دوتا تابع اصلی داریم erosion و Dilation

p\_img = np.zeros((img.shape[0] + 2, img.shape[1] + 2))

p\_img[1 : img.shape[0] + 1, 1 : img.shape[1] + 1] = img

اینجا من pad کردم تابع ورودیم را برای اینکه اندیس ها رو بتونم حرکت بدم روی کل آرایه ورودی

for i in range(1, img.shape[0] + 1):

for j in range(1, img.shape[1] + 1):

کار بعدی که انجام شده من اومدم روی پیکسل ها دانه دانه حرکت کردم این 2 تا حلقه for این کار را انجام میدهد.

p = p\_img[i - 1 : i + 2, j - 1 : j + 2]

بعد برای هر برای هر پیکسل یک همسایگی 33x را از اینجا برداشتم

er\_img[i - 1, j - 1] = (p \* sl)[sl == 1].min()

ضربدر structure element کردم. در عمل erosion وقتی شما محدوده همسایگی را در sl ضرب میکنید 3 تا اتفاق ممکن است بیافتد:

اتفاق اول اون پیکسلی که شما ضرب میکنید در sl خودش و تمام همسایگیهاش 0 شود و اگر این اتفاق بیافتد نتیجه ای که میشه گرفت این هست که وقتی p پیکسل را در sl ضرب میکنید خروجی هم باید صفر شود خب پیکسل خودش صفر است با ضرب نظیر به نظیر همه صفر میشود و وقتی min میگیریم خروجی 0 خواهد شد.

er\_img[i - 1, j - 1] = (p \* sl)[sl == 1].min()

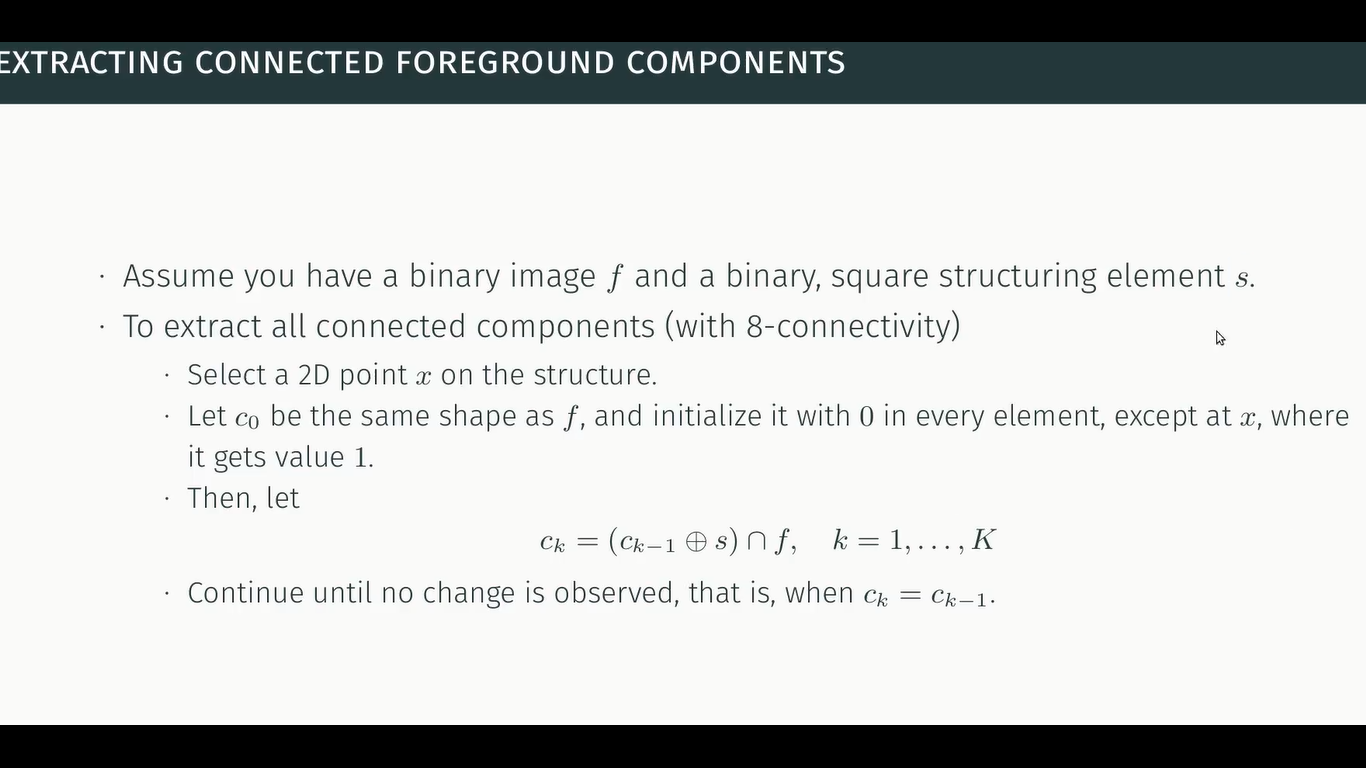
حالت دوم این هست که این p پیکسل شما و همسایگیش 0 نباشد در اینصورت با ضرب در sl طبیعتا بعضی جاها برای ما 1 ممکن است بدست بیاد وقتی 1 بدست بیاد خب شما sl مقدار 1 داره دیگه و وقتی min بگیریم چون erosion هست اگر hit اتفاق بیافتد شما باید خروجی 0 برگردونیم یعنی اون پیکسل را 0 کنیم طبیعتا چون بعضی جاها خواهد گرفت و بعضی جاها نخواهد گرفت در ضرب با همدیگه match نخواهد شد و min خواهد شد صفر و اینجا صفر خواهد شد.

و تنها در صورتی در erosion خروجی 1 خواهد شد که تمام پیکسل هایی که در همسایگی وجود دارند و تمام پیکسل های 1 که در sl قرار دارد با هم match شوند. برای همین اون پیکسلهایی نگاهم میکنیم که مقدار sl==1 باشند. اگر همش مقدار 1 داشته باشند یعنی کاملا sl با این شی ما همسایگی پیکسل های ما هماهنگ باشد match باشد Min که بگیریم مقدار 1 خواهد داشت و خروجی 1 خواهد شد.

dl\_img[i - 1, j - 1] = (p \* sl).max()

در dialation همه این اتفاقات میافتد تنها تفاوت در dialation شما وقتی sl و همسایگی p پیکسل را در هم ضرب میکنید خب اگر به صورت کامل match بشه شما باید 1 برگردونید اگر حتی بخشی هم برخورد داشته باشه شکا باز باید 1 برگردونید فقط در صورتی پیکسل خروجی شما 0 میشه که همسایگی ما همش 0 باشدکه در اینصورت max 0 میشود و خروجی 0 را خواهیم داشت.

تابع بعدی تابعی است که برای ما connected component ها را برای ما بر میگردونه



ما یک sl را به صورت مربعی در نظر میگیریم و فرض کنید ورودی ما f اون تصویری که میخواهیم connected component ها ار از داخلش پیدا کنیم f باشد.

کاری که انجام میدیم این هست که میایم یک نقطه غیر صفرx داخل f پیدا میکنیم

بعد یه تصویر جدید میسازیم c0 که دقیقا مثل f هست هم اندازه f است همش 0 است بجز محل پیکسل x که همش 1 است.

پس یه تصویر جدید میسازیم که هم اندازه f است همه پیکسلهاش 0 است بجز اون پیکسل x که الان پیدا کردیم.

بعد این کار رو به صورت iterative تکرار میکنیم. کاری که انجام میدیم تصویر c را با sl مربعی شما عمل dialation را انجام میدیم بعد اشتراکش را بدست میاریم با تصویر اصلی f که میخوایم از component ها را بدست بیاریم. و این کار را به صورت بازگشتی انجام میدیم تا زمانیکه تصویر ck تغییری پیدا نکند. و وقتی تغییری پیدا نکرد اون چیزی که پیدا شده یک connected component برای ما خواهد بود.

sl = np.ones((3, 3))

sl را مربعی 1 در نظر گرفتم.

cc = np.zeros(img.shape)

تمام connected component هایی را که پیدا میکنم در cc خواهد بود.

label = 1

کاری که اینجا میکنم برای هرکدام از cc هایی که پیدا کردم داخل تصویر cc به پیکسلهای اون عدد label را انتصاب میدم cc اول را که پیدا کردم پیکسلها عدد 1 خواهد بود دومی عدد 2 خواهد بود و...

while True:

c = np.zeros\_like(img)

flag = False

while i < img.shape[0]:

if j == img.shape[1]:

j = 0

while j < img.shape[1]:

if img[i, j]:

flag = True

کاری که اول اینجا انجام میدم توی این 2 تا حلقه تو در تو یک پیکسل که مقدار 1 داشته باشه از داخل تصویر ورودیم پیدا میکنم

if flag:

break

و اگر پیکسل یکی پیدا میکردم یعنی کار تموم شده بود دیگه پس تا زمانی که پیکسل یکی وجود داشته باشه flag وقتی که از حلقه هم خارج بشه مقدار true داره وگرنه اگر از حلقه خارج بشه و مقدار false داشته باشه یعنی کلا کار تمام شده و تمام پیکسل ها را استفاده کردیم.

i += 1

if not flag:

break

اگر flag ، false نباشد i و j نشان دهنده اندیس اون پیکسل x هستند پس

c[i, j] = 1

c همان c است داخل اسلاید بود. کار بعدی که انجام میدم

s = 1

یعنی s مجنوع پیکسل هایی است که داخل c هستند c صفر است ولی فقط یک دانه پیکسل داخلش 1 است.

while True:

c = im\_dilate(c, sl) & img

کاری که انجام میدم یعنی همون چیزی که به صورت بازگشتی انحام مشود. تصویر c را با sl ، dialation ش را انجام میدم و اشتراک بین دو تصویر باینری با & با تصویر ورودی img انجام میدم بعد c تغییر پیدا میکنه

temp = c.sum()

بعد جمع پیکسل های c را دوباره حساب میکنم

if temp == s:

اگر جمع پیکسل های c برابر شد با جمع پیکسلهایی که قبلا داشتیم اون موقع کار تموم شده یعنی connected component در c قرار دارد.

else:

s = temp

اگر برابر نبود s را تغییر میدم و جمع پیکسل ها برابر با temp قرار میدم و این کار را هی تکرار میکنم.

print(s)

این print برای این اینجاست که مجموع پیکسل ها را هی حساب کنه و چاپ کنه و ببینیم که کار داره جلو میره. این حلقه که تموم شده داخل connected component ,c پیدا شده.

cc[c == 1] = label

اون جاهایی که c==1 است داخا تصویر cc که تمام connected component ها بعدا داخلش ذخیره بشن

مقدار label را بشون انتصاب میدم .

img[c == 1] = 0

label += 1

بعد مقدار label را یکی افزایش میدم که برای connected component بعدی مقدار lable یکی بیشتر شده باشد.

ضمن اینکه اون پیکسلهایی connected component که تا حالا پیدا کرده بودیم را تو تصویر اصلی برابر با 0 قرار میدم که تو دور بعدی در پیدا کردن connected component دیگه شرکت نکنند. و در نهایت cc را برمیگردانیم.

img\_1 = mpimg.imread('img1.tif', format='gray')

img\_1 = img\_1[:, :, 1] > 0

اینجا تصویر اول خونده میشه

sl\_1 = np.ones((3, 3))

sl\_2 = np.array([[0., 1, 0], [1, 1, 1], [0, 1, 0]])

دوتا sl چنانچه تمرین خواسته بود را ایجاد کردم

img\_1\_dil\_1 = im\_dilate(img\_1, sl\_1)

img\_1\_dil\_2 = im\_dilate(img\_1, sl\_2)

img\_1\_er\_1 = im\_erode(img\_1, sl\_1)

img\_1\_er\_2 = im\_erode(img\_1, sl\_2)

dialation و erosion را روشون انجام دادم این بخش اول کار بود.

بخش دوم خواسته بود با استفاده از عملگر morphology لبه این را بدست بیاورید.

img\_2 = mpimg.imread('img2.tif', format='gray')

img\_2 = img\_2[:, :, 1] > 0

img\_2\_dil = im\_dilate(img\_2, sl\_1)

img\_2\_b = img\_2\_dil & np.logical\_not(img\_2)

برای اینکه لبه را بدست بیارم کافیه که من تصویر اولیه را dialation روش انجام بدم و بعد تصویر dialate شده را & کنم با not تصویر اولیه.

بع بخش آخر این بود که connected component ها را پیدا کنم

img\_3 = mpimg.imread('img3.tif', format='gray')

img\_3 = img\_3 > 0

تصویر اولیه را به عنوان ورودی میخونم کار بعدی که انجام میدم این هست که

cc = find\_ccs(img\_3)

همون تابع connected component را صدا میزنم

n\_ccs = cc.max()

ماکسیمم label که اونجا وجود داشت میشه تعداد connected component های من

cc\_colored = color.colorlabel.label2rgb(cc)

با استفاده از این تابع یک تصویر رنگی که نمایش داده میشه ایجاد میکنم و هر کدام از اون پیکسلها که مقدار 1,2,3,4 قرار دادیم را یک رنگ اختصاص خواهد داد.