به نام خدا

گزارش پروژه پایانی درس بینایی ماشین

عنوان: تایپ چشمی

استاد: آقای دکتر محمدی

بنفشه کریمیان- عطیه سروی

مقدمه

در ابتدا تلاش کردیم از روش‌های Canny و Hue برای تشخیص مردمک چشم استفاده کنیم تا بیضی چشم را به دست آوریم اما متاسفانه به علت پایین بودن کیفیت وب کم این روش‌ها موفق نبود.

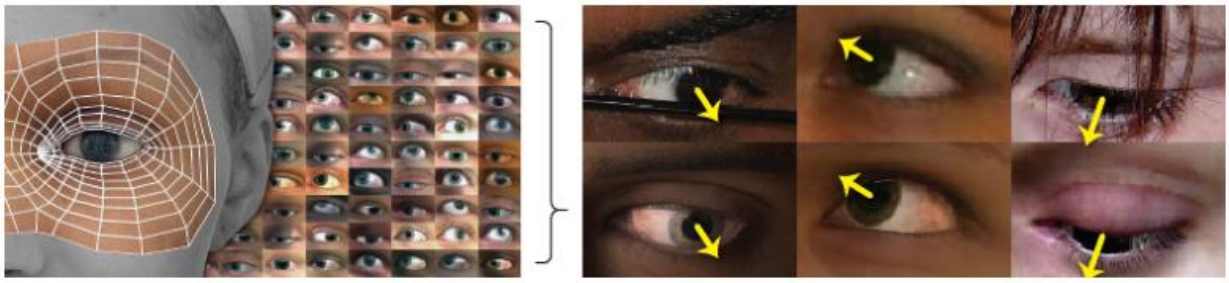
سپس تصمیم به استفاده از شبکه‌های کاانولوشنی گرفتیم. همان‌طور که بدیهی است, برای تعلیم مدلی که بتواند به طور خودکار مردمک چشم را در تصویر پیدا کندنیاز به دیتاست داریم. برای این پروژه ما از دو دیتاست استفاده کردیم که در ادامه به بررسی آن‌ها می‌پردازیم.

دیتاست اول:

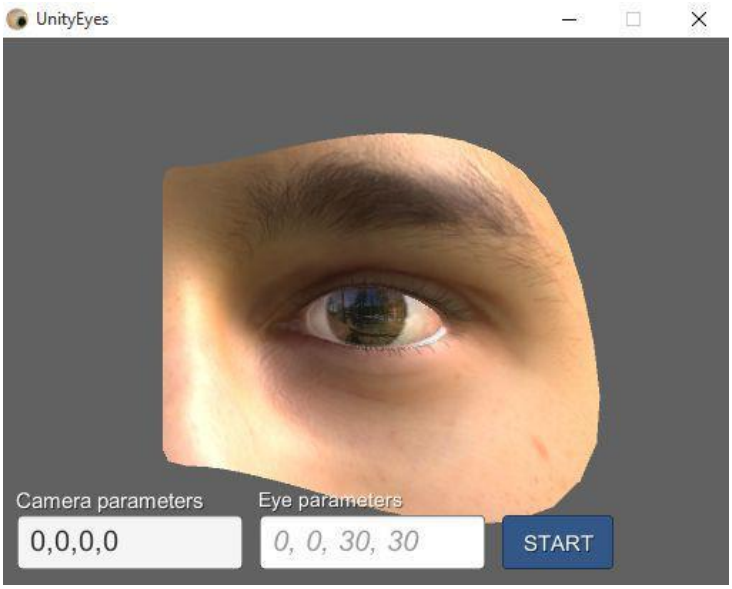
دیتاست اول مربوط به یک برنامه از دانشگاه کمبریج بود که با unity پیاده‌سازی شده بود و با گرفتن پارامترهایی مانند زاویه سر و میزان چرخش مردمک چشم تصویر چشم مصنوعی تولید می‌نمود. با استفاده از این برنامه 40 هزار تصویر چشم مصنوعی به عنوان دیتاست تولید کردیم.

لینک نرم‌افزار تولید این دیتاست:

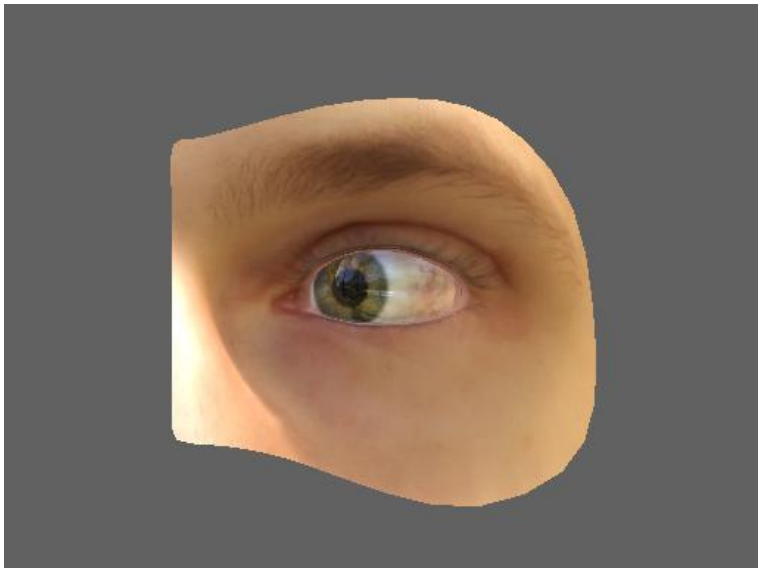
https://www.cl.cam.ac.uk/research/rainbow/projects/unityeyes/



عکس‌های مربوط به دیتاست اول



در شکل بالا camera parameter برابر با 0 است. به این معنی که دوربین از مستقیم تصویر برداری می‌کند که گویا سر فرد حرکت نمی‌کند. همچنین eye Parameters نشان‌دهنده‌ی میزان چرخش چشم است.



نمونه‌ی خروجی

سپس پیش پردازش‌هایی انجام دادیم که فقط ناحیه چشم را را استخراج کند. (ناحیه های بریده شده و سیاه نامنطم را حذف و اندازه‌ی تصاویر را متناسب کردیم). در جستجوهای انجام شده‌ی خود به مقاله‌ای از دانشگاه استنفرد (convolutional neural network for eye tracking algorithm) برخوردیم که مراحل کار این مقاله به صورت ذیل شرح داده شده بودند:

ابتدا پس از پیدا کردن چشم تصویر میبایست نرمال شود تا از تاثیر نور و سایر عوامل خارجی کاسته شود، سپس تصاویر میبایست reshape شوند تا شکل یکسان و متاسبی داشته باشند. نرمال شده‌ی یک تصویر از تفاضل آن تصویر از میانگین و تقسیم کردن نتیجه بر انحراف معیارآن تصویربدست می‌آید. در مرحله بعد دسته‌بندی تصاویر با استفاده از اندازه‌گیری فاصله تصاویر از یکدیگر انجام می‌شود. سپس تصاویری با بیشترین فاصله از باقی تصاویر را یافته و به آن‌ها عددی اختصاص می‌دهدیم. این عدد در مرحله بعد به عنوان لیبل استفاده می‌شود به گونه‌ای که برای هر تصویر نزدیک ترین تصویر بین تصاویر منتخب مرحله قبل پیدا شده و لیبل مربوطه به عنوان لیبل این تصویر استفاده میشود. این مقاله یک ساختار برای شبکه عصبی نیز پیشنهاد داده‌است. ما در این پروژه پس از انجام مراحل فوق تصاویر را با سه نوع لیبل به CNNای هم ساختار با شبکه‌ی مقاله دادیم. 1) به

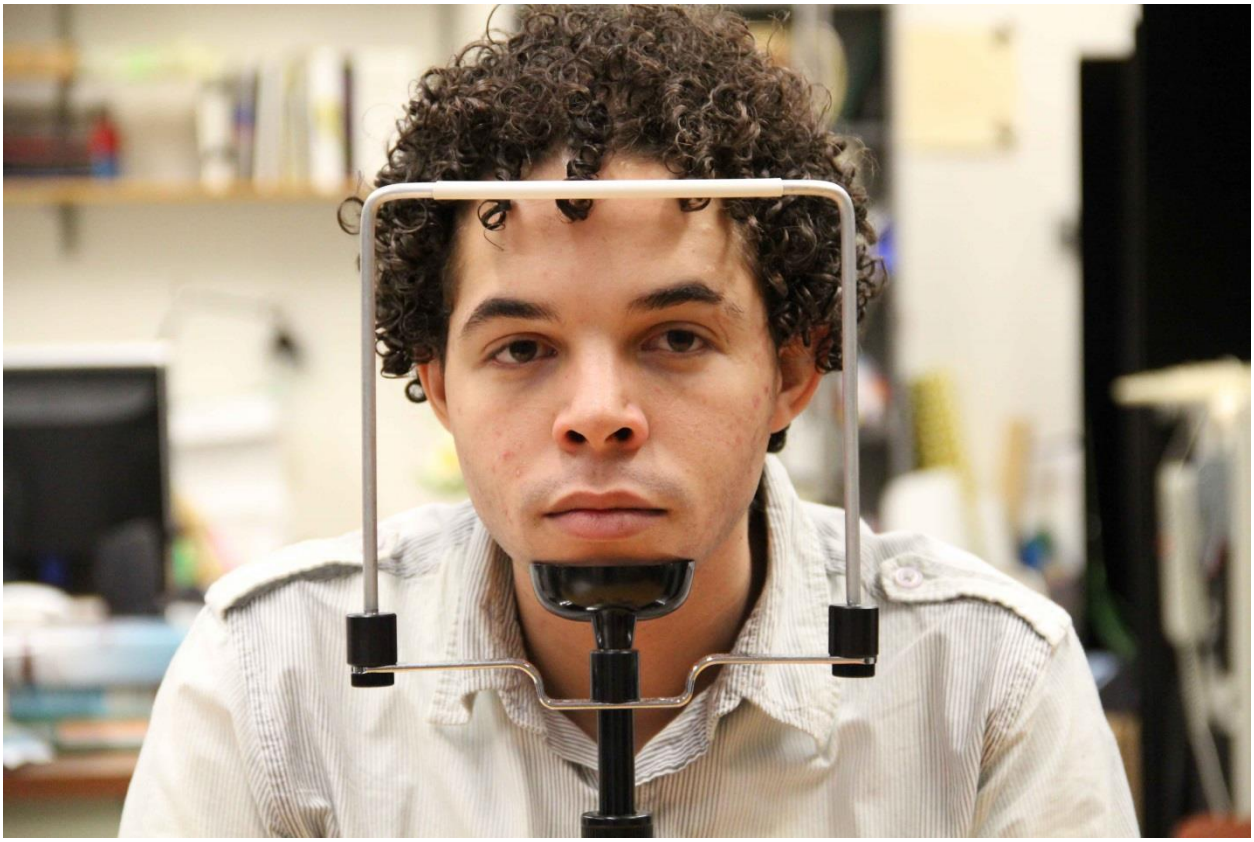
حالت عادی یعنی با مختصات سه بعدی مربوطه. 2) با گذاشتن عددهایی به عنوان لیبل برای تصاویر منتخب و یافتن نزدیک ترین تصویر منتخب برای تصاویر دیگر(خروجی CNN همانند مقاله‌ی مربوطه یک ارایه به اندازه‌ی تعداد عکس‌های منتخب به صورت باینری میباشد), نظیر روش ذکر شده. 3) با کذاشتن مختصات تصاویر منتخب به عنوان لیبل (خروجی CNN سه بعد است) به جای اختصاص دادن یک category برای هر تصویر. با مقایسه نتایج این سه حالت, بهترین حالت, حالت سوم با نزدیک 54 درصد دقت بود.

از مدل بدست آمده در روش بالا در برنامه all.py استفاده کردیم. به گونه‌ای که ابتدا با نگاه کردن به 4 گوشه صفحه و زدن دکمه اسپیس 4 تصویر از 4حالت چشم گرفته شده و به مدل داده میشود. پس از تخمین مختصات توسط مدل و با داشتن مختصات 4 گوشه‌ی صفحه‌ی خود calibrate انجام می‌دهیم. ماتریس بدست آمده برای تبدیل مختصات فریم‌های چشم بعدی پس از تخمین مختصات توسط مدل استفاده می‌شود.

دیتاست دوم:

این دیتاست مربوط به دانشگاه کلمبیا است و شامل 56 شرکت‌کننده است که در آن به 21 نقطه در صفحه نگاه می‌کنند (بعضی از این افراد عینک بر چشم دارند).

لینک دیتاست: http://www.cs.columbia.edu/CAVE/databases/columbia_gaze/



ابتدا ناحیه چشم را از این تصاویر استخراج کردیم، سپس این تصاویر را نرمال کردیم این داده ها به سه صورت رنگی, سیاه سفید و تصویر بعد از زده شدن الگوریتم otsu به شبکه عصبی هم ساختار با روش قبل داده شده‌اند (فایل اجرای این برنامه all2.py است) اما نتایج در هر سه حالت مطلوب نبودند. (درصد در بهترین حالت 20 درصد است)

پس از روش‌های بالا از روشی دیگر برای تشخیص نقطه‌ی نگاه تصویر استفاده کردیم. این روش به این صورت است که با اجرای برنامه مرکز هر حرف را نگاه کنیم (در اینجا هر 21 ناحیه) و از چشممان در آن حالت عکس گرفته و هر عکس گرفته‌شده را به یک حرف که در همان ناحیه قرار دارد, مپ می‌کنیم. در حین اجرای برنامه, پس از هر 8 فریم, برنامه نزدیک ترین تمپلیت و حرف مرتبط با آن را به هر فریم اختصاص می‌دهد و حرفی که در بیتشر فریم‌ها تشخیص داده شود, به عنوان خروجی چاپ می‌شود. برنامه all3.py این روند را اجرا می‌کند.

نکات قابل ذکر:

- Optimizer های استفاده شده عبارتند از: adam ، msprep و SGD

- Loss function ها عبارتند از: categorical_crossentropy برای خروجی‌های اسمی (categorical) و برای خروجی‌های عددی mse (یا همان کمترین مربع خطاها) و mae (یا همان کمترین خطای مطلق)
- برای CNN مربوط به دیتاست Colombia, SGD با learning rate خیلی کم (در حدود0.0001) نتایج بهتری داد.

اصلاحات بعد از ارائه:

- هنگام تایپ کردن درصورتی که کاربر از دوربین دور شود و در مقابل دوربین قرار نگیرد تایپ کردن متوقف می‌شود. (با یک دستور ایف در وایل اصلی برنامه که اندازه فریم چشم دریافتی را چک میکند)
- در پیاده‌سازی سوم یک تابع برای فیچرهای مختلف اضافه شد که برای هر تصویر وکتوری از ویژگی‌ها باز میگرداند که می‌تواند بجای فاصله پیکسل با پیکسل استفاده شود.