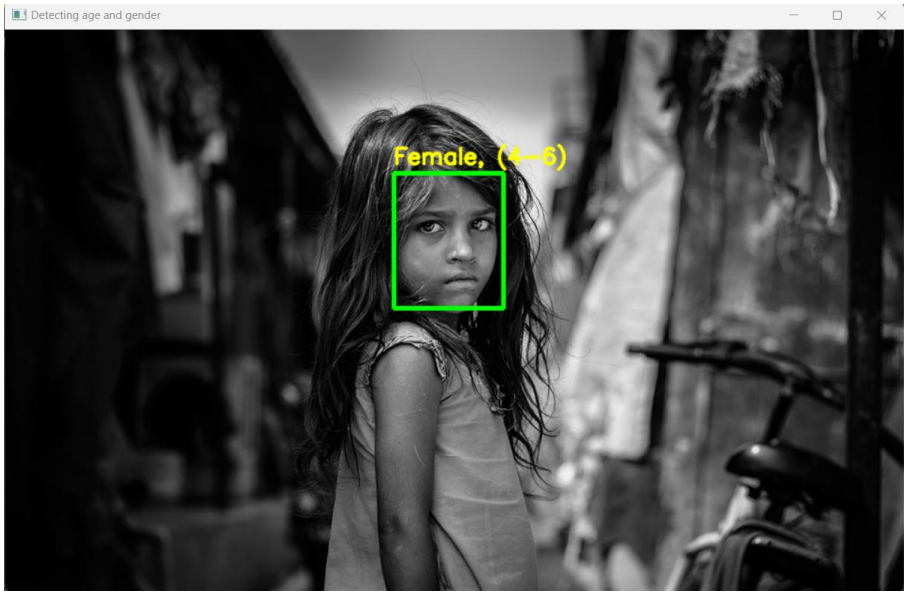


Figure 4 تصویر خروجی **girl2**

Figure 5 تصویر خروجی **woman1**

تصویر خروجی minion :

No face detected

همانطور که انتظار میرفت این الگوریتم به خوبی کار میکند.

توضیح فایل‌ها و کتابخانه‌های موردنیاز

OpenCV: برای خواندن تصاویر، تبدیل فضای رنگی و انجام وظایف پردازش تصویر استفاده می‌شود.

Easygui: برای باز کردن فایل برای انتخاب تصویر استفاده می‌شود.

Numpy: برای انجام عملیات عددی مورد استفاده قرار می‌گیرد.

matplotlib.pyplot: برای نمایش تصاویر استفاده می‌شود.

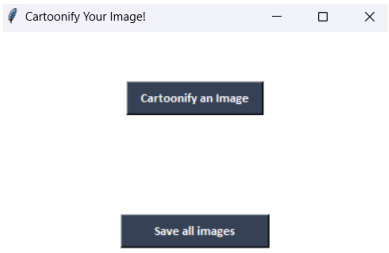
Os: برای مدیریت مسیرها در سیستم عامل استفاده می‌شود.

Tkinter: برای ایجاد عناصر رابط کاربری گرافیکی مانند دکمه‌ها و برچسب‌ها استفاده می‌شود.

PIL: برای کار با تصاویر و تبدیل آن‌ها به شیءهای قابل استفاده در Tkinter مورد استفاده قرار می‌گیرد.

توضیح الگوریتم استفاده شده:

ابتدا یک پنجره tkinter با اندازه و عنوان خاص ایجاد می‌شود که برچسب و دکمه‌هایی برای تعامل با کاربر ایجاد میکند. سپس با کلیک بر روی گزینه Carttonify an image یک فایل باز میشود که میتوانیم تصویر مورد نظر را انتخاب کنیم.



حال تصویر انتخاب شده خوانده می‌شود و به فرمت RGB تبدیل می‌شود. بررسی می‌شود که آیا تصویر با موفقیت بارگذاری شده است یا خیر. تصاویر را برای نمایش تغییر اندازه می‌دهد. (در کد اصلی این قابلیت نبود به همین دلیل عکس ها عریض میشدند.) سپس تصویر را به حالت خاکستری تبدیل کرده و با اعمال فیلتر میانه بر روی آن، تصویر را شارپ‌تر می‌کند. از روش آستانه‌ای تطبیقی برای تشخیص لبه‌ها استفاده می‌کند تا مدل کارتونی ای برای تصویر ایجاد کند. سپس از فیلتر bilateral برای حذف نویز و حفظ لبه‌ها در تصویر اصلی استفاده می‌شود.

حال اگر تصویر رنگی را با لبه ها ترکیب کنیم تصویر کارتونی نهایی به وجود می آید.

کد را طوری تغییر دادیم که همه عکس های هر مرحله را سیو کند.

نام فایل جدید قابل اجرا :Cartoonifier2.py میباشد.

نتایج :

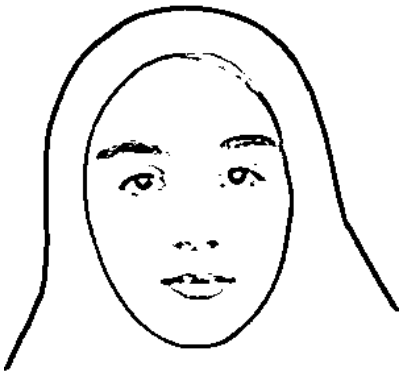


Figure 6عکس تشخیص لبه

Figure 7عکس کارتونی شده

Colors

توضیح فایل‌ها و کتابخانه‌های موردنیاز

فایل تصویر : تصویر مورد نظر با نام "colorsimage.jpg" که شامل صحنه‌ای از رنگ‌هاست، با استفاده از کتابخانه OpenCV میخوانیم.

فایل داده‌ها :یک فایل CSV به نام "colors.csv" که شامل اطلاعاتی درباره رنگ‌ها است، با استفاده از کتابخانه Pandas میخوانیم. این فایل شامل اطلاعاتی مانند نام رنگ، کد HEX ، و مقادیر RGB است.

برنامه سپس از یک پنجره گرافیکی استفاده می‌کند تا کاربر بتواند روی تصویر کلیک کند و رنگ مورد نظر را انتخاب کند.

توضیح در مورد الگوریتم استفاده شده:

الگوریتم تشخیص رنگ در این برنامه شامل دو تابع زیر است:

recognize_color(R,G,B) :

این تابع برای تشخیص نام رنگ بر اساس مقادیر RGB ورودی عمل می‌کند. برای هر رنگ موجود در فایلCSV ، اختلافات مقادیر RGB محاسبه شده و نامی که کمترین اختلاف را دارد انتخاب می‌شود.

mouse_click(event, x, y, flags, param) :

این تابع به دبل کلیک روی تصویر حساس است. وقتی روی تصویر کلیک می‌کنیم، مقادیر RGB نقطه کلیک شده در تصویر خوانده می‌شوند و بر روی تصویر یک مستطیل با رنگ متناظر با این نقطه و یک متن شامل نام رنگ و مقادیر RGB نمایش داده می‌شود. برنامه به طور مداوم تا زمانی که کلید Esc را فشار دهیم، ادامه پیدا می‌دهد و پس از آن پنجره بسته میشود.

نتایج :

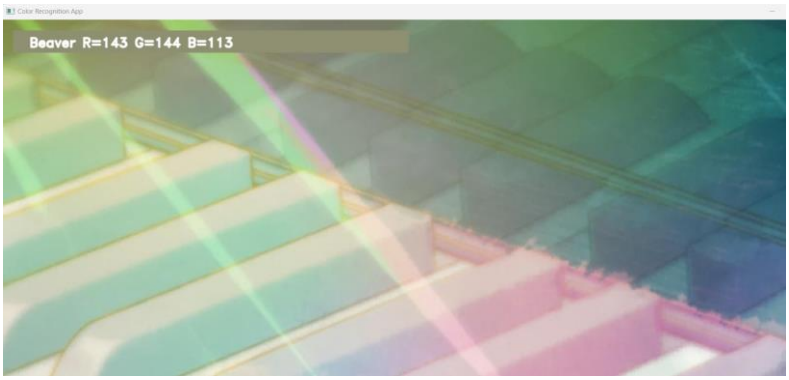


Figure 8رنگ تشخیص داده شده

Detect Face and Blur

توضیح فایل‌ها و کتابخانه‌های موردنیاز:

OpenCV: این یک کتابخانه قدرتمند برای پردازش تصویر است. در اینجا، از cv2 برای خواندن تصویر، پردازش آن، اعمال فیلتر گوسی (blur) و انجام تشخیص چهره با استفاده از یک مدل شبکه عصبی استفاده شده است.

numpy: این کتابخانه برای کار با آرایه‌ها و محاسبات عددی در Python استفاده می‌شود. در اینجا برای انجام عملیات محاسباتی روی مختصات و ماتریس‌ها که از تصویر و مدل استفاده می‌شود، به کار گرفته شده است.

os.path.dirname: از ماژول os.path استفاده می‌کند. برای مدیریت مسیرها و نام فایل‌ها استفاده میشود. اینجا برای ایجاد مسیرهای فایل استفاده شده است، به طوری که برنامه بتواند فایل‌های مدل موردنیاز را به صورت نسبی به پروژه خود متصل کند.

توضیح در مورد الگوریتم استفاده شده:

از مدل شبکه عصبی کانوولوشنی با استفاده از فایل‌های Res10... و deploy.. ساخته شده است.

تصویر ورودی خوانده می‌شود و سپس با استفاده از `cv2.dnn.blobFromImage()` پیش‌پردازش می‌شود. این شامل تغییر اندازه تصویر و کاهش میانگین RGB تصویر با (104.0, 177.0, 123.0) می‌باشد.

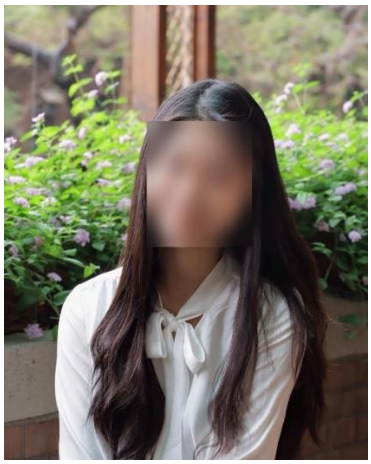
تصویر پیش‌پردازش شده به عنوان ورودی به مدل داده میشود و خروجی آن بدست می آید.

خروجی شبکه شامل مستطیلی است که چهره را در تصویر ورودی مشخص می‌کند. اگر اطمینان (confidence) بیشتر از 0.4 باشد، آن چهره از تصویر اصلی را برش میدهیم و با استفاده از `cv2.GaussianBlur()` اثر blur را اعمال می‌کنیم. سپس تصویر blur شده را در جایگاه خود در تصویر اصلی قرار می‌دهیم.

در نهایت، تصویر نهایی که چهره اش با اثر blur تغییر یافته را ذخیره می‌کنیم.

نتایج :

Figure 9 محو کردن چهره



Face recognition

توضیح فایل‌ها و کتابخانه‌های موردنیاز:

برای اجرای این بخش باید کتابخانه‌های زیر را داشته باشیم:

OpenCV

OpenCV-contrib

pillow

توضیح در مورد الگوریتم استفاده شده:

سیستم تشخیص چهره از الگوریتم هیستوگرام الگوهای باینری محلی (LBPH) استفاده می‌کند.

این روش تصویر ورودی را به سیاه و سفید تبدیل می‌کند و سپس آن را به چندین شبکه تقسیم می‌کند. برای هر شبکه، LBP محاسبه شده و یک هیستوگرام که نمایانگر ظاهر چهره است، ایجاد می‌شود. این هیستوگرام‌ها به یک بردار ویژگی واحد ترکیب شده و برای دسته‌بندی استفاده می‌شوند.

این الگوریتم به تغییرات نور و حالات چهره مقاوم است.

این بخش شامل سه قطعه کد میباشد که باید به ترتیب ران شوند:

1. در فایلface_taker.py ابتدا اسم فرد پرسیده شده و سپس 30 تصویر از چهره فرد گرفته میشود. (میتوانیم تعداد عکس ها را افزایش دهیم تا کیفیت الگوریتم افزایش پیدا کند.)

همچنین فایل names.json را با شماره و نام کاربر به‌روزرسانی می‌کند. عکس ها هم در فولدر image ذخیره میشوند.

2. در فایل face_train مدل با الگوریتم LBPH (Local Binary Patterns Histograms) با تصاویری که تا اینجا ذخیره کرده است آموزش داده میشود و در فایل trainer.yml ذخیره میشود.
3. در نهایت در فایل face_recognizer.py مدل آموزش دیده بارگذاری میشود و چهره ای که در حال حاضر روبروی وب کم است تشخیص داده شده و اسم و میزان قطعیت در کنار چهره به نمایش در میاید.

نتایج :

وقتی کمی سر را میچرخاندم تشخیص دچار مشکل میشد. برای حل این مشکل میتوان تعداد عکس ها را زیاد کرد و کمی سر تا در طول عکاسی چرخاند.

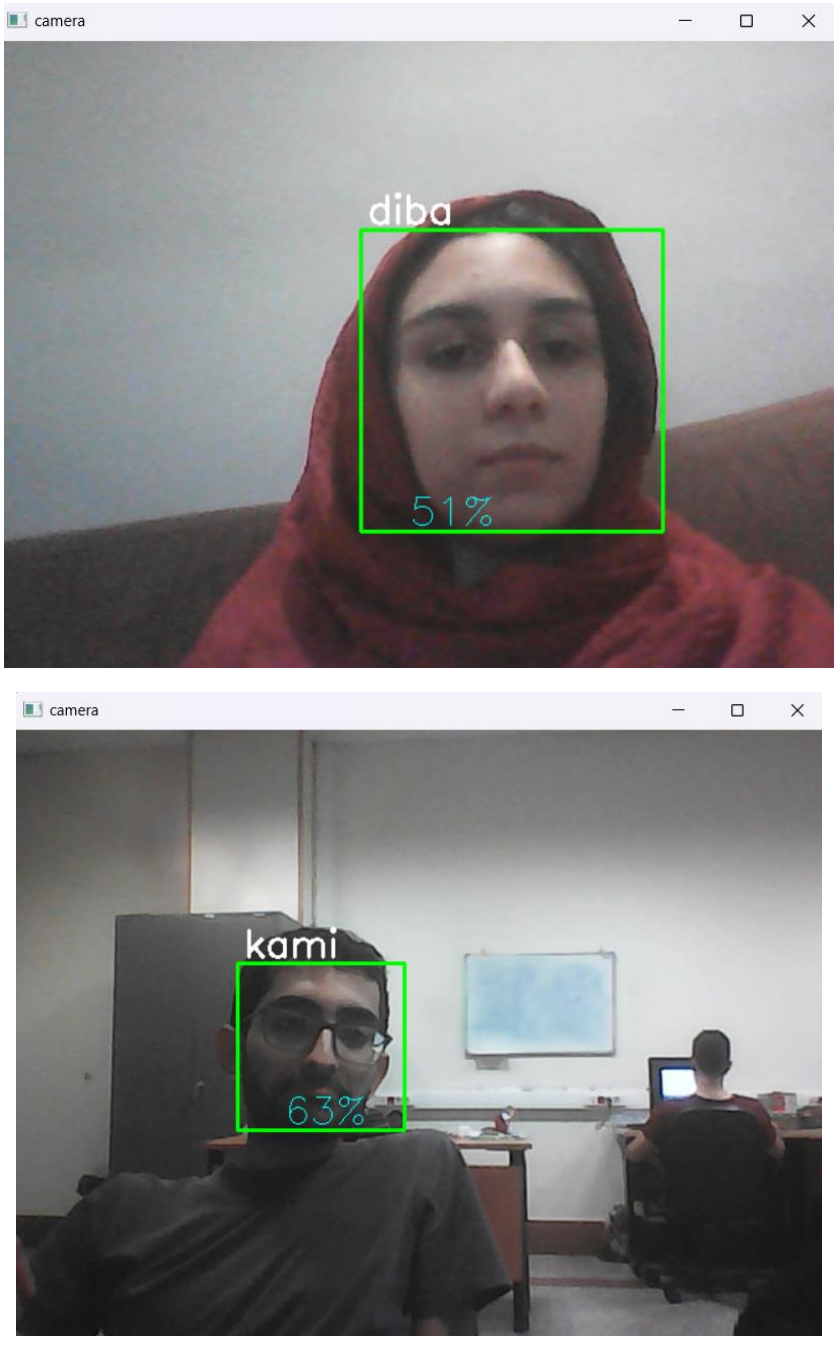


Figure 10تشخیص چهره به صورت **real-time**

Image to Sketch

توضیح فایل‌ها و کتابخانه‌های موردنیاز:

OpenCV: برای پردازش تصویر استفاده میشود.

توضیح در مورد الگوریتم استفاده شده:

خواندن تصویر : نام تصویر ورودی را به تابع `cv2.imread` میدهیم.

تبدیل رنگی به خاکستری :تصویر مورد نظر به تصویر خاکستری تبدیل می‌شود با استفاده از تابع `cv2.cvtColor`.

نات کردن تصویر : تصویر خاکستری با استفاده از تابع `cv2.bitwise_not` معکوس می‌شود.

اعمال بلور : بلور Gaussian به تصویر معکوس شده اعمال می‌شود.

معکوس کردن بلور : دوباره تصویر بلور شده معکوس می‌شود.

تقسیم تصویر خاکستری و بلور معکوس : برای ایجاد حالت طراحی شده، تصویر خاکستری و بلور معکوس با مقیاس 256 تقسیم می شوند. . این تقسیم باعث می‌شود تا جزئیات تصویر با وضوح بیشتری نمایش داده شوند و ظاهر طرح دستی ایجاد شود.

ذخیره تصویر نهایی :تصویر نهایی را ذخیره میکنیم.

نتایج :

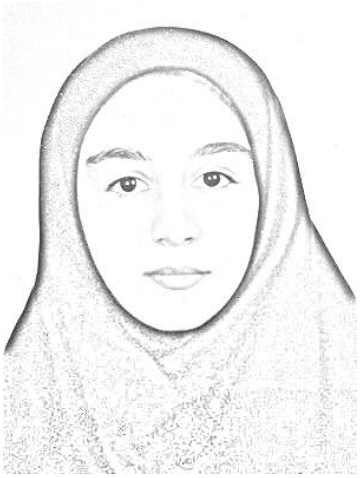


Figure 11 تصویر تبدیل شده به نقاشی و طراحی

QR code

توضیح فایل‌ها و کتابخانه‌های موردنیاز:

OpenCV: برای پردازش تصویر استفاده میشود.

توضیح در مورد الگوریتم استفاده شده:

در دو حالت از روی عکس و از روی وبکم میتوانیم این کار را انجام دهیم.

خواندن تصویر :تصویر ورودی با استفاده از تابع `cv2.imread` خوانده می‌شود.

ایجاد شیQRCodeDetector : یک شی از کلاس `QRCodeDetector` برای شناسایی و رمزگشایی کدهای QR ایجاد می‌شود.

شناسایی کد های QR : با استفاده از تابع `detectAndDecodeMulti`کدهای QR در تصویر شناسایی و رمزگشایی می‌شوند. این تابع اطلاعات رمزگشایی شده و نقاط مختصات کدهای QR را برمی‌گرداند.

چاپ نتایج :نتایج شناسایی و اطلاعات رمزگشایی شده چاپ می‌شوند. با استفاده از تابع `cv2.polylines`دور کدهای QR شناسایی شده مستطیل کشیده می‌شود اگر کدی شناسایی نشود، با رنگ قرمز و اگر شناسایی شود با رنگ سبز نمایش داده می‌شود. همچنین کد ها به صورت دو بعدی هستند و کد های یک بعدی با این کد قابل تشخیص نیستند. در اکثر کالاها کد ها به صورت یک بعدی بودند بنابراین محصولی در خانه نیافتم که با آن نتیجه را نمایش دهم.

ذخیره تصویر نهایی :تصویر نهایی با استفاده از تابع `cv2.imwrite` ذخیره می‌شود.

برای استفاده از وبکم:

باز کردن دوربین وب :با استفاده از تابع `cv2.VideoCapture`دوربین وب باز می‌شود.

خواندن فریم‌ها از دوربین وب :با استفاده از تابع `cap.read` فریم‌های ویدیو از دوربین وب خوانده می‌شوند.

شناسایی و رمزگشایی کدهای QR: با استفاده از تابع `detectAndDecodeMulti`کدهای QR در هر فریم شناسایی و رمزگشایی می‌شوند.

نمایش نتایج :نتایج شناسایی شده و اطلاعات رمزگشایی شده در تصویر نمایش داده می‌شوند.

بستن دوربین وب :با فشار دادن کلید 'q' برنامه خاتمه یافته و دوربین وب بسته می‌شود و تمام پنجره‌های OpenCV بسته می‌شوند.

نتایج :

تشخیص بارکد از روی عکس**Figure 12**

Figure 13تشخیص بارکد از روی وبکم

Remove Background

توضیح فایل‌ها و کتابخانه‌های موردنیاز:

این پروژه با استفاده از کتابخانه carvekit میتواند پس زمینه هر تصویری را حذف کند. این ابزار از چندین segmentation networks برای شناسایی اجسام در تصاویر و چندین روش پیش‌پردازش و پس‌پردازش برای بهبود دقت و کیفیت برش استفاده می‌کند

کتابخانه ها:

torch: برای استفاده از مدل‌های یادگیری عمیق

Ipython: برای نمایش تصاویر و مدیریت آپلود فایل‌ها

Google.colab: برای آپلود فایل‌ها در محیط Google Colab

Carvekit: برای انجام عملیات پیش‌پردازش، تفکیک‌سازی و پس‌پردازش تصاویر

توضیح در مورد الگوریتم استفاده شده:

این پروژه در محیط Google Colab و با استفاده از GPU ران میشود:

فعال کردن GPU:

برای استفاده از GPU در Google Colab ، به مسیر Edit → Notebook Settings میرویم و از منوی Hardware Accelerator، گزینه GPU را انتخاب میکنیم.

نصب کتابخانه ها و دانلود مدل‌ها:

مدل‌های مورد نیاز برای پردازش تصاویر را دانلود میکنیم. با دستور download_all() اینکار انجام میشود.

تنظیمات:

پس از نصب و دانلود مدل‌ها، تنظیمات پیش‌پردازش و پس‌پردازش را مشخص میکنیم.

آپلود و پردازش تصاویر:

خواسته میشود که تصاویر خود را آپلود کنیم.

نمایش نتایج:

نتایج پردازش به صورت تصاویر برش‌خورده و بدون پس‌زمینه نمایش داده می‌شود.

نتایج :

Figure 14 تصویر بدون پس زمینه

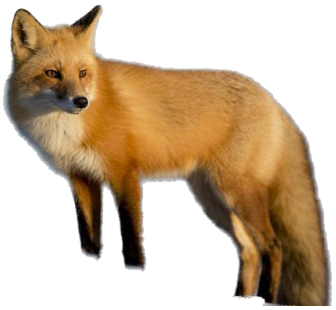


Figure 15 تصویر بدون پس زمینه روباه

Remove Shadow

توضیح فایل‌ها و کتابخانه‌های موردنیاز:

OpenCV و numpy و matplotlib را از قبل میشناسیم.

Scikit را باید نصب کنیم. برای استفاده از skimage.measure برای برچسب‌گذاری و اندازه‌گیری بخش های متصل به هم.

توضیح در مورد الگوریتم استفاده شده:

تابع median_filter : این تابع یک فیلتر میانه را بر روی یک نقطه خاص در تصویر اعمال می‌کند.

تابع edge_median_filter : این تابع فیلتر میانه را بر روی پیکسل‌های مرزی کانتورهای مشخص شده اعمال می‌کند.

تابع display_region : این تابع برای نمایش ناحیه‌های مختلف تصویر استفاده می‌شود و شامل تصویر اصلی، ناحیه سایه، ناحیه برش خورده و تصویر اصلاح شده است.

تابع correct_region_lab و correct_region_bgr: این توابع برای اصلاح ناحیه سایه با استفاده از میانگین مقادیر LAB یا BGR استفاده می‌شوند. این توابع ناحیه سایه را با ناحیه بدون سایه مقایسه می‌کنند و نسبت‌های مورد نیاز برای اصلاح رنگ را محاسبه می‌کنند.

تابع process_regions : این تابع نواحی مختلف تصویر را پردازش می‌کند و سایه‌ها را حذف می‌کند.

تابع calculate_mask : این تابع ماسک نواحی سایه را محاسبه می‌کند. این کار با تبدیل تصویر به فضای رنگی LAB و اعمال آستانه‌های مختلف انجام می‌شود.

تابع remove_shadows : این تابع ماسک نواحی سایه را محاسبه کرده و نواحی سایه را پردازش می‌کند تا سایه‌ها را حذف کند.

تابع process_image_file : این تابع تصویر را می‌خواند، سایه‌ها را حذف می‌کند و نتایج نهایی را نمایش می‌دهد و تصویر بدون سایه را ذخیره می‌کند.

نتایج :

عکس با سایه:

عکس بدون سایه:

تصویر بدون سایه**Figure 16**