علی آراسته – ۴۰۰۲۰۶۱۵۴

پردازش و تحلیل تصاویر پزشکی – تمرین سری چهارم

بخش تئوری

سوال ۱: در الگوریتم JPEG، برای ذخیره‌سازی داده، ابتدا فضای رنگ تصویر به فضای رنگ $Y'C_bC_r$ منتقل می‌شود. در این فضای رنگ، Y' معرف روشنایی پیکسل، و C_b و C_r معرف رنگ‌های آبی و قرمز پیکسل هستند. از آنجایی که چشم انسان به روشنایی بیشتر از رنگ حساس است، می‌توان کانال‌های مربوط به رنگ، یعنی C_b و C_r را با ضریب ۲ در هر جهت downsample کرد. سپس هر یک از کانال‌های روشنایی و رنگ، به طور مجزا وارد الگوریتم فشرده‌سازی داده می‌شوند. نحوه عملکرد این الگوریتم بدین صورت است که ابتدا ماتریس ورودی را به بلوک‌های ۸*۸ تقسیم می‌کند و سپس مراحل زیر را برای هر بلوک به صورت جداگانه اجرا می‌نماید:

۱) بازه تعریف مقادیر بلوک به بازه‌ای با طول مشابه به مرکزیت صفر منتقل می‌شود.

۲) تبدیل کسینوسی گسسته دو بعدی نوع دوم بر روی بلوک مورد نظر اعمال می‌شود.

۳) بر حسب نرخ فشرده‌سازی مورد نظر، ماتریس کوانتیزاسیون مناسب انتخاب می‌شود

۴) برای هر پیکسل مقدار حاصل از تبدیل کسینوسی به مقدار متناظر در ماتریس کوانیزاسیون تقسیم می‌شود و خروجی با استفاده از عملیات گرد کردن، به یک عدد صحیح تبدیل می‌شود.

۵) رشته حاصل از حرکت زیگ زاگ روی ماتریس حاصل از قسمت قبل، با استفاده کد هافمن مناسب، کدگذاری می‌شود.

ارتباط این روش با sparse modeling در استفاده از تبدیل کسینوسی نهفته است. زیرا با استفاده از این عملیات می‌توان بلوک متناظر را بر حسب پایه‌های این تبدیل، به صورت تنک نمایش داد.

سوال ۲: الگوریتم K-SVD، در واقع تعمیمی از الگوریتم K-means است. در الگوریتم K-means هدف یافتن بهترین ماتریس کد برای تعریف نمونه‌های داده به وسیله نزدیک‌ترین همسایه آن است. این کار با استفاده از مسئله بهینه‌سازی زیر صورت می‌گیرد:

$$\min_{D,X}\{\|Y - DX\|_F^2\} \qquad \text{subject to} \quad \forall i, x_i = e_j \text{ for some } j$$

در نتیجه این امر، الگوریتم K-means، برای بازسازی هر ستون ماتریس Y، یک ستون ماتریس D را تکرار می‌کند. برای نرم کردن این شرط، الگوریتم K-SVD، تعریف نمونه‌های داده به صورت یک ترکیب خطی تنک از ستون‌های ماتریس D را به عنوان هدف تعیین می‌کند. در این الگوریتم، مسئله بهینه‌سازی مذکور به صورت زیر در می‌آید:

$$\min_{D,X}\{\|Y - DX\|_F^2\} \qquad \text{subject to} \quad \forall i, \|x_i\|_0 \le T$$

که معادل مسئله زیر است:

$$\min_{D,X} \sum_i \|x_i\|_0 \qquad \text{subject to} \quad \|Y - DX\|_F^2 \le \epsilon$$

برای حل مسئله فوق، از روش‌های Iterative استفاده می‌شود. یعنی، در ابتدا فرض می‌شود، ماتریس D معلوم است و بهترین ماتریس ضریب X محاسبه می‌گردد. از آنجایی که پیدا کردن ماتریس ضریب X بهینه، غیر ممکن است، الگوریتم‌های approximation pursuit مورد استفاده قرار می‌گیرند.

در ادامه جستجو برای یک ماتریس کد D بهتر انجام می‌گیرد. از آنجایی که پیدا کردن همه ماتریس کد D به طور یکجا ممکن نیست، فرآیند از طریق به‌روزرسانی ستون به ستون ماتریس کد D صورت می‌پذیرد. به‌روزرسانی ستون j ام ماتریس کد D، از بازنویسی تابع هزینه به صورت زیر بدست می‌آید:

$$\min_{d_j}\left\{\left\|E_j\Omega_j - d_j x_j^T\Omega_j\right\|_F^2\right\}$$

در عبارت فوق، E_k ماتریس خطای حاصل از در نظر نگرفتن ستون j ام ماتریس کد D برای بازسازی ماتریس Y است؛ یعنی $E_k = Y - \sum_{i\neq j} d_i x_i^T$ است. همچنین ماتریس Ω_j، ماتریس حاصل از کنار هم قرار دادن ستون‌هایی از ماتریس همانی است که درایه متناظر آن ستون در x_i^T مخالف صفر است. در نتیجه جواب مسئله بهترین تخمین مرتبه یک ماتریس $E_j\Omega_j$ است که به وضوح برابر اولین ستون ماتریس U در تجزیه SVD آن است.

دو مرحله فوق آنقدر ادامه می‌یابند تا تغییرات ماتریس‌های D و X در دو گام متوالی بسیار کوچک باشد.

اگر فرض شود که ماتریس Y پیش از اضافه شدن نویز، از روی k اتم به صورت تنک، قابل بازسازی بوده است، می‌توان از خروجی مسئله فوق به عنوان نسخه نویززدایی شده ماتریس Y استفاده نمود.

در واقع الگوریتم فوق روشی برای بازسازی تنک تصاویر ارائه می‌دهد؛ زیرا اگر فرض شود ماتریس Y معرف یک تصویر است، الگوریتم فوق، روشی برای بازسازی تنک این ماتریس از روی ستون‌های ماتریس D ارائه می‌دهد. شایان ذکر است میزان تنک بودن خروجی به میزان خطای قابل قبول در بازسازی بستگی دارد و با تنظیم مناسب پارامتر T قابل دستیابی است.

علت آن که در الگوریتم فوق از نرم فروبینیوس استفاده شده است، در نظر گرفتن نویز به صورت نویز جمع‌شونده گوسی است. برای درک این موضوع فرض کنید هدف یافتن ماتریس‌های D و X به گونه‌ای است که در صورت داشتن ماتریس DX به عنوان نسخه بدون نویز، احتمال رویت ماتریس Y در اثر نویز جمع‌شونده گوسی، بیشینه باشد. در نتیجه مسئله مورد نظر به صورت زیر خواهد بود:

$$\max_{D,X}\sum_i\sum_j \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(Y_{ij}-(DX)_{ij})^2}{2\sigma^2}} = \max_{D,X} e^{-\frac{\|Y-DX\|_F^2}{2\sigma^2}} = \min_{D,X}\{\|Y-DX\|_F^2\}$$

در نتیجه در صورت در نظر گرفتن نویز دیگری برای مدل‌سازی، باید به طور متناسب با آن، تعریف تابع هزینه و در اثر آن، روش حل دو مرحله مسئله نیز تغییر کنند.

سوال ۳: در صورت سوال قید شده است که برای بازسازی هر یک از سیگنال‌های ۶۴ بعدی، تنها می‌توان از سه اتم موجود در ماتریس کد استفاده کرد. اگر فرض شود، اتم‌های ماتریس کد به گونه‌ای انتخاب شده اند که هیچ چهار اتمی در یک فضای سه بعدی قرار نمی‌گیرند، تعداد زیرفضاهای تشکیل شده برای بازیابی تنک، که همگی سه بعدی و متمایز هستند، برابر است با:

$$\binom{100}{3} = \frac{100!}{97!*3!} = \frac{100*99*98}{3!} = 161700$$

سوال ۴: ماتریس کد D مورد نظر برابر است با:

$$D = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

همچنین بردار x برابر است با:

$$x = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

در نتیجه به وضوح بهترین بازسازی تنک بردار x بر اساس ماتریس کد D برابر است با:

$$y = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 0 \end{pmatrix}$$

زیرا با ضرب ماتریس کد D در بردار y، بردار x بدست می‌آید:

$$Dy = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

سوال ۵: Basis pursuit، مسئله بهینه‌سازی به صورت زیر است:

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

که در آن x یک بردار N تایی، A یک ماتریس $M*N$ و y یک بردار M تایی است. از این مسئله می‌توان به عنوان جایگزینی برای مسئله بهینه‌سازی مربوط به تنک‌سازی به صورت زیر استفاده نمود:

$$\min_{D,X} \sum_i \|x_i\|_0 \qquad \text{subject to} \quad \|Y - DX\|_F^2 \leq \epsilon$$

Matching pursuit، الگوریتمی حریصانه برای حل تقریبی مسئله بهینه‌سازی زیر است:

$$\min_x \{\|y - Dx\|_2^2\} \qquad \text{subject to} \quad \|x\|_0 \leq T$$

نحوه عملکرد این الگوریتم بدین صورت است که در هر مرحله، بهترین تخمین رتبه یک بردار r بر اساس ستون‌های ماتریس کد D را بدست می‌آورد. این کار از طریق محاسبه ضریب همبستگی بین بردار r و ستون‌های ماتریس کد D انجام می‌گیرد. فرآیند مذکور، T مرحله تکرار می‌شود تا نتیجه نهایی حاصل شود. بردار r در ابتدای مرحله اول، برابر بردار y است و در انتهای هر مرحله به صورت زیر تغییر می‌کند:

$$r^{i+1} = r^i - < r^i, d_{j^*} > d_{j^*} \qquad \text{where} \quad j^* = \underset{j}{\operatorname{argmax}} \frac{| < r^i, d_j > |}{\|d_j\|_2}$$

از این الگوریتم می‌توان به عنوان جایگزینی برای مسئله بهینه‌سازی مربوط به تنک‌سازی به صورت زیر استفاده نمود:

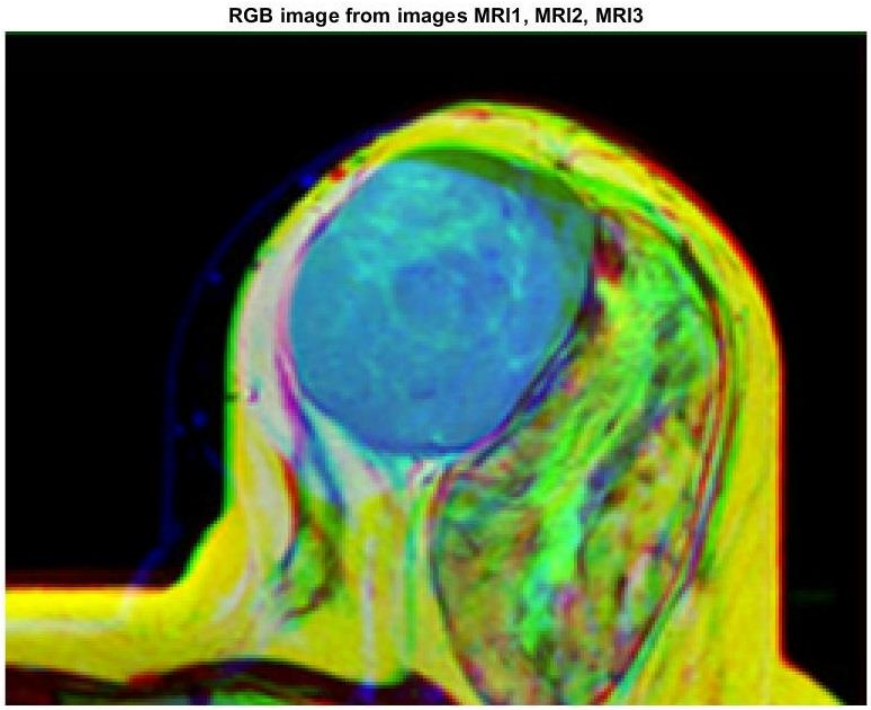
$$\min_{D,X} \{\|Y - DX\|_F^2\} \qquad \text{subject to} \quad \forall i, \|x_i\|_0 \leq T$$

بسته به کاربرد هم مسئله basis pursuit و هم الگوریتم matching pursuit می‌توانند برای نرم کردن قیدهای مسئله sparse representation مورد استفاده قرار بگیرند. مسئله basis pursuit بهترین جواب ممکن برای بازسازی بردار y بر اساس ستون‌های ماتریس کد D را ارائه می‌دهد؛ اما این مسئله، پارامتری برای تعیین میزان تنک بودن بردار x ندارد و در صورتی که بردار y بر اساس ستون‌های ماتریس کد D به صورت تنک قابل بیان نباشد، تقریب مناسبی برای مسئله sparse representation نخواهد بود. از طرف دیگر الگوریتم matching pursuit ممکن است خطای بازسازی زیادی داشته باشد؛ اما در این الگوریتم میزان تنک بودن به صورت دقیق با تعیین تعداد مراحل قابل کنترل است و در نتیجه خروجی این مسئله همواره یک خروجی زیربهینه برای مسئله sparse representation خواهد بود.

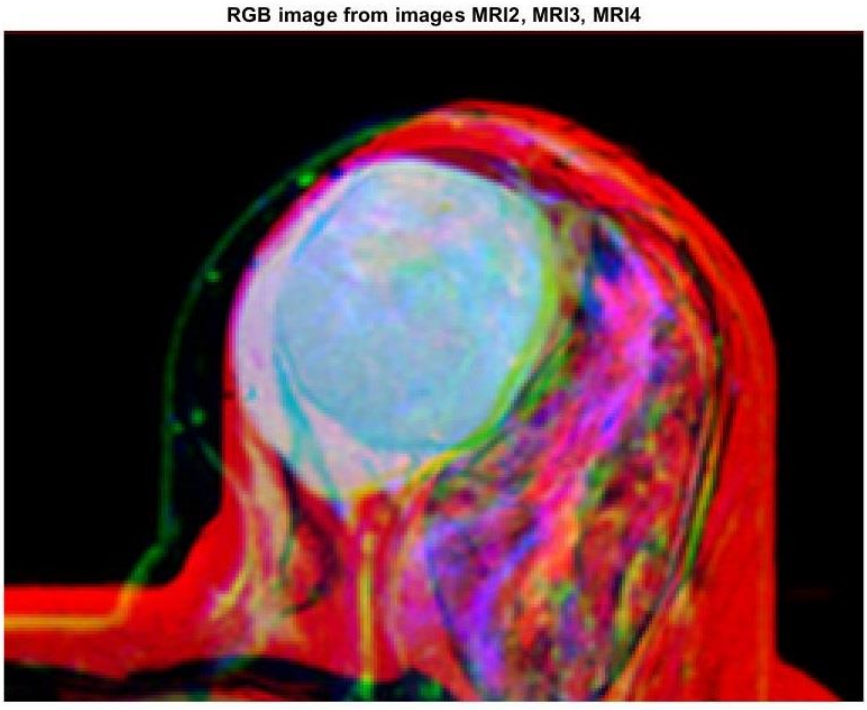
بخش شبیه‌سازی

سوال ۱: روش‌های ناحیه بندی مبتنی بر خوشه‌یابی

۱-۱: برای مشاهده هم‌زمان تصاویر، ۳ تصویر دلخواه از ۴ تصویر MRI را به عنوان کانال‌های رنگی RGB استفاده می‌کنیم. خروجی به صورت زیر بدست می‌آید:



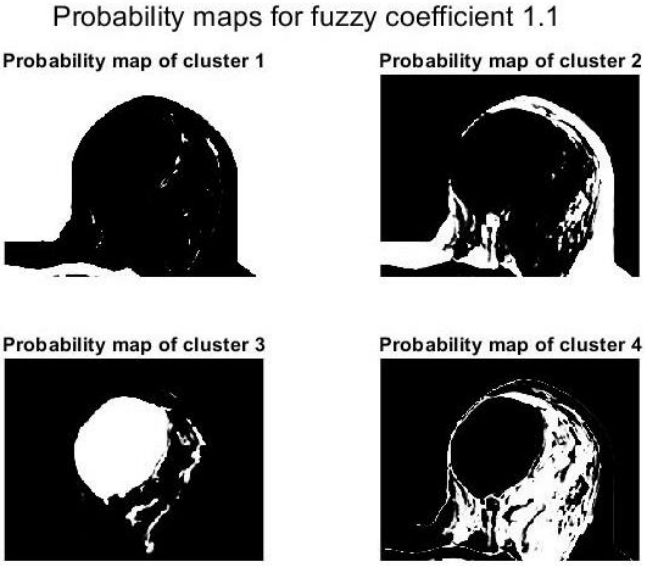
تصویر ۱، کانال رنگی R، تصویر ۲، کانال رنگی G و تصویر ۳، کانال رنگی B را تشکیل می‌دهد.



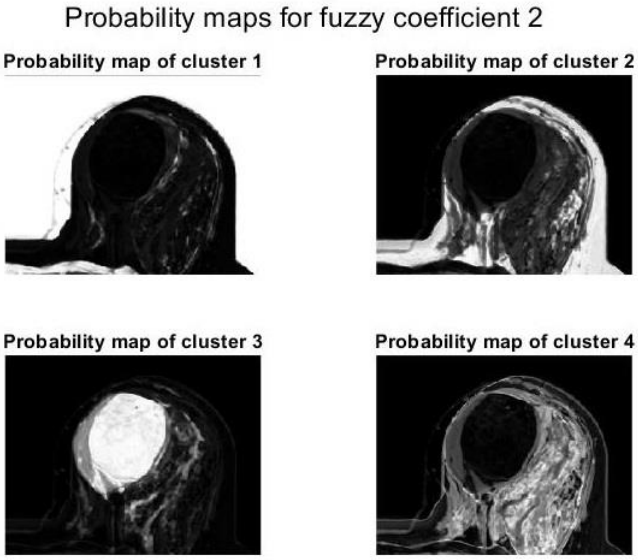
تصویر ۲، کانال رنگی R، تصویر ۳، کانال رنگی G و تصویر ۴، کانال رنگی B را تشکیل می‌دهد.

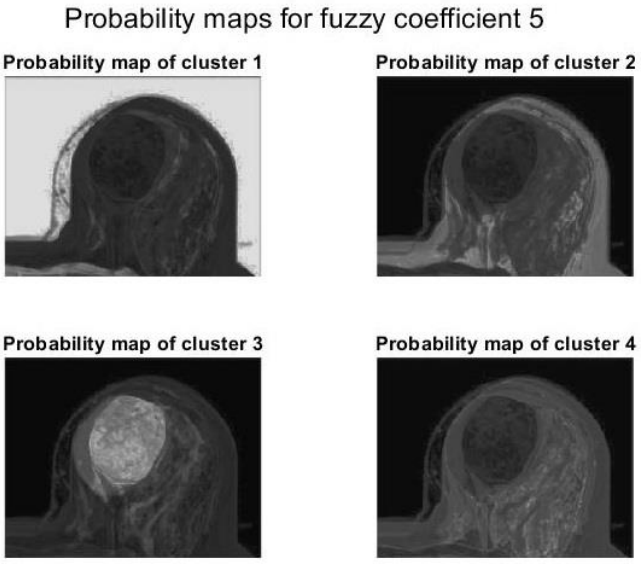
با توجه به تصاویر، علاوه بر کلاس پس‌زمینه، می‌توان ۳ کلاس دیگر در نظر گرفت.

۲-۱: با استفاده از تابع fcm متلب، نقشه‌های احتمال را بدست می‌آوریم. خروجی به صورت زیر بدست می‌آید:



ناحیه‌بندی به صورت سخت (ضریب فازی ۱/۱) است.

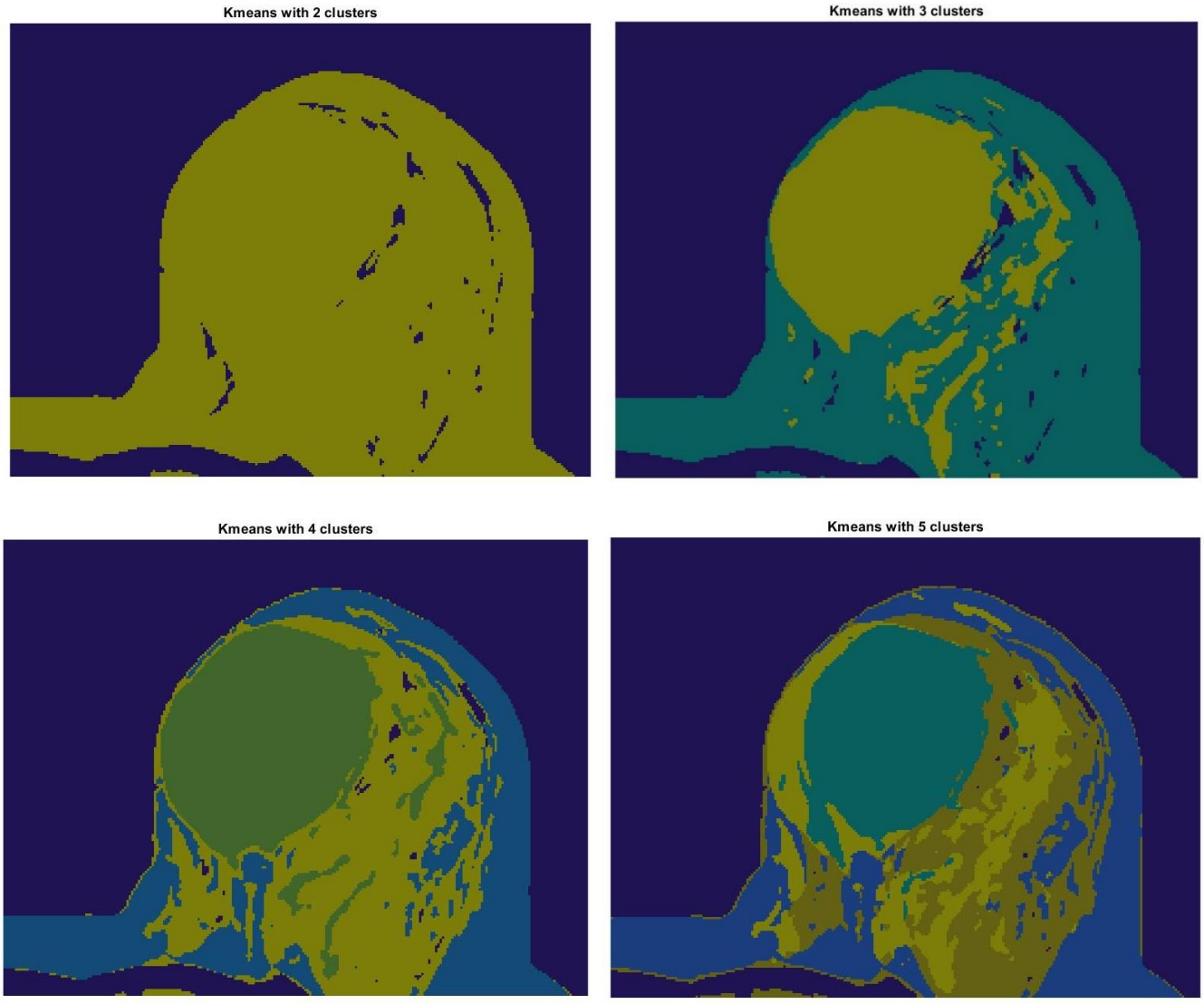




ناحیه‌بندی به صورت نرم (ضریب‌های فازی ۲ و ۵) است.

به طور مشخص به ازای ضریب فازی ۱/۱ (ناحیه‌بندی سخت)، مقادیر u_{ij} به مقادیر نزدیک به صفر یا یک میل می‌کنند و در نتیجه نقشه‌های احتمال چهار ناحیه تقریبا شبیه bitmap است و هر پیکسل تنها به یک کلاس تعلق دارد. با افزایش مقدار ضریب فازی به مقادیر ۲ و ۵ (ناحیه‌بندی نرم)، مقادیر u_{ij} به سمت داشتن مقادیر یکسان حرکت می‌کنند و لذا نقشه‌های احتمال از حالت صفر و یکی خارج می‌شوند و ماهیت خاکستری به خود می‌گیرند. همچنین در این حالت هر پیکسل به همه کلاس‌های موجود با احتمال‌های مناسب متعلق است.

۳-۱: ابتدا با استفاده از روش K-means ناحیه‌بندی اولیه را برای تعداد کلاس‌های مختلف (۲ تا ۵ کلاس) بدست می‌آوریم. خروجی به صورت زیر بدست می‌آید:

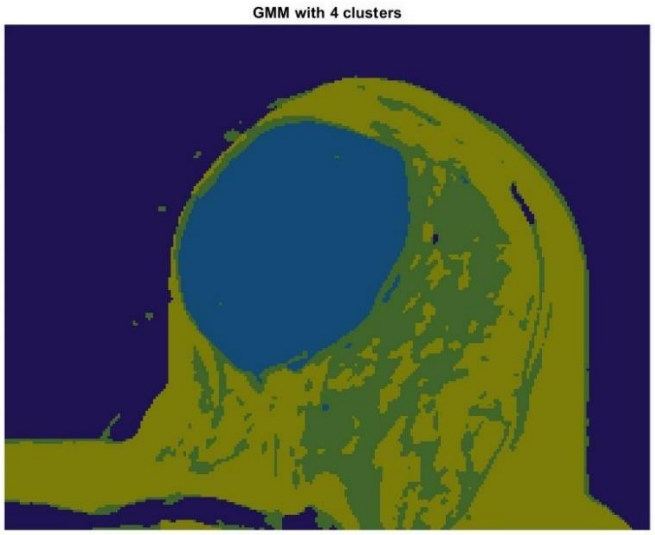


با توجه به تصاویر، علاوه بر کلاس پس‌زمینه، می‌توان ۳ کلاس دیگر در نظر گرفت.

با تغییر جزئی تابع fcm متلب، تابع adjusted_fcm را با خصوصیات مذکور، پیاده‌سازی می‌کنیم. سپس از خروجی روش K-means به عنوان شرایط اولیه برای مقادیر u_{ij} استفاده می‌کنیم. نتیجه ناحیه‌بندی با دقت خوبی مانند قسمت قبل است؛ اما تعداد iteration لازم برای رسیدن به جواب به مقدار زیادی کاهش یافته است. با توجه به این موضوع، می‌توان در ابتدا با استفاده از روش K-means، که سرعت همگرایی بالایی دارد، مسئله ناحیه‌بندی را حل کرد و سپس با قرار دادن نتیجه آن به عنوان شرایط اولیه روش FCM، سرعت همگرایی این روش را افزایش داد.

۴-۱: ابتدا با استفاده از تابع fitgmdist، یک توزیع GMM ۴ مولفه‌ای (به ازای ۴ کلاس ناحیه‌بندی) به سطح روشنایی پیکسل‌ها در ۴ تصویر fit می‌کنیم و سپس بر اساس این توزیع در هر پیکسل، محتمل‌ترین کلاس را انتخاب می‌کنیم.

خروجی به صورت زیر بدست می‌آید:

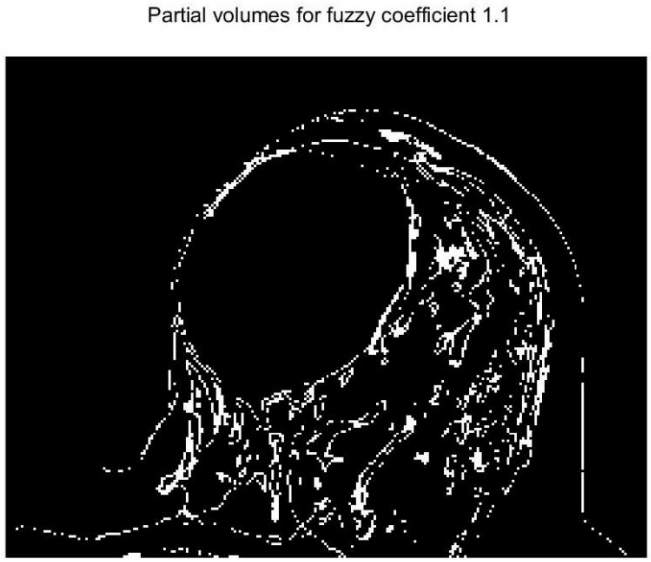


۵-۱: برای رسم تصویری با خصوصیات مذکور، بر اساس تعداد کلاس‌های ناحیه‌بندی و مقدار ضریب فازی، آستانه‌ای بر روی احتمال محتمل‌ترین کلاس ناحیه‌بندی تعریف می‌کنیم.

$$\text{threshold} = \max\left(\frac{1.25}{\text{number of clusters}}, \frac{1}{\text{fuzzy coefficient}}\right)$$

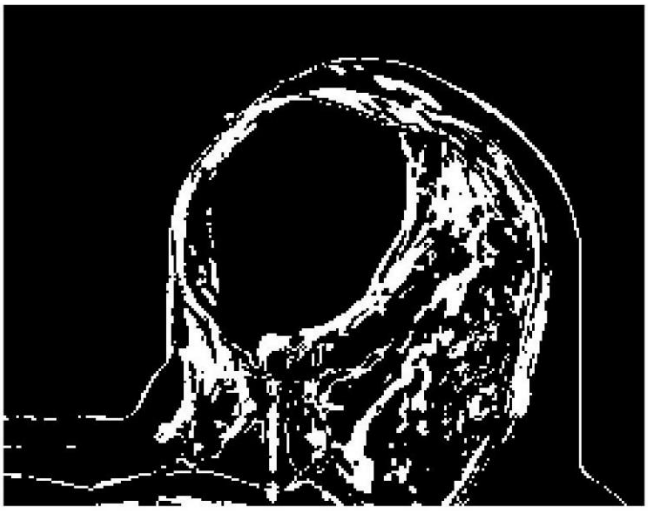
اگر احتمال محتمل‌ترین کلاس ناحیه‌بندی برای یک پیکسل کمتر از این مقدار آستانه باشد، آن پیکسل به عنوان پیکسلی تحت تاثیر partial volume شناسایی می‌شود.

خروجی به صورت زیر بدست می‌آید:

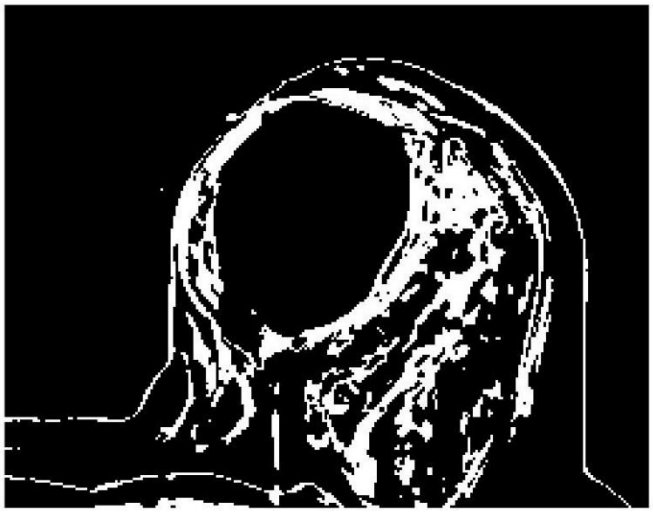


ناحیه‌بندی به صورت سخت (ضریب فازی ۱/۱) است. در نتیجه به علت کوچک بودن ضریب فازی، مقادیر u_{ij} به مقادیر نزدیک به صفر یا یک میل می‌کنند و نواحی کمی به عنوان partial volume شناسایی می‌شوند. این نواحی به گونه‌ای معرف مرز بین کلاس‌های مختلف ناحیه‌بندی هستند.

Partial volumes for fuzzy coefficient 2



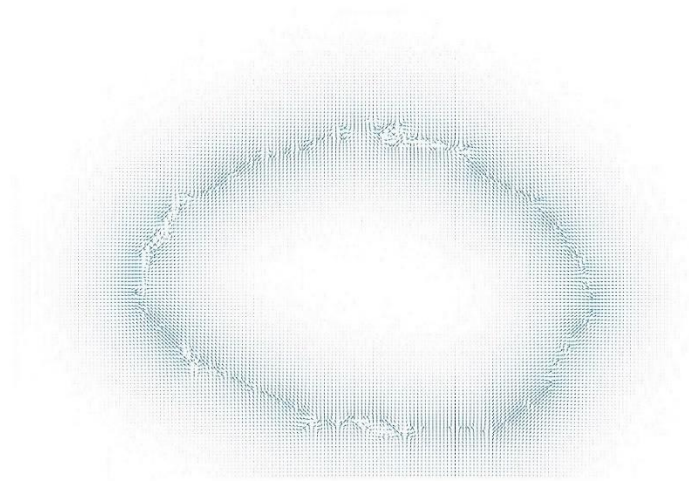
Partial volumes for fuzzy coefficient 5



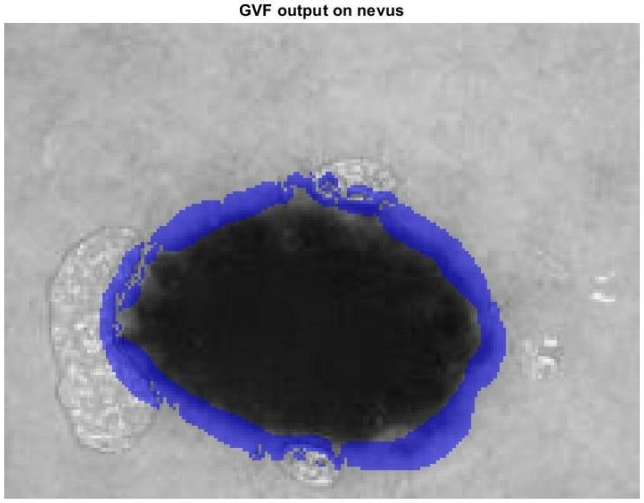
ناحیه‌بندی به صورت نرم (ضریب‌های فازی ۲ و ۵) است. در نتیجه به علت بزرگ بودن ضریب فازی، مقادیر u_{ij} به سمت داشتن مقادیر یکسان حرکت می‌کنند و نواحی زیادی به عنوان partial volume شناسایی می‌شوند. مساحت این نواحی با افزایش ضریب فازی، افزایش می‌یابد.

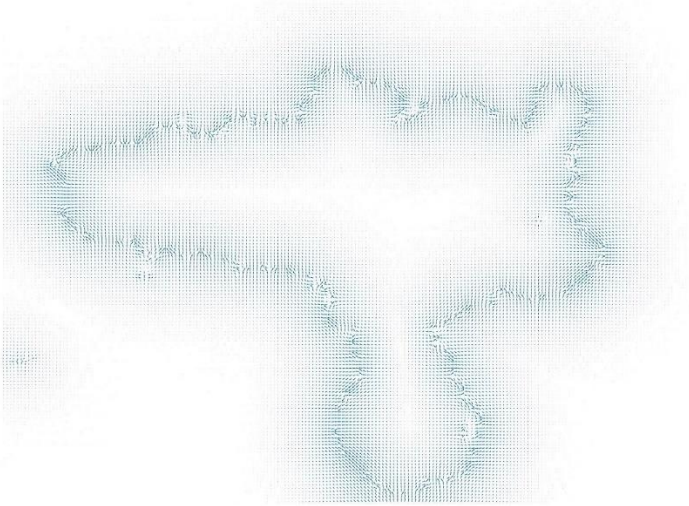
سوال ۲: روش‌های GVF و Basic Snake

۲-۱: ابتدا از روش GVF استفاده می‌کنیم. خروجی به صورت زیر بدست می‌آید:

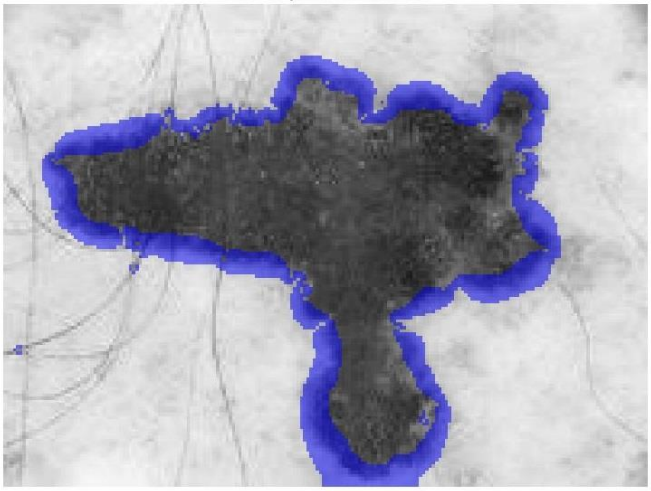


GVF output on nevus

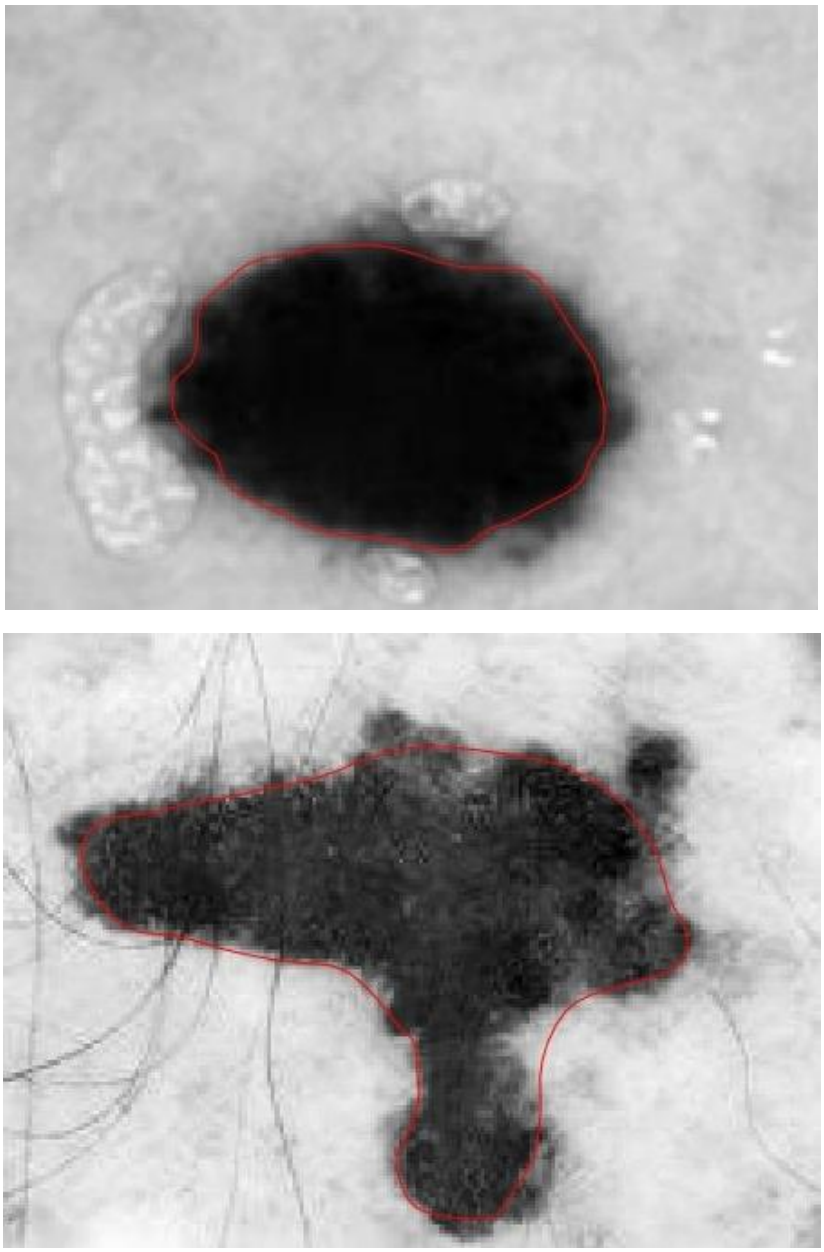




GVF output on melanoma



سپس از روش Basic Snake استفاده می‌کنیم. خروجی به صورت زیر بدست می‌آید:



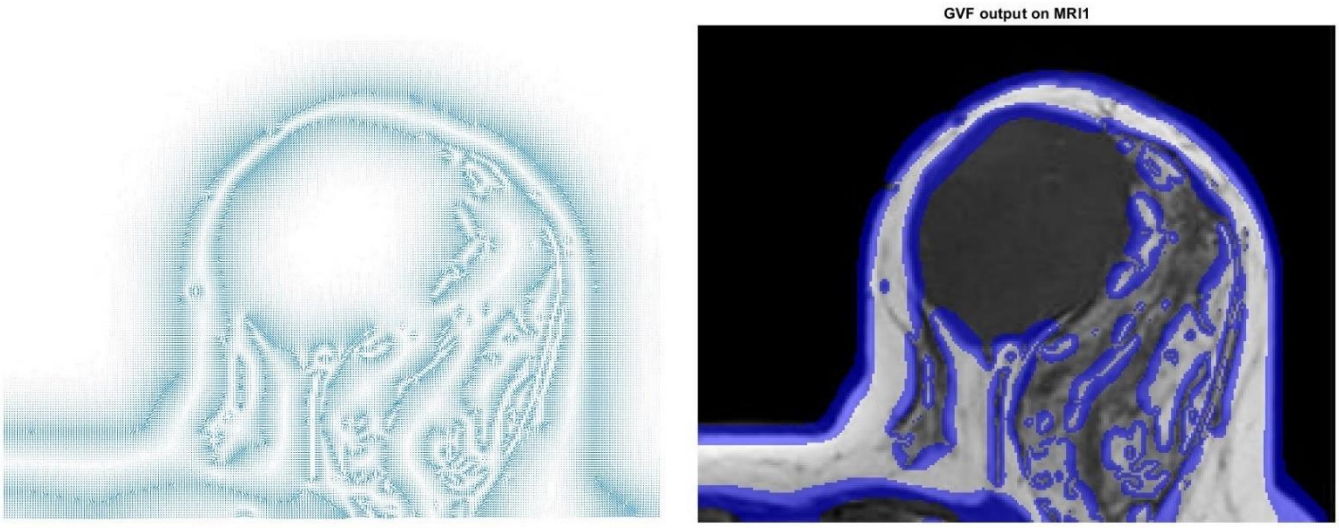
روش GVF بسیار ساده اجرا می‌شود و نیازی به تعیین پارامترهای زیاد ندارد. اما از آنجایی که خروجی این روش یک بردار برای هر پیکسل است، زمانی که می‌خواهیم با استفاده از اندازه بردار مذکور در هر پیکسل، مرز ناحیه را تعیین کنید، مشکلاتی از قبیل ضخیم شدن مرز، غیر یکنواخت بودن عرض مرز، گسسته بودن مرز و ... ممکن است پیش بیاید.

روش Basic Snake سختی‌های زیادی برای اجرای مناسب دارد. به طور دقیق، ابتدا باید تعدادی نقطه برای ساخت خم بسته اولیه انتخاب شوند و سپس مقدار تعداد زیادی پارامتر تعیین گرددد تا در آخر الگوریتم قابل اجرا باشد. این روش به شدت به نحوه انتخاب نقاط اولیه و مقدار پارامترهای مسئله، حساس است و در نتیجه در تصویر melonama که دارای یک مرز غیر محدب است، زحمت زیادی برای رسیدن به جواب مناسب لازم است. اما از نظر جواب نهایی، این روش عملکرد خوبی دارد و خروجی همواره یه خم بسته هموار است.

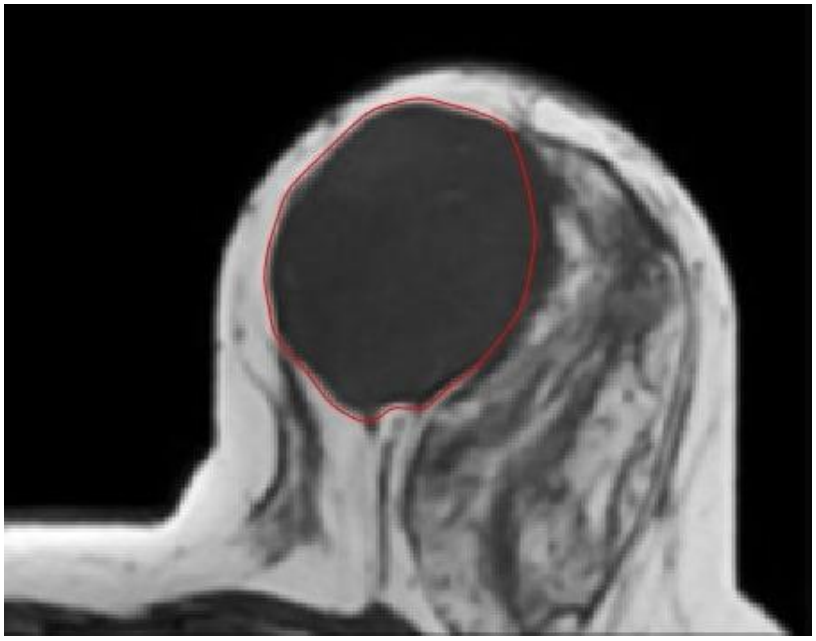
در انتها، هر دو روش برای هر دو تصویر عملکرد مناسبی دارند؛ اما به طور کلی رسیدن به جواب مناسب با روش GVF بسیار راحت‌تر از روش Basic Snake است.

۲-۲: روش‌های مذکور را بر روی تصویر ۱ اعمال می‌کنیم. خروجی به صورت زیر بدست می‌آید:

روش GVF:



روش Basic Snake:

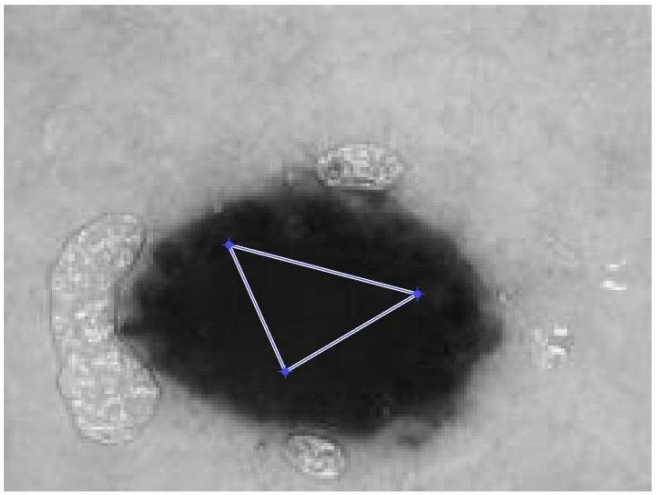


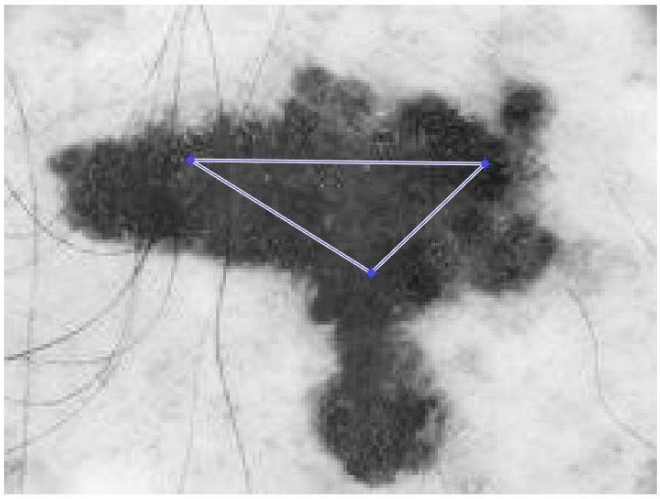
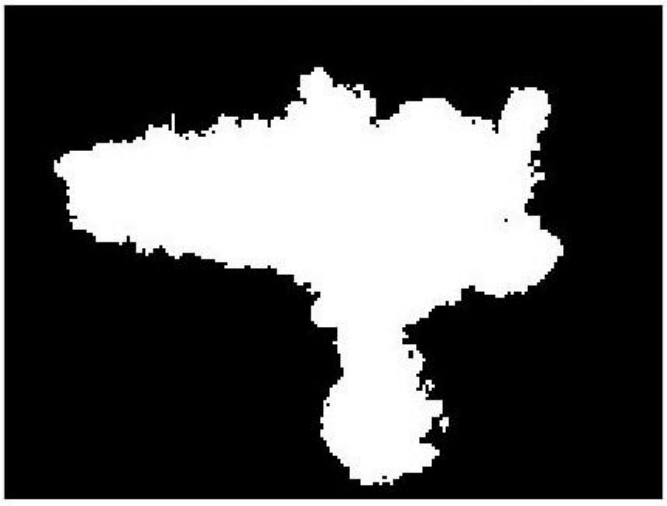
روش‌های GVF و Bsic Snake تنها می‌توانند تصویر را به دو ناحیه تقسیم کنند. در نتیجه در مقایسه با روش FCM، که تعداد کلاس‌های ناحیه‌بندی به سادگی قابل تنظیم است، عملکرد ضعیف‌تری دارند. همچنین رسیدن به خروجی خوب در روش FCM راحت‌تر است. اما این دو روش در جدا کردن یک ناحیه از ناحیه‌های دیگر، دقت قابل قبولی دارند.

در روش GVF، با توجه به این که مقدار بردار پیکسل‌های مختلف از یکدیگر مجزا هستند، می‌توانند حول مرز نواحی جدا از هم متعلق به کلاس ناحیه‌بندی مورد نظر، مقادیر بزرگی اتخاذ کنند و در نتیجه امکان تشخیص نواحی جدا از هم وجود دارد. اما در روش Basic Sanke، از آنجایی که در هر مرحله از اجرای الگوریتم، تنها یک خم بسته وجود دارد، امکان تشخیص نواحی جدا از هم متعلق به یک کلاس ناحیه‌بندی به طور هم‌زمان وجود ندارد. البته می‌توان به ازای هر یک از نواحی جدا از هم، یک بار این روش را اجرا کرد؛ اما عمل کردن بدین صورت ممکن است به علت‌های مختلفی از جمله نیاز به وجود فرد متخصص، زمان‌بر بودن و ...، معقول و منطقی نباشد.

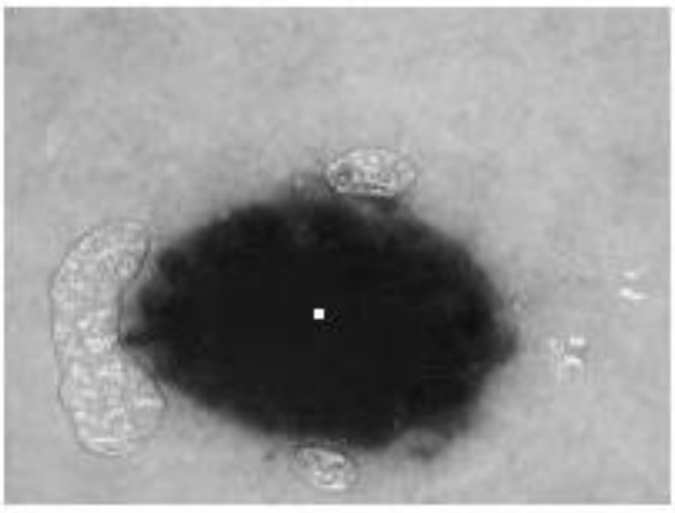
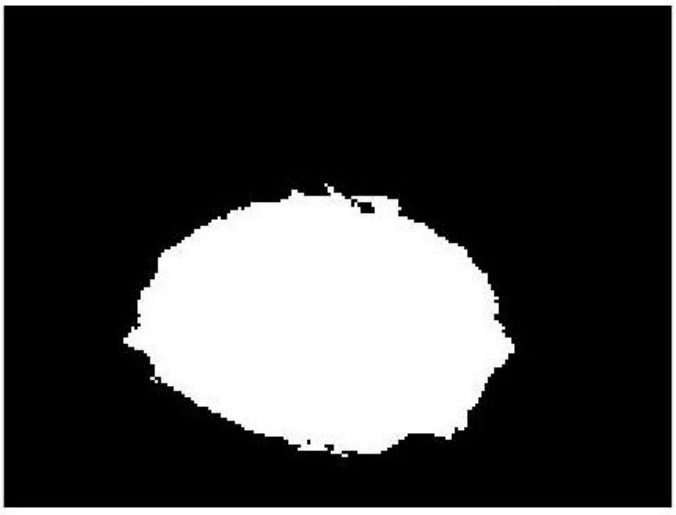
سوال ۳: روش chan-vese

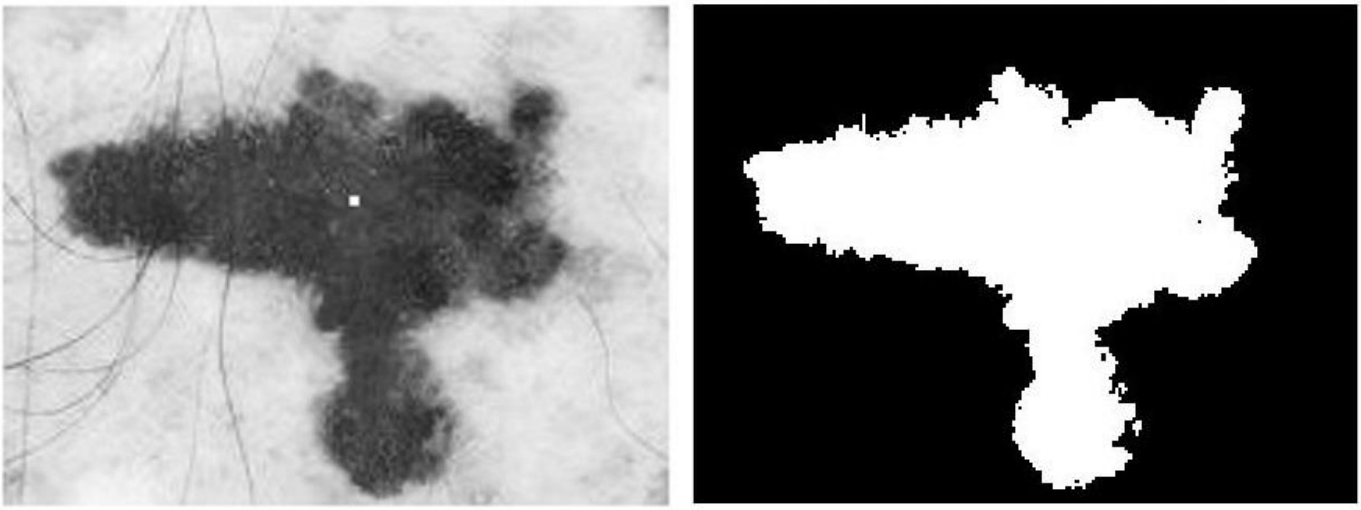
۳-۱: ابتدا حالتی را در نظر می‌گیریم که ماسک اولیه (که قسمتی از ضایعه است) توسط کاربر ایجاد شود. خروجی به صورت زیر بدست می‌آید:



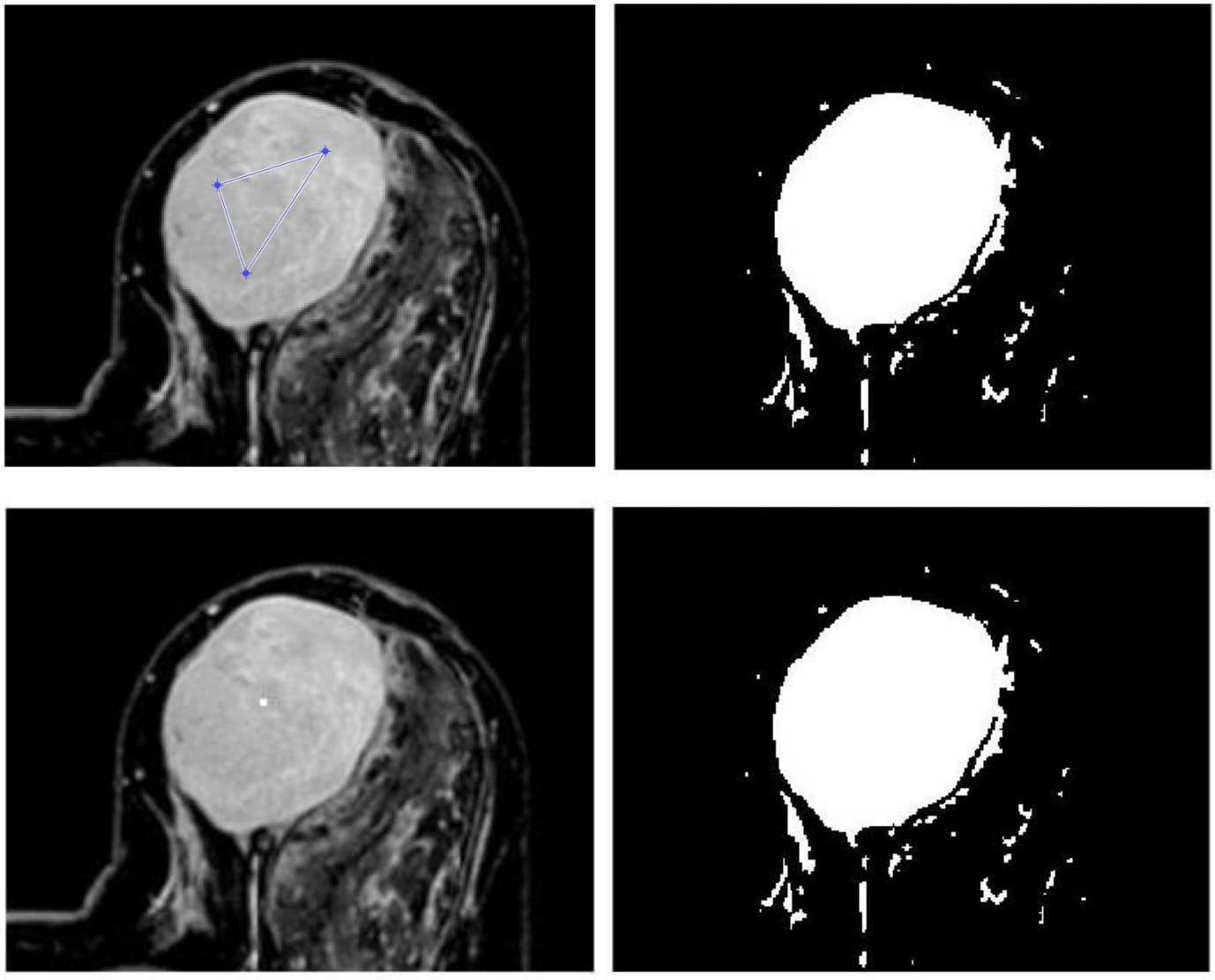



سپس حالتی را در نظر می‌گیریم که ماسک اولیه به صورت یک مربع کوچک در ابعاد ۹*۹ باشد که مرکز آن توسط کاربر تعیین شود. خروجی به صورت زیر بدست می‌آید:



۲-۳: حالت‌های مذکور را بر روی تصویر ۳ اعمال می‌کنیم. خروجی به صورت زیر بدست می‌آید:



این روش نیز مانند روش‌های GVF و Bsic Snake تنها می‌تواند تصویر را به دو ناحیه تقسیم کند. اما عملکرد آن در جدا کردن دو ناحیه مذکور از یکدیگر، بسیار خوب است؛ به طوری که حتی با تعیین یک نقطه در ناحیه مورد نظر، با دقت بسیار بالا ناحیه مورد نظر را از ناحیه دیگر جدا می‌کند. همچنین در این روش نیازی به تعیین پارامترهای زیاد نیست و به سادگی می‌توان از آن استفاده کرد.

برای خودکار کردن این روش به صورت زیر عمل می‌کنیم:

ابتدا یک فیلتر median بر روی تصویر اعمال می‌کنیم و سپس با توجه به آن که ناحیه مورد نظر روشن‌تر از جاهای دیگر تصویر است، یکی از پیکسل‌هایی که در تصویر حاصل، دارای مقدار بیشینه است را به دلخواه انتخاب می‌کنیم و در انتها از این پیکسل به عنوان پیکسل انتخابی در حالتی که ماسک اولیه به صورت یک مربع کوچک در ابعاد ۹*۹ حول یک پیکسل مرکزی است، استفاده می‌کنیم.