



تمرین شماره 4

درس پردازش و تحلیل تصاویر پزشکی

استاد درس : دکتر فاطمیزاده

نام : محمد سینا حسن نیا

شماره دانشجویی : 96108515

بخش تئوری

سوال 1

در مورد الگوریتم JPEG و ارتباط آن با sparse modeling تحقیق کنید.

پاسخ:

JPEG یک روش متداول فشرده سازی با اتلاف برای تصاویر دیجیتال، به ویژه برای آن دسته از تصاویر تولید شده توسط عکاسی دیجیتال است. درجه فشرده سازی قابل تنظیم کردن می باشد که این امکان یک trade-off قابل انتخاب را بین اندازه ذخیره سازی و کیفیت تصویر فراهم می کند. JPEG به طور معمول فشرده سازی 10:1 را با کاهش کمی در کیفیت تصویر به دست می آورد. همچنین از زمان معرفی آن در سال 1992، JPEG پرکاربردترین استاندارد فشرده سازی تصویر در جهان و همچنین پرکاربردترین فرمت تصویر دیجیتال، بوده است. اصطلاح "JPEG" مخفف Joint Photographic Experts Group است که این استاندارد را در سال 1992 ایجاد کردند. فشرده سازی تصویر JPEG بر اساس تبدیل کسینوس گسسته (DCT) است. JPEG تا حد زیادی مسئول تکثیر تصاویر دیجیتال و عکس های دیجیتال در سراسر اینترنت و بعداً رسانه های اجتماعی بود. حال که با jpeg به صورت ابتدایی آشنا شدیم می خواهیم به بررسی JPEG compression بپردازیم. همانطور که در بالا نیز گفتیم JPEG از یک فرم فشرده سازی با اتلاف بر اساس تبدیل کسینوس گسسته (DCT) استفاده می کند. این عملیات ریاضی هر فریم/فیلد منبع ویدیویی را از حوزه فضایی (2 بعدی) به حوزه فرکانس تبدیل می کند. یک مدل ادراکی مبتنی بر سیستم روان بصری انسان، اطلاعات با فرکانس بالا، یعنی تغییرات شدید در شدت و رنگها را کنار می گذارد. در حوزه تبدیل، فرآیند کاهش اطلاعات را کوانتیزه کردن می نامند.

به عبارت ساده تر، کوانتیزاسیون روشی است برای کاهش large number scale به مقیاس کوچکتر، و دامنه تبدیل یک نمایش راحت از تصویر است زیرا زیرا ضرایب فرکانس بالا، که کمتر از سایر ضرایب به تصویر کلی کمک می کنند، مشخصاً مقادیر کمی با قابلیت تراکم بالا هستند. سپس ضرایب کوانتیزه شده توالی شده و بدون تلفات در جریان بیت خروجی بسته بندی می شوند. در واقع این عملیات کوانتیزیشن که بر مبنای این فکت که سیستم بصری انسان، اطلاعات با فرکانس بالا، یعنی تغییرات شدید در شدت و رنگها را کنار می گذارد روی مولفه های فرکانس بالا انجام می شود نمایان کننده sparse modelling می باشد که در واقع با عملیات بالا

روی فرکانس های بالا (جاهایی که چشم انسان تشخیص نمی دهد) باعث sparse تر شدن می شوند و این بازگو کننده sparse modelling می باشد.

سوال 2

مراحل الگوریتم K-SVD مورد استفاده در نویز زدایی از تصاویر را شرح دهد. ارتباط آن با بازنمایی تنک تصاویر توضیح دهد. اگر به جای نویز جمع شونده گوسی از نویز دیگری برای مدل سازی نویز استفاده کنیم چه تغییراتی در روابط مدل سازی تنک لازم است ایجاد شود ؟

پاسخ: الگوریتمی مطابق زیر را در نظر می گیریم. در این الگوریتم تصویر نویزی \mathbf{Y} که با نویز جمع شونده گوسی با انحراف معیار σ نویزی شده است را دینویز می کنیم. پارامتر های الگوریتم به شرح زیر می باشد:

N: block size

K: dictionary size

J: number of training iteration

λ : lagrange multiplier

C: noise gain

هدف الگوریتم مینیمم کردن تابع زیر می باشد:

$$\min_{\mathbf{X}, \mathbf{D}, \mathbf{A}} \left\{ \lambda \|\mathbf{Y} - \mathbf{X}\| + \sum_{ij} \mu_{ij} \|\alpha_{ij}\|_0 + \sum_{ij} \|\mathbf{D}\alpha_{ij} - R_{ij}X\|_2^2 \right\}$$

- برای این کار ابتدا در مرحله $\mathbf{X}=\mathbf{Y}$ initialization را قرار می دهیم و \mathbf{D} را یک دیکسنری overcomplete و DCT انتخاب می کنیم.
- ۱ بار تکرار می کنیم:

○ با استفاده از هر نوع الگوریتم pursuit با تخمین زدن جواب معادله Sparse coding stage :

زیر بردارهای نمایش j برای هر α_{ij} را patch می محاسبه می کنیم:

$$\forall_{ij} \min_{\alpha_{ij}} \|\alpha_{ij}\|_0 \quad \text{s.t.} \quad \|R_{ij}\mathbf{X} - \mathbf{D}\alpha_{ij}\|_2^2 \leq (C\sigma)^2.$$

- برای هر ستون $k=1, 2, \dots, k$ در \mathbf{D} آپدیت به صورت زیر است:
- Patch هایی که از این اتم استفاده می کنند را پیدا می کنیم :

$$\omega_l = \{(i, j) | \alpha_{ij}(l) \neq 0\}.$$

▪ برای هر \mathbf{x} برای هر $(i, j) \in \omega_l$ index خطای نمایش آن را محاسبه می کنیم:

$$\mathbf{e}_{ij}^l = R_{ij}\mathbf{X}_{ij} - \sum_{m \neq l} \mathbf{d}_m \alpha_{ij}(m).$$

▪ را به صورتی تعیین می کنیم که ستون های آن به صورت زیر باشند

$$\{\mathbf{e}_{ij}^l\}_{(i,j) \in \omega_l}$$

▪ با استفاده از SVD-decomposition داریم :
حال باید :

Choose the updated dictionary \tilde{d}_l to be the first column of U
update the coefficient values $\{\alpha_{ij}(l)\}_{(i,j) \in \omega_l}$ to be the entries of V multiplied by $\Delta(1,1)$

• حال خواهیم داشت :

$$X = \left(\lambda \mathbf{I} + \sum_{ij} R_{ij}^T R_{ij} \right)^{-1} \left(\lambda \mathbf{Y} + \sum_{ij} R_{ij}^T \mathbf{D} \alpha_{ij} \right)$$

الگوریتم بالا را می توان برای دینویز کردن با شرایط گفته شده استفاده کرد. در واقع همانطور که در بالا می بینیم در این الگوریتم در هر ایتریشن در تلاشیم تا برای هر patch نمایشی sparse به دست آوریم. \mathbf{X} به دست آمده که یک عبارت بزرگ است در واقع دارد می گوید میانگیری از patch های دینویز شده باید انجام شود با ریلکیشنی که با میانگین گرفتن با تصویر نویزی ارجینال به دست می آید.(نوعی میانگین گیری بین تصاویر نویزی و تصاویر بازسازی شده از تصاویر تنک انجام می شود. همچنین به عنوان نکته تکمیلی ماتریس معکوس در عبارت فوق یک ماتریس قطری است، و بنابراین، محاسبه بالا را می توان بر روی پیکسل به پیکسل انجام داد. اگر به جای نویز جمع شونده گوسی از نویز دیگری برای مدل سازی نویز استفاده کنیم باید sparse coding stage را چهار تغییراتی کنیم. این تغییرات برای مثال می تواند به صورت استفاده از نرم های دیگری نظیر نرم یک برای محاسبه باشد

سوال 3

فرض کنید می خواهیم بازنمایی تنک از سیگنال های 64 بعدی داشته باشیم. بدین منظور دیکشنری ای با 100 اتم درنظر میگیریم، اگر بخواهیم که تعداد اتم های مورد استفاده برابر با 3 باشد چه تعداد زیر فضا برای بازنمایی تنک میتوان متصور شد؟

پاسخ: در اینجا در فضایی 64 بعدی به تعداد 100 اتم پایه در نظر گرفته ایم. با فرض این که اتم های دیکشنری متعامد باشند خواهیم داشت :

$$\binom{100}{3} = \frac{100 * 99 * 98}{6} = 161700$$

بنابراین 161700 زیرفضا برای بازنمایی تنک می توان متصور شد.

سوال 4

با در نظر گرفتن یک دیکشنری 2 در 4 با اتمهای (1 و 2) و (0 و 1) و (1 و 0) یک بازنمایی تنک برای بردار $x = [2, 2]^T$ پیدا کنید.

پاسخ: برای این سوال داریم :

$$\begin{cases} x = Dy \\ D = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \Rightarrow y = [0 \ 0 \ 2 \ 0]^T \\ x = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \end{cases}$$

سوال 5

برای relax کردن قید های مسئله sparse representation الگوریتم های Basis pursuit و matching pursuit مورد استفاده قرار میگیرند. در مورد آنها به اختصار توضیح داده و آنها را با هم مقایسه کنید.

پاسخ: ابتدا الگوریتم basic pursuit و سپس matching pursuit را توضیح می دهیم:

یک مسئله بهینه سازی ریاضی به فرم زیر می باشد:

$$\min_x \|x\|_1 \quad \text{subject to} \quad y = Ax,$$

که در آن x یک بردار $N \times 1$ (سیگنال)، y یک بردار $M \times 1$ از مشاهدات (اندازه گیری ها)، A یک ماتریس تبدیل $M \times N$ (معمولًا ماتریس اندازه گیری) و $\|y\|_2$ است. معمولاً در مواردی استفاده می شود که یک سیستم معادلات خطی $y = Ax$ وجود دارد که باید دقیقاً برآورده شود و sparse undetermined در نظر گرفتن نرم L1 باشد. هنگامی که مطلوب است که برابری دقیق $Ax = y$ در ازای یک x تنک تر مبادله شود، حذف نویز مبتنی بر basis pursuit ترجیح داده می شود.

matching pursuit یک روش sparse approximation است که بهترین matching pursuit دیتای چندبعدی را در گستره یک دیکشنری redundant projection over-complete یا به عبارتی پیدا می کند. ایده اصلی این است که تقریباً یک سیگنال از فضای هیلبرت H را به صورت جمع وزنی از اتمها که از D گرفته شده است، نشان دهد. تقریبی با N اتم به صورت زیر نشان داده میشود:

$$f(t) \approx \hat{f}_N(t) := \sum_{n=1}^N a_n g_{\gamma_n}(t)$$

که γn ستون gn از ماتریس D و a_n یکفاکتور اسکالر ضریب وزن دهنی به اتم gn است. البته به طور معمول هر اتم D در این جمع استفاده نمی شود. در عوض الگوریتم matching pursuit اتم ها را یکی یکی انتخاب می کند تا به صورت greedy خطای تقریب را کاهش دهد. این امر با یافتن اتمی که بیشترین ضرب داخلی را (با فرض این که اتم ها نرمالیزه شد ها ند) داشته باشد انجام می شود و بعد تقریبی را که فقط در آن از همین یک اتم استفاده شده باشد از سیگنال اصلی کم میکنیم و این فرآیند را تا زمانی که سیگنال به طور رضایت بخشی تجزیه شود که یعنی نرم residual خیلی کوچک شود که نرم residual به صورت زیر محاسبه می شود:

$$R_{N+1} = f - \hat{f}_N.$$

اگر Rn به سرعت به صفر همگرا شود، آنگاه فقط چند اتم لازم است تا یک تقریب خوب برای f بدست آوریم. چنین نمایش های تنکی برای کدگذاری و فشرده سازی سیگنال مطلوب هستند. به طور دقیق تر، مشکل sparsity که matching pursuit برای حل آن در نظر گرفته شده است. به عبارتی مسئله ای که الگوریتم matching pursuit تمايل دارد آن را به صورت تقریبی حل کن به صورت زیر است:

$$\min_x \|f - Dx\|_2^2 \text{ subject to } \|x\|_0 \leq N,$$

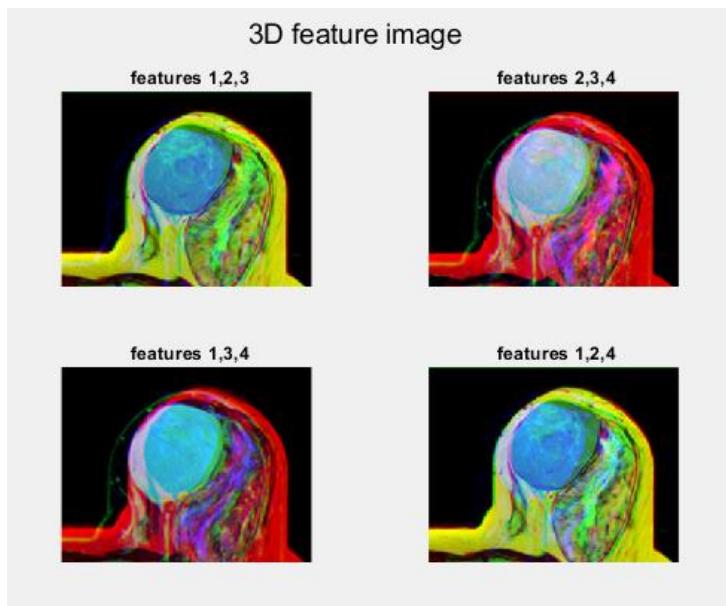
در basis pursuit 1 از نرم استفاده شده است که این به دلیل این است که کار کردن با نرم صفر به دلیل مشتق ناپذیر بودن آن سخت است. بنابراین از نرم L1 استفاده شده است این در حالی است که در الگوریتم matching pursuit سعس داریم نرم صفر را نرم کنیم و آن را حل کنیم.

بخش شبیه‌سازی

سوال 1

پاسخ:

الف) طبق خواسته سوال به صورت سه بعدی برحسب ویژگی‌ها رسم می‌کنیم:



به صورت کلی با این بررسی می‌توان دید که ما یک پس زمینه داریم که در همه یکسان است و یقیناً یک کلاستر باید به ان نسبت داد. همچنین ما تومور را داریم که به صورت مشخص در هر ۴ تصویر مشخص است. همچنین با توجه به تصاویر بالا می‌توان ۲ کلاستر دیگر نیز اختصاص داد. بنابراین مطابق آن چه در بالا ذکر شد ما ۴ کلاستر را در نظر می‌گیریم.

ب) حال میخواهیمتابع FCM را ببروی تصویر اعمال کنیم. در این حالت ما تابع هزینه زیر را در حال مینیمیم کردن هستیم.

$$J = \frac{1}{N} \sum_{j=1}^K \sum_{i=1}^N u_{ij}^2 \|x_i - \omega_j\|_2^2, \quad s.t. \left(\sum_{j=1}^K u_{ij} - 1 = 0 \right), \forall i = 1, 2, \dots, N$$

در آن مقدار u مقدار ضریب نسبیت یا احتمالی است که ما به هر کلاستر داریم نسبت میدهیم در نهایت با حل معادله به صورت زیر باید عمل کنیم:

$$u_{ij} = \frac{\frac{1}{\|x_i - \omega_j\|_2^2}}{\sum_{l=1}^K \frac{1}{\|x_i - \omega_l\|_2^2}} (*)$$

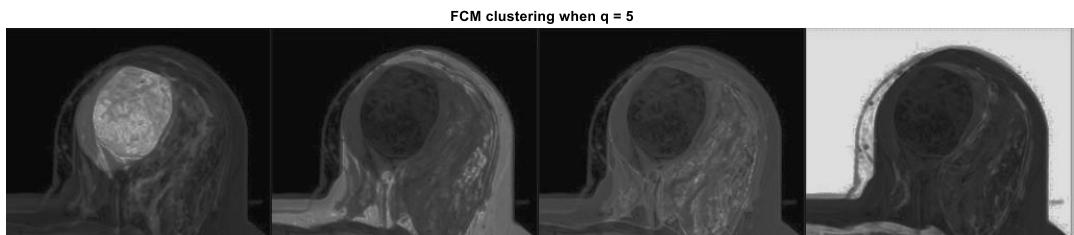
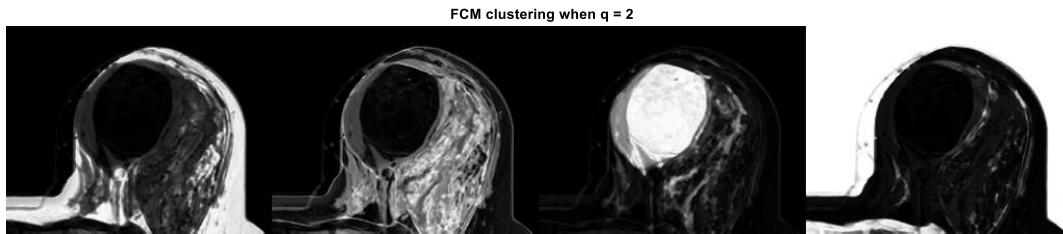
$$\omega_j = \frac{\sum_{i=1}^N u_{ij}^2 x_i}{\sum_{i=1}^N u_{ij}^2} (**)$$

که در این حالت ما به صورت تکراری معادلات را عددی حل میکنیم مقدار u را ابتدا بروز میکنیم و براساس ان مقدار W و این روند تکرار میابد. در این بخش ابتدا ازتابع متلب برای fcm استفاده میکنیم در این حالت ما ورودی ضریب فازی q را داریم که) همانطور که نامش اشاره میکند(به اینصورت است که تعیین میکند تا چه حد قدرت کلاستر ها و احتمال انها مشخص باشد یعنی در نزدیکی یک به هارد کلاسترینگ میرسیم و هرچه بیشتر میشود نرم و نرم تر میشود و احتمال وجود یک پیکسل در دو کلاستر میتواند نزدیک به هم باشد. به صورت عادی در متلب تا ۱۱۱ ایتریشن کد اجرا میشود ولی اگر تغییرات از حد اپسیلون داخلی تعریف شده کمتر باشد اجرا متوقف میشود و مراکز و احتمالات را در خروجی میدهد ما برای این کار ابتدا مجبور هستیم تصاویر ویژگی را به بردار تبدیل کنیم و سپس آن را به تابع دهیم خروجی احتمالها را دریافت کرده و رسم میکنیم به صورت

زیر میشود:

FCM clustering when $q = 1.1$





در هر کدام از حالات می توان مشاهده نمود که بافت مورد نظر به دست آمده است. همانطور که مشاهده می شود به ازای ضریب ۱.۱ ناحیه بندی سخت انجام می شود. در این حالت خوش بندی به clustering عادی نزدیک می شود (در واقع میتوان روش k-means را روش ناحیه بندی در نظر بگیریم که خروجی برابر با fcm در حالت $q=1$ دارد). در این حالت شاهد یک حالت ۰ و ۱ ای خواهیم بود. (به عبارت دیگر احتمال تعلق به هر کلاس یا بسیار نزدیک صفر و یا بسیار نزدیک ۱ است. هنگامی که ضریب مقدار فازی را بزرگ می کنیم می بینیم که احتمال تعلق دیگر بسیار نزدیک صفر یا یک نیستند و این به آن معناست که یک پیکسل می تواند همزمان به چند خوش تعلق داشته باشد. بنابراین همانطور که مشاهده می شود در این حالت ها از حالت صفر و یکی فاصله می گیریم و رنگ های خاکستری در شکل ها مشاهده می کنیم. بنابراین همانطور که گفته شد به ازای $q=2,5$ شاهد ناحیه بندی نرم هستیم.

ج) در واقع میتوان روش k-means را روش ناحیه بندی در نظر بگیریم که خروجی برابر با fcm در حالت $q=1$ دارد. به ازای تعداد کلاس های مختلف داریم :

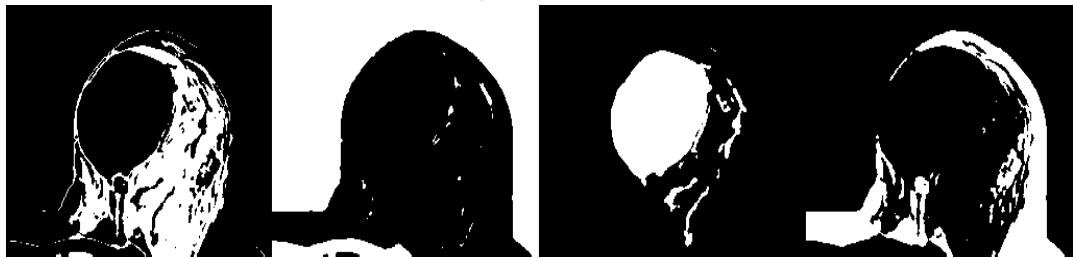
k means output for number of clusters = 2



k means output for number of clusters = 3



k means output for number of clusters = 4



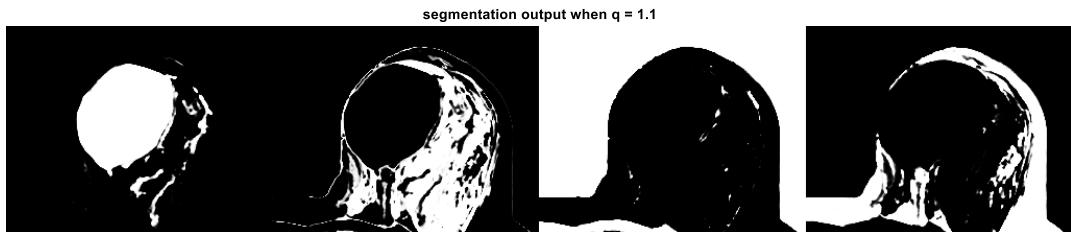
k means output for number of clusters = 5



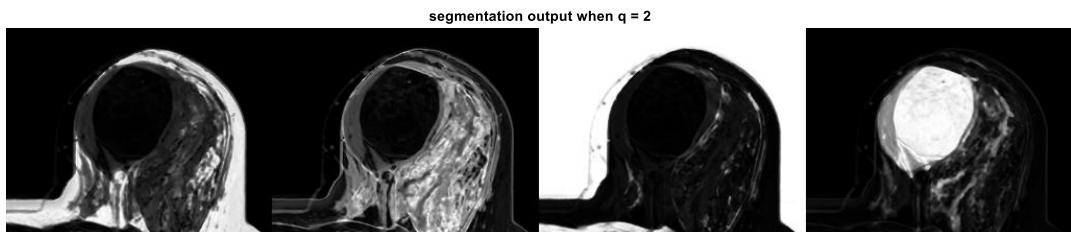
از شکل‌های بالا در می‌باشیم به ازای 2 خوشة تنها یک گراند جدا شده است. همچنین به ازای تعداد کلاستر 3 نیز دو تا از بافت‌ها به عنوان یک خوشه شناسایی شده‌اند. به ازای 5 کلاستر نیز در یکی از بافت‌ها شاهد این

هستیم که انگار به دو بخش تقسیم شده است (سومی و اولی از سمت راست). بنابراین بنابر آنچه در بالا گفتیم تعداد کلاستر 4 می تواند مناسب باشد.

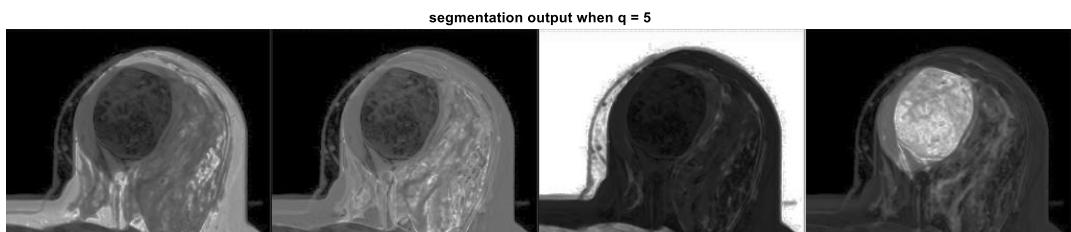
برای بخش بعدی این سوال با استفاده ازتابع جدید داریم:



code ended at iteration = 10



code ended at iteration = 17



code ended at iteration = 30

همان گونه که مشاهده می شود نتایج با نتایج قبلی تقریباً یکسان است اما تفاوت چشمگیری در تعداد ایتریشن ها شاهد می باشیم به طوری که در روش قبل که از حالت تصادفی استفاده می شود به ترتیب شاهد 58,35,30

ایتریشن می باشیم در حالی که در روش جدید شاهد 30,17,10 ایتریشن می باشیم. بنابراین تعداد ایتریشن ها با انتخاب حالت اولیه مناسب تر کاهش می یابد.

(۵) برای این قسمت داریم:



(۶) می دانیم که *partial volume* زمانی رخ میدهد که به خاطر رزولوشن محدود ما پیکسلی دارای اطلاعات بخش متفاوت باشد (به خاطر نزدیک بودن ترکیب خم ها) برای همین میتوانیم بگوییم این نقطه ترکیبی از ویژگی های چند بخش را دارد برای همین وقتی با روش *fcm* میاییم و *probability map* ها را تولید میکنیم در این حالات و نقاط احتمال یک نقطه در دو بیشترین احتمال به هم نزدیک است (میتواند در همه باشد ولی اگر در دو تا بیشترین نزدیک باشد یعنی به قدر کافی تفکیک انجام نشده است) ما کاری که انجام میدهیم این است که احتمال ها را بدست میاوریم و ببروی نسبت آن دو ترشلد میگذاریم. هر چه ترشولد را بزرگ تر انتخاب کنیم درجه سخت گیری کمتر می شود. همچنین باید ضریب خوش بندی را به قدری بزرگ اختیار کنیم تا به گونه ای نشود که همه پیکسل ها را متعلق به یک دسته بدانیم و از طرفی نباید آن قدر کوچک انتخاب کنیم که ناحیه هایی که به یک خوش متعلق هستند به صورت حجم جزئی طیقه بندی شوند. با توجه به موارد بالا مقدار $q=5$ و ترشولد را برابر 1.07 در نظر می گیریم. داریم :

PVE map when $q = 5$

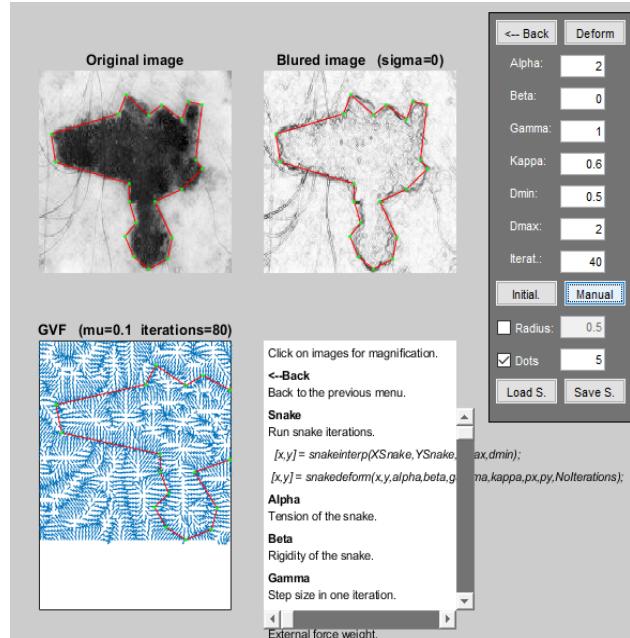
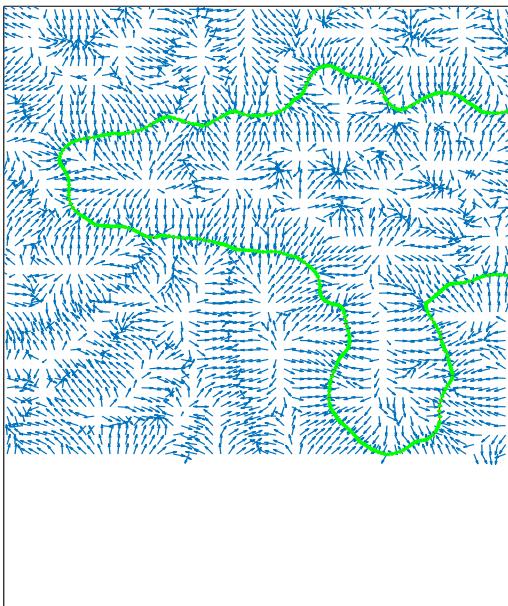


سؤال 2

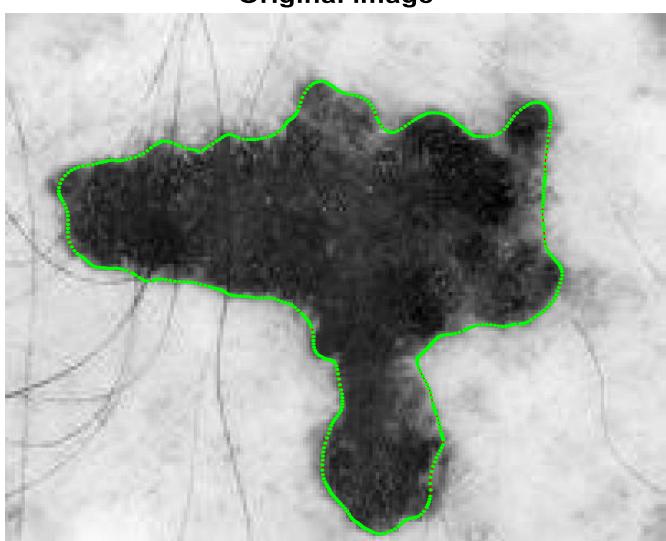
:Melanoma

GVF

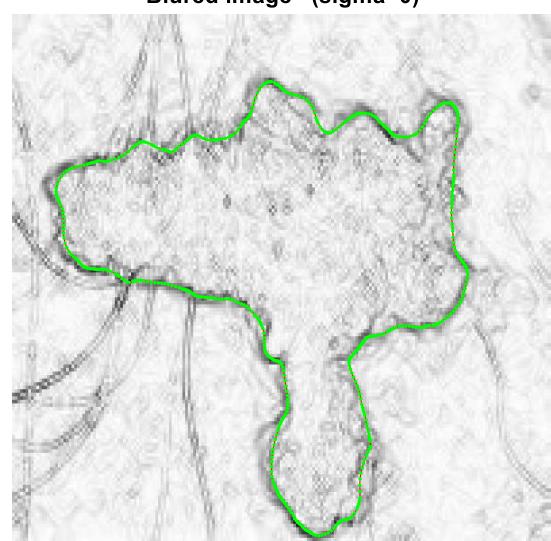
GVF ($\mu=0.1$ iterations=80)



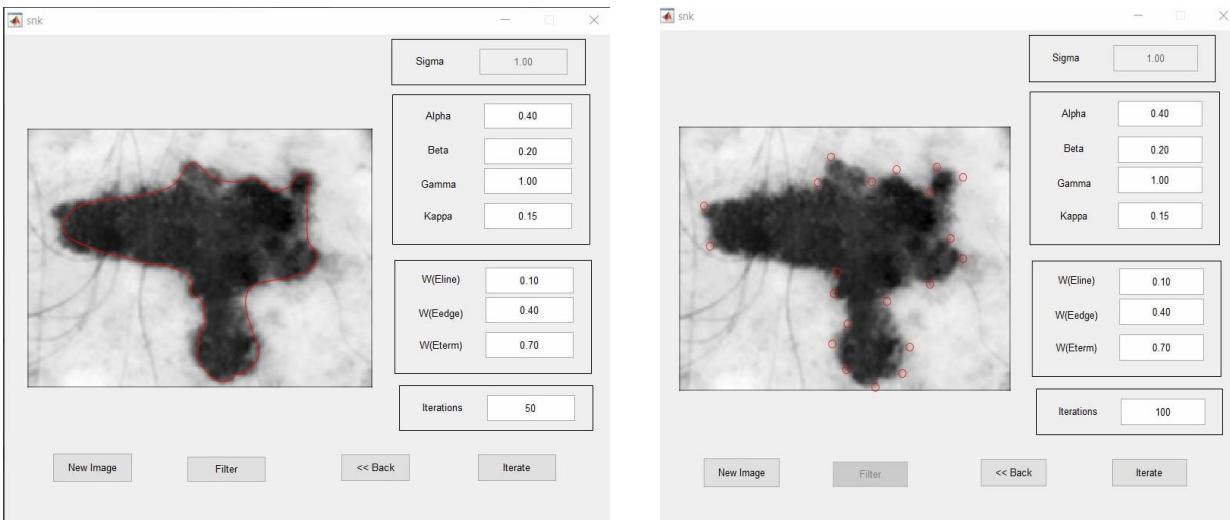
Original image



Blurred image ($\sigma=0$)



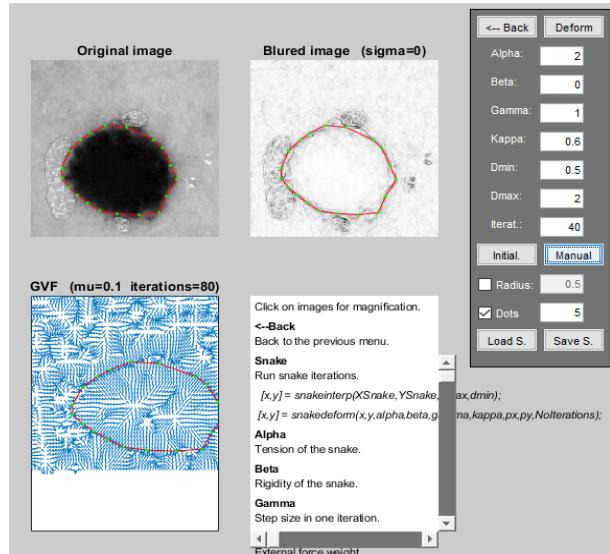
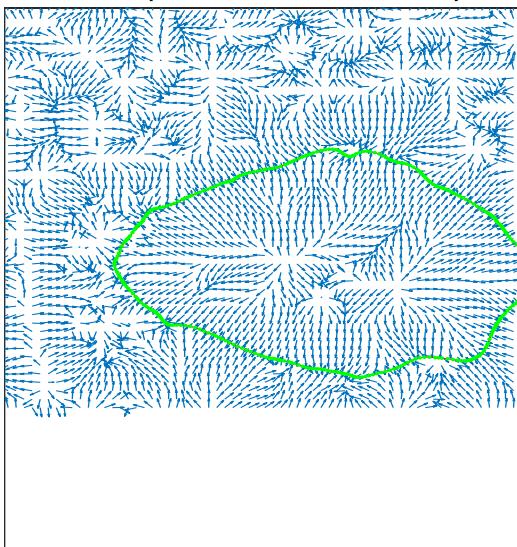
Basic Snake

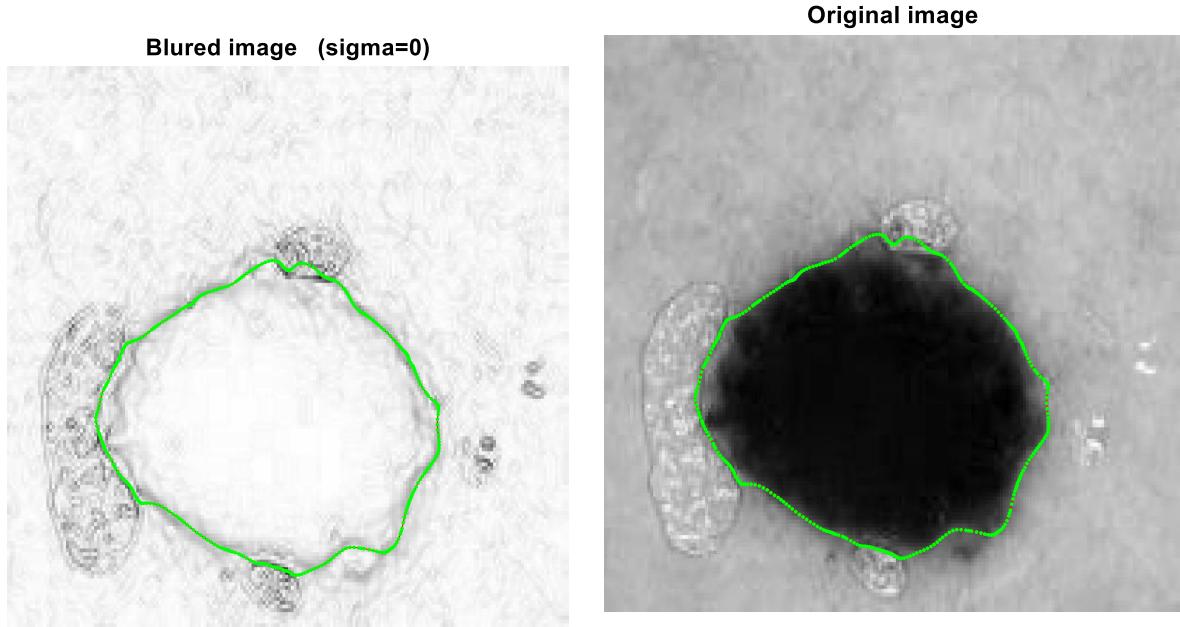


:Neveus

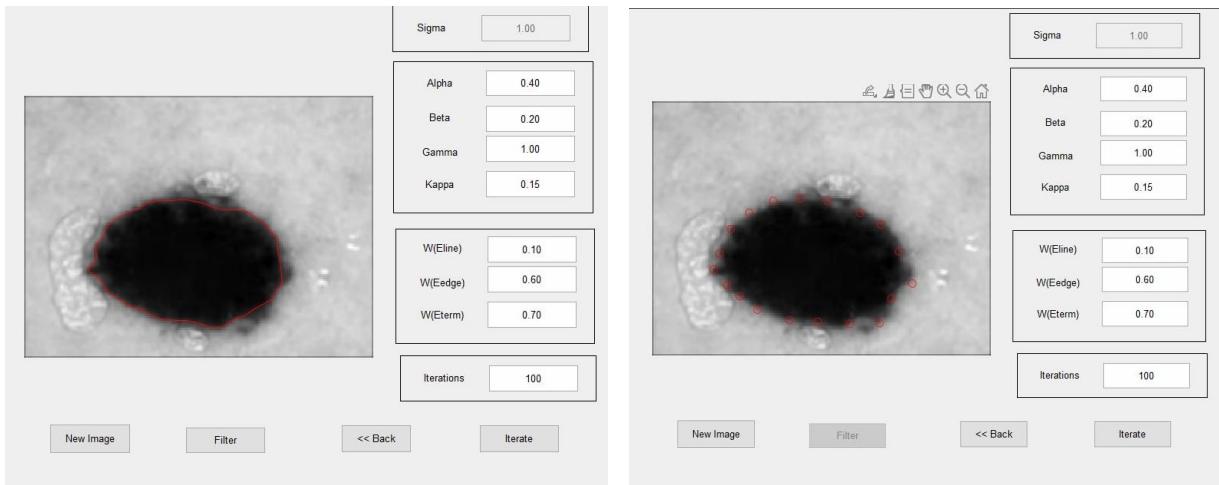
:GVF

GVF (mu=0.1 iterations=80)





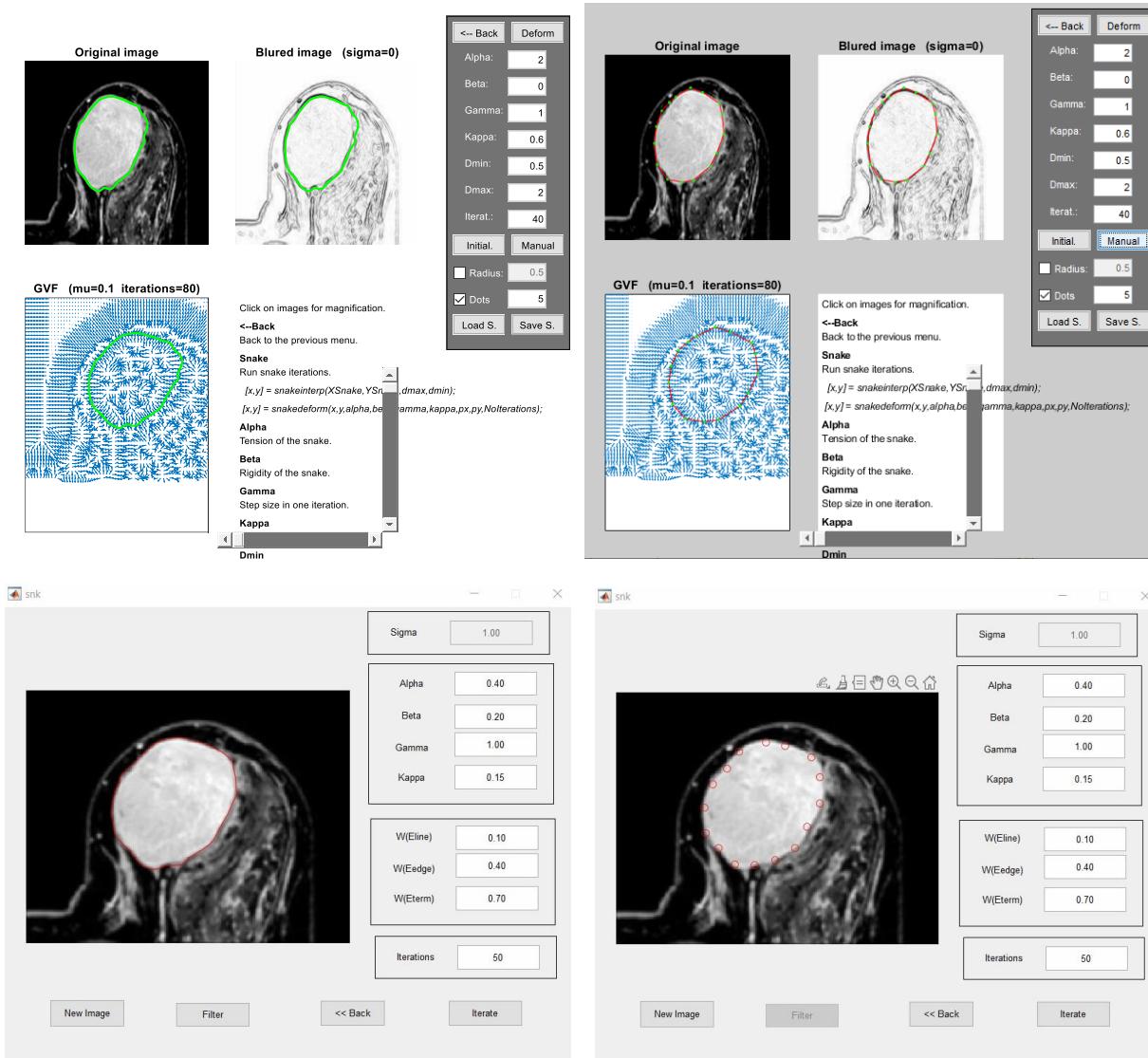
:Basic Snake



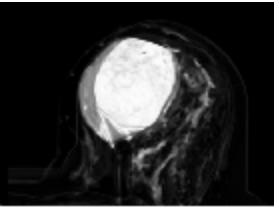
همانطور که در بالا می بینیم عملکرد برای nevus که یک شکل ساده است در هر دو روش GVF و basic snake تقریباً یکسان است. حال به سراغ melanoma می روم که شکل پیچیده تری است. در این شکل کاملاً مشخص است که GVF عملکرد بهتری نسبت به basic snake داشته است . همانطور که دیده می شود GVF به خوبی در نواحی ای که انحنا داشته است انحنا را دنبال کرده است در حالی که basic snake در نواحی ای که انحنا زیاد است دچار مشکل شده است و آن را دنبال نکرده است. به صورت کلی مشکل Basic snake این است که فرورفتگی ها به شدت روی ان اثر دارند چون در این میان گرادیان عکس که مرز ها را مشخص میکند

کوچک است در ناحیه فرورفتگی ها و به خوبی نمیتواند نسبت به بقیه نیروها مجبور کند که فرم درست را بگیرید.
 البته می توان با افزایش تعداد نقاط اولیه این مشکلات را حل کرد.

(ب)



همانطور که دیده می شود دو روش بالا عملکرد و دقت بهتری برای شناسایی تومور دارند. اگر بخواهیم دقیق تر بگوییم برای مثال در روش FCM برخی نواحی که تومور نیستند را تومور تشخیص داده است در حالی که همچنین اتفاقی برای GVF و basic snake رخ نداده است.



FCM

FCM به صورت کلی در یک مورد نسبت به GVF و basic snake برتری دارد و آن هم این است که GVF نمی توانند نواحی جدا از هم را در حالی که متعلق به یک کلاس هستند را جدا کنند در حالی که FCM برای چنین کاربردهایی نیز کاربرد دارد و با آن می شود این کار را کرد. (در واقع عامل اصلی این است که در ارزشی درونی طول اثر مهمی دارد و وقتی فاصله زیاد باشد اثر بیشتر شدن طول خیلی غالب است و این توانایی را میگیرد پس این کار را نه در روش GVF و نه basic snake نمیتوان مشاهده کرد)

سوال 3

الف) تابع `chanvese.m` نوشته شده است. این تابع دارای 2 ورودی است که یکی تصویر و دیگری مربوط به تعیین نحوه حالت کاری تابع می باشد. ورودی 2 می تواند دو ورودی "boundary" و "center" را بگیرد. با استفاده از تابع نوشته شده برای دو تصویر گفته شده داریم:



با استفاده از مود "boundary"



با استفاده از مود "center"



با استفاده از مود "boundary"



با استفاده از مود "center"

ب) برای این قسمت نتیجه مانند زیر می باشد.



با استفاده از مود "boundary"



با استفاده از مود "center"

با توجه به تصاویر بالا و مطابق با قسمت های قبلی می دانیم همانطور که در بالا دیده می شود این الگوریتم مقداری تشخیص نادرست داشته است . به طور کلی و با توجه به قسمت های قبلی از نظر دقیق تریب الگوریتم FCM و chanvese (به شرط تعداد نقطه مناسب)، Basic sanke و GVF خودکرد خوبی دارند اما از نظر خودکار بودن به ترتیب FCM، chanvese،Basic sanke و GVF مشخص است در واقع یک trade-off بین میزان خودکار بودن یک الگوریتم و میزان دقیق آن وجود دارد. به این صورت که هر چه دقیق افزایش پیدا کند از میزان خودکار بودن کاسته می شود و بالعکس.

برای روشی که بدون دخالت کاربر و به صورت خودکار بتواند این کار را انجام دهد ابتدا با توجه به این که می‌دانیم ناحیه مورد نظر روشن تر از بقیه می‌باشد با استفاده از تابع `multithresh` یک ترشولد گذاری می‌کنیم و ماسک مورد نظر را با توجه به روشن تر بودن ناحیه مورد نظر به صورت `image > thresh(4)` تعیین می‌کنیم . حال که ماسک مورد نظر را به دست آورده‌یماند قبل عمل می‌کنیم. به عنوان نتیجه داریم:

Visualizat‌ion of figures(automatic mode)



در اینجا با توجه به این که خوش‌بندی را به صورت خودکار انجام داده ایم همانطور که می‌بینیم کیفیت پایین تری داریم که این مورد انتظار است و مطابق با تئوری است. در واقع به صورت کلی هر چه اتوماتیک تر شویم از کیفیت خوش‌بندی کاسته می‌شود.