دانشگاه صنعتی شریف

دانشکده مهندسی برق

مسعود ناطقی ۹۶۱۰۲۵۶۷

تمرین پردازش و تحلیل تصاویر پزشکی

دکتر فاطمی‌زاده

۱. الگوریتم jpeg روشی برای فشرده‌سازی (با اتلاف) عکس‌های دیجیتال است به نحوی که (اگر درجه فشرده‌سازی چندان بالا نباشد) تصویر فشرده‌شده تفاوت چندانی با تصویر اصلی نکند. این درجه فشرده‌سازی قابل تغییر است و با توجه به tradeoff بین حجم ذخیره‌سازی تصویر و کیفیت تصویر، تعیین می‌شود. Jpeg می‌تواند بدون از دست دادن بخش قابل توجهی از تصویر به درجه فشرده‌سازی ۱۰:۱ برسد. Jpeg بر اساس تبدیل فوریه کسینوسی (DCT) انجام می‌شود و در آن سیگنال تصویر از فضای مکان به فضای فرکانسی نگاشت می‌شود و در فضای نگاشت نوعی quantization انجام می‌شود به نحوی که مولفه‌های فرکانس بالای تصویر که برای چشم انسان قابل تشخیص نیستند حذف می‌شوند و با حذف شدن این مولفه‌ها و quantization، فشرده‌سازی تصویر صورت می‌گیرد. حذف کردن مولفه‌های فرکانس بالا به نحوی همان sparse modeling را نشان می‌دهد که در واقع ضرایب مولفه‌های فرکانس بالا را برابر صفر قرار می‌دهیم و ضرایب sparse تر می‌شوند. [1]

۲. هدف این الگوریتم[2]، مینیمم کردن عبارت زیر است:

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

۱- ابتدا قرار می‌دهیم $X = Y$ و D را یک دیکشنری overcomplete از خانواده DCT انتخاب می‌کنیم.

۲- j بار تکرار می‌کنیم:

مرحله کد کردن تنک: در این مرحله با استفاده از هر نوع الگوریتم pursuit بردارهای نمایش α_{ij} برای هر یک از patchها را به دست می‌آوریم.

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

که در رابطه بالا C گین نویز است.

مرحله آپدیت کردن دیکشنری: هر اتم دیکشنری $l = 1, 2, \dots, k$ را بصورت زیر آپدیت می‌کنیم:

ابتدا patchهایی که از این اتم استفاده می‌کنند را پیدا می‌کنیم: $\omega_l = \{(i,j) | \alpha_{ij}(l) \neq 0\}$

سپس برای هر patch پیدا شده خطای نمایش را بصورت زیر محاسبه می‌کنیم:

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

ماتریس E_l را که ستون‌های آن همان e_{ij}^lها هستند تشکیل می‌دهیم و روی آن SVD را اعمال می‌کنیم: $\mathbf{E}_l = \mathbf{U}\boldsymbol{\Delta}\mathbf{V}^T$

[1] https://en.wikipedia.org/wiki/JPEG

[2] Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries, Michael Elad, Michal Aharon

اتم آپدیت شده بصورت $\hat{d}_l = U(:,1)$ در نظر می‌گیریم و بردارهای نمایش بصورت ضرب V در $\Delta(1,1)$ می‌شود.

در آخر تخمین حذف نویز شده X بصورت زیر به دست می‌آید:

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

در این الگوریتم سعی می‌شود تا برای هر patch در هر iteration از الگوریتم، نمایشی تنک به دست بیاید و در نهایت در قسمت آخر الگوریتم که X به دست می‌آید در واقع نوعی میانگین‌گیری بین تصویر نویزی (λY) و تصاویر بازسازی شده از تصاویر تنک $(D\alpha_{ij})$ انجام می‌شود و بنابراین ارتباط این تصاویر با بازسازی تصاویر از روی نمایش تنک مشخص می‌شود.

اگر از نویز دیگری غیر از نویز گوسی برای مدل کردن نویز استفاده کردیم، می‌بایست بخش مرحله کد کردن تنک را تغییر دهیم. به این صورت که در بخش مشخص شده در رابطه زیر از نرم دیگری برای محاسبه خطا استفاده کنیم:

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

مثلا اگر از نویز لاپلاسی برای مدل کردن نویز استفاده می‌کنیم رابطه بصورت زیر تغییر می‌یابد:

$$\forall_{ij} \ \min_{\alpha_{ij}} \|\alpha_{ij}\|_0 \quad s.t. \ \ \|R_{ij}X - D\alpha_{ij}\|_1 \leq \epsilon$$

۳. با فرض اینکه اتم‌های دیکشنری متعامد (یا فرض نرم‌تر، مستقل) باشند، می‌توان $\binom{100}{3}$ زیرفضا برای بازنمایی تنک متصور شد در صورتی که دقیقا هر سیگنال نمایشی با ترکیب فقط و فقط ۳ اتم داشته باشد.

۴.

$$x = Dy \rightarrow \begin{bmatrix}2\\2\end{bmatrix} = \begin{bmatrix}2 & 1 & 1 & 0\\1 & 0 & 1 & 1\end{bmatrix}\begin{bmatrix}0\\0\\2\\0\end{bmatrix} \rightarrow y = \begin{bmatrix}0\\0\\2\\0\end{bmatrix}$$

۵. روش matching pursuit[3] یک الگوریتم sparse approximation است و اتم‌هایی را انتخاب می‌کند که در آن داده چندبعدی بیشترین مقدار بازتاب (projection) را روی آن اتم‌ها دارد. اتم‌ها مربوط به یک دیکشنری overcomplete هستند. ایده اصلی این است که سیگنال f را که از فضای هیلبرت H آمده است، بصورت جمع وزن‌دار اتم‌های دیکشنری D (که تعداد اتم‌هایش محدود (برابر N) است) بنویسیم.

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

[3] https://en.wikipedia.org/wiki/Matching_pursuit

که در این رابطه g_{γ_n} ستون (اتم) γ_n-ام از D است. در این الگوریتم از تعداد کمی از اتم‌ها برای نمایش سیگنال استفاده می‌شود. این الگوریتم سعی می‌کند هر بار اتم‌هایی را انتخاب کند تا بصورت بیشینه (greedly) خطای بازسازی کم شود. این کار هر بار با انتخاب اتمی انجام می‌شود که بیشترین ضرب داخلی را با سیگنال تولید می‌کند (البته با فرض اینکه اتم‌ها نرمالیزه هستند). سپس سیگنال بازسازی شده در هر مرحله از سیگنال اصلی کم می‌کنیم و این کار را مرتبا تکرار می‌کنیم تا سیگنال تا حد مطلوب تجزیه شود. به عبارت دیگر نرم باقی‌مانده آن از آستانه‌ای کوچکتر شود که باقی‌مانده را بصورت زیر تعریف می‌کنیم:

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

اگر R_N سریعا به صفر میل کند به این معناست که از اتم‌های کمتری از D استفاده می‌شود. الگوریتم matching pursuit در واقع قصد دارد که مساله بهینه‌سازی زیر را حل کند که یک مساله NP-hard است و بنابراین سعی می‌شود از روش‌های approximation مانند matching pursuit استفاده شود.

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

عکس زیر صحبت‌های بالا را در مورد این الگوریتم بیان می‌کند:

Algorithm Matching Pursuit
Input: Signal: $f(t)$, dictionary D with normalized columns g_i.
Output: List of coefficients $(a_n)_{n=1}^N$ and indices for corresponding atoms $(\gamma_n)_{n=1}^N$.
Initialization:
  $R_1 \leftarrow f(t)$;
  $n \leftarrow 1$;
Repeat:
  Find $g_{\gamma_n} \in D$ with maximum inner product $|\langle R_n, g_{\gamma_n} \rangle|$;
  $a_n \leftarrow \langle R_n, g_{\gamma_n} \rangle$;
  $R_{n+1} \leftarrow R_n - a_n g_{\gamma_n}$;
  $n \leftarrow n + 1$;
Until stop condition (for example: $\|R_n\| < \text{threshold}$)
return

الگوریتم basis pursuit[4] در واقع مساله بهینه‌سازی زیر است:

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

که در آن $x \in \mathbb{R}^{N\times 1}$ و $y \in \mathbb{R}^{M\times 1}$ و $A \in \mathbb{R}^{M\times N}$ و $M > N$ است که معمولا در مواردی اعمال می‌شود که می‌خواهیم یک سیستم underdetermined از معادلات به فرم $y = Ax$ را به گونه‌ای حل کنیم که x تنک‌ترین جواب ممکن با در نظر گرفتن نرم L_1 باشد. همانطور که در اسلایدهای درس توضیح داده‌شد با توجه به اینکه کار کردن با نرم صفر سخت است (به دلیل مشتق ناپذیر بودن) بنابراین در مواردی که نرم L_1 معادل نرم صفر است، از نرم L_1 معادلا به جای نرم صفر برای مساله زیر استفاده می‌شود.

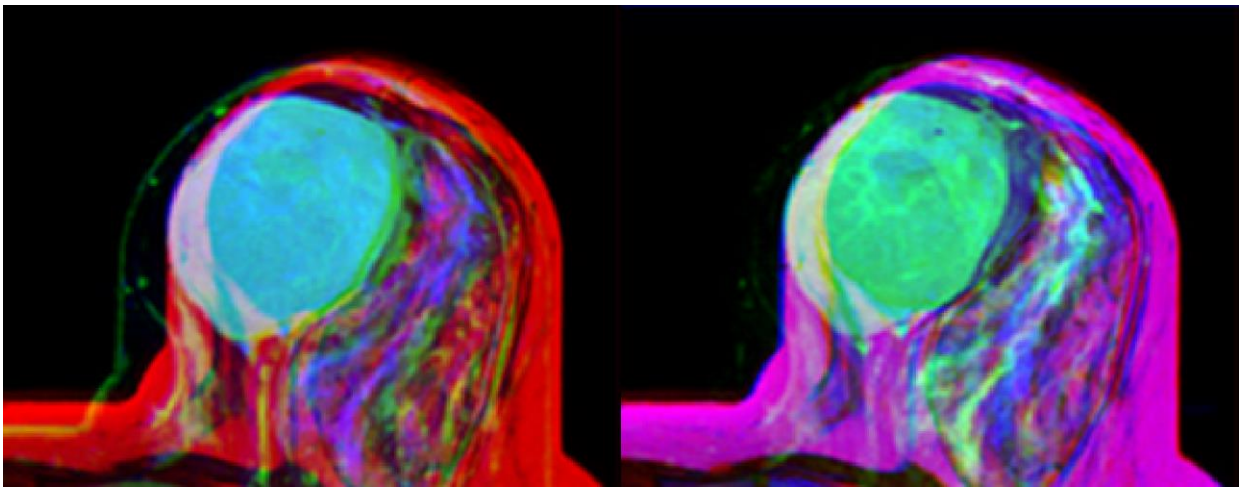
[4] https://en.wikipedia.org/wiki/Basis_pursuit

$$\min_{x} \|x\|_0 \quad subject\ to \quad y = Ax$$

در حالی که رویکرد دیگر این است که تابع نرم صفر را نرم‌تر کنیم و آن را حل کنیم که در واقع الگوریتم matching pursuit نیز همین کار را انجام می‌دهد.

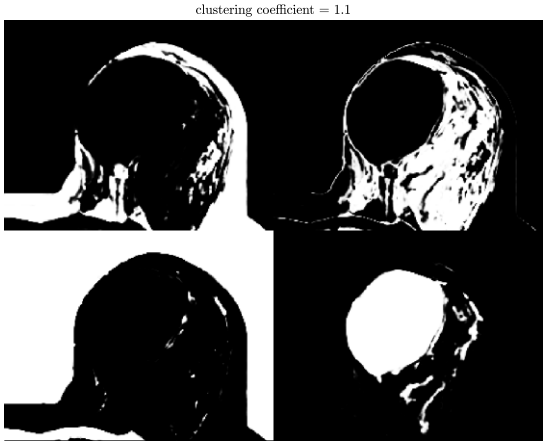
۱ روش‌های ناحیه‌بندی مبتنی بر خوشه‌یابی

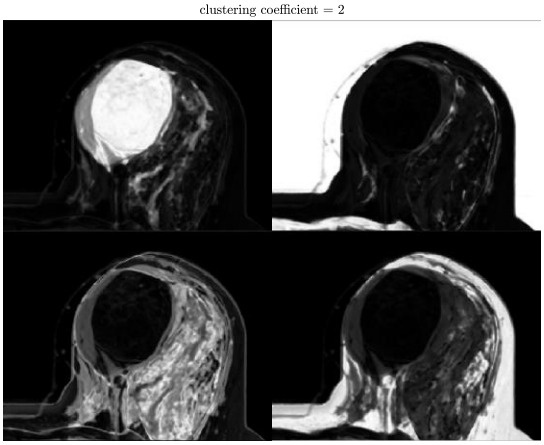
۱.۱

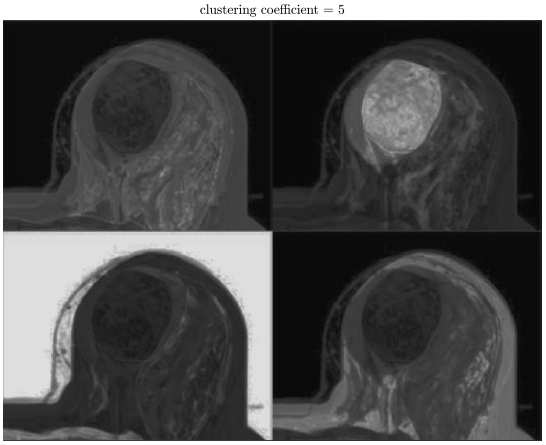


با توجه به تصاویر بالا ۴ طیف رنگ غالب در دو توصیر دیده می‌شود که به ترتیب بیانگر پس‌زمینه (background)، تومور، بافتی شبیه مایع و بافت پوستی هستند. بنابراین در قسمت‌های بعدی تعداد خوشه‌ها را برابر ۴ در نظر می‌گیریم.

۲.۱



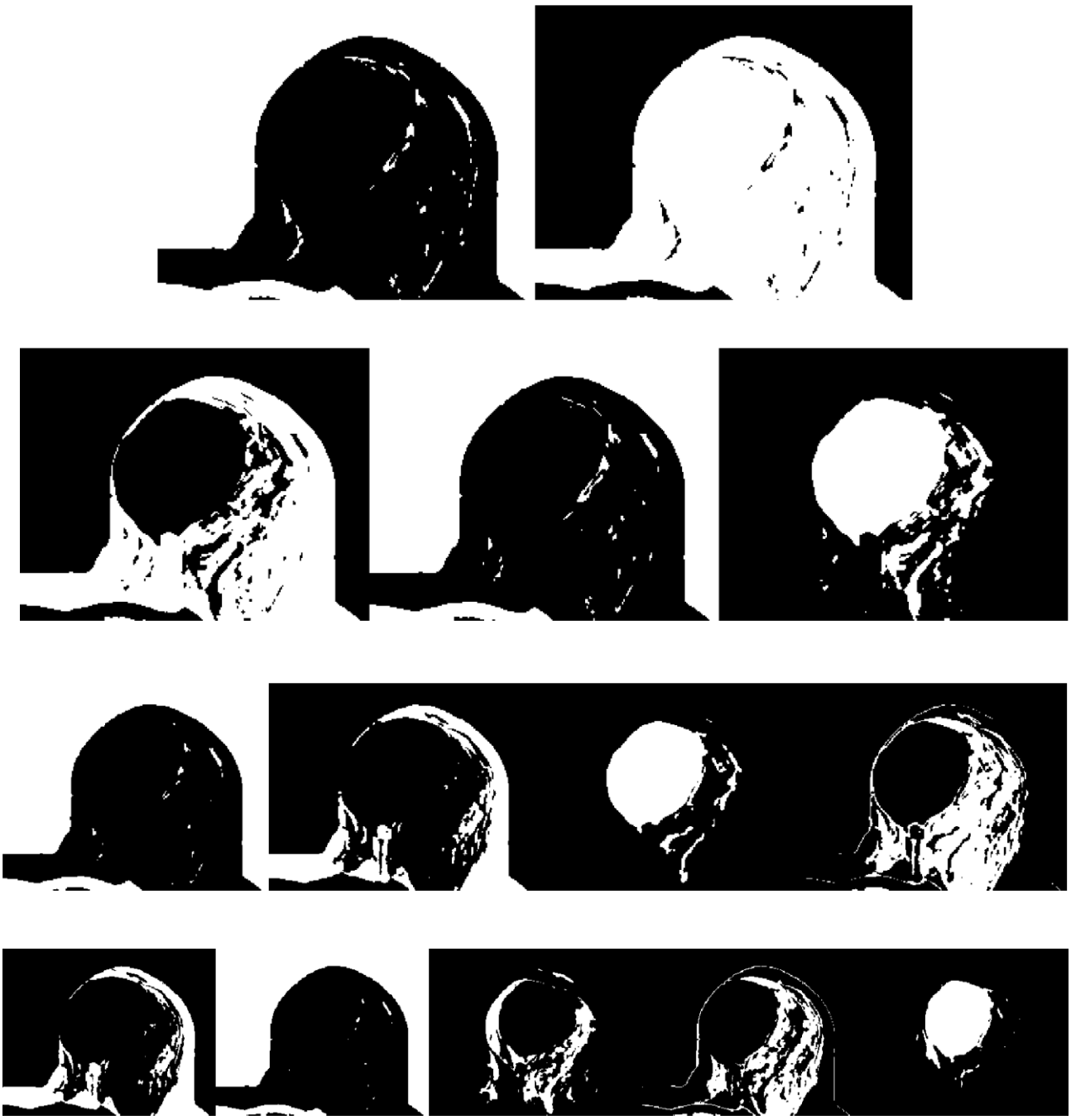




هر چه ضریب ناحیه‌بندی به ۱ نزدیک‌تر باشد، خوشه‌بندی به clustering عادی نزدیک‌تر می‌شود به نحوی که به ازای ضریب ۱.۱ عملا فرقی بین FCM و kmeans دیده نمی‌شود و نقشه احتمال خیلی به حالت باینری ۰ و ۱ نزدیک است. به ازای ضرایب بزرگ‌تر مقادیر تعلق کوچکتر از ۱ بیشتر رخ می‌دهد و یک پیکسل می‌تواند همزمان به چند خوشه تعلق داشته باشد و نقشه احتمال آن از حالت باینری فاصله می‌گیرد و همانطور که مشاهده می‌شود رنگ‌ها نرم‌تر می‌شوند. این بیانگر خوشه‌بندی نرم است.

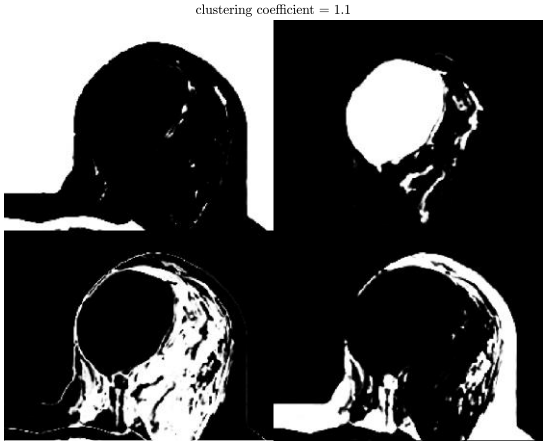
ملاحظه می‌شود در تمامی حالات چهار بافتی که در بخش اول توضیح دادیم یافت شده‌اند.

۳.۱

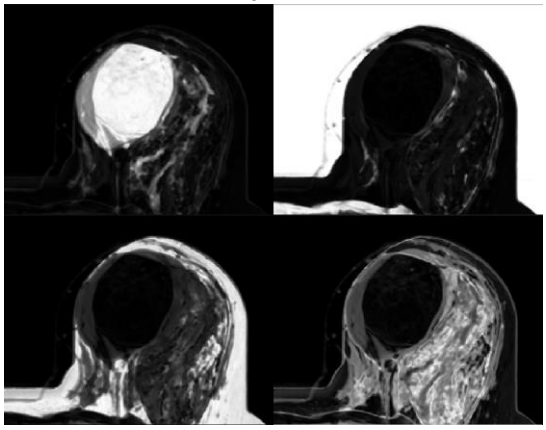


با دو خوشه تنها background و foreground جدا شده‌اند. با سه خوشه، بافت شبیه مایع و بافت پوستی به عنوان یک خوشه شناسایی شده‌اند. با پنج خوشه مشاهده می‌کنیم که بافت شبیه مایع به دو بخش تقسیم شده است که بخشی از آن در خوشه دوم (از راست) و بخش دیگر در خوشه سوم (از راست) قرار دارد. بنابراین انتخاب بهینه کماکان همان ۴ خوشه است.

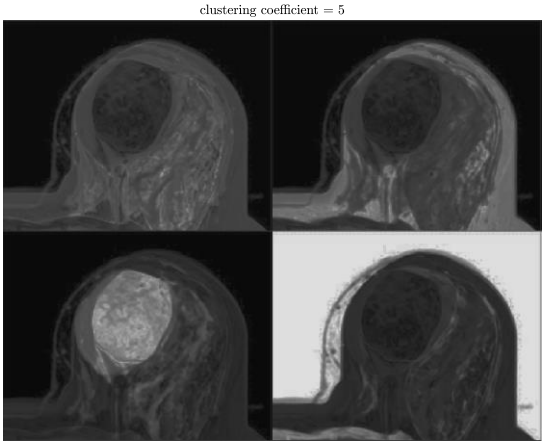
clustering coefficient = 1.1



clustering coefficient = 2

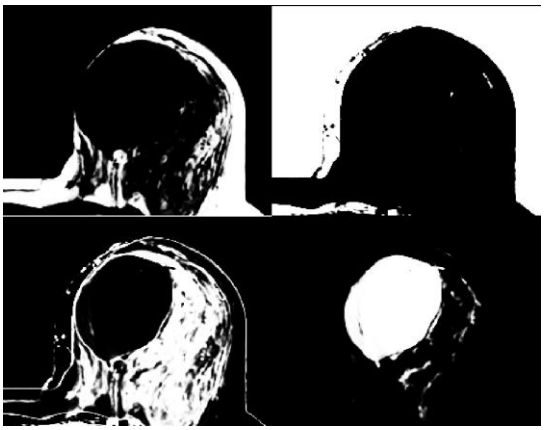


clustering coefficient = 5

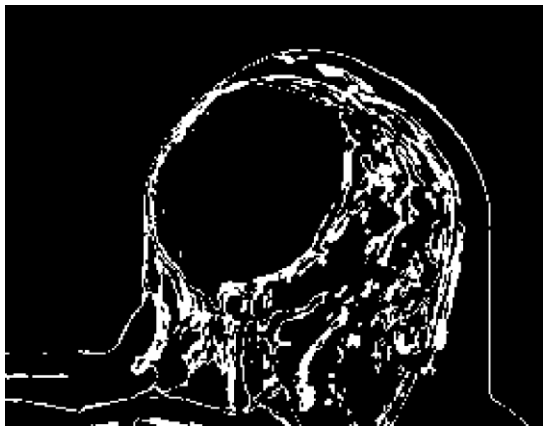


نتایج این روش با حالت قبل تقریبا یکسان است اما تفاوت آن‌ها در تعداد iteration لازم برای همگرایی است که اختلاف فاحش بین این دو روش را نشان می‌دهد. در روش قبل که از حالت اولیه تصادفی استفاده می‌کردیم برای ضرایب ۱.۱، ۲ و ۵ به ترتیب ۳۹، ۴۴ و ۷۰ iteration برای همگرایی نیاز داشتیم اما با انتخاب هوشمندانه‌تر برای حالت اولیه تعداد iteration ها به ترتیب به ۲۷، ۲۶ و ۴۷ تقلیل می‌یابد.

۴.۱

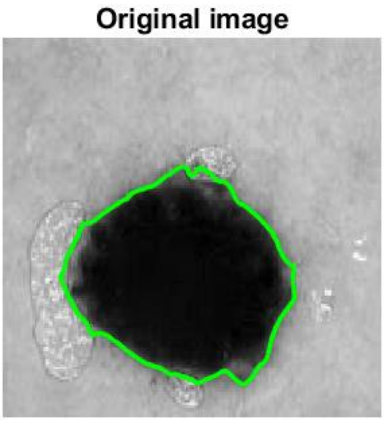


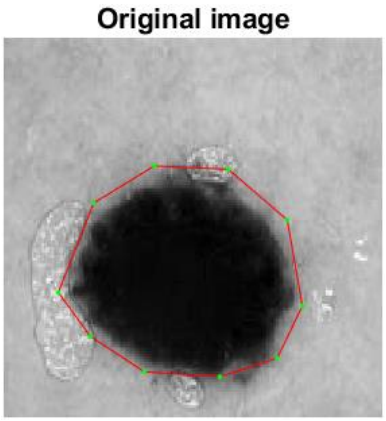
۵.۱ وقتی اثر حجم جزئی داریم، یعنی بعد از اینکه با FCM خوشه‌بندی را انجام می‌دهیم، برای بعضی پیکسل‌ها احتمال تعلق به خوشه‌ها برای ۲ خوشه یا بیشتر مقدار قابل توجهی است. معیار را به این صورت تعریف می‌کنیم که به ازای هر پیکسل، ابتدا احتمال تعلق‌ها را مرتب (sort) می‌کنیم و بعد نسبت بزرگ‌ترین و کوچک‌ترین احتمال تعلق را پیدا می‌کنیم. اگر این نسبت از یک آستانه کوچکتر باشد، آن پیکسل حجم جزئی است. هر چه ترشهولد را کوچکتر انتخاب کنیم، درجه سخت‌گیری بیشتر می‌شود و باید توجه داشت که ضریب خوشه‌بندی را به گونه‌ای انتخاب کنیم که به حد کافی بزرگ باشد تا نواحی حجم جزئی را تشخیص دهد و همه پیکسل‌ها را متعلق به تنها یک دسته ندانیم. از طرفی ضریب خوشه‌بندی نباید خیلی بزرگ باشد تا حتی نواحی که یقینا به یک خوشه تعلق دارند، بصورت حجم جزئی طبقه‌بندی شوند. مقدار این ضریب را برابر ۵ قرار می‌دهیم و آستانه را برابر ۱.۱ در نظر می‌گیریم. نتیجه بصورت زیر شد:

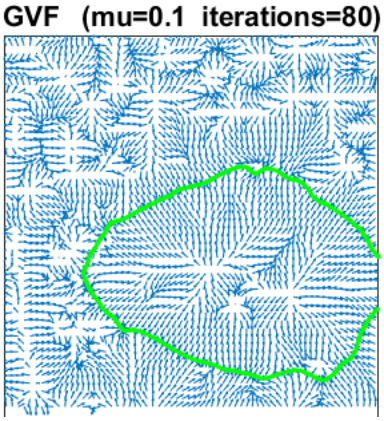


۲ روش‌های GVF و Basic Snake:

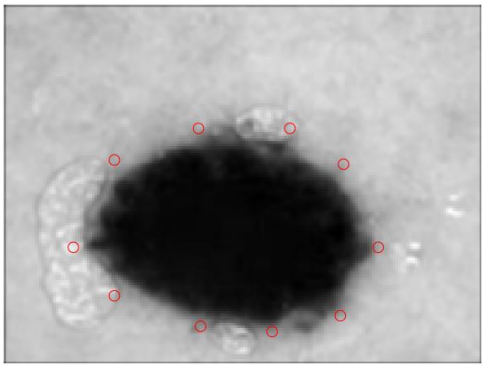
۲-۱ nevus: روش GVF:

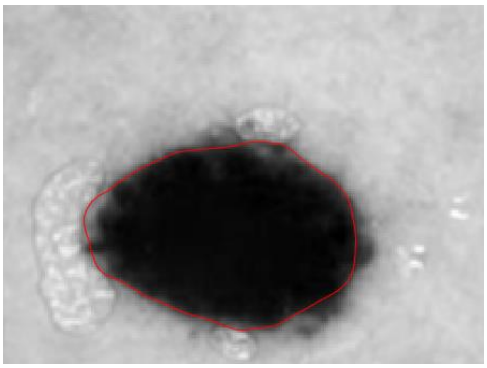




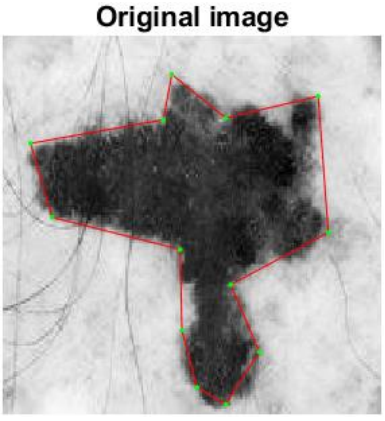


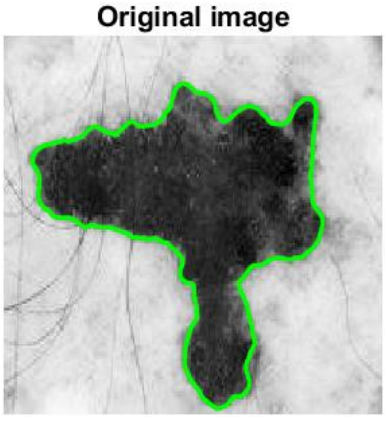
روش Basic Snake:

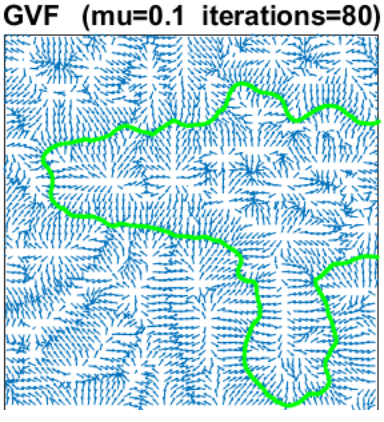




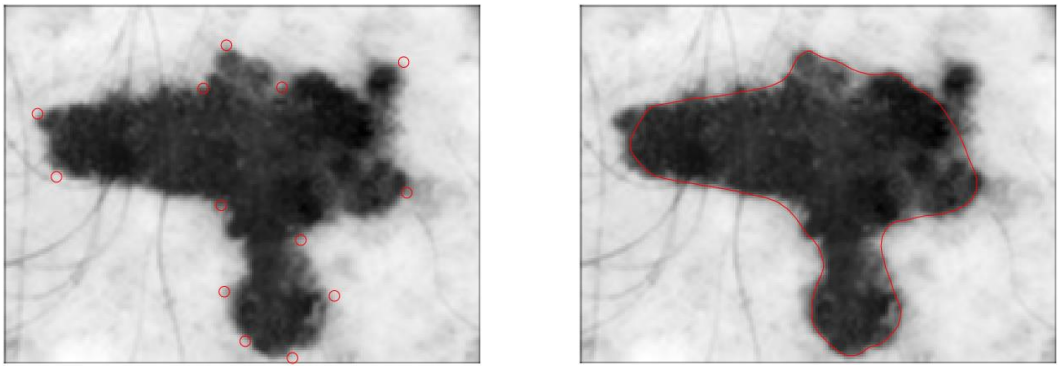
melanoma: روش GVF:







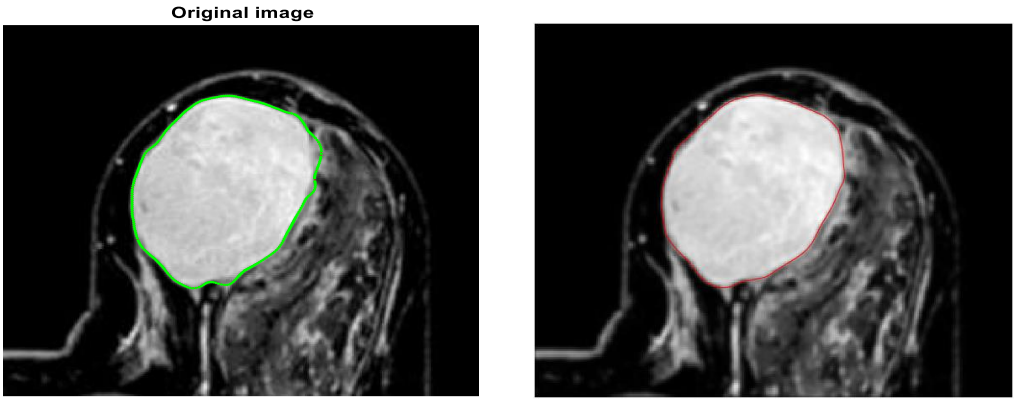
روش Basic Snake:



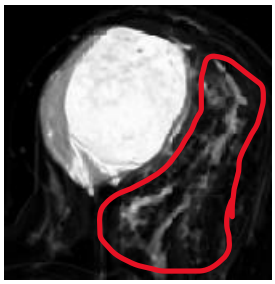
تنها پارامتری که در تولباکس GVF تغییر دادیم پارامتر α بود که مقدار آن را برابر ۲ گذاشتیم. در تولباکس Basic Snake مقدار پارامترهای W(Eline)، W(Eedge) را به ترتیب برابر ۰.۱ و ۰.۶ قرار دادیم و مابقی پارامترها را بدون تغییر گذاشتیم.

ملاحظه می‌شود که دو روش GVF و Basic Snake برای ضایعه nevus تقریبا مشابه عمل کرده‌اند. زیرا شکل ضایعه به نسبت ساده است. با وجود این عملکرد GVF کمی بهتر است زیرا هیچ بخشی از ضایعه را از دست نداده است اما Basic Snake برخی بیرون‌زدگی‌ها را در جنوب شرقی ضایعه از دست داده است. تفاوت این دو روش به خوبی در مورد ضایعه melanoma مشخص است. ملاحظه می‌کنیم با تعداد نقاط اولیه یکسان، روش GVF عملکرد به مراتب بهتری دارد و روش Basic Snake (با تنظیم دقیق پارامترها) در ناحیه‌بندی نواحی که در آن‌ها curvature زیاد است دچار مشکل شده است و خمی هموار به دست آورده است. این مشکل با زیاد کردن تعداد نقاط اولیه و دقت بهتر در انتخاب مکان آن‌ها قابل حل می‌باشد که نیاز این روش را به یک کاربر متخصص نشان می‌دهد در حالیکه با استفاده از GVF از نقاط کمتر و دقت به مراتب پایین تری برای انتخاب نقاط استفاده کردیم.

۲.۲



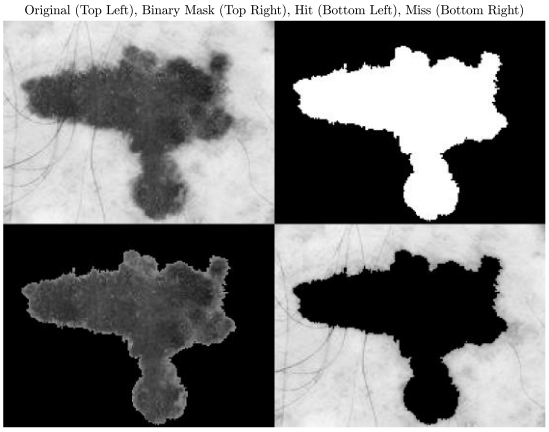
این دو روش به وضوح عملکرد بسیار بهتری برای شناسایی تومور دارند. روش FCM برخی نواحی را در عکس به اشتباه به جای تومور شناسایی می‌کرد که در زیر آن را نشان داده ایم:

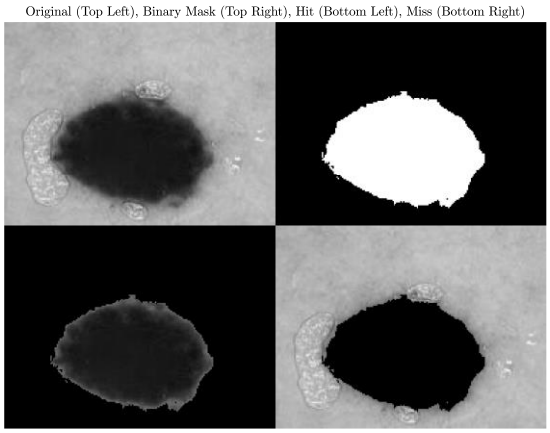


بطور مثال نواحی مشخص شده با رنگ قرمز مربوط به تومور نیستند و با استفاده از روش FCM تومور تشخیص داده شده‌اند. این نواحی در روش GVF و Basic Snake به عنوان تومور شناسایی نمی‌شوند. در روش‌های GVF و Basic Snake بر خلاف روش FCM نمی‌توان نواحی جدا از هم اما متعلق به یک کلاس را شناسایی کرد. زیرا این روش‌ها یک خم بسته را در خروجی تحویل می‌دهند که این مزیت روش‌های pixel classification مانند FCM را نشان می‌دهد که برای چنین کاربردهایی قابل استفاده هستند.

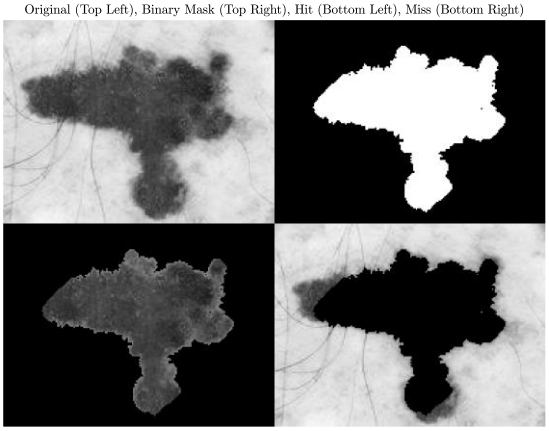
۳ روش chan-vese:

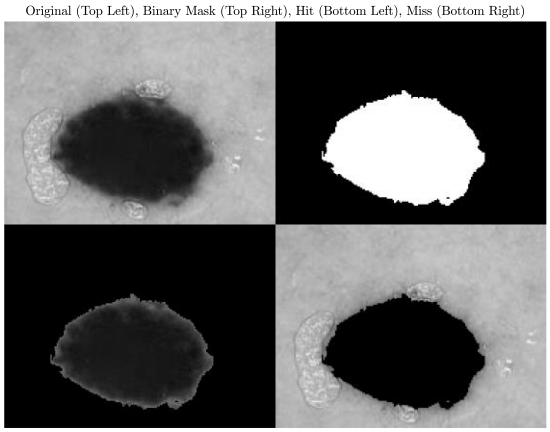
۳-۱: ایجاد ماسک توسط کاربر:



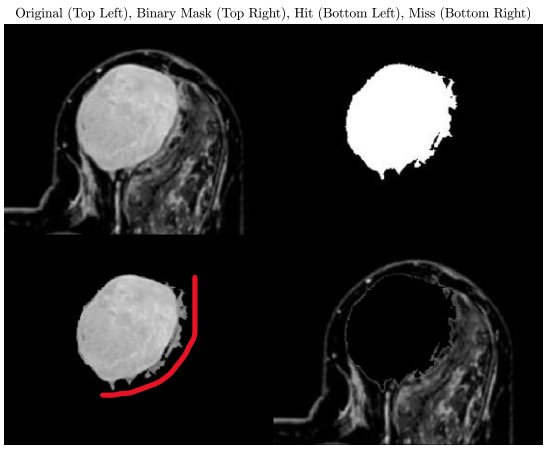


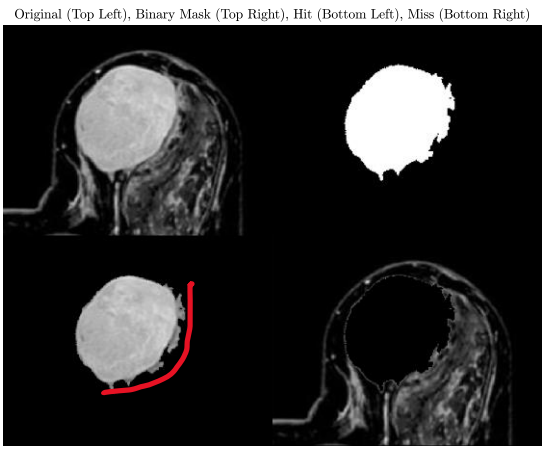
انتخاب یک نقطه توسط کاربر (ماسک ۹×۹):





۳-۲





در روش سمت چپ مرکز مربع ۹×۹ را تعیین کردیم و در سمت راست مرز ناحیه را رسم کردیم. به نظر می‌رسد روش chan-vese در تشخیص ناحیه تومور برخی قسمت‌ها را به اشتباه تومور حساب کرده است (نواحی قرمز رنگ). یعنی FP آن به نسبت

روش‌های GVF و Basic-Snake بیشتر است و از روش FCM کمتر است. اما مزیت این روش نسبت به روش‌های GVF و Basic-Snake خودکار بودن آن است. در دو روش قبلی می‌بایست پارامترهای زیادی را tune می‌کردیم تا بتوانیم به جواب نسبتا خوبی برسیم. بنابراین بطور خلاصه داریم:

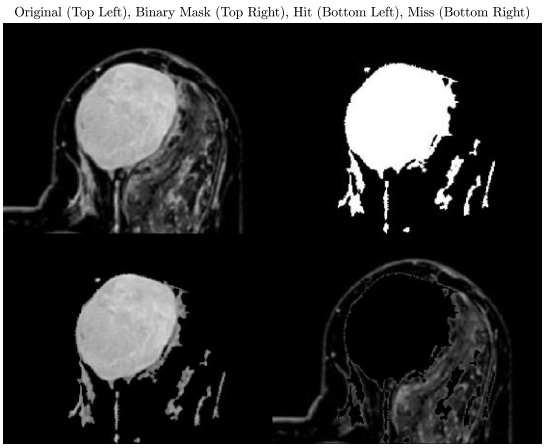
از نظر دقت:

$$FCM < chan - vese < Basic\ Snake < GVF$$

از نظر خودکار بودن:

$$GVF < Basic\ Snake < chan - vese < FCM$$

برای خودکار کردن این فرآیند، با توجه به اینکه تعداد خوشه‌ها را می‌دانیم (۴تا) و هم‌چنین می‌دانیم خوشه مورد نظر ما از مابقی خوشه‌ها روشن‌تر است، می‌توانیم از الگوریتم otsu برای پیدا کردن آستانه‌های مناسب با هر یک از خوشه‌ها استفاده کرد و در نهایت با آستانه‌گذاری روی تصویر با استفاده از بزرگ‌ترین آستانه به دست‌آمده برای خوشه مورد نظر (که از مابقی خوشه‌ها روشن‌تر است)، ماسک اولیه تصویر را به دست می‌آوریم و مانند قبل عمل می‌کنیم.



نتیجه این روش را در تصویر بالا نشان داده‌ایم که ملاحظه می‌شود کیفیت خوشه‌بندی پایین آمده است. که البته مورد انتظار بود زیرا هر چه فرآیند خوشه‌بندی را خودکارتر کنیم از کیفیت آن کاسته می‌شود. در بخش‌های قبلی نیز نمونه آن را دیدیم. جایی که با انتخاب یک نقطه درون ناحیه کیفیت خوشه‌بندی به مراتب پایین‌تر از حالتی بود که در آن مرز را انتخاب می‌کردیم.