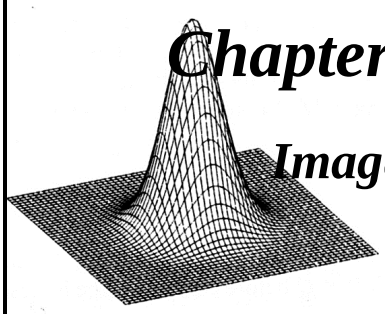


Chapter 7

Image Segmentation

بخش بندی تصویر

پیش نمایش(*preview*)

در فصل قبل کار را با روشهای پردازش تصویر آغاز کردیم. داده‌های ورودی و خروجی این روش تصاویری هستند که در روشی که داده‌های ورودی آن تصویر هستند به کار برده می‌شوند. ولی داده‌های خروجی خصوصیاتی هستند که از آن تصاویر استخراج می‌شوند. بخش بندی گام مهم دیگری در این جهت است.

در روند بخش بندی تصویر به ناحیه‌ها یا اجزای تشکیل دهنده خود تقسیم می‌شود. سطح بخش بندی بستگی به مسئله مورد نظر دارد. وقتی اشیای مورد نظر در برنامه تفکیک می‌شوندبخش بندی باید متوقف شود. مثلاً وقتی روند خودکار سازی امور در امور الکترونیکی بررسی می‌شود، مسئله تحلیل تصاویر محصولات با هدف تعیین وجود یا نبود برخی ناهنجاری‌های خاص همچون مولفه‌های مفقود شده یا مسیرهای قطع شده است. تداوم بخش بندی فراتر از مرحله شناسایی این عناصر فایده‌ای ندارد.

بخش بندی تصاویر مهم یکی از دشوارترین کارها در پردازش تصاویر است. میزان دقت بخش بندی برای تعیین موفقیت یا شکست روشهای تحلیل رایانه‌ای به کار برده می‌شود. به همین دلیل است که برای بهبود این بخش بندی باید دقت زیادی کرد. در برخی موارد همچون برنامه‌های بازدید امور صنعتی گاهی باید محیط مورد نظر مهار شود. در برخی موارد همچون بکارگیری حسگرها از راه دور کاربر فقط می‌تواند با انتخاب حسگرهای تصویربردار کنترل خود را اعمال کند.

الگوریتم‌های بخش بندی تصویرهای تک رنگ مبتنی بر یکی از دو خصوصیت اصلی مقادیر شدت رنگ یعنی گسستگی(disconTinuiTy) و تشابه (similariTy) هستند. در مقوله اول، مثلاً تصویر بر اساس تغییرات ناگهانی شدت رنگ بخش بندی می‌شود. روش اصلی در مقوله دوم بر اساس تقسیم بندی تصویر به ناحیه‌هایی است که طبق یک سری معیارهای از پیش تعیین شده مشابه هستند.

در این فصل برخی از روشهای دو مقوله فوق را با توجه به کاربرد آنها در تصاویر تک رنگ بررسی می‌کنیم .

این کار را با روشهای مناسب برای کشف گسستگی‌ها همچون نقطه‌ها، خطها و لبه‌ها به کار برده‌ایم. آشکارسازی لبه‌ها مدتها نکته اصلی الگوریتم بخش بندی بوده است. غیر از آشکارسازی لبه‌ها بخشهای خطدار لبه با استفاده از روشهای مبتنی بر تبدیل Hough نیز انجام می‌شوند. مقدمه شگردهای آستانه‌یابی بعد از مبحث آشکارسازی لبه‌ها قید شده است. آستانه‌یابی یکی از روشهای اصلی بخش بندی است که مخصوصاً در برنامه‌هایی که سرعت مهم است بسیار کاربرپسند است. بعد از مبحث آستانه‌یابی روشهای بخش بندی مبتنی بر ناحیه‌ها قرار دارند. این بخش را با مبحث مطالعه علمی شکلهای بخش بندی به نام بخش بندی شیبدار(waTershed segmenTaTion) است. این روش بسیار جذاب است زیرا ناحیه‌ها و چهارچوب تعریف شده‌ای در آن ایجاد می‌شود که آگاهی داشتن از تصاویر ایجاد شده در یک برنامه خاص برای بهبود نتایج بخش بندی به کار برده می‌شود.

۷.۱نقطه، خط و آشکارسازی لبه‌ها(PoinT , Line , and Edge DeTecTion)

در این بخش شگردهای آشکارسازی سه نوع گسستگی اصلی شدت رنگ یعنی نقطه‌ها، خطها و لبه‌ها را در تصاویر دیجیتالی بحث می‌کنیم. متداول‌ترین روش برای جستجوی گسستگی‌ها اجرای Mask روی تصاویر به روشهای تشریح شده در بخش‌های ۳.۴ . ۳.۵ است. برای اجرای این روش در نقابهای ۳*۳ مجموع محصولات ضریبها با شدت رنگ مندرج در Mask محاسبه می‌شوند. یعنی واکنش Rدر Mask در هر نقطه از تصویر با رابطه زیر مشخص می‌شود:

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9$$
$$= \sum_{i=1}^{9} w_i z_i$$
$$|R| \geq T$$

در اینجا z شدت عناصر تصویری مربوط به ضریب Mask wi است. واکنش Mask با توجه به کانون آن تعریف می‌شود.

۷.۱.۱. نقطه یابی(PoinT DeTecTion)

یافتن نقاط مجزا در ناحیه‌هایی که شدن رنگ (تقریباً) ثابت است ساده است. با استفاده از Mask تصویر ۷.۱ نقطه مجزا در محل استقرار Mask دیده می‌شود.

$$|R| \geq T$$

−1	−1	−1
−1	8	−1
−1	−1	−1

تصویر ۷.۱: تقابی برای نقطه یابی

T آستانه مثبت است. نقطه یابی در نرم افزار MATLAB با استفاده از تابع imfilTer انجام می‌شود و می‌توان از Mask تصویر ۷.۱ یا سایر Mask های مشابه استفاده کرد. حداکثر واکنش Mask در زمانی است که روی یک نقطه مجزا قرار بگیرد، و واکنش آن در جاهایی که شدت ثابت است صفر است.

اگر مقدار T مشخص باشد، روش نقطه یابی با فرمان زیر اجرا می‌شود:

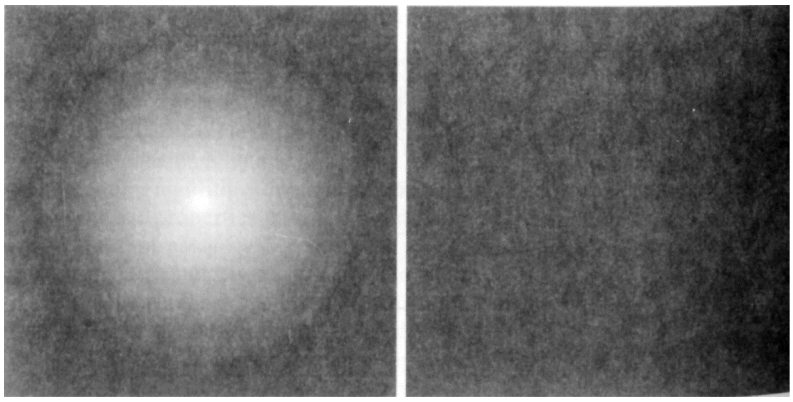
```
>> g = abs(imfilTer(bouble(f), w)) >= T;
```

در اینجا (f)تصویر ورودی است، w یک Mask مناسب نقطه یاب است، (مثلاً Mask تصویر ۷.۱) و g تصویر حاصل از آن است. در بخش ۳.۴.۱ گفتیم که imfilTer خروجی را به ورودی تبدیل می‌کند بنا بر این بنا بر این برای جلوگیری از بریده شدن زودهنگام مقادیر در صورتی که ورودی از نوع uinT8 باشد، از(double(f استفاده می‌شود. علتش آن است که abs در عملیات خود داده‌های حاوی عدد صحیح را قبول نمی‌کند. تصویر خروجی g از نوع منطقی است و مقادیر آن ۰ و ۱ هستند. اگر مقدار T مشخص نشده باشد، کمیت آن بر اساس نتایج فیلتر شده انتخاب می‌شود، در این صورت، رشته فرمان قبل به ۳ مرحله تجزیه می‌شود: (۱) محاسبه تصویر فیلتر شده abs imfilTer(double(f),w) (۲) یافتن مقدار T با استفاده از داده‌های تصویر فیلتر شده (۳) مقایسه تصویر فیلتر شده با T. مثال این روش در همین جا قید شده است.

مثال ۷.۱: نقطه یابی:

در عکس ۷.۲.(a) تصویری با نقاط مشکی نیمه نامرئی در قسمت خاک ربع شمال شرقی دیده می‌شود. اگر f نمایانگر این تصویر باشد، محل نقطه به شرح زیر پیدا می‌شود:

```
>> w = [-1 -1 -1; -1 8 -1; -1 -1 -1];
>> T = max(g(:));
>> g = g >= T;
>> imshow(g)
```



تصویر ۷.۲: تصویر مقیاس خاک با نقاط مشکی شبه نامرئی در ناحیه خاکستری ربع شمال شرقی

(b) تصویر نمایانگر نقطه آشکار شده (نقطه بزرگ شده تا به راحتی استفاده شود)

اگر T حداکثر مقدار تصویر فیلتر شده g باشد، در این صورت، اگر نقاط موجود در gطوری پیدا شود که g<=T باشد، در این صورت، نقاط دارای بزرگترین واکنش را پیدا می‌کنیم. فرض بر آن است که کلیه این نقاط مجزا هستند و در یک پس زمینه تقریباً ثابت قرار دارند. آزمایش روی T با استفاده از عملگر =< برای ایجاد هماهنگی در نشانه‌گذاری به کار برده شد. از آنجائی که T حداکثر مقدار موجود در g است، هیچ نقطه‌ای در g نیست که مقدار آن بیشتر از T باشد. در تصویر ۷.۲(b) یک نقطه مجزا قرار داشت که وقتی T را روی max g تنظیم می‌کردیم، شرایط g>=T تحقق می‌یافت.

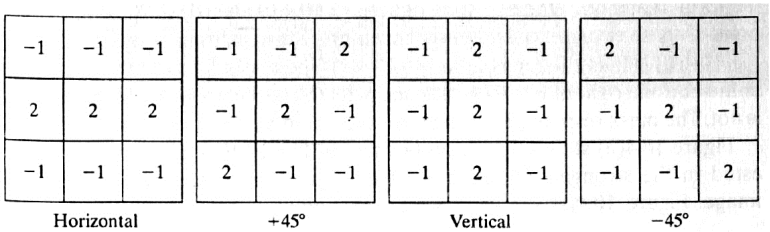
یک روش دیگر نقطه یابی پیدا کردن نقاط همجوار اندازه m*n است که در این حالت اختلاف بین حداکثر و حداقل مقادیر از مقدار خاصی از T بیشتر است. این روش با استفاده از تابع ordfilT2 که در بخش ۳.۵.۲ معرفی کردیم اجرا می‌شود.

```
>> g = imsubtract(ordfilt2(f, m*n, ones(m, n)), …
                               ordfilt2(f, 1, ones(m, n)));
>> g = g >= T;
```

به آسانی مشخص می‌شود که با انتخاب T=max(g(:)) همان نتیجه تصویر ۷.۲. b به دست آورده می‌شود. استفاده از فرمول قبل بسیار انعطاف پذیر تر از بکارگیری Mask در تصویر ۷.۱ است. مثلاً اگر بخواهیم اختلاف بین حداکثر مقدار و مقدار پیکسلهای بعد از آن را در یک ناحیه محاسبه کنیم، عدد ۱ را در سمت راست عبارت قبل با m*n-1 قرار می‌دهیم. سایر شکلهای متغیر این فرمول به شکلی مشابه ایجاد می‌شوند.

۷.۱.۲. آشکارسازی خط(Line Detection)

مسئله پیچیده بعدی خط یابی است. نقابهای تصویر ۷.۳ را در نظر بگیرید. اگر Mask اول حول یک تصویر گردانده شود، به خطوطی که جهت گیری افقی (با ضخامت یک پیکسل دارند) واکنش شدیدتری نشان می‌دهد. اگر پس زمینه ثابت باشد، وقتی خط از ردیف وسط Mask عبور کند، حداکثر واکنش ایجاد می‌شود. Mask دوم تصویر ۷.۳ به خطوطی با جهتگیری مثبت ۴۵ درجه بهترین واکنش را نشان می‌دهد. به همین ترتیب Mask سوم به خطوط عمودی و Mask چهارم به خطوط دارای جهت منفی ۴۵ درجه واکنش نشان می‌دهد. توجه داشته باشید که جهت هر Mask با ضریبی بزرگتر (یعنی ۲ برابر) سایر جهتها سنجیده می‌شود. مجموع ضریب هر Mask صفر است یعنی در نواحی که شدت رنگ ثابت باشد، واکنش Mask صفر است.



تصویر ۷.۳: نقابهای خط یاب

اگر فرض کنیم R_1, R_2, R_3, R_4 واکنش‌های نقابها در تصویر ۷.۳ از چپ به راست باشند، مقادیر ریاRها در معادله بخش قبل نشان داده شده است. فرض کنید این چهار Mask به صورت انفرادی در تصاویر اجرامی‌شوند. اگر در نقطه خاصی از تصویر $|R_i|>|R_j|$ برای تمام موارد $j\#i$ صادق باشد، در این صورت، آن نقطه مربوط به خطی در جهت Mask i می‌شود مثلاً اگر در نقطه‌ای در تصویر $|R_i|>|R_j|$ و $j=2, 3, 4$ باشد، آن نقطه بیشتر از خطوط افقی ارتباط دارد. ممکن است بخواهیم خطوط را به شکلی خاص آشکار کنیم. در این صورت، Mask مربوط به آن جهت را انتخاب می‌کنیم و آستانه داده‌های خروجی آن را همچون معادله بخش قبل پیدا می‌کنیم. به عبارت دیگر اگر بخواهیم کلیه خطوط یک تصویر را در جهت تعریف شده در یک Mask خاص آشکار کنیم، ابتدا Mask را در تصویر اجرا می‌کنیم و بعد قدر مطلق نتیجه را آستانه یابی می‌کنیم. نقاط باقیمانده قویترین واکنشها هستند که در خطوطی که ضخامت یک پیکسل دارند مطابق با جهت تعریف شده توسط Mask هستند. مثال زیر نمایانگر این روش است:

مثال ۷.۲: آشکارسازی خطوط در یک جهت خاص

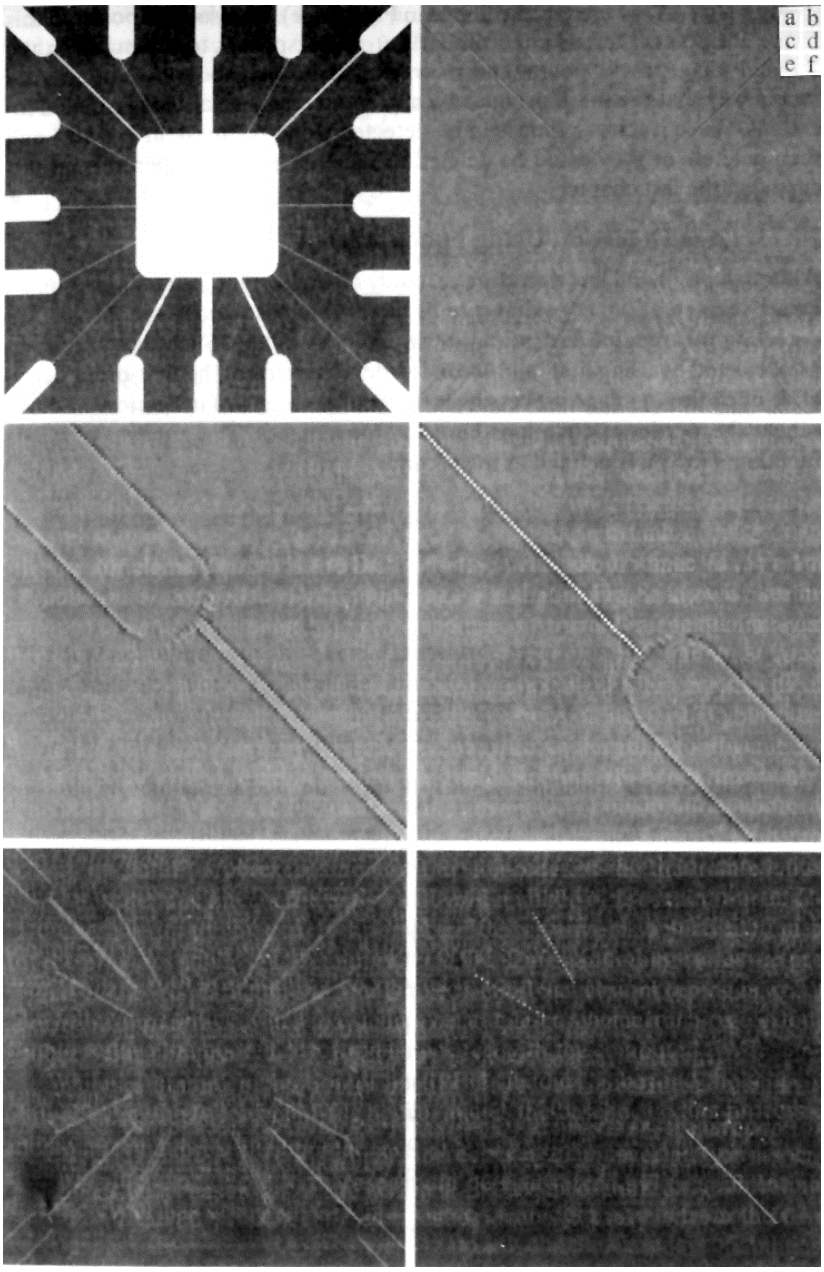
در تصویر ۷.۴.(a) بخش دیجیتالی دودویی یک Mask متصل به سیم در مدار الکترونیکی دیده می‌شود. اندازه تصویر ۴۸۶*۴۸۶ پیکسل است. فرض کنید می‌خواهیم کلیه خطوط با ضخامت یک پیکسل را در جهت منفی ۴۵ درجه پیدا کنیم. برای این کار از آخرین Mask تصویر ۷.۳ استفاده می‌کنیم. تصاویر ۷.۴(b) الی (f)با استفاده از فرمان زیر ایجاد شدند، و f تصویر داخل ۷.۴.(a) است.

```
>> w = [2 -1 -1; -1 2 -1; -1 -1 2];
>> g = imfilter(double(f), w);
>> imshow(g, [ ])             % Fig. 10.4(b)
>> gtop = g(1:120, 1:120);
>> gtop = pixeldup(gtop, 4);
>> figure, imshow(gtop, [ ])   % Fig. 10.4(c)
>> gbot = g(end-119:end, end-119:end);
>> gbot = pixeldup(gbot, 4);
>> figure, imshow(gbot, [ ])   % Fig. 10.4(d)
>> g = abs(g);
>> figure, imshow(g, [ ])      % Fig. 10.4(e)
>> T = max(g(:));
>> g = g >= T;
 % Fig. 10.4(f)   >> figure, imshow(g)
```

سایه‌های تیره‌تر از پس زمینه خاکستری در تصویر ۷.۴. b مطابق با مقادیر منفی هستند. دو بخش اصلی یکی در بالا و سمت چپ و دیگری در پایین و سمت راست در جهت منفی ۴۵ درجه هستند. در تصاویر ۷.۴.(c)و (d)بخش‌های بزرگنمایی شده نشان داده شده است. توجه داشته باشید که خط مستقیم در این بخش از تصویر ۷.۴(d) بسیار بزرگتر از بخش موجود در ۷.۴.(c)است. علت آن است که جزء موجود در انتها و سمت راست تصویر ۷.۴.(a) یک پیکسل ضخامت دارد در صورتی که تصویر بالا سمت چپ اینطور نیست. واکنش Mask در

جزء دارای ضخامت یک پیکسل قویتر است. قدر مطلق تصویر b.۷.۴در ۷.۴.ایی دیده می‌شود. از آنجائی که می‌خواهیم قویترین واکنش را بررسی کنیم، فرض می‌کنیم T معادل حداکثر مقدار موجود در این تصویر است.

در تصویر ۷.۴.(f)نقاطی که مقادیر آنها شرط g>=T را برقرار می‌کند با رنگ سفید مشخص شده‌اند.



تصویر ۷.۴: (a)) تصویری از Mask متصل به سیم (b) نتایج پردازش با آشکارساز منفی ۴۵ درجه (c) بزرگنمایی ناحیه چپ بالا b(d) بزرگنمایی بخش راست پایین b(e) کلیه نقاط سفید که مقادیر آنها شرط g>=T را برقرار می‌کند. (نقاط (f)اندکی بزرگ شده‌اند تا رویت آنها راحتتر باشد

gتصویر داخل عکس (e.۷.۴) ااست. نقاط مجزا در این تصویر نقاطی هستند که واکنش‌های نسبتاً قوی به Mask نشان داده‌اند. این نقاط و قسمت‌های همجوار آنها در تصویر اصلی طوری قرار گرفته‌اند که Mask حداکثر واکنش را در نقاط مجزا ایجاد کرده است. این نقاط مجزا را می‌توان با استفاده از عکس ۷.۱آشکار و بعد حذف کرد و یا می‌توان آنها را با استفاده از عملگرهای تغییر شکل که در فصل آخر بحث شدند حذف کرد.

۷.۱.۳. آشکارسازی لبه‌ها با استفاده از تابع edge

(Edge Detection Using Function edge)

گرچه خط یابی و نقطه یابی مبحثهای مهمی در قطعه بندی تصویر هستند، ولی آشکارسازی لبه‌ها متداولترین روش برای پی بردن به گسستگی‌های معنی‌دار مقادیر شدت است. این گسستگی‌ها با استفاده از مشتقهای درجه اول و دوم مشخص می‌شوند. اولین مشتق درجه یک منتخب در پردازش تصویر ضریب شیب است که در بخش ۶.۶.۱. تعریف شده. در اینجا معادلات مربوطه تکرار شده‌اند. شیب تابع ۲ بعدی $F(x, y)$ با بردار زیر تعریف می‌شود:

?

بزرگی این بردار(vector) به شرح زیر است:

?

با حذف ریشه دوم مقدار نزدیک به این کمیت برای ساده کردن محاسبات به دست می‌آید

?

و یا از قدر مطلق استفاده می‌شود:

?

این مقادیر تقریبی مشتق هستند یعنی در ناحیه‌هایی که شدت رنگ ثابت است صفر هستند و مقدار آنها متناسب با تغییر شدت رنگ در ناحیه‌هایی است که مقادیر پیکسل آنها متغیر است. معمولاً حداکثر مقدار این شیب یا مقادیر نزدیک به آن را با عبارت ضریب زاویه بیان می‌کنند.

یکی از خصوصیات اصلی بردار ضریب زاویه آن است که اشاره به حداکثر تغییرات (f)در مختصات (x, y) دارد زاویه‌ای که حداکثر مقدار تغییر در آن روی می‌دهد به شرح زیر است:

?

یکی از نکات اصلی نحوه برآورد مشتقهایG_y ,G_x به صورت دیجیتالی است. روشهای گوناگون استفاده شده توسط تابع h در این بخش بحث شده است.

برای محاسبه مشتقهای درجه دوم در پردازش تصویر از لاپلاسی استفاده می‌شود که در بخش ۳.۵.۱ معرفی کردیم. یعنی لاپلاس یک تابع ۲ بعدی f (x, y) از مشتقهای درجه دوم به شرح زیر تشکیل می‌شود:

?

لاپلاس به ندرت به تنهایی برای آشکارسازی لبه به کار برده می‌شود و بسیار نسبت به پارازیتها حساس است، بزرگی آن باعث ایجاد لبه‌های دوگانه می‌شود و نمی‌تواند جهت لبه را آشکار کند. ولی همان طور که در این بخش گفتیم، وقتی لاپلاس با سایر شگردهای آشکارسازی لبه به کار برده شود مکمل قدرتمندی است. گرچه لبه‌های دوگانه مانع از آن می‌شوند که مستقیماً برای آشکارسازی لبه به کار برده شود، ولی از این خصوصیت می‌توان در یافتن محل لبه‌ها نیز استفاده کرد.

با توجه به مبحث قبل، نکته اصلی در آشکارسازی لبه پیدا کردن محلهایی در تصویر است که شدت رنگ با استفاده از یکی از استانداردهای زیر به سرعت تغییر کند.

۱. مکانهایی را پیدا کنید که اولین مشتق شدت رنگ بیشتر از سطح آستانه باشد.

۲. جاهایی را پیدا کنید که دومین مشتق شدت رنگ مقطع صفر داشته باشد.

تابع edge در IPT چندین برآورد کننده مشتق بر اساس معیارهای فوق دارد. در برخی از این تخمینها می‌توان مشخص کرد که آیا آشکارساز لبه نسبت به لبه‌های افقی، عمودی و یا هر دو حساس است یا خیر. ترکیب کلی این تابع به شرح زیر است:

```
[g, t] = edge(f, 'method', parameters)
```

در اینجا f تصویر ورودی است. meThod یکی از روشهای فهرست بندی شده در جدول ۱ است و ParameTers یکی از مقادیر مشخص تشریح شده در بخش زیر است. g در داده‌های خروجی یک آرایه منطقی است و یکها در محلی است که نقاط لبه در f و صفرها در جاهای دیگر شناسایی شده‌اند. پارامتر T اختیاری است و استانه استفاده شده در لبه در آن ارائه شده است تا مشخص شود کدام ضریب زاویه‌ها این قدرت را دارند که نقاط لبه نامیده شوند.

لبه یاب Sobel برای ساختن مقادیر دیجیتالی نزدیک به اولین مشتقهای G_x, G_y از نقابهای تصویر b.۷.۵ است می‌کند. در واقع ضریب زاویه در کانون در این ناحیه توسط آشکارساز Sobel محاسبه می‌شود.

$$
\begin{aligned}
g &= [G_x^{\,2} + G_y^{\,2}]^{1/2} \\
&= \{[(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)]^2 \\
&\quad + [(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)]^2\}^{1/2}
\end{aligned}
$$

Edge Detector	Basic Properties
Sobel	Finds edges using the Sobel approximation to the derivatives shown in Fig. 10.5(b).
Prewitt	Finds edges using the Prewitt approximation to the derivatives shown in Fig. 10.5(c).
Roberts	Finds edges using the Roberts approximation to the derivatives shown in Fig. 10.5(d).
Laplacian of a Gaussian (LoG)	Finds edges by looking for zero crossings after filtering $f(x, y)$ with a Gaussian filter.
Zero crossings	Finds edges by looking for zero crossings after filtering $f(x, y)$ with a user-specified filter.
Canny	Finds edges by looking for local maxima of the gradient of $f(x, y)$. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. Therefore, this method is more likely to detect true weak edges.

جدول ۷.۱: لبه یابهای موجود در تابع edge

لبه یاب (Edge Detection)	خصوصیات اساسی Basic Properties
Sobel	لبه‌ها را با استفاده از نزدیک ساز Sobel با توجه به مشتقهای تصویر ۷.۵.bپیدا می‌کند.
prewitt	لبه‌ها را با استفاده از نزدیک ساز Sobel با توجه به مشتقهای تصویر ۷.۵.(c)پیدا می‌کند.
Roberts	لبه‌ها را با استفاده از نزدیک ساز Sobel با توجه به مشتقهای تصویر ۷.۵.(d)پیدا می‌کند.
لاپلاس گوسی(LOG)	لبه‌ها را با توجه به تقاطع‌های صفر بعد از فیلترسازی f (x, y) با فیلتر گوسی پیدا می‌کند.
مقطع صفر (zero crossing)	لبه‌ها را با جستجوی تقاطعهای صفر بعد از فیلترسازی f (x, y) با فیلتر سفارشی کاربر پیدا می‌کند.
Canny	لبه‌ها را با جستجوی حداکثر مقدار موضعی ضریب زاویه f (x, y) پیدا می‌کند. ضریب زاویه با استفاده از مشتق فیلتر گوسی محاسبه می‌شود. در این روش برای آشکارسازی لبه‌های قوی و ضعیف از ۲ آستانه استفاده می‌شود و فقط در صورتی از لبه‌های ضعیف استفاده می‌کند که با لبه‌های قوی مرتبط باشند. بنا بر این احتمال آشکارسازی لبه‌های ضعیف واقعی در این روش بیشتر است.

اگر g>=T در موقعیت (x, y) درست باشد، در آن صورت آن را پیکسل لبه می‌نامیم و T مقدار استانه خاص است.

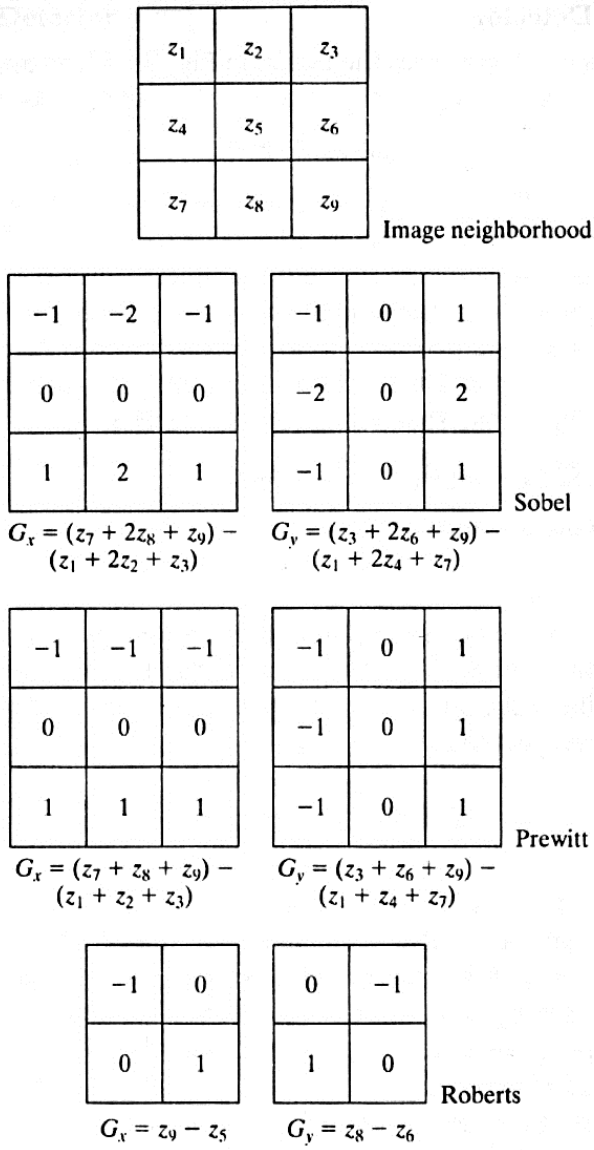
با توجه به بخش ۳.۵.۱ می‌دانیم که برای اجرای لبه یاب Sobel ابتدا تصویر (f)با استفاده از imfilTer با Mask چپ در تصویر b.۷.۵فیلتر می‌شود، سپس (f)با Mask دیگر فیلتر می‌شود، سپس مجذور مقادیر پیکسل هر تصویر فیلتر شده گرفته می‌شود، ۲ نتیجه با هم جمع می‌شود، و ریشه دوم آنها محاسبه می‌شود. این روند در ورودیهای دوم و سوم جدول ۷.۱ نیز روی می‌دهد. تابع edge عملیات قبل را با یک دور فراخوانی تابع انجام می‌دهد، و خصوصیات دیگر همچون پذیرش مقدار آستانه و یا تعیین خودکار مقدار آستانه به آن افزوده می‌شود. تابع edge روشهایی برای آشکارسازی لبه دارد که مستقیماً با imfilTer اجرا نمی‌شوند. ترکیب کلی فراخوانی لبه یاب Sobel به شرح زیر است:

```
[g, t] = edge(f, 'sobel', T, dir)
```

در اینجا f تصویر ورودی است، T سطح آستانه خاص است، و dir نمایانگر جهت لبه‌های کشف شده است که می‌توانند افقی، عمودی و یا هر دو(both) (که حالت پیش فرض است) باشند. همان طور که قبلاً گفتیم g یک تصویر منطقی است که در جاهایی که لبه آشکار شده است مقدار یکها را دارد و در جاهای دیگر صفرها را دارد. پارامتر T در داده‌های خروجی اختیاری است. این همان مقدار استانه‌ای است که توسط تابع edge استفاده می‌شود. اگر مقدار T مشخص شود در این صورت، t = T می‌شود.

ولی اگر مقدار T مشخص نشود و یا تهی باشد، تابع edge مقدار T را بر اساس استانه‌ای که به صورت خودکار تعیین شده است تخصیص می‌دهد و سپس از آن برای لبه یابی استفاده می‌کند. یکی از دلایل اصلی درج T در شناسه خروجی آن است که مقدار اولیه استانه را به دست آوریم. در صورت برقرار بودن یکی از ترکیبهای زیر تابع edge به صورت پیش فرض از آشکارسازی Sobel استفاده می‌کند.

G=edge(f) [g,t]=edge(f)



تصویر ۷.۵: برخی از نقابهای لبه یاب و مشتقهای درجه یکی که اجرا می‌کنند

لبه یاب Prewitt (Prewitt Edge Detector)

لبه یاب Prewitt برای ساختن مقادیر دیجیتالی نزدیک به اولین مشتقهای G_x, G_y از نقابهای تصویر ۷.۵.(c)استفاده می‌کند. ترکیب فراخوانی آن به شرح زیر است:

```
[g, t] = edge(f, 'prewitt', T, dir)
```

مقادیر مشخص این تابع مانند پارامترهای Sobel هستند. اجرای لبه یاب Prewitt از نظر محاسباتی از لبه یاب Sobel آسانتر است ولی معمولاً نتایج پرپارازیت ایجاد می‌کند. (می‌بینید که ضریب مقدار دردو لبه یاب Sobel باعث هموار شدن می‌شود).

لبه یاب Roberts (Rorets Edge Detector)

لبه یاب Roberts برای ساختن مقادیر دیجیتالی نزدیک به مشتقهایG_y ,G_x از نقابهای تصویر ۷.۵.(d) استفاده می‌کند. تابع فراخوانی کلی آن به شرح زیر است:

```
[g, t] = edge(f, 'roberts', T, dir)
```

مقادیر مشخص این تابع همانند پارامترهای Sobel است. لبه یاب Roberts یکی از قدیمیترین لبه یابها در پردازش تصاویر دیجیتالی است و در تصویر ۷.۵.(d) دیده می‌شود که ساده‌ترین نیز هست. این لبه یاب کمتر از سایر موارد در تصویر ۷.۵ استفاده می‌شود. علت تا حدی کاربرد محدود آن است (مثلاً متقارن نیست و نمی‌توان برای آشکارسازی لبه‌هایی که مضرب ۴۵ درجه هستند از آن استفاده کرد). ولی برای راه اندازی سخت افزارهایی استفاده می‌شود که سرعت و سادگی خصوصیت اصلی آنها است.

لاپلاس یک لبه یاب LOG گوسی(Laplacian of a Gaussian (LOG)Detector)

تابع گوسی زیر را در نظر بگیرید:

?

در اینجا $r^2 = x^2 + y^2$ و δ انحراف معیار است. این یک تابع هموارساز است که وقتی با یک تصویر تلفیق شود، آن را تیره می‌کند. مقدار تیره شدن بر اساس a تعیین می‌شود. لاپلاس این تابع (مشتق دوم با توجه به r به شرح زیر است:

?

این تابع را لاپلاس گوسی می‌نامیم چون که مشتق دوم عملکرد خطی دارد. تلفیق تصویر با$\Delta^2 h(r)$ مانند چرخش آن با تابع هموارساز و محاسبه لاپلاس نتیجه است. این خصوصیت اصلی لبه یاب Log است. وقتی $\Delta^2 h(r)$ تلفیق شود دو تاثیر دارد: تصویر را هموار می‌کند (بنا بر این پارازیتها را کاهش می‌دهد)، و لاپلاس را محاسبه می‌کند که یک تصویر دو لبه حاصل می‌شود. برای یافتن لبه‌ها تقاطع‌های صفر بین لبه‌های دوگانه باید مشخص شود.

```
[g, t] = edge(f, 'log', T, sigma)
```

در اینجا Sigma انحراف معیار است و سایر پارامترها را قبلاً شرح دادیم. مقدار پیش فرض Sigma ۲ است. تابع edge از لبه‌هایی که قویتر از T نباشند صرف نظر می‌کند. اگر T مشخص نشود و یا تهی باشد، تابع edge مقدار را به صورت خودکار انتخاب می‌کند. وقتی مقدار T صفر باشد، لبه‌هایی ایجاد می‌شود که خط تزاز بسته دارند و این یکی از خصوصیات Log است.

آشکارساز تقاطع‌های صفر(zero_Crossing Detector)

این لبه یاب مانند روش Log عمل می‌کندولی تلفیق با استفاده از تابع فیلتر h انجام می‌شود. ترکیب فراخوان به شرح زیر است:

```
[g, t] = edge(f, 'zerocross', T, H)
```

سایر مقادیر در لبه یاب Log تشریح شده است.

لبه یاب Canny (Canny Edge Detector)

لبه یاب Canny قدرتمندترین آشکارسازی است که تابع Edge ایجاد می‌کند. خلاصه این روش به شرح زیر است:

۱. تصویر با استفاده از یک فیلتر گوسی(Goussian Filter) با انحراف معیار خاص δ برای کاهش پارازیت(noise) هموار می‌شود.

۲. ضریب زاویه موضعی و $g(x,y)=[G_x^2+G_y^2]^{1/2}$ و جهت Edge در هر نقطه محاسبه می‌شود. می‌توان برای محاسبه G_x, G_y از هر یک از روشهای جدول ۷.۱ استفاده کرد. نقطه لبه‌دار نقطه‌ای است که شدت آن در جهت ضریب زاویه حداکثر باشد.

۳. نقطه‌های لبه‌دار تعیین شده در ۲ باعث ایجاد برآمدگی در تصویر حاوی دامنه شیب می‌شود. این الگوریتم در امتداد این برآمدگی‌ها ادامه دارد و کلیه پیکسلهایی را که کاملاً روی برآمدگی‌ها نیستند صفر می‌کند تا یک خط باریک در داده‌های خروجی ایجاد شود. این فرایند را عامل بازدارنده غیرحداکثر می‌نامیم. سپس آستانه پیکسلهای برآمدگی با استفاده از T_1 وT_2 به دست می‌آید. آن دسته از پیکسلهای برا که مقدار آنها بیشتر از T_2 باشد، پیکسلهای قوی لبه هستند. پیکسلهای برآمدگی با مقادیری بین T_1 وT_2 پیکسلهای ضعیف لبه هستند.

۴. بالاخره این الگوریتم با ترکیب پیکسلهای ضعیف که رابطه هشتگانه با پیکسلهای قوی دارند لبه‌ها را به یکدیگر ربط می‌دهد.

ترکیب لبه یاب Canny به شرح زیر است:

```
[g, t] = edge(f, 'canny', T, sigma)
```

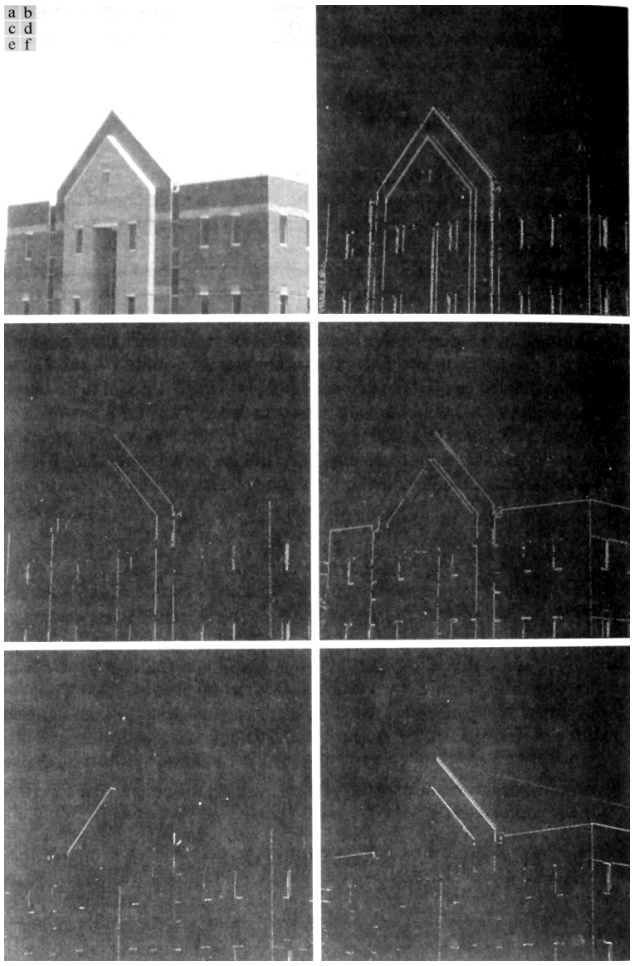
در اینجا T یک بردار است $T=[T_1,T_2]$ و حاوی ۲ آستانه‌ای است که در مرحله ۳ روش قبل شرح دادیم. اگر fدر شناسه خروجی درج شده باشد؛ یک بردار دو عنصری است که مقادیر دو آستانه الگوریتم را دارد. مابقی ترکیب همان است که برای سایر روشها شرح دادیم، و اگر T مشخص نشده باشد، شامل محاسبات خودکار می‌شود. مقدار پیش فرض Sigma , ۱ است.

می‌توان لبه‌های عمودی تصویر (f)در عکس ۷.۶.(a) را با فرمانهای زیر استخراج و نمایش داد:

```
>> [gv, t] = edge(f, 'sobel', 'vertical');
```

```
>> imshow(gv)
>> t
t =
      0.0516
```

همان طور که در تصویر ۷.۶(b) دیده می‌شود، لبه‌های اصلی نتیجه عمودی هستند لبه‌های شیبدار مولفه‌های افقی و عمودی دارند بنا بر این آنها نیز آشکار می‌شوند) می‌توان با مشخص کردن سطح آستانه بالاتر لبه‌های ضعیف را پاک کرد. مثلاً عکس ۷.۶(c)با استفاده از فرمان زیر ایجاد شد:



تصویر ۷.۶. (a)) تصویر اصلی (b) نتیجه تابع Edge با استفاده از Sobel Mask عمودی با آستانه‌ای که به صورت خودکار تعیین شده است. (c) نتیجه با استفاده از یک آستانه خاص (d) نتیجه تعیین لبه‌های افقی و عمودی با یک آستانه خاص (e) نتیجه محاسبه لبه‌ها در زاویه ۴۵ درجه با استفاده از Mask و آستانه خاص (f) نتیجه محاسبه لبه‌ها در زاویه منفی ۴۵ درجه با استفاده از Mask و استانه خاص

```
>> gv = edge(f, 'sobel', 0.15, 'vertical');
```

استفاده از همان مقدار T در فرمان زیر

```
>> gboth = edge(f, 'sobel', 0.15);
```

منجر به تشکیل تصویر ۷.۶(d) شد که اصولاً لبه‌های افقی و عمودی را نشان می‌دهد. تابع Edge نمی‌تواند لبه‌های Sobel را در زاویه‌های مثبت و منفی ۴۵ درجه محاسبه کند. برای محاسبه این لبه‌ها باید Mask را مشخص کنیم و از imfilTer استفاده کنیم مثلاً تصویر ۷.۶(e) با استفاده از فرمانهای زیر ایجاد شد:

```
>> w45 = [-2 -1 0; -1 0 1; 0 1 2]
w45 =
      -2    -1    0
      -1    0     1
      0     1     2
>> g45 = imfilter(double(f), w45, 'replicate');
>> T = 0.3*max(abs(g45(:)));
>> g45 = g45 >= T;
>> figure, imshow(g45);
```

قویترین لبه در عکس ۷.۶(e)لبه دارای جهتگیری ۴۵ درجه است. وقتی از این Mask با همان توالی فرمانها استفاده شود، باعث جهتگیری لبه‌ها در منفی ۴۵ درجه می‌شود که این موضوع در عکس ۷.۶(f)نشان داده شده است.

استفاده از گزینه‌های Prewitt و Roberts در این تابع مطابق با همان رویه‌ای است که در لبه یاب Sobel تشریح شد.

در اینجا عملکرد نسبی لبه یابهای Sobel ، LOG و Canny را بررسی می‌کنیم. هدف آن است که با استخراج لبه‌های اصلی تصویر ساختمانی (f)در عکس ۷.۶(a) یک نقشه تمیز از آن تهیه کنیم و جزئیات نامربوط همچون بافت ظریف دیوارهای آجری و بام سفال پوش شده را از آن حذف کنیم. لبه‌های اصلی مورد نظر در این بحث گوشه ساختمان، پنجره، چهارچوب آجری ورودی ساختمان که پیرامون دو سوم از ساختمان بالاتر از سطح زمین قرار دارد.

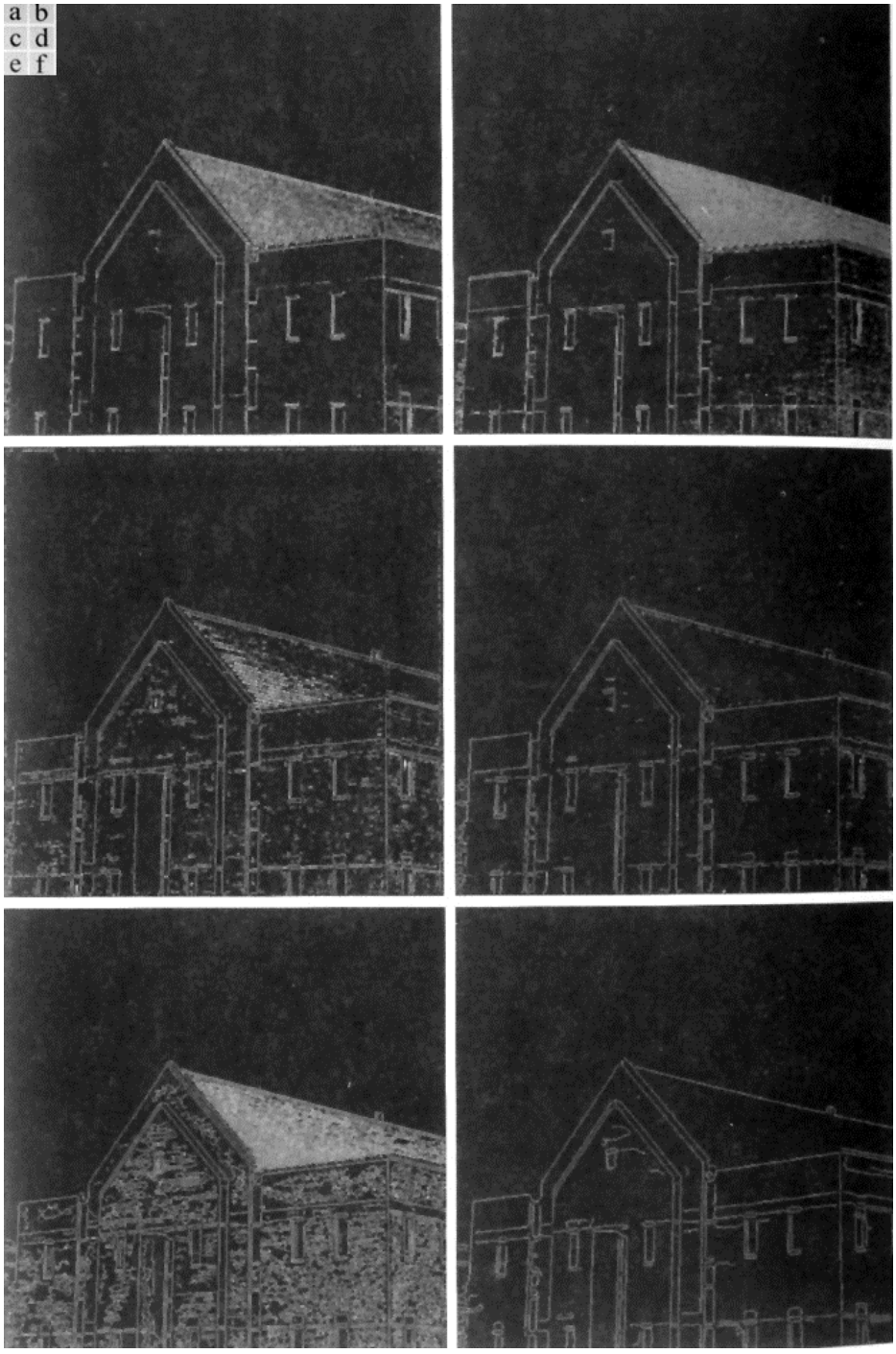
در ستون سمت چپ عکس ۷.۷ تصویر به دست آمده لبه‌ها با استفاده از ترکیب گزینه‌های Sobel ، LOG و Canny نشان داده شده است.

```
>> [g_sobel_default, ts] = edge(f, 'sobel');              % Fig. 10.7(a)
>> [g_log_default, tlog] = edge(f, 'log');                % Fig. 10.7(c)
>> [g_canny_default, tc] = edge(f, 'canny');              % Fig. 10.7(e)
```

مقادیر آستانه در قسمت خروجی ناشی از محاسبات قبل به شرح زیر است::

Ts=0.074 ,tlog=0.0025 ,tc=[0.019,0.047]

مقادیر پیش فرض Sigma برای گزینه‌های Canny و LOG به ترتیب ۲.۰ و ۱.۰ بودند. به غیر از تصویر Sobel ، نتایج پیش فرض اختلاف زیادی با شرایط لازم برای ایجاد نقشه‌هایی با لبه‌های تمیز داشتند.



تصویر ۷.۷: ستون چپ: مقادیر پیش فرض لبه یابهای Sobel ، LOG و Canny ستون چ(c) نتایج کنش و واکنش برای استخراج خصوصیات اصلی در عکس ۷.۶(a) با حذف جزئیات نامربوط. تا حالا لبه یاب Canny بهترین نتایج را تولید کرده است.

این کار با مقادیر پیش فرض آغاز شد و پارامترهای هر یک از گزینه‌هابا T هدف استخراج خصوصیات اصلی تغییر داده شدند و جزئیات نامربوط تا حد امکان حذف شد. نتایج ستون سمت راست تصویر ۷.۷ با فرمانهای زیر به دست آمد:

```
>> g_sobel_best = edge(f, 'sobel'. 0.05);                    % Fig. 10.7(b)
>> g_log_best = edge(f, 'log', 0.003, 2.25);                 % Fig. 10.7(d)
>> g_canny_best = edge(f, 'canny', [0.04 0.10], 1.5);        % Fig. 10.7(f)
```

همان طور که در عکس b.۷.۷دیده می‌شود، وقتی سعی کردیم نوار واقعی و لبه چپ ورودی را استخراج کنیم، میزان اختلاف نتایج Sobel با هدف بیشتر شد. نتیجه LOG در عکس ۷.۷.(d) بهتر از نتیجه Sobel و مقدار پیش فرض LOG است ولی نمی‌تواند لبه چپ ورودی اصلی و یا نوار فشرده دور ساختمان را استخراج کند. نتایج Canny تاکنون بهتر از ۲ مورد دیگر بوده است. توجه کنید که چگونه لبه سمت چپ ورودی به خوبی آشکار شده است، همچون نوار فشرده و سایر جزئیات مانند هواکش روی بام بالای ورودی اصلی. لبه یاب Canny هم خصوصیات مطلوب را آشکار می‌کند و هم تمیزترین نقشه را از لبه‌ها تهیه می‌کند.

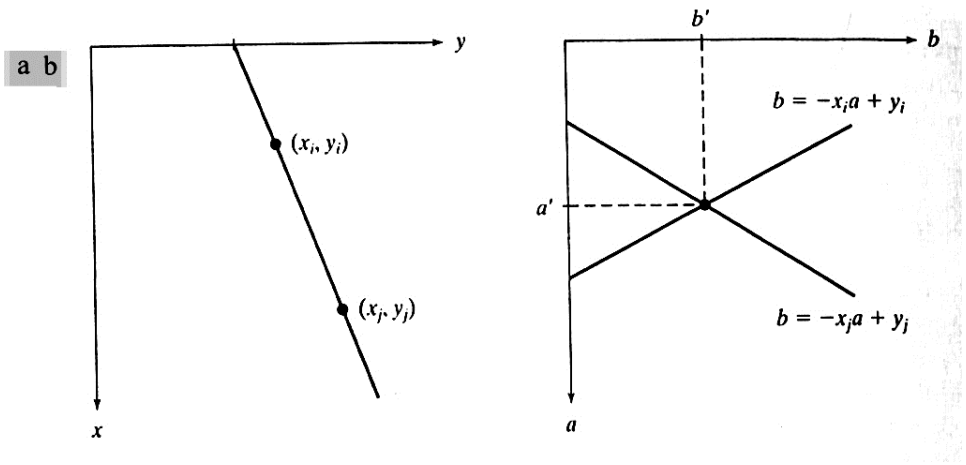
۷.۲. آشکارسازی خطوط با استفاده از تبدیل Hough (Line Detection Using the Hough Transformation)

در روشهای قبل پیکسلهایی روی لبه‌ها ایجاد شد. ولی این خصوصیات عملاً به ندرت ویژگی‌های یک لبه را دارند. علت آن وجود پارازیت‌ها، گسستگی در لبه‌های روشن متغیر، و سایر جلوه‌های ایجاد کننده گسستگی‌های کاذب است. بنا بر این بعد از اجرای الگوریتم‌های آشکارسازی لبه پیکسلهای لبه به صورت واحدهای معنی‌دار درمی‌آیند. یک روش که برای یافتن و ایجاد ارتباط بین بخش‌های یک تصویر به کار برده می‌شودتبدیل Houghاست.

فرض کنید نقطه‌هایی در یک تصویر دودویی وجود دارد و می‌خواهیم زیرمجموعه‌های این نقاط را پیدا کنیم که روی خطوط مستقیم قرار دارند. یک راه آن است که ابتدا همه خطوط تعیین شده در هر جفت نقطه را پیدا کنیم و سپس همه زیرمجموعه‌های آن نقطه را که نزدیک به خطوط خاص هستند بیابیم.مسئله این روش آن است که ابتدا باید $n(n-1)/2 \sim n_2$ تعداد خط پیدا شود و بعد تعداد $n(n(n-1))/2 \sim n_3$ مورد قیاس با هر نقطه تا همه خطوط انجام شود. این روش از نظر محاسباتی در همه موارد غیر از برنامه‌های عادی بازدارنده است.

ولی در تبدیل Houghیک نقطه (x_i, y_i) را در نظر گرفته و همه خطوط از آن عبور می‌کنند.

بنا بر این خطوط زیادی از (x_i, y_i) عبور می‌کنند، که همه آنها معادله مختصات شیب $(y_i=ax_i+b)$ را برای مقادیر خاصی از a, b برآورده می‌کنند با نوشتن این معادله$b=-x_ia+y_i$ و توجه به مقطع ab (که آن را فضای مقادیر مشخص نیز می‌نامیم) معادله خطوط تکی نهادهای دوگانه (x_i, y_i)تشکیل می‌شود. نقطه دوم(x_j, y_j) نیز خطی در فضای پارامترمرتبط با آن دارد و این خط(a',b') با خط(x_i, y_i) تقاطع دارد در اینجا a شیب و b مختصات خط حاوی مقطع هردو (x_i, y_i)و(x_j, y_j) است. در واقع کلیه نقاط این خط خطوطی در فضای پارامتر دارند که تقاطع آن در (a', b') است. این مفاهیم در عکس ۷.۸ به تصویر کشیده شده است.

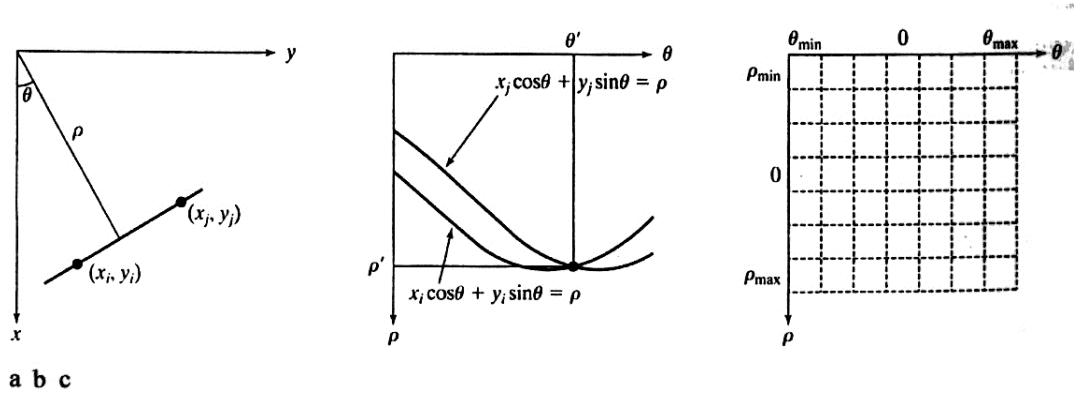


تصویر ۷.۳: (a)) سطح x y (b) فضای پارامتر

می‌توان خطوط فضای پارامتر را مطابق با کلیه نقاط تصویر ترسیم کرد. سپس خطوط تصویر(x_i, y_i) بر حسب محل برخورد خطوط فضای پارامتر مشخص می‌شوند. یک مسئله در این روش آن است که A (که شیب خط است) هر چه خط به جهت عمودی نزدیکتر شود به سمت بینهایت میل می‌کند. یک روش برای حل این مسئله نمایش خط به شکل عادی است:

$$x\cos\theta + y\sin\theta = \rho$$

خصوصیات هندسی پارامترهای ρ, θ در عکس ۷.۹.(a) نشان داده شده است. در خط افقی $\theta=0$ است و ρ معادل مختصات مثبت x است. در یک خط عمودی θ مساوی ۹۰ درجه است و ρ معادل مختصات مثبت و یا منفی y است هر منحنی عکس ۷.۹.بی نمایانگر خطوطی است که از یک نقطه خاص(x_i, y_i) عبور می‌کنند. خط تقاطع مطابق با خطی است که از (xi, yi) و (x_j, y_j) عبور می‌کند.



تصویر ۷.۹: تعیین مقادیر کمی خطوط در سطح x y (b) منحنی‌های سطح $\rho\theta$ نقطه تقاطع مطابق با پارامترهای اتصال خطوط است (c) تقسیم سطح $\rho\theta$ به سلولهای انباشته

ریشه جذابیت محاسباتی تبدیل Houghتقسیم مقادیر فضای $\rho\theta$ به سلولهای انباشتگر است که در تصویر ۷.۹.(c)دیده می‌شود. معمولاً جایی که$(\rho_{min},\rho_{max}),(\theta_{min},\theta_{max})$ حداکثر مقدار به شرح زیر است: (d)فاصله بین گوشه‌های تصویر است. سلول در مختصات (i, j) با مقدار انباشته $A(i, j)$ مطابق با کادر مربوط به مختصات فضای پارامتر(ρ_i,θ_i) است. این سلولها در ابتدا صفر هستند سپس در هر یک از نقاط غیر از پیش زمینه در (x_k, y_k) در سطح تصویر، فرض می‌کنیم θ معادل هر یک از مقادیر فرعی محور θ است و با استفاده از معادله زیر برای ρ مطابق با آن حل می‌شود. مقادیر ρ حاصل از آن به نزدیکترین مقدار سلول در امتداد محور ρ گرد می‌شوند. سپس سلول انباشتگر بزرگ می‌شود. مقدار Q در $A(, j)$ در انتهای این روند یعنی Q عدد نقطه در سطح x y روی خط x قرار دارد. دقت میزان همراستایی این نقاط بستگی به تعداد تقسیم بندیهای سطح $\rho\theta$دارد.

تابعی برای محاسبه تبدیل Houghارائه شده است. این تابع از ماتریسهای Sparse(Sparse matrix) استفاده می‌کند که ماتریسهایی هستند که تعداد کمی عنصر غیرصفر دارند. این خصوصیات امتیازهایی در فضای ذخیره ماتریسها و زمان محاسبه ایجاد می‌کند. برای تبدیل ماتریس A به ماتریس Sparse با استفاده از تابع sparse به شکل زیر عمل می‌کنیم.

```
S = sparse(A)
>> A = [    0     0     0     5
            0     2     0     0
            1     3     0     0
            0     0     4     0     ];
>> S = sparse(A)
S =
      (3,1) 1
      (2,2) 2
      (3,2) 3
      (4,3) 4
      (1,4) 5
```

عناصر غیر صفر S همراه با شاخص‌های ردیف و ستون در خروجی قرار می‌گیرند. عناصر با ستونها مرتب می‌شوند.

ترکیبی که بیشتر با تابع Sparse به کار می‌رود از ۵ شناسه تشکیل می‌شود:

```
S = sparse(r, c, s, m, n)
```

در اینجا r و c به ترتیب بردارهای شاخص‌های ردیف و ستون عناصر غیرصفر ماتریسی که می‌خواهیم به قالب Sparse تبدیل کنیم هستند. پارامتر Sبرداری حاوی مقادیر مطابق با زوج‌های شاخص (r, c) است و m, n ابعاد ستون و ردیف ماتریس حاصل از آن است. مثلاً ماتریس Sمثال قبل را می‌توان مستقیماً با استفاده از فرمان زیر ایجاد کرد:

```
>> S = sparse([3 2 3 4 1], [1 2 2 3 4], [1 2 3 4 5], 4, 4);
```

ماتریس Sparse , Sتوسط یکی از ترکیب‌های قابل اجرا ایجاد می‌شود، می‌توان با استفاده از تابع full که ترکیب آن به شرح زیر است:

دوباره کل این ماتریس را به دست آوریم.

```
A = full(S)
```

برای تحلیل لبه یابی Houghدر نرم افزار MATLAB ابتدا تابع hough.m را می‌نویسیم تا تبدیل Houghرا محاسبه کند.

```
function [h, theta, rho] = hough(f, dtheta, drho)
%HOUGH Hough transform.
%     [H, THETA, RHO] = HOUGH(H, DTHETA, DRHO) computes the hough
%     transform of the image F. DTHETA specifies the spacing (in
%     degrees) of the Hough transform bins along the theta axis. DRHO
%     specifies the spacing of the Hough transform bins along the rho
%     axis. H is the Hough transform matrix. It is NRHO-by-NTHETA,
%     where NRHO = 2*ceil(norm(size(F))/DRHO) – 1, and NTHETA =
%     2*ceil(90/ DTHETA). Note that if 90/ DTHETA is not a integer, the
%     actual angle spacing will be go / ceil(90/ DTHETA).
%
%     THETA is an NTHETA-element vector containing the angle (in
%     degrees) corresponding to each column of H. RHO is an
%     NRHO-element vector containing the value of the corresponding to
%     each row of H.
%
%     [H, THETA, RHO] = HOUGH(F) computes the hough transform using
%     DTHETA = 1 and DRHO = 1.

if nargin < 3
      drho = 1;
end
if nargin < 2
      dtheta = 1;
end
f = double(f);
[M, N] = size(f);
theta = linspaced(-90, 0, ceil(90/ dtheta) + 1);
theta = [theta – fliplr(theta(2:end-1))];
ntheta = length(theta);
D = sqrt((M – 1)^2 + (N – 1)^2);
q = ceil(D/drho);
nrho = 2*q – 1;
rho = linspace(-q*drho, q*drho, nrho);

[x, y, val] = find(f);
x = x – 1;  y = y – 1;
% Initialize output.
h = zeros(nrho, length(theta));
% To avoid excessive memory usage, process 1000 nonzero pixel
% values at a time.
for k = 1:ceil(length(val)/1000)
      first = (k – 1)*1000 + 1;
      last = min(first+999, length(x));

      x_matrix          = repmat(x(first:last), 1, ntheta);
      y_matrix          = repmat(y(first:last), 1, ntheta);
      val_matrix        = repmat(val(first:last), 1, ntheta);
      theta_matrix      = repmat(theta, size(x_matrix, 1), 1)*pi/180);
      rho_matrix        = x_matrix.*cos(theta_matrix) + …
            y_ matrix.*sin(theta_matrix);
```

```
slope = (nrho – 1) / (rho(end) – rho(1));
rho_bin_inedx = round(slope*(rho_matrix – rho(1) + 1);

theta_bin_index = repmat(1:ntheta, size(x_matrix, 1), 1);
% Take advantage of the fact that the SPARSE function, which
% constructs a sparse matrix, accumulates values when input
% indices are repeated. That's the behavior we want for the
% Hough transform. We want the output to be a full (nonsparse)
% matrix, however, so we call function FULL on the output of
% SPARSE.
h = h + full(sparse(rho_bin_index(:), theta_bin_index(:), …
                  val_matrix(:), nrho, ntheta));
end
```

مثال ۷.۵: نمایش تبدیل Hough :

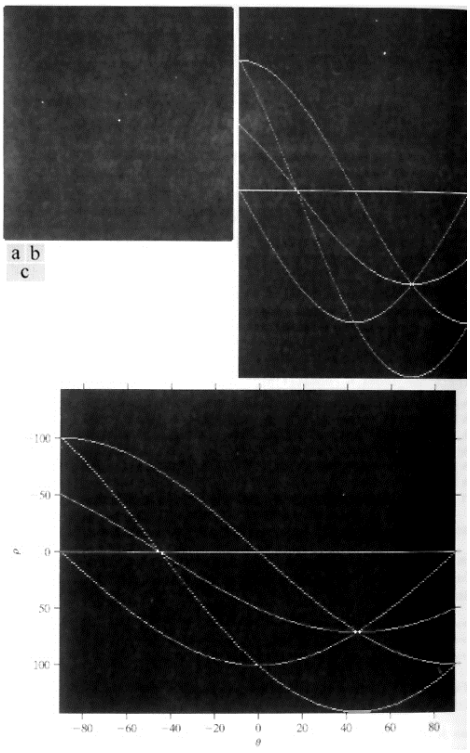
در این مثال کاربرد تابع Houghرا در یک تصویر دودویی ساده نشان داده‌ایم ابتدا تصویری حاوی پیکسلهای پیش زمینه در چند نقطه درست می‌کنیم.

```
>> f = zero(101, 101);
>> f(1, 1)        = 1;  f(101, 1) = 1;    f(1, 101) = 1;
>> f(101, 101)    = 1;  f(51, 51) = 1;
```

تصویر آزمایشی در عکس ۷.۷.(a) دیده می‌شود. سپس تبدیل Houghرا محاسبه کرده و نمایش می‌دهیم.

```
>> H = hough(f);
>> imshow(H, [ ])
```

نتیجه در تصویر ۷.۷.bنشان داده شده است که به شکلی آشنا با imshow دیده می‌شود. ولی بهتر است تبدیل Houghرا در مقیاسی بزرگتر تجسم کنیم.



تصویر ۷.۱۰. (a)) تصویر دودویی با ۵ نقطه (۴ نقطه در گوشه هستند) (b) نمایش تبدیل Houghبا استفاده از imshow (c) نمایش تبدیل Houghبا برچسب گذاری محور (نقطه‌های (a) بزرگتر شده‌اند تا به راحتی دیده شوند.

در قطعه بعدی , Hough را با ۳ شناسه خروجی فرامی‌خوانیم. ۲ شناسه خروجی دوم مقادیر θ و ρ را مطابق با هر ستون و ردیف ماتریس تبدیل Houghدارند. دو بردار θ و ρ را می‌توان به عنوان شناسه‌های ورودی اضافی برای مهار کردن برچسب گذاری محورهای افقی و عمودی در اختیار imshow قرار داد. همچنین گزینه noTruesize را در اختیار imshow می‌گذاریم. تابع محور برای روشن کردن برچسب گذاری محور و پر شدن شکل مستطیل در تصویر است. تابعهای xlabel, ylabel بخش ۳.۳.۱ برای برچسب گذاری محورها با استفاده از نشانه‌گذاری LaTex با حروف یونانی استفاده می‌شود.

```
>> [H, theta, rho] = hough(f);
>> imshow(theta, rho, H, [ ], 'notruesize')
>> axis on, axis normal
>> xlabel('\theta'), ylabel('\rho')
```

نتیجه برچسب گذاری در تصویر ۷.۷.(c)نشان داده شده است. تقاطع ۳ منحنی در زاویه ۴۵ درجه حاکی از آن است که ۲ مجموعه گزینه همراستای سه تایی در (f)وجود دارند. تقاطع این دو منحنی حاکی از آن است که ۴ مجموعه نقطه همراستا در امتداد خطوط افقی و عمودی وجود دارند.

۷.۲.۱. آشکارسازی نقطه اوج تبدیل Hough(Hough Transform Peak Detection)

مرحله اول در استفاده از تبدیل Houghبرای خط یابی و ارتباط برای آشکارسازی نقطه اوج است. یافتن یک مجموعه معنی‌دار از نقاط اوج متمایز در تبدیل Houghدشوار است. با توجه به تعیین کمیت فضای تصویر دیجیتالی و فضای پارامتر تبدیل Houghو این که لبه‌های تصویر کاملاً مستقیم نیستند، تقاط اوج تبدیل Houghمعمولاً در بیش از یک سلول تبدیل Houghقرار دارند. یک راه برای حل این مسئله روش زیر است:

۱. سلول تبدیل Houghرا که حاوی بیشترین مقدار است پیدا کنید، و موقعیت آن را ثبت کنید.

۲. سلول‌های تبدیل Houghرا بلافاصله کنار حداکثر مقدار یافت شده در مرحله اول صفر کنید.

۳. آنقدر این کار را تکرار کنید تا تعداد مطلوب نقاط اوج پیدا شود و یا این که به یک آستانه خاصی دست یابید.

تابع houghpeaks این راه کار را اجرا می‌کند.

```
function [r, c, hnew] = houghpeaks(h, numpeaks, threshold, nhood)
%HOUGHPEAKS Detect peaks in Hough transform.
%     [R, C, HNEW] = HOUGHPEAKS(H, NUMPEAKS, THESHOLD, NHOOD)
%     detects peaks in the Hough transform matrix H. NUMPEAKS specifies the
%     maximum number of peak locations to look for. Values of H below
%     THESHOLD will not be considered to be peaks. NHOOD is a
```

```
%     two-element vector specifying the size of the suppression
%     neighborhood. This is the neighborhood around each peak that is
%     set to zero after the peak is identified. The elements of NHOOD
%     must be positive, odd integers. R and C are the row and column
%     coordinates of the identified peaks. HNEW is the Hough transform
%     with peak neighborhood suppressed.
%
%     If NHOOD is omitted, it defaults to the smallest odd values >=
%     size(H)/50. If THRESHOLD is omitted, it defaults to
%     0.5*max(H(:)). If NUMPEAKS is omitted, it defaults to 1.
if nargin < 4
      nhood = size(h) / 50;
      % make sure the neighborhood size is odd.
      Nhood = max(2*ceil(nhood/2) + 1, 1);
end
if nargin < 3
      threshold = 0.5 * max(h(:));
end

done = false;
hnew= h; r = [ ]; c = [ ];
while ~done
      [p, q] = find(hnew == max(hnew(:)));
      p = p(1); q = q(1);
      if hnew(p, q) >= threshold
            r(end + 1) = p; c(end + 1) = q;
            % Suppress this maximum and its close neighbors.
            p1 = p – (nhood(1) – 1)/2; p2 = p + (nhood(1) – 1)/2;
            q1 = q – (nhood(2) – 1)/2; q2 = q + (nhood(2) – 1)/2;
            [pp, qq] = ndgrid(p1:p2, q1:q2);
            pp = pp(:); qq = qq(:);
            % Throw away neighbor coordinates that are out of bounds in
            % the rho direction.
            badrho = find((pp < 1) | (pp > size(h, 1)));
            pp(badrho) = []; qq(badrho) = [];

            % For coordinate that are not of bounds in the theta
            % direction, we want to consider that H is antisymmetric
            % along the rho axis fot theta = +/- 90 degree.
            theta_too_low = find(qq < 1);
            qq(theta_too_low) = size(h, 2) + qq(theta_too_low);
            pp(theta_too_low) = size(h, 1) + pp(theta_too_low) + 1;
            theta_too_high = find(qq > size(h, 2));
            qq(theta_too_high) = qq(theta_too_high) - size(h, 2);
            pp(theta_too_high) = size(h, 1) – pp(theta_too_high) + 1;
            % Convert to linear indices to zero out all the values.
            Hnew(sub2ind(size(hnew), pp, qq)) = 0;
            done = length(r) == numpeaks;
      else
            done = true;
]     end
end
```

تابع Houghpeaks در مثال ۷.۶ نشان داده شده است.

۷.۲.۲ آشکارسازی خط تبدیل Houghو برقراری ارتباط(Hough Transform Line Detection and Linking)

پس از شناسایی نقاط اوج در تبدیل Houghباید مشخص شود آیا بخش‌هایی از خط با نقاط اوج ارتباط دارند یا خیر و این که محل آغاز و پایان آنها کجا است. اولین قسمت کار در هر یک از نقاط اوج یافتن کلیه پیکسلهای غیرصفر در تصویری است که نقشی در آن نقطه داشته است. برای این کار تابع hougpixels را می‌نویسیم:

```
function [r, c] = houghpixels(f, theta, rho, rbin, cbin)
%HOUGHPIXELS Compute image pixels belonging to Hough transform bin.
%     [R, C] = HOUGHPIXELS(F, THETA, RHO, RBIN, CBIN) compute the
%     row-column indices (R, C) for nonzero pixels in image F that map
%     to a particular Hough transform bin, (RBIN, CBIN). RBIN and CBIN
%     are scalars indicating the row-column bin location in the Hough
%     transform matrix returned by function HOUGH. THETA and RHO are
%     the second and third output arguments from the HOUGH function.
[x, y, val] = find(f);
x = x -1; y = y – 1;
theta_c = theta(cbin) * pi / 180;
rho_xy = x*cos(theta_c) + y*sin(theta_c);
nrho = length(rho);
slope = (nrho – 1)/(rho(end) – rho(1)) + 1);
rho_bin_index = round(slope*(rho_xy-rho(1)) + 1);
idx = find(rho_bin_index == rbin);
r = x(idx) + 1; c = y(idx) + 1;
```

پیکسلهایی که با استفاده از تابع Houghپیکسل با نقاط خاصی مرتبط شده‌اند، باید به خطهایی با قطعات مشخص تقسیم شوند. تابع houghlines از راه کار زیر استفاده می‌کند:

۱. عنصر تصویری را در محل خودش θ -90 درجه بچرخانید طوری که در امتداد یک خط عمودی قرار بگیرند.

۲. موقعیت پیکسل را بر اساس مقادیر چرخانده شده x مرتب کنید.

۳. برای یافتن فاصله‌ها از تابع diff استفاده کنید و از فاصله‌های اندک چشم پوشی کنید. با این کار بخش‌های مجاور خط که با فاصله کمی از یکدیگر تفکیک شده‌اند ترکیب می‌شوند.

۴. اطلاعات مربوط به آن قسمت از خط را که طولانی‌تر از حداقل طول استانه است برگردانید.

```
function lines = houghlines(f, theta, rho, rr, cc, fillgap, minlength)
%HOUGHLINES Extract line segments based on the Hough transform.
%     LINES = HOUGHLINES(F, THETA, RHO, RR, CC, FILLGAP,
%     MINLENGTH) extracts line segments in the image F associated with particular
%     bins in a Hough transform. THETA and RHO are vectors returned by
%     function HOUGH. Vectors RR and CC specify the rows and columns
%     of the Hough transform bins to use in searching for line
%     segments. If HOUGHLINES finds two line segments associated with
%     the same Hough transform bin that are separated by less than
```

```
%     FILLGAP pixels, HOUGHLINES merges them into a single line
%     segment. FILLGAP defaults to 20 if omitted. Merged line
%     segments less than MINLENGTH pixels long are discarded.
%     MINLENGTH defaults to 40 if omitted.
%
%     LINES is a structure array whose length equals the number of
%     merged line segments found. Each elements of the structure array
%     has these fields:
%
%           point1      End-point of the line segment; two-element vector
%           point2      End-point of the line segment; two-element vector
%           length      Distance between pint1 and point2
%           theta       Angle (in degrees) of the Hough transform bin
%           rho         Rho-axis position of the Hough transform bin

if nargin < 6
      fillgap = 20;
end
if nargin < 7
      minlength = 40;
end

numlines = 0; lines = struct;
for k = 1:length(rr)
      rbin = rr(k);     cbin = cc(k);
      % Get all pixels associated with Hough transform cell.
      [r, c] = houghpixels(f, theta, rho, rbin, cbin);
      if isempty(r)
            continue
      end
      % Rotate the pixel location about (1, 1) so that they lie
      % approximately along a vertical line.
      omega = (90 – theta(cbin)) * pi / 180;
      T = [cos(omega) sin(omega); - sin(omega) cos(omega)];
      xy = [r -1 c-1] * T;
      x = sort(xy(:, 1));
      % Find the gaps larger than the threshold.
      diff_x = [diff(x); Inf];
      idx = [0; find(diff_x > fillgap)];
      for p = 1:length(idx) – 1
            x1 = x(idx(p) + 1); x2 = x(idx(p + 1));
            linelength = x2 – x1;
            if linelength >= minlength
                  point1 = [x1 rho(rbin)]; point2 = [x2 rho(rbin)];
                  % Rotate the end-point locations back to the original
                  % angle.
                  Tinv = inv(T);
                  point1= point1 * Tinv;  point2 = point2 * Tinv;

                  numlines = numlines+1;
                  lines(numlines).point1 = point1 + 1;
                  lines(numlines).point2 = point2 + 1;
                  lines(numlines).length = linelength;
                  lines(numlines).theta   = theta(cbin);
                  lines(numlines).rho = rho(rbin);
            end
      end
end
```

مثال ۷.۶: استفاده از تبدیل Houghبرای آشکارسازی خط و برقراری ارتباط:

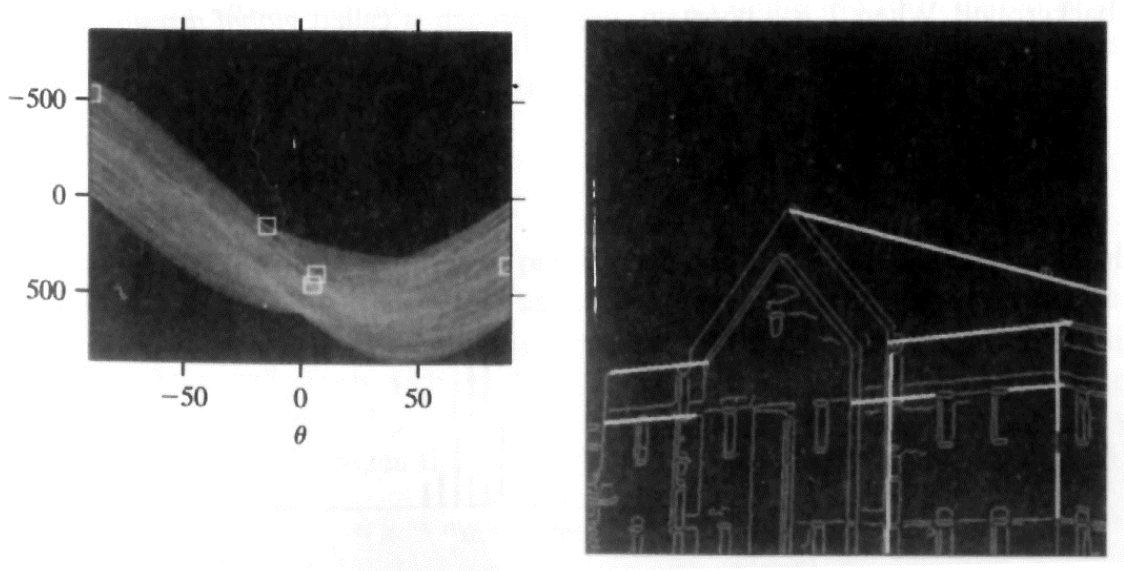
در این مثال از تابعهای Hough، Houghپیکس و HoughLines برای یافتن قطعه‌هایی از خط تصویر دودویی f در عکس ۷.۷.(f)استفاده می‌شود. ابتدا تبدیل Houghرا با استفاده از فاصله گذاری زاویه‌دار ظریفتر از مقدار پیش فرض ۰.۵ به جای ۱.۰محاسبه و نمایش می‌دهیم.

```
>> [H, theta, rho] = hough(f, 0.5);
>> imshow(theta , rho, H, [ ], 'notruesize'), axis on, axis normal
>> xlabel('\theta'), ylabel('\rho')
```

سپس با تابع houghpeaks ,۵ نقطه اوج تبدیل Houghرا که ممکن است مهم باشند پیدا می‌کنیم.

```
>> [r, c] = houghpeaks(H, 5);
>> hold on
>> plot(theta(c), rho(r), 'linestyle', 'none', …)
      'marker', 's', 'color', 'w')
```

در عکس ۷.۱۱.(a) تبدیل Houghرا قرار گرفتن نقاط اوج در روی آن می‌بینید. در آخر برای یافتن و ایجاد ارتباط بین قطعات خط از تابع HoughLines استفاده می‌کنیم.



تصویر ۷.۱۱: (a)) تابع Houghبا انتخاب ۵ نقطه اوج (b) قطعه‌های خط که مطابق با نقطه اوج تبدیل Houghهستند.

با استفاده از imshow, hold on , plot قطعات خط را روی تصویر دودویی اصلی قرار می‌دهیم.

```
>> lines = houghlines(f, theta, rho, r, c)
>> figure, imshow(f), hold on
>> for k = 1:length(lines)
xy = [lines(k).point1; lines(k).point2];
plot(xy(:,2), xy(:,1), 'LineWidth', 4, 'Color', [.6 .6 .6]);
end
```

در عکس ۷.۱۱(b) تصویر با قطعات آشکار شده دیده می‌شود که به صورت خط قطور خاکستری روی آن قرار گرفته‌اند.

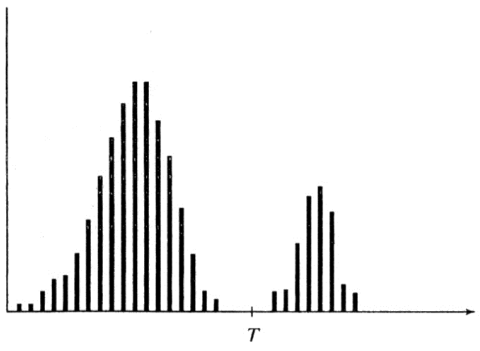
۷.۳. آستانه یابی (در اینجا منظور بازسازی تصویر به دو رنگ با توجه به مقدار آستانه است) Thresholding:

آستانه یابی به دلیل خصوصیات ذاتی و اجرای ساده نکته اصلی قطعه بندی تصاویر است. نکته این مبحث روشهای انتخاب مقدار آستانه به صورت خودکار است. همچون روشی برای تغییر آستانه بر حسب خصوصیات نقاط همجوار تصویر در نظر می‌گیریم.

فرض کنید سابقه نمای شدت رنگ که در عکس ۷.۱۲ نشان داده شده است مطابق با تصویر $f(x, y)$ باشد، و اشیای روشن روی پس زمینه تیره طوری قرار گرفته باشند که بتوان عناصر تصویری اشیا و پس زمینه را به ۲ حالت اصلی تقسیم کرد. یک روش برای استخراج اشیا از پس زمینه انتخاب آستانه T است که این حالتها را تفکیک می‌کند. در این صورت در هر نقطه (x,y) که شرط $f(x, y) > T$ برقرار باشد، آن را نقطه شی(Object Point) می‌نامیم در غیر این صورت آن را نقطه زمینه(background point) می‌نامیم. به عبارت دیگر تصویر آستانه یابی شده $g(x, y)$ به شکل زیر تعریف می‌شود:

؟

آن دسته از عناصر تصویری که برچسب ۱ دارند مطابق با اشیا هستند در صورتی که عناصری که برچسب ۰ دارند مطابق با زمینه هستند. وقتی T ثابت باشد، این روش را آستانه یابی کلی می‌نامیم.



تصویر ۷.۱۲: انتخاب آستانه با تحلیل دیداری سابقه نمای دو حالته

روشهای انتخاب آستانه کلی در بخش ۷.۳.۱ تشریح شده است. روش تغییر آستانه در بخش ۷.۳.۲ بررسی شده است که آن را آستانه یابی موضعی(local thresholding) می‌نامیم.

۷.۳.۱. آستانه یابی کلی(Global Thresholding)

یک روش انتخاب آستانه بررسی خصوصیات ظاهری پیشینه نمای(histogram) تصویر است. سابقه نمای تصویر ۷.۱۲ دو حالت متمایز دارد، در نتیجه، به راحتی می‌توان آستانه T را برای تفکیک آنها انتخاب کرد. روش دیگر انتخاب T روش آزمایش و خطا است یعنی ناظر آنقدر آستانه‌های مختلف را محک می‌زند تا به وجود بهترین آنها پی ببرد. انجام چنین کاری مخصوصاً در محیطهای تعاملی موثر است مثلاً وقتی که کاربر بتواند آستانه را با ابزار کنترل گرافیکی همچون لغزنده گرافیکی تغییر دهد و نتایج را بلافاصله پس از انجام کار مشاهده کند.

Gonzalez & Woods[2002] برای انتخاب خودکار آستانه روش تکراری زیر را توصیف می‌کنند:

۱. مقدار اولیه T را تخمین بزنید، (پیشنهادهایی می‌شود نقطه وسط بین مقادیر حداقل و حداکثر شدت رنگ را در نظر بگیرید).
۲. تصویر را با استفاده از T قطعه بندی کنید. در این صورت دو گروه عنصر تشکیل می‌شود: یکی G_1 که حاوی کلیه عناصری است که شدت آنها بزرگتر یا مساوی T است و دومی G_2 حاوی عناصر تصویری که مقدار آنها کمتر از T است.
۳. میانگین شدت μ_1 و μ_2 را برای عناصر تصویری نواحی G_1, G_2 محاسبه کنید.
۴. مقدار استانه جدید را محاسبه کنید.
۵. مراحل ۲ تا ۴ را آنقدر تکرار کنید تا مقدار اختلاف T در تکرارهای متوالی کمتر از پارامتر از پیش تعیین شده T_0 شود.
در مثال ۷.۷ نحوه اجرای این روند در نرم افزار MATLAB نشان داده شده است.
این جعبه ابزار تابعی به نام grayThresh ایجاد می‌کند که آستانه را با استفاده از روش Otsuمحاسبه می‌کند. ابتدا پیشینه نمای(histogram) هنجارساز شده را به صورت یک تابع چگالی متمایز در نظر می‌گیریم:

?

در اینجا n کل تعداد عناصر موجود در تصویر است n_q تعداد عناصر تصویری است که شدت رنگ آنها در سطح r_q است. و L کل تعداد سطوح شدت رنگ تصویر است. حالا فرض کنید استانه k طوری انتخاب شود که C_0 مجموعه عناصر تصویری با سطوح [0, 1,…, k-1]

باشد و C_1 مجموعه عناصر تصویری با سطوح [K,K+1,…,L – 1] باشد. در روش Otsuمقدار آستانه K انتخاب می‌شود و اختلاف بین طبقات δ^2B بیشینه می‌شود.

?

$$w_0 = \sum_{q=0}^{k-1} p_q(r_q)$$
$$w_1 = \sum_{q=k}^{L-1} p_q(r_q)$$
$$\mu_0 = \sum_{q=0}^{k-1} qp_q(r_q) / w_0$$
$$\mu_1 = \sum_{q=k}^{L-1} qp_q(r_q) / w_1$$
$$\mu_T = \sum_{q=0}^{L-1} qp_q(r_q)$$

یعنی تابع grayThresh تصویری را انتخاب می‌کند، پیشینه نمای(histogram) آن را محاسبه می‌کند، و مقدار آستانه‌ای را که δ^2B را بیشینه می‌کند پیدا می‌کند. مقدار آستانه هنجارمند شده بین ۰.۰ و ۱.۰ است. ترکیب فراخوان grayThresh به شرح زیر است:

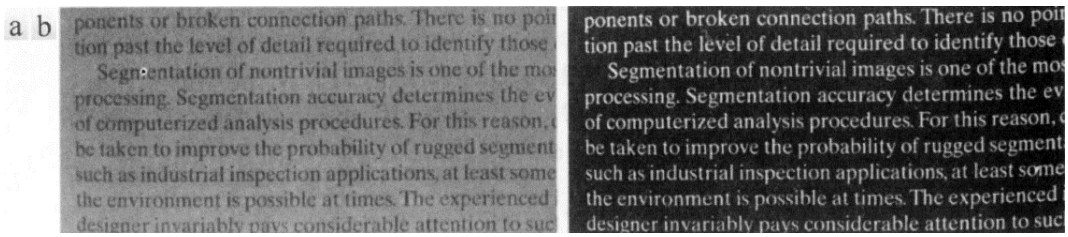
```
T = graythresh(f)
```

در اینجا f تصویر ورودی و T آستانه حاصل از آن است. برای قطعه بندی تصویر از T در تابع im2bw استفاده می‌کنیم زیرا استانه تا حیطه [0,1] هنجارمند شده است. قبل از استفاده باید مقیاس بندی صحیح روی آن انجام شود. مثلاً اگر fاز نوع uinT8 باشد، قبل از استفاده از آن T را ضربدر ۲۵۵ می‌کنیم.

مثال ۷.۷: محاسبه آستانه کلی :

در این مثال روش تکراری و روش Otsuبرای کار با تصویر مقیاس خاکستری fدر متن اسکن شده عکس ۷.۱۳.(a) تشریح شده است. روش تکراری را می‌توان به شرح زیر اجرا کرد.

```
>> T = 0.5*(double(min(f(:))) + double(max(f(:))));
>> done = false;
>> while ~done
      g = >= T;
      Tnext = 0.5*(mean(f(g)) + mean(f(~g)));
      Done = abs(T – Tnext) < 0.5;
      T = Tnext;
end
```



تصویر ۷.۱۳: (a)) متن اسکن شده (b) متن آستانه یابی شده با استفاده از تابع GrayThresh

در اینجا T معادل ۷۱.۴۷ است.

```
>> T2 = graythresh(f)
T2 =
      0.3961
>> T2 * 255
ans =
      101
```

در استانه یابی با استفاده از این دو روش تصاویری ایجاد می‌شود که نمی‌توان آنها را از یکدیگر تشخیص داد. در عکس ۷.۱۳.ب آستانه یابی تصویر با استفاده از T_2 نشان داده شده است.

۰.۳.۲. آستانه یابی موضعی (Local Thresholding)

همان طور که قبلا گفتیم وقتی روشنایی پس زمینه ناهموار باشد، روشهای آستانه یابی کلی ناکارآمد هستند. یک روش متداول اجرای پیش پردازش برای رفع مسائل نوری است. سپس یک استانه کلی روی تصویر پیش پردازش شده اعمال می‌شود. نتایج بهبود استانه یابی که با اجرای یک عملگر به شکل کلاه دراز انجام شد. سپس grayThresh روی نتایج به کار برده شد. این روند معادل استانه یابی $f(x, y)$ است و تابع آستانه یاب $T(x, y)$ به شکل موضعی تغییر می‌کند.

$$g(x,y) = \begin{cases} 1 & if \;\; f(x,y) \geq T(x,y) \\ 0 & if \;\; f(x,y) < T(x,y) \end{cases}$$

$$T(x,y) = f_0(x,y) + T_0$$

تصویر $f_0(x, y)$ شکل ظاهری باز شده f است و T_0 ثابت نتیجه اجرای تابع grayThresh روی F_0 است.

۷.۴. قطعه بندی منطقه‌ای(Region_Based Segmentation)

هدف از قطعه بندی تقسیم بندی تصویر به ناحیه‌های مختلف است. در بخش‌های ۷.۱ و ۷.۲ برای بررسی این مسئله مرز بین ناحیه‌ها با توجه به گسستگی سطوح شدت رنگ پیدا شد در صورتی که در بخش ۷.۳ قطعه بندی با توجه به حد آستانه میزان پراکندگی خصوصیات عناصر تصویری همچون شدت رنگ انتخاب شد. در این بخش به روشهای قطعه بندی با توجه به یافتن مستقیم ناحیه‌ها می‌پردازیم.

۷.۴.۱. فرمولهای اساسی(Basic Formulation)

فرض کنید R نمایانگر کل ناحیه تصویر است. قطعه بندی فرایندی است که طی آن R به n عدد ناحیه فرعی تقسیم می‌شود.

$(a)\quad \bigcup_{i=1}^{n} R_i = R.$

$(b)\quad R_i$ is a connected region, i = 1,2,…,n.

$(c)\quad R_i \cap R_j = \phi \ \text{for all } i \text{ and } j, i \neq j.$

$(d)\quad P(R_i) = TRUE \ \text{for } i = 1,2,..,n.$

$(e)\quad P(R_i \cap R_j) = FALSE \ \text{for any adjacent regions } R_i \text{ and } R_j .$

در اینجا P(Ri) یک محمول منطقی است که در نقطه‌های مجموعه Ri, θ مجموعه تهی است.

طبق شرط a قطعه بندی باید کامل باشد، یعنی هر عنصر تصویری باید در یک ناحیه باشد. طبق شرط دوم نقاط یک ناحیه باید به شکلی از پیش تعریف شده مرتبط شوند (مثلاً ارتباطهای ۴ تایی یا ۸ تایی). شرط C حاکی از آن است که ناحیه‌ها باید گسسته باشند. شرط d مربوط به خصوصیاتی است که عناصر داخل ناحیه قطعه بندی شده باید داشته باشند. اگر کلیه عناصر تصویری Ri سطح خاکستری یکسان داشته باشند PRi=True می‌شود. در شرط e می‌بینیم که ناحیه‌های همجوار Ri, Rj بر حسب محمول P متفاوت هستند.

۷.۴.۲. توسعه ناحیه‌ها (Region Growing)

در روند توسعه ناحیه‌ها گروه‌های اصلی و فرعی عناصر تصویری بر اساس استانداردهای از پیش تعیین شده رشد طبقه بندی می‌شوند. معمولاً کار با مجموعه‌ای از نقاط هسته‌دار (Seeds Points) آغاز می‌شد سپس آن دسته از عناصر تصویری که خصوصیاتی مشابه با هسته دارند (مثلاً رنگ یا سطح خاکستری خاص) با آن مرتبط می‌شوند.

انتخاب یک یا چند نقطه شروع بر اساس ماهیت مسئله است که در مثال ۷.۸ نشان داده شده است. وقتی اطلاعات برای استدلال کردن موجود نباشد، یک روش آن است که خصوصیاتی را که نهایتاً برای تخصیص عناصر تصویری در روند توسعه به کار برده می‌شوند محاسبه شوند. اگر مجموع مقادیر در نتایج این محاسبات نشان داده شود، آن پیکسلهایی که خصوصیات آنها باعث می‌شود در کانون قرار بگیرند هسته نامیده می‌شوند.

انتخاب معیار تشابه هم بستگی به مسئله مورد نظر و هم نوع داده‌های تصویری موجود دارد. مثلاً تحلیل تصاویر نقشه برداری ماهواره بستگی به استفاده از رنگها دارد. بدون داشتن اطلاعات مندرج در تصاویر رنگی حل این مسئله دشوار یا حتی غیرممکن می‌شود. وقتی تصاویر تک رنگ باشند، تحلیل ناحیه‌ها باید با مجموعه توصیفگرهای مبتنی بر سطوح شدت رنگ (همچون بافت تصویر) و خصوصیات فضایی آن باشد.

اگر اطلاعات ارتباطهای همجوار در فرایند توسعه ناحیه استفاده نشود، توصیفگرها ممکن است نتایج گمراه کننده تولید کنند. مثلاً یک رشته عناصر تصویری با سه نوع مقدار شدت رنگ متمایز را در نظر بگیرید. اگر عناصری که شدت رنگ یکسان دارند بدون توجه به ارتباطها در یک گروه در نظر گرفته شوند، نتایج قطعه بندی طوری می‌شود که طبق اصول این مبحث مفهومی ندارد.

مسئله دیگر در توسعه ناحیه تعیین فرمول قانون توقف است. وقتی عناصر تصویری شرایط لازم برای درج در یک ناحیه را نداشته باشند، روند توسعه ناحیه متوقف می‌شود. معیارهایی همچون شدت رنگ، بافت و رنگ ماهیت موضعی دارند و مسئله توسعه ناحیه‌ها در آنها لحاظ نمی‌شود.

یکی از معیارهایی که باعث افزایش قدرت الگوریتم توسعه ناحیه می‌شود مربوط به مسئله اندازه و شباهت بین یک عنصر تصویری محتمل و عناصر تصویری رشد کرده تا آن مرحله هستند (طوری که اطلاعات مقایسه شدت این عناصر محتمل و میانگین شدت ناحیه توسعه)و شکل ناحیه توسعه مورد استفاده قرار می‌گیرد. استفاده از این نوع توصیفگرها بر اساس این فرضیه است که الگویی از نتایج مورد انتظار در دسترس باشد.

برای تشریح نحوه قطعه بندی ناحیه‌ها با یک مثال در نرم افزار MATLAB تابع m را به نام regiongrow ابداع می‌کنیم، تا توسعه اساسی ناحیه‌ها توسط آن انجام شود. ترکیب این تابع به شرح زیر است:

```
[g, NR, SI, TI] = regiongrow(f, S, T)
```

در اینجا f تصویری است که قطعه بندی می‌شود. پارامتر Sمی‌تواند یک آرایه (به همان اندازه f) یا با کمیت عددی باشد. اگر Sیک آرایه باشد، باید در کلیه مختصاتی که هسته‌ها قرار دارند دارای یک باشد و در جاهای دیگر حاوی صفرها باشد. چنین ارایه‌ای را می‌توان با بررسی یا تابع هسته یاب بیرونی تعیین کرد. اگر Sکمیت عددی باشد، شدت را طوری تعریف می‌کند که کلیه نقاط داخل fبا همان مقدار نقاط هسته‌دار می‌شوند. T نیز می‌تواند یک ارایه (با همان اندازه اف) و یا یک کمیت عددی باشد. اگر T یک آرایه باشد، برای هر موضع در

fیک مقدار آستانه دارد. اگر T کمیت عددی باشد، یک استانه عمومی را تعریف می‌کند. مقدار آستانه برای آن است که ببینیم آیا عنصر تصویری به قدر کافی شبیه به هسته‌های ۸ تایی است یا خیر.

مثلاً اگر S=a, T=b باشد، و شدت رنگ را با هم مقایسه کنید در صورتی یک عنصر تصویری (طبق آزمایش آستانه یابی) شبیه به a است که قدر مطلق اختلاف بین میزان شدت و a کمتر از یا معادل b باشد. اگر عنصر تصویری مورد نظر از ۸ طریق به یک یا چند کمیت هسته مربوط باشد، در این صورت، این عنصر تصویری عضو یک یا چند ناحیه محسوب می‌شود. اگر Sو T آرایه باشند، این موضوع مصداق پیدا می‌کند. تنها تفاوتش آن است که مقایسه با توجه به موضع‌های مناسب تعریف شده در Sو مقادیر T مطابق با آن انجام می‌شود.

g تصویر قطعه بندی شده در داده‌های خروجی است و اعضای هر ناحیه با یک عدد صحیح برچسب گذاری شده‌اند. پارامترNR تعداد ناحیه‌های گوناگون است پارامترSI تصویری حاوی نقاط هسته است و پارامتر TI تصویری حاوی عناصر تصویری است که قبل از پردازش ارتباطهای آنها، در آزمایش استانه یابی تایی د شده‌اند. در اینجا SI,TI هم اندازه fهستند.

کد تابع regiongrow به شرح زیر است: توجه داشته باشید که برای کم کردن تعداد نقاط هسته در هر ناحیه در S(وقتی Sیک آرایه باشد) از تابع bwmorph استفاده می‌شود و از تابع imreconsTrucT برای یافتن عناصر تصویری مربوط به هر هسته استفاده می‌شود.

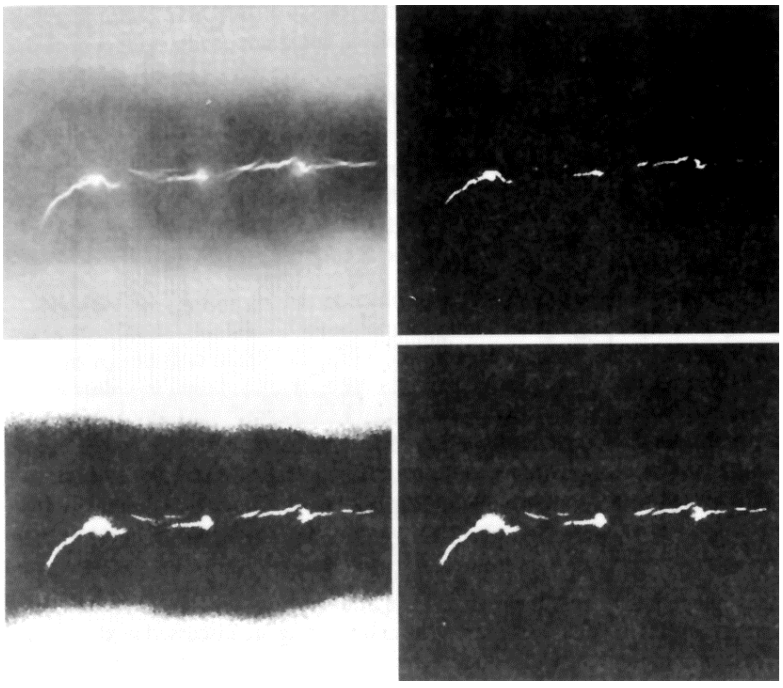
```
function [g, NR, SI, TI] = regiongrow(f, S, T)
%REGIONGROW Perform segmentation by region growing.
%     [G, NR, SI, TI] = REGIONGROW(F, SR, T). S can be an array (the
%     same size as F) with a 1 at the coordinates of every seed point
%     and Os elsewhere. S can also be a single seed value. Similarly,
%     T can be an array (the same size as F) containing a threshold
%     value for each pixel in F. T can also be a scalar, in which
%     case it becomes a global threshold.
%     on the output, G is the result of region growing, with each
%     region labeled by a different integer, NR is the number of
%     region, SI is the final seed image used by the algorithm, and TI
%     is the image consisting of the pixels in F that satisfied the
%     threshold test.
f  = double(f);
% If S is a scalar, obtain the seed image.
if nume(S) = =1
      SI = f = = S;
      S1 = S;
else
      % S is an array. Eliminate duplicate, connected seed locations
      % to reduce the number of loop executions in the following
      % sections of code.
      SI = bwmorph(S, 'shrink', Inf);
      J = find(SI);
      S1 = f(J); % Array of seed values.
end

TI = false(size(f));
for k = 1:length(S1);
      seedvalue = S1(k);
      S = abs(f – seedvalue) <= T;
      TI = TI | S;
end
% Use function imreconstruct with SI as the marker image to
```

```
% obtain the regions corresponding to each seed in S. Function
% bwlabel assigns a different integer to each connected region.
[g, NR] = bwlabel(imreconstruct(SI, TI));
```

مثال ۷.۸: استفاده از خصوصیت توسعه ناحیه برای آشکارسازی تخلخلهای جوشکاری:

در عکس ۷.۱۴.(a) تصویر جوشکاری را با اشعه x (در ناحیه تیره افقی) دیده می‌شود که دارای چندین ترک و تخلخل است (نوارهای سفید درخشان در وسط تصویر به صورت افقی امتداد دارند) از تابع regiongrow برای قطعه بندی ناحیه‌های منطبق بر قسمتهای جوشکاری نشده استفاده می‌کنیم. این قسمت‌های قطعه بندی شده برای بررسی و درج تحقیقات در پایگاه داده و کنترل دستگاه‌های خودکار جوشکاری و سایر برنامه‌های متعدد به کار برده می‌شوند.

مرحله اول این کار تعیین نقاط هسته‌دار اولیه است. در این برنامه مشخص است که برخی از عناصر تصویری جوشکاری‌های معیوب حداکثر مقدار دیجیتال را که در این مورد ۲۵۵ است دارند. طبق این اطلاعات فرض می‌کنیم S=255 است. در مرحله بعد آرایه آستانه را انتخاب می‌کنیم. در این مثال خاص T=65 است. این رقم بر اساس تحلیل پیشینه نما(histogram) در عکس ۷.۱۵ است و نمایانگر اختلاف بین ۲۵۵ و محل اولین فرورفتگی اصلی سمت چپ است که نشاندهنده حداکثر شدت رنگ در ناحیه تیره جوشکاری است. همان طور که قبلاً گفتیم، یک عنصر تصویری برای برقراری ارتباط ۸ تایی باید حداقل به یک عنصر در ناحیه مورد نظر مربوط باشد.

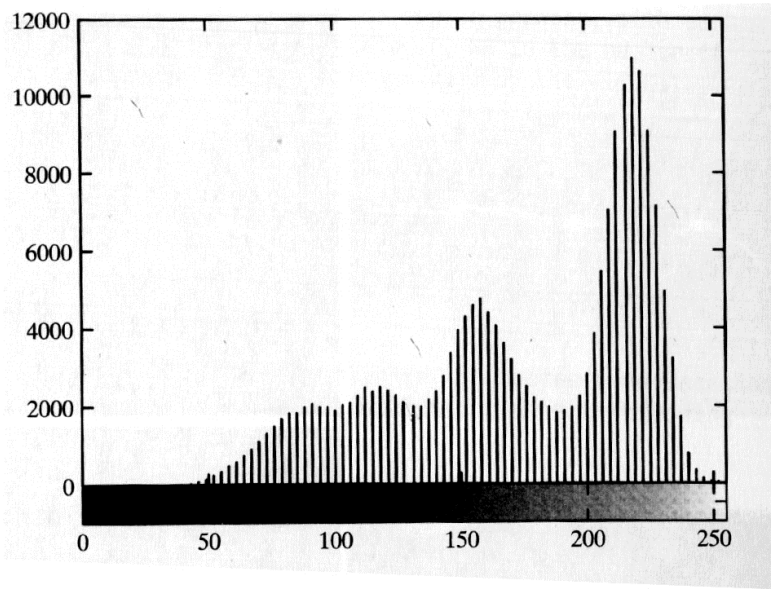


تصویر ۷.۱۴: (a)) تصویر نمایانگر جوشکاری معیوب (b) نقاط هسته‌دار تصویر دودویی که کلیه عناصر تصویری تایید شده در آزمایش آستانه یابی به رنگ سفید نشان داده شده است. (c) نتایج بعد از تحلیل کلیه عناصر تصویری برای نقاط هسته‌دار ۸ تایی

اگر یک عنصر تصویری به بیش از ۱ ناحیه مرتبط باشد، ناحیه‌ها با تابع regiongrow توسعه داده می‌شوند.

نقاط هسته‌دار (تصویرSI) در عکس b.۷.۱۴دیده می‌شوند. در این مورد تعداد آنها زیاد است زیرا هسته‌ها با توجه به کلیه نقاط تصویر و مقدار ۲۵۵ مشخص شده‌اند. عکس (c).۷.۱۴تصویرTI است. کلیه نقاطی که در آزمایش استانه یابی تایید شده‌اند در آن دیده می‌شوند. یعنی نقاط با شدت z_i مانند $|zi-s|<=T$. نتایج استخراج کلیه عناصر تصویری در عکس (c).۷.۱۴که مربوط به نقاط هسته‌دار بوده‌اند در عکس (d).۷.۱۴دیده می‌شوند. این همان تصویر قطعه بندی شده g است. با مقایسه این تصویر با نوع اصلی متوجه می‌شویم که در روش توسعه ناحیه جوشکاری‌های معیوب با دقتی معقول قطعه بندی شده‌اند.

با نگاه به پیشینه نمای(histogram) تصویر ۷.۱۵ متوجه می‌شویم که در عکس ۷.۱۵ نمی‌توان با استفاده از هر یک از روشهای آستانه یابی بخش ۷.۳ به همان راه حل دست یافت. استفاده از ارتباط یکی از شرایط اصلی چنین موردی بوده است.



تصویر ۷.۱۵: پیشینه نمای تصویر ۷.۱۴ (a)

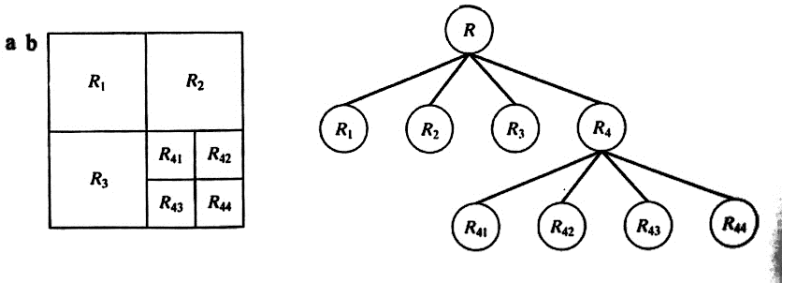
۷.۴.۳ تقسیم بندی و ترکیب کردن ناحیه‌ها (Region Splitting & Merging)

در روشی که بحث کردیم ناحیه‌ها بر اساس نقاط هسته‌دار توسعه داده می‌شوند. راه دیگر آن است که در ابتدا تصویر را به مجموعه‌ای از ناحیه‌های گسسته اختیاری تقسیم کنیم، و سپس برای برآوردن شرایط قید شده در بخش ۷.۴.۱ ناحیه‌ها را تقسیم بندی یا ترکیب کنیم. اصول تقسیم بندی و ترکیب در زیر تشریح شده است.

فرض کنید R نمایانگر کل حیطه تصویر است. سپس محمول p را انتخاب کنید. یک روش قطعه بندی R تقسیم آن به ربع‌های کوچکتر است طوری که در هر ناحیه $R_i, P(R_i)$= True باشد. این کار را با کل ناحیه انجام می‌دهیم. اگر $P(R_i)$ = false باشد، تصویر را به چند ربع تقسیم می‌کنیم. اگر P در هر یک از این ربعها false باشد، آن ربع اصلی را به ربعهای فرعی تقسیم می‌کنیم الی آخر. این شگرد تقسیم بندی خاص به شکل نمودار درختی چهارگانه نمایش داده می‌شود.

در این درخت هر گره چهار نتیجه دارد که در عکس ۷.۱۶ دیده می‌شوند. (ناحیه‌های فرعی مطابق با گره‌های این نمودار درختی گاهی ناحیه چهارگانه(quadregion) یا تصاویر چهارگانه(quadimages) نامیده می‌شوند. توجه داشته باشید که ریشه این درخت مطابق با کل تصویر است و هر گره مطابق با تقسیم بندی فرعی به ۴ زیرمجموعه است. در این مورد فقط R_4 دارای زیرمجموعه است.

اگر فقط تقسیم بندی انجام شده باشد، قسمت نهایی معمولاً حاوی ناحیه‌های همجوار با خصوصیات یکسان است. برای رفع این نقص می‌توان از تقسیم بندی و ترکیب استفاده کرد.



برای تحقق شرایط بخش ۷.۴.۱ فقط باید ناحیه‌های همجوار ترکیب شوند تا عناصر ادغام شده آنها محمول p را تحقق بخشد.

یعنی اگر $P(R_jUR_k)$ = True باشد، ۲ ناحیه همجوار R_j, R_k با هم ترکیب می‌شوند.

بحث فوق را می‌توان در روش زیر خلاصه کرد طوری که در هر مرحله

۱. در هر ناحیه Ri که شرط $P(R_i)$= False برقرار است آن را به چهار ربع گسسته تقسیم کنید.

۲. وقتی تقسیم بندی بیشتر امکان‌پذیر نیست، هر یک از ناحیه‌های همجوار R_i, R_k را که در آنها $P(R_jUR_k)$ = True است با هم ترکیب کنید.

۳. وقتی تقسیم بندی بیشتر امکان‌پذیر نیست این روند را متوقف کنید.

مضمون فوق را می‌توان با تغییرات گوناگونی اجرا کرد. اگر ۲ ناحیه مجاور R_i, R_j با هم ترکیب شوند و هر کدام محمول جدایی را تحقق بخشد، برخی شرایط مهم ساده می‌شود. در نتیجه الگوریتم ساده‌تر و سریعتر اجرا می‌شود زیرا محک زدن محمول محدود به ربع‌های انفرادی است. همان طور که در مثال ۷.۹ دیده می‌شود، نتایج قطعه بندی در این روند ساده سازی مطلوب هستند. اگر از این رویکرد در مرحله ۲ در بالا استفاده شود، کلیه ربعهایی که شرایط محمول را داشته باشند، پر از یک می‌شوند و ارتباط آنها را می‌توان به راحتی با استفاده از تابع imreconstruct بررسی کرد. این تابع می‌تواند ربعهای همجوار را به شکلی مطلوب ترکیب کند. ربعهایی که شرایط محمول را تحقق نبخشند، پر از صفر می‌شوند تا یک تصویر قطعه بندی شده ایجاد شود.

تابعی که در IPT برای تجزیه و تحلیل نمودار درختی به کار برده می‌شود، qtdecomp است. ترکیب مورد نظر در این بخش به شرح زیر است:

```
S = qtdecomp(f, @split_test, parameters)
```

F تصویر ورودی است. Sماتریس Sparse حاوی نمودار درختی است. اگر S (k m) غیرصفر باشد، (k m) گوشه چپ فوقانی بلوک در تجزیه و تحلیل است و بلوک به اندازه S(k m) است. تابع spliT_TesT برای آن است که ببینیم آیا یک ناحیه باید تقسیم شود یا خیر (می‌توانید به مثال تابع spliTmerge مراجعه کنید). ParameTer مقادیر مشخصی هستند که spliT_TesT به آنها نیاز دارد (و با کاما از یکدیگر جدا می‌شوند). مکانیسم آن شبیه به موارد بحث شده در بخش ۳.۴.۲ در تابع colTfilT است.

برای به دست آوردن مقدار عناصر تصویری ربعها در تجزیه نمودار درختی از تابع qTgeTblk با ترکیب زیر استفاده می‌کنیم:

```
[vals, r, c] = qtgetblk(f, s, m)
```

در اینجا Vals آرایه‌ای حاوی مقادیر قطعه‌هایی به اندازه m*m در روند تجزیه fدر نمودار درختی است و Sماتریس Sparseای است که qtdecomp آن را تولید می‌کند. پارامترهای c, r تابعهای حاوی مختصات ردیف و ستون گوشه‌های چپ فوقانی بلوکها هستند.

برای نمایش عملکرد تابع qtdecomp یک تابعM با قابلیت‌های ترکیب سازی و تقسیم بندی ایجاد می‌کنیمکه از روش ساده سازی فوق استفاده می‌کند و اگر هر یک از این ناحیه‌ها شرایط محمول را تحقق بخشد، ۲ ناحیه با یکدیگر ترکیب می‌شوند. تابعی که ما آن را splitmerge می‌نامیم با ترکیب زیر فراخوانی می‌شود.

```
g = splitmerge(f, mindim, @predicate)
```

(f)تصویر ورودی و gتصویر خروجی است که هر کدام از ناحیه‌های مرتبط در آن با یک عدد صحیح متفاوت برچسب گذاری شده است. پارامتر mindim اندازه کوچکترین قطعه‌ای را تعریف می‌کند که در بلوک مجاز به تجزیه شدن است. این پارامتر باید عدد صحیح مثبت به توان ۲ باشد.

تابع predicaTe تابعی است که کاربر تعریف می‌کند و باید در مسیر نرم افزار MATLAB قرار داده شود. ترکیب آن به شرح زیر است:

```
flag = predicate(region)
```

این تابع باید طوری نوشته شود که در صورتی که عناصر تصویری ناحیه محمول تعریف شده در تابع توسط رمز را تحقق بخشند مقدار True (با ۱ منطقی را تولید می‌کند). در غیر این صورت مقدار flag باید false باشد (که یک صفر منطقی است). طرز کار با این تابع در مثال ۷.۹ تشریح شده است.

تابع splitmerge ساختار ساده‌ای دارد. ابتدا تصویر با استفاده از تابع qtdecomp بخش‌بندی می‌شود. تابع split_test برای این که مشخص کند آیا ناحیه‌ای باید تقسیم شود یا خیر از predicate استفاده می‌کند چون که وقتی ناحیه‌ای به ۴ قسمت تقسیم می‌شود، مشخص نیست که کدام یک از این ۴ ناحیه در آزمایش محمول تایید می‌شود. بنا بر این باید ناحیه‌ها را بعداً بررسی کرد تا معلوم شود کدام ناحیه در تصویر بخش‌بندی شده در آزمایش تایید می‌شود. تابع predicate نیز برای همین منظور به کار برده می‌شود. هر ربع که در آزمایش تایید شود با یکها پر می‌شود. هر کدام که نشود با صفر پر می‌شود. با انتخاب عنصر هر ناحیه که پر از ۱ است آرایه شاخص ایجاد می‌شود. این ارایه همراه با تصویر بخش‌بندی شده استفاده می‌شود تا ارتباطهای همجوار ناحیه تعیین شوند. تابع imreconstruct برای این منظور به کار برده می‌شود.

رمز تابع splitmerge بعد از آن است. تابع محمول ساده نشان داده شده در بخش رمزها در مثال ۷.۹ استفاده شده است. توجه داشته باشید که اندازه تصویر ورودی مربعی ایجاد شده که ابعاد آن حداقل اعداد صحیح به قوه ۲ هستند که تصویر را فرا می‌گیرند. این یکی از شرایط تابع qtdecomp است تا مشخص شود که تقسیم بندی تا اندازه ۱ امکان‌پذیر است.

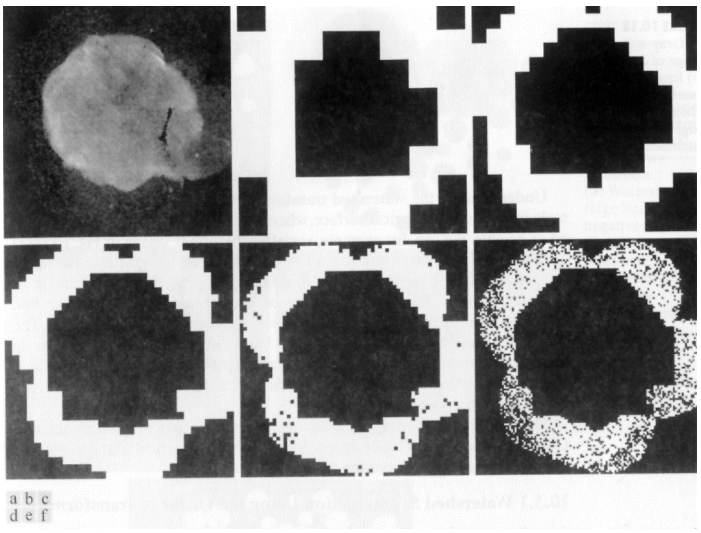
```
function g = splitmerge(f, mindim, fun)
%SPLIMERGE Segment an image using a split-and-merge algorithm.
%     G = SPLIMERGE(F, MINDIM, @PREDICATE) segments image F by using a
%     split-and-merge approach based on quadtree decomposition. MINDIM
%     (a positive integer power of 2) specifies the minimum dimension
%     of the quadtree regions (subimages) allowed. If necessary square
%     program pads the input power of 2. This guarantees that the
%     size that is an integer power of 2. This guarantees that the
%     algorithm used in the quadtree decomposition will be able to
%     split the image down to blocks of size 1-by-1. The result is
%     cropped back to the original size of the input image. In the
%     output, G, each connected region is labeled with a different
%     integer.
%
%     Note that in the function call we use @PREDICATE for the value of
%     fun. PREDICATE is a function in the MATLAB path, provided by the
%     user. Its syntax is
%
%     FLAG = PREDICATE(REGION) which must return TRUE if the pixels
%     in REGION satisfy the predicate defined by the code in the
%     function; otherwise, the value of FLAG must be FALSE.
%
%     The following simple of function PREDICATE is used in
```

```
%     Example 10.9 of the book. It sets FLAG if the
%     intensities of the pixels in REGION have a standard deviation
%     that exceeds 10, and their mean intensity is between 0 and 125.
%     Otherwise FLAG is set to false.
%
%     function flag = predicate(region)
%     sd = std2(region);
%     m = mean2(region);
%     flag = (sd > 10) & (m > 0) & (m < 125);

% Pad image with zeros to guarantee that function qtdecomp will
% split regions down to size 1-by-1.
Q = 2^nextpow2(max(size(f)));
[M, N] = size(f);
f = padarray(f, [Q – M, Q – N], 'post');

% Perform splitting first.
S = qtdecomp(f, @split_test, mindim, fun);
% Now merge by looking at each quadergion and setting all its
% elements to 1 if the block satisfies the predicate.

% Get the size of the largest block. Use full because S is sparse.
Lmax = full(max(S(:)));
% Set the output image initially to all zeros. The MARKER array is
% used later to establish connectivity.
g = zeros(size(f));
MARKER = zeros(size(f));
% Begin the merging stage.
for k = 1:Lmax
      [vals, r, c] = qtgetblk(f, s, k);
      if ~isempty(vals)
            % Check the predicate for each of the regions
            % of size k-by-k with coordinates given by vectors
            % r and c.
            for I = 1:length(r)
                  xlow = r(I); ylow = c(I);
                  xhigh = xlow + k – 1; yhigh = ylow + k – 1;
                  region = f(xlow:xhigh, ylow:yhigh);
                  flag = feval(fun, region);
                  if flag
                        g(xlow:xhigh, ylow:yhigh) = 1;
                        MARKER(xlow, ylow) = 1;
                  end
            end
      end
end
% Finally, obtain each connected region and label it with a
% different integer value using function bwlabel.
g = bwlabel(imreconstruct(MARKER, g));

% Crop and exit
g = g(1:M, 1:N);
%-------------------------------------------------------------------------------
----%
function v = split_test(B, mindim, fun)
% THIS FUNCTION IS PART OF FUNCTION SPLIT-MERGE. IT DETERMINES
% WHETHER QUADREGIONS ARE SPLIT. The function return in v
% logical 1s (TRUE) for the blocks that should be split and
% logical 0s (FALSE) for those that should not.
```

```
% Quadregion B, passed by qtdecomp, is the current decomposition of
% the image into k blocks of size m-by-m.

% k is the number of regions in B at this point in the procedure.
k = size(B, 3);
% Perform the split test on each block. If the predicate function
% (fun) returns TRUE, the region is split, so we set the appropriate
% element of v to TRUE. Else, the appropriate element of v is set to
% FALSE.
v(1:k) = false;
for I = 1:k
      quadregion = B(:, :, I);
      if size(quadregion, 1) <= mindim
            v(I) = false;
            continue
      end
      flag = feval(function, quadregion);
      if flag
            v(I) = true;
      end
end
```

مثال ۷.۹: قطعه بندی تصویر با تقسیم بندی و ترکیب ناحیه‌ها:

حلقه صورت فلکی قو که با اشعه x برداشته شده در عکس ۷.۱۷.(a) دیده می‌شود. این عکس به اندازه ۲۵۶*۲۵۶ است. هدف از این مثال قطعه بندی حلقه‌ای با چگالی کمتر پیرامون کانون متراکم است. ناحیه مورد نظر خصوصیاتی دارد که به قطعه بندی آن کمک می‌کند. ابتدا مشخص می‌کنیم که این داده‌های تصادفی هستند بنا بر این انحراف معیار آنها باید بیشتر از انحراف معیار پس زمینه (که صفر است)، و ناحیه کانونی باشد. میانگین شدت ناحیه حاوی داده‌های حلقه بیرونی باید بیشتر از میانگین پس زمینه (که صفر است) و کمتر از میانگین ناحیه کانونی باشد. بنا بر این می‌توان با استفاده از این دو پارامتر ناحیه مورد نظر را قطعه بندی کرد. تابع محمولی که به صورت مثال در مطالب مستند تابع splitmerge بیان شد حاوی اطلاعاتی درباره این مسئله است. پارامترها با محاسبه انحراف معیار و میانگین نقاط گوناگون عکس ۷.۱۷.(a) تعیین شدند.



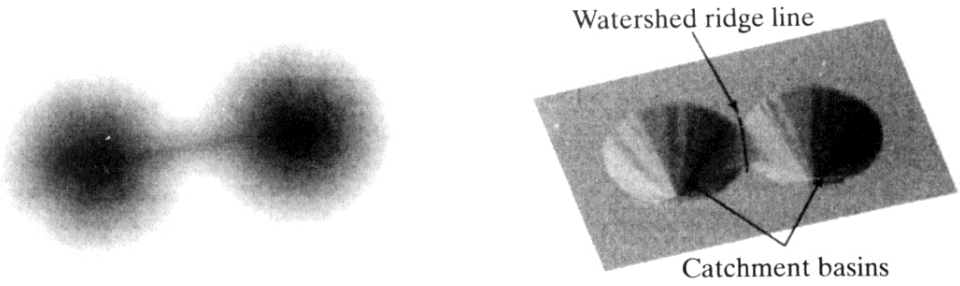
تصویر ۷.۱۷: قطعه بندی تصویر به روش ترکیب و تقسیم بندی (b) تصویر اصلی (c) نتایج قطعه بندی با استفاده از تابع spliTmerge با مقادیر mindim به ترتیب معادل ۳۲، ۱۶، ۸، ۴ و ۲

در عکس ۷.۱۷(b) نتایج قطعه بندی عکس ۷.۱۷.(a) با تابع splitmerge و مقادیر mindim به ترتیب ۳۲، ۱۶، ۸، ۴ . ۲ ارائه شده است. نتایج قطعه بندی در تمام تصاویر متناسب با مقدار mindim است.

کلیه نتایج عکس ۷.۱۷ قطعه بندی معقول هستند. اگر یکی از این ناحیه‌ها به عنوان نقابی برای استخراج ناحیه مورد نظر از تصویر اصلی به کار برده شود، در این صورت، نتایج عکس ۷.۱۷.(d)بهترین گزینه هستند زیرا ناحیه جامد با بیشترین جزئیات است. یک جنبه مهم این روش آن است که می‌تواند اطلاعات دامنه مسئله‌دار را از تابع predicaTe ضبط کند و به روند قطعه بندی کمک کند.

۷.۵. قطعه بندی با استفاده از تبدیل شیب‌دار(Segmentation Using the Watershed Transformation)

عبارتهای حوضه آبگیر و جلگه رودخانه در مورد ناحیه‌هایی به کار می‌روند که آب آنها توسط رودخانه‌های گوناگون تخلیه می‌شود. حوضه آبگیر(catchment basin)ناحیه‌ای است که آب آن از رودخانه یا مخزن است. عبارت تبدیل ناحیه تجمع(watershed transform) از این مفاهیم برای پردازش تصویرهایی با مقیاس خاکستری استفاده می‌کند طوری که بتوان انواع مسائل قطعه بندی تصاویر را با آن حل کرد.



تصویر ۷.۱۸: (a)) تصویر مقیاس خاکستری حبابهای تیره (b) تصویر مشاهده شده از سطح با خطوط شیبدار و حوضه تجمع مواد

برای درک تبدیل شیبدار باید یک تصویر مقیاس خاکستری را همچون سطحی مه تحلیل موضعی شده است در نظر بگیریم طوری که مقادیر f(x y) به صورت ارتفاع تفسیر شده باشند. به این ترتیب می‌توان تصویر ساده عکس ۷.۱۸.(a) را به صورت سطح ۳ بعدی در تصویر ۷.۱۸ (b)در نظر گرفت. اگر تصور کنیم باران روی این سطح می‌ریزد، پر واضح است که آب در دو ناحیه که حوضه آبگیر نامیده می‌شوند جمع می‌شود. بارانی که دقیقاً روی لبه خط حوضه آبگیر می‌ریزد، در یکی از دو حوضه آبگیر جمع می‌شود. در این روش تبدیل ابتدا حوضه

آبگیر و خطوط برآمده در مقیاس خاکستری پیدا می‌شوند. برای حل مسائل قطعه بندی تصویرها ابتدا تصویر مقدماتی به تصویری دیگر تبدیل می‌شود که حوضه آبگیر آن اشیا و ناحیه‌هایی هستند که می‌خواهیم شناسایی کنیم.

Gonzalez & Woods[2002] روشهای محاسبه این نوع تبدیل را به طور مفصل شرح داده‌اند. Vinsent & Soille[2003] الگوریتم مورد استفاده در IPT را ابداع کرده‌اند.

۷.۵.۱. قطعه بندی حوضه آبگیر با استفاده از تبدیل مسافت

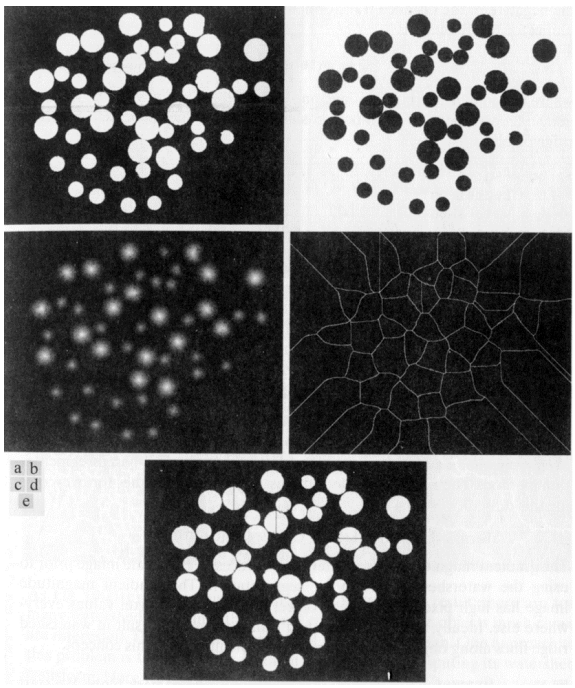
(Watershed Segmentation Using the Distance Transformation)

ابزاری که معمولاً در تبدیل حوضه آبگیر در قطعه بندی استفاده می‌شود تبدیل مسافت است. تبدیل مسافت(distance transform) یک تصویر دودویی کار ساده‌ای است. این مسافت هر عنصر تصویری تا نزدیکترین عنصر حاوی مقداری غیر از صفر است. تبدیل مسافت در تصویر ۷.۱۹ دیده می‌شود. یک ماتریس تصویری دودویی کوچک در تصویر ۷.۱۹.(a) دیده می‌شود. تبدیل مطابق با آن در تصویر ۷.۱۹.bدیده می‌شود. توجه داشته باشید که تبدیل مسافت عناصر تصویری دارای یک مقدار صفر است. تبدیل مسافت را می‌توان با تابع bwdisT در IPT محاسبه کرد که ترکیب فراخوان آن به شرح زیر است:

```
D = bwdist(f)
```

1	1	0	0	0	0.00	0.00	1.00	2.00	3.00
1	1	0	0	0	0.00	0.00	1.00	2.00	3.00
0	0	0	0	0	1.00	1.00	1.41	2.00	2.24
0	0	0	0	0	1.41	1.00	1.00	1.00	1.41
0	1	1	1	0	1.00	0.00	0.00	0.00	1.00

تصویر ۷.۱۹: (a)) تصویر دودویی کوچک (b) تبدیل مسافت



تصویر ۷.۲۰: (a)) تصویر دودویی (b) مکمل تصویر (c) تبدیل مسافت ت) خطوط برجسته منفی تبدیل مسافت (e) خطوط برجسته قرار داده شده روی تصویر دودویی اصلی قطعه بندی افراطی دیده می‌شود.

مثال ۷.۷: قطعه بندی یک تصویر دودویی با استفاده از تبدیلهای مسافت و حوضه آبگیر:

در این مثال نحوه استفاده از تبدیل مسافت با تبدیل حوضه آبگیر IPT برای قطعه بندی حبابهای مدور دیده می‌شود که برخی از آنها با یکدیگر تماس دارند. در اینجا می‌خواهیم تصویر میخ چوبی f را که در عکس b.۹.۲۹دیده می‌شود پیش پردازش کنیم. ابتدا همان طور که در بخش ۷.۳.۱ گفتیم تصویر را با استفاده از im2bw و grayThresh به نوع دودویی تبدیل می‌کنیم.

```
>> g = im2bw(f, graythresh(f));
```

نتیجه در عکس (a).۷.۲ نشان داده شده است. مراحل بعدی عبارتند از تکمیل تصویر، محاسبه تبدیل مسافت آن، و محاسبه تبدیل حوضه منفی تبدیل مسافت، با استفاده از تابع waTershed

ترکیب فراخوانی این تابع به شرح زیر است:

```
L = watershed(f)
```

L ماتریس برچسب(Laber Matrix) است. اعداد صحیح مثبت L مطابق با حوضه آبگیر هستند، و مقادیر صفر نمایانگر عناصر تصویری خطوط برجسته هستند.

```
>> gc = ~g;
>> D = bwdista(gc);
>> L = watershed(-D);
>> w = L = = 0;
```

تصویر تکمیل شده و تبدیل مسافت آن در عکسهای b.۷.۲۰و (c).۷.۲۰نشان داده شده است. از آنجائی که آن دسته از عناصر تصویریL که صفر هستند همان پیکسلهای برجسته تجمع یافته هستند، در خط آخر رمز قبل تصویر دودویی W محاسبه می‌شود که فقط این عناصر تصویری را نشان می‌دهد. این تصویر دارای برآمدگی‌های حوضچه‌ای شکل در عکس (d).۷.۲۰ نشان داده شده است. خصوصیات منطقی، تصویر دودویی اصلی، و مقدار مکمل W باعث تکمیل قطعه بندی می‌شود که در عکس (e).۷.۲۰ نشان داده شده است.

```
>> g2 = g & -w;
```

توجه داشته باشید که برخی اشیای تصویر (e).۷.۲۰نادرست تقسیم شده‌اند. این را قطعه بندی افراطی(over segmentation) می‌نامیم و مسئله‌ای است که زیاد در روشهای قطعه بندی پیش می‌اید. روشهای گوناگون غلبه بر این مشکل دردو بخش بعد قید شده است.

۷.۵.۱. قطعه بندی حوضچه‌ای شکل با استفاده از ضریب زاویه(Watershed Segmentation Using Gradiants)

قبل از استفاده از تبدیل حوضچه‌ای شکل برای قطعه بندی از دامنه ضریب زاویه برای پیش پردازش تصویر با مقیاس خاکستری استفاده می‌کنیم. مقادیر عناصر تصویری دامنه ضریب زاویه در امتداد لبه اشیاء بالا و در جاهای دیگر پایین است. در نتیجه تبدیل حوضچه‌ای شکل منجر به تشکیل خطوط برجسته در امتداد لبه اشیا می‌شود. این موضوع در مثال بعد به تصویر کشیده شده است.

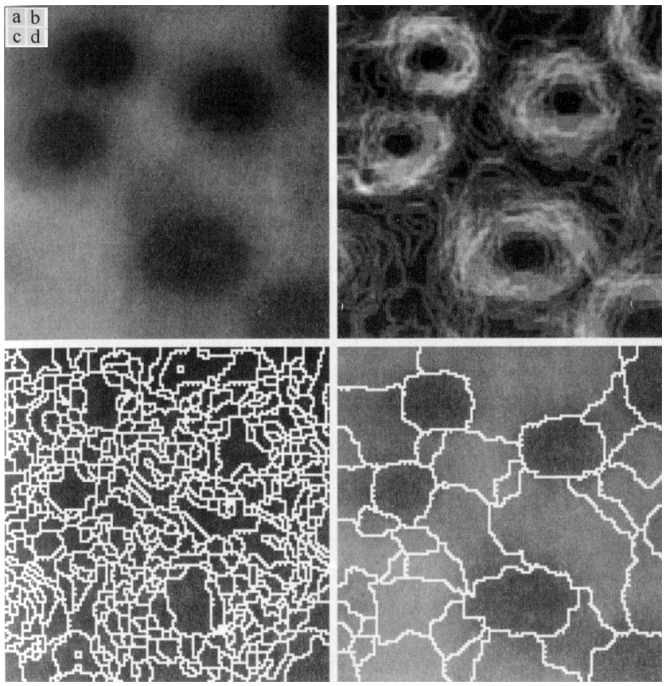
مثال ۷.۱۱:

تصویر (f)حاوی چند حباب تیره در عکس ۷.۲۱.(a) به تصویر کشیده شده است. ابتدا دامنه ضریب زاویه آن را با استفاده از روش فیل خطی بخش ۷.۱ و یا شکل ضریب زاویه که در بخش ۹.۶.۱ توصیف شد محاسبه می‌کنیم.

```
>> h = fspecial('sobel');
>> fd = double(f);
>> g = sqrt(imfilter(fd, h, 'replicate') .^ 2 + …
            imfilter(fd, h, 'replicate') .^ 2);
```

دامنه ضریب زاویه تصویر gدر عکس ۷.۲۱(b) نشان داده شده است. بعد تبدیل حوضچه‌ای شکل ضریب زاویه را محاسبه می‌کنیم و لبه‌های برجسته حوضچه را پیدا می‌کنیم.

```
>> L = watershed(g);
>> wr = L == 0;
```



تصویر ۷.۲۱: (a)) تصویر مقیاس خاکستری حبابهای کوچک (b) تصویر دامنه ضریب زاویه (c) تبدیل حوضچه bکه قطعه بندی افراطی در آن دیده می‌شود (d) تبدیل حوضچه تصویر با ضریب زاویه هموار قطعه بندی افراطی هنوز در آن دیده می‌شود.

نتیجه قطع در عکس ۷.۲۱.(c)مطلوب نیست. تعداد خطوط برجسته زیاد است و مطابق با اشیای مورد نظر ما نیست. این مثال دیگری از قطعه بندی افراطی است. یک راه حل این مسئله هموارسازی تصویر ضریب زاویه قبل از محاسبه تبدیل حوضچه‌ای شکل آن است.

```
>> g2 = imclose(imopen(g, ones(3, 3)), ones(3, 3));
>> L2 = watershed(g2);
>> wr2 = L2 == 0;
>> f2 = f;
>> f2(wr2) = 255;
```

۲ خط آخر در رمز قبلی خطوط برجسته حوضچه‌ای شکل را به صورت خطوط سفید در تصویر اصلی روی هم قرار می‌دهند. مقادیر متناظر در عکس ۷.۲۱.(d)دیده می‌شود. گرچه عکس ۷.۲۱.(c)بهبود یافته است ولی هنوز خطوطی برجسته خارج از آن است و مشکل است که بگوییم کدام یک از حوضچه‌ها با اشیای مورد نظر ارتباط دارند. بخش بعد مربوط به بهبود قطعه بندی حوضچها است که به این دشواریها مربوط می‌شود.

۷.۵.۳ کنترل قطعه بندی حوضچه با نشانه گذار(Marker_Controlled Watershed Segmentation)

اجرای تبدیل حوضچه‌ای روی تصویر زاویه‌دار به دلیل پارازیت‌ها و سایر ناهمواری‌های ضریب زاویه منجر به قطعه بندی افراطی می‌شود. مسائل بعد از آن آنقدر جدی هستند که نتیجه را ناکارآمد می‌کنند. بنا بر این تعداد زیادی از ناحیه‌ها باید قطعه بندی شوند. یک راه حل این مسئله محدود کردن تعداد ناحیه‌های مجاز با استفاده از سایر اطلاعات در فرایند قطعه بندی است.

یک روش برای کنترل قطعه بندی استفاده از نشانه‌گذارها است. نشانه‌گذار(Marker) یکی از مولفه‌های وابستگی به تصویر است. نشانه‌گذارهای داخلی(internal marker) آنهایی هستند که در داخل اشیای مورد نظر هستند. همین طور برخی نشانه‌گذارهای بیرونی که در پس زمینه قرار دارند. از این نشانه‌گذارها برای تغییر تصویر زاویه‌دار با استفاده از روش توصیف شده در مثال ۷.۱۲ بهره‌برداری می‌شود. روشهای گوناگون برای محاسبه نشانه‌گذارهای داخلی و بیرونی وجود دارد که در اکثر آنها باید از روشهای فیلترسازی خطی و غیرخطی و پردازش شکل که در فصل قبل تشریح شد استفاده شود. روش انتخاب بستگی به ماهیت تصاویر مربوط به برنامه دارد.

مثال ۷.۱۲: تصویری از مهار قطعه بندی حوضچه‌ای شکل با نشانه‌گذار:

در این مثال کنترل قطعه بندی حوضچه‌ای شکل با نشانه‌گذار در آبکاری با جابجایی الکترونها در تصویر ۷.۲۲.(a) دیده می‌شود. برای انجام این کار نتایج محاسبه تبدیل حوضچه‌ای شکل تصویر زاویه‌دار را بدون پردازش دیگری به دست می‌آوریم.

```
>> h = fspecial('sobel');
>> fd = double(f);
>> g = sqrt(imfilter(fd, h, 'replicate') .^ 2 + …
           imfilter(fd, h, 'replicate') .^ 2);
>> L = watershed(g);
>> wr = L == 0;
```

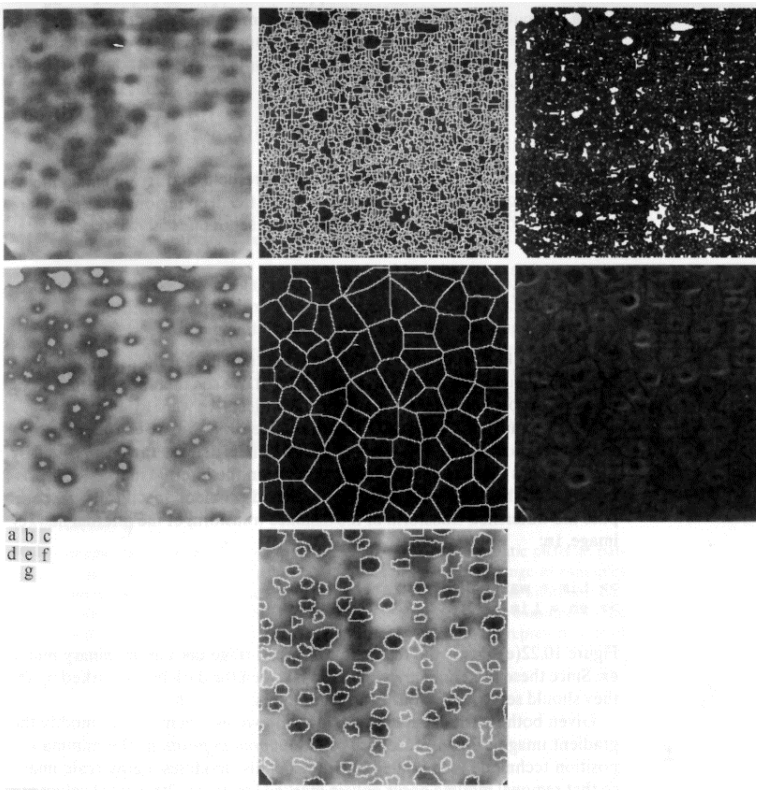
نتیجه در عکس b.۷.۲۲بیش از حد قطعه بندی شده است. علت آن تا حدی مربوط به تعداد زیاد کمینه‌های موضعی است. ترکیب فراخوان آن به شرح زیر است:

```
rm = imregionalmin(f)
```

(f)یک تصویر مقیاس خاکستری است و rf یک تصویر دودویی است که عناصر پیش زمینه آن کمینه‌های موضعی را علامت‌گذاری کرده‌اند. برای این که ببینیم چرا تابع حوضچه‌ای اینقدر حوضچه‌های کوچک ایجاد می‌کند می‌توان از imregionalmin در تصویر زاویه‌دار استفاده کرد.

```
>> rm = imregionalmin(g);
```

اکثر کمینه‌های موضعی عکس c).۷.۲۲)خیلی کم عمق هستند و جزئیاتی دارند که به مسئله قطعه بندی نامرتبط است. برای حذف این کمینه‌های بیرونی در IPT از تابع imextendedmin استفاده می‌کنیم.



تصویر ۷.۲۲: (a)) تصویر ژله‌ای ۲) قطعه بندی افراطی به دلیل اعمال تبدیل حوضچه‌ای به تصویر زاویه‌دار (c) کمیت موضعی با دامنه شیبدار (d) نشانه‌گذاریهای داخلی (e) نشانه‌گذاریهای بیرونی ج) حیطه تغییر یافته ضریب زاویه چ) نتیجه قطعه بندی

بنا بر این نقاط پست تصویر(low spots) که نسبت به آستانه ارتفاع پایینتر از نقاط پیرامون خود هستند محاسبه می‌شود. برای بررسی ادامه تبدیل کمینه‌ها(minima) و عملیات مرتبط به مطالب[Soill[2003 مراجعه کنید. ترکیب فراخوان این تابع به شرح زیر است:

```
im = imextendedmin(f, h)
```

(f)یک تصویر مقیاس خاکستری است. h استانه ارتفاع است.(e)ام یک تصویر دودویی است که عناصر تصویری پیش زمینه آن کمینه‌های موضعی عمیق را نشانه‌گذاری می‌کنند. در اینجا برای به دست آوردن مجموعه نشانه‌گذارهای داخلی از تابع imexTendedmin استفاده می‌کنیم.

```
>> im = imextendedmin(f, 2);
>> fim = f;
>> fim(im) = 175;
```

همان طور که در عکس ۷.۲۲.(d)دیده می‌شود در ۲ خط آخر کمینه‌ها به صورت حبابهای خاکستری روی تصویر اصلی قرار گرفته‌اند. حبابهای حاصل از آن، اشیایی را که می‌خواهیم قطعه بندی کنیم به خوبی علامت‌گذاری می‌کنند.

سپس نشانه‌گذارهای بیرونی یا عناصر تصویری را می‌یابیم که مطمئن هستیم به پس زمینه تعلق دارند. در این روش پس زمینه را با پیکسلهایی علامت‌گذاری می‌کنیم که دقیقاً در نقطه وسط بین نشانه‌گذارهای داخلی قرار دارند. برای انجام این کار یک مسئله دیگر حوضچه‌ها را باید حل کنیم. تبدیل حوضچه‌ای تبدیل مسافت تصویر نشانه‌گذار داخلی را باید محاسبه کنیم.

```
>> Lim = watershed(bwdist(im));
>> em = lim ==0;
```

خطوط برجسته تصویر دودویی em در عکس ۷.۲۲.(e)دیده می‌شود. از آنجائی که این خطوط در وسط حبابهای مشکی im قرار دارند به خوبی می‌توانند نقش نشانه‌گذارهای بیرونی را ایفا کنند.

حالا با استفاده از نشانه‌گذارهای بیرونی و داخلی شیب زاویه را با استفاده از روشی به نام تداخل کمینه تغییر می‌دهیم.

در این روش یک تصویر مقیاس خاکستری طوری تغییر می‌کند که کمینه‌های موضعی فقط در قسمت‌های علامت‌گذاری شده دیده می‌شوند. سایر مقادیر پیکسلها برای برطرف کردن کمینه‌های موضعی زیاد می‌شود. تابع imimposemin این کار را انجام می‌دهد و ترکیب فراخوانی آن به شرح زیر است:

```
mp = imimposemin(f, mask)
```

در اینجا (f)یک تصویر مقیاس خاکستری است و mask یک تصویر دودویی است که عناصر پیش زمینه آن کمینه‌های موضعی تصویر خروجی mp را علامت‌گذاری می‌کنند.

با قرار دادن کمینه‌های موضعی در محل نشانه‌گذارهای بیرونی و داخلی تصویر زاویه‌دار را تغییر می‌دهیم.

```
>> g2 = imimposemin(g, im | em);
```

نتیجه در عکس ۷.۲۲.(f)نشان داده شده است. حالا می‌توانیم تبدیل حوضچه‌ای تصویر زاویه‌دار را که با نشانه‌گذار تغییر داده شده است محاسبه کنیم و به خطوط برآمده حاصل از آن نگاه کنیم.

```
>> L2 = watershed(g2);
>> f2 = f;
>> f2(L2 == 0) = 255;
```

۲ خط آخر خطوط برجسته حوضچه‌ای شکل را روی تصویر اصلی قرار می‌دهند. نتیجه به صورت قطعه بندی بهبود یافته در عکس ۷.۲۲.gنشان داده شده است.

انتخاب نشانه‌گذار با روشهای ساده و پیچیده که به اندازه، شکل، محل، فاصله نسبی، بافت و غیره توجه می‌شود انجام می‌شوند. اگر از نشانه‌گذار استفاده شود اطلاعاتی پیشاپیش برای حل مسئله قطعه بندی داریم. هومنز برای حل مسائل قطعه بندی و سطوح بالا از اطلاعات بدون تحقیقات قبلی استفاده می‌کرد. نمونه بارز آن استفاده از محتوا یا بافت است. یکی از مزایای بزرگ این روش آن است که قطعه بندی با این حوضچه‌ها زمینه‌ای برای بهره‌برداری از این نوع اطلاعات ایجاد می‌کند.

خلاصه مطالب(Summary)

قطعه بندی تصاویر مرحله‌ای اساسی در اکثر مسائل تشخیص الگوهای تصویری و تحلیل صحنه‌ها است. همان طور که در مثالهای این بخش گفتیم انتخاب روش قطعه بندی بستگی به خصوصیات مسئله مورد نظر دارد. گرچه روشهای این بخش کامل نیستند ولی نمایانگر شگردهای کاربردی متداول هستند.