در ابتدا تصویر اصلی را به ۳ قسمت مساوی تقسیم کردم و به ترتیب از بالا به پایین چنل های B و G و R آن ها را نامیدم. حال برای آنکه پیدا کنیم که چگونه این ۳ تصویر روی هم مچ می شوند، لازم است که ابتدا یکی از چنل ها را فیکس قرار دهم و چنل های دیگر را روی آن حرکت دهم تا مقدار best_i و best_j ای را پیدا کنم که به ازای آن چنل ها روی هم فیکس می شوند. بدین منظور چنل آبی را فیکس در نظر گرفتم و با انجام یک for اقدام به پیدا کردن best_i و best_j کردم. در ابتدا چنل سبز را در نظر گرفتم. هدف این است که best_i و best_j پیدا کنم که اگر چنل سبز روی چنل آبی در جهت y ها به اندازه best_y و روی چنل x ها به اندازه best_j جابه جا شود آن ها فیکس خواهند شد.

بدین منظور با دو for تو در تو اقدام به حرکت دادن چنل سبز روی چنل آبی کردم. و به ازای هر بار، محاسبه کردم که اختلاف دو ماتریکس (در ناحیه ای که مطابق اند) چه قدر می شود. این کار را هم برای L_1 و هم برای L_2 انجام دادم. (ورودی L تابع find_match_B_and_G_channelهمین مقدار را مشخص می کند)

حال به ازای هر کدام از i و j ها که این مقدار (جمع توان ۲ درایه ها) کم ترین مقدار شد، این بهترین نتیجه برای ما خواهد بود.

چون به ازای مقادیر زیاد (مثلا حدود ۱۰۰ پیکسل) انجام این for تودرتو بسیار طول میکشید، بعد از انجام یکبار آن که روی مقادیر 100+ و -100 که بیش از دو ساعت طول کشید، محدوده for ها را به محدوده کوچکی حول جواب اصلی محدود کردم تا پیدا کردن جواب کم تر طول بکشد.

این for ها را یکبار برای فیکس کردن چنل سبز روی آبی و بار دیگر برای فیکس کردن چنل قرمز روی آبی انجام دادم و با best_i و best_j هایی که بدست آمد، عکس ها را روی هم انداختم که نتیجه قابل مشاهده است. با وجود چندین پیکسل اختلاف، به نظرم نتیجه قابل قبول است.

در کنار فایل اصلی f03.py یک فایل helper.py قرار دارد که تابع هایی که از آنها استفاده کرده ام را در آنجا نوشته ام تا به خوانایی کدم کمک کند.