a\_img = cv2.imread('a.png', 0)

یک a کراپ کردم و اسمش a.png

h = a\_img.shape[0]  
w = a\_img.shape[1]

ارتفاع و عرض تصویر را بدست میاریم

c = signal.correlate2d(text\_img, a\_img)

بعد کرولیشن تصویر ورودیمون a\_text و a\_img را بدست میاریم. وقتی کرولیشن حساب میشه خروجی هایی بیشترین مقدار را خواهد داشت که تصویر 2 تا a روی همدیگه قرار بگیرند

m = c.max()

بنابرایم max تصویر کرولیشن c را بدست میاریم

inds = (c == m).nonzero()

و پیدا میکنیم اون maximum در چه نقطه ای اتفاق افتاده و در نهایت اون نقطه inds برای ما مشخص میکنه کجاها a قرار داره

ax = plt.subplots(1)  
ax[1].imshow(text\_img, cmap='gray')

بنابراین کاری که میکنیم یک subplot از خود تصویر رسم میکنیم

rec = Rectangle((inds[1][i] - w, inds[0][i] - h), w, h, edgecolor='r', facecolor='none')  
ax[1].add\_patch(rec)

و بعداز اون کاریم که انجام میدیم یک مستطیل رسم میکنیم اونجاهایی که پیدا شده اندیس های مختلفش مشخص میشود کجاها تصویر قرار داره

a\_img = cv2.rotate(a\_img, 2)

چون a های ما بصورت سطری و ستونی قرار دارند برای حالت دوم حالتی که a ها عمودی هستند کاری که انجام میدیم بر خلاف جهت عقربه های ساعت تصویر a\_img را میچرخوانیم این 2 یعنی چرخواندن در 90 درجه بر خلاف عقربه های ساعت میتوانستم این را هم بنویسم. cv2.cv2.ROTATE\_90\_COUNTERCLOCKWISE

سعی کنیم منطبقش کنیم بر a های به صورت عمودی بعد که تصویر a\_img را جرخواندم

h = a\_img.shape[0]  
w = a\_img.shape[1]

ارتفاع و عرض را بدست میارم و

c = signal.correlate2d(text\_img, a\_img)

همان کار قبلی را انحام میدم

برای فضای فرکانسی هم منطقمان این است اما اگر یم تصویر را کرولیشنشان را با هم پیدا کنیم معادل این است که در فضای فرکانسی در هم ضربشان کنیم فقط تنها تفاوتی که وجود داره وقتی دوتا تصویر کرولیشنشان را بدست میاریم تصویر دومی a\_img باید سطرها و ستون هاش معکوس شود

a\_img = a\_img[::-1, ::-1]

سطرها و ستون ها منهای یک میشوند

رابطه بین corolation 2 تا signal در فضای مکان و را بطه آنها در فضای فرکانس که اگر یکی از آنها را ما معکوس کنیم و اینها رو در فضای فرکانس ببریم به فضای FFD و ضرب کنیم مثل این است که corolation شان را بدست آوردیم

max\_w = max(text\_img.shape[1], a\_img.shape[1])  
max\_h = max(text\_img.shape[0], a\_img.shape[0])

ماکسیمم عرض و ارتفاع تصویر اول و دوم را بدست آوردم زیاد هم نیاز نبود میدانیم که تصویر text\_img عرضش بیشتر خواهد بود و ارتفاعش هم طبیعتا بیشتر خواهد بود

text\_img = np.pad(text\_img,   
 [(0, max\_h - text\_img.shape[0]),   
 (0, max\_w - text\_img.shape[1])],   
 mode='constant', constant\_values=0)

بعد اندازه تصویر اونی که کوچکتر هست با 0 pad میشه و اون یکی نخواهد شده چون ما گفتیم به اندازه اون چیزی که کمتر دارد max عرض

a\_img = np.pad(a\_img,   
 [(0, max\_h - a\_img.shape[0]),   
 (0, max\_w - a\_img.shape[1])],   
 mode='constant', constant\_values=0)

هر دو تصویر با 0 pad میشوند تا هم اندازه شوند

text\_img\_fft = np.fft.fft2(text\_img)  
a\_img\_fft = np.fft.fft2(a\_img)

بعد میبریم در فضای فرکانس

return np.real(np.fft.ifft2(text\_img\_fft \* a\_img\_fft))

و در همدیگر ضربشان میکنیم و بعد از اون fft2 اینورس میگیریم ازشون برشون میگردونیم به فضای مکان و بخش real را به عنوان خروجی بر میگردانیم این همان کرولیشن را انجام میدهد ولی از فضای فرکانس استفاده میکند.

h = a\_img.shape[0]  
w = a\_img.shape[1]  
c = process\_in\_freq\_domain(text\_img, a\_img)

منطق همان قبلی است تنها تفاوت این است که وقتی من از فضای فرکانس استفاده میکنم چون یک بار میبرم به فضای fft و از فصای fft بر میگردانم اتفاقی که میافتد این هست که یک سری خطا این وسط اتفاق میافتد خطای گرد کردن و اینها و من وقتی corolation را از طریق فضای فرکانس بدست میارم یک مقدار خطا داشته باشم بعضی جاها a که با هم دیگه match میشوند از لحاظ کرولیشنی ممکن است یکیشون از دیگری بیشتر بشه چون چندتا a بصورت افقی داشتیم ممکن است یکی از بقیه بیشتر بشه

m = c.max()  
inds = (c >= 0.99 \* m).nonzero()

من فکر کردم max را بدست بیارم و آنهایی را به من بدهد که بزرگتر یا مساوی 99 درصد اون ماکسیموم هستند. چون حالا یا خطا است یا تو فضای فرکانس یک اتفاقاتی میافتد به هر حال یک سری خطا اینجا بوجود میاد که برای تمام a ها این با هم دیگه این مقدار ماکسیموم برابر نمیشود بنابراین برای اینکه تمام a ها را داشته باشیم 99 درصد مقدار ماکسیموم را در نظر گرفتم که برام پیدا کنه و برگردونه

برای a های عمودی هم همون منطق دوباره تکرار شده بدست میاره و ذخیره میکنه.