img = cv2.imread('hti.png')

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

تصویر را خوندم و تبدیلش کردم به یک gray scale

canny\_thr1 = 50

canny\_thr2 = 100

edges = cv2.Canny(gray, threshold1 = canny\_thr1, threshold2 = canny\_thr2)

بعد لبه های تصویر را بدست آوردم. این 2 تا عدد threshold بالایی و پایینی edge detector canny هستند.

rho\_res = 1

theta\_res = np.pi / 180

گامهایی که استفاده میشه برای مقدار رو یک پیکسل یک پیکسل و برای تتا یک درجه یک درجه.

lines = cv2.HoughLinesP(edges, rho\_res, theta\_res, 200, minLineLength=200)

اینها ورودیهای hough transform هستند. این 200 تعداد حداقل پیکسل های است که یک خط باید داشته باشه تا ما اینجا بتونیم ببینیمش.

miLineLength این هم حداقل طول خطوط است با توجه به ابعاد تصویر

print(lines.shape)

این رو میتونیم نداشته باشیم. گفتم که چندتا خط را اینجا پرینت کنه که طبیعتا چندتا خط پیدا کردیم که 2 تاست.

cv2.line(img, (x1, y1), (x2, y2), (0 ,0 ,255), 1)

و درنهایت هر خط را رسم میکنیم این رنگ خط است این ابتدا و انتهای خط است و این ضخامت خط است که 1 پیکسل در نظر گرفته شده.

apple = cv.imread('apple.png')

apple = cv.resize(apple, (256, 256))

orange = cv.imread('orange.png')

orange = cv.resize(orange, (256, 256))

تصاویر را اینجا خوندم. چون هربار ما تصاویر را resize میکنیم تو پیرامید و کوچک میکنیم. ابعاد تقسیم بر 2 میشود من resize کردم به 256 که راحت تر انجام شود.

apple\_gp = [apple]

orange\_gp = [orange]

گاس ایمپلمنت را برای هردو میسازیم. لایه اول خود apple و orange هستند.

n\_layers = 5

5 تا لایه دیگه هم بش اضافه کردیم .

for i in range(n\_layers):

apple = cv.pyrDown(apple)

هر بار down sample میکنم apple را

apple\_gp.append(apple)

و اضافه میکنیم به گاوس ایمپلمنتیشن.

orange = cv.pyrDown(orange)

orange\_gp.append(orange)

apple\_lp = [apple\_gp[n\_layers]]

orange\_lp = [orange\_gp[n\_layers]]

در تکرار بعدی laplace gausian را باید حساب کنیم. در لاپلاس ایمپلمنت آخرین لایه ای که داریم میشه لایه اول گاسین ایمپلمنت آخرین لایه این 2تا با هم برابر هستند

for i in range(n\_layers, 0, -1):

از لایه انتهایی گاوسی ایمپلمنت شروع میکنیم از لایه 5 ام لایه ماقبل آخر

up = cv.pyrUp(apple\_gp[i])

هر لایه را هم برای apple هم برای orange , upsample را انجام میدهیم

sub = cv.subtract(apple\_gp[I – 1], up)

بعد upsample شدش رو از لایه بعدی کم میکنیم تا لاپلاس ایمپلمنت او لایش بدست بیاد

apple\_lp.append(sub)

و اضافش کنیم به لیستی که بارای لاپلاس ایمپلمنت apple است و برای orange هم همین کار رو انجام میدیم.

merge = []

for la, lo in zip(apple\_lp, orange\_lp):

در مرحله بعدی تک تک این لاپلاسین ها را بر میداریم برای merge کردن چون گفته که بصورت قطری تصویرها رو با هم merge کنیم

mask = np.zeros((m, n))

ما نیاز به ماسکی داریم که با استفاده از اون تصاویر را برش بدیم

mask[np.tril\_indices(m, -1)] = 1

یه ماتریس بالا مثلثی و پایین مثلثی میخوایم که اینجا یه پایین مثلثی ایجاد میکنیم. این 1- که اینجا وجود داره به این دلیل هست که ما نمیخواهیم مقادیر قطر اصلی 1 باشد

for i in range(0, m, 2):

mask[i, i] = 1

اما بصورت یکی درمیان 1 قرار دادم در قطر اصلی

mask\_a = np.stack((mask, mask, mask), axis=2)

و تصاویر ما چون 3 تا کانال دارن RGB ماسک ما هم باید 3 تا کانال داشته باشد پس یک ماسک 3 کاناله اینجا میسازیم

mask = np.zeros((m, n))

mask[np.triu\_indices(m, 0)] = 1

و برای اون یکی هم باید ماتریس بالا مثلثی باشد که قطرش 1 است

for i in range(0, m, 2):

mask[i, i] = 0

اینجا بصورت یکی درمیان قطر را صفر میکنیم یعنی به صورت زیگزاگی قطر اصلی یک پیکسل برا orange است یک پیکسل برا apple

a = np.multiply(la, mask\_a)

کاری که انجام میدم ماسک اول را در لاپلاسین apple ضرب میکنم

o = np.multiply(lo, mask\_o)

ماسک دوم را در لاپلاسین orange ضرب میکنم که ضرب کردن معادل این است ما تصاویر را برش میدهیم.

m = np.add(a, o)

بعد تصاویر را که به هم add کنیم تصویری خاهد بود که نصف از هر کدام است

merge.append(m)

بعد لایه های لاپلاسین را اضافه میکنم.

در واقع ما لاپلاسین ها را که بدست آوردیم بعد یه هرم جدید میسازیم که تو اون نصف از لاپلاسین هرکدام از تصاویر است.

apple\_orange = merge[0]

بعد که اینکار را انجام دادیم سعی تصویر اولیه را reconstruct کنیم

for i in range(1, n\_layers + 1):

apple\_orange = cv.pyrUp(apple\_orange)

apple\_orange = cv.add(apple\_orange, merge[i])

شروع میکنیم از پایین هرم دانه دانه تصاویر را upsample میکنیم و وقتی upsample کردیم اضافه میکنیم به تصویر apple\_orange که داریم .

در ابتدا لایه اول هرمی merge که ایجاد کردیم اضافه میشه به apple\_orange ما تصویر اصلی ما بعد دانه دانه اون تصاویر merge شده ای که داشتیم تصاویرشان نصف apple < نصف orange بود را upsample میکنیم و اضافه میکنیم به اون تصاویر قبلی که داشتیم در نهایت تصویر برای ما reconstruct خواهد شد.