توضیح تابع sift_feature_matching:
ابتدا به کمک sift, نقاط کلیدی و توصیفگر های 2 تصویر را به دست آورده. سپس با استفاده از یک الگوریتم brute force به نام bf.matcher, بین نقاط کلیدی از تصویر 1 و 2 با استفاده از توصیفگر ها, نقاطی که به یکدیگر شبیه هستند را به دست آورده. سپس نقاط تطبیق یافته را بر اساس فاصله اقلیدسی توصیفگر ها مرتب کرده. در انتها نقاط را به 2 لیست اضافه کرده.

توضیح تابع ransac_homography:
دقت کنید که صرفا برای یافتن ماتریس تبدیل 4 نقطه به 4 نقطه از تابع آماده استفاده شده و بقیه قسمت ها کاملا از صفر پیاده سازی شده. به این گونه عمل شده که برای یافتن بهترین تبدیل, به تعداد num_iterations یا همان w در ransac لوپ زده ایم, در هر ایتریشن 4 نقطه کلیدی از تصویر 1 انتخاب کرده و میبینیم که به چه نقاطی تبدیل شده. با استفاده از این نقاط ماتریس H را به کمک تابع getPercpectiveTransform میابیم. حال این تبدیل را روی کل نقاط تست کرده و تعداد inlier ها را پیدا کرده و در آخر بهترین تبدیل را میابیم.

توضیح تابع stitch_images:
در این تابع به کمک تابع warp, تمام نقاط موجود در تصویر 1 را تحت تبدیل H به یک تصویر 900*480 میبریم. این اعداد به صورت دستی به دست آمده. سپس تصویر 2 را سر جای اصلیش میگذاریم. همان طور که در خروجی میبینیم, شکل ها به خوبی به یکدیگر چسبیده اند و یعنی تابع تبدیلمان درست است.