دانشگاه صنعتی شریف

دانشکده ریاضی

درس بینایی ماشین

تمرین سوم

دانشجو :

محمدشهاب سپهری ۹۶۱۰۱۷۷۶

استاد درس : دکتر کمالی تبریزی

دقت کنید در کد تحویل داده شده فولدری به نام Config وجود دارد که پارامترهای مربوط به سوالات در آن قرار دارند و این فولدر حتما در هنگام اجرا باید در فولدر اصلی پروژه باشد تا کدها بتوانند پارامترهای مورد نیاز را از آن بخوانند. همچنین نتایج در فولدری به نام Results (که در صورتی که این فولدر موجود نباشد به صورت خودکار ساخته می‌شود) ذخیره می‌شوند. همچنین فولدر دیگر به نام Utils در فولدر اصلی پروژه وجود دارند که در آن فایل‌های حاوی توابع استفاده شده قرار دارند. در ضمن داده‌ها هم در فولدر Data قرار دارند (البته مسیرها را می‌توان با تغییر فایل‌های کانفیگ تغییر داد).

همچنین فایل‌ها و نتایج نیز از طریق لینک کدها و نتایج قابل دسترسی هستند.

۱ نقاط و خطوط محو شدن (Vanishing Point and Lines)

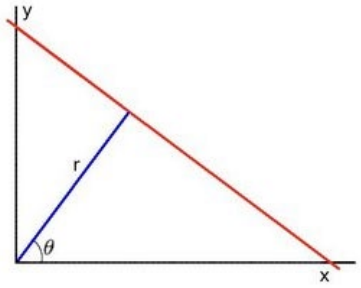
بخش ۱ – به دست آوردن نقاط محو شدن و خط محو شدن افق

خطوط موازی راستاهای مختلف را به صورت اتوماتیک و به کمک الگوریتم Hough بدست آوردم. نحوه کار به این صورت است:

ابتدا نقاط لبه (edge) عکس را به کمک الگوریتم Canny بدست می‌آورم. این الگوریتم عکس را به همراه دو ترشهولد بالا و پایین می‌گیرد. این الگوریتم امتیاز هر پیکسل را برای لبه بودن حساب می‌کند سپس نقاط با امتیاز کمتر از ترشهولد پایین را دور میریزد و نقاط با امتیاز بالاتر از ترشهولد بالا را به عنوان نقاط لبه‌ای قطعی می‌گیرد. برای نقاط با امتیاز بین دو ترشهولد نیز اگر آن‌ها به نقاط لبه اصلی نزدیک بودند آن‌ها را به عنوان لبه می‌گیرد. این الگوریتم در نهایت آن‌ها را در قالب یک عکس می‌دهد.

بعد از بدست آوردن نقاط لبه آن‌ها را به الگوریتم Hough دادم. این تابع عکس مربوط به نقاط لبه را می‌گیرد و براساس نمایش قطبی خطوط (براساس r و θ به مانند شکل ۱) سعی می‌کند خطوط موجود در عکس را بیابد. این الگوریتم در داک خود opencv به طور مفصل توضیح داده شده و توضیح آن خارج از حوصله این گزارش است. البته باید اشاره کنم که این الگوریتم یک نسخه احتمالاتی هم دارد که بهینه‌تر است اما من از آن استفاده نکردم. وروی‌های این تابع عکس لبه‌ها، رزولوشن r، رزولوشن θ، ترشهولد مربوط به تجمیع خطوط (ترشهولد مربوط به تعداد رای‌ها یک خط {در این الگوریتم هر خط تعدادی رای‌دهنده دارد}) و دو ترشهولد مربوط به کمینه و بیشینه

زاویه خطوط هستند. البته این تابع ورودی‌های دیگری هم دارد اما مقدار default دارند و من آن‌ها را تغییر ندادم. حال خروجی این الگوریتم تعدادی r و θ می‌شود که از روی آن‌ها به سادگی می‌توان معادله خط را براساس معادله $-\sin(\theta)\,y - \cos(\theta)\,x - r = 0$ بدست آورد.



شکل ۱

حال پس از بدست آوردن خطوط باید بهترین محل برخورد را بدست آورد. در اینجا من دو ایده داشتم. ایده اول این بود که به کمک svd بهترین تخمین از نقطه محل برخورد این خطوط را بیابم. دقت کنید در حالت ایده‌آل انتظار داریم همه این خطوط در یک نقطه همگرا باشند و لذا اگر A ماتریسی باشند که سطرهای آن ضرایب معادلات خطوط باشند (۳ ضریب برای هر خط)، انتظار داریم که معادله $Ax = 0$ جواب داشته باشد. حال در اینجا به دلیل نویزهای موجود این معادله جواب ندارد و لذا به کمک svd بهترین جواب آن را بدست می‌آوریم. ایده دوم این بود که به کمک RANSAC بهترین خط را بدست آورم. هر دوی این الگوریتم‌ها به دلیل نویزهای موجود نتیجه‌شان خوب نبود به همین دلیل سعی کردم آن‌ها را ترکیب کنم. الگوریتم نهایی به این صورت است که چندین بار (بین ۱۰ تا ۱۰۰هزار تکرار) تعدادی خط (در کد من ۳) را انتخاب می‌کنم و بهترین نقطه برخورد آن‌ها را به کمک svd بدست می‌آورم. سپس برای این نقطه تعداد خطوطی که به آن نزدیک‌اند (براساس یک ترشهولد که در اینجا ۵۰ بود) را حساب می‌کنم و به عنوان ساپورت آن نقطه قرار می‌دهم. درنهایت نقطه با بیشترین ساپورت را به عنوان نقطه برخورد اعلام می‌کنم.

دقت کنید چون زاویه خطوط می‌تواند اندکی خطا داشته باشد و این خطا باعث شود نقطه برخورد آن‌ها بسیار متفاوت شود، RANSAC عادی با مشکل زیادی روبرو می‌شود چون نقطه برخوردهای ۲تایی را بررسی می‌کند.

حال در اینجا من ۳ بار این الگوریتم را اجرا کردم تا ۳ نقطه محو شدن مذکور را بدست آورم. در هر سه آن‌ها رزولوشن مربوط به r و θ را ۱ و ۱ درجه دادم.

برای V_x ترشهولدهای الگوریتم Canny را برابر ۵۰۰ و ۷۰۰ دادم و تنها خطوط با زوایای بین ۹۰ تا ۱۰۵ درجه (با توجه به عکس) را گرفتم و ترشهولد Hough را برابر ۳۵۰ دادم.
برای V_y ترشهولدهای الگوریتم Canny را برابر ۲۰۰ و ۴۰۰ دادم و تنها خطوط با زوایای بین ۸۵ تا ۹۰ درجه (با توجه به عکس) را گرفتم و ترشهولد Hough را برابر ۳۵۰ دادم.
برای V_z ترشهولدهای الگوریتم Canny را برابر ۵۰۰ و ۷۰۰ دادم و تنها خطوط با زوایای بین منفی ۵ تا ۵ درجه (با توجه به عکس) را گرفتم و ترشهولد Hough را برابر ۴۵۰ دادم.
این مقادیر ترشهولدها را براساس عکس و با آزمون و خطا بدست آوردم. خطوط بدست آمده در هر حالت را در نوت بوک مربوط به این سوال قرار دادهام.
تعداد تکرار الگوریتم نهایی نیز برای سه نقطه V_x، V_y و V_z را به ترتیب ۱۰۰، ۲۰ و ۱۰هزار قرار دادم. در نهایت با بدست آوردن نقاط محو شدن آنها را رسم کرده و h را نیز حساب کردم. نتایج را میتوانید در فولدر Results مشاهده کنید.
خروجی کد نیز به صورت زیر شد:

```
Vx:
[9.45983319e+03 2.63725943e+03 1.00000000e+00]
Vy:
[-2.38315041e+04  3.86043297e+03  1.00000000e+00]
Vz:
[-2.20025854e+03 -1.08724706e+05  1.00000000e+00]
h:
[-3.67167220e-02 -9.99325714e-01  2.98281523e+03]
```

بخش ۲ – به دست آوردن فاصله کانونی و نقطه اساسی و زاویه دوربین

از سه نقطه محو شدن بدست آمده در قسمت قبل استفاده میکنیم. چون سه نقطه محو شدن محدود داریم، از معادلات اول سری اول موجود در اسلایدها برای محاسبه نقطه اساسی و فاصله کانونی استفاده کردم. این معادلات به صورت زیر بودند:

$$\mathbf{v}_1 = \begin{bmatrix} a_1 \\ b_1 \\ 1 \end{bmatrix} \quad \mathbf{v}_2 = \begin{bmatrix} a_2 \\ b_2 \\ 1 \end{bmatrix} \quad \mathbf{v}_3 = \begin{bmatrix} a_3 \\ b_3 \\ 1 \end{bmatrix}$$

$$(a_1 - a_3)p_x + (b_1 - b_3)p_y = a_2(a_1 - a_3) + b_2(b_1 - b_3)$$
$$(a_2 - a_3)p_x + (b_2 - b_3)p_y = a_1(a_2 - a_3) + b_1(b_2 - b_3)$$

$$f^2 = -p_x^2 - p_y^2 + (a_1 + a_2)p_x + (b_1 + b_2)p_y - (a_1a_2 + b_1b_2)$$

که vها نقاط محو شدن ما هستند. برای حل این معادلات ابتدا دو معادله اول را حل کردم تا مختصات نقطه اساسی بدست آید سپس به کمک معادله سوم فاصله کانونی را حساب کردم.

ماتریس کالیبراسیون را به کمک رابطه زیر بدست آوردم:

$$\mathbf{K} = \begin{bmatrix} f & 0 & P_x \\ 0 & f & P_y \\ 0 & 0 & 1 \end{bmatrix}$$

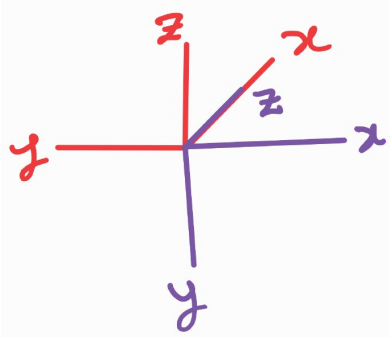
حال به کمک باز افکنش می‌توان ماتریس دوران دوربین یا همان R را بدست آورد. در اینجا معادلات به صورت زیر است (معادلات را از اسلایدها برداشته‌ام):

$$\lambda_i = \frac{1}{\sqrt{\mathbf{v}_i^t \, \mathbf{B} \, \mathbf{v}_i}}$$

$$\mathbf{R} = \mathbf{K}^{-1}[\lambda_1 \mathbf{v}_1 \quad \lambda_2 \mathbf{v}_2 \quad \lambda_3 \mathbf{v}_3]$$

که در اینجا vها همان نقاط محو شدن و B نیز همانطور که می‌دانیم برابر $K^{-t}K^{-1}$ است.

به کمک این معادلات R را حساب کردم. حال دقت کنید در شرایط ایده‌آل ما انتظار داشتیم که محورهای دوربین و فضا به صورت زیر باشند:



که در اینجا محورهای قرمز مربوط به فضای کلی و محورهای بنفش مربوط به دوربین‌اند (این جهت‌ها با توجه به جهت‌های قراردادی opencv درنظر گرفته شده‌اند).

لذا در حالتی که دوربین ما کاملا تراز باشد می‌توان گفت انتظار داریم ماتریس R آن به صورت زیر باشد:

$$R_{opt} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}$$

حال برای کاملا تراز کردن ماتریس باید آن را به ماتریس $R_1 = R_{opt}R^{-1}$ تبدیل کنیم. در اینجا من به کمک روش اویلر، تجزیه ماتریس دوران R_1 را در جهت‌های مختلف بدست آوردم. ترتیب جهت‌ها را هم به گونه‌ای قرار دادم که اولین دوران حول محور Z، دومین دوران حول محور X و سومین دوران حول محور Y باشد. این کار را به کمک کلاس Rotation از کتاب‌خانه scipy.spatial.transform انجام دادم. حال خروجی این کار ۳ زاویه است که به ترتیب سه دوران گفته شده را توصیف می‌کنند و اگر ابتدا دوران حول محور Z، سپس دوران حول محور X و درنهایت دوران حول محور Y را انجام دهیم، دوران حاصل معادل R_1 باشد (البته به دلیل خطاها و نویزهای موجود R_1 دقیقا یک ماتریس دوران نیست و ماتریس حاصل اندکی با R_1 متفاوت است). در نهایت خروجی این بخش به صورت زیر شد:

```
f: 13846.33
P:
[1837.53915895 1172.77028494]
Angle of rotation around Z axis: 2.10 degrees
Angle of rotation around X axis: 7.18 degrees
```

در نوت بوک مربوط به این بخش من نتیجه عکس حاصل از دو دوران اول را قرار داده‌ام. دقت کنید که برای اعمال دوران باید یک هموگرافی اعمال کنیم. این هموگرافی به صورت $H = KR_{des}K^{-1}$ است چون می‌دانیم که ماتریس هموگرافی عکس به صورت $H = KR$ است و لذا برای اعمال دوران مورد نظر باید هموگرافی ما به صورت $KR_{des}K^{-1}$ باشد (R_{des} دورانی است که می‌خواهیم اعمال کنیم). همانطور که از عکس موجود در نوت بوک پیداست، عکس حاصل خواص مورد نظر را دارد.

بخش ۳ – صاف کردن تصویر

همانطور که در بالا توضیح داده شد برای اعمال دوران مدنظر باید هموگرافی را به صورت $KR_{des}K^{-1}$ قرار داد که در اینجا R_{des} حاصل ضرب دو ماتریس دوران یکی حول محور Z با زاویه ۲.۱ درجه و دیگری حول محور X با زاویه ۷.۱۸ درجه است (ماتریس اول از سمت راست ضرب شده است). ماتریس هموگرافی بدست آمده در این قسمت به صورت زیر شد (البته دقت کنید در کد هموگرافی نهایی یک آفست هم دارد که برای این است که عکس در کادر بیفتد و مشابه اینکار را در تمرین‌های گذشته نیز انجام داده بودیم):

```
desired homography matrix:
[[ 9.99936201e-01 -2.00679051e-02  9.24301144e+00]
 [ 3.67442830e-02  1.00207120e+00 -1.80975495e+03]
 [ 3.30772339e-07  9.02065327e-06  9.80971429e-01]]
```

همچنین عکس این قسمت در فولدر Results قابل مشاهده است.

۲ هندسه اپیپولار

برای محاسبه نقاط متناظر از کدهای تمرین‌های قبلی خودم استفاده کردم و نقاط متناظر را به کمک SIFT حساب کردم. برای بدست آوردم ماتریس فاندامنتال از کتاب‌خانه opencv و تابع findFundamentalMat استفاده کردم. این تابع نقاط متناظر را گرفته و ماتریس فاندامنتال را حساب می‌کند. همچنین برای اینکه این تابع از RANSAC استفاده کند باید flag مناسب را به آن ورودی داد.

برای بدست آوردن نقاط اپیپول به کمک svd دو معادله $Fe = 0$ و $F^t e' = 0$ را حل کردم. البته لازم به ذکر است که نقاط اپیپول خارج از تصویرها قرار داشتند.

برای بدست آوردن خطوط متناظر از حقیقت استفاده کردم که برای یک نقطه، نقطه متناظرش در عکس دیگر و همچنین نقطه اپیپول عکس دیگر هر دو روی خط اپیپول متناظر با نقطه اول قرار دارند و لذا برای بدست آوردن این خط نقطه اپیپول عکس دیگر را در نقطه متناظر نقطه اولیه ضرب خارجی کردم (هندسه پراجکتیو) و خطوط را بدست آوردم.

خروجی‌های این قسمت به صورت زیر شدند:

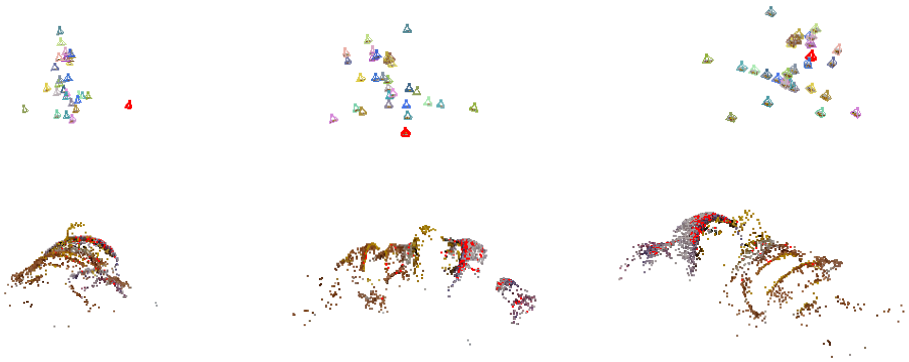
```
Fundamental matrix:
[[-3.29021331e-09 -6.59685480e-08 -1.73541423e-04]
 [-9.38155929e-08 -1.94334972e-09 -1.45794274e-03]
 [-1.50560577e-04  1.79488589e-03  1.00000000e+00]]


e:
[[-15502.03893994]
 [ -1857.49742598]]


e_prime:
[[27283.67189917]
 [-2561.72422798]]
```

۳ سه بعدی سازی

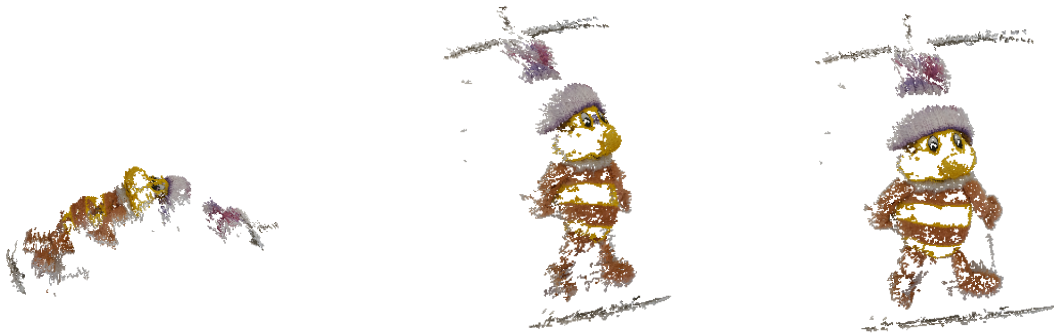
برای این سوال از Visual SFM استفاده کردم. جسم مورد نظر یک عروسک روی زمین است که ۳۰ عکس از آن گرفته‌ام. نتایج را در فولدر Results قرار داده‌ام و همچنین عکس‌ها در فولدر Data قرار دارند. در زیر تعدادی عکس از خروجی‌ها آورده‌ام:

یکی از عکس‌های اصلی:

ابر نقاط:



بافت:



۴ تشخیص صحنه با استفاده از سبد کلمات

در تمام بخش‌ها این سوال من قسمتی (۱۰ درصد) از داده‌های آموزش (برای قسمت اول بردارهای ۲۵۶تایی و برای دو قسمت بعدی هیستوگرام‌ها) را به عنوان داده‌های validation جدا کردم و از آن‌ها برای تعیین هایپرپارامترها کمک گرفتم.

همچنین در تمام قسمت‌ها از کتاب‌خانه sklearn استفاده کرده‌ام.

بخش ۱ – نمایش ساده و نزدیک‌ترین همسایه

در اینجا همه عکس‌ها را به کمک دستور resize از کتاب‌خانه opencv به اندازه بهینه می‌برم و سپس عکس خروجی را به یک بردار تبدیل می‌کنم.

برای ساختن مدل یک مدل nearestNeighbors از کتاب‌خانه sklean ساختم و سپس به کمک دستور fit داده‌های ترین را به آن آموزش دادم و به کمک دستور score دقت آن را روی داده‌های ولیدیشن و تست محاسبه کردم.

در اینجا من اندازه‌های مختلف از ۱۰ تا ۵۴ (با گام‌های ۶ تایی) و kهای بین ۱ تا ۱۰ را به ازای هر دو نرم گفته شده بررسی کردم که بهترین نتیجه مربوط به نرم L_1 و k برابر ۱ و سایز ۱۶ شد! این بررسی در نوت بوک مربوط به این سوال قرار دارد. دقت نهایی نیز ۲۳.۰۷ درصد بود.

بخش ۲ – سبد کلمات و نزدیک‌ترین همسایه

برای این بخش و بخش بعدی لازم است که یک دیکشنری از ویژگی‌های بصری داده‌ها درست کنیم. برای این کار من ابتدا به کمک همان دستور استفاده شده در سوال ۱ روی تمام عکس‌های آموزش SIFT را اجرا کردم و برای هر عکس بردارهای توضیح[1] دهنده‌اش را نرمالایز کردم. سپس همه این بردارهای توضیح دهنده را را در یک لیست قرار دادم و روی آن‌ها k means زدم. برای k means با توجه به جستجوی‌هایی که داشتم عددی که معمولا برای k خوب است، ۱۰ برابر تعداد کلاس‌ها است (به دلیل کند بودن ساخت داده‌های این قسمت و زمان اجرای بالای آن، نمی‌توانستم kهای زیادی را تست کنم) و لذا k را ۱۵۰ قرار دادم. همچنین به دلیل تعداد زیاد بردارهای توضیح دهنده (بیش از ۱.۵ میلیون) مجبور به استفاده از MiniBatchKmeans شدم. این کلاس در کتاب‌خانه

[1] descriptor

sklearn قرار دارد و تفاوت آن با kmeans عادی این اسن که هر بار تنها به اندازه batchsizeاش داده از داده‌های آموزش انتخاب می‌کند و روی آن‌ها الگوریتم را اجرا می‌کند. این الگوریتم سرعت بسیار بیشتری داشت. برای batchsize من از عدد ۱۵۰ (باز هم ده برابر تعداد کلاس‌ها و البته این بار براساس محدودیت سیستم خودم) استفاده کردم. بعد از ساخته شدن kmeans می‌توان از آن برای ساخت هیستوگرام‌ها استفاده کرد.

برای ساخت هیستوگرام یک عکس ابتدا بردارهای توضیح دهنده آن را با SIFT حساب می‌کنم و بعد از نرمالایز کردنشان آن‌ها را به مدل kmeansام می‌دهم و بر این اساس می‌توانم کلاس هر یک از بردارهای توضیح دهنده را بیابم و سپس براساس آن هیستوگرام را بدست آورم. البته این هیستوگرام باید نرمالیزه باشد.

به کمک روش بالا هیستوگرام مربوط به داده‌های آموزش و تست را ساختم و سپس داده‌های آموزش را به بخش ولیدیشن و آموزش تقسیم کردم.

حال از اینجا به بعد مشابه قسمت قبل یک مدل knn ساختم و با دستور fit آن را آموزش دادم و با دستور score دقت آن را حساب کردم. سپس به ازای kهای مختلف دقت آن را روی داده‌های ولیدیشن بررسی کردم و درنهایت بهترین مدل را انتخاب کردم.

بهترین k برابر ۲۷ و دقت مدل نهایی روی داده‌های تست برابر ۳۸.۷ درصد شد. البته دقت این مدل روی داده‌های ولیدیشن ۴۲.۸ بوده است.

```
best accuracy: 42.81%
best k: 27
test accuracy: 38.67%
```

بخش ۳ – سبد کلمات و SVM

ساخت دیکشنری این قسمت دقیقا مشابه قسمت قبل است. برای مدل svm از مدل SVC کتاب-خانه sklearn استفاده کردم. کرنل آن را چند جمله‌ای گرفتم و درجه آن را با بررسی kهای مختلف بدست آوردم. این مدل نیز مشابه مدل‌های knn دستورهای fit و score دارد. در نهایت نتیجه آن به صورت زیر شد:

```
best validation accuracy: 48.49%
best degree: 2
test accuracy: 57.19%
```