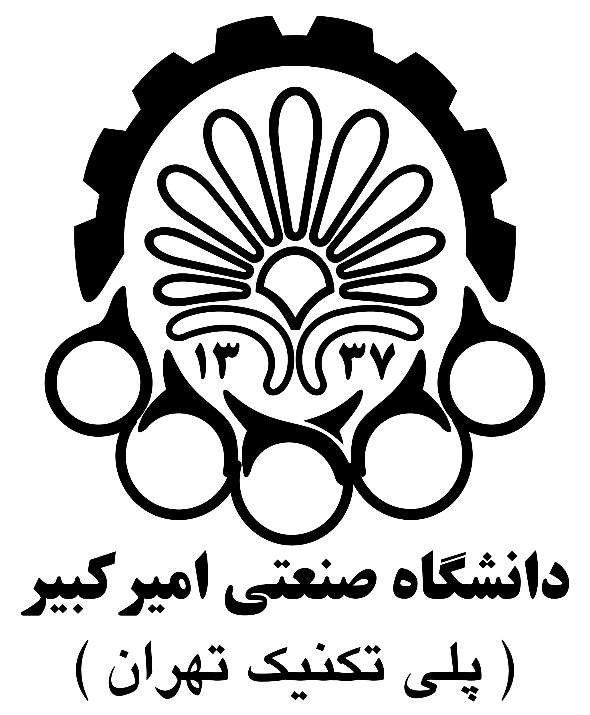
****

**دانشکده مهندسی کامپیوتر**

**تمر ین چهارم درس بینایی ماشین**

**دکتر صفابخش**

**غلامرضا دار 400131018**

**بهار 1401**

فهرست مطالب

[الف) 3](#_Toc120119247)

[ب) 5](#_Toc120119248)

[ج) 11](#_Toc120119249)

[د) 17](#_Toc120119250)

# الف)

استفاده مستقیم از K-Means بر روی مقادیر پیکسل­های تصویر، مشکلات زیادی دارد. یکی از مشکلات این کار، در نظر نگرفتن **موقعیت** **مکانی** پیکسل­ها و عدم توجه به مقادیر پیکسل­های همسایه است. نتیجه این امر این است که پیکسل­هایی با مقدار سطح خاکستری یکسان در یک خوشه قرار می­گیرند فارغ از اینکه واقعا به یک شئ یا یک بافت متعلق هستند یا خیر.

نتیجه قطعه­بندی دو تصویر داده شده به کمک این روش را در تصاویر زیر مشاهده می­کنید.



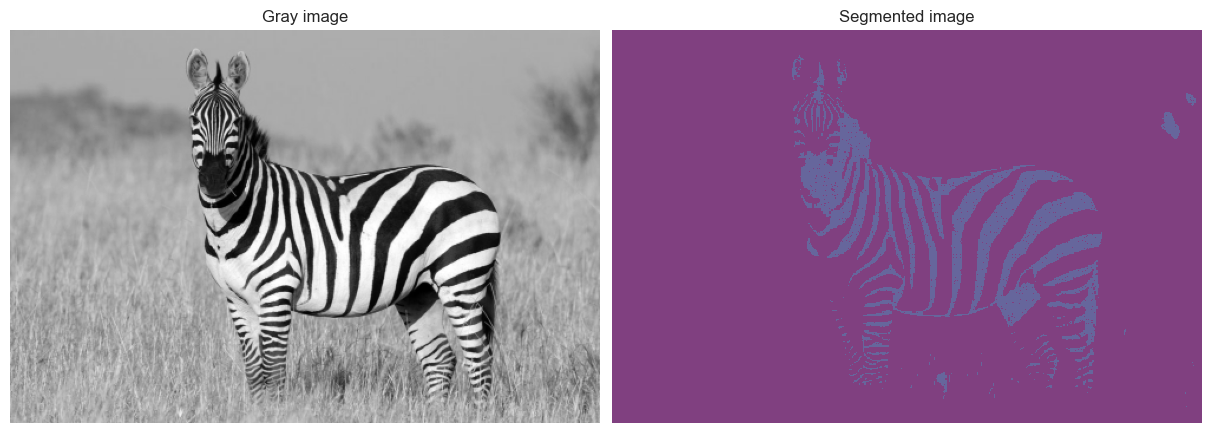
قطعه بندی با استفاده از K=3



قطعه بندی با استفاده از K=2



قطعه بندی با استفاده از K=3



قطعه بندی با استفاده از K=2

# ب)

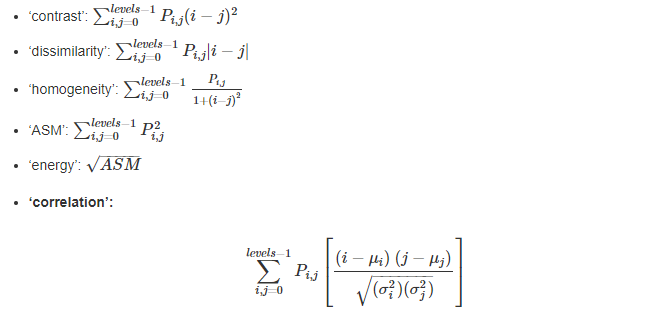
در این بخش از سوال می­خواهیم با کمک روش گفته شده در صورت سوال، تصاویر داده شده را قطعه بندی معنایی کنیم.



طبق پیشنهاد سوال، ابتدا تصویر داده شده را به پچ­هایی با اندازه مشخص تقسیم می­کنیم. دوست داریم اطلاعات موجود در یک پچ، همگن باشد و به طور مشخص بیان­گر یک بافت خاص باشد. در تصویر زیر می­بینید که پچ­های مربوط به بدن گورخر و گندم­زار به خوبی از یک­دیگر قابل تمییز هستند.



در ادامه با استفاده از لینک­های فراهم شده در صورت سوال، به استخراج ویژگی­هایی به کمک GLCM برای تصاویر پرداختیم. با مراجعه به وب­سایت skimage می­بینیم که ویژگی­های آماری متعددی را در اختیار داریم.



همچنین این ویژگی­ها به ازای زوایای ورودی و فواصل پیکسلی مختلف محاسبه می­شوند و به عنوان ویژگی در اختیار ما قرار داده می­شوند. در ادامه این ویژگی­های استخراج شده برای هر پیکسل از تصویر را به عنوان ورودی به الگوریتم K-Means می­دهیم. پیکسل­های مختلف بر خلاف بخش الف تمرین، این­بار نه تنها بر اساس مقدار سطح خاکستری بلکه بر اساس تعداد بسیار زیادی ویژگی محلی خوشه بندی می­شوند. نتیجه این امر این است که پیکسل­هایی با بافت یکسان، در خوشه­های یکسان قرار می­گیرند.

در صفحه بعد، تعدادی از نتایجی که با استفاده از پارامتر­های مختلف به­دست آمده را مشاهده خواهید کرد. لازم به ذکر است که زمان اجرای این الگوریتم بسیار بالا بود و بدون استفاده از پردازش موازی پچ­ها غیرممکن بود!

با استفاده از پردازش موازی پچ­ها، اجرای الگوریتم بر روی تصویر 1000 در 667 پیکسل با اندازه پچ حدود 30 و تعداد سطوح خاکستری حدود 30، به چیزی بین 10 تا 20 ثانیه زمان نیاز داشت. (16 threads @ 4.7GHz). بدون استفاده از موازی سازی و کاهش تعداد سطوح خاکستری، اجرای الگوریتم برای هر تصویر به چندین دقیقه زمان نیاز داشت.

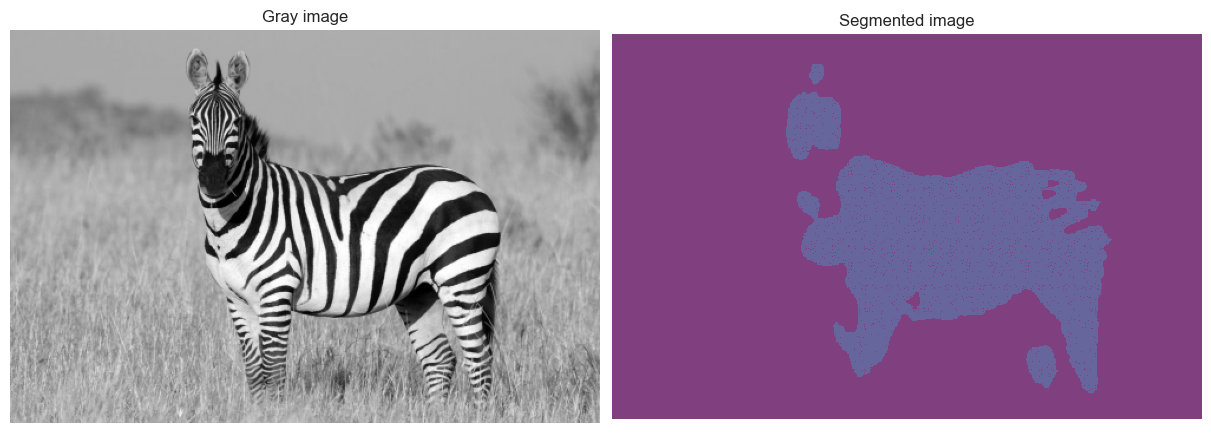
levels=2, patch\_size=5, metrics=["dissimilarity"], distances = [1], angles = [0]



levels=2, patch\_size=5, metrics=["ASM"], distances = [1], angles = [0]



levels=20, patch\_size=21, metrics=["contrast"], distances = [1,5], angles = [0, np.pi/2]



levels=20, patch\_size=21, metrics=["homogeneity"], distances = [1,5], angles = [0, np.pi/2]



levels=20, patch\_size=21, metrics=["dissimilarity"], distances = [1,5], angles = [0, np.pi/2]



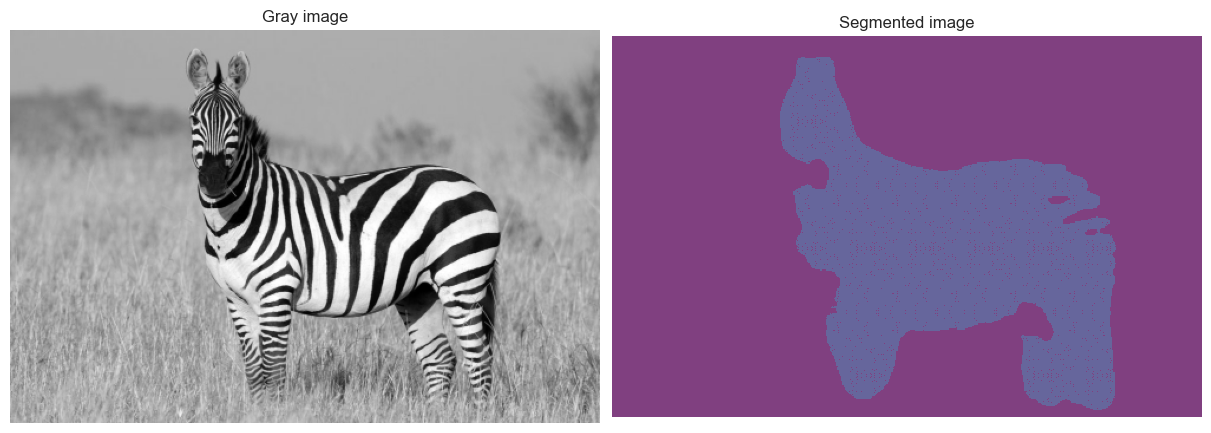
levels=20, patch\_size=41, metrics=["dissimilarity"], distances = [1,5], angles = [0, np.pi/2]



levels=20, patch\_size=35, metrics=["dissimilarity"], distances = [1,5], angles = [0, np.pi/2]



levels=20, patch\_size=31, metrics=["dissimilarity"], distances = [1,5], angles = [0, np.pi/2]



همان­طور که مشاهده می­شود، نتایج ناحیه پس زمینه از گورخر تقریبا جدا شده اند و اختلاف مقدار سطح روشنایی راه­راه­های بدن گورخر باعث نشده­اند در خوشه­های مختلف قرار بگیرند. همچنین یکی از پر اهمیت ترین، پارامتر­ها در این الگوریتم، اندازه پچ است. اندازه پچ بسیار کوتاه باعث می­شود بافت­های بزرگ تشخیص داده نشوند، و اندازه پچ بزرگ باعث می­شود اطلاعات شکل اجسام از بین برود و نواحی بسیار نرم شوند.

levels=50, patch\_size=3, metrics=["dissimilarity"], distances = [1,3,5], angles = [0, np.pi/4, np.pi/2, 3\*np.pi/4]



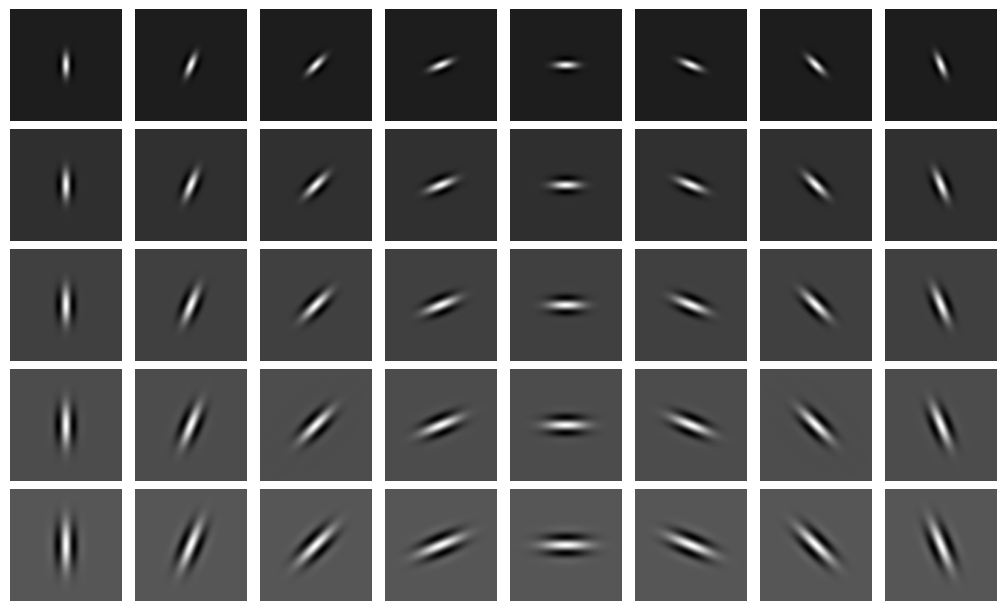
levels=50, patch\_size=21, metrics=["dissimilarity"], distances = [1], angles = [0, np.pi/4, np.pi/2, 3\*np.pi/4]



# ج)

برای روش دوم، نیازی به تقسیم بندی تصاویر به پچ­های مختلف نیست. در این روش، بر روی هر تصویر تعدادی فیلتر اعمال میکنیم. در نتیجه، برای هر **پیکسل** از تصویر، به تعداد **فیلتر­ها** ویژگی خواهیم داشت. این ویژگی­ها می­توانند معانی مختلفی داشته باشند و در مقاصد مختلفی استفاده شوند.

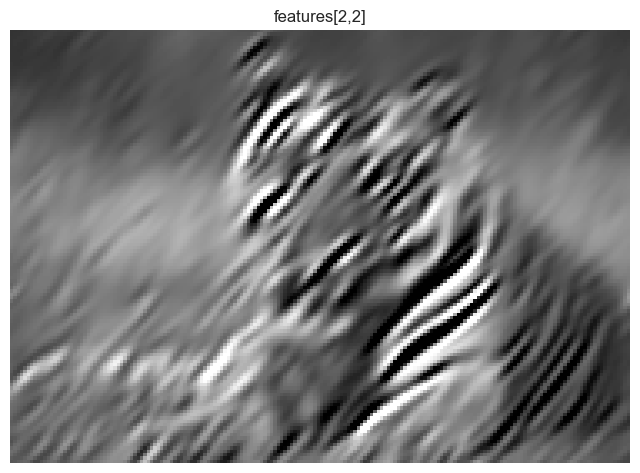
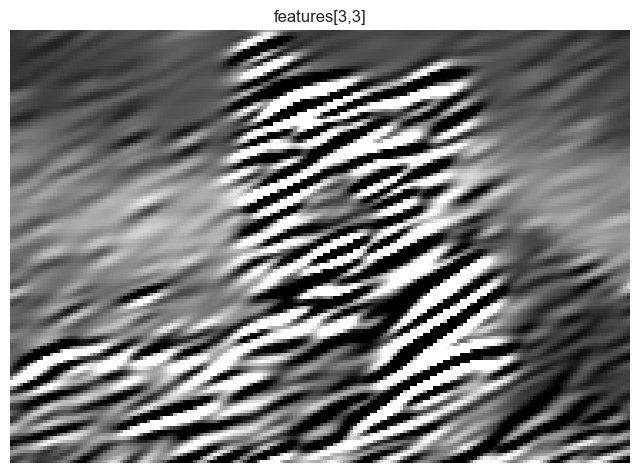
در گام اول، با دادن پارامتر­های متعدد به تابع تولید کننده فیلتر­های گابور، تعداد 40 فیلتر گابور با 5 مقیاس مختلف و 8 زاویه مختلف می­سازیم. این فیلتر­ها به تصاویر اعمال می­شوند و هر کدام ویژگی­های خاصی را از تصویر استخراج می­کنند.



بانک فیلتر­های گابور

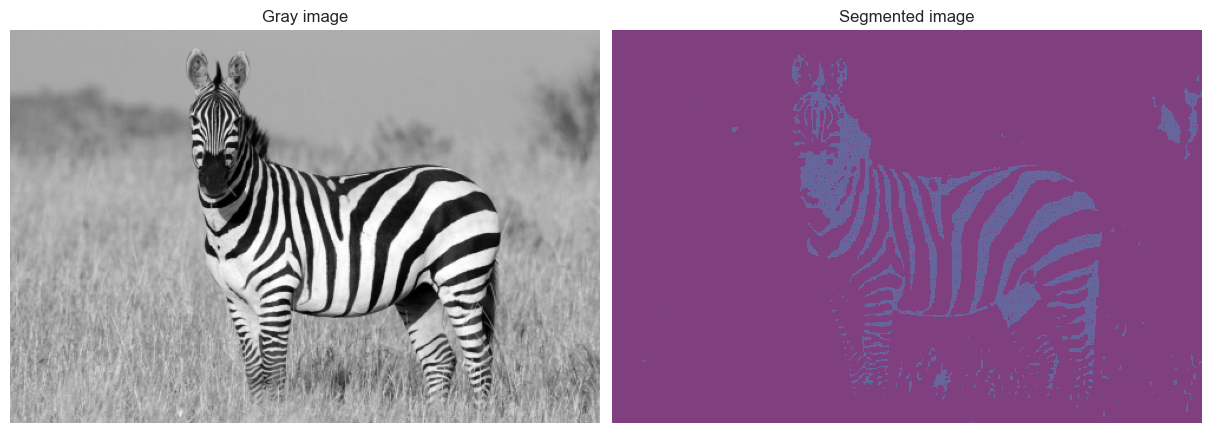
نتیجه اعمال این فیلتر­ها به تصویر را در این صفحه مشاهده می­کنید.



در ادامه این ویژگی­ها را به صورت یک لیست یک بعدی در می­آوریم و برای هر پیکسل به الگوریتم K-Means می­دهیم. این الگوریتم بر اساس تعداد زیادی ویژگی از هر پیکسل، پیکسل­ها را خوشه بندی می­کند. نتایج این خوشه­بندی را در صفحه­های بعد خواهید دید. انتظار می­رود این روش، به دلیل داشتن اطلاعات کمی در مورد همسابه­های هر پیکسل، از روش بخش (**الف**) نتیجه بهتری بدهد اما در مقابل روش بخش (**ب)** که اطلاعات بین پیکسلی زیادی داشت شکست بخورد.

scales=5, angles=8, ksize=35, lambd=5, gamma=0.5, psi=0



scales=5, angles=8, ksize=51, lambd=5, gamma=0.5, psi=0



scales=5, angles=8, ksize=71, lambd=31, gamma=0.5, psi=0



scales=5, angles=8, ksize=35, lambd=5, gamma=0.5, psi=0

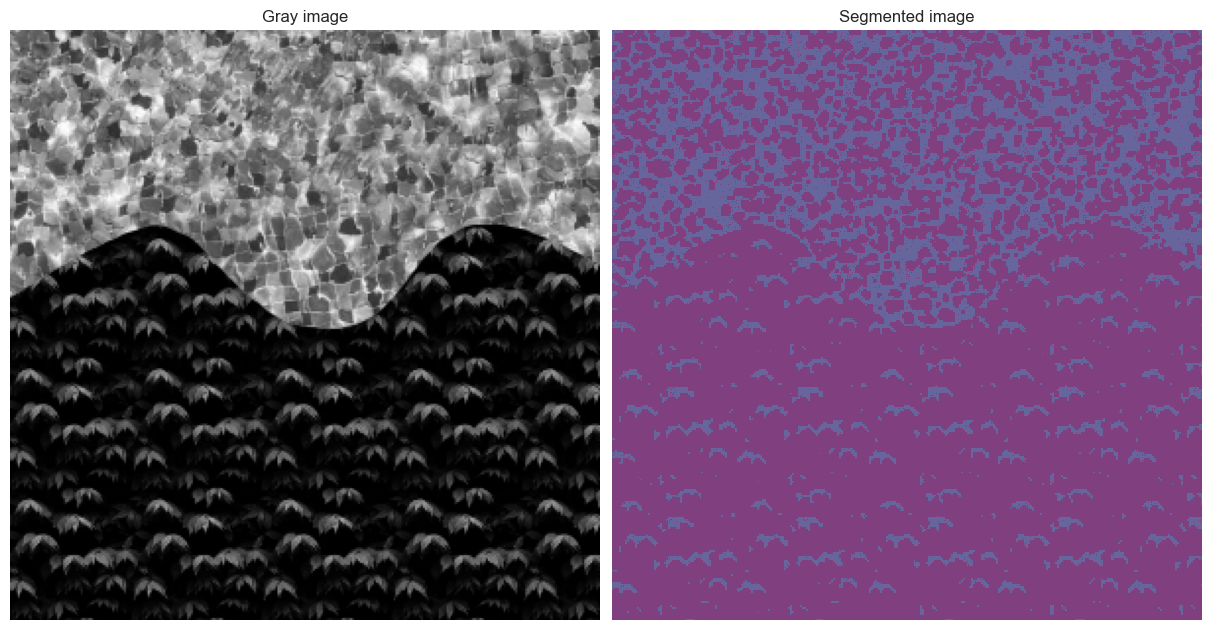


scales=5, angles=8, ksize=5, lambd=1, gamma=0.5, psi=0



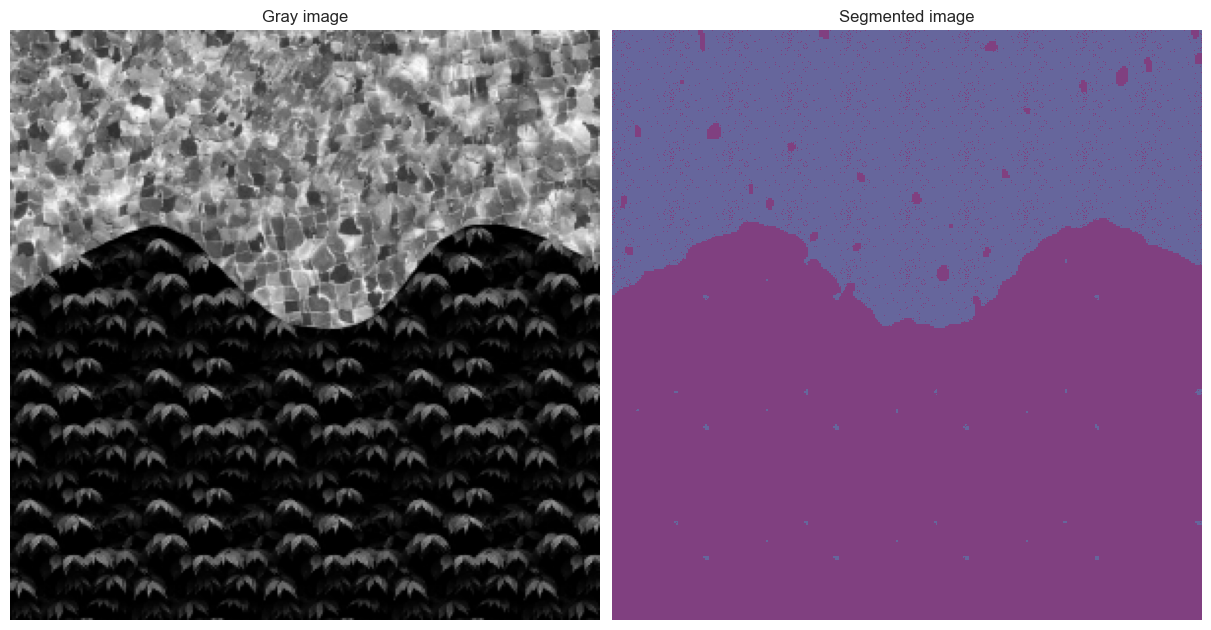
همچنین برای اطمینان از کارکرد الگوریتم، آن را بر روی تصویر ساده زیر اجرا کردیم.

scales=5, angles=8, ksize=71, lambd=5, gamma=0.5, psi=0



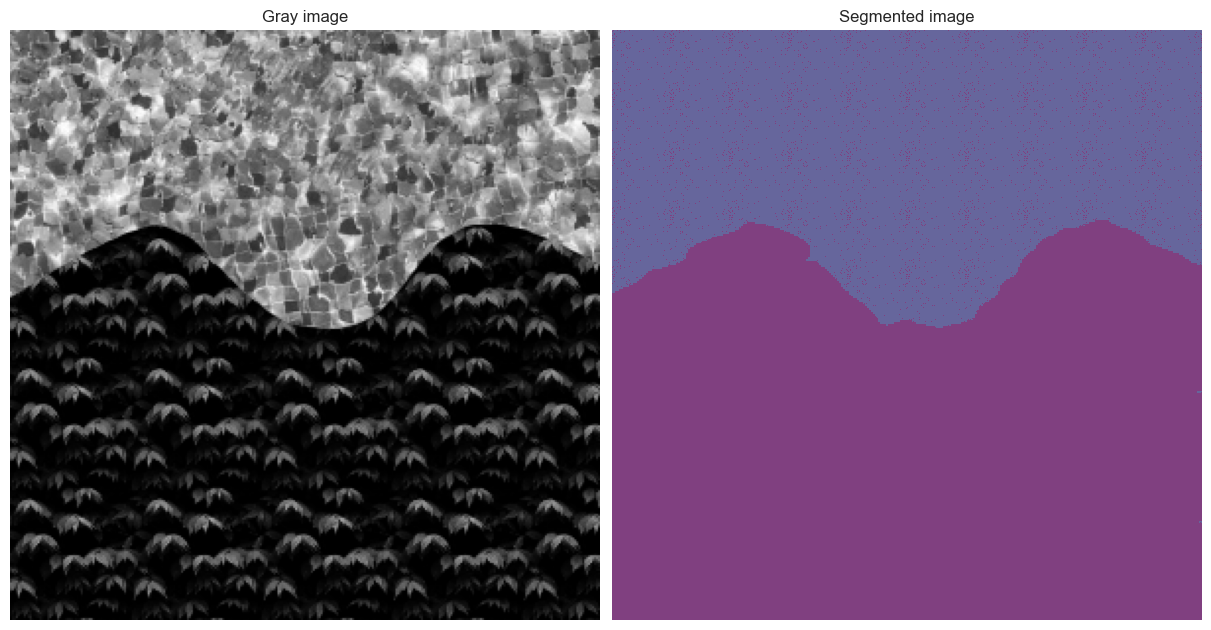
کاهش تعداد فیلترها به یک عدد:

scales=1, angles=1, ksize=41, lambd=30, gamma=0.5, psi=0



اهمیت وجود فیلترهای متعدد(مقایسه کنید با تصویر قبل):

scales=5, angles=8, ksize=41, lambd=30, gamma=0.5, psi=0



# د)

با توجه به تصاویر خروجی که در دو بخش دیدیم می­توان این نتیجه را گرفت که روش GLCM با توجه به استفاده از پچ­هایی از تصاویر، اطلاعات محلی را به خوبی بیشتری حفظ می­کند. و روش Gabor Filter به دلیل اعمال تعدادی فیلتر به کل تصویر، بسیار سریع­تر است. تفاوت سرعت اجرای این دو الگوریتم در شرایط ذکر شده در این پیاده سازی حدودا 20 برابر بود. روش Gabor Filters به تغییرات مقدار سطح خاکستری بسیار حساس بود و به عنوان مثال در تصویر گورخر حتی با اندازه کرنل بالا، خطوط گورخر را در خوشه­های مختلف قرار می­داد.

* لینک­های و منابع مفید:

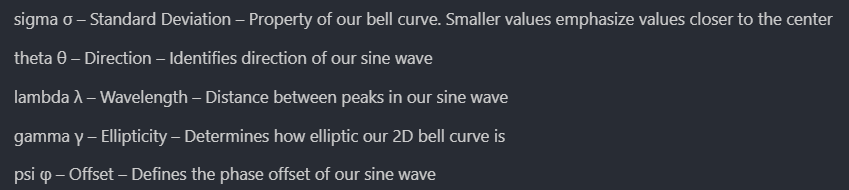
<https://github.com/alfianhid/Feature-Extraction-Gray-Level-Co-occurrence-Matrix-GLCM-with-Python/blob/master/Feature_Extraction_Gray_Level_Co_occurrence_Matrix_(GLCM)_with_Python.ipynb>

<https://scikit-image.org/docs/stable/api/skimage.feature.html#skimage.feature.graycomatrix>

<https://www.robots.ox.ac.uk/~vgg/research/texclass/without.html>

<https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_glcm.html#sphx-glr-auto-examples-features-detection-plot-glcm-py>

* پارامتر­های تابع **cv2**.**getGaborKernel :**



پایان