پیدا کردن خطوط سفید خیابان به کمک پردازش تصویر (فاز نهایی)

در این فاز قرار است مشکلات فاز پیش را حل کنیم. اگر یادتان باشد، در فاز قبلی یک سری مشکلات داشتیم که حل نمی‌شد (کوچک‌ترینش سایه درخت‌ها بود). در این فاز سعی می‌کنیم (واقعا سعی می‌کنید لازم نیست به نتیجه برسید) که آن مشکلات را حل کنیم. همچنین در این فاز با چند تبدیل جدید آشنا خواهیم شد. برای شروع مثل پروژه قبل لازم است که آدرس گیت زیر رو بارگذاری کنید و طبق همین راهنما، مراحل را جلو بروید. اگر در فاز قبلی OpenCV یا خود محیط اجرایی را نصب کردین که هیچ؛ واگرنه همچنان فقط به OpenCV نیاز دارید! (برای چگونگی نصبش به داکیومنت فاز قبلی مراجعه کنین)

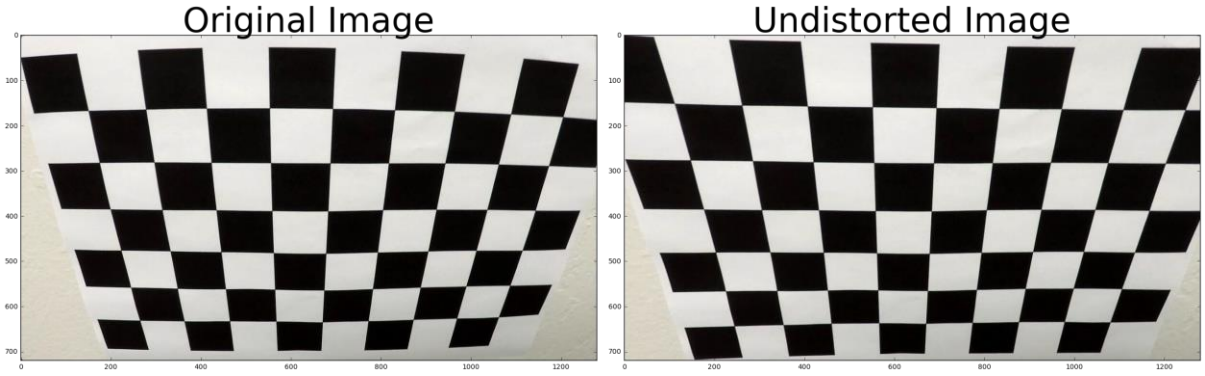
https://github.com/udacity/CarND-Advanced-Lane-Lines

۱- Distortion Correction:

در این پروژه مشکلی که عکس‌های ما دارند، آن است که به خاطر لنز دوربین و پرفکت نبودن آن، عکس‌ها انحنا دارند که این موضوع باعث می‌شود خطهایی که ما تشخیص می‌دهیم انحنا داشته باشند. برای همین اصطلاحا باید عکس‌هایمان را کالیبره کنیم؛ یعنی انحنایشان را از بین ببریم. معمول‌ترین کار برای این موضوع استفاده از صفحه‌ی شطرنج است. چرا که می‌دانیم این صفحه مربع است. برای همین یک سری عکس از این صفحه شطرنج با دوربین گرفته و تابع کالیبر را از آن پیدا می‌کنیم. سپس برای کالیبره کردن عکس‌هایی که از خیابان می‌گیریم از همین تابع بدست آمده استفاده می‌کنیم.

یک سری عکس صفحه‌ی شطرنج در پوشه پروژه موجود است. در خود OpenCV دو تابع است که این کار را برایتان انجام می‌دهند. پس نیاز نیست شما کار خیلی خاصی انجام دهید.

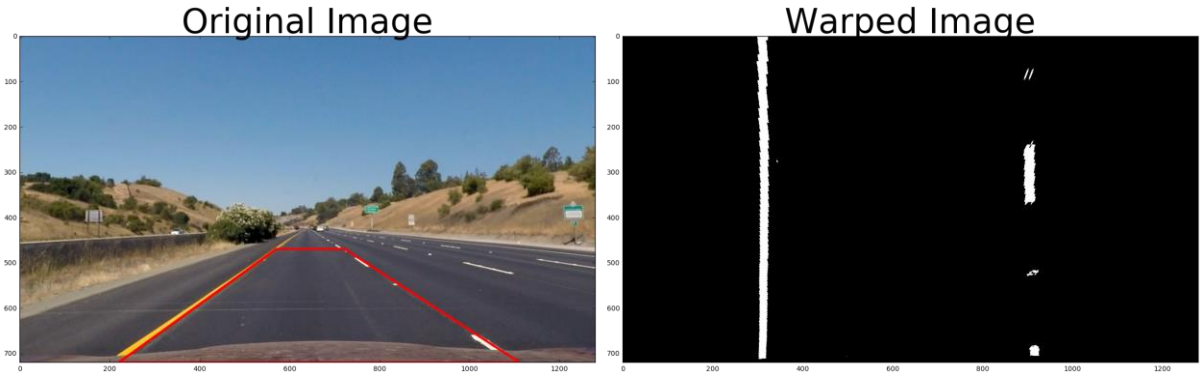
1. findChessboardCorners
2. calibrateCamera



۲- Perspective Transform یا Bird Eye View:

یکی از ساده‌ترین تبدیل‌های موجود در پردازش تصویر همین تبدیل است. دومین مشکل دوربین برای ما این است که تصاویر از جلو گرفته می‌شوند ولی ما برای تشخیص خط‌ها ترجیح می‌دهیم عکس‌ها از بالا (آسمان) گرفته شوند تا راحت‌تر بتوانیم رگرسیون بگیریم. این تبدیل یک تبدیل خطی از یک فضای دوبعدی به یه فضای دوبعدی دیگر است. به این صورت که شما ۴ نقطه از تصویر فعلی‌تان انتخاب می‌کنین و تعیین می‌کنید که در فضای جدید می‌خواهید به کدام نقاط نگاشته شوند. (طبعا وارون این تبدیل کمک می‌کند که عکس آخرتان را برعکس کنید!)

پیاده‌سازی این تبدیل بسیار ساده و به کمک OpenCV و تابع warpPerspective قابل انجام است. صرفا کافی است که داکیومنت آن را مطالعه کرده و از همین تابع استفاده کنید.



Sobel Transform -۳

از این تبدیل به عنوان آخرین تبدیل استفاده خواهید کرد. با استفاده از این تبدیل سایه‌ها و خطوط اضافی حذف می‌شوند. این تبدیل که عملا یک کانوولشن گسسته است، به شما کمک می‌کند تا خطوط مرزی (لبه‌ها) را شناسایی کنید. به این صورت که در هر جهت x - y می‌توانید این تبدیل را پیاده کنین.

برای انجام این تبدیل می‌توانید از تابع Sobel استفاده کنین و به عنوان آرگومان‌های آن، عکس و یک بازه را تعیین نمایید. (برای بدست آوردن بازه مناسب باید کمی آزمون و خطا (بازی!) کنید)

Color Filter -۴

قسمت جالب این پروژه، انجام فیلتر نهایی است. تا اینجا اکثر برخوردهایی که با عکس‌ها داشتیم به صورت RGB بوده. یکی دیگه از روش‌های بیان کردن رنگ‌ها HSV و HSL است. در لینک زیر می‌توانید اطلاعات بیشتری در مورد این که این نمایش چطور کار می‌کند پیدا کنید.

https://en.wikipedia.org/wiki/HSL_and_HSV

خودتان حدس بزنید که چطور می خواهید از فیلتر استفاده کنید! سعی کنید فقط عکستان را به این فرمت تبدیل کنید و سعی کنید با پارامترهای H و S و L بازی کنین (!) تا ببینید کدام پارامتر احتمالا فقط به دردتان خواهد خورد. (در هر مرحله یکی از آن‌ها را حذف کنید و ببینید عکس خروجیتان به چه صورت خواهد شد)

۵- جمع‌بندی

بعد از این که همه این کارها را کردید، رگرسیون بگیرین!

۶- نکات نهایی

این پروژه از فاز قبلی سخت‌تر است. فلذا اگر به جواب نرسیدید هم اشکالی ندارد. صرفا در هر مرحله همان کاری که خواسته شده را پیاده کنید و تا هر کجا رسیدین آپلود کنید. همچنان در صورت داشتن سوال، اگر زیاد بود یک جلسه رفع اشکال برگزار کنیم. اگر کم بود به alimohammad1995@gmail.com ایمیل بزنید و یا بیایید شرکت. تا جای ممکن نمره اضافی در نظر خواهم گرفت، و به صورت کلی هر کار جالبی که توانستید، انجام دهید. و در موقع تحویل در مورد آن حرف خواهیم زد.