بسمه تعالی

گزارش کار پروژه درس برنامه سازی پیشرفته

موضوع پروژه :

تشخیص اثر انگشت به وسیله پردازش تصویر و درست کردن محیط access control

نام و نام خانوادگی اعضای گروه :

رسول خزایی لکی

سینا محمودی

شماره دانشجویی:

۹۶۲۳۰۴۲

۹۶۲۳۰۹۸

استاد مربوطه :

دکتر جهانشاهی

تدریس یار مربوطه :

مهندس احسانی

تیر ماه ۱۳۹۸

فهرست

مقدمه و هدف از اجرای پروژه :

با توجه به افزایش روز افزون استفاده از تکنولوژی و نیاز مبرم به پروتکل هایی برای کنترل ورود و خروج افراد در یک مکان مشخص و مشخص کردن هویت این افراد که امروزه از آن به عنوان access control یاد می شود، بر آن شدیم تا در حد دانش خود برنامه پایه ای برای این که چنین کاری را به وسیله اثر انگشت هر شخص که منحصر به فرد خود اوست ، پیاده سازی کنیم و بتوانیم به وسیله آن مشخصات کاربران برنامه را در یک محیط نگه داری کرده و در صورت لزوم از آنها بهره ببریم.

نحوه کار با قسمت گرافیکی برنامه :

*در ابتدا باید فایل db.py اجرا شود تا دیتابیس برنامه ساخته شود. سپس از فایل main.py برای اجرای برنامه استفاده شود.

در ابتدا با اجرای برنامه، صفحه ای باز می شود که روی آن ۲ دکمه مشاهده می شود یکی برای دسترسی به قسمت admin و دیگری برای دسترسی به قسمت کاربران عادی است.

admin :

در این قسمت ابتدا کاربر باید از ((Admin)) به عنوان username و از ((admin)) به عنوان password برای ورود به محیط مدیریتی استفاده کند. پس از اینکه وارد محیط مدیریتی شد، در آنجا بو وسیله ابزارهای تهیه شده می تواند از یک فایل عکس اثر انگشت مشخصات افرادی را که مشخصاتی تقریبا نزدیک به آن عکس را دارد، به دست آورد. بدین صورت که تمامی فایل های موجود در برنامه آدرس شان در یک دیتابیس مشخص ذخیره شده است و با دادن آدرس فایلی که می خواهیم، آن را با هر یک از آنها مقایسه می کند و اگر مشابه بودند مشخصات فرد را در TABLE مشخص شده اضافه می کند. با دکمه دیگر کاربر میتواند به وسیله username فرد مورد نظر او را از لیست کاربران عادی برنامه حذف کند و با دیگری از قسمت مدیریتی خارج شود.

کاربران عادی :

در این قسمت کاربر می تواند به وسیله دکمه register و ثبت مشخصات خود و کلیک کردن دوباره روی دکمه register می تواند مشخصات خود را در دیتابیس برنامه ذخیره کند و پس از آن با وارد کردن مشخصات خود وارد محیط برنامه شود. در محیط برنامه کاربر می تواند رمز عبوری را تغییر داده و همچنین عکس جدید به دیتابیس اضافه کند. هر کاربر توانایی اضافه کردن فقط یک فایل به عنوان اثر انگشت دارد.

الگوریتم پردازش تصویر :

نازک سازی :

در این قسمت ما از کتابخانه های cv2 و numpy استفاده می کنیم.

ابتدا عکس را به صورت سیاه و سفید خوانده که روشنایی پیکسل ها را از ۱ تا ۲۵۶ میکند و عمل cv2.threshold(img, 127, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU) را انجام می دهیم که با آستانه ۱۲۷ عکس سیاه و سفید را به باینری تیدیل می کند.

حال این پیکسل های ۰ تایپ float را به int تبدیل میکنیم و در thinned_thresh کپی میکنیم.

الگوریتم نازک سازی :

این الگوریتم بر روی پیکسل های سیاه با هشت همسایه عمل می کند. این به این معنی است که پیکسل هایی که در مرز و گوشه های تصویر یافت می شوند، تحلیل نمی شوند. برای آن پیکسل هایی که تحلیل می شوند، سفارش زیر نشان داده شده است P1 پیکسل سیاه که مورد تجزیه و تحلیل قرار گرفته است.

P9	P2	P3
P8	P1	P4
P7	P6	P5

حال دو مقداری را که در مراحل بعدی الگوریتم استفاده می کنیم تعریف کنیم:

A(p1) :

تعداد صفر به یک (سفید به سیاه) عوض شده در دنباله P2-> P3-> P4-> P5-> P6-> P7-> P8-> P9-> P2. است. با نگاهی به شکل ۲، می بینیم که ما با چرخ زنی در جهت عقربه های ساعت اطراف P1، و به عنوان یک نتیجه P2 باید دو بار ظاهر شود (P3 <-P2 و P2 <-P9 انتقال)

B(p1) :

تعدادی از همسایگان سیاه و سفید در اطراف P1. به عبارت دیگر، چه تعداد از P2 تا P9 سیاه است؟

مرحله اول :

برای این مرحله، ما به دنبال پیکسل های P1 هستیم که پنج شرایط را برآورده می کنند. اگر پیکسل P1 مطابق پنج شرایط باشد، آن را به سفید تعریف می کند. پیکسل ها بعد از اینکه تمام پیکسل های واجد شرایط در تصویر قرار گرفته اند، به سفید تبدیل می شوند، در غیر این صورت همگرایی در آرایه نازک وجود نخواهد داشت.

شرایط :

۱. پیکسل سیاه باشد و دارای ۸ همسایه. pixel_is_black(arr, x, y)

۲. عداد پیکسل های سیاه و سفید همسایه برای P1 حداقل 3 و حداکثر 8 است.
pixel_has_2_to_6_black_neighbors(arr, x, y)

۳. تعداد انتقالهای سفید به سیاه در اطراف P1 برابر است با ۱. یعنی A (P1) = 1.
pixel_has_1_white_to_black_neighbor_transition(arr, x, y)

۴. حداقل یکی از P2، P4 یا P6 سفید است (برابر با ۰).
at_least_one_of_P2_P4_P6_is_white(arr, x, y)

۵. حداقل یکی از P4، P6، یا P8 سفید است (برابر با ۰).
at_least_one_of_P4_P6_P8_is_white(arr, x, y)

تمام پیکسل های P1 که با پنج معیار فوق مطابقت دارند، به سفید تبدیل میشوند (۰).

مرحله ۲ :

سه قسمت اول مرحله دوم با مرحله اول یکسان است. قسمت چهار و پنج کمی تغییر می کنند - P6 به قسمت P8 در قسمت چهارم تغییر می کند و P4 در قسمت پنج به P2 تغییر می کند.

شرایط ۱و ۲و ۳ همان شرایط مرحله اول است.

۴. حداقل یکی از P2، P4 یا P8 سفید است (برابر با ۰).
at_least_one_of_P2_P4_P8_is_white(arr, x, y)

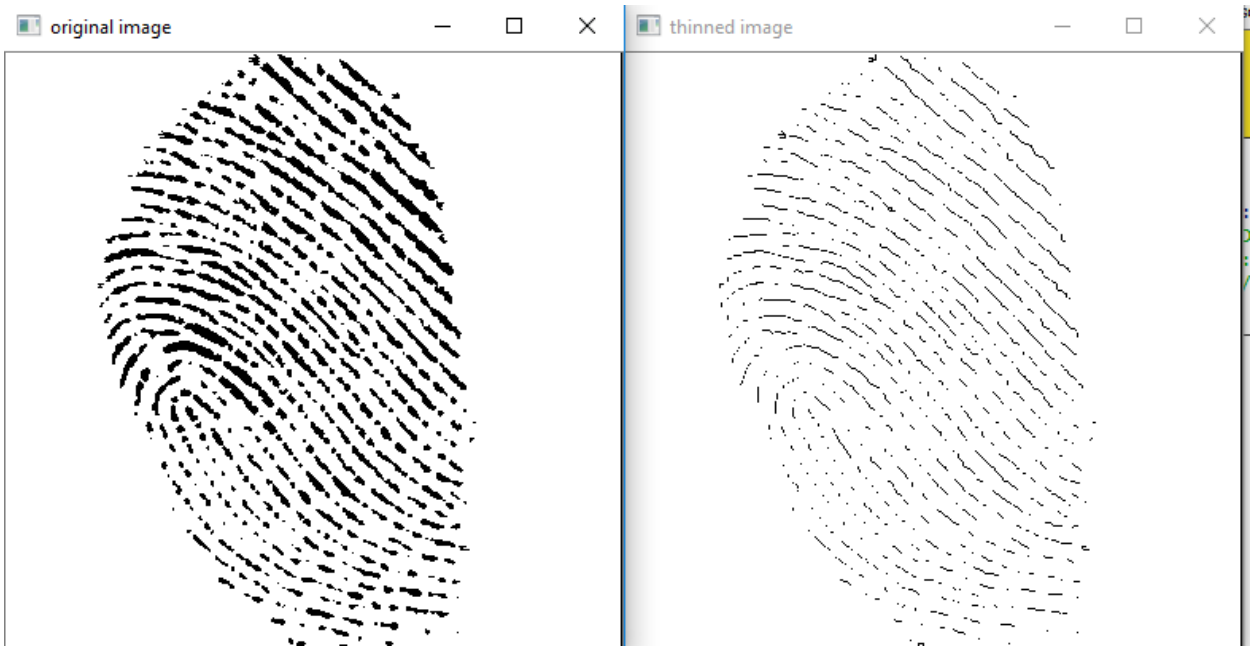
۵. حداقل یکی از P2، P6، یا P8 سفید است (برابر با ۰).
at_least_one_of_P2_P6_P8_is_white(arr, x, y)

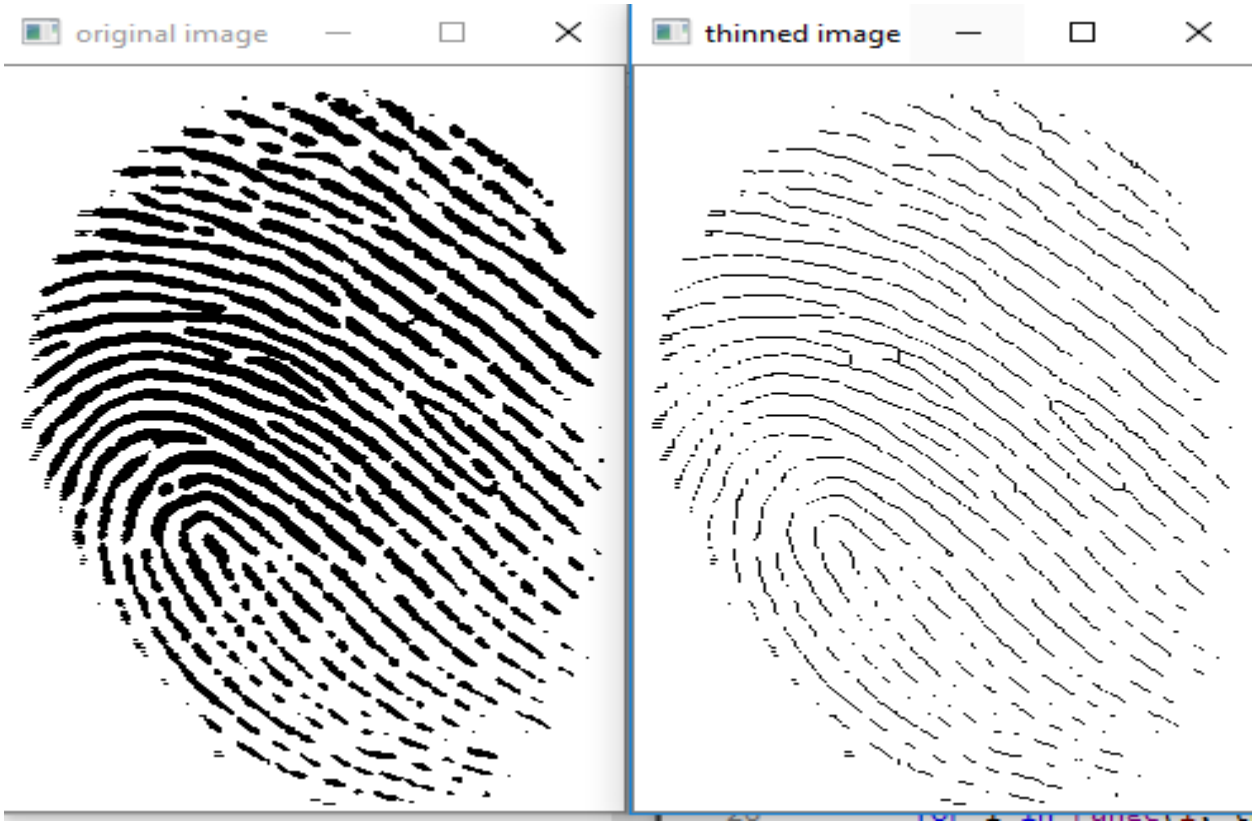
حال برای هر یک از شرایط بالا تابع تعریف میکنیم و آنها را در ماژول check.py می ریزیم .

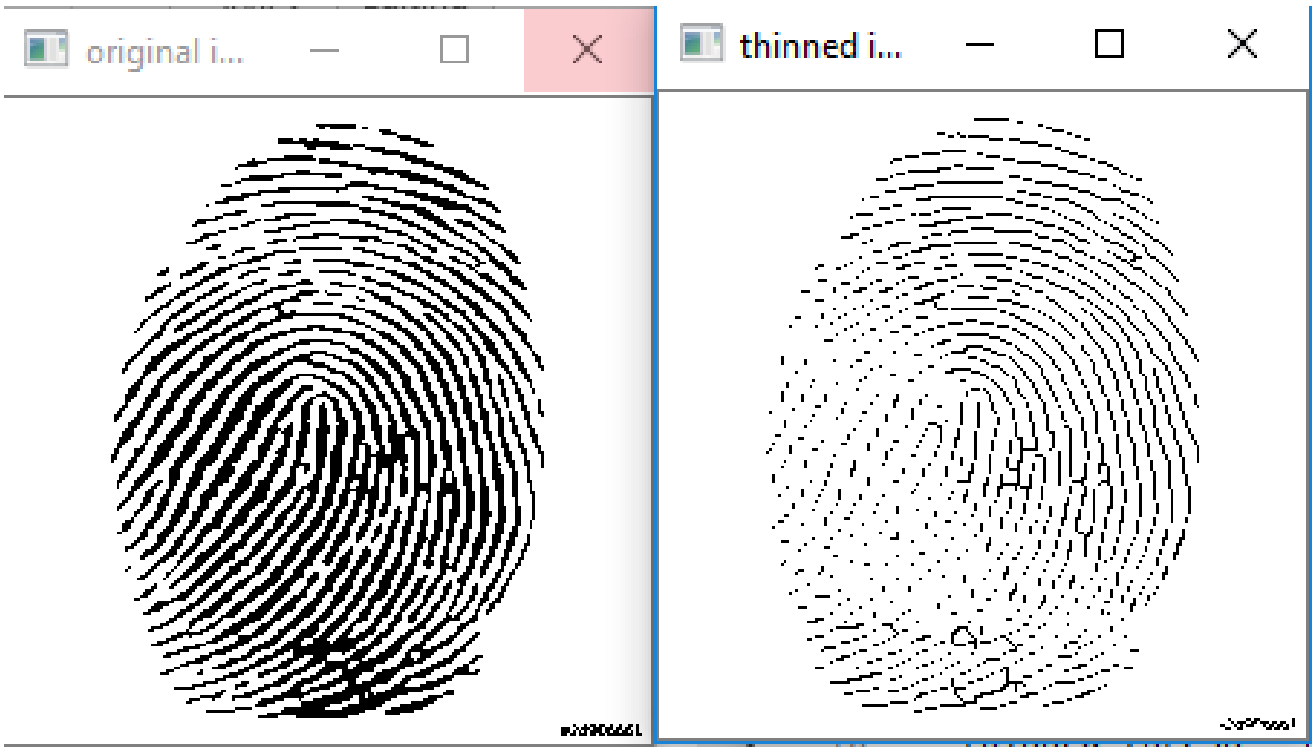
حال برای thinning در ماژول make_thin الگوریتم نازک یازی را پیاده سازی می کنیم.

حالا که ما توابع توصیف شرایط الگوریتم را ایجاد کرده ایم می توانیم از طریق پیکسل های تصویر تکرار کنیم و هر پیکسل را در برابر شرایط و چرخش تا زمانی که همگرایی رسیده است، بررسی کنیم.

نمونه ای از خروجی ها :





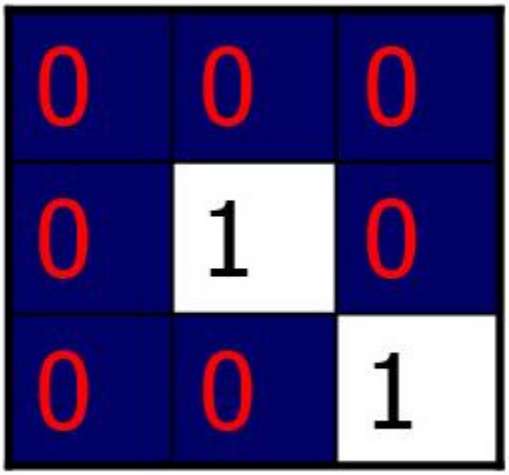

original i...
thinned i...

الگوریتم پردازش اصلی :

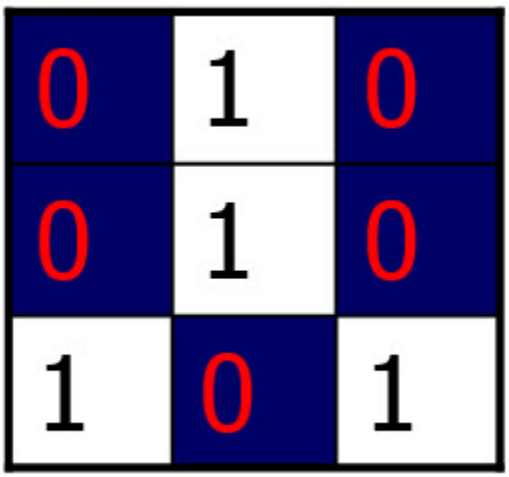
اکنون پس از نازک سازی الگوریتم پردازش اصلی را شروع می کنیم.

اولین کار بعد از نازک سازی پیدا کردن نقاط خاص اثر انگشت است برای این کار نقطه هایی از اثر انگشت مشخص می شوند که :

۱. نقاط انتهایی هستند:



۲. مرکز دوشاخگی هستند:



برای نقاط انتهایی پیکسل هایی را انتخاب می کنیم که خود سیاه و دارای ۱ همسایه سیاه هستند

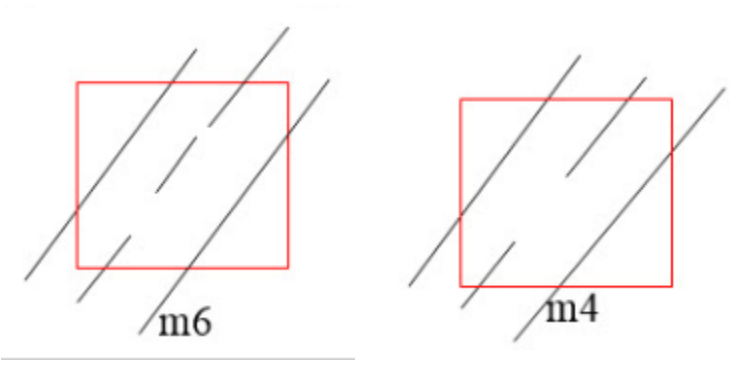
این کار را با تابع make_1_angle که در ماژول check.py قرار دارد انجام میدهیم.

برای دوشاخگی ها هم پیکسل هایی را مشخص می کنیم که خود سیاه و دارای ۳ همسایه سیاههند . این کار را با تابع make_3_angle که در ماژول check.py قرار دارد انجام میدهیم.

این دو تابع را در ماژول find_end فراخوانی کرده و نقاط خاص را با نقاط سیاه و بقیه را با سفید مشخص کرده و در یک آرایه f هم سایز با عکس اصلی ذخیره می کنیم.

با این الگوریتم نقاط انتهایی اطراف اثر انگشت نیز انتخاب می وند که مد نظر ما نیستند پس باید اصلاح شوند پس برای این کار نقاطی که در سمت چپ و یا راست آنها بیش از ۲۵ پیکسل سفید موجود باشد آن ها را حذف میکنیم و یعنی عدد پیکسل در نظر گرفته شدده را سفید میکنیم.

علاوه بر این نقاط دیگری نیز هستند که به اشتباه انتخاب می شوند :



برای حل این مشکل نیز پیکسل هایی که به عنوان نقاط انتهایی انتخاب شده اند و فاصله آن ها کمتر از ۱۰ پیکسل باشد را نیز در آرایه f به سفید تبدیل می کنیم.

برای چک کردن مطابقت دو اثر انگشت نیز بعد از به دست آوردن آرایه های f مربوط به هر کدام ،

توسط تابع matching تعداد نقاطی که باهم مطابقت دارند را به دست می آوریم.

ممکن است که دو عکس به اندازه ۱۵ پیکسل نسبت به حالت قبلترش جا به جا شده باشد(ماکزیمم تعداد پیکسل انتقال داده شده را ۱۵ پیکسل در نظر می گیریم)

یعنی که نقاط خاصش به اندازه ۱۵ پیکسل در جهات مختلف جا به جا شده باشد پس در این الگوریتم این ویزگی را نیز اعمال میکنیم. و تعداد نقاط مطابق را بر میگردانیم.

تابع load نیز برای نشان دادن تصویر دو آرایه به دست آمده برای دو عکس پردازش شده نوشته شده که در اینجا خروجی در نظر گرفته نشده.

طرز کار برنامه در قسمت پردازش تصویر :

تابع comp دو آدرس دریافت می کند تا دو عکس را با هم مقایسه کند سپس تابع process را دو بار و هر بار با یکی از آدرس ها فراخوانی می کنیم که آرایه f بحث شده در صفحه قبل را برمی گرداند.

برای مقایسه دو آرایه ماکزیمم تعداد پیکسل های دو عکس را در count ذخیره می کنیم.

و تابع matching را با دو آرایه به دست آمده فراخوانی می کنیم که تعداد پیکسل های منطبق دو عکس را بر میگرداند .

حال اگر نسبت تعداد نقاط تطبیق داده شده به count بیشتر از ۰,۵ باشد تابع comp ۱ را برمیگرداند یعنی که دو عکس یکی هستند و در غیر این صورت ۰ برمیگرداند.

*به علت این که مرحله های 4 , 3 که در تعریف پروژه آمده بودند ضروری نیستند و سرعت برنامه را کاهش می دهند این مراحل از پروژه تعریف شده و با نظر تدریس یار حذف شد .