دانشگاه صنعتی شریف

دانشکده ریاضی

درس بینایی ماشین

تمرین دوم

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 پانوراما و پردازش ویدیو

بخش ۱

در ابتدا به کمک کلاس VideoCapture از کتاب‌خانه opencv فریم‌های ویدیو را خواندم و آن‌ها را ذخیره کردم. همچنین چون رزولوشن فریم‌ها بالا بود و باعث کندی زیاد کد می‌شد، رزولوشن را یک چهارم کردم. برای محاسبه هموگرافی میان دو عکس از کد سوال ۳ تمرین قبل خودم استفاده کردم که به کمک SIFT نقاط مهم و بردارهای ویژگی‌شان را بدست می‌آوردم و سپس به کمک RANSAC ماتریس هموگرافی بهینه را پیدا می‌کردم. البته در اینجا یک تغییری در کد دادم که بتواند مقادیر بیشینه و کمینه مختصات‌های عکس بعد از تبدیل را بدهد. این ۴ عدد (۲ تا برای کمینه و بیشینه x و ۲ تا برای کمینه و بیشینه y) را با ضرب کردن ۴ راس عکس در هموگرافی و بررسی مختصات‌های آن‌ها بدست آوردم. به کمک این ۴ عدد می‌توان کوچک‌ترین اندازه عکس لازم برای اینکه کل تبدیل عکس در آن جا شود و همچنین مقدار offset (میزانی که در تمرین قبل عکس را شیفت می‌دادیم تا خروجی warpPerspective بریده نباشد) را بدست آورد.

برای ساختن مربع در عکس ۴۵۰ و محاسبه معادل آن در عکس ۲۷۰ به این صورت کردم مربعی در وسط عکس ۴۵۰ درنظر گرفتم (طول ضلع آن ۰.۵ ارتفاع عکس است و هم اندازه و هم موقعیت نسبی آن در کانفیگ‌ها قابل تنظیم است). سپس نقاط متناظر ۴ راس آن را به فضای Projective بردم (با اضافه کردن بعد سوم ۱) و سپس این نقاط را در وارون ماتریس هموگرافی محاسبه شده ضرب کردم تا نقاط مربوط به ۴ راس در عکس ۲۷۰ را در فضای Projective بدست آوردم و

سپس از روی آن مختصات‌های ۴ راس را در عکس ۲۷۰ بدست آوردم. حال چون در تبدیل هموگرافی خط را به خط می‌برد می‌توانیم با وصل کردن این ۴ خط به هم (با همان ترتیبی که در قبل به هم وصل شده بودند) تصویر مربع را بدست آوریم.
برای مرحله بعد به کمک ۴ عدد خروجی مربوط به مقادیر بیشینه و کمینه مختصات‌های عکس بعد از تبدیل سایز عکس ترکیب شده را بدست آوردم (می‌دانیم تصویر عکس ۲۷۰ در صفحه عکس ۴۵۰ باید مقداری شیفت بخورد و لذا عکس ۴۵۰ نیز باید به همان اندازه شیفت بخورد تا موقعیت نسبی آن‌ها نسب به هم تغییر نکند سپس بر این اساس می‌توان بیشینه و کمینه x و y مختصات عکس‌ها را بدست آورد و لذا سایز مورد نیاز را تعیین کرد). سپس به کمک warpPerspective عکس ۲۷۰ را به عکسی با این اندازه بدست آمده تبدیل کردم (دقت کنید که در اینجا آفست را هم باید لحاظ کنیم). سپس مختصات‌های این عکس بزرگ که مربوط به عکس ۴۵۰ بودند را به کمک تساوی ماتریسی برابر خود عکس ۴۵۰ قرار دادم (باز هم باید offsetها را اعمال کنیم). همانطور که در نتایج نیز آمده است خروجی به وضوح یک بریدگی در وسط خود دارد.

بخش ۲

در اینجا محاسبه هموگرافی‌ها به مانند قبل است (البته باید حواسمان به این باشد که برای بدست آوردن غیرمستقیم هموگرافی ۹۰ به ۴۵۰ باید ماتریس هموگرافی ۲۷۰ به ۴۵۰ را از سمت چپ در ماتریس هموگرافی ۹۰ به ۲۷۰ ضرب کنیم و مشابه همین عمل را برای عکس ۸۱۰ انجام دهیم). برای بدست آوردن اندازه تصویر از همان مقادیر کمینه و بیشینه مختصات‌ها که در قسمت قبل نیز توضیح دادم استفاده کردم.
اما بخش اصلی این سوال در نحوه چسباندن عکس‌ها به هم بود. برای اینکار از برنامه نویسی پویا استفاده کردم. نحوه کار به این صورت است که تابعی به نام merger تعریف کردم که در ورودی دو عکس ۱ و ۲ با اندازه برابر می‌گیرد و عکس ۱ را از چپ به عکس ۲ می‌چسباند. روش به این صورت است که ابتدا دو عکس متناظر با مقادیر این ۲ عکس در نقاط اشتراکشان ایجاد می‌کنیم. سپس این عکس‌ها را تا جای ممکن کراپ می‌کنیم (به نحوی که بالاترین و پایین‌ترین سطر تماما مشکی نباشند و به همین ترتیب برای ستون اول و آخر). حال اختلاف این دو عکس را حساب می‌کنیم و در ماتریس تفاضل (نرم ۲ البته توجه کنید که خانه‌های آن را برای جلوگیری از اورفلو

تقسیم بر ۱۰۰۰۰ کرده‌ام) می‌ریزیم. در مرحله بعد یک ماتریس با ابعاد این عکس‌ها می‌سازیم که تمام خانه‌های آن بینهایت‌اند. این ماتریس قرار است ماتریس هزینه رسیدن به هر خانه باشد که توسط برنامه نویسی پویا آن را پر می‌کنیم. در ابتدا باید تعدادی خانه این ماتریس را مقدار دهیم. در حالت عادی کافی بود که سطر اول این ماتریس را مقدار دهیم اما در اینجا باید دقت کرد که عکس مربوط به اشتراک به صورت یک متساوی‌الاضلاع است و مستطیلی نیست و لذا نمی‌توان از آن روش استفاده کرد. در عوض به این صورت عمل می‌کنیم که در تمام ستون‌ها اولین خانه ناصفر را می‌گیریم و اگر این خانه از حدی پایین‌تر نبود (این حد را ۰.۳ طول ضلع گذاشتم و وظیفه آن این است که جلوی مقدار دادن به اضلاع افقی متوازی الاضلاع را بگیرد دقت کنید که هدف ما این است که ضلع بالایی متوازی الاضلاع را مقدار دهیم و چون یافتن آن کند و سخت است از این تقریب برای آن استفاده کردم) مقدار آن را برابر مقدار خانه متناظرش در ماتریس تفاضل قرار می‌دهیم.

بعد از این مرحله از سطر دوم شروع می‌کنیم و مقدار هر خانه را برابر کمینه مقدار بین ۳ خانه بالایی‌اش به علاوه مقدار خانه متناظرش در ماتریس تفاضل قرار می‌دهیم (این روش به صورت مفصل در کلاس‌های ترم پیش پردازش تصویر توضیح داده شده بود و من نیز با مطالعه مطالب آن این روش را پیاده کردم. در آنجا به دقت توضیح داده می‌شود که چرا به اینگونه عمل می‌کنیم و لذا از تکرار آن پرهیز می‌کنم.). البته دقت کنید در این حالت ما تنها حرکت میان سطرهای عکس را مجاز گذاشته‌ایم و این در حالی است که در اصل می‌توانستیم در ستون‌ها نیز حرکت کنیم اما محاسبه آن حالت بسیار زمان‌بر و سنگین است و لذا از این روش به عنوان تقریبی برای آن استفاده کرده‌ام.

بعد از پر شدن ماتریس هزینه‌ها نوبت به یافتن انتهای مسیر بهینه است. این نقطه، نقطه با کمترین هزینه روی ضلع پایین متوازی الاضلاع است که باید آن را پیدا کنیم. برای یافتم این نقطه نیز از همان روش قبل استفاده کردم. در یکی از دو عکس اشتراک (مثلا عکس مربوط به مقادیر عکس ۱) به ازای هر سطر پایین‌ترین خانه را یافتم و اگر از حدی بالاتر نبود (با همان حد ۰.۳ طول ضلع) آن را جزو نقاط ضلع پایین در نظر می‌گیرم. سپس بین تمام این نقاط نقطه با کمترین هزینه را به عنوان نقطه پایان مسیر انتخاب می‌کنم.

بعد از یافتن نقطه پایان مسیر نوبت به یافتن خود مسیر می‌رسد. برای اینکار نیز به این صورت عمل می‌کنیم که هر بار برای یافتن نقطه قبلی محاسبه می‌کنیم که کدام یک از سه خانه بالایی کمینه است و آن خانه، خانه قبلی مسیر است. البته باید بررسی کنیم که اگر مقدار خانه فعلی برابر مقدار متناظرش در ماتریس تفاضل بود، این خانه از همان خانه‌های ضلع بالایی است و لذا نقطه شروع است.

حال از روی این مسیر یک ماسک برای دو تصویر می‌سازیم به نحوی که خانه‌های چپ مسیر را از عکس ۱ و خانه‌های راست آن را از عکس ۲ برمی‌داریم (البته در عمل پیاده سازی ماسک اندکی پیچیده‌تر است، این مسیر در بخشی از عکس بزرگ اصلی قرار دارد و آن عکس را به دو قسمت مجزا تقسیم نمی‌کند. برای رفع این مشکل من به این صورت عمل کردم که از نقطه ابتدای آن به صورت عمودی رو به بالا می‌روم و از نقطه انتهای آن هم به صورت عمودی رو به پایین می‌روم و یک مسیر در عکس بزرگ اصلی می‌سازم). نتایج این روش در قابل قبولی خوب بودند. البته مسیر چسباندن عکس‌ها تا حدودی پیدا بود.

در نهایت این کار را ۴ بار تکرار کردم (برای عکس ۹۰ و ۲۷۰ و سپس برای نتیجه آن‌ها و عکس ۴۵۰ و به همین ترتیب) تا پانوراما‌ی کلی بدست آید.

بخش ۳

در اینجا ابتدا لازم است که سایز ویدیو را تعیین کنیم. برای این کار ۲۰ فریم اول، آخر و وسط (مجموعا ۶۰ فریم) را انتخاب کردم و براساس مقادیر بیشینه و کمینه مختصات آن‌ها سایز عکس را بدست آوردم. در اصل باید تنها به کمک فریم اول، آخر و وسط می‌توانستیم سایز ویدیو را تعیین کنیم اما به دلیل لرزش‌های موجود در ویدیو در مجموع ۶۰ فریم را بررسی کردم.

بقیه کار محاسبه هموگرافی است. برای محاسبه هموگرافی، اولا دقت کنید که هموگرافی ۵ فریم کلیدی را داریم. حال برای فریم‌های بین ۱ تا ۹۰ ابتدا هموگرافی بین آن‌ها و فریم ۹۰ را می‌یابم سپس از راست در هموگرافی فریم ۹۰ به ۴۵۰ ضرب می‌کنم. برای فریم‌های بین ۹۰ تا ۲۷۰ نیز هموگرافی آن‌ها را به فریم ۲۷۰ حساب می‌کنم. برای فریم‌ها بین ۲۷۰ تا ۳۶۰ هموگرافی را مستقیم حساب می‌کنم و برای بقیه نیز به همین ترتیب از فریم ۶۳۰ و ۸۱۰ استفاده می‌کنم.

در اینجا هموگرافی فریم ۴۵۰ را I می‌دادم که تابع warpPerspective را دچار مشکل می‌کرد لذا خانه اول آن را به جای ۱ برابر ۱.۰۰۰۰۰۱ قرار دادم که این تابع به مشکل نخورد.

بخش ۴

برای بدست آوردن پس‌زمینه ایده من این بود که از فریم‌های ویدیوی قبلی استفاده کنم و عکس پس‌زمینه را بسازم. حال در این جا دو روش هست:

- برای هر پیکسل بین مقادیر ناصفر (هر سه مقدار rgb) میانگین بگیریم.
- برای هر پیکسل بین مقادیر ناصفر (هر سه مقدار rgb) مد بگیریم.

من هر دو روش را پیاده کردم البته روش دوم به دلیل آنکه باید در آن واحد مقادیر ۹۰۰ فریم را برای همه‌ی پیکسل‌ها داشته باشد به رم زیادی نیاز دارد. لذا عکس را به برش‌های ۳۰۰ ستونی تقسیم کردم و برای هر برش جداگانه محاسبات را انجام دادم. در این حالت حدود ۱۳ گیگ از رم مصرف می‌شود و لذا در موقع اجرا حتما مطمعن شوید که این مقدار رم دارید! دقت کنید در محاسبه مد حتما باید خانه‌های تمام خانه‌های تمام صفر هر پیکسل را حذف کنیم (چون تعداد آن‌ها زیاد است و مد خانه تمام صفر می‌شود) که این کار کد را کند می‌کرد. لذا برای بهتر کردن سرعت کد در هر مرحله به جای صفرها یک شماره جدید اضافه می‌کنم. یعنی در ابتدا یک ماتریس به ابعاد عکس می‌گیرم که تمام خانه‌های آن ۲۵۶ هستند (می‌دانیم که رنگ‌های rgb همگی کمتر از ۲۵۶اند) سپس در هر مرحله پیکسل‌هایی از عکس که تمام صفر اند را با خانه‌های متناظرشان در این ماتریس عوض می‌کنم و سپس خانه‌های متناظر آن‌ها در این ماتریس را به علاوه ۱ می‌کنم. در این صورت به جای صفرهای خانه‌های متمایز داریم و می‌توان کل عکس‌ها را در یک ماتریس نگه داشت و با سرعت بیشتر (بدون لوپ for) مد را برای همه پیکسل‌ها یکجا (به کمک دستور mode از کتاب‌خانه scipy.stats) حساب کرد.

لازم به ذکر است که نتایج این دو روش خیلی خوب نمی‌شدند. لذا از دو ایده استفاده کردم.

- برای هر پیکسل به جای حذف خانه‌های تمام صفر، خانه‌هایی که جمع سه رنگ rgb آن‌ها کمتر از ۳۷ است را حذف می‌کنم.
- بعد از مشاهدات بنظر می‌آمد نتیجه روش مد در سمت راست عکس خوب و در سمت چپ آن بد است. برای نتیجه روش میانگین دقیقا عکس این اتفاق افتاده بودو لذا من سه پنجم

سمت چپ نتیجه روش میانگین را گرفتم (بقیه خانه‌ها را صفر کردم) و آن را با سه پنجم سمت راست نتیجه روش مد یکی کردم. روش یکی کردن نیز استفاده از همان تابع merger بود که در بخش پانوراما از آن استفاده کرده بودیم. نتیجه خروجی از هر دو نتیجه بهتر بود و لذا آن را به عنوان نتیجه اصلی ذخیره کردم اما نتایج مربوط به روش‌های مد و میانگین را نیز در فایل نتایج قرار دادم (با اجرای کد نیز تولید می‌شوند).

بخش ۵

در اینجا از همان هموگرافی‌های استفاده شده برای ساخت ویدیو بخش ۳ استفاده کردم (البته ماتریس offset را هم باید درنظر گرفت). به ازای هر هموگرافی وارون آن را روی عکس ساخته شده در بخش قبل اعمال کردم و نتیجه را روی صفحه‌ای به اندازه ویدیو اصلی انداختم و در نهایت از روی تمام این عکس‌ها فیلم مربوطه را ساختم.

بخش ۶

در اینجا ابتدا فریم‌های ویدیو قسمت قبل را در یک فولدر ریختم. حال ۹۰۰ فریم برای ویدیو پیش‌زمینه و ۹۰۰ فریم متناظر برای ویدیوی اصلی داریم. روش که من پیاده کردم این است که به ازای هر دو فریم متناظر، اختلاف این دو را حساب می‌کنم (نتیجه با نرم ۲ است اما نرم ۱ هم پیاده شده) و سپس یک فیلتر moving average را روی این ماتریس اختلاف اعمال می‌کنم تا نقاط تنها در آن از بین بروند و نویز کمتر شود. سپس یک ترشهولد (در اینجا ۰.۳) را روی نرمالیزه شده ماتریس حاصل اعمال می‌کنم. درنهایت نقاطی که ۱ هستند را به عنوان پس‌زمینه درنظر می‌گیرم. البته نتایج این بخش خوب نبود و برای مثال ساختمان‌ها اغلب به عنوان پس زمینه حساب می‌شدند. بخشی از علت این مشکل به دلیل لرزش دوربین است که باعث شده تصویر نهایی ساختمان‌ها تا حدی تار باشد و چون رنگ ساختمان‌ها (به خصوص ساختمان راست) حالت راه راه دارد (تعدادی تیرگی در ساختمان کرم رنگ هست) به همین دلیل ماتریس اختلاف روی آن مقدار زیادی میگیرد و همین امر باعث خطا می‌شود. البته اعمال فیلتر تا حدی این مشکل را بهتر کرد اما نتوانست آن را کامل رفع کند.

بخش ۷

در اینجا از ایده ساده‌ای استفاده کردم. در بخش ۵ از تصویر پانوراما‌ی پس‌زمینه به کمک وارون ماتریس‌های هوموگرافی فریم‌هایی با ابعاد ویدیو اصلی می‌ساختیم و با آن‌ها یک ویدیو تولید می‌کردیم. در اینجا به جای آن که سایز هر فریم را برابر با ابعاد ویدیو اصلی بگیریم، عرض آن را ۱.۵ برابر می‌کنیم. فقط باید توجه کرد که در انتهای ویدیو سمت راست فریم‌ها سیاه می‌شود (این امر منطقی است چون برای فریم‌های انتهایی دیتایی از سمت راست آن‌ها نداریم). برای رفع این مشکل فریم‌ها را تاجایی می‌سازم که ستون آخر فریم تولید شده تمام صفر نباشد. به همین دلیل در هنگام اجرای کد، درصد اجرای این بخش در حدود ۷۰ درصد متوقف می‌شود.

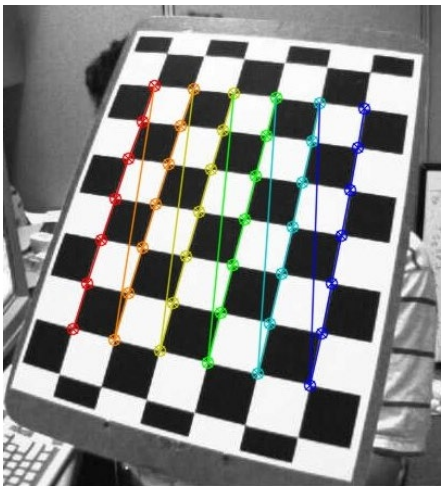
بخش ۸

برای این بخش ایده پیاده‌سازی شده به صورت زیر است:

برای هر فریم مرکز آن را می‌گیریم و تصویر آن را در صفحه فریم ۴۵۰ درنظر می‌گیریم. انتظار ما این است که این نقاط در سطرهای یکسانی باشند و همچنین مقدارسطر آن‌ها به صورت پیوسته تغییر کند. برای یکی کردن سطرها به ازای هر فریم مقدار شیفت متناظر تا نقطه مرکزی آن را محاسبه می‌کنم و به کمک آن یک ماتریس انتقال می‌سازم و مختصات آن فریم را بعد از بردن به صفحه فریم ۴۵۰ از راست در این ماتریس ضرب می‌کنم. برای محور x نیز ابتدا کمینه و بیشینه x را در حالتی که مختصات x مرکز فریم ۴۵۰ را ۰ گرفته‌ام حساب می‌کنم. سپس فرض می‌کنم که از فریم اول تا آخر ۹۰ درجه دوران داشته باشیم. حال اگر dx اختلاف بین کمینه و بیشینه مقدار x باشد، مقدار x در هر خانه از رابطه $x_{min} + dx \times \sin\left(\frac{num-1}{899}\right)$ بدست می‌آید که num شماره فریم است. حال بر این اساس تصویر هر فریم را به اندازه مورد نظر در صفحه فریم ۴۵۰ شیفت می‌دهم البته در این شیفت دادن‌ها باید مواظب بود که تصویر از صفحه خارج نشود برای این کار سایز صفحه را به صورت دستی و با آزمون و خطا تعیین کردم. مرحله بعدی برگرداندن تصویر فعلی به صفحه فریم مربوطه است. برای این کار بین تصویر فعلی (تصویر شیفت یافته شده در صفحه فریم ۴۵۰) و فریم اصلی یک هموگرافی پیدا می‌کنم و به کمک آن تصویر را برمی‌گردانم. برای برگرداندن نیز از warpPerspective با سایز خروجی به اندازه فریم‌های تصویر اصلی استفاده می‌کنم.

۲ کالیبراسیون دوربین

در این سوال از توابع آماده کتاب‌خانه opencv استفاده کردم. به کمک دستور findChessboardCorners می‌توان با دادن ابعاد صفحه شطرنجی و همچنین عکس مربوطه مختصات گوشه‌های صفحه شطرنجی را یافت (مانند عکس زیر که از داک opencv آورده‌ام):



حال برای بدست آوردن ماتریس کالیبراسیون به این صورت عمل می‌کنیم. فرض می‌کنیم که صفحه شطرنجی در فضای ۳ بعدی ثابت است و حرکت‌ها مربوط به دوربین است. لذا می‌توان فرض کرد که مختصات‌های ۳ بعدی مربوط به نقاط گوشه صفحه شطرنجی در فضا ثابت‌اند. حال برای سادگی فرض می‌کنیم که صفحه شطرنجی روی صفحه $Z=0$ قرار دارد و مختصات گوشه بالا راست آن ۰ و ۰ است و همانطور که می‌دانیم فاصله بین هر دو گوشه کنار هم ۲۲ میلی‌متر است. به این صورت یک لیست از مختصات نقاط گوشه صفحه شطرنجی می‌سازیم که این لیست ثابت است (زیرا فرض کرده‌ایم صفحه شطرنجی‌مان ثابت است.

حال خروجی‌های findChessboardCorners مختصات‌های نقاط متناظر این نقاط ۳ بعدی در صفحه دوربین هستند. همچنین می‌توان به کمک دستور cornerSubPix دقت این نقاط را بیشتر کرد. البته این تابع را من روی عکس‌های داده شده تست کردم و نتیجه تغییری نکرد و لذا از آن استفاده نکردم.

در نهایت ما به ازای هر عکس می‌توانیم برداری از آبجکت‌های سه بعدی و برداری از مختصات‌های ۲ بعدی متناظر آن‌ها پیدا کنیم. حال تمام این بردارها را به ازای عکس‌هایی که می‌خواهیم در دو لیست می‌ریزیم (یک لیست برای مختصات‌های ۳ بعدی و یک لیست برای مختصات‌های ۲ بعدی).

در اینجا تابع calibrateCamera با گرفتن این دو لیست و همچنین سایز عکس‌ها می‌توان ماتریس کالیبریشن را به ما بدهد. همچنین اگر flag مربوط به ثابت بودن نقطه principal point را به آن بدهیم، این نقطه را در وسط تصاویر می‌گیرد و محاسبات را انجام می‌دهد.

در بین خروجی‌های calibrateCamera، بردارهای rotation و translation و همچنین distortion قرار دارند. به کمک distortion و تابع getOptimalNewCameraMatrix می‌توان دیستورشن ماتریس کالیبریشن را کم کرد.

همچنین برای خطا می‌توان از معیار Reprojection error استفاده کرد که در این معیار نقاط سه بعدی را به کمک ماتریس کالیبراسیون به نقاط ۲ بعدی می‌بریم و اختلاف نقاط بدست آمده با نقاط واقعی را حساب کرده و میانگین می‌گیریم. نتایج مربوط به ۴ حالت گفته شده به صورت زیر شدند:

```
Results for not fixed PP:


#################   Task: 1   #################
K:
[[2.92811735e+03 0.00000000e+00 9.04399027e+02]
 [0.00000000e+00 2.94909182e+03 5.52474689e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


error:0.048033016335553756


K without distortion:
[[9.91774719e+02 0.00000000e+00 1.29173137e+03]
 [0.00000000e+00 1.22104675e+03 4.49395351e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


#################   Task: 2   #################
K:
[[2.98954136e+03 0.00000000e+00 9.81052207e+02]
 [0.00000000e+00 2.98467270e+03 5.51504968e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


error:0.052088595986264995


K without distortion:
[[3.04293823e+03 0.00000000e+00 9.67869251e+02]
 [0.00000000e+00 2.95801025e+03 5.52178622e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

```
#################   Task: 3   #################
K:
[[3.04925817e+03 0.00000000e+00 7.07938030e+02]
 [0.00000000e+00 3.03791333e+03 5.52658800e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


error:0.05830997002355178


K without distortion:
[[3.02683521e+03 0.00000000e+00 7.01593894e+02]
 [0.00000000e+00 3.00579126e+03 5.53654713e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


#################   Task: 4   #################
K:
[[2.98435855e+03 0.00000000e+00 8.26871368e+02]
 [0.00000000e+00 2.99231411e+03 5.18171284e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


error:0.056653778937226416


K without distortion:
[[528.61553955   0.         280.96299366]
 [  0.         506.72329712 898.18433963]
 [  0.           0.           1.        ]]
```

```
Results for fixed PP:


################   Task: 1   ################
K:
[[2.91338355e+03 0.00000000e+00 7.49500000e+02]
 [0.00000000e+00 2.93174385e+03 4.99500000e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


error:0.049383954659421705


K without distortion:
[[2.85157568e+03 0.00000000e+00 7.48918290e+02]
 [0.00000000e+00 2.86982690e+03 4.98380284e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


################   Task: 2   ################
K:
[[2.98173835e+03 0.00000000e+00 7.49500000e+02]
 [0.00000000e+00 2.97097280e+03 4.99500000e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


error:0.05435829525468683


K without distortion:
[[3.05879028e+03 0.00000000e+00 7.48763515e+02]
 [0.00000000e+00 2.94823096e+03 4.97486525e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

```
################   Task: 3   ################
K:
[[3.03619768e+03 0.00000000e+00 7.49500000e+02]
 [0.00000000e+00 3.02674327e+03 4.99500000e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


error:0.058683946745854185


K without distortion:
[[2.99091260e+03 0.00000000e+00 7.47076832e+02]
 [0.00000000e+00 2.97633667e+03 4.98821270e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


################   Task: 4   ################
K:
[[2.99029105e+03 0.00000000e+00 7.49500000e+02]
 [0.00000000e+00 2.99666624e+03 4.99500000e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]


error:0.05697263729353706


K without distortion:
[[2.90186548e+03 0.00000000e+00 7.47748629e+02]
 [0.00000000e+00 2.90685083e+03 4.97883754e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

همانطور که مشاهده می‌شود خطای ماتریس بدست آمده با تصاویر ۱ تا ۱۰ کمینه است و اگر بخواهیم فاصله کانونی را براساس آن بگوییم این فاصله برابر (۲۹۳۱، ۲۹۱۳) میلی‌متر می‌شود (البته این اعداد تا حدی بزرگ بنظر می‌آیند و منطقی است برحسب پیکسل باشند اما با توجه به داک توابع و چون مختصات‌های سه بعدی را برحسب میلی‌متر دادم این اعداد را برحسب میلی‌متر گزارش کردم).