بسم الله الرحمن الرحیم

تشخیص حروف به کمک پردازش تصویر

سارا قائمی ، امین انصاریان

۱۱ تیر ۱۳۹۵

۱ فاز اول: تشخیص مشت و رسم اولیه به کمک OpenCV

در مرحله اول بر روی Anaconda راه اندازی OpenCV

انجام شد. برای هرگونه تشخیص دست یا مشت در پردازش تصویر، چندین راه وجود دارد، یکی از راه حل های ابتدایی، توجه به رنگ پوست دست می باشد. رنگ پوست خصیصه بسیار خوبی برای تشخیص دست میباشد. اگر در تصویر ورودی از دوربین کامپیوتر با استفاده از فیلتر های خاص، قسمت هایی که رنگ دست دارد را به رنگ سفید، و بقیه قسمت ها را به رنگ مشکی نشان دهیم، میتوان به راحتی دست را در تصویر تشخیص داد. اما این روش مشکل های متعددی دارد. مهم ترین مشکل این روش، همرنگ بودن صورت انسان با دست او میباشد. اگر در تصویر دوربین، صورت شخص هم حاضر باشد، پس از گذشتن از فیلتر، صورت هم به رنگ سفید نمایش داده خواهد شد. مشکل دوم این روش این است که باید لباس آستین دار به تن داشت. البته در این روش با استفاده از بعضی فیلتر ها، با توجه به بزرگتر بودن سر از دست، قادر به تشخیص دست از سر شده اند. ولی اگر دست به تصویر نزدیکتر باشد این روش هم با مشکل مواجه میشود. خروجی فیلتر تشخیص رنگ بدن که توسط افراد مختلف پیاده سازی شده است را در شکل ۱ و ۲ میبینید.

روش استفاده شده در این پروژه، استفاده از تعداد زیادی از لایه های پست سر هم قرار گرفته از classifier های ضعیف دست میباشد.

شکل ۱: تشخیص رنگ بدن

شکل ۲: تشخیص رنگ بدن

۱.۱ فایل های xml

برای تشخیص مشت در یک تصویر دوربین میبایست تعداد زیادی نمونه عکس که شامل نمونه های درست و غلط میباشد توسط الگوریتم های machine learning آموزش داده شود(train شود). منظور از نمونه های درست عکس هایی هستند که مشت در آن ها وجود دارد و منظور از نمونه های غلط عکس هایی هستند که مشت در آن ها وجود ندارد.

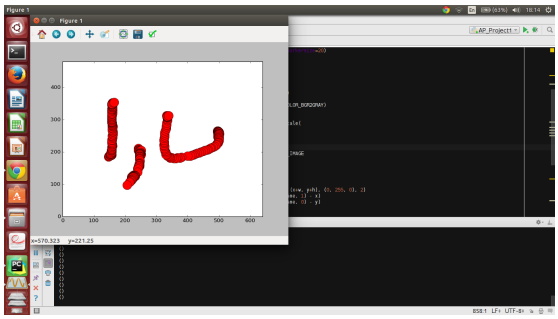
برای train تشخیص دهنده(فایل xml) مجموعه ای شامل حدود ۵۰۰ عکس مشت از اعضای آزمایشگاه تهیه شد و در تمامی این عکس ها، ناحیه دست به دقت مشخص گردید. سپس مجموعه شامل حدود ۵۰۰ عکس از محیط آزمایشگاه و اطراف آزمایشگاه و صورت ها و لباس ها و رنگ های مختلف بدون حضور هیچ دستی تهیه شد.

در این پروژه از این فایل برای تشخیص مشت استفاده شد.

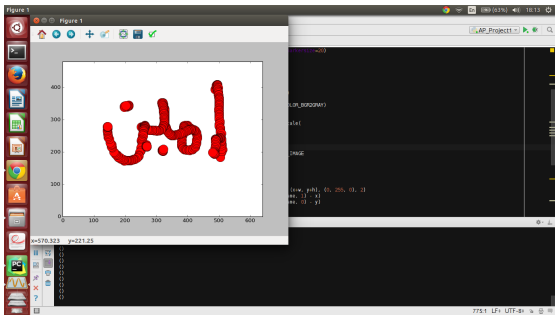
۲.۱ تشخیص مشت

بر روی فایل xml اشاره شده در زیربخش قبلی، یک CascadeClassifier اجرا میشود. CascadeClassifier از تعدادی مرحله تشکیل شده است که هر مرحله شامل گروهی از تشخیص دهنده های ضعیف است. این تشخیص دهنده های ضعیف، طبقه بندی های ساده ای هستند که میتوانند با درصد صحت پایین، وجود و یا عدم وجود شی مورد نظر در عکس را مشخص میکند. هر مرحله از CascadeClassifier توسط تکنیکی موسوم به Boosting آموزش داده میشوند. Boosting قابلیت آموزش یک تشخیص دهنده قوی از تعدادی تشخیص دهنده های ضعیف را حاصل میکند. Boosting با استفاده از میانگین گیری وزن دار از تشخیص های ضعیف گرفته شده، یک تشخیص قوی تر درباره حضور و یا عدم حضور شی مورد نظر در تصویر میدهد.

باید توجه داشت که برای پیاده سازی اکثر الگوریتم های پردازش تصویر ابتدا تصویر را سیاه و سفید میکنند که به این منظور OpenCV خود تابعی را در اختیار قرار داده است.



شکل ۳: نمونه اجرای فاز ۱

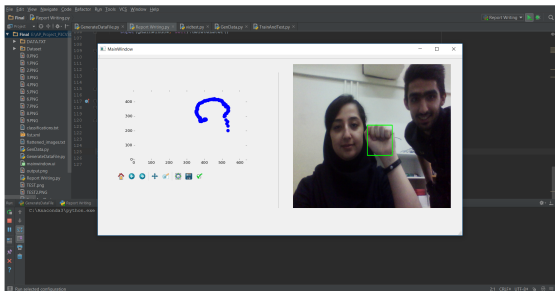


شکل ۴: نمونه اجرای فاز ۱

پس از تغییر رنگ تصویر، تابع detectMultiScale بر روی هر frame تصویر اجرا میشود. شیوه کار این تابع به این صورت است که تصویر را به نواحی مستطیلی که با هم همپوشانی دارند تقسیم کرده و با این کار تصویر را جارو میکند. که نتیجه آن مختصات مستطیل خروجی ای است که بالاترین شباهت را به مشت دارد. این تابع پارامتر هایی قابل تنظیم برای عملکرد خود در اختیار برنامه نویس قرار میدهد. از مهم ترین این پارامتر ها میتوان به scaleFactor که مقیاس کاهش اندازه در تصویر میباشد، minNeighbors که حداقل تعداد همسایه برای هر مستطیل را مشخص میکند، minSize که حداقل اندازه مستطیل برای اینکه مشت باشد را میدهد اشاره کرد.

در انتهای برای نشان دادن مشت های تشخیص داده شده با استفاده از خروجی تابع فوق مستطیلی به دور مشت رسم میشود. در فاز اول فقط این مختصات ها ذخیره شده و سپس در یک نمودار جداگانه با استفاده از کتابخانه matplotlib رسم میشود.

دو نمونه از خروجی کد فاز اول در شکل ۳ و ۴ آمده است



شکل ۵: GUI

۲ فاز ۲: رسم بلادرنگ نمودار و GUI

در این فاز بخش پردازش تصویر تغییری نکرده و فقط به رسم بلادرنگ(realtime) نمودار پرداخته شده است. همچنین در محیط Qt Designer یک GUI (Graphical User Interface) یا رابط کاربری گرافیکی طراحی شد.

۱.۲ GUI

برای طراحی GUI برای نشان دادن تصویر هر frame از یک label استفاده شده است. همچنین برای بخش نمودار هم از label استفاده شده است. نحوه استفاده از هرکدام در بخش کد نمودار بلادرنگ آمده است. نمونه ای از عکس این رابط کاربری در شکل ۵ آمده است.

۲.۲ Plot Realtime

برای اینکه بتوان به صورت بلادرنگ نمودار نقاط مشت ها را رسم نمود باید بخش نمودار را در یک Thread جداگانه انجام داد.

کد فاز دوم در کل شامل ۳ کلاس اصلی میباشد. یکی کلاس ControlWindow که در واقع کلاس مربوط به GUI است و کلاس اصلی محسوب میشود، کلاس دوم کلاس QtCapture که در آن پردازش های مربوط به پردازش تصویر صورت میگیرد، و آخری کلاس PlotThread که کار رسم نمودار بلادرنگ را انجام میدهد. در این پروژه برای رسم بلادرنگ نمودار همان طور که در کلاس به تفضیل توضیح داده شده بود عمل شده است. نکته قابل توجه در این بخش از کد آن است که برای دسترسی به مختصات های مشت ها چون در QtCapture هستند در ControlWindow برای رسم نمودار دسترسی وجود نداشت. به این منظور از مفهوم Pass by Reference استفاده شده است و با توجه به اینکه list در پایتون mutable هستند بنابراین Pass by Reference میشوند.

شکل ۶: نمونه حرف A

۳ فاز سوم: تهیه بانک داده ، آموزش آن و تشخیص حروف و اعداد

پس از رسم بلادرنگ نمودار در فاز قبل حال به سراغ بخش تشخیص متن از روی دست خط می رویم، شیوه ی کلی کار به این صورت است که در ابتدا با استفاده از الگوریتم های یادگیری ماشین(Machine Learning) یک بانک داده از نمونه هایی که توسط دست کشیده شده اند، تهیه می شود. سپس با استفاده از آموزش این بانک (dataset) متن از روی نوشته تشخیص داده می شود که به تفصیل بیان خواهد شد.

۱.۳ طراحی دیتاست،تهیه دیتاست

ابتدا مجموعه بزرگی از نمونه های مختلف برای حروف و اعداد تهیه شد. در شیوه ی معمول به این صورت می باشد که به اندازه ۱۰ برابر تعداد پیکسل های هر نمونه ی مورد بررسی (در اینجا ۱۰*۲۸*۲۸) نمونه برای دقت بالای کار نیاز می باشد که به علت حجم بالای آن و ضیق وقت، تنها به تعدادی از حروف (A,B,E,H,I,M,N,O,P,R,S,Z) و اعداد، در تعداد نمونه در حدود ۶۰ عدد اکتفا شد که نمونه ی آن برای حرف A در شکل ۶ آمده است.

حال به بررسی شیوه تهیه دیتاست می پردازیم

پس از اعمال فیلتر سیاه و سفید بر روی عکس آن را از یک فیلتر عبور می دهیم به صورتی که تنها رنگ سفید از آن خارج شود، سپس با مشتق گرفتن از عکس، لبه های آن خارج شده و درواقع به دور هرکدام از آیتم های بالا یک کانتور(حلقه بسته) کشیده می شود، از این کانتور ها برای train کردن استفاده می کنیم.

دیتاست مورد نظر از دو بخش ماتریس های حروف(flattened images) و بخش بندی های مختلف(classification) تشکیل شده است.

حال با استفاده از یک حلقه ، برای هر کانتور از کاربر حرف مورد نظر گرفته شده و سپس به آن نسبت داده می شود. باید در نظر داشت که هرکدام از عکس های کانتور ها را برای افزایش دقت، تغییر اندازه داده و کوچک می کنیم.

سپس هر کدام از کلید ها را را در فایل(classification) به عنوان عدد اسکی آن ذخیره می نماییم و ماتریس عکس های تنظیم شده مورد نظر را نیز در (flattened images) ذخیره می نماییم که در نهایت این

شکل ۷: تهیه دیتاست

دو در دو فایل متنی ذخیره می شوند و دیتاست ما آماده ی
شدن می باشد. یک مرحله از تهیه دیتا ست به صورت شکل ۷ می باشد.

۲.۳ Training

در این مرحله از الگوریتم ریاضی نزدیک ترین همسایه برای train کردن استفاده می کنیم که در ابتدا به اختصار به توضیح آن می پردازیم

نزدیک‌ترین همسایه K Nearest Neighbor (KNN) :

روش k نزدیک‌ترین همسایه یک گروه شامل k رکورد از مجموعه رکوردهای آموزشی که نزدیک‌ترین رکوردها به رکورد آزمایشی باشند را انتخاب کرده و بر اساس برتری رده یا برچسب مربوط به آن‌ها در مورد دسته رکورد آزمایشی مزبور تصمیم‌گیری می‌نماید. به عبارت ساده‌تر این روش رده‌ای را انتخاب می‌کند که در همسایگی انتخاب شده بیشترین تعداد رکورد منتسب به آن دسته باشند. بنابراین رده‌ای که از همه رده‌ها بیشتر در بین k نزدیک‌ترین همسایه مشاهده شود، به عنوان رده رکورد جدید در نظر گرفته می‌شود. ایده اصلی روش KNN این است که اگر موجودی مثل اردک راه برود و مثل اردک quack quack کند، پس حتما یک اردک است.

استفاده از الگوریتم KNN نیازمند تعیین سه موضوع می‌باشد:

باید یک مجموعه رکورد داشته باشیم.

ـ یک معیار محاسبه شباهت نیز باید داشته باشیم.

ـ مقدار k نیز باید مشخص شود تا بتوان بر اساس آن عمل نمود. برای مسائل دسته بندی دودویی معمولا در نظر گرفتن مقادیر فرد برای k بهتر است. زیرا امکان پیروز شدن یکی از دو دسته را افزایش می‌دهد. برای مسائل رده‌بندی چند رده‌ای باید عدد k را بزرگ‌تر از تعداد رده ها و نیز متفاوت با عدد تعداد رده‌ها از نظر زوج یا فرد بودن در نظر گرفت. یعنی اگر تعداد رده‌ها زوج باشدباید k نهایی را فرد در نظر گرفت و بالعکس.

در رده‌بندی‌های KNN برای دسته‌بندی کردن یک رکورد با دسته نامشخص به صورت زیر عمل می‌شود:

```
kNearest = cv2.ml.KNearest_create()

kNearest.train(npaFlattenedImages, cv2.ml.ROW_SAMPLE, npaClassifications)
```

شکل ۸: آموزش

ـ فاصله رکورد جدید از همه رکوردهای آموزشی محاسبه می‌شود.

ـ نزدیک‌ترین همسایه‌ها مشخص می‌شوند.

ـ از برچسب دسته k نزدیک‌ترین همسایه، برای پیش‌بینی دسته رکورد جدید استفاده می‌شود. به این صورت که بین k رکورد رأی‌گیری شده و دسته‌ای که بیش‌ترین تعداد دفعات دیده‌شدن را در بین این k رکورد داراست، به عنوان دسته رکورد جدید در نظر گرفته خواهد شد. انتخاب مقدار k در این روش دسته‌بندی بسیار مهم و کلیدی است. اگر مقدار k خیلی کوچک انتخاب شود، الگوریتم به نویز حساس می‌شود. در واقع نویزها نزدیک آن رکورد ممکن است ایجاد اشتباه کنند. اگر مقدار k خیلی بزرگ انتخاب شود، ممکن است در میان نزدیک‌ترین همسایه‌ها، رکوردهایی از دسته‌های دیگر نیز قرار بگیرند.

در ادامه ی کار از ماژول (ml) Machine Learning پایتون ، یک شی KNN به صورت زیر ایجاد می کنیم و با استفاده از تابع ()train به سان شکل ۸ ، یادگیری ماشین را بر روی دو فایل مورد نظر که در بالا تولید شد انجام می دهیم.

در اینجا دیتاست نهایی برای تست کردن روی عکس ها آماده می باشد.

۳.۳ تشخیص متن

بعد از خواندن عکس مورد نظر برای تست کردن و اعمال کردن فیلتر های سیاه سفید، گوسی و سپس عبور دادن رنگ های سفید، کانتور های موجود در عکس را استخراج می نماییم و به آن ها مشخصات (x,y,width,height) را برای استفاده بهینه تر نسبت می دهیم. سپس با استفاده از یک حلقه روند تشخیص را شروع می نماییم

نکته: ابتدا برای پردازش روی عکس های نوشته شده با دست، یک مستطیل دور کانتور های درون عکس می کشیم که نمایش دهنده آیتم های آن می باشد و در یک پنجره آن را نمایش می دهیم، که در عکس زیر آورده شده است، اما در نهایت رابط کاربری از کد حذف شده اند.

حال به ترتیب با کوچک کردن عکس، تبدیل آن به ماتریس یک بعدی مقادیر صحیح (برای استفاده در الگوریتم تشخیص) و سپس تبدیل حالت قبل به مقادیر float آن را به ورودی الگوریتم KNN می دهیم که خروجی اسکی کاراکتر را به ما می دهد که به سان شکل ۹ آن رابه یک رشته تبدیل میکنیم که بعد از خروج از حلقه کانتور ها تمامی مقادیر رشته ای را به هم چسبانده و آن را بر میگرداند.مثالی از این برنامه در شکل ۱۰ آمده است.

```
strCurrentChar = str(chr(int(npaResults[0][0])))
```

شکل ۹: کاراکتر خروجی

شکل ۱۰: رشته خروجی نهایی