بسمه تعالی
دانشگاه صنعتی شریف
دانشکده علوم کامپیوتر
Image processing - دکتر مصطفی کمالی
HW3 q1 report

سوال یک -Hough Transform:

این سوال از دو فایل تشکیل شده ، فایل اول به نام function.py که توابع مربوطه در آن قرار دارند و فایل کد اصلی به نام q1.py ، در ابتدا توابع موجود در function.py را توضیح میدهیم که 9 تابع در آن موجود است .

normalize : این تابع به اینصورت است که ماتریس ورودی را گرفته ، کمترین مقدارش را از هم درایه ها کم میکند که باعث میشود کمترین مقدار برابر صفر شود ، سپس همه درایه ها را تقسیم بر بیشترین مقدار کرده و سپس ضربدر 255 کرده و به نوع uint8 میبرد ، لذا باعث میشود که درایه های ماتریس ورودی که اگر بین a تا b بوده اند بین 0 تا 255 و به صورت integer قرار بگیرند و مناسب نمایش و ذخیره کردن شوند.

hough_space : این تابع برای ساختن ماتریس accumulator و ایجاد Hough space به کار میرود ، به اینصورت که عکس ورودی را ابتدا سیاه سفید کرده و سپس مربعی میکند و یک فیلتر گاوسی سه در سه با زیگما صفر اعمال میکند تا نتیجه اندکی بهتر شود ، سپس با استفاده از تابع cv2.canny ، edge ها را در تصویر بدست می‌آورد ، فاصله d برابر رند شده‌ی قطر این عکس مربعی شده است و آن را حساب میکنیم .برای تشکیل فضای hough به اینصورت عمل میکنیم که بازه θ از π- تا π به تعداد 360 عدد یا به عبارتی هر یک درجه ، همچین ρ از مقدار d- تا d به تعداد 2d میباشد یعنی هر اختلاف دو ρ متوالی برابر یک است ، سپس ماتریس accumulator را به تعداد سطر برابر تعداد ρ ها و ستون‌ها به تعداد θ ها میباشد. برای پر کردن accumulator به اینصورت عمل میکنیم که که درایه به درایه در تصویر edge جلو میرویم و اگر یک آن درایه جز edge بود ، برای مختصات آن و برای هر θ ، مقدار ρ را طبق فرمولی که در اسلایدهای درس آمده یعنی $x . \cos(\theta) + y . \sin(\theta)$ مقدار ρ متناظر را بدست میآوریم ، سپس در ماتریس accumulator در ستون θ مربوطه میبینیم که این ρ بدست آمده به ρ متناظر با کدام سطر نزدیک تر است ، آن سطر را پیدا کرده و سپس درایه مربوطه را یک عدد زیاد میکنیم و یک vote دیگر به آن اضافه میشود ، در نهایت ماتریس accumulator بدست میآید که متناظر فضای hough است و سپس آنرا به همراه تصویر edge در خروجی میدهیم ، نکته ای که باید توجه کرد اینست که در درس گفته شد که مبدا x,y را مرکز عکس و بگیرید ، ولی ما گوشه بالا چپ گرفتیم و که پیاده سازی ساده تر شده و تفاوتی از نظر مفهوم نمیکند.

find_lines : این تابع سه ورودی دارد ، ورودی اول عکس اصلیست که مربعی فرض شده و صرفا dimension آن نیاز است ، سپس ماتریس accumulator و یک threshold میگیرد که مقدار default برای threshold 150 فرض شده ، حال به استفاده از scipy.signal.argrelextrema ماکزیمم های محلی در ماتریس accumulator را بدست می‌آوریم که اگر از مقدار threshold بیشتر بود ، θ , ρ متناظر را به عنوان θ, ρ خط بدست آمده به آرایه lines اضافه میکنیم و در نهایت lines را در خروجی میدهیم.

Make_mb : این تابع ارایه my_lines را میگیرد که هر سطر آن دو مقدار θ , ρ دارد و سپس مقدار شیب و عرض از مبدا خط متناظر را بدست می‌آورد و در آرایه lines به اینصورت اضافه میکند که هر سطر آن برابر m,b,θ,ρ است و اگر اختلاف مقدار θ با ±π , ±π/2 کمتر از 0.2 درجه باشد ، خط عمودی / افقی در نظر گرفته شده و به جای m مقدار ρ و به جای b مقدار ±np.inf میگذاریم که مثبت یا منفی بودن np.inf نشان دهنده عموی یا افقی بودن خط است .

avg_near_line(lines_in, th1=0.036, th2=0.15) : این تابع خطوطی که اختلاف ρ آنها کمتر از th1 که مقدار پیشفرض آن برابر 3.6 درصد و اختلاف θ آنها کمتر از th2 باشد (15 درصد مقدار پیشفرض) را در یک دسته قرار داده و میانگین θ , ρ کل آن دسته را به عنوان یک خط در نظر میگیرد و به اینصورت تعداد خط های ورودی را را کاهش میدهد و خروجی میدهد.

select_parallel_lines(in_lines, th=2) : این تابع لیست خطوط ورودی را بر اساس ρ مرتب میکند و آنهایی که ρ آنها اختلافشان از 2 درصد کمتر باشد را موازی فرض میکند و در یک دسته قرار میدهد ، سپس دسته هایی که تعداد خطوط در آنها بیشتر از th باشد در نظر میگیرد و بقیه را دور میریزد ، با این کار خط هایی که تعداد خطوط موازی آناه از حدی بیشتر از نگه داشته شده و بقیه دور ریخته میشوند که برای نگه داشتم خطوط روی صفحه شطرنج استفاده شده است .

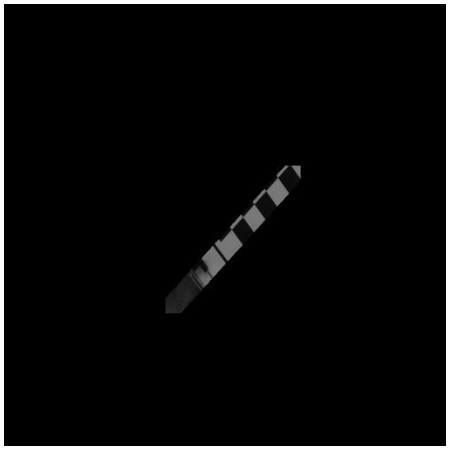
draw_line : این تابع به اینصورت است که که خطوط داده شده را روی تصویری که ورودی آن آمده است رسم میکند، به اینصورت که برای x=±4000 مقدار y متناظر را بدست میآورد و خطی بین این دو نقطه رسم میکند. همچنین اگر خط عمودی و یا افقی باشد با توجه به ρ آن خط آنرا رسم میکند و تصویر نهایی را در خروجی میدهد.

Intersection : این تابع به اینصورت است محل تلاقی هر دو خطی که در آرایه خطها در ورودی آن داده مشود را درصورت وجود با استفاده از فرمولهایی که با استفاده از هندسه دکارتی مقدماتی بدست میآید محاسبه میکند و اگر شیب دو خط خیلی نزدیک هم باشدبه عنوان خطا حساب کرده و محل تلاقی را حساب نمیکند ، بلکه ما به دنبال محل تلاقی خطهای تقریبا عمود برای گوشه های صفحه شطرنج هستیم. سپس با همان تابع avg_near_line محل های تلاقی که نزدیک هم هستند را میانگین میگیریم که مقدار threshold ها هم متناسب شده است ، پیاده سازی تابع avg_near_line به صورتی است که هم برای خطوط هم برای نقاط نزدیک هم کار میکند.

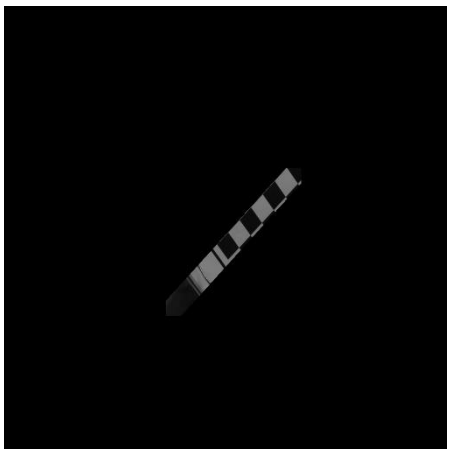
شاید مشکل ترین بخش این سوال جدا کردن خط های روی شطرنج بود ، برای این کار ابتدا از تابع find_chess_lines استفاده شده و سپس خروجی آنرا به select_parallel_lines داده ایم تا خطهای اضافی باقیمانده را حذف کند.

find_chess_lines : این تابع برای هر خط داده شده یک نوار دو طول آن خط در بالا و پایین عکس با یک margin در نظر میگیرد که ای نوار در طول کل خط نیست بلکه از کناره های عکس مقداری فاصله دارد که این margin و فاصله را با آزمون و خطا بدست آوریم ، سپس برای این دو margin روش pattern recognition را یک بار برای margin پایینی در بالایی و بار دیگر برعکس انجام داده و میانگین میگیریم .از عدد بدست آمده که با استفاده از روش normalized cross correlation یک عدد بین 0 و 1 میشود ، حال اگر خط روی صفحه شطرنجی باشد ، این دو margin شبیه همند و این عدد حدود 0.7 درمیاید که یک بازه تعیین کرده ایم که خطوطی که این عدد برای آنها در این بازه قرار دارد را جدا میکند ، برای یک خط که در صفحه شطرنجی قرار دارد این عدد و همین طور margin های بالا و پایین که در کد به نام mask_up , mask_down است به صورت زیر است : و عدد بدست آمده برابر : 0.6295

Mask_up



Mask_down



حال فقط خطهایی که این عدد برای آنها بین 0.53 تا 0.78 است را نگه میداریم .

حال که توابع را توزیح دادیم به فایل q1.py میپردازیم ، این فایل یک تابع main دارد که اسم عکس که میخواهیم را ورودی میگیرد و به ما در خروجی مقدار : [edge, acc, acc_big, img_line, img_chess_line, img_corners] میدهد که edge برابرتصویری مربوط به لبه هاست ، acc برابر ماتریس accumulator و یا hough space ماست ، همچنین چون از نظر ظاهری این ماتریس برای نمایش دادن خوب نبود علاوه بر آن ، اندازه آن را اسکیل کرده و اندکی به مربعی نزدیک کردیم و مقادیرش را بزرگ و بین 0 تا 255 بردیم و یک عکس اضافه بر عکس های خواسته شده ذخیره کردیم که مربوطه به همین است و نام آن به صورت res03-hough-space-big.jpg است . img_line برابر عکسی

که خطها رویش کشیده شده ، img_chess_line عکسی که خطهای شطرنج در آن کشیده شده ، img_corners عکسی که گوشه ها در آن رسم شده است .

در ابتدای تابع main عکس را ورودی گرفته ، dimension را نگه میداریم و آنرا مربعی میکنیم ، سپس با تابع hough_space ماتریس acc و edge را بدست میآریم و با استفاده از تابع fine_lines و ماتریس acc ، خطها را بدست میآریم و به تابع avg_near_line میدهیم تا خطهای خیلی نزدیک میانگین گرفته شوند ، سپس با استفاده از دو تابع find_chess_lines و select_parallel_lines خطهای روی شطرنج را پیدا میکنیم و در نهایت نقاط تداخل آنها را باتابع intersection بدست آورده و سپس خطها را با draw_lines رسم میکنیم و سپس عکس های بدست آمده را به dimension اولیه برمیگردانیم و از حالت مربعی خارج میکنیم.

حال برای هر دوعکس ، عکس های خواسته شده را بدست آورده و ذخیره میکنیم