An Introduction to Computer Vision Algorithms

By:

Mehdi Safarpour,
Akbar Jafari,
Mostafa Charmi,
Saeed Nasehi

مقدمه‌ای بر بینایی کامپیوتری

نویسندگان :

مهدی صفرپور

اکبر جعفری

دکتر مصطفی چرمی

سعید ناصحی بشرزاد

با سپاس ازسه وجود مقدس:

آنان که ناتوان شدند تا ما به توانایی برسیم ...

موهایشان سپید شد تا ماروسفید شویم ...

و عاشقانه سوختند تا گرمابخش وجود ما و روشنگر راهمان باشند ...

پدرانمان

مادرانمان

استادانمان

مقدمه

دوربین‌های دیجیتال همه‌ی دنیای اطراف انسان‌ها را زیر نظر گرفته اند.هرجا که دوربینی باشد، می‌توان پردازنده‌ای نیز قرار داد که اطلاعات دوربین را تفسیر کند و در صورت لزوم ابزاری را کنترل کند. روش‌ها و الگورتیم‌های که با استفاده از آنها تصویر گرفته شده توسط یک دوربین با یک پردازنده‌ی دیجیتال تفسیر می‌شوند را به طور عمومی پردازش تصویر و بینایی کامپیوتر (و گاها بینایی ماشین) می‌گویند. .بینایی کامپیوتر و پردازش تصویر کاربرد گسترده‌ای در صنعت دارد. سیستم‌های نظارت و بازرسی اتوماتیک، ناوبری و هدایت روباتیک، تشخیص و درک تصویر و ده‌ها کاربرد دیگر را به طور روزمره در زندگی اطراف خود مشاهده می‌کنیم. از طرفی این مباحث در دنیای آکادمیک هم جایگاه تثبیت‌شده ای یافته‌اند. به همین دلیل نیاز به منابع فراگیری این تکنیک‌ها احساس می‌شود. این کتاب، گردآوری مباحثی است که به نظر نویسندگان، شالوده‌ی اصلی بینایی کامپیوتری را تشکیل می‌دهند. از خواننده‌ی این کتاب انتظار می‌رود با مفاهیم و تکنیک‌های پایه‌ای پردازش تصویر آشنایی داشته باشد و حداقل به یکی از زبان‌های برنامه‌نویسی مسلط باشد. فصول این کتاب تقریبا مستقل از هم هستند و هریک را می‌توان با توجه به نیاز، مستقل از فصول دیگر مطالعه کرد. در آخر ما نویسندگان، امیدواریم با ارائه‌ی این کتاب تاثیری مثبتی (هرچند اندک) بر پیشرفت علمی شما خواننده‌ی گرامی داشته باشیم.

هرظرفی با جای دادن چیزی درآن پرمی‌شود، جزظرف دانش که به جای دادن، فزونی یابد. امام علی (ع)

فهرست مطالب

فصل ۱

هندسه‌ی تصویر برداری

۱.۱ معرفی

در اولین فصل از این کتاب ما راجع به هندسه‌ی تصویر بـرداری خـواهیم آموخـت. تبـدیلات مختلفـی را کـه در مسائل مختلف در بینایی ماشین استفاده خواهند را بحث خواهیم کرد. این تبدیلات در گرافیک کامپیوتری هم کـار برد دارند. راجع سیستم‌های مختصات یاد خواهیم گرفت و اینکه چگونه می‌توان یک سیستم مختصات را با سیستم مختصات دیگری مرتبط کرد. در این فصل تبدیلات انتقال،چرخش، تغییر مقیاس و پرسپکتیو را بحث خواهیم کرد. این تبدیلات به ما کمک می‌کنند که موقعیت و جهت قرارگیری آن را وقتی که انتقال می‌یابد، چرخانده می‌شـود و یا تغییر مقیاس داده می‌شود را بدانیم.یک ماتریس برای هریک از این تبدیلات استخراج خواهیم کرد.

دنیایی که ما در آن زندگی می‌کنیم سه بعدی است، یعنی هر نقطه در فضا می‌تواند سه نقطه‌ی X، Y وZ مشـخص شود. تصویر یک صفحه‌ی دو بعدی است، بنابراین به مختصات دو تایی X وY برای نمایش یک نقطه در تصـویر نیاز داریم. یک بعد در فرآیند تصویر برداری از بین می‌رود. یکی از اهداف مهم بینایی کامپیوتری بازیابی ایـن بعـد گمشده است. متدهایی که برای بازیابی اطلاعات سه بعدی از تصاویر دو بعدی استفاده می‌شود روش‌های شکل از (X) نام دارند که (X) می‌تواند استریو (stereo)، حرکت (motion)، سایه زنی (shading)، بافـت (texture) و.... باشد. این متدها را به صورت مفصل در این کتاب بحث خواهیم کرد. با این حال در این فصل روی استخراج ماتریس دوربین و کالیبراسیون دوربین تمرکز خواهیم کرد.

۱.۲ انتقال و تغییر مقیاس

یک نقطه را روی یک شی با مختصات (X1 ,Y1 , Z1) در نظر بگیرید. فرض کنید کـه شـی بـه انـدازه dx، dy وdz به ترتیب در جهات X، Y و Z جابجا شده است. مختصات جدید نقطه به صورت زیر می‌شود:

$$X2 = X1 + dx \qquad (1.1)$$
$$Y2 = Y1 + dy \qquad (1.2)$$
$$Z2 = Z1 + dz \qquad (1.3)$$

این معادلات را می‌توان به فرم ماتریسی به صورت زیر نوشت

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \qquad (1.4)$$

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = T \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \qquad (1.5)$$

که $T = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$ است و ماتریس انتقال نامیده می‌شود. ماتریس تبدیل معکوس به صورت زیر است.

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -dx \\ 0 & 1 & 0 & -dy \\ 0 & 0 & 1 & -dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

خودتان می‌توانید چک کنید که $TT^{-1} = T^{-1}T = I$ می‌شود (I ماتریس همانی می‌باشـد).به طـرز مشـابهی اگر یک شی به اندازه S_x و S_y و S_z در راستاهای X، Y و Z به ترتیب تغییر مقیاس پیدا کند مختصات جدید بـه صورت زیر به دست می‌آیند:

$$X2 = X1 \times S_x \qquad (1.6)$$
$$Y2 = Y1 \times S_y \qquad (1.7)$$
$$Z2 = Z1 \times S_z \qquad (1.8)$$

این معادلات را می‌توان به صورت زیر به فرم ماتریسی نوشت:

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \qquad (1.9)$$

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = S \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \qquad (1.10)$$

که $S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ ماتریس تغییر مقیاس نام دارد. عکس ماتریس تغییر مقیاس نیز به صورت

$S^{-1} = \begin{bmatrix} \frac{1}{S_x} & 0 & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 & 0 \\ 0 & 0 & \frac{1}{S_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ است.

۱.۳ چرخش

برداری را فرض کنید که به (X1, Y1 , Z1) اشاره می‌کند (شکل ۱.۱ الف) فرض کنید R طول بردار و Y زاویه‌ی بردار V با محور X را نشان می‌دهند. فرض کنید که بردار حول محور Z به اندازه θ در جهت خلاف عقربه‌های ساعت چرخانده می‌شود. مختصات نقطه جدید (X2, Y2 , Z2) پس از چرخش از نقطه‌ی مختصات قبلی و اندازه‌ی چرخش به دست می‌آید. مثلث شکل ۱.۱ الف را در نظر بگیرید که ضلع اش با محور Xها زاویه ی φرا می‌یابد. با استفاده از روابط مثلثاتی می‌توانیم بنویسیم:

$$X1 = R\cos\varphi \qquad (1.12)$$
$$Y1 = R\sin\varphi \qquad (1.13)$$

به طرز مشابهی برای مثلث شکل ۱.۱ الف که ضلع اش یک زاویه‌ی $\varphi + \theta$ با محور X می‌سازد می‌توانیم بنویسیم:

$$X2 = R\cos(\theta+\varphi) = R\cos\theta\cos\varphi + R\sin\theta\sin\varphi \qquad (1.14)$$
$$Y2 = R\sin(\theta+\varphi) = R\sin\theta\cos\varphi + R\cos\theta\sin\varphi \qquad (1.15)$$

با جایگذاری دو معادله‌ی بالاتر در دو معادله‌ی اخیر به دست می‌آوریم:

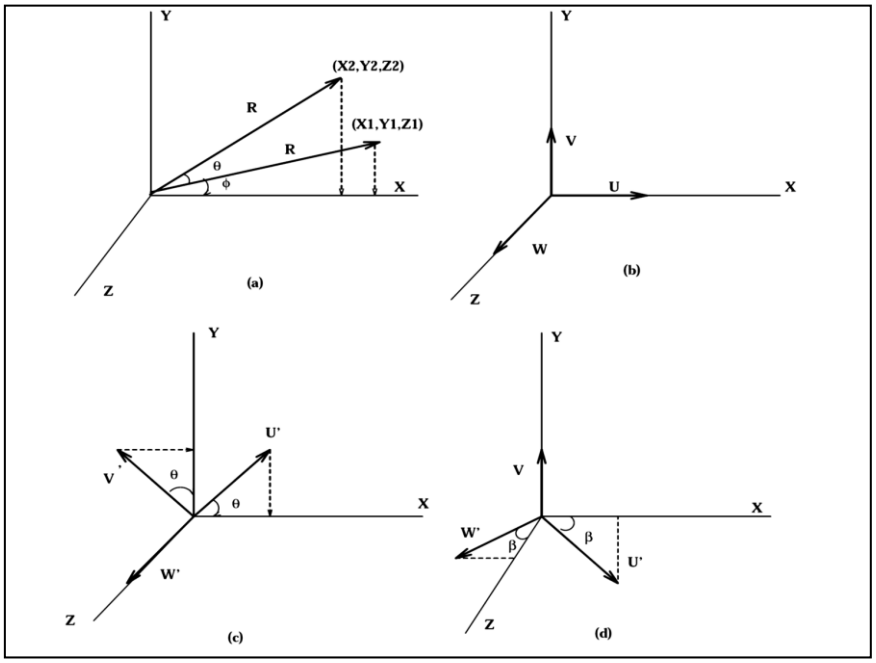
$$X2 = X1\cos\theta - Y1\sin\theta \qquad (1.16)$$
$$Y2 = X1\sin\theta + Y1\cos\theta \qquad (1.17)$$

این معادلات را می‌توان به فرم ماتریسی به صورت زیر نوشت:

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \qquad (1.18)$$

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = R_\theta^Z \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{1.19}$$

که $R_\theta^Z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ و ماتریس چرخش است. از یک بالا نویس برای مشخص کردن محور چرخش و از یک زیر نویس برای مشخص کردن زاویه‌ی چرخش استفاده خواهیم کرد و همچنین فرض می‌کنیم که چرخش در خلاف جهت حرکت عقربه‌های ساعت است. عکس ماتریس چرخش به صورت $(R_\theta^Z)^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ می‌باشد. عکس چرخش به اندازه‌ی θ معادل چرخش به اندازه‌ی $-\theta$ حول همان محور است. از این رو $(R_\theta^Z)^{-1} = R_Z^{-\theta}$.علاوه بر این، ماتریس چرخش یک ماتریس متعامد بهنجار (Orthonormal) است، یعنی $(R_\theta^Z)^T R_\theta^Z = R_\theta^Z (R_\theta^Z)^T = I$ از این رو عکس ماتریس R در واقع ترانهاده‌ی آن است.



شکل ۱.۱ چرخش (الف) چرخش حول محور Z. (ب) سیستم مختصات UYZ. (ج)UVWچرخانده شده به اندازه‌ی θ حول محور Z در جهت خلاف عقربه‌های ساعت. (د) UVW چرخانده شده به اندازه‌ی B حول محور Y در جهت عقربه‌های ساعت.

راه دیگر استخراج ماتریس چرخش در نظر گرفتن دو سیستم مختصات (X،Y،Z) و (U،V،Z) (مشابه آنچه در شکل ۱.۱ ب نشان داده شده است) می‌باشد. فرض کنید سیستم مختصات UVW را حول محور Z به اندازه‌ی θ به صورت نشان داده شده در شکل ۱.۱.ج می‌چرخانیم. توجه کنید که محور W روی محورZ ثابت می‌ماند. با این حال محورهای U و V تغییر می‌کنند. محورهای جدید U'، V' و W' را می‌توان بر حسب محورهای قدیمیW،V،U نوشت. محور V' دو مؤلفه دارد، یکی در طول محور X و دیگری در طول محور Y. توجه کنید که مؤلفه‌ی Z ندارد. بردار جدید V' به صورت $(\cos\theta\ ,\sin\theta\ ,0)$ معین می‌شود. مشابهاً، بردار V' به صورت $(-\sin\theta\ ,\cos\theta\ ,0)$ فرض می‌شود. از آن جایی که بردار W تغییر نکرده است به صورت (0 , 0 , 1) ثابت می‌ماند. حالا ماتریس چرخش می‌تواند با قرار دادن مختصات بردارهای U'، V' و W' به ترتیب در اولین و دومین و سومین ستون و یک بردار(0 , 0 , 0 , 1) در ستون چهارم تشکیل شود. ماتریس حاصله دقیقاً همانند ماتریس R_θ^Z است.

این تکنیک می‌تواند برای استخراج ماتریس چرخش حول هر محوری استفاده شود. برای مثال، چرخش حول محور Y در جهت ساعت گرد به صورت زیر به دست می‌آید (شکل ۱.۱ د)را ببینید:

$$R_{-\beta}^{Y}=\begin{bmatrix}\cos\beta & 0 & -\sin\beta & 0\\ 0 & 1 & 0 & 0\\ \sin\beta & 0 & \cos\beta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

۱.۴ تبدیل پرسپکتیو

مدل ساده‌ی شده‌ی فرآیند تشکیل تصویر در شکل ۱.۲ الف نمایش داده شده است. در مدل دوربین سوراخ دار (همان دوربینی که جعبه را سوراخ می‌کردیم و یک تصویر در جعبه تشکیل می‌شد) عدسی یک نقطه فرض می‌شود.

نقاط جهانی (X،Y،Z) از یک نقطه به تصویر مختصات تصویر تحت تبدیل پرسپکتیو تبدیل می‌شوند. فرض کنید که مبدأ سیستم مختصات مرکز دوربین باشد (که با O در شکل ۱.۲ الف نشان داده شده است)و f فاصله‌ی کانونی دوربین باشد (فاصله‌ی از عدسی تا صفحه‌ی تصویر). با استفاده از معادله تشابه مثلث‌ها می‌توانیم بنویسیم:

$$\frac{-y}{Y}=\frac{f}{Z} \qquad (1.20)$$

از آن جایی که تصویر همیشه از بالا به پایین تشکیل می‌شود، مرسوم است که یک علامت منفی جلوی y در عبارت بالا استفاده شود. از معادله‌ی بالا مختصه‌ی y تصویر زیر به دست می‌آید:

$$y = \frac{-f\,y}{Z} \tag{1.21}$$

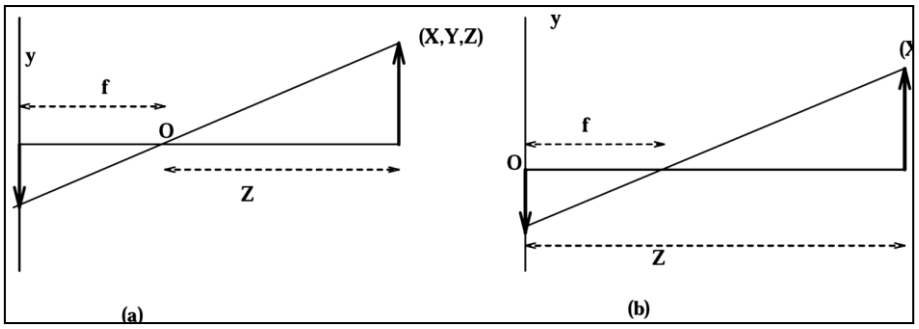
مشابهاً مختصه‌ی x تصویر به این صورت به دست می‌آید:

$$x = \frac{-f\,X}{Z} \tag{1.22}$$

معادلات بالا نشان دهنده‌ی تبدیل پرسپکتیو با مبدأ در عدسی هستند. اگر مبدأ مشابه آنچه در شکل ۱.۲ ب نشان داده شده،به سمت تصویر حرکت کند تبدیل پرسپکتیو به صورت به صورت معادلات زیر تعریف می‌شود:

$$x = \frac{f\,X}{f-Z} \tag{1.23}$$

$$y = \frac{f\,Z}{f-Z} \tag{1.24}$$



شکل ۱.۲: مدل دوربین (الف) مبدأ در عدسی (ب) مبدأ در صفحه‌ی عدسی تصویر

امکان استخراج یک ماتریس پرسپکتیو مشابه ماتریس‌های انتقال، تغییر مقیاس و دوران به خاطر طبیعت معادلات ۱.۲۳ و ۱.۲۴ وجود ندارد و ما نیاز به آوردن (معرفی) تبدیل دیگری به نام تبدیل همگن (Homogeneous) داریم. تبدیل همگن مختصات سیستم کارتزین (X،Y ،Z) را به مختصات سیستم همگن (KX،KY،KZ،K)تبدیل می‌کند. در این تبدیل هر مختصه‌ی X،Y و Z به یک ضریب ضرب می‌شود و آن ضریب (K) به عنوان چهارمین مؤلفه بردار ضمیمه می‌شود. متشابهاً، معکوس همگن مختصات همگن تصویر $(C_{h_1}, C_{h_2}, C_{h_3}, C_{h_4})$ را به مختصات کارتزین $(\frac{C_{h_1}}{C_{h_4}}, \frac{C_{h_2}}{C_{h_4}}, \frac{C_{h_3}}{C_{h_4}})$، با تقسیم همه مؤ لفه‌ی چهارم تبدیل می‌کند.

اکنون تبدیل پرسپکتیو می‌تواند به صورت زیر تعریف شود:

$$p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{f} & 1 \end{bmatrix}$$

تبدیل پرسپکتیو که سیستم مختصات جهانی همگن را با سیستم مختصات تصویری همگن مرتبط می‌کند به صورت زیر تعریف می‌شود.

$$\begin{bmatrix} kX \\ kY \\ kZ \\ -\frac{kZ}{f}+k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix}$$

به راحتی می‌توانیم مختصات کارتزین تصویر را از معادلات بالا به صورت زیر استخراج کنیم:

$$x = \frac{f\,X}{f-Z} \tag{1.25}$$

$$y = \frac{f\,Y}{f-Z} \tag{1.26}$$

که دقیقاً مثل معادلات ۱.۲۳ و ۱.۲۴ هستند. عکس تبدیل پرسپکتیو به صورت $P^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{bmatrix}$ می‌باشد.

۱.۵ مدل دوربین

تبدیل پرسپکتیو دو سیستم مختصات جهانی و سیستم مختصات تصویر را وقتی دوربین در مبدأ سیستم مختصات جهانی قرار گرفته مربوط می‌کند. با این حال در دنیای واقعی نیاز داریم که دوربین را جا به جا کنیم یا بچرخانیم تا تصویر شی مورد علاقه را به میدان دید دوربین بیاوریم. از این رو، مدل کامل دوربین شامل چندین تبدیل دیگر در کنار تبدیل پرسپکتیو می‌شود.

در یک مدل ساده، فرض کنید که دوربین ابتداً در مرکز مختصات جهانی قرار داشته و بعد به اندازه‌ی $(X_0\ , Y_0, Z_0)$ (ماتریس G) منتقل شده و سپس حول محور Zها به اندازه‌ی θدر خلاف جهت عقربه‌های ساعت (ماتریس $R_{-\theta}^{Z}$) و حول محور Xها به اندازه‌ی Y در خلاف جهت عقربه‌های ساعت چرخانده شود و بعد به اندازه‌ی $(r_1\ , r_2\ , r_3)$ دوباره انتقال داده شود. در این حالت سیستم مختصات جهانی (W_h) به صورت زیر به سیستم مختصات دوربین (C_h) مرتبط می‌شود:

$$C_h = PCR_{-\phi}^{X}R_{-\theta}^{Z}GW_h \tag{1.27}$$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{f} & 1 \end{bmatrix}$$

$$R_{-\theta}^{Z} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R_{-\phi}^{Z} = \begin{bmatrix} \cos\phi & \sin\phi & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & -r_0 \\ 0 & 1 & 0 & -r_0 \\ 0 & 0 & 1 & -r_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

توجه کنید که اینجا عکس تبدیلات برای انتقال و دوران استفاده شده اند. این به این خاطر است که سیستم مختصات مربوط به دوربین است که منتقل و چرخانده شده نه شی مورد نظر. اکنون مختصات کارتزین تصویر به صورت معادلات زیر محاسبه می‌شوند:

$$x = f\,\frac{(X-X_0)\cos\theta+(Y-Y_0)\sin\theta-r_1}{(X-X_0)\sin\theta\sin\phi+(Y-Y_0)\cos\theta\sin\phi-(Z-Z_0)\cos\phi+r_3+f} \qquad (1.28)$$

$$x = f\,\frac{(X-X_0)\sin\theta\cos\phi+(Y-Y_0)\cos\theta\cos\phi+(Z-Z_0)\sin\phi-r_2}{(X-X_0)\sin\theta\sin\phi+(Y-Y_0)\cos\theta\sin\phi-(Z-Z_0)\cos\phi+r_3+f} \qquad (1.29)$$

۱.۶ کالیبراسیون دوربین

در بخش قبل مدل دوربین را بررسی کردیم، که مختصات جهانی را با مختصات تصویر مرتبط می‌کند. فرض شد که فاصله‌ی کانونی دوربین، زاویه‌ی چرخش‌ها و مقدار جابه‌جایی انتقال معلوم هستند. روش جایگزین برای محاسبه‌ی مدل دوربین استفاده از دوربین به عنوان یک وسیله‌ی اندازه‌گیری برای مشخص کردن مجهولات در مدل دوربین است. این فرآیند کالیبراسیون دوربین نام دارد. در این روش، چند نقطه در فضای سه بعدی با مختصات معلوم و نقاط متناظرشان روی مختصات تصویر برای کالیبره کردن دوربین استفاده می‌شود.

می‌توان تصویر کردن یک مختصات سه بعدی همگن روی صفحه‌ی تصویر (از معادله‌ی ۱.۲۷) را به صورت زیر خلاصه کرد:

$$C_h = AW_h \tag{1.30}$$

$$\begin{bmatrix} C_{h_1} \\ C_{h_2} \\ C_{h_3} \\ C_{h_4} \end{bmatrix} = \begin{bmatrix} a_{11}a_{12}a_{13}a_{14} \\ a_{21}a_{22}a_{23}a_{24} \\ a_{31}a_{32}a_{33}a_{34} \\ a_{41}a_{42}a_{43}a_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.31}$$

که C_hو W_h به ترتیب مختصات همگن دوبین و جهانی هستند، و $A = PCR_{-\phi}^{X}R_{-\theta}^{Z}G$، نشان‌دهنده‌ی ماتریس دوربین است. هدف در کالیبراسیون دوربین مشخص کردن ماتریس A با استفاده از نقاط سه‌بعدی معلوم و نقاط متناظرشان در مختصات تصویر است. ماتریس A یک ماتریس ۴×۴ است که ۱۶ مجهول دارد. با این حال C_{h_a} در مختصات تصویر بی معنی است. چون تصویر فقط دو بعد دارد، مختصات تصویر می‌توانند با C_{h_1}، C_{h_2} و C_{h_4} نمایش داده شوند. از این رو نیازی نیست که ردیف سوم ماتریس را تعیین کنیم. اکنون، فقط ۱۲ مجهول باقی در ماتریس A باقی می‌مانند. بعد در این بخش خواهیم فهمید که می‌توان به دلخواه یکی از ۱۲ مجهول را ثابت انگاشت و ۱۱ تای بقیه را تعیین کرد.

همانطور که می‌دانیم، مختصات (x , y)کارتزین تصویر به صورت زیر هستند:

$$x = \frac{C_{h_1}}{C_{h_4}} \tag{1.32}$$

$$y = \frac{C_{h_2}}{C_{h_4}} \tag{1.33}$$

از معادلات ۱.۳۱ تا ۱.۳۳ به دست می‌آوریم:

$$C_{h_1} = a_{11}X + a_{12}Y + a_{13}Z + a_{14} = C_{h_4}x \tag{1.34}$$

$$C_{h_2} = a_{21}X + a_{22}Y + a_{23}Z + a_{24} = C_{h_4}y \tag{1.35}$$

$$C_{h_4} = a_{41}X + a_{42}Y + a_{43}Z + a_{44} \tag{1.35}$$

با جایگذاری C_{h_1}، C_{h_2} و C_{h_4}در معادلات ۱.۳۲ تا ۱.۳۳ و دو باره چیدن آن‌ها عبارت زیر را به دست می‌آوریم:

$$a_{11}X + a_{12}Y + a_{13}Z + a_{14} - xa_{41}X - xa_{42}Y - xa_{43}Z - xa_{44} = 0 \tag{1.37}$$

$$a_{21}X + a_{22}Y + a_{23}Z + a_{24} - ya_{41}X - ya_{42}Y - ya_{43}Z - ya_{44} = 0 \tag{1.38}$$

در معادلات ۱۲ مجهول وجود دارد (a_{11} , … … , a_{44}) و ۵ معلوم (y،x،Z،X ، Y). یک نقطه در سه بعد با مختصات (X،Y،Z) و نقاط متناظرش روی تصویر (y و x) دو معادله با ۱۲ مجهول می‌دهد، n تا از این نقاط Zn معادله می‌دهند که می‌توان برای حل ۱۲ مجهول استفاده شوند.

$$a_{11}X_1 + a_{12}Y_1 + a_{13}Z_1 + a_{14} - x_1a_{41}X_1 - x_1a_{42}Y_1 - x_1a_{43}Z_1 - x_1a_{44} = 0 \quad (1.39)$$

$$a_{11}X_2 + a_{12}Y_2 + a_{13}Z_2 + a_{14} - x_2a_{41}X_2 - x_2a_{42}Y_2 - x_2a_{43}Z_2 - x_2a_{44} = 0 \quad (1.40)$$

$$\vdots$$

$$a_{11}X_n + a_{12}Y_n + a_{13}Z_n + a_{14} - x_na_{41}X_n - x_na_{42}Y_n - x_na_{43}Z_n - x_na_{44} = 0 \quad (1.41)$$

$$a_{21}X_1 + a_{22}Y_1 + a_{23}Z_1 + a_{24} - y_1a_{41}X_1 - y_1a_{42}Y_1 - y_1a_{43}Z_1 - y_1a_{44} = 0 \quad (1.42)$$

$$a_{21}X_2 + a_{22}Y_2 + a_{23}Z_2 + a_{24} - y_2a_{41}X_2 - y_2a_{42}Y_2 - y_2a_{43}Z_2 - y_2a_{44} = 0 \quad (1.43)$$

$$\vdots$$

$$a_{21}X_n + a_{22}Y_n + a_{23}Z_n + a_{24} - y_na_{41}X_n - y_na_{42}Y_n - y_na_{43}Z_n - y_na_{44} = 0 \quad (1.44)$$

در این معادلات زیر نویس روی (X ،Y،Z) و (x,y) شماره‌ی نقطه را نشان می‌دهد. سیستم بالا را می‌توان به فرم ماتریسی به صورت زیر نوشت:

$$\begin{bmatrix} X_1Y_1Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ X_2Y_2Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ & & & & & & \vdots & & & \\ X_nY_nZ_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n & -x_n \\ 0 & 0 & 0 & 0 & X_1Y_1Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ 0 & 0 & 0 & 0 & X_2Y_2Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ & & & & & & \vdots & & & \\ 0 & 0 & 0 & 0 & X_nY_nZ_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n & -y_n \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{41} \\ a_{42} \\ a_{43} \\ a_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (1.45)$$

یا

$$CP = 0 \quad (1.46)$$

که C یک ماتریس با ابعاد ۲n×۱۲، P یک بردار با ابعاد ۱×۱۲ و O نیز یک بردار به اندازه‌ی ۱×۱۲ است. این سیستم یک سیستم همگن است که چندین جواب دارد. از این رو، می‌توانیم به دلخواه یکی از مجهولات را برداریم و مجهولات باقی مانده را تعیین کنیم. فرض کنید $a_{44} = 1$ باشد در این صورت سیستم بالا را می‌توان به صورت زیر باز نویسی کرد.

$$\begin{bmatrix} X_1Y_1Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ X_2Y_2Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ & & & & & & \vdots & & & \\ X_nY_nZ_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n & -x_n \\ 0 & 0 & 0 & 0 & X_1Y_1Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ 0 & 0 & 0 & 0 & X_2Y_2Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ & & & & & & \vdots & & & \\ 0 & 0 & 0 & 0 & X_nY_nZ_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n & -y_n \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{41} \\ a_{42} \\ a_{43} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad (1.47)$$

$$DQ = R \quad (1.48)$$

در سیستم بالا ماتریس D و بردار R معلوم هستند. می‌توانیم ۱۱ مجهول باقی مانده در بردار Q را با شبه معکوس D به دست بیاوریم، اگر حداقل ۵.۵ نقطه‌ی سه بعدی و نقاط متناظرشان روی تصویر مشخص باشند. با ضرب معادله‌ی بالا در D به دست می‌آوردیم:

$$D^T DQ = D^T R \quad (1.49)$$

$$Q = (D^T D)^{-1} D^T R \quad (1.50)$$

وقتی Q معین شود، ماتریس A معلوم می‌شود که کالیبراسیون دوربین را تکمیل می‌کند.

۱.۷ بازیابی پارامترهای دوربین

در این بخش به یک مسئله‌ی معکوس جالب می‌پردازیم که با بازیابی پارامترهای دوربین از ماتریس دوربین سرو کار دارد. برای مثال، یک عکس را که با یک دوربین نامعلوم از یک محل نامعلوم گرفته شده و عکس بریده شده و یا بزرگ نمایی شده است را در نظر بگیرید. چگونه می‌توانیم حالت قرار گیری و جهت دوربین و اینکه تصویر تا چه حد بریده یا بزرگ نمایی شده را تعیین کنیم؟ این در چندین حوزه کاربرد دارد. برای مثال در ناوبری اتوماتیک، یک موشک کروز می‌تواند به ماتریس تبدیل دوربین را از یک مدل عوارض زمین که روی برد ذخیره شده است به دست بیاورد و سپس پارامترهای دوربین را که موقعیت وسیله و جهت حرکت آن را تعریف می‌کنند را به دست

آورد. همچنین یک دوربین ثابت که یک بازوی روباتی را نظاره می‌کند می‌تواند موقعیت بازو را مشخص کند. علامت گذاری چندین نقطه روی بخشی از بازوی روباتیک امکان استخراج راحت تر این اطلاعات را می‌دهد و Ground truth (اطلاعات اندازه‌پیری شده) را برای کالیبراسیون تصویر را فراهم می‌کند. پارامترهای دوربین را می‌توان از ماتریس تبدیل دوربین استخراج کرد و موقعیت و جهت قرارگیری بازو را می‌توان با نسبت آن به دوربین ثابت به دست آورد.

۱.۷.۱ موقعیت دوربین

یک نقطه‌ی سه بعدی همانند X1 را با مختصات (x1 , y1 , z1 , 1) (شکل ۱.۴ را ببینید) در نظر بگیرید. اگر ماتریس دوربین A، معلوم باشد می‌توانیم مختصات تصویر نقطه را با معادله‌ی ۱.۳۰ که $U1 = A \times X1$ و $U1 = (C_{h_1}, C_{h_2}, C_{h_3}, C_{h_4})$ است توصیف کنیم. اگر $U1$ را به A^{-1} ضرب کنیم، می‌توانیم $X1$ اصلی را باز گردانیم. فرض کنید C_{h_3} را برابر صفر کنیم، در این صورت $U1' = (C_{h_1}, C_{h_2}, 0, C_{h_4})$ خواهد بود. اکنون $U1'$ را با عبارت $A^{-1}U1' = X11$ را بازتاب می‌دهیم. یک نقطه‌ی سه‌بعدی جدید $X11$ را به دست می‌آوریم که باید روی خطی قرار بگیرد که $X1$ و L را (شکل ۱.۴ را ببینید) به هم متصل می‌کند. مشابهاً می‌توان نقطه‌ی سه‌بعدی دیگری مانند $X2$ را که $X2 = (X2, Y2, Z2, 1)$ است را انتخاب و X21 را تولید کرد که روی خط متصل کننده‌ی $X2$ و مرکز بازتابش قرار می‌گیرد. اکنون به آسانی می‌توانیم L را با قطع دادن این دو خط پیدا کنیم.

۲.۷.۱ جهت دوربین

جهت دوربین همان جهت صفحه‌ی تصویر تعریف می‌شود. از شکل ۱.۵ واضح است که وقتی یک شی به سمت دوربین حرکت می‌کند، تصویرش از مرکز تصویر در طول محور X دور می‌شود. وقتی شی روی عدسی است تصویر در بی نهایت تشکیل می‌شود. تنها راهی که تصویر یک نقطه با موقعیت محدود (نه در بی نهایت) در بی نهایت تشکیل شود این است که چهارمین جزء مختصات همگن تصویر صفر شود. از معادله‌ی ۱.۳۰ داریم:

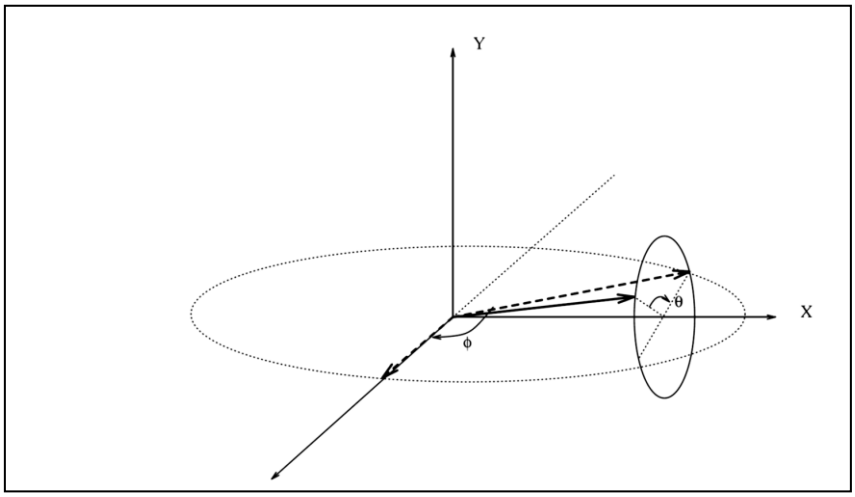
$$\begin{bmatrix} C_{h_1} \\ C_{h_2} \\ C_{h_3} \\ 0 \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.51}$$

که محدودیت زیر بر جهت دوربین قرار می‌دهد:

$$a_{41}X + a_{42}Y + a_{43}Z + a_{44} = 0 \tag{1.52}$$

این معادله‌ی یک صفحه است که از عدسی L می‌گذرد و موازی صفحه‌ی تصویر است. از این معادله مشخص است که (a_{41}, a_{42}, a_{43})، بهنجار به صفحه‌ی تصویر موازی با دوربین است.

جهت بر حسب دوران در جهت ساعت گرد حول محور X به اندازه‌ی θ و سپس دوران حول Y به اندازه‌ی ϕ (همانطور که در شکل ۱.۳ نشان داده شده است) به صورت زیر به دست می‌آید:



شکل ۱.۳: جهت دوربین

$$\theta = \arctan\frac{a_{42}}{-a_{43}} \tag{1.53}$$

$$\phi = \arcsin\frac{-a_{41}}{\sqrt{a_{41}^2 + a_{42}^2 + a_{43}^2}} \tag{1.54}$$

همچنین امکان استخراج فاصله‌ی کانونی، مقیاس دهی و سایر پارامترهای دوربین از ماتریس دوربین وجود دارد. با این وجود، استخراج این پارامترها کمی‌پیچیده است و ما آن را در این کتاب پوشش نمی‌دهیم. خواننده علاقهمند برای جزئیات بیشتر می‌تواند به مرجع (۳۳) در بخش مراجع مراجعه کند.

۱.۸ فرمول رد ریگه

فرض کنید می‌خواهیم بردار V را حول بردارn به اندازه θ دوران دهیم. می‌توانیم بردار V را به دو بخش تجزیه کنیم، یکی هم خط با n و دیگری عمود بر n، به صورت زیر (تصویر ۱.۶ را ببینید):

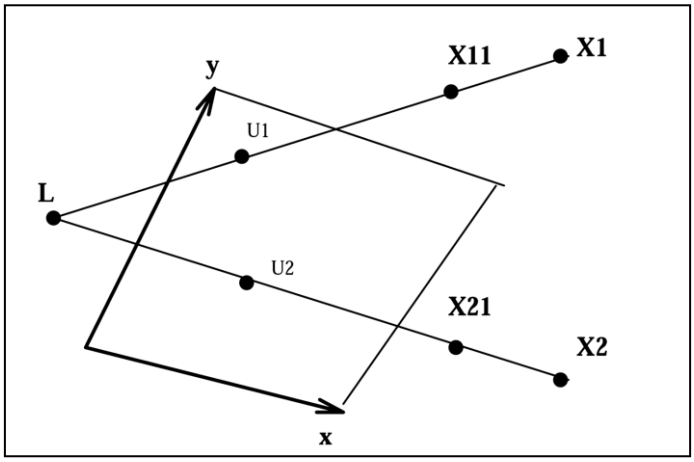
$$v = (v.n)n + (v - (v.n)n) \tag{1.55}$$

مؤلفه‌ی اول نا متغیر با دوران است (چون هم جهت با n است) در حالی که متغیر دوم دچـار دوران صـفحه‌ای بـا زاویه‌ی θ می‌شود که می‌توان نوشت:

$$\cos\theta\ (\ v-(v.n)n)+\sin\theta\big(n\times(v-(v.n)n)\big). \tag{1.56}$$

که به دست می‌دهد:

$$v'=\cos\theta\ v+\sin\theta\ n\ \times v+(1-\cos\theta)\ (v.n)n \tag{1.57}$$



شکل ۱.۴: تعیین محل قرار گیری دوربین. تصویر نقاط سه بعدیX1 و X11 رویU1 و تصویر نقاط X2 و X21 رویV2 تشکیل شده است. تقاطع خط گذرنده ازX1وX11 با خط گذرنده ازX2 و X21، L است که محل قرار گیری دوربین می‌باشد.

یا

$$v'=\ v+\sin\theta\ n\ \times v+(1-\cos\theta)\ n\times(\ n\times v). \tag{1.58}$$

که $n\times(\ n\times v)=(v.n)n-v$است. اکنون، از معادله‌ی بالا می‌توان ماتریس دوران R_θ^nرا به صورت زیـر نوشت:

$$R_\theta^n=I+\sin\theta\ X(n)+(\ 1-\cos\theta)\ X(n)^2 \tag{1.59}$$

که X(n) عملکرد ضرب برداری به n می‌باشد، یعنی ماتریس نا متقارن تشکیل شده با مؤلفه‌های n:

$$X(n)=\begin{bmatrix}0 & -n_z & n_y\\ n_z & 0 & -n_x\\ -n_y & n_x & 0\end{bmatrix} \tag{1.60}$$

بنابراین برای نمایش دوران، به بردار n و زاویه‌ی θ نیاز داریم. می‌توانیم این را بیشتر ساده کنیم و دوران را بـا فقـط یک بردار نشان دهیم، به این صورت:

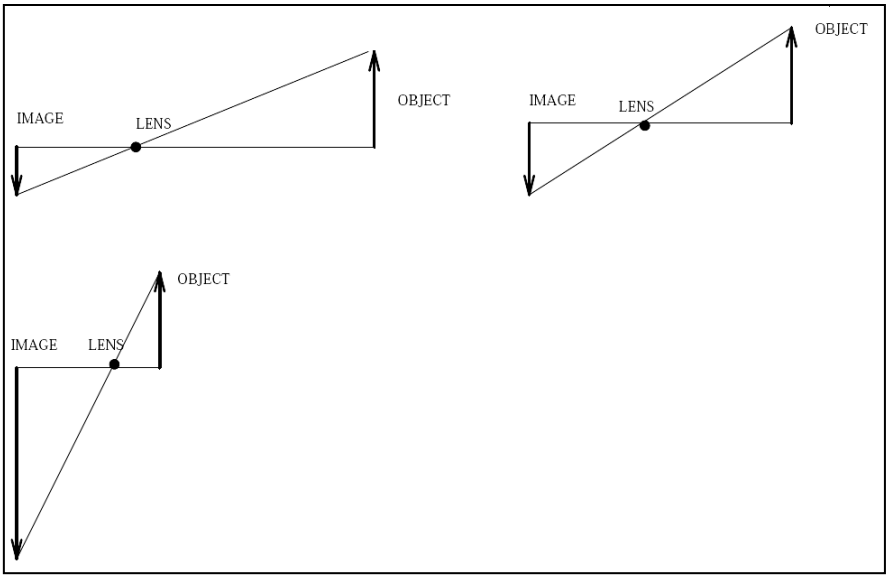
$$r=\ \|r\|\frac{r}{\|r\|}=\ \theta\ n \tag{1.61}$$

بزرگی r، مقدار دوران و جهت r محور دوران هستند. در آخر ماتریس دوران را می‌توان به صورت زیر نوشت:

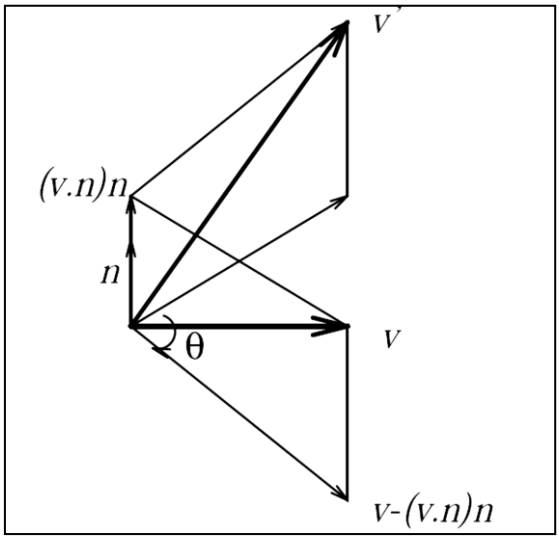
$$R^r = R^r_{\|r\|} = I + \sin\theta \frac{X(r)}{\|r\|} + (1 - \cos\theta) \frac{X(r)^2}{\|r\|^2} \quad (1.62)$$

که X(r) به صورت زیر به دست می‌آید:

$$X(r) = \begin{bmatrix} 0 & -r_z r_y \\ r_z & 0 & -r_x \\ -r_y r_x & 0 \end{bmatrix} \quad (1.63)$$



شکل ۱.۵: تعیین جهت دوربین. وقتی تصویر به عدسی نزدیک می‌شود، تصویر آن از مرکز دوربین در محور Y دور می‌شود. وقتی تصویر روی عدسی قرار گیرد در بی نهایت تشکیل می‌شود.



شکل ۱.۶: دوران r حول n با زاویه‌ی θ

۱.۹ کواترنیون‌ها (چهارگانه)

کواترنیون یک زوج $q = (s, w)$ است که s یک اسکالر است و معمولا بخش حقیقی g نامیده می‌شود و w یک بردار سه بعدی است و بخش موهومیqنامیده می‌شود. دو کواترونیون را می‌توان مانند هر بردار n بعدی دیگری جمع زد. با این وجود ضریب دو کواترنیون به صورت زیر به دست می‌آید:

$$(s, w) * (s', w') = (ss' - w.w', w \times w' + sw' + s'w) \qquad (1.64)$$

که " × "و "." نشان دهنده‌ی ضرب برداری و نقطه‌ای معمولی هستند. مزدوج و فرم کواترنیون به صورت زیر تعریف می‌شود:

$$q = (s, -w) \qquad (1.65)$$

$$|q|^2 = q * q = \|w\|^2 + s^2 \qquad (1.66)$$

برای هر دوران R یک بردار سه بعدی،P، دو کواترنیون ممکن وجود دارد، q و q- که در $Rp = q * p * q$ صدق می‌کنند (P با فرض بخش حقیقی صفر به کواترنیون تبدیل می‌شود و مشابهاً بخش حقیقی کواترنیون حاصل در سمت راست نیز صفر می‌کنیم تا با بردار دوران یافته در سمت چپ مطابقت کند).

کواترنیون q از جهت واحد بردار محور دوران n و زاویه‌ی آنθبا فرمول زیر محاسبه می‌شود:

$$q = \left(\cos(\tfrac{\theta}{2}), \sin(\tfrac{\theta}{2})\right) \qquad (1.67)$$

مشابهاً برای هر زوج کواترنیون (q , -q)یک ماتریس دوران یکتا وجود دارد،

$$R = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2(bc - ad) & 2(bd + ac) \\ 2(bc + ad) & a^2 + c^2 - b^2 - d^2 & 2(cd - ab) \\ 2(bd - ac) & 2(cd + ab) & a^2 + d^2 - b^2 - c^2 \end{bmatrix} \quad (1.68)$$

که q = (a , b , c , d).

۱.۱۰ تخمین حالت قرار گیری

در تخمین حالت،مسئله تعیین جهت و موقعیت یک شی (دوربین) است که منتج به تصویر مجموعه نقاطی از دو بعد می‌شود. یک کاربرد مهم تخمین حالت در تشخیص اشیاء مبتنی بر مدل است. در تشخیص اشیاء مبتنی بر مدل دو بعدی به سه بعدی، یک مدل سه بعدی از جسم و تصویر دو بعدی اش از یک نقطه‌ی مناسب دوربین گرفته می‌شود، هدف تعیین سه چرخش و سه انتقال نسبت به همان سیستم پایه مختصات است.

فرض کنید (X', Y', Z') مختصات سه بعدی یک نقطه هستند و (x', y') نقاط تصویر شده در مختصات تصویر هستند. فرض کنید شی (دوربین) اول به اندازه‌ی $(\phi_x , \phi_y , \phi_z)$ دوران یافته که با آن (X', Y', Z') را به (X , Y , Z) نگاشت می‌شود. سپس، شی به اندازه‌ی (T_x , T_y , T_z) انتقال یافته و در آخر تصویر با تبدیل پرسپکتیو با استفاده از سیستم مختصات خم مرکز با عدسی و فرض f مثبت ساخته می‌شود. مختصات تصویر پس از تبدیل پرسپکتیو به صورت زیر به دست می‌آید:

$$(x' , y') = \left(\frac{f(X+T_x)}{Z+T_Z} , \frac{f(Y+T_Y)}{Z+T_z}\right) \quad (1.69)$$

برای ساده کردن این معادلات به بای استفاده از انتقال (T_X , T_Y)از (D_X , D_Y)استفاده خواهیم کرد که جابه جایی تصویر x(کوچک)و y(کوچک) هستند و انتقال سه بعدی در جهت Z است. حالا، مختصات تصویر به این صورت به دست می‌آیند:

$$(x', y') = \left(\frac{f\,X}{Z+D_z} + D_X , \frac{f\,Y}{Z+D_Z} + D_Y\right) = (f\,X_c + D_x , fY_c + D_Y) \quad (1.70)$$

که $c = \frac{1}{Z+D_Z}$.

تخمین حالت را می‌توان به صورت یک مسئله‌ی بهینه سازی فرموله کرد که خطا را بین مختصات تصویر و مدل حداقل می‌کند (۱۲). خطا بین مؤلفه‌های مدل و تصویر را می‌توان به صورت زیر تغییرات کوچک در شش مجهول $(D_X , D_Y , D_Z , \phi_X , \phi_Y , \phi_Z)$و مشتق آن‌ها نسبت به مختصات تصویر بیان نمود. از آن جایی که خطاها معلوم هستند، تغییرات کوچک در پارامترها می‌تواند به صوذت تکرار شونده از این معادلات محاسبه شوند و تخمین بعدی با اضافه کردن این تغییرات به تخمین قبلی به دست آید. معادله‌ی خطا، $E_{x'}$، x' در مختصه‌ی x

تصویر را می‌توان به صورت زیر بر حسب مجموع ضرب مشتقات جزئی آن ضرب در مقادیر تصحیح خطا (با استفاده از تقریب مرتبه اول سری تیلور) به صورت زیر نوشت:

$$E_{x'} = \frac{\partial x'}{\partial D_x}\Delta D_x + \frac{\partial x'}{\partial D_Y}\Delta D_Y + \frac{\partial x'}{\partial D_Z}\Delta D_Z + \frac{\partial x'}{\partial \phi_x}\Delta \phi_x + \frac{\partial x'}{\partial \phi_Y}\Delta \phi_Y + \frac{\partial x'}{\partial \phi_Z}\Delta \phi_Z \qquad (1.71)$$

مشتقات جزئی در معادله‌ی بالا را می‌توان به سادگی از ۱.۷۰ محاسبه کرد. برای مثال $\frac{\partial x'}{\partial D_x} = 1$ و $\frac{\partial x'}{\partial \phi_Y} = f\ \frac{\partial X}{\partial \phi_Y}\frac{1}{Z+D_Z} - \frac{f\,X}{(Z+D_Z)^2}\frac{\partial Z}{\partial \phi_x}$. می‌توانیم $\frac{\partial X}{\partial \phi_Y}$و $\frac{\partial Z}{\partial \phi_Y}$ را از ماتریس دوران، R_ϕ^Y محاسبه کنیم. همانطور که می‌دانید اگر (X', Y', Z') حول محور Y به اندازه ϕ_Yدوران یابد، مختصات جدید (X،Y،Z) به صورت زیر به دست می‌آیند:

$$X = X'\cos\phi_Y + Z\sin\phi_Y \qquad (1.72)$$

$$Y = Y' \qquad (1.73)$$

$$Z = -X'\sin\phi_Y + Z'\cos\phi_Y \qquad (1.74)$$

و بعد

$$\frac{\partial X}{\partial \phi_Y} = -X'\sin\phi_Y + Z'\sin\phi_Y \qquad (1.75)$$

$$\frac{\partial Y}{\partial \phi_Y} = 0 \qquad (1.76)$$

$$\frac{\partial Z}{\partial \phi_Y} = -X'\cos\phi_Y - Z'\sin\phi_Y = -X \qquad (1.77)$$

می‌توانیم سایر مشتقات جزئی را هم به طور مشابه بیابیم. می‌توان معادله‌ی دوم هم برای مختصه‌ی y' تصویر هم مشابه معادله‌ی ۱.۷۱ نوشت و مشتق‌های جزئی‌اش را حساب کرد. مشتقات جزئی x'و y' بر حسب پارامترهای دوربین در شکل ۱.۷ داده شده اند. برای m نقطه‌ی تصویر می‌توانیم $2 \times m$معادله با ش مجهول بنویسیم. این سیستم خطی از معادلات را می‌توان به صورت زیر نوشت:

$$A\Delta = E \qquad (1.78)$$

که A ماتریس مشتق، Δ بردار شش پارامتر مجهول $(D_X, D_Y, D_Z, \phi_X, \phi_Y, \phi_Z)$برای تعیین جهت و محل دوربین،E بردار عبارت خطا است. اکنون، مسئله محاسبه یΔاست که می‌توان با پردازش حداقل مربعات به صورت زیر حل کرد:

$$\Delta = (A^T A)^{-1} A^T E \qquad (1.79)$$

شکل ۱.۹ این متد را برای تخمین حالت نمایش می‌دهد.در تخمین حالت می‌توانیم از خطوط هم به جای نقاط استفاده کنیم. یک رویکرد ممکن اندازه‌ی خطا با اندازه گیری فاصله‌ی عمودی هر نقطه از خط متناظرش در تصویر و سپس گرفتن مشتق بر حسب فاصله به جای x'و y' است. معادله‌ی خط را می‌توان به صورت زیر نوشت:

$$\frac{-m}{\sqrt{m^2+1}}x+\frac{1}{\sqrt{m^2+1}}y=d$$

که m شیب خط و d فاصله‌ی عمودی از مبدأ است. معادله‌ی خط از نقطه‌ی (x',y') را می‌توان از معادله‌ی ۱.۸۰ با جایگذاری (x',y')به جای (x,y) و d' به جای d به دست آورد. سپس فاصله‌ی عمودی نقطه‌ی (x',y') از خط به صورت $d-d'$یه دست می‌آید. محاسبه‌ی مشتقات d' برای استفاده در روش‌های تکرار شونده آسان است، چون مشتقات d' ترکیب خطی x' و y'هستند که از قبل آن‌ها را داشتیم.

	x'	y'
D_X	1	0
D_Y	0	1
D_Z	$-fc^2X$	$-fc^2Y$
ϕ_X	$-fc^2XY$	$-fc(Z+cY^2)$
ϕ_Y	$fc(Z+cX^2)$	fc^2XY
ϕ_Z	$-fcY$	fcX

شکل ۱.۷: مشتقات جزئی مختصلت تصویر بر حسب هر کدام از پارامترهای دوربین

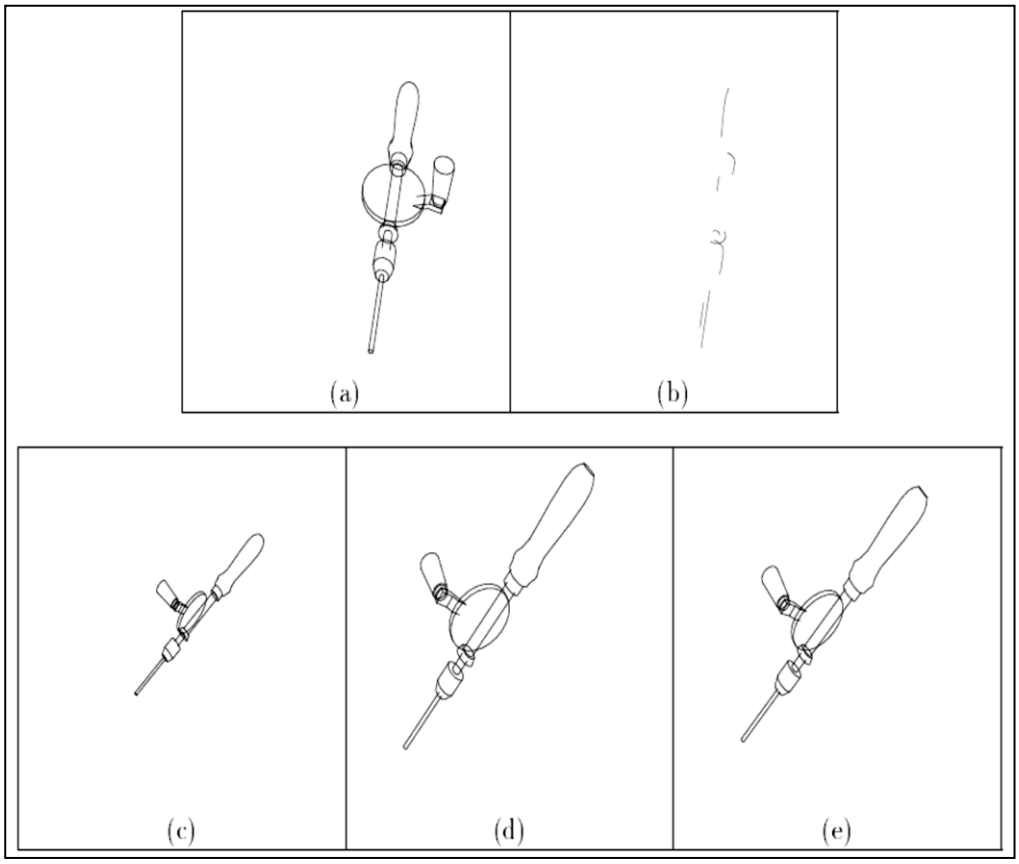
۱: با حدس اولیه‌ای از شش پارامتر شروع کن $(D_X^0, D_Y^0, D_Z^0, \phi_X^0, \phi_Y^0, \phi_Z^0)$. (مثلاً همه‌ی آن‌ها را برابر صفر قرار بده).

۲: تبدیل را به مدل اعمال کن و مدل را به صفحه‌ی تصویر با محاسبه‌ی (x',y') تصویر کن.

۳: همه‌ی خطاهای $E_{X'}$و $E_{Y'}$را حساب کن. اگر خطاها قابل قبول بودند، خارج شو.

۴: تغییرات در شش پارامتر $(\Delta D_X, \Delta D_Y, \Delta D_Z, \Delta\phi_X, \Delta\phi_Y, \Delta\phi_Z^0)$ تبدیل را با استفاده پردازش حداقل مربعات پیدا کن. به گام (۲) برو.

شکل ۱.۸ الگوریتم تخمین حالت.



شکل ۱.۹ (الف) مدل سه بعدی دریل (ب) حلرحواره‌ی لبه‌ها، وقتی در حالت نامعلوم است. (ج) وضعیت دریل پس از یک تکرار (د) وضعیت دریل پس از دو تکرار (ر) وضعیت دریل پس از سه تکرار که حالت صحیح دریل را نشان می‌دهد.

تمرینات

۱- ماتریس دوران یک شی با دوران ۳۰ درجه حول محور Z و بعد یک دوران ۶۰ درجه حول محور X سپس یک دوران حول محور ۹۰ درجه حول محور Y را پیدا کنید. همه‌ی دوران‌ها در جهت ساعتگرد هستند.

۲- ماتریس دوران ماتریس برای یک دوران یک شی به اندازه‌ی ϕ حول محور X و ψ حول محور X و θ حول محور Z را پیدا کنید. همه‌ی دوران‌ها خلاف و پاساعت گرد هستند.

۳- بردار (7,3,2) را در نظر بگیرید که حول محور Z به اندازه‌ی ۹۰ درجه چرخانده شده و سپس حول محور Y به اندازه‌ی ۹۰ درجه چرخانده و در آخر به اندازه‌ی (7 , 3- , 4) انتقال داده شده است. مشخصات جدید بردار را بیابید. همه‌ی دوران‌ها ساعت‌گرد هستند.

۴- نشان دهید عبارات زیر صحیح هستند:

$$R_{90}^{Y}R_{90}^{X} = R_{270}^{Z}R_{90}^{Y}\,, R_{90}^{Y}R_{180}^{X} = R_{180}^{Z}R_{90}^{Y}$$
$$R_{180}^{y}R_{90}^{X} = R_{180}^{Z}R_{270}^{X}\,, R_{180}^{Y}R_{270}^{X} = R_{180}^{Z}R_{90}^{X}$$

۵- معادلات ۱.۲۸ و ۱.۲۹ را اثبات کنید.

فصل ۲

تشخیص لبه

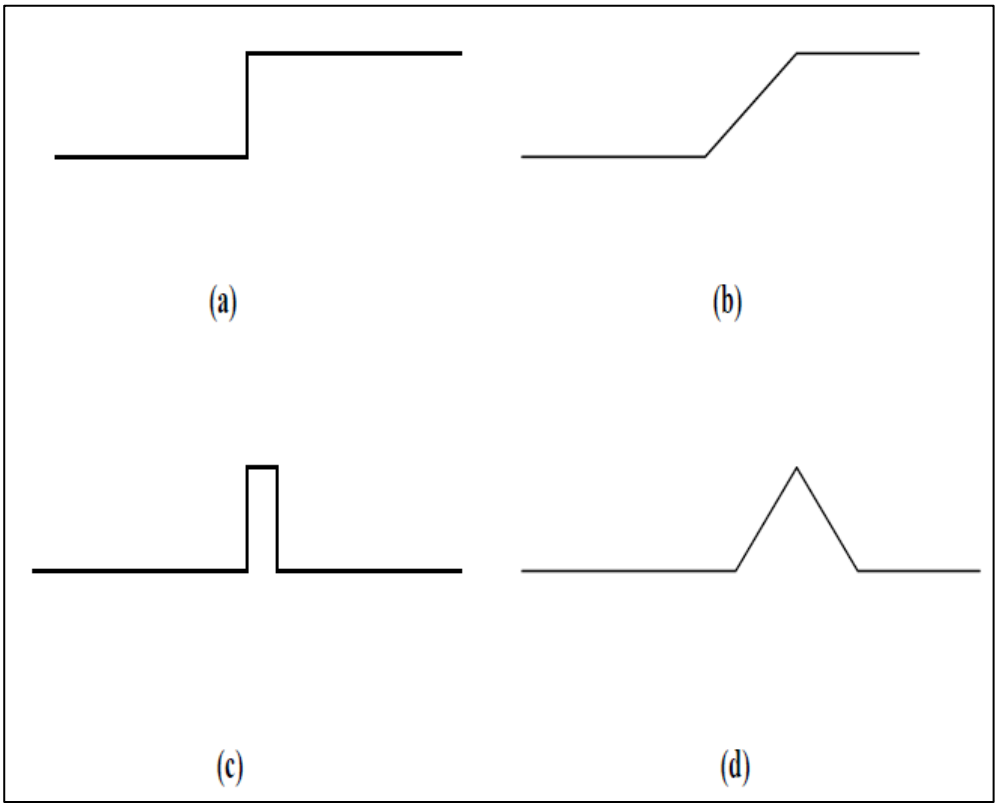
۲.۱ مقدمه

تشخیص لبه که توجه بسیاری از محققان را به خود جلب کرده است یکی از مهمترین حوزه‌ها در بینایی کامپیوتر در سطح پایین می‌باشد. بهر حال، به نظر می‌رسد مسئله تشخیص لبه‌ها در محیط‌های واقعی هنوز به عنوان یک مسئله غیر قابل حل شناخته می‌شود. تشخیص لبه مهم است چون موفقیت پردازش سطوح بالاتر بشدت وابسته به لبه‌های خوب می‌باشد. تصاویر سطح خاکستری شامل مقادیر زیادی داده اند، که بیشتر آنها نا مربوط و پس زمینه تصویر و سایر چیزها می‌باشند. بنابراین در مرحله اول، تلاش برای کاهش میزان داده‌ها صورت می‌پذیرد. اشیا از پس زمینه جدا می‌شوند و موجودیت‌هایی مثل لبه که مهم می‌باشند شناسایی می‌شوند.

برای نمونه در تکنیک استریو، تصویر از دو موقعیت متفاوت که برای گرفتن اطلاعات عمق به کار می‌روند گرفته می‌شود. همه تکنیک‌های استریو با "مطابقت دهی" سروکار دارند. در مطابقت نشانه‌ها از تصویر چپ و راست به منظور گرفتن عدم تطابق دو تصویر با یکدیگر مطابقت داده می‌شوند. اطلاعات لبه نقش مهمی‌در انتخاب نشانه‌ها دارد. در حرکت، تعیین حرکت اشیا با شناسایی لبه‌ها و گوشه‌های تغییر زمانی انجام می‌شود. رویکردهایی برای تعیین ساختار از حرکت وجود دارند که در آنها ساختار سه بعدی اشیا از حرکت محاسبه می‌شود و نیاز به تطابق نشانه‌ها دارند. متدهای تشخیص اشیا فقط بر پایه شکل دو بعدی که از اطلاعات لبه استفاده می‌کند می‌باشد. مهترین الگوریتم برای شناسایی به طور جزئی اشیایی که فرض می‌شود در لبه‌های خوب قرار دارند را مسدود می‌کند.

۲.۲ انواع لبه‌ها

از آنجایی که تابع سطح خاکستری در لبه‌های شی بطور قابل توجهی تغییر می‌کند، می‌توان لبه را (به صورت ایده آل) توسط تابع پله‌ای مدل کرد. با این حال در تصاویر واقعی، سطح خاکستری در لبه‌های یک شی بطور ناگهانی تغییر نمی‌کنند و بطور پیوسته و آرام آرام تغییر می‌کنند. ابن لبه‌ها می‌توانند توسط تابع شیب مدل شوند. گاهی اوقات دو پله یا دو شیب خیلی بخ هم نزدیک می‌شوند در این حال اینها به ترتیب لبه‌های سقف و میخی نامیده می‌شوند. انواع لبه‌های ممکن در شکل ۲.۱ نشان داده است.



شکل ۲.۱: انواع لبه‌ها (a) پله ای، (b) شیب، (c) میخی و (d) سقف

۲.۳ سه مرحله در تشخیص لبه

اغلب تشخیص لبه شامل سه مرحله می‌باشد: فیلترینگ، تفریق‌گیری(دیفرانسیل) و تشخیص. در مرحله فیلترینگ تصویر از طریق فیلتر برای از بین بردن نویز بکار می‌رود. نویز می‌تواند بدلیل تاثیرات ناخواسته در نمونه برداری، کوانتیزسازیون، محوشدگی و عدم تمرکز دوربین و بی نظمی‌هایی در ساختار سطح تصویر بوجود آید. سرانجام، در مرحله دیفرانسیل نقاطی که تغییرات شدت در آن مهم می‌باشد مکان یابی می‌شود.

در رویکردهای اولیه مرحله فیلترینگ حذف شده بود، مرحله دیفرانسیل با استفاده از متد اختلاف–محدود انجام می‌شد و تشخیص با موقعیت یابی نقاط پیک در گرادیان تابع شدت با استفاده ازیک آستانه انجام می‌گرفت. در این متدها، فیلترینگ خیلی مهم نمی‌باشد چونکه فقط تصاویر ترکیبی و صحنه‌های صنعتی در محیط‌های کنترل شده در نظر گرفته می‌شدند. بعدا، سوبل تابع میانگین قبل از مر حله دیفرانسیل معرفی کرد. از آنجایی که میانگین نوعی از فیلترینگ می‌باشد، نتایج بهتری با عملگر سوبل بدست می‌آید. تعیین آستانه وابسته به نوع کاربرداز یک تصویر به تصویر دیگر متغیر می‌باشد. انتخاب اتوماتیک آستانه هنوز به عنوان یک مسئله سخت محسوب می‌شود.

هالاریک مرحله دیفرانسیل را ابتدا با جایگذاری چند جمله‌ای پیوسته در سطح خاکستری همسایه‌های یک پیکسل انجام می‌دهد و سپس مشتقات جزئی چندجمله‌ای را پیدا می‌کند. در این متد، مرحله فیلترینگ مشخص نمی‌باشد ولی عمل جایگذاری به طور اساسی سطح خاکستری تصویر را صاف میکند و بنابراین یک مرحله فیلتر را انجام می‌دهد. به طور بصری، خطای جایگذاری درجه صافی را کنترل می‌کند. کنی از گاوسین در مرحله فیلترینگ استفاده می‌کند و مشتق مرتبه اول همراه با جهت گرادیان در طی مرحله دیفرانسیل‌گیری محاسبه می‌کند. در تشخیص لبه مرحله کنی، نقاط پیک تشخیصی در مشتق خروجی برای مستقر کردن نقاط لبه بکار می‌رود. هیلدرث و مار هم از گاوسین به عنوان فیلتر استفاده می‌کنند. آنها همچنین مرحله دیفرانسیل‌گیری را با استفاده از لاپلاسین را که مجموع مشتق جزئ مرتبه دوم محور x,y است انجام می‌دهد.

۲.۴ مرحله فیلترینگ

مهمترین فیلتر، فیلتر میانگین است. در این فیلتر میانگین، سطح خاکستری در هر عنصرتصویر(پیکسل) با میانه سطوح خاکستری همسایگی نزدیک حول یک پیکسل جایگزین می‌شود. این روش نویز تصادفی میانگین گرفته می‌شود. فرض کنید fتصویر مورد نظر ما باشد که می‌خواهیم فیلتر کنیم و h تصویر فیلتر شده باشد و یک همسایگی ۳ × ۳ حول یک پیکسل را در نظر بگیرید. پس

$$h(x,y) = f(x-1,y-1)\frac{1}{9} + f(x-1,y)\frac{1}{9} + f(x-1,y+1)\frac{1}{9} + f(x,y-1)\frac{1}{9} + f(x,y)\frac{1}{9} + f(x,y+1)\frac{1}{9} + f(x+1,y-1)\frac{1}{9} + f(x+1,y)\frac{1}{9} + f(x+1,y+1)\frac{1}{9} \qquad (2.1)$$

$$h(x,y) = f(x-1,y-1)g(-1,-1) + f(x-1,y)g(-1,0) + f(x-1,y+1)g(-1,1) + f(x,y-1)g(0,-1) + f(x,y)g(0,0) + f(x,y+1)g(0,1) + f(x+1,y-1)g(1,-1) + f(x+1,y)g(1,0) + f(x+1,y+1)g(1,1) \qquad (2,2)$$

$$h(x,y) = \sum_{i=-1}^{i=1}\sum_{j=-1}^{j=1} f(x+i,y+j)g(i,j) \qquad (2,3)$$

$$h(x,y) = f(x,y) * g(x,y) \qquad (2,4)$$

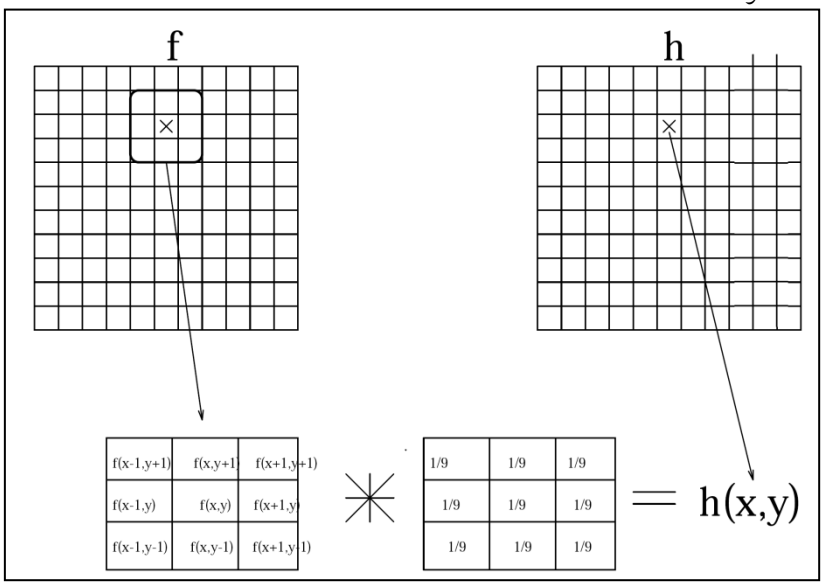
در اینجا که g فیلتر و * عمل پیچش را نشان می‌دهند. فیلتر در این حالت ۳ × ۳ است و هر پیکسل در فیلتر به 1/9 مقدار دهی شده است. فیلتر g با تصویر f پیچش می‌شوند ونتیجه تصویر فیلتر شده h می‌باشد. عمل درهمی‌بطور گسترده در بینایی ماشین و پردازش تصویر استفاده می‌شود. عمل درهمی‌می‌تواند بعنوان گذر حول

فیلتر ماسک (ماسک g) در تصویر از چپ به راست، بالا به پایین که در شکل ۲.۲ نشان داده شده است شبیه سازی شود. ماسک درمرکز هر پیکسل در تصویر f قرار دارد و مجموع ارتباط پیکسل با پیکسل حاصل محاسبه می‌شود.

فیلتر میانگین به صورت متعادل میانگین‌گیری را انجام می‌دهند (وزن همه‌ی پیکسل‌ها برابر است). به هر پیکسل حول همسایگی وزن یکسانی تخصیص داده می‌شود. فیلتر مفید دیگر گاوسین نامیده می‌شود. در فیلتر گاوسین وزن هر پیکسل نسبت به مرکز پیکسل به طور معکوس می‌باشد و بصورت زیر تعریف می‌شود:

$$g(x,y) = e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{2.5}$$

که σ انحراف استاندارد گاوسین می‌باشد و اندازه ماسک را کنترل می‌کند. ماسک $g(x,y)$ می‌تواند برای مقدار داده شده σ با دادن مقادیر مختلف به x,y و جایگذاری در فرمول بالا تولید شود. برای مثال، با x=-1، y=-1، و $\sigma = 1$ به دست می‌آوریم $G(-1,-1) = e^{\frac{-((-1)^2+(-1)^2)}{2(1)^2}} = 0.367$ معمول است ضرایب در فیلتر گاوسین را به نزدیکترین عدد صحیح اندازه بگیریم. چونکه سطوح تصویر خاکستری اعداد صحیح می‌باشند، ساده تر است که ضرب صحیح را به عنوان ضرب اعشاری انجام دهیم. یک ماسک گاوسین برای $\sigma = 1$ که در ۲۵۵ ضرب شده است و به نزدیکترین عدد صحیح گرد شده است در شکل ۲.۳ نشان داده شده است. توجه کنید که این یک ماتریس ۱۳ × ۱۳ می‌باشد. برای مقادیر $x,y > 6$، مقدار خیلی $G(x,y)$ خیلی کوچک می‌شود. بنابراین از مقادیرشان صرف نظر می‌شود. با این حال، برای مقادیر بزرگتر σ، اندازه ماسک بایستی بیشتر باشد. ازآنجایی که تابع گاوسین کمتر از ۱٪ مقدار ماکزیممش برای نقاط در فاصله‌ی $3\sigma < x,y$ می‌شود، بهتر این است که از یک ماسک با اندازه $6\sigma + 1$ استفاده شود.



شکل ۲.۲ عملیات پیچش

۲.۵ مرحله مشتق گیری

مشتق تابع پیوسته f بصورت زیر تعریف می‌شود:

$$f' = \frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} \tag{2.6}$$

از آنجایی که تصاویر گسسته می‌باشند. مقدار مینیمم ΔX می‌تواند ۱ باشد. حالا فرمول بالا می‌تواند بصورت زیر ساده شود.

$$f' = \frac{df}{dx} = f(x) - f(x - 1) \tag{2.7}$$

این به عنوان یک تقریب گسسته یک مشتق نامیده می‌شود. یک سطرعمودی از یک تصویر که در شکل a.۲.۴ نشان داده است را در نظر بگیرید. مشتق آن با استفاده از فرمول بالا در شکل b.۲.۴ نشان داده شده است.

0	0	0	0	1	2	2	2	1	0	0	0	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	3	11	26	50	73	82	73	50	26	11	3	0
1	6	20	50	93	136	154	136	93	50	20	6	1
2	9	30	73	136	198	225	198	136	73	30	9	2
2	11	34	82	154	225	255	225	154	82	34	11	2
2	9	30	73	136	198	225	198	136	73	30	9	2
1	6	20	50	93	136	154	136	93	50	20	6	1
0	3	11	26	50	73	82	73	50	26	11	3	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	0	0	0	1	2	2	2	1	0	0	0	0

شکل ۲.۳: ماسک گاوسین با $\sigma = 2$

دقت کنید که مشتق می‌تواند با اعمال یک ماسک بر f محاسبه شود همانند عملیات فیلترکردن یک تصویر. در این حالت ماسک شامل دو عنصر باشد که در شکل d.۲.۴ نشان داده شده است. f یک لبه پله‌ای است و مشتق آن یک مقدار بیشتر در محل یک لبه می‌باشد. چندین ماسک دیگر وجود دارد که بطور گسترده برای محاسبه مشتق اول که در شکل e.۲.۴ و f.۲.۴ نشان داده شده است. تصاویر بصورت دو بعدی می‌باشند. یک بردار گرادیان مربوط به $f(x,y)$ به صورت (f_x, f_y) است که f_x مشتقی در جهت x می‌باشد.و f_y مشتقی در جهت y می‌باشد. بزرگی گرادیان M در جهت θ بصورت زیر تعریف می‌شود.

$$\theta = \arctan\frac{f_y}{f_x} \tag{2.8}$$

$$M = \sqrt{{f_x}^2 + {f_y}^2} \tag{2.9}$$

مشتق مستقیم در جهت θ بصورت زیر تعریف می‌شود.

$$f'_\theta = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta$$

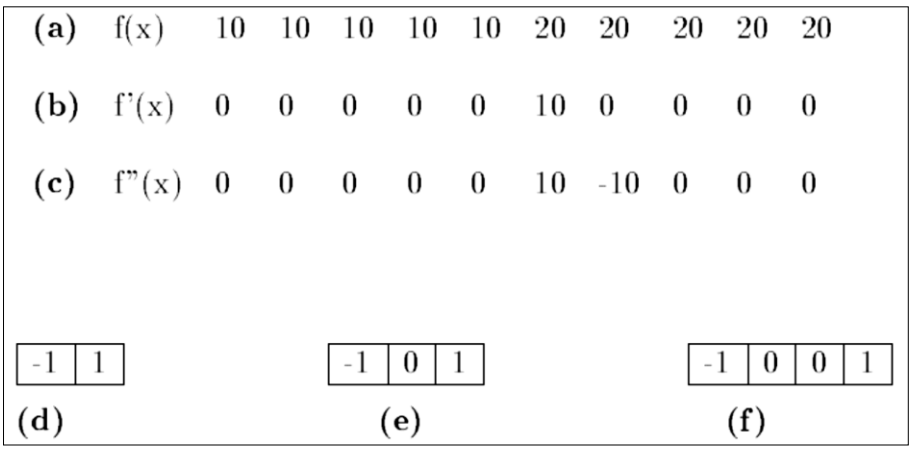
چندین ماسک برای محاسبه گرادیان (f_x, f_y) وجود دارند که برخی از آنها در شکل ۲.۶ نشان داده است.

۲.۶ مرحله تشخیص

۲.۶.۱ بزرگی گرادیان نرمال شده

فرض کنید که $M(x,y) = \sqrt{f_x^2(x,y) + f_y^2(x,y)}$ بزرگی گرادیان در پیکسل (x,y) باشد. $N(x,y)$ را که به عنوان گرادیان نرمال شده بین ۰ تا ۱۰۰ بصورت زیر تعریف می‌کنیم:

$$N(x,y) = \frac{M(x,y)}{max_{i=1,\dots,n\,,j=1,\dots,n}\ M(i,j)} \times 100 \tag{2.10}$$



شکل ۲.۴: دیفرانسیل. (a) یک تابع تک بعدی $f(x)$. (b) مشتق $f'(x)$ با استفاده از تقریب گسسته. (c) $f''(x)$ مشتق دوم. (d) ماسک مشتق. (e) – (f) دو ماسک مشتق اضافی.

سپس می‌توان لبه‌ها با اعمال یک آستانه T روی بزرگی گرادیان N بدست آورد.

$$E(x,y) = \begin{cases} 1 & if\ \ N(x,y) > T \\ 0 & otherwise \end{cases} \qquad (2.11)$$

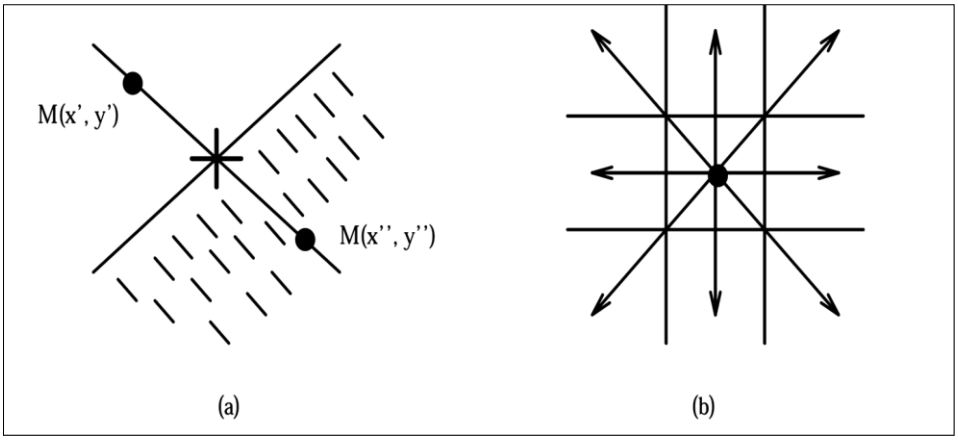
که $E(x,y)$ نگاشت لبه‌ها است.

۲.۶.۲ حذف غیرماکزیما

در متد تشخیصی بالا، ما فقط از بزرگی گرادیان استفاده کردیم و از گرادیان جهت استفاده نکردیم. جهت گرادیان همیشه عمود بر لبه می‌باشد و سطوح خاکستری اغلب در آن جهت تغییر می‌کند. اگر سطح خاکستری پیکسل معینی بطور قابل توجهی در جهت گرادیان تغییر نکند، پس آن پیکسل احتمالا نقطه لبه نمی‌باشد. ما نیاز به حذف آن نقطه (غیر ماکزیما) خواهیم شد. بطور ریاضی حذف غیرماکزیما بصورت زیر تعریف می‌شود:

$$M(x,y) = \begin{cases} M(x,y) & if(M(x,y) > M(x',y')and \\ & if\ M(x,y) > M(x'',y'') \\ 0 & otherwise \end{cases} \qquad (2.12)$$

که $M(x',y')$ و $M(x'',y'')$ بزرگی گرادیان در دو سمت لبه (x,y) در جهت گرادیان هستند که در شکل 2.5.a نشان داده شده است. این مرحله نقاطی که نقاط لبه بالقوه نباشند را حذف خواهد کرد. حالا گرادیان بهبود داده شده را می‌توان به صورت بخش قبلی نرمال کرد. و آستانه را برای تولید یک نگاشت لبه بهتر اعمال کردلازم است که جهت گرادیان به تعدادی ثابتی از جهات کوانتیزه شود. در طی حذف غیرماکزیما، بزرگی گرادیان به هشت عدد که شامل ۰، ۴۵، ۹۰، ۱۳۵،... و ۱۳۵ کوانتیزه می‌شود. شکل 2.5.b پیکسل‌های تطابق در طی مرحله حذف غیر ماکزیما را مقایسه می‌کند.



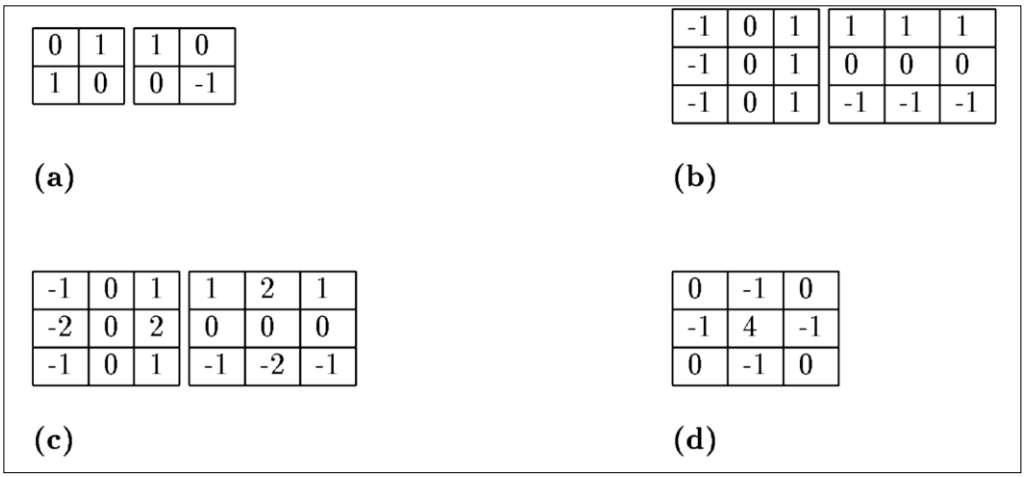
شکل ۲.۵: (a) حذف غیرماکزیما. (b) هشت جهت گرادیان ممکن

۲.۷ کلاس‌بندی روش‌های تشحیص لبه

روشهای تشخیص لبه در سه طبقه کلاس‌بندی می‌شود. اولین طبقه با عملگرهای ۳ × ۳ و ۵ × ۵ سروکار دارد که شامل عملگرهای Prewitt، Robert، Sobel و لاپلاسین می‌باشد. در دومین طبقه، عملگرهایی که بر پایه نوعی از تناسب سطح هستند اضافه می‌شوند. عملگرهایی از Hueckel، Hartly و متد مدل Harlalick در این طبقه استفاده می‌شوند. سرانجام، در سومین طبقه طرح‌های تشخیص لبه شامل تکنیک‌هایی است که از مشتقات گاوسین استفاده می‌کند. اخیرا، عملگرها در این کلاس کاملا محبوب شده اند زیرا این عملگرها برای صحنه‌های واقعی مناسب‌تر هستند. گاوسین از یک نوع میانگین وزن دار سطح خاکستری تصویر بمنظور حذف نویز استفاده می‌کند. این عملگرها از ماسک‌های نسبتا بزرگتری استفاده می‌کنند بنابراین از لحاظ محاسباتی گران‌تر می‌باشند. با این حال، تلاش‌هایی برای کاهش پیچیدگی محاسباتی با استفاده از بعضی از خصوصیات عملگرها صورت گرفته است و پیاده سازی بر اساس مدارات VLSI پیشنهاد شده است.

۲.۸ عملگرهای گرادیان

تعدادی از عملگرهای ۳ × ۳ و ۵ × ۵ شناخته شده هستند. در ابتدا چنین عملگرهایی توسط Robert پیشنهاد شده است. علاوه برعملگر Robert، عملگرهای Sobel و Prewit به طور گسترده استفاده می‌شدند. این عملگرها اساسا گرادیان را در موقعیت پیکسل و در جهات اصلی(افقی و عمودی و قطری) محاسبه می‌کنند. معمولی ترین جهات، جهت‌های X و Y می‌باشند ولی جهات قطری و غیر قطری نیز استفاده می‌شود. محاسبه گرادیان با متد اختلاف محدود برای دیفرانسیل انجام می‌شود. شکل ۲.۶ عملگرهای Robert، Sobel و Prewitt در ساده ترین شکل ممکن نشان می‌دهد. عملگرهای Sobel و Prewitt وزن‌های مساوی به پیکسل‌ها اختصاص می‌دهند. ولی درعملگر Sobel پیکسل‌های نزدیکتر وزن‌های بیشتری داده می‌شود.



شکل ۲.۶: عملگرهای کلاسیک مشتق‌گیری. (a). عملگر Robert. (b). عملگر Prewitt. (c). عملگر Sobel و (d) عملگر لاپلاسین

محاسبه بزرگی گرادیان که شامل جذر نیز می‌باشد که هزینه بر می‌باشد. بنابراین، تقریب‌های ساده شده متفاوتی از بزرگی گرادیان از قبیل $\max(|\Delta x|,|\Delta y|)$(a) و $(|\Delta x|+|\Delta y|)$(b) استفاده شده است. عبارت اولی هزینه محاسباتی جذر و مربع اجتناب می‌کند. علاوه بر این، آن مقادیری را در همان محدوده به عنوان اندازه خاکستری اصلی تولید می‌کند که برای نمایش راحتتر می‌باشد. در حالی که تقریب دوم رنجی بیشتری از محدوده مقادیرممکن را دارد، یک بایس برای و بر خلاف لبه‌های قطری معرفی می‌کند.

عملگر لاپلاسین همچنین به عنوان عملگر لبه استفاده می‌شده است. لاپلاسین بصورت زیر تعریف می‌شود:

$$\Delta^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

این یک عملگر مشتق وابسته جهات می‌باشد. تقریب گسسته بصورت زیر می‌باشد:

$$\Delta^2 f(x,y) = -[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) + 4f(x,y)$$

ماسک پیچش برای این عملگر در شکل 2.6.d نشان داده شده است. در تصویر نویزدار، نویز مقادیر لاپلاسین بیشتری نسبت به لبه‌ها تولید می‌کند.

مرحله فیلترینگ واضحی در عملگرهای اولیه وجود ندارد. آنها صحنه‌های ترکیبی و صحنه‌های واقعی که روشنایی در آن کنترل شده است بهتر عمل می‌کنند. لبه‌های بدست آمده ضخیم می‌باشند بنابراین نیاز به الگوریتم نازک‌سازی می‌باشد. به هر حال، از آنجایی که آنها هزینه محاسباتی کمتری دارند هنوز در صنعت محبوب می‌باشند. به طور معمول اندازه‌های ماسک این عملگرها ۵ × ۵ و ۳ × ۳ می‌باشد. ماسک‌های بزرگتر برای بعضی از این عملگرها می‌تواند محاسبه شود اما لبه‌های بدست خیلی ضخیم به دست می‌آیند. Haralick گزارش می‌دهد که " از تجربیات بدست آمده واضح است که عملگرهای گرادیان خوب بایستی اندازه‌های همسایه بیشتر از ۳ × ۳ داشته باشند. متاسفانه، اندازه‌های همسایه بزرگتر لبه‌های ضخیم تری را تولید می‌کنند." این ما را به سمت استفاده از مشتقات گوسی هدایت می‌کند که در آنها می‌توان پنجره‌های بزرگتری انتخاب کرد و در عین حال ضخامت لبه‌ها نیز قابل قبول بماند.

در این فصل چند روش لبه‌یابی پرکاربرد از قبیل Sobel، Canny، لاپلاسین ارائه و بررسی می‌شوند. ابتدا به بررسی لبه‌یابهای ساده پرداخته و سپس روش‌های پیچیده‌تر را معرفی می‌کنیم.

۲.۹ آشکارساز لبه Canny

افراد زیادی آشکارساز لبه کنی (Canny) را به عنوان آشکارساز لبه نهایی در نظر می‌گیرند. با این آشکارساز لبه‌های تمیز و نازک متصل به هم برای لبه‌های نزدیک همدیگر حاصل می‌شود. در صورتی که از بسته پردازش تصویر همانند متلب استفاده می‌کنید احتمالا تابع آشکارساز لبه کنی را مشاهده کرده اید. در ادامه به توضیح این آشکارساز لبه می‌پردازیم. آشکارساز لبه کنی الگوریتم آشکارساز چند مرحله‌ای است که مراحل آن به صورت زیر فهرست می‌شود:

۱- پیش پردازش

۲- محاسبه گرادیان

۳- حذف غیر بیشینه‌ها

۴- آستانه گذاری هیسترزیس

دو پارامتر مهم الگوریتم آشکارساز لبه، آستانه بالا و آستانه پایین در مرحله چهارم هستند. آستانه بالا برای علامت گذاری لبه‌های قطعی به کار می‌رود در حالی که آستانه پایینی برای یافتن پیکسلهای محو به کار می‌رود که در واقع بخشی از یک لبه هستند

در مجموع آشکارسازهای لبه به نویز حساس هستند. مقداری صاف کردن با کرنل گوسی باعث کاهش حساسیت آشکارساز لبه به نویز می‌شود. در بسته‌های نرم افزاری صاف کردن با هسته گوسی به صورت اتوماتیک انجام نمی‌گیرد. بنابراین عمل صاف کردن باید توسط شما انجام گیرد. در مرحله دوم امتداد و بزرگی گردایان در هر پیکسل از تصویر محاسبه می‌شود. بزرگی گردایان در یک نقطه تعیین کننده حضور احتمالی یک لبه در آن پیکسل است. بزرگی گرادیان بزرگ به معنی تغییرات سریع رنگ است که یک لبه را نتیجه می‌دهد. در حالی که بزرگی گرادیان پایین تغییرات سریعی را نتیجه نمی‌دهد و بنابراین آن نقطه لبه در نظر گرفته نخواهد شد. از سوی دیگر جهت گرادیان نشان می‌دهد که لبه به چه صورتی امتداد دارد. به منظور محاسبه گردایان از کرنل استاندارد سوبل استفاده می‌شود. در ادامه بزرگی گرادیان از فرمول زیر محاسبه می‌شود:

$$m = \sqrt{G_x^2 + G_y^2} \qquad (2.13)$$

در ادامه جهت گرادیان از فرمول زیر محاسبه می‌شود:

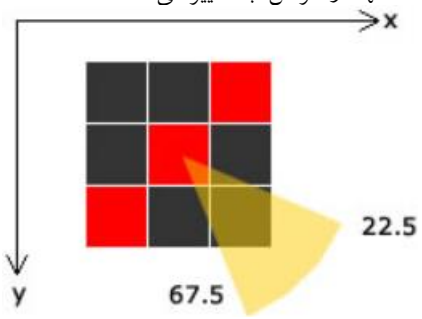
$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \qquad 2.14$$

در این دو فرمول، G_y و G_x تقریبی از مشتق در نقطه X و Y هستند. به محض آنکه بزرگی‌های گرادیان و جهت‌های گرادیان برای تمام پیکسلهای تصویر محاسبه شد مرحله اصلی آشکارسازی لبه شروع می‌شود.

در مرحله سوم مقادیر غیر بیشینه در مجاورت مقادیر بیشینه حذف می‌شوند. به بیان دیگر اگر یک پیکسل بیشینه نباشد مقدار آن پیکسل به صفر نشانده می‌شود. بدین منظور شما بر روی تمام پیکسل گام بر می‌دارید. امتداد پیکسل در یکی از چهار بین به صورت شکل زیر قرار می‌گیرد:

شکل ۲.۷: جهت‌های ممکن برای یک لبه

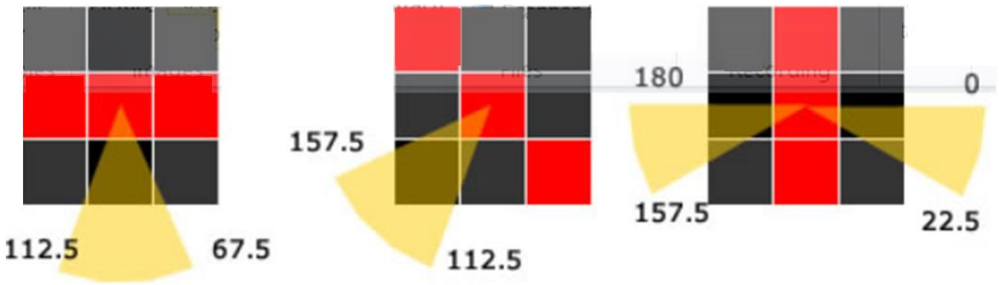
فرض کنید که در یک گام در پیکسل با سطح خاکستری هستید. در این وضعیت چهار لبه محتمل وجود دارد. لبه می‌تواند از شما به جنوب برود (همسایه‌های سبز رنگ) یا از شرق به غرب (همسایه‌های آبی رنگ) و یا یکی از قطرها (همسایه‌های زرد یا قرمز رنگ). پس با به کارگیری جهت گرادیان در پیکسل جاری جهت لبه تخمین زده می‌شود. چهار حالت محتمل به صورت جداگانه برای وارسی حذف غیربیشینه در نظر گرفته می‌شود. در ادامه یکی از حالتهای ممکن را به صورت جزیی توضیح می‌دهیم. سه حالت دیگر با اختلافهای جزیی با همدیگر مشابه هستند. امتداد گرادیانی از ۲۲.۵ تا ۶۷.۵ درجه را در نظر بگیرید. اگر امتداد گرادیان در این محدوده باشد تغییر در جهت نشان داده شده در شکل زیر روی می‌دهد (از گوشی چپ بالایی به گوشه راست پایینی). این بدان معنی است که لبه از گوشه راست بالایی به گوشه چپ پایین (پیکسلهای قرمز رنگ) کشیده شده است. در این مرحله برای بررسی تعلق پیکسل قرمز رنگ مرکزی به یک لبه نیاز به بررسی بیشینه بودن گرادیان در این نقطه داریم. این کار با مقایسه بزرگی گرادیان پیکسل مرکزی با بزرگی گرادیان پیکسل چپ بالایی و بزرگی گرادیان پیکسل راست پایینی دارد. اگر بزرگی گرادیان پیکسل مرکزی بیشینه باشد و در ضمن بزرگی آن بزرگتر از آستانه بالا باشد این پیکسل به عنوان لبه علامت زده می‌شود. توجه داشته باشید که لبه همواره به جهت گردایان عمود است و شدتهای در امتداد یک لبه تغییر نمی‌کنند بلکه آنها در عرض لبه تغییر می‌کنند.



شکل ۲.۸: امتداد گرادیانی در محدوده ۲۲.۵ تا ۶۷.۵ درجه (لبه با پیکسلهای قرمز رنگ مشخص شده است)

مابقی امتدادهای گرادیانی به صورت موازی در نظر گرفته می‌شوند:

امتداد گرادیان در محدوده ۶۷.۵ تا ۱۱۲.۵: گرادیان از بالا به پایین است. این بدان معنی است که لبه از چپ به راست است. پس بزرگی گرادیان پیکسل مرکزی با بزرگیهای گرادیان پیکسل بالای و پیکسل پایینی مقایسه می‌شود(شکل ۲). امتداد گردایان در محدوده ۱۱۲.۵ تا ۱۵۷.۵: گرادیان در امتداد قطر دیگر است. امتداد گرادیان در محدوده ۰ تا ۲۲.۵ یا ۱۵۷.۵ تا ۱۸۰ درجه: گرادیان افقی است. پس لبه عمودی است. پس شما پیکسلهای چپ و راست را وارسی می‌کنید.



شکل ۲.۹: سه حالت باقی مانده از جهت‌های گرادیان (لبه‌ها با پیکسلهای قرمز رنگ علامت گذاری شده است و پیکسل مرکزی با پیسکل‌های سیاه مقایسه می‌شوند)

بعد از مرحله حذف غیر بیشینه به لبه‌های نازک می‌رسیم که در برخی از نقاط شکستگیهایی دارد. این فاصله‌های خالی با آستانه گذاری هیسترزیس پر می‌شوند. در مرحله پیش پیکسلهایی که بزرگی گرادیانی بیشتر از آستانه بالا دارند به عنوان لبه علامت گذاری شدند. حالا با به کارگیری اطلاعات جهت و آستانه پایین لبه‌ها رشد می‌یابند. این ایده به صورت زیر بیان می‌شود:

۱- اگر پیکسل جاری لبه نیست پیکسل بعدی را بررسی کن.

۲- در صورتی که لبه است دو پیکسل در امتداد لبه را بررسی کن (عمود به جهت گرادیان). اگر یکی از آنها یا هر دو آنها شرایط زیر را داشتند شما می‌توانید این پیکسلها را هم به عنوان لبه علامت گذاری کنید:

امتداد مشابه با امتداد پیکسل مرکزی داشته باشند.

بزرگی گرادیان بزرگتر از آستانه پایین باشد.

آنها نسبت به همسایه‌هایشان بیشینه باشند

۳- مراحل بالا را آنقدر تکرار کنید تا تغییری در تصویر ایجاد نشود. وقتی تصویر دیگر تغییر نکرد شما لبه‌های کنی را به دست آورده اید.

شکل ۲.۱۰. خروجی الگوریتم Canny

در ادامه نحوه پیاده سازی آشکارساز لبه کنی در OpenCV آورده شده است.

تعریف تابع به صورت زیر است:

```
Mat MyCanny(Mat src, int upperThreshold, int lowerThreshold,
double size = 3)
{
```

ما یک تصویر را به تابع می‌فرستیم و آستانه بالا و پایین تعیین می‌کنیم و نیز اندازه کرنل آشکارساز لبه سوبل را تعیین می‌کنیم. در این قسمت بخشهای وارسی خطا، کانالهای تصویر و غیره را در نظر نمی‌گیریم. فرض می‌کنیم که تصویر ورودی یک کاناله (سطح خاکستری) است و به فرمت CV_8UC1 است. در واقع هر پیکسل یک بایت فضا در حافظه اشغال می‌کند.

در صورت لزوم می‌توانید ماتی گوسی در تصویر ایجاد کنید. این کار با کلون src به صورت زیر انجام می‌شود:

```
Mat workImg = Mat(src);

workImg = src.clone();

// Step 1: Noise reduction
cv::GaussianBlur(src, workImg, cv::Size(5, 5), 1.4);
```

در ادامه، بزرگی گرادیانها و امتداد گرادیانها به صورت جداگانه محاسبه می‌شوند. در ابتدا ما اپراتور سوبل را به تصویر اعمال می‌کنیم:

```
// Step 2: Calculating gradient magnitudes and directions
Mat magX = Mat(src.rows, src.cols, CV_32F);
Mat magY = Mat(src.rows, src.cols, CV_32F);
cv::Sobel(workImg, magX, CV_32F, 1, 0, size);
cv::Sobel(workImg, magY, CV_32F, 0, 1, size);
```

بعد شیب در هر نقطه محاسبه می‌شود. این کار با تقسیم مشتق Y به مشتق X به صورت زیر صورت می‌گیرد:

```
Mat direction = Mat(workImg.rows, workImg.cols, CV_32F);
cv::divide(magY, magX, direction);
```

سپس بزرگی گرادیان در هر پیکسل محاسبه می‌شود:

```
Mat sum = Mat(workImg.rows, workImg.cols, CV_64F);
Mat prodX = Mat(workImg.rows, workImg.cols, CV_64F);
Mat prodY = Mat(workImg.rows, workImg.cols, CV_64F);
cv::multiply(magX, magX, prodX);
cv::multiply(magY, magY, prodY);
sum = prodX + prodY;
cv::sqrt(sum, sum);
```

در واقع ما بزرگی $\sqrt{G_x^2 + G_y^2}$ را برای هر نقطه محاسبه می‌کنیم.
ما نیاز به مشخص کردن نقاطی داریم که واقعا لبه هستند. در واقع در این نقاط بزرگی گرادیان بزرگتر از آستانه بالا است. مقداردهی‌های اولیه زیر صورت می‌گیرد:

```
Mat returnImg = Mat(src.rows, src.cols, CV_8U);

returnImg.setTo(cv::Scalar(0));          // Initialie image to
return to zero

// Initialize iterators
cv::MatIterator_<float>itMag = sum.begin<float>();
cv::MatIterator_<float>itDirection = direction.begin<float>();

cv::MatIterator_<unsigned char>itRet = returnImg.begin<unsigned
char>();

cv::MatIterator_<float>itend = sum.end<float>();
```

در بازگشت به تصویر می‌رسیم که لبه‌های کنی را در خود دارد. ما تمام گامها برای بزرگی، امتداد، و تصاویر بازگشتی را مقداردهی اولیه نموده ایم. حالا هر پیکسل به صورت زیر بررسی می‌شود:

```
for(;itMag!=itend;++itDirection, ++itRet, ++itMag)
{
const cv::Point pos = itRet.pos();

float currentDirection = atan(*itDirection) * 180 / 3.142;

while(currentDirection<0) currentDirection+=180;

*itDirection = currentDirection;

if(*itMag<upperThreshold) continue;
```

ما پیکسل جاری را در ذخیره می‌کنیم و البته جهت گرادیان بر حسب درجه محاسبه و آن را ذخیره می‌کنیم. سپس اگر بزرگی گرادیان به اندازه کافی بزرگ نبود (بزرگتر از آستانه بالا نبود) از آن عبور می‌کنیم. حالا خط زیر را اضافه می‌کنیم:

```
bool flag = true;
```

این متغیرهای بول نشان دهنده لبه بودن یا نبود پیکسل جاری هستند. در نهایت، اگر تمام تست‌ها انجام شد و اگر مقدار درست را داشت در آن صورت پیکسل را لبه در نظر می‌گیریم. بعد ما هر بین جهتی را در نظر می‌گیریم:

```
if(currentDirection>112.5 && currentDirection <=157.5)
{
       if(pos.y>0 && pos.x<workImg.cols-1 &&
*itMag<=sum.at<float>(pos.y-1, pos.x+1)) flag = false;
       if(pos.y<workImg.rows-1 && pos.x>0 &&
*itMag<=sum.at<float>(pos.y+1, pos.x-1)) flag = false;
}
```

اگر بزرگی گرادیان بین ۱۱۲.۵ و ۱۵۷.۵ درجه بود آنگاه به ۱۳۵ گرد می‌شود. این بدان معنی است که لبه از چپ بالا به راست پایین می‌رود. پس شما نیاز به بررسی پیکسلهای راست بالا و چپ پایین برای شرط بیشینه دارید. توجه کنید که شما نباید به مرزها وارد شوید. اگر بزرگی پایین تر از همسایه‌ها باشد پرچم به غلط نشانده می‌شود. ما کارهای مشابهی با بین‌های دیگر انجام می‌دهیم:

```
else if(currentDirection>67.5 && currentDirection <= 112.5)
{
if(pos.y>0 && *itMag<=sum.at<float>(pos.y-1, pos.x)) flag =
false;
if(pos.y<workImg.rows-1 && *itMag<=sum.at<float>(pos.y+1,
pos.x)) flag = false;    }
else if(currentDirection > 22.5 && currentDirection <= 67.5) {
  if(pos.y>0 && pos.x>0 && *itMag<=sum.at<float>(pos.y-1,
pos.x-1)) flag = false;
  if(pos.y<workImg.rows-1 && pos.x<workImg.cols-1 &&
*itMag<=sum.at<float>(pos.y+1, pos.x+1)) flag = false;   }
else  {
  if(pos.x>0 && *itMag<=sum.at<float>(pos.y, pos.x-1)) flag =
false;
  if(pos.x<workImg.cols-1 && *itMag<=sum.at<float>(pos.y,
pos.x+1)) flag = false;  }
```

بعد از این نردبان if-else ما خواهیم فهمید که پیکسل جاری به صورت قطع لبه است یا نه. در واقع پرچم این مطلب را نشان می‌دهد. بر این مبنا پیکسل به عنوان سفید در تصویر بازگشتی علامت گذاری می‌شود:

```
if(flag)
{
      *itRet = 255;
}
}
```

این مرحله حلقه مرحله سوم را کامل می‌کند.

حالا به بخش انتهای الگوریتم وارد می‌شویم. ما پرچمی‌را نگهداری می‌کنیم که نشان می‌دهد آیا در گام قبلی تغییری صورت گرفته است یا نه. در صورتی که تغییر صورت گرفته باشد ما نیاز به بررسی دوباره تصویر داریم. در غیر اینصورت تصویر نهایی حاصل می‌شود:

```
// Step 4: Hysterysis threshold
bool imageChanged = true;
int i=0;
while(imageChanged)
{
imageChanged = false;
printf("Iteration %d\
", i);
i++;
```

در شروع حلقه ما ImageChanged به غلط (false) می‌نشانیم. اگر پیکسهای لبه جدید پیدا شدند این پرچم به درست (true) تغییر می‌کند. سپس تکرارشونده‌ها دوباره مقداردهی اولیه می‌شوند:

```
itMag = sum.begin<float>();
itDirection = direction.begin<float>();
itRet = returnImg.begin<unsigned char>();
itend = sum.end<float>();
for(;itMag!=itend;++itMag, ++itDirection, ++itRet)
{
```

سپس، موقعیت و امتداد گرادیان پیکسل جاری تعیین می‌شود:

```
cv::Point pos = itRet.pos();
if(pos.x<2 || pos.x>src.cols-2 || pos.y<2 || pos.y>src.rows-2)
continue;
float currentDirection = *itDirection;
```

اگر ما نزدیک یک لبه باشیم از آن پیکسلها عبور می‌کنیم. ما قرار داد می‌کنیم که پیکسل با شدت ۲۵۵ به معنی پیکسل لبه جدید است و همسایگانش وارسی نشده اند. بعد از بررسی همسایه‌ها مقدارش به ۶۴ تغییر می‌کند. هر دو عدد اختیاری هستند. ما می‌توانستیم ۱۷۸ را به جای ۶۴ انتخاب کنیم:

```
// Do we have a pixel we already know an edge?
if(*itRet==255)
{
*itRet=(unsigned char)64;
```

اگر پیکسلی مقدارش ۲۵۵ بود ما مقدار آن را به ۶۴ تغییر می‌دهیم. سپس جهتها را بررسی می‌کنیم و آنها را به صورت زیر پردازش می‌کنیم:

```
if(currentDirection>112.5 && currentDirection <= 157.5)
{
if(pos.y>0 && pos.x>0)
{
if(lowerThreshold<=sum.at<float>(pos.y-1, pos.x-1) &&
returnImg.at<unsigned char>(pos.y-1, pos.x-1)!=64 &&
direction.at<float>(pos.y-1, pos.x-1) > 112.5 &&
direction.at<float>(pos.y-1, pos.x-1) <= 157.5 &&
sum.at<float>(pos.y-1, pos.x-1) > sum.at<float>(pos.y-2, pos.x)
&&
sum.at<float>(pos.y-1, pos.x-1) > sum.at<float>(pos.y, pos.x-
2))
{
returnImg.ptr<unsigned char>(pos.y-1, pos.x-1)[0] = 255;
imageChanged = true;
}
}
if(pos.y<workImg.rows-1 && pos.x<workImg.cols-1)
{
if(lowerThreshold<=sum.at<float>(cv::Point(pos.x+1, pos.y+1))
&&
returnImg.at<unsigned char>(pos.y+1, pos.x+1)!=64 &&
direction.at<float>(pos.y+1, pos.x+1) > 112.5 &&
direction.at<float>(pos.y+1, pos.x+1) <= 157.5 &&
sum.at<float>(pos.y-1, pos.x-1) > sum.at<float>(pos.y+2, pos.x)
&&
sum.at<float>(pos.y-1, pos.x-1) > sum.at<float>(pos.y,
pos.x+2))
{
returnImg.ptr<unsigned char>(pos.y+1, pos.x+1)[0] = 255;
imageChanged = true;
}}}
```

این کد بزرگی است. در ادامه کار بخشهای مختلف را توضیح می‌دهیم.

اگر جهت پیکسل جاری بین ۱۱۲.۵ و ۱۵۷.۵(حدود ۱۳۵) باشد آنگاه لبه از چپ بالا به راست پایین می‌رود. پس شما پیکسلها در این دو موقعیت را بررسی می‌کنید تا مشخص شود پیکسل میانی لبه است یا نه. شما این کار را به صورت زیر انجام می‌دهید:

۱- بزرگی گرادیان در این نقاط بزرگتر از آستانه پایین است.

۲- اگر پیکسل قبلا بررسی نشده باشد مقدار آن به ۶۴ تغییر می‌یابد

۳- جهت در آن پیکسل در همان بین ۱۱۲.۵ تا ۱۵۷.۵ قرار دارد

۴- شرط به کار رفته در حذف غیر بیشینه همچان صادق است:

اگر تمام شرایط بالا صادق بودند شما مقدار همسایه را به ۲۵۵ تغییر می‌دهید. این پیکسل لبه جدید است و سبب می‌شود تا به درست تغییر یابد.

ما پردازش مشابهی برای سه بین دیگر انجام می‌دهیم:

```
else if(currentDirection>67.5 && currentDirection <= 112.5)
   {
   if(pos.x>0)
   {

if(lowerThreshold<=sum.at<float>(cv::Point(pos.x-1, pos.y)) &&
   returnImg.at<unsigned char>(pos.y, pos.x-1)!=64 &&
   direction.at<float>(pos.y, pos.x-1) > 67.5 &&
   direction.at<float>(pos.y, pos.x-1) <= 112.5 &&
   sum.at<float>(pos.y, pos.x-1) > sum.at<float>(pos.y-1,
pos.x-1) && sum.at<float>(pos.y, pos.x-1) >
sum.at<float>(pos.y+1, pos.x-1))
 {
  returnImg.ptr<unsigned char>(pos.y, pos.x-1)[0] = 255;
   imageChanged = true;
   }
   }
   if(pos.x<workImg.cols-1)
 {

if(lowerThreshold<=sum.at<float>(cv::Point(pos.x+1, pos.y)) &&
   returnImg.at<unsigned char>(pos.y, pos.x+1)!=64 &&
   direction.at<float>(pos.y, pos.x+1) > 67.5 &&
```

```
    direction.at<float>(pos.y, pos.x+1) <= 112.5 &&
    sum.at<float>(pos.y, pos.x+1) > sum.at<float>(pos.y-1,
pos.x+1) && sum.at<float>(pos.y, pos.x+1) >
sum.at<float>(pos.y+1, pos.x+1))
  {
   returnImg.ptr<unsigned char>(pos.y, pos.x+1)[0] = 255;
     imageChanged = true;
         }
      }
    }
 else if(currentDirection > 22.5 && currentDirection <= 67.5)
{
 if(pos.y>0 && pos.x<workImg.cols-1)
  {

if(lowerThreshold<=sum.at<float>(cv::Point(pos.x+1, pos.y-1))
&& returnImg.at<unsigned char>(pos.y-1, pos.x+1)!=64 &&
direction.at<float>(pos.y-1, pos.x+1) > 22.5 &&
direction.at<float>(pos.y-1, pos.x+1) <= 67.5 &&
sum.at<float>(pos.y-1, pos.x+1) > sum.at<float>(pos.y-2, pos.x)
&& sum.at<float>(pos.y-1, pos.x+1) > sum.at<float>(pos.y,
pos.x+2))
{
  returnImg.ptr<unsigned char>(pos.y-1, pos.x+1)[0] = 255;
  imageChanged = true;
 }
}
if(pos.y<workImg.rows-1 && pos.x>0)
{

if(lowerThreshold<=sum.at<float>(cv::Point(pos.x-1, pos.y+1))
&& returnImg.at<unsigned char>(pos.y+1, pos.x-1)!=64 &&
direction.at<float>(pos.y+1, pos.x-1) > 22.5 &&
direction.at<float>(pos.y+1, pos.x-1) <= 67.5 &&
sum.at<float>(pos.y+1, pos.x-1) > sum.at<float>(pos.y, pos.x-2)
&& sum.at<float>(pos.y+1, pos.x-1) > sum.at<float>(pos.y+2,
pos.x))
 {
returnImg.ptr<unsigned char>(pos.y+1, pos.x-1)[0] = 255;
 imageChanged = true;
}
}
}
else
{
if(pos.y>0)
{

if(lowerThreshold<=sum.at<float>(cv::Point(pos.x, pos.y-1)) &&
returnImg.at<unsigned char>(pos.y-1, pos.x)!=64 &&
```

```
 (direction.at<float>(pos.y-1, pos.x) < 22.5 ||
 direction.at<float>(pos.y-1, pos.x) >=157.5) &&
 sum.at<float>(pos.y-1, pos.x) > sum.at<float>(pos.y-1, pos.x-
1) && sum.at<float>(pos.y-1, pos.x) > sum.at<float>(pos.y-1,
pos.x+2))
   {
returnImg.ptr<unsigned char>(pos.y-1, pos.x)[0] = 255;
imageChanged = true;
  }
}
if(pos.y<workImg.rows-1)
{

if(lowerThreshold<=sum.at<float>(cv::Point(pos.x, pos.y+1)) &&
returnImg.at<unsigned char>(pos.y+1, pos.x)!=64 &&
direction.at<float>(pos.y+1, pos.x) < 22.5 ||
direction.at<float>(pos.y+1, pos.x) >=157.5) &&
sum.at<float>(pos.y+1, pos.x) > sum.at<float>(pos.y+1, pos.x-1)
&& sum.at<float>(pos.y+1, pos.x) > sum.at<float>(pos.y+1,
pos.x+1))
{
returnImg.ptr<unsigned char>(pos.y+1, pos.x)[0] = 255;
imageChanged = true;
                        }
                    }
                }
            }
        }
    }
```

و این حلقه را برای گام چهار پایان می‌دهد. بعد از پایان حلقه با حلقه کوچک زیر مقدار ۶۴ پیکسلها به ۲۵۵ تغییر می‌کند:

```
cv::MatIterator_<unsigned char>current =
returnImg.begin<unsigned char>();    cv::MatIterator_<unsigned
char>final = returnImg.end<unsigned char>();
    for(;current!=final;++current)
    {
        if(*current==64)
            *current = 255;
    }
    return returnImg;
}
```

در نهایت، تصویر را بر می‌گردانیم.

فصل ۳

بخش بندی ناحیه

۱.۳ معرفی

در این فصل بخش بندی تصویر به چند ناحیه را خواهیم آموخت. این یک رویکرد مکمل بـه روش‌هـای لبه یابی است که در فصل پیش آموختیم. در لبه یابی تصویر را بـا یـافتن حـدود اشـیاء مـی‌یـابیم. حـدود جاهایی هستند که شدت تغییر می‌کند. در روش مبتنـی بـر ناحیـه، نـواحی اشـغال شـده توسـط اشـیاء را می‌یابیم. در این روش‌ها پیکسل‌هایی را که شبیه هم هستند را در یک ناحیه گروه بنـدی مـی‌کنـیم. اولـین بخش این فصل ناحیه بندی سید[1] را بررسی می‌کند. در روش بخش بندی سـید بـر مبنـای توزیـع سـطح خاکستری، شدت‌ها و اتصال پیکسل‌ها یک بخش بندی خام از تصویر به دست می‌آید. با این حال ممکن است بخش بندی سید به تعداد زیادی از نواحی کوچک را به خاطر نویز و سایر فاکتورها منـتج شـود. در بخش دوم این فصل، روش‌های را برای رشد و گسترش نواحی بحث می‌کنیم. این روش‌ها بـرای ترکیـب نواحی همسایه‌ی کوچک و تبدیل آن‌ها به نواحی بزرگتر که اشیاء را بهتر نشان می‌دهند، استفاده می‌شود.

۲.۳ تعریف بخش

یک بخش از تصویر(f(x،y قسمتی $R_1, R_2, \dots, R_n$ است که قیدهای زیر را ارضا می‌کند:

۱. $\bigcup_{i=1}^{n} R_i = f(x,y)$

۲. $R_i \cap R_j = \phi \, , i \neq j$

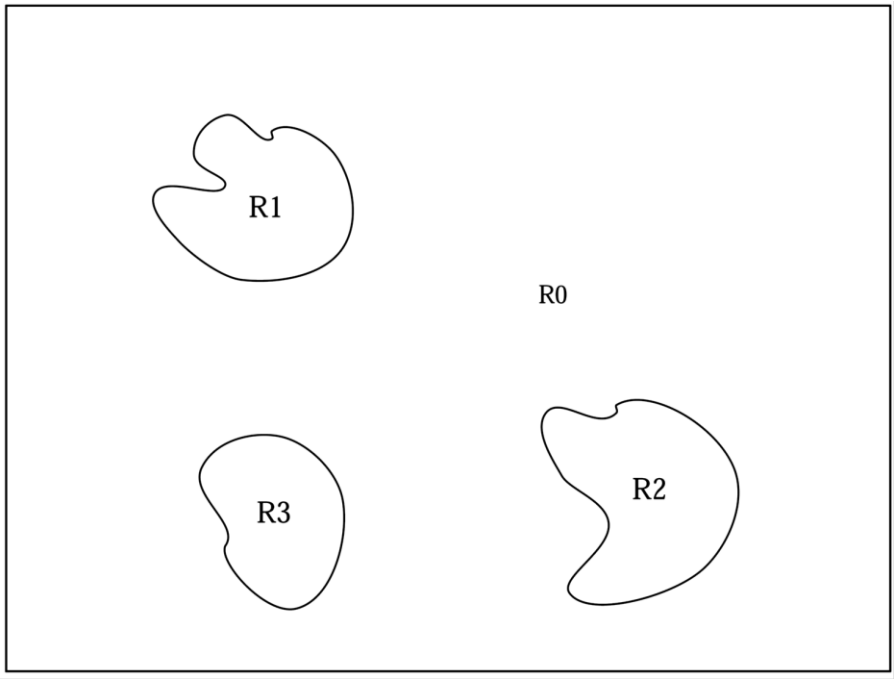
۳. هر قسمت از تصویر یک یا چند گزاره را ارضا کند. مثال‌های از این به صورت زیر هستند:

همه‌ی پیکسل‌ها در هر قسمت R_i باید شدت خاکستری برابر داشته باشد.

---

[1]- Seed

همه‌ی پیکسل‌ها در هر قسمت R_i نباید بیشتر از ΔX در سطح خاکستری تفاوت داشته باشند.
همه‌ی پیکسل‌ها در هر قسمت R_i نباید بیشتر از ΔX در سطح خاکستری از مقدار میانگین متفاوت باشد. انحراف معیار شدت همه‌ی پیکسل‌ها در قسمت R_i کوچک باشد.



شکل ۱.۳ بخش بندی تصویر

۳.۳ بخش بندی ساده

در موارد ساده که $f(x,y)$ در بردارنده‌ی یک شئ است، تصویر سطح خاکستری را می‌توان به تصویر باینری $B(x,y)$ متناظر تبدیل کرد. در این تصویر باینری،پیکسل‌های شئ باید 1 و پیکسل‌های پس زمینه 0 باشند. برای مشخص کردن یک تصویر باینری باید از یک آستانه به صورت زیر استفاده کنیم:

$$B(x,y) = \begin{cases} 1 & if\ f(x,y) < T \\ 0 & otherwise \end{cases} \qquad (۳.۱)$$

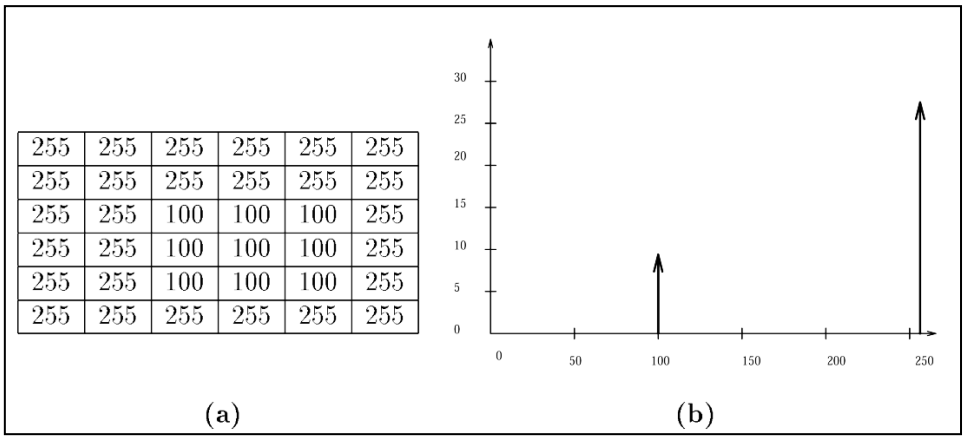
حالات دیگر رابطه‌ی بالا می‌تواند استفاده از دو آستانه T_1 و T_2 یا حتی چندین آستانه به صورت $Z = \{T_1, T_2, \dots, T_k\}$ باشد.

$$B(x,y) = \begin{cases} 1 & if\ T_1 < f(x,y) < T_2 \\ 0 & otherwise \end{cases} \quad (۳.۲)$$

$$B(x,y) = \begin{cases} 1 & if\ f(x,y) \in Z \\ 0 & otherwise \end{cases} \quad (۳.۳)$$

۱.۳.۳ آستانه‌ها و هیستوگرام‌ها

توزیع سطوح خاکستری را می‌توان برای مشخص کردن آستانه‌ی استفاده شده در تصاویر بانیزی استفاده کرد. نمودار هیستوگرام تعداد پیکسل داخل یک تصویر که دارای مقدار مشخصی از سطح خاکستری هستند را به صورت تابعی از سطوح خاکستری تصویر ارائه می‌کند.



شکل ۲.۳:(a) تصویر ساده‌ی سنتز شده با یک شئ، (b) هیستوگرام تصویر نشان داده شده در (a)

برای یک تصویر سنتز شده، این هیستوگرام شامل دو خیزک متفاوت می‌شود. در شکل ۲.۳ a. یک تصویر ۶×۶ ساده با یک شئ نشان داده شده است. سطوح خاکستری پیکسل متناظر شئ ۱۰۰ است، و پیکسل‌های متناظر پس زمینه ۲۵۵ هستند.

هیستوگرام این تصویر در شکل ۲.۳ ب نمایش داده شده است. چون این هیستوگرام دو خیزک دارم،دو مدله[1] نامیده می‌شود. هیستوگرام یک تصویر واقعی ممکن است این گونه خیزک‌های واضحی نداشته باشد. در عوض ممکن است از قله‌ها و دره‌ها مثل آنچه در شکل ۳.۳ آماده تشکیل شده باشد تصاویر واقعی به ندرت پله‌ای هستند. سطوح خاکستری در لبه‌ها به تدریج از پس زمینه به پیش زمینه تغییر می‌کنند. این نتایج به صورت پیک‌ها و دره‌ها در هیستوگرام دیده می‌شوند.

[1]- Bimodal

پیکسل‌های تقریبا مشابه یک کلاس را تشکیل می‌دهند (مثلا تصویر شیءای مثل صورت انسان). هیستوگرام یک تصویر برای هر کلاس از اشیاء یک پیک و همچنین برای پس زمینه یک پیک بزرگ خواهد داشت. برای تمایز بین k کلاس از اشیاء، باید k تا آستانه انتخاب کنیم،که هر کدام بین دو پیک قرار دارند. شکل 3.4.2 یک تصویر متشکل ازسه شئ را نمایش می‌دهد، و هیستوگرام‌های مربوطه در شکل ۳.۴ ب نشان داده شده اند.

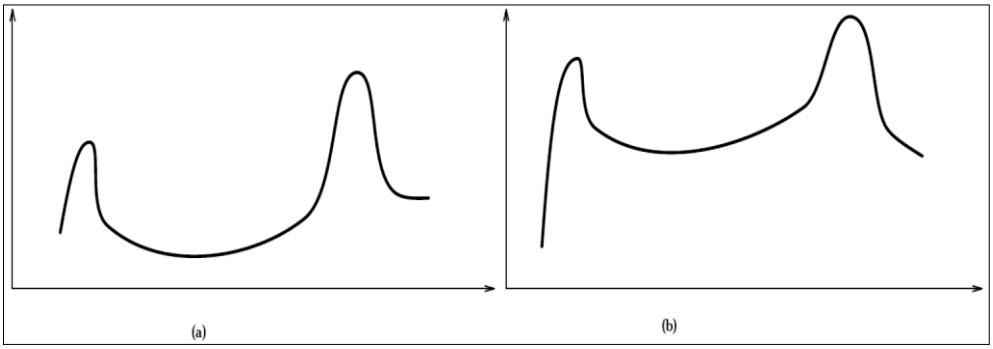
بانیزی را می‌توان به صورت زیر تولید کرد:

$$B_1(x,y) = \begin{cases} 1 & if\ 0 < f(x,y) < T_1 \\ 0 & otherwise \end{cases}$$

$$B_2(x,y) = \begin{cases} 1 & if\ T_1 < f(x,y) < T_2 \\ 0 & otherwise \end{cases}$$

$$B_3(x,y) = \begin{cases} 1 & if\ T_2 < f(x,y) < T_3 \\ 0 & otherwise \end{cases}$$

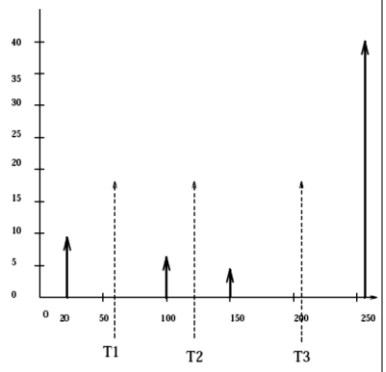
یک مسئله‌ی مهم در رابطه با هیستوگرام این است که فقط بین کلاس‌های اشیاء تمایز قائل می‌شود و هیچ اطلاعات مکانی راجع به آن‌ها نمی‌دهد. از این رو، یک تصویر یک صفحه‌ی نصفه سیاه و نصفه سفید، تصویر صفحه شطرنج و تصویر نقاط تصادفی سیاه و سفید با توزیع برابر همه، یک هیستوگرام دارند.



شکل ۳.۳:(a)–(b) مثال‌هایی از هیستوگرام دو مدله

255	255	255	255	255	255	255	20
255	255	255	100	100	255	20	20
255	255	255	100	100	255	20	20
255	255	255	100	100	255	20	20
255	255	255	255	255	255	20	20
255	255	255	255	255	255	255	255
150	150	255	255	255	255	255	255
150	150	255	255	255	255	255	255

(a)



(b)

شکل۳.۴:(a) تصویر متشکل از سه شئ. (b) هیستوگرام تصویر (a)

۲.۳.۳ آزمون پیک بودن

هیستوگرام تصاویر واقعی ممکن است به خاطر نویز حاوی پیک‌های کوچکی باشند. از این رو، همه‌ی پیک‌هارا نمی‌توان در بخش بندی استفاده کرد. قبل از اینکه بتوان پیک‌های هیستوگرام را در بخش بندی استفاده کرد، نیاز داریم که پیک‌های اصل را که متناظر نواحی تصویر هستند را مشخص کنیم. از آزمون پیک بودن برای این منظور استفاده می‌کنیم. یک پیک خوب است اگر تیز و عمیق باشد. یک پیک تیز است اگر ناحیه‌ی زیر آن کوچک باشد. ما از نسبت مساحت مستطیل در بردارنئه‌ی پیک به تعداد پیکسل‌های زیر پیک N، به عنوان معیاری برای تیزی پیک استفاده می‌کنیم. (شکل ۵.۳ را ببینید). عمق پیک ارتفاع نسبی پیک است. در این آزمون از اطلاعات زیر استفاده می‌کنیم:

۱.عرض پیک در سطح خاکستری از دره تا دره $= W$

۲.ارتفاع پیک $= P$

۳.مقادیر دره‌ها در دو طرف پیک $= V_a\ , V_b$

۴.تعداد پیکسل‌های تصویر پوشش داده شده توسط پیک $= N$

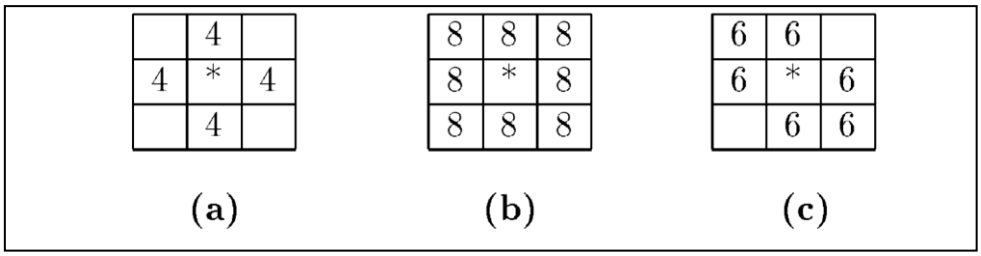
تیزی یک پیک به صورت نسبت $\frac{N}{(W \times P)}$ تعریف می‌شود. اگر این نسبت ۱ باشد، بدترین حالت ممکن را داریم، یک مستطیل. هرچه این نسبت کوچکتر شود، پیک تیزتر می‌شود. نسبت ارتفاع دره‌ها به ارتفاع پیک به صورت $\frac{(V_a + V_b)}{2P}$ است. در واقع آزمون پیک بودن ضرب این نسبت‌ها است:

$$Peakiness = \left(1 - \frac{(V_a + V_b)}{2P}\right) \times \left(1 - \frac{N}{(W \times P)}\right)$$

اگر معیار پیک بودن بزرگتر از آستانه باشد، در این صورت برای بخش بندی استفاده خواهد شد.

۴.۳ الگوریتم‌های عناصر متصل

تا اینجا، ما فقط از اطلاعات سطح خاکستری برای بخش بندی استفاده نمودیم و از اطلاعات مکانی نواحی هیچ استفاده‌ای نکردیم. همان طور که قبلاً ذکر شده، هیستوگرام هیچ اطلاعات مکانی نمی‌دهد. برای یافتن یک گروه از پیکسل‌های متصل در تصویر باید از یک الگوریتم عناصر متصل استفاده کنیم. سه نوع اتصال همانند آنچه در شکل ۶.۳ نشان داده شده وجود دارد. در اتصال چهارگانه (شکل ۶.۳ الف) یک پیکسل متصل به چهار همسایه‌ی خود در بالا،پایین و چپ و راست متصل در نظر گرفته می‌شود چون در فاصله‌ی یک از هم هستند. در اتصال هشت گانه عناصر قطری هم در نظر گرفته می‌شوند. از این رو، یک پیکسل متصل به همه‌ی همسایگانش در هشت طرف آن در نظر گرفته می‌شود (شکل ۶.۳ ب)



شکل ۶.۳: اتصال پیکسل‌ها.(a)اتصال چهار گانه (b)اتصال هشت گانه (c) اتصال شش گانه۱

۱.تصویر باینری را از چپ به راست و از بالا به پایین اسکن کن
۲.اگر پیکسل بدون برچسب با مقدار ۱ رسیدی به آن بر چسب بزن
۳.به صورت بازگشتی همه‌ی همسایه‌هاییکسل یافت شده را با همان برچسب، برچسب گذای کن اگر مقدار آن‌ها نیز ۱ است.
۴. وقتی همه‌ی پیکسل‌های دارای مقدار ۱ بر چسب گذاری شدند، متوقف شو.

۷.۳: الگوریتم عناصر متصل بازگشتی

در حوزه‌ی گسسته مناسب نیست که از فاصله‌ی اقلیدوسی استفاده کنیم. فاصله‌ی اقلیدوسی مرکز یک پیکسل از همسایه‌های قطری اش $\sqrt{2}$ است. در تصویر، فقط مقادیر صحیح موقعیت پیکسل‌ها معتبر است. از این رو، از سایر فاصله‌ها، مثل فاصله‌ی شطرنجی استفاده می‌شود. فاصله‌ی شطرنجی به صورت زیر تعریف می‌شود:

$$D_{ch}\big((x_1,y_1)(x_2,y_2)\big) = \max(|x_1 - x_2|, |y_1 - y_2|) \qquad (۳.۴)$$

بر طبق این فاصله همه‌ی هشت همسایه‌ی پیکسل مرکزی فاصله‌ی ۱ از آن دارند. ار این رو متصل در نظر گرفته می‌شوند. همچنین می‌توانیم از اتصال شش گانه که در شکل ۶.۳ C. نشان داده شده استفاده کنیم. در اتصال شش گانه هر دو عنصر قطری متصل در نظر گرفته می‌شوند.

در دو زیربخش آتی دو الگوریتم را بحث می‌کنیم: الگوریتم‌های بازگشتی و ترتیبی برای یافتن عناصر متصل در تصویر.

۱.۴.۳ الگوریتم بازگشتی

الگوریتم ارائه شده در شکل ۷.۳ به خوبی کار می‌کند و به آسانی پیاده سازی می‌شود. مشکل اینجاست که بازگشتی است. این الگوریتم، اگر روی یک کامپیوتر کوچک با پشته‌ی محدود پیاده شود ممکن است به راحتی به سرریز دچار شود، که توجه بیشتر برنامه نویس را می‌طلبد. برای یک تصویر ۵۱۲×۵۱۲ با یک چهارم میلیون پیکسل ممکن است چندین هزار فراخوانی داشته باشد.

۱.تصویر باینری را از چپ به راست و ار بالا به پایین اسکن کن.

۲.اگر پیکسل بدون بر چسبی مقدار "۱ " داشت، طبق قواعد زیر برچسب جدیدی به آن بزن:

0 → L

L 1 → L L

L 1 → M L

0 1 → 0 L

1 → 0 L

۳.کلاس‌های معادل بر چسب‌ها را مشخص کن

۴.در گذر دوم، به تمام المان‌ها در کلاس‌های معادل یک بر چسب تخصیص بده

۸.۳: الگوریتم عناصر پیوسته ترتیبی

۱.هیستوگرام تصویر داده شده را بگیر
۲.هیستوگرام را با میانگین گیری فرم کن تا پیک‌های کوچک حذف شوند.
۳. پیک‌ها و دره‌های کاندید را در هیستگرام مشخص کن
۴.یک‌های خوب را با اعمال آزمون پیک بودن تشخیص بده.
۵.با استفاده از آستانه‌ها در دره‌ها تصویر را بخش بندی کن
۶.الگوریتم عناصر متصل را اعمال کن

۹.۳: گام‌های بخش بندی با استفاده از هیستوگرام

۵.۳ بخش بندی دانه‌ای

حالا می‌توانیم گام‌های مختلف در الگوریتم بخش بندی دانه‌ای را به صورت شکل ۹.۳ خلاصه کنیم. در گام نخست، هیستوگرام تصویر محاسبه می‌شود. ممکن است هیستوگرام به خاطر نویز و نوردهی نامناسب حاوی پیک کوچکی باشد. طی گام دوم هیستوگرام نرم می‌شود تا این پیک‌های کوچک از بین بروند. در اصل،هر کدام از فیلترهایی که در فصل قبل بحث شد را می‌توان اینجا استفاده کرد. در اثر موارد، میانگین گیری روی ۳ عنصر هیستوگرام کافی است. گاهی، هیستوگرام های نویزی تر را می‌توان با چند بار فیلتر کردن نرم کرد. در گام سوم، پیک‌ها و دره‌های کاندید به راحتی با یافتن ماکسیمم و مینیمم محلی در هیستوگرام مشخص می‌شوند. سپس پیک‌های خوب با استفاده از آزمون پیک بودن، تعیین می‌شوند. در گام پنجم، تصویر با استفاده از آستانه‌ها در دره‌های بین پیک‌ها، بخش بندی می‌شود. گام آخر، برای مشخص کردن مجموعه‌ای از عناصر متصل در تصویر استفاده می‌شود. نتایج برای بخش بندی تصویر در شکل ۱۰.۳ نمایش داده شده اند.

۶.۳ رشد ناحیه

بخش بندی با استفاده از توزیع سطح خاکستری و اتصال پیکسل‌ها را می‌توان به عنوان بخش بندی دانه‌ای در نظر گرفت. این بخش بندی نیاز به پالایش با استفاده از اطلاعاتی راجع به شکل نواحی و معانی نواحی دارد. همچنین این بخش بندی را می‌تواند تعداد زیادی ناحیه کوچک را مشخص کند که باید نواحی همسایه آن‌ها را ترکیب کرد.ما الگوریتم‌های مختلفی را برای رشد نواحی بحث خواهیم کرد.

۱.۶.۳ الگوریتم تقسیم و ترکیب

این الگوریتم ساده ترین، الگوریتم برای رشد ناحیه است. در این الگوریتم بر حسب چند گزاره، به صورت ترتیبی نواحی تقسیم و سپس ترکیب می‌شوند تا آنکه هیچ چیزی برای تقسیم و ترکیب باقی نماند (همانند آنچه در شکل ۱۱.۳ نشان داده شده است) الگوریم به صورت خلاصه در شکل ۱۱.۳ نمایش داده شده است.

طرز کار الگوریتم را می‌توان با تصویر ساده‌ی شکل ۱۱.۳ نشان داد. در این مثال، از یک گزاره‌ی بسیار ساده استفاده می‌کنیم: (*مقادیر سطح خاکستری تمام پیکسل‌های درون یک ناحیه یکسان است*). مقادیر پیکسل‌ها در یک ناحیه یکسان است. برای شروع، کل تصویر را یک ناحیه در نظر می‌گیریم. از آن جایی که همه‌ی پیکسل‌های تصویر یک سطح خاکستری ندارند، تصویر به چهار بخش تقسیم می‌شود (شکل ۱۲.۳ الف).

سپس،سعی می‌شود که هر در ناحیه همسایه ترکیب شوند. اما هیچ یک از بخش‌ها شرط گزاره‌ی ذکر شده در بالا را ارضا نمی‌کنند و از این رو هیچ ترکیبی رخ نمی‌دهد. بعد همه‌ی بخش‌ها به جز بخش قرار گرفته در بالا، سمت چپ هریک به چهار بخش تقسیم می‌شوند (شکل ۱۲.۳ ب را ببینید). در این مرحله سه زیر بخش می‌توانند ترکیب شده به یک بخش بزرگتر تبدیل شوند (شکل۱۲.۳ ج). نهایتاً طی،همه‌ی نواحی شرط گزاره را ارضا کرده و هیچ تقسیم و ترکیب اضافه‌ای نیاز نیست.

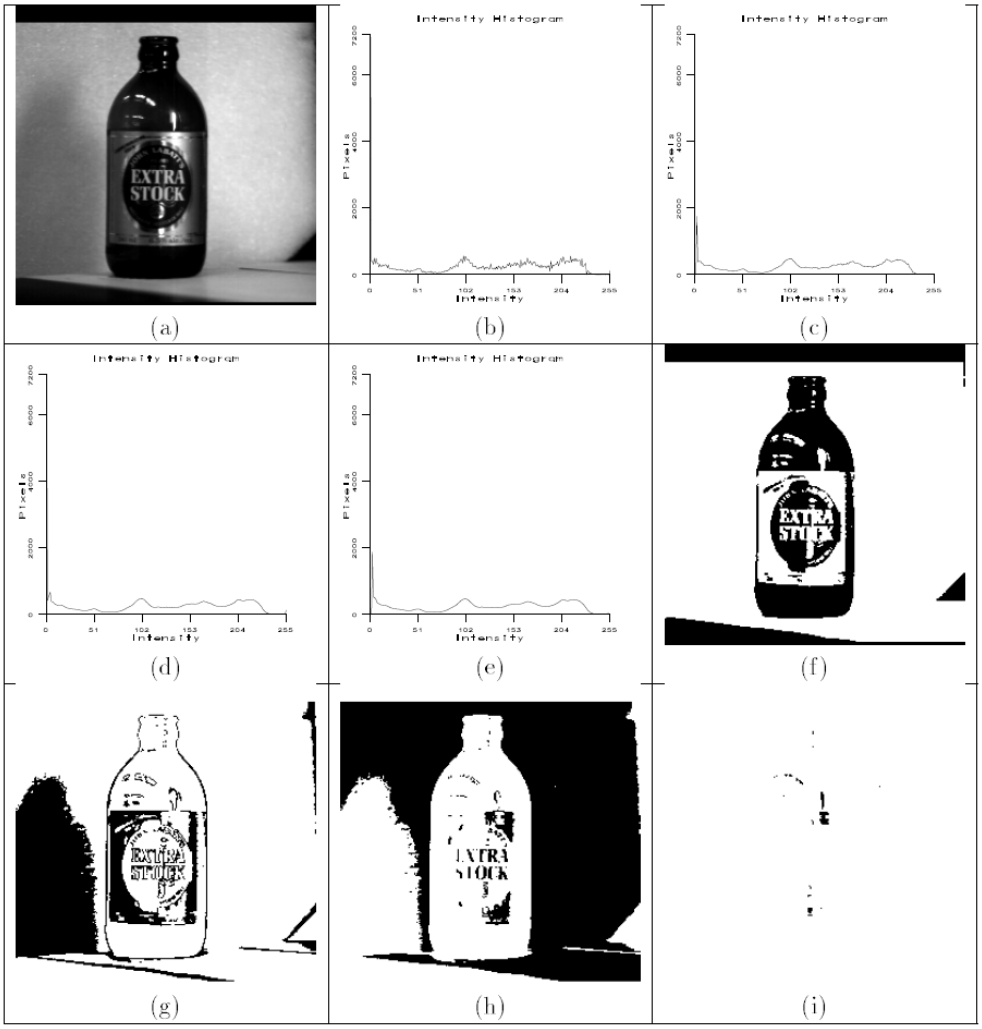
۲.۶.۳: الگوریتم بیگانه خوار

این الگوریتم را ذوب حدود[1] نیز می‌نامند. ایده حذف (ذوب) حدود ضعیف بین دو ناحیه‌ی همسایه ودر نتیجه ترکیب این دو ناحیه است. شهود سلول‌های بیگانه خوار برای حد بین دو ناحیه استفاده شده است. در این الگوریتم، لبه‌های شکاف بین دو ناحیه استفاده می‌شوند.(همانند شکل ۱۵.۳ الف). لبه‌ها را می‌توان به آسانی با یک فوق شبکه[2] $((2n+1)(2n+1))$ به صورت شکل ۱۵.۳ ب نمایش داد. قدرت لبه بین نقطه‌ی A در ناحیه‌ی R_1 و نقطه‌ی B در ناحیه‌ی R_2 به صورت اختلاف مطلق سطوح خاکستری به صورت زیر به دست می‌آید:

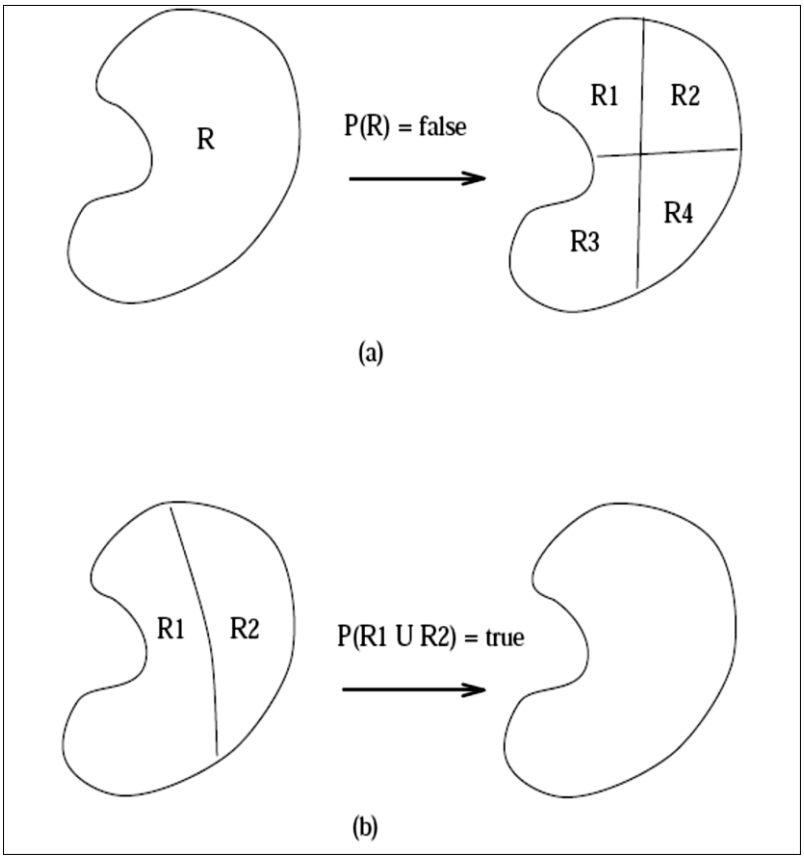
$$S(A,B) = \left|f(x_1\ ,y_1) - f(x_2,y_{2)}\right|$$

[1]- Boundary melting

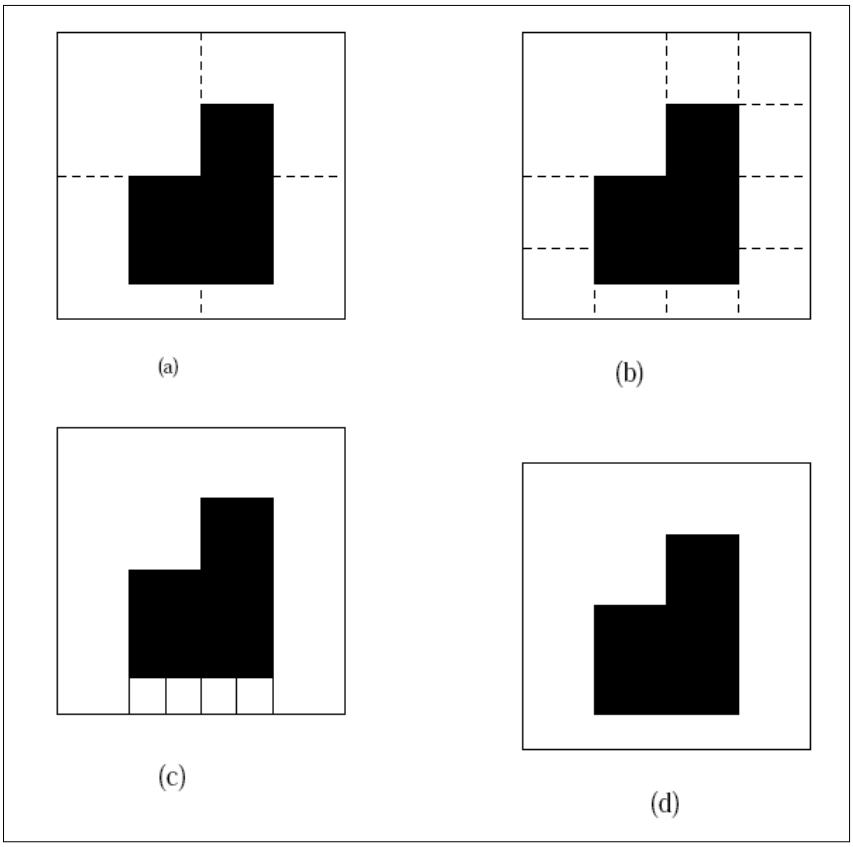
[2]-Grid



شکل ۱۰.۳: فرآیند نرم کردن هیستوگرام با استفاده از آستانه‌ی پیک بودن ۰.۱ (a) تصویر اصلی (b) هیستوگرام تصویر (a)، تعداد پیک‌ها برابر ۹۳، تعداد پیک‌ها پس از آزمون پیک بودن برابر ۱۸.(c) هیستوگرام پس از نرم کردن، تعداد پیک‌ها برابر ۵۴، تعداد پیک‌ها پس از آزمون پیک بودن برابر ۷. (d) هیستوگرام پس از نرم کردن، تعداد پیک‌ها برابر ۲۱۸، تعداد پیک‌ها پس از آزمون پیک بودن برابر ۷. (e) هیستوگرام پس از نرم کردن تعداد پیک‌ها برابر ۱۱، تعدادپیک‌ها پس از آزمون پیک بودن برابر ۴ و (f) نواحی مربوط به پیک ۱ (۰.۵). (g) نواحی مربوط به پیک ۲ (۱۱۶–۴۰). (i) نواحی مربوط پیک ۳ (۲۴۲–۱۱۶) . (h) نواحی مربوط به پیک ۴ (۲۴۳۰۰۲۵۵)



شکل ۱۱.۳: تقسیم و ترکیب (الف) اگر $p(r) = false$، ناحیه‌ی R به چهار زیر ناحیه‌ی R_1،R_2،R_3 و R_4تقسیم می‌شود و اگر $p(R_1 \cup R_2) = True$دو ناحیه‌ی R_1،R_2 ترکیب می‌شوند.



شکل ۱۲.۳: نمایش الگوریتم و تقسیم و ترکیب

۱.ناحیه‌ی Rرا به چهار بخش همجوار تقسیم کن اگر که Rگزاره را نقص کند.

$$Predicate(R) = false$$

۲. دو ناحیه‌ی مجاور R_1 و R_2 را ترکیب کن اگر

$$Predicate\ (R_1 \bigcup R_2) = true$$

۳. وقتی هیچ تقسیم و ترکیب دیگری ممکن نیست متوقف شو.

شکل ۱۳.۳: الگوریتم تقسیم و ترکیب

۱.دو ناحیه را ترکیب کنید اگر

$$\frac{W\ (Boundary)}{\min(P_1, P_2)} > T_2 \qquad 0 \leq T_2 \leq 1$$

که P_1 محیط بخش R_1و R_2 محیط بخش R_2 است.

۲. دو ناحیه را ترکیب کن اگر

$$\frac{W\ (Boundary)}{M} > T_3 \qquad 0 \leq T_3 \leq 1$$

که M تعداد کل نقاط روی لبه است.

شکل ۱۴.۳: الگوریتم بیگانه خوار

لبه ضعیف قلمداد می‌شود اگر (A و B) S زیر یک آستانه باشد:

$$W(A\ ,B) = \begin{cases} 1 & if\ S(A\ ,B) < T_1 \\ 0 & otherwise \end{cases}$$

ضعیف بودن کل لبه، مجموع اندازه‌ی ضعف همه‌ی نقاط لبه است:

$$W(Boundary) = \sum_{\forall\ A\ ,B} W\ (A\ ,B)$$

الگوریتم در شکل ۱۴.۳ آورده شده است. دو شهود اینجا اتخاذ شده است: ضعیف بودن و بیگانه خواری. شهود ضعیف بودن ساده و سر راست است. اکر نسبت نقاط ضعیف مرز به تعداد کل نقـاط مـرز بـین دو ناحیه‌ی مجاور از یک حد آستانه‌ای کمتر باشد، آنگاه این دو ناحیه ترکیب می‌شوند. با این حال این شهود متمایل است کل نواحی را ترکیب کند. بنابراین، نیاز یه شهود بیگانه خواری داریم، که شکل ناحیه منتج را در نظر می‌گیرد. نتایج بخش بندی تصویر در شکل ۱۶.۳ نمایش داده شده اند.

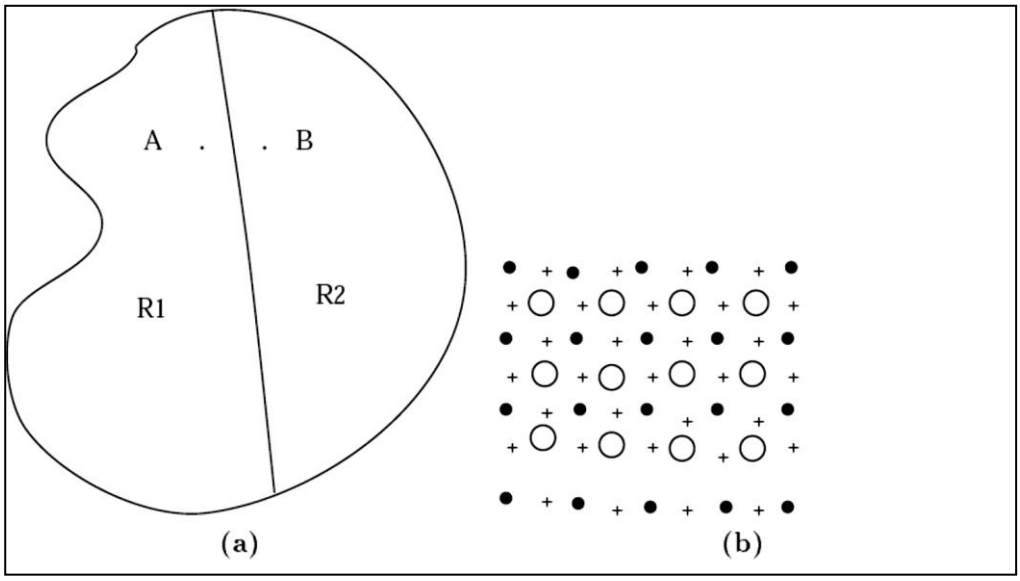
۳.۶.۳ آزمون نسبت شباهت

در این روش از توزیع سطح خاکستری پیکسل‌های بخش‌ها برای تصمیم گیری راجع بـه ترکیـب نـواحی استفاده می‌کنیم. طبیعت این روش احتمالی است. دو فرض H_1 و H_2 را به صورت زیر در نظر می‌گیریم:

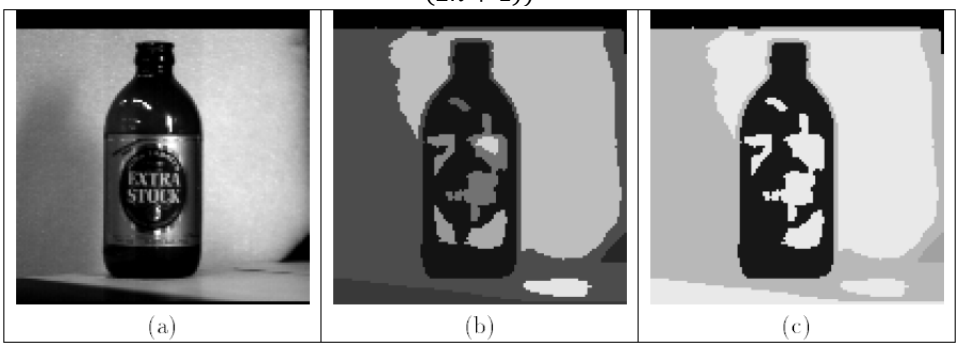
H_1: دو ناحیه وجود دارد

H_2: یک ناحیه وجود دارد

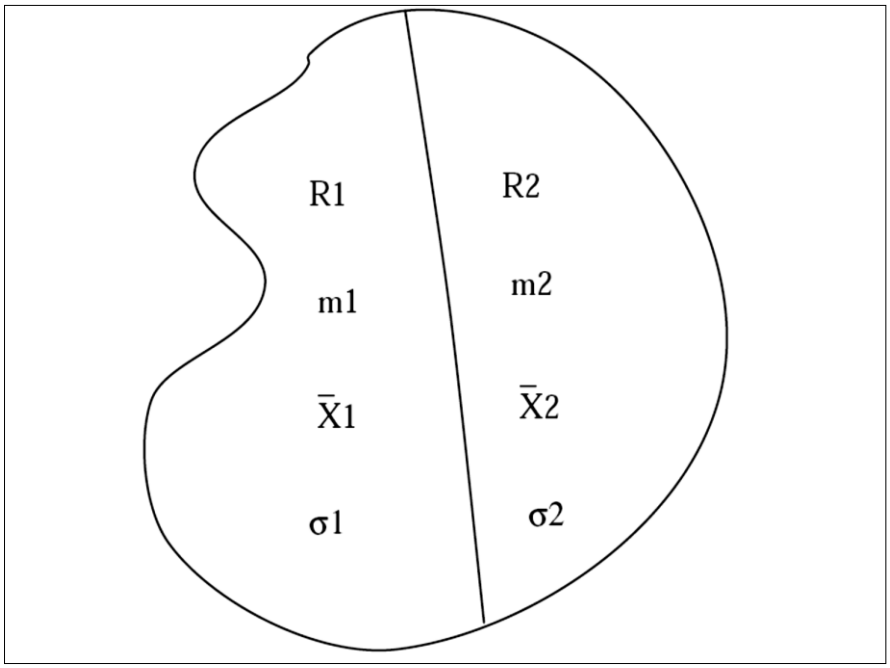
فرض خواهیم کرد که در هریک از بخش‌ها، توزیع سطح خاکسـتری تقریبـاً یـک توزیـع گوسـی اسـت. یعنی، اگر یک پیکسل به صورت تصادفی انتخاب شود، احتمال آنکه سطح خاکستری X را دارا باشـد بـه صورت زیر است:



شکل ۱۵.۳: ترکیب نواحی با استفاده از الگوریتم بیگانه خوار. (a) دو بخش R_1 و R_2. (b) فوق شبکه‌ی $((2n+1) \times (2n+1))$



شکل۱۶.۳: (a) تصویر بطری، (b) بخش بندی دانه‌ای (c) بعد از ترکیب بخش‌ها



شکل ۱۷.۳: ترکیب نواحی با استفاده از الگوریتم نسبت شباهت

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}$$

که x یا (X) میانگین شدت پیکسل‌هاست و σانحراف معیاراست.

فرض کنید که بخش R^1، m تا پیکسل دارد و x_1و σ_1 به ترتیب میانگین و انحراف معیار بخش R_1 هستند (شکل ۱۷.۳). همچنین فرض کنید که سطوح خاکستری مستقل از یکدیگر هستند. در این صورت، احتمال بودن $(x_1 , x_2 , \dots , x_{m_1})$در R_1 به صورت زیر است:

$$P(x_1 , x_2 , \dots , x_{m_1}) = (\frac{1}{\sqrt{2\pi}\sigma})^{m_1} e^{-\frac{m_1}{2}} \qquad (۳.۵)$$

به طرز مشابهی، می‌توانیم احتمالات را برای R_2 و ناحیه‌ی $R = R_1 \cup R_2$به صورت زیر تعریف کنیم:

$$P(x_{m_1+1}, x_{m_1+2}, \dots, x_{m_1+m_2}) = (\frac{1}{\sqrt{2\pi}\sigma_2})^{m_2} e^{-\frac{m_2}{2}} \qquad (۳.۶)$$

$$P(x_1 , x_2 , \dots, x_{m_1}, \dots, x_{m_1+1}, x_{m_1+2}, \dots, x_{m_1+m_2}) = (\frac{1}{\sqrt{2\pi}\sigma_2})^{m_1+m_2} e^{-\frac{m_1+m_2}{2}}$$

حالا احتمالات H_1 و H_2 به صورت زیر تعریف می‌شوند:

$$P(H_1) = P(x_1, x_2, \dots, x_{m_1}) \times P(x_{m_1+1}, x_{m_1+2}, \dots, x_{m_1+m_2}) = (\frac{1}{\sqrt{2\pi}\sigma_1})^{m_1} e^{-\frac{m_1}{2}} \times (\frac{1}{\sqrt{2\pi}\sigma_1})^{m_2} e^{-\frac{m_2}{2}} \quad (۳.۷)$$

$$P(H_2) = P(x_1, x_2, \dots, x_{m_1,\dots,} x_{m_1+1}, x_{m_1+2}, \dots, x_{m_1+m_2}) = (\frac{1}{\sqrt{2\pi}\sigma_2})^{m_1+m_2} e^{-\frac{m_1+m_2}{2}} \quad (۳.۸)$$

نسبت تشابه به صورت زیر است:

$$LH = \frac{P(H_1)}{P(H_2)} = \frac{(\sigma_0)^{m_1+m_2}}{(\sigma_1)^{m_1}(\sigma_2)^{m_2}}$$

اگر $LH < T$ بخش ترکیب می‌شوند.

۷.۳ گراف همجواری نواحی

گراف‌های همجواری نواحی (RAGs) ساختارهای بسیار مفیدی برای الگوریتم‌های ترکیب هستند. در این نمودار، نواحی به عنوان گره‌ها و نواحی همجوار به صورت یال‌هایی بین گره‌ها در نظر گرفته می‌شوند. یک تصویر ساده متشکل از پنج ناحیه ۱،۲،۳،۴ و ۵ را به صورت شکل ۱۸.۳ الف در نظر بگیرید. پیکسل‌های متناظر یک ناحیه با شماره‌ی ناحیه مشخص شده اند. فرض کنید که یک گزاره برای ترکیب نواحی استفاده شده است. ابتدا، نواحی ۳ و۴ ترکیب می‌شوند. تصویر با نواحی به روز شده در شکل ۱۸.۳ ب نمایش داده شده اند. سپس ناحیه ۳ با ناحیه Z به صورت نشان داده شده در شکل ۱۸.۳ ج ترکیب می‌شوند. در آخر، ناحیه یZ با ناحیه‌ی ۱ ترکیب می‌شوند. در این فرآیند بر چسب نواحی چندین بار تغییر داده می‌شود. برای مثال، بر چسب پیکسل‌های متعلق به ناحیه‌ی ۴ به ۳ و سپس به Z تغییر می‌کند و نهایتاً ۱ می‌شوند. در این مثال تصویر بسیار کوچک است و هر ناحیه تعداد کمی‌پیکسل تشکیل می‌شود. اما، غیر معمول نیست که چندین هزار پیکسل در هر ناحیه قرار داشته باشیم. در این صورت، تغییر برچسب‌ها نیازمند توان پردازشی افزون تری است. از این رو، به جای تغییر برچسب نواحی هر پیکسل طی فرآیند ترکیب، عملیات ترکیب را رویRAG به کار می‌بندیم. RAGهای متناظر اشکال ۱۸.۳ (a تا d) در اشکال ۱۸.۳ (e تا آخر) نمایش داده شده اند. وقتی ناحیه‌ی ۴ تصویر با ناحیه‌ی

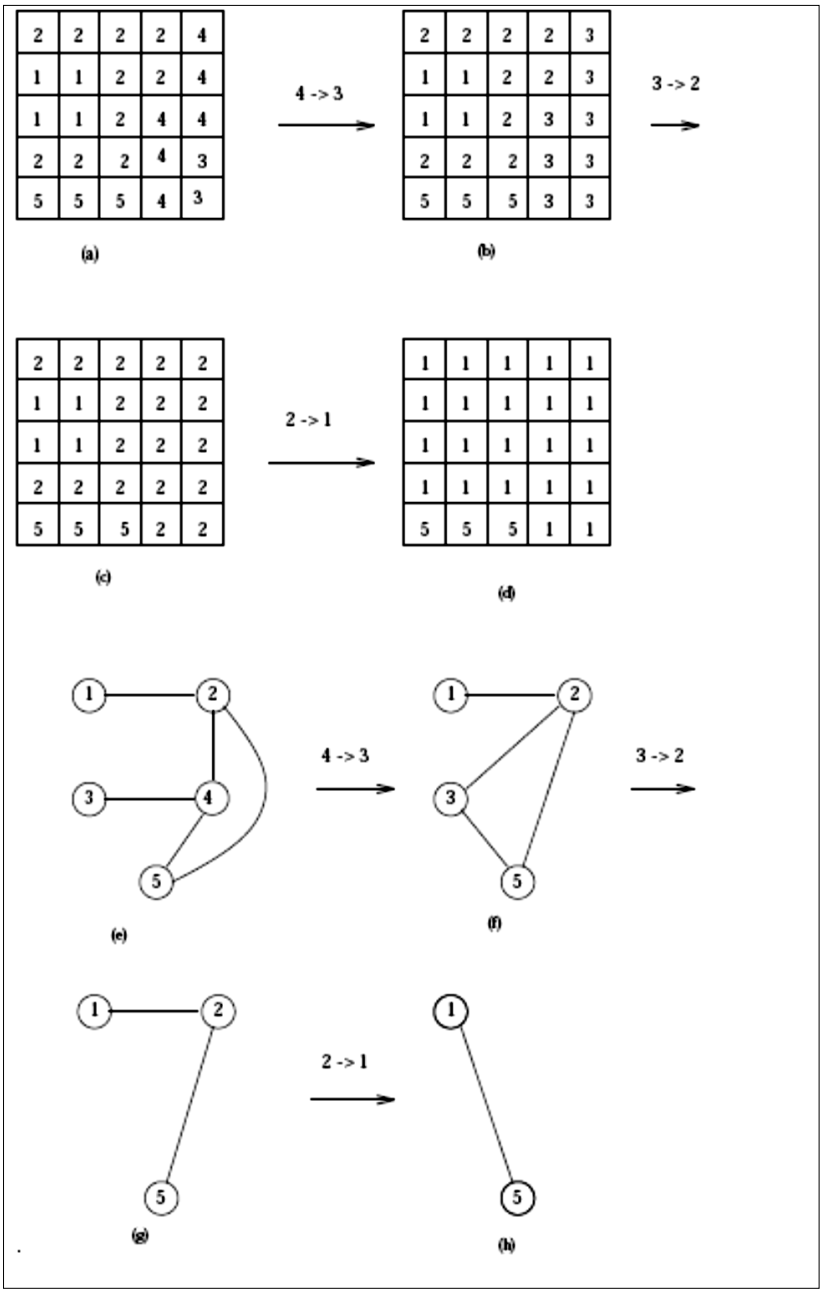
۳ ترکیب می‌شوند، گره‌ی ۴ در حذف شده و یال‌ها طوری تغییر داده می‌شوند که نماینده تصویر جدید همانطور که در شکل f.۱۸.۳ نمایش داده شده است، باشد. همچنین اینکه برچسب‌های ۳ و ۴ یکسان هستند، ضبط می‌شود. عملیات ترکیب برای تصویر شکل ۱۸.۳ a در h.۱۸.۳ با بر چسب‌های معادل و تصویر اولیه اش در شکل a.۱۸.۳ نمایش داده شده اند.

۳.۸ مشکلات رشد ناحیه

۱.تعداد آستانه‌های استفاده شده در الگوریتم. برخی الگوریتم‌ها از یک یا دو آستانه استفاده می‌کنند اما برخی دیگر چندین آستانه دارند.

۲.مرتبه‌ی ترکیب بسیار مهم است. بخش بندی نهای به شدت وابسته به مرتبه‌ای است که نواحی ترکیب شده اند.

۳.بخش بندی دانه‌ای نیز مهم است. اگر بخش بندی اولیه که با آن شروع می‌کنیم معقول باشد، رشد ناحیه با استفاده از اطلاعات شکل و معنایی آن را بهتر می‌کند.



شکل ۱۸.۳: گراف‌های همجواری نواحیRAG

با این حال، اگر بخش بندی اولیه خیلی بد باشد، رشد نواحی کمکی نخواهد کرد.

همانطور که گفته شد، لبه‌یابی و بخش بندی نواحی فرآیندهای مکمل هم هستند. در اصل ما نواحی را با لبه‌های اشیاء مشخص می‌کنیم و لبه‌ها نواحی را تشکیل می‌دهند. با این وجود، چندین تفاوت بین این دو نوع بخش بندی وجود دارد.

۱.بخش بندی نواحی به مرزهای بسته منتهی می‌شود، در حالی که مرزهای به دست آمده با لبه یابی لزوماً بسته نیستند.

۲.لبه یابی تقریباً محلی است. بخش بندی نواحی عمومی‌تر است.

۳.بخش بندی را می‌توان با استفاده از تصاویر چند طیفی (مثلاً تصاویر رنگی) بهبود داد در حالی که بهبود زیادی از طریق در روش‌های لبه یابی ایجاد نمی‌شود.

۴. موقعیت مرزها (لبه‌ها) در روش‌های لبه یابی محلی می‌شود اما این در مورد بخش بندی ممکن است صدق نکند.

۹.۳. ویژگی‌های هندسی نواحی

وقتی صحنه به نواحی تقسیم می‌شود، می‌توانیم ویژگی‌های نواحی را مشخص کنیم که در شناسایی الگو کاربرد دارد. در این بخش، چند تا از ویژگی‌های مهم را بحث می‌کنیم. فرض خواهیم کرد که نواحی با تصاویر $m \times n$ بانیزی، $B(x,y)$ نمایش داده می‌شوند.

مساحت: مساحت یا اندازه‌ی ناحیه تعداد پیکسل‌های اشغال شده توسط آن ناحیه است که به صورت زیر به دست می‌آید:

$$A = \sum_{x=0}^{m} \sum_{y=0}^{n} B(x,y) \qquad (۳.۱۰)$$

شبه مرکز: شبه مرکز شبیه مرکز نواحی با شکل‌های نامشخص است و می‌تواند برای تعیین موقعیت ناحیه استفاده شود. شبه مرکز، $(\bar{x}, \bar{y})$ به صورت زیر است:

$$\bar{x} = \frac{\sum_{x=0}^{m} \sum_{y=0}^{n} xB(x,y)}{A} \qquad (۳.۱۱)$$

$$\bar{y} = \frac{\sum_{x=0}^{m} \sum_{y=0}^{n} yB(x,y)}{A} \qquad (۳.۱۲)$$

ممان[1]: ممان‌های اول، M_x^1، M_y^1 و ممان‌های دوم M_x^2، M_y^2، به ترتیب به صورت زیر تعریف می‌شوند:

[1]- Moments

$$M_x^1 = \sum_{x=0}^{m}\sum_{y=0}^{n} x\, B(x,y) \quad (۳.۱۳)$$

$$M_y^1 = \sum_{x=0}^{m}\sum_{y=0}^{n} y\, B(x,y) \quad (۳.۱۴)$$

$$M_x^2 = \sum_{x=0}^{m}\sum_{y=0}^{n} x^2\, B(x,y) \quad (۳.۱۵)$$

$$M_y^2 = \sum_{x=0}^{m}\sum_{y=0}^{n} y^2\, B(x,y) \quad (۳.۱۶)$$

محیط[1]: محیط یک ناحیه تعداد پیکسل‌های روی مرز آن است. پیکسلی که حداقل یک پیکسل از پس زمینه اش فاصله دارد، پیکسل مرزی کفته می‌شود.

فشردگی[2]: فشردگی، C، به صورت زیر تعریف می‌شود:

$$C = 4\pi \frac{A}{P^2} \quad (۳.۱۷)$$

که A، مساحت ناحیه و P محیط ناحیه است. فشردترین ناحیه یک دایره است، با فشردگی برابر ۱ (مساحت دایره با شعاع R برابر πR^2، محیط آن برابر $2\pi R$ است). فشردگی یک شکل مربعی $\frac{\pi}{4}$ است.

جهت: جهت یک ناحیه، θ، را می‌توان با مشخص کردن، محورممان دوم اینرسی (شکل۱۹.۳) محاسبه کرد. با مینیمم کردن عبارت زیر می‌توان آن را محاسبه کرد:

$$E = \int\int (x\sin\theta - y\cos\theta + \rho)^2\, B(x,y)dx\, dy \quad (۳.۱۸)$$

می‌توان نشان داد که مینیمم کردن عبارت بالا (تمرینات را ببینید) به روابط پایین منتج می‌شود:

$$\sin 2\theta = \pm \frac{b}{\sqrt{b^2+(a-c)^2}} \quad (۳.۱۹)$$

$$\cos 2\theta = \pm \frac{a-c}{\sqrt{b^2+(a-c)^2}} \quad (۳.۲۰)$$

که

$$a = \int\int x'^2 B(x,y)dx'dy' \quad (۳.۲۱)$$

$$b = 2\int\int x'y' B(x,y)\, dx'dy' \quad (۳.۲۲)$$

$$c = \int\int y'^2 B(x,y)dx'dy' \quad (۳.۲۳)$$

$x' = x - \bar{x}$ و $y' = y - \bar{y}$، برای حوزه‌ی گسسته، a،b، x به صورت زیر می‌باشند:

$$a = \sum\sum x^2 B(x,y) - A\bar{x}^2 \quad (۳.۲۴)$$

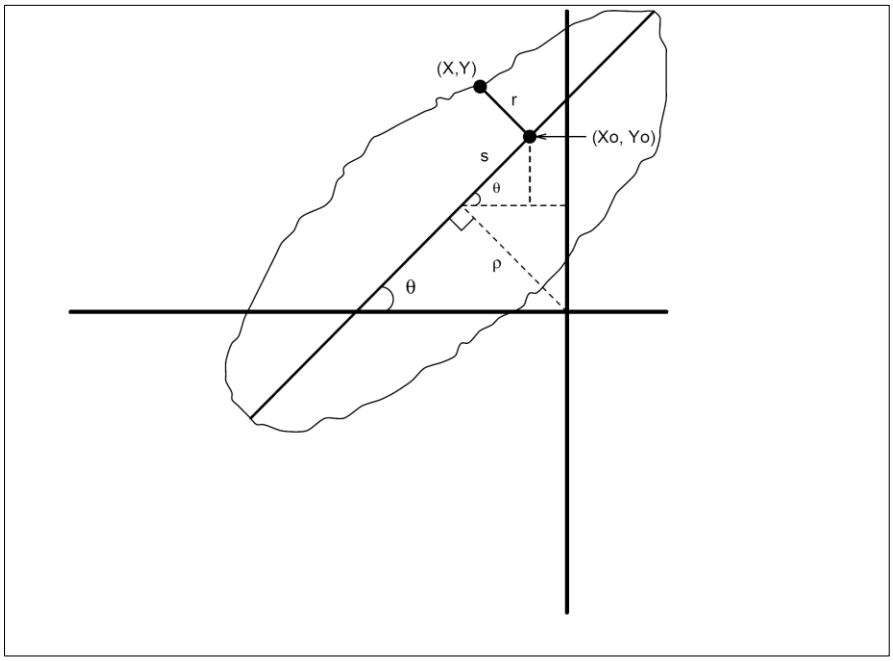
---

[1] - Perimeter

[2] - Compactness

$$b = 2\sum\sum xyB(x,y) - A\bar{x}\bar{y} \quad (۳.۲۵)$$

$$c = \sum\sum y^2B(x,y) - A\bar{y}^2 \quad (۳.۲۶)$$



شکل ۱۹.۳: مشخص کردن جهت یک ناحیه‌ی بی شکل

تمرینات

۱. الگوریتم ترتیبی sequential را با اتصال هشت گانه بنویسید.

۲. الگوریتمی‌براب مشخص کردن کلاس‌های معادل طراحی کنید.

۳. معادلات ۳.۶،۳.۵ و ۳.۷ را اثبات کنید.

۴. پیچیدگی محاسبات الگوریتم‌های ترتیبی و بازگشتی عناصر متصل را مشخص کنید.

۵. نشان دهید که معادله‌ی خط نشان داده شده در شکل ۱۹.۳ به صورت زیر است.

$$x\,sin\,\theta - y\cos\theta + \rho = 0 \quad (۳.۲۷)$$

نشان دهید که نزدیک ترین نقطه یخط مبدأ در $(-\rho\sin\theta\,, +\rho\cos\theta)$ و معادلات پارامتری بـرای یـک نقطه $(x_0\,, y_0)$ روی خط به صورت زیر هستند.

$$x_0 = -\rho\sin\theta + s\cos\theta \quad (۳.۲۸)$$

$$y_0 = +\rho\cos\theta + s\sin\theta \qquad (۳.۲۹)$$

که s به صورت نشان داده شده در شکل ۱۹.۳ است.

۶. فاصله‌ی r، بین نقطه (x،y) و یک خط به صورت زیر است، به جای x_0 و y_0، معادل آن‌ها در ۳.۲۸ و ۳.۲۹ را جایگذاری کنیم و از نتیجه را نسبت به s مشتق بگیرید و برابر صفر قرار دهید. حالا نشان دهید که s به صورت :

$$s = x\cos\theta + y\sin\theta \qquad (۳.۳۰)$$

است.

۷. معادله‌ی ۳.۳۱ را در معادلات ۳.۲۸ و ۳.۲۹ جایگذاری کنید و نشان دهید که

$$r^2 = (x\sin\theta - y\cos\theta + \rho)^2 \qquad (۳.۳۱)$$

۸. نشان دهید که مشتق معادله ۳.۱۸ نسبت به P و برابر قرار دادن آن با صفر منتج می‌شود به:

$$A\,(\bar{x}\sin\theta - \bar{y}\cos\theta + \rho) = 0 \qquad (۳.۳۲)$$

که $(\bar{x}, \bar{y})$ شبه مرکز و A مساحت است.

۹. با استفاده از $x' = x - \bar{x}$ و $y' = y - \bar{y}$، نشان دهید که E در معادله‌ی ۳.۱۸ به صورت زیر به دست می‌آید:

$$E = a\,sin^2\theta - b\sin\theta\cos\theta + c\,cos^2\theta \qquad (۳.۳۳)$$

که a و b و cدر معادلات ۳.۲۱ تا ۳.۲۳ نمایش داده شده اند و نشان دهید که E را می‌توان به صورت زیر نوشت:

$$E = \frac{1}{2}(a+c) - \frac{1}{2}(a-c)\cos 2\theta - \frac{1}{2}\,b\sin 2\theta \qquad (۳.۳۴)$$

۱۰. با گرفتن مشتق از معادله ۳.۳۴ نسبت به θ و برابر صفر قرار دادن آن، نشان دهید که

$$\tan 2\theta = \frac{b}{a-c}$$

در نتیجه،۳.۱۹ و ۳.۲۰ را تصدیق کنید.

فصل ۴

شکل ۲ بعدی

۱.۴ معرفی

در این فصل ما تکنیک‌های مختلف نمایش شکل حدود و ناحیه‌ها را در یک تصویر بخش‌بندی شده را بحث خواهیم کرد. این فرم از نمایش‌ها، انتزاعی شده‌ی لبه‌ها و ناحیه‌ها به صورت سمبلیک هستند ککه در شناسایی اشیا مفید خواهند بود. در اینجا بیشتر با عملیات‌های گروه‌سازی[1] سروکار خواهیم داشت. برای نمونه در گروه‌سازی نقاط روی لبه به خطوط مستقیم، دایره‌ها و بیضی‌ها و مانند اینها. برای نمایش ناحیه‌ها ما تبدیل محور میانی[2]، هرم‌ها و درخت‌های چهارتایی[3] را بحث خواهیم کرد. برای نمایش حدود تبدیل‌هاف[4] و تبدیل‌هاف تعمیم‌داده شده[5]، کد زنجیری و شماره‌ی شکل را بررسی خواهیم کرد. یک نمایش خوب از شکل باید فشرده، کامل و بدون ابهام و پایا باشد.

۲.۴تبدیل‌هاف

تبدیل‌هاف می‌تواند برای نمایش منحنی‌های در صفحه مثل خطوط، دایره‌ها و سهمی‌هایی که با عبارت-های تحلیلی توصیف می‌شوند استفاده شود. ارتباطاتی بین تبدیل‌هاف و برازش حداقل مربعات وجود دارد. در روش حداقل مربعات یک سیستم فرامعین از معادلات برای محاسبه‌ی راه حل استفاده می‌شود. از تعداد مجهولات بیشتر است. در تبدیل‌هاف از یک سیستم فرومعین برای محاسبه‌ی جواب استفاده می‌شود. از اینرو تعداد معادلات کمتر از تعداد مجهولات است. در این حالت چندین جواب به دست می‌آید. در واقع، در بسیاری از حالات، در تبدیل‌هاف یک قید محدودکننده[6] برای محاسبه‌ی همه‌ی جواب‌های ممکن استفاده می‌شود. به این طریق هر قید محدود کننده به مجموعه‌ای از جواب‌های ممکن

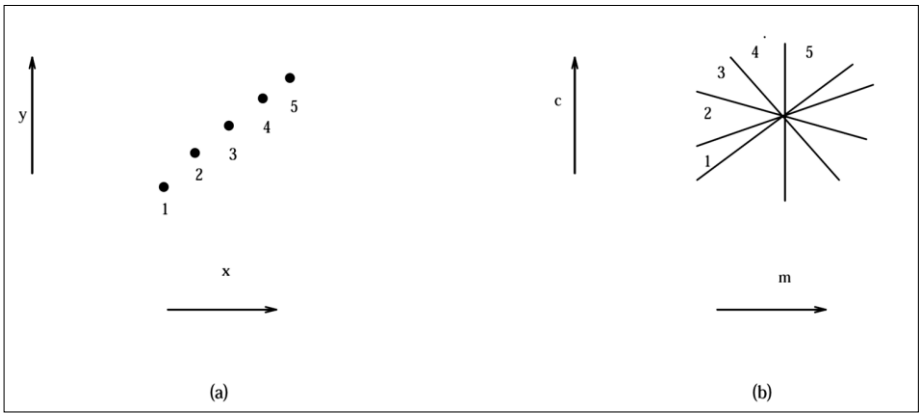
---

1 - Grouping Operations
2 - Medial axis transform
3 - Quad trees
4 - Hough Transform
5 - Generalized
6 - Constraint

منتج می‌شود. جوابی که بیشترین رای‌ها را داشته باشد به عنوان پاسخ نهایی انتخاب می‌شود. یک حالت سومی‌هم هست که در آن تعداد معادلات با تعداد مجهولات برابر است. این روش RANSAC[1] نام دارد. گام‌های اصلی در RANSAC به صورت زیر هستند:

۱- حداقل تعدادی ازمحدودکننده‌ها را به صورت تصادفی برای به دست آوردن یک جواب تخمینی انتخاب کن.

۲- خطای بین جواب تخمینی به دست آمده و تمام نقاط داده را پیدا کن. اگر خطا کمتر از تحمل است (یک مقدار آستانه ای) آن را به عنوان جواب نهایی انتخاب کن و خارج شو وگرنه به ۱ برو.



شکل ۱-۱ برازش خط با استفاده از تبدیل‌هاف; a- نقاط در فضای x و y ; b- خطوط در فضای c-m متناظر به نقاط در فضای x-y. این خطوط در یک نقطه یکدیگر را قطع می‌کنند.

۴.۲.۱ خطوط راست

در این بخش از تبدیل‌هاف برای برازش معادله‌ی یک خط به نقاط روی یک لبه استفاده خواهیم کرد.معادله‌ی خط به صورت زیر داده شده است:

$$y = mx + c \qquad (۴.۱)$$

که c و m شیب وعرض از مبدا هستند معادله‌ی بالا می‌توانست به صورت زیر نوشته شود:

$$c = (-x)m + y$$

این معادله‌ی خط در فضای $\mathrm{c} - \mathrm{m}$ با شیب $-x$ و عرض از مبدا y است.

[1] - Random Sampling and consensus

از این رو یک نقطه (x, y) در فضای y و x به یک خط در فضای $m - c$ نگاشت می‌شود.(شکل ۴.۱ را ببینید). فرض کنید که چند نقطه‌ی (x_1, y_1) ,(x_2, y_2),(x_3, y_3) و... (x_n, y_n) را در یک فضای $x - y$ روی یک لبه داریم و میخواهیم یک خط به آن‌ها برازش کنیم. هر نقطه در فضای (x, y) به یک خط در فضای $c - m$ نگاشت می‌شود. این خطوط به یک نقطه در فضای c-m نگاشت میشوند. آن نقطه $(\hat{c}, \hat{m})$شیب و عرض از مبدا تخمین زده شده یک خط در فضای (x, y) را به دست می‌دهد. الگوریتم برازش خطوط مستقیم به نقاط لبه با استفاده از تبدیل‌هاف در شکل ۴.۲ داده شده است.

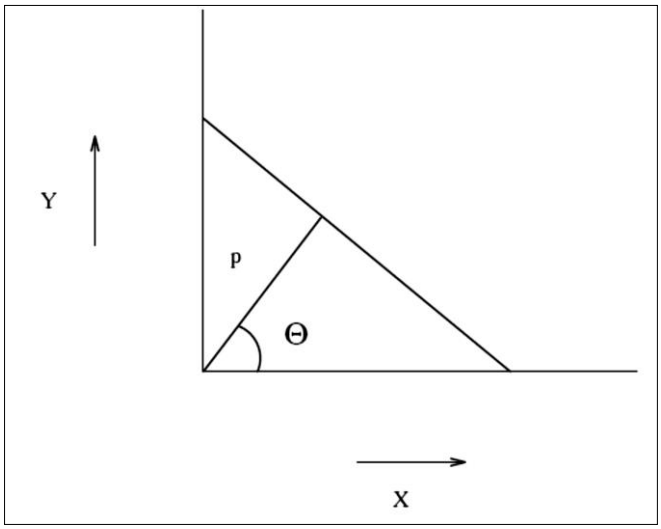
پارامتری کردن خط داده شده در معادله ۴.۱ یک مسئله دارد. خطوط موازی محور هاy دارای شیب m برابر بی نهایت هستند که نمیتوان آن را در کامپیوتر نمایش داد. یک روش دیگری پارامتری کردن خط نشان داده در شکل ۴.۳ به صورت زیر است:

$$p = xcos\theta + ysin\theta \qquad (۴.۲)$$

که θ زاویه محورx و خط p است که از مبدا به صورت عمود بر خطی که میخواهیم تشخیص داده شود رسم می‌کنیم. در این موردθ , و p مقادیر متناهی خواهند داشت. مزیت دیگر استفاده از اینطور پارامتری کردن این است که θ را میتوان حین فرآیند تشخیص لبه از روی زاویه شیب محاسبه کرد. از این رو به جای تکرار برای همه‌ی مقادیر ممکن θ, می‌توانیم θ را از زاویه شیب به دست آوریم. به این طریق, پیچیدگی محاسبات کاهش می‌یابد. الگوریتم برای برازش خط با استفاده فرم قطبی خط راست در شکل ۴.۴ داده شده است.

۱. پارامترهای فضا $P[c_{min},\ldots,c_{max},m_{min},\ldots,m_{max}]$ را کوانتیزه کن.
۲. برای هر نقطه‌ی (x, y) انجام بده
برای $(m = m_{min}, m \leq m_{max}, m++)$

۴.۲. الگوریتم تبدیل‌هاف برای برازش خط مستقیم



شکل ۴.۳. شکل قطبی معادله‌ی خط راست

۱. به صورت تکرارشونده g^k را به صورت زیر بنویسید:

$$g^k(x,y) = \begin{cases} \max[0,(g^{k-1}(p,q)-1)] \; if \; g^{k-1}=0 \\ g^{k-1} \qquad\qquad otherwise \end{cases} \tag{۴.۳۶}$$

۲. $\forall(p,q)$ طوری که فاصله‌ی $\big((x,y),(p,q)\big) \leq 1$.

شکل ۴.۴. الگوریتم تبدیل‌هاف با استفاده از فرم قطبی معادله‌ی خط راست

۱. پارامترهای فضایی را کوانتیزه کن

$$P\,[x_{0_{\min}},\dots,x_{0_{max}},\dots.,y_{0_{\min}},\dots,x_{0_{max}},x_{0_{\min}},\dots,x_{0_{max}}]$$

۲.برای هر نقطه‌ی لبه‌ای (x,y) انجام بده:

$$\text{for}(x_0 = x_{0_{\min}}, x_0 \leq x_{0_{\max}}, x_0 ++)$$
$$\text{for}(x_0 = y_{0_{\min}}, y_0 \leq y_{0_{\max}}, y_0 ++)$$
$$\text{do } (x-x_0)^2 + (y-y_0)^2 = r^2.$$

۴. نقاط ماکسیمم نسبی را در فضا پیدا کنید.

۴.۵. الگوریتم تبدیل‌هاف برای برازش دایره

۱.پارامترهای فضایی را کوانتیزه کن

$$P\ [x_{0\min}, \dots, x_{0max}, \dots., y_{0\min}, \dots, x_{0max}, x_{0\min}, \dots, x_{0max}]$$

۲.برای هر نقطه‌ی لبه‌ای (x, y) انجام بده:

$\text{for}(r = r_0, r \le r_{max}, r++)$
$x_0 = x - rcos\theta$
$y_0 = x - rsin\theta$
$P[x_0, y_0, r] = P[x_0, y_0, r] + 1$

۴. نقاط ماکسیمم نسبی را در فضا پیدا کنید.

۴.۶. الگوریتم تبدیل‌هاف برای برازش دایره با استفاده از فرم قطبی معادله‌ی یک دایره

۴.۲.۲ دایره

معادله‌ی یک دایره به مرکز (x_0,y_0) با شعاع r به صورت زیر است:

$$(x - x_0)^2 + (y - y_0)^2 - r^2 = 0 \qquad (4.3)$$

در این مورد ما سه مجهول داریم: y_0, x_0 و r. از این رو فضای پارامتری سه بعدی است. الگوریتم برای برازش منحنی با استفاده از تبدیل‌هاف در شکل ۴.۵ داده شده است. اطلاعات زاویه‌ی گرادیان در این مورد هم می‌تواند برای کاهش پیچیدگی محاسباتی نوع دیگری پارامتری کردن یک دایره به صورت زیر داده شده است:

$$x_0 = x - rcos\theta \qquad (۴.۴)$$

$$y_0 = y - rsin\theta \qquad (۴.۵)$$

که θ زاویه‌ی گرادیان است. الگوریتم ساده شده برای برازش دایره در شکل ۴.۶ داده شده است.

۴.۳ تبدیل‌هاف تعمیم داده شده

تاکنون تبدیل‌هاف برای تشخیص اشیایی که قابل بیان با عبارات تحلیلی بودند استفاده شد. با این حال, بسیاری از اشکال در دنیای واقعی را نمی‌توان با عبارات تحلیلی توصیف کرد. در این موارد, تبدیل‌هاف تعمیم داده شده را می‌توان استفاده کرد تا هر شکل دلخواهی را بیابیم. اطلاعات درون معادله‌ی شکل را می‌توان به صورت یک جدول به نام جدول R- گرفت. اولین گام در تبدیل‌هاف تعمیم داده شده, توسعه‌ی این جدول R- است. ابتدا مرکز شی به صورت زیر مشخص می‌شود:

$$x_c = \frac{(\sum_{x=0}^{x=n}\sum_{y=0}^{y=n} f(x,y)x)}{(\sum\sum f(x,y)} \qquad (۴.۶)$$

$$y_c = \frac{(\sum_{x=0}^{x=n}\sum_{y=0}^{y=n} f(x,y)y)}{(\sum\sum f(x,y)} \qquad (۴.۷)$$

که $f(x,y)$ تصویر باینری است که به صورت زیر تعریف می‌شود:

$$f(x,y) = \begin{cases} 1 & if\ (x,y)\ \in Object \\ 0 & otherwise \end{cases} \quad (۴.۸)$$

سپس, برای هر نقطه‌ی (x,y), بردار $r = (x',y')$ (همانگونه که در شکل ۴.۷.۸ نشان داده) به صورت زیر مشخص می‌شود:

$$x_c = x + x' \quad (۴.۹)$$

$$y_c = y + y' \quad (۴.۹)$$

اطلاعات راجع به بردار r در جدول R- جمع آوری می‌شوند(شکل b.۴.۷) که با زاویه‌ی گرادیان اندیس-گذاری شده‌اند. برای هر مقدار ممکن زاویه گرادیان در جدول R- یک ردیف وجود دارد. در ستون دوم این جدول بردارهای r با همان زاویه‌ی θ ذخیره شده اند.

طی فاز تشخیص, هدف کشف وجود یک شکل مشخص است که توسط جدول R- تعریف شده و تشخیص محل قرارگیری آن است. برای هر نقطه‌ی لبه, زاویه‌ی گرادیان تعیین می‌شود و برای اندیس گذاری جدول R- استفاده می‌شود. آرایه‌ی انباره توسط همه‌ی بردارهای r موجود در ورودی به صورت زیر افزوده می‌شود:

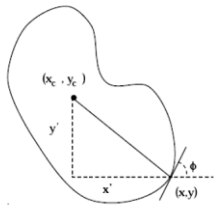
$$x_c = x + x' \quad (۴.۱۱)$$

$$y_c = y + y' \quad (۴.۱۲)$$

محل شکل با مشخص کردن ماکسیمم محلی در آرایه‌ی انباره $p[x_c, y_c]$ مشخص می‌شود. الگوریتم کامل در شکل ۴.۸ ارائه شده است. این الگوریتم قادر به تشخیص نسخه‌ی چرخیده شده و تغییر مقیاس شده‌ی اشیاء نخواهد بود. با این حال, برای سیستم‌های بینایی توانایی تشخیص نسخه‌ی چرخیده شده, تغییر مقیاس داده شده و جابه‌جا شده‌ی اشیاء بسیار مهم است. از این رو, ما اینکه چگونه الگوریتم را برای دستیابی به این مهم تصحیح کنیم را بررسی خواهیم کرد. می‌دانیم که اگر (x',y') حول محور Z با زاویه‌ی θ چرخیده شوند, مختصات جدید (x'',y'') به صورت زیر به دست می‌آیند:

$$x'' = x'cos\theta + y'sin\theta \quad (۴.۱۵)$$

$$y'' = -x'sin\theta + y'cos\theta \quad (۴.۱۶)$$



ϕ_1	$r_1^1, r_2^1, r_3^1, \ \dots\ , r_{m_1}^1$
ϕ_2	$r_1^2, r_2^2, r_3^2, \ \dots\ , r_{m_2}^2$
ϕ_3	$r_1^3, r_2^3, r_3^3, \ \dots\ , r_{m_3}^3$
•	•
•	•
•	•
ϕ_n	$r_1^n, r_2^n, r_3^n, \ \dots\ , r_{m_n}^n$

(a) (b)

شکل ۴.۷. تبدیل‌هاف تعمیم یافته. (a) یک شکل دلخواه با مرکز (x_c, y_c). زاویه‌ی گرادیان در نقطه‌ی $(x, y), p$ می‌باشد و x' و y' به ترتیب فاصله‌ی افقی و عمودی نقطه‌ی (x, y) از مرکز شی می‌باشند. (b). جدول R- اولین ستون حاوی زاویه‌های گرادیان و ستون دوم حاوی بردارهای x' و y' است.

۱. پارامتری‌های فضا را کوانتیزه کن.

$$P\ [x_{0_{\min}}, \dots, x_{0_{max}}, \dots., y_{0_{\min}}, \dots, x_{0_{max}}, x_{0_{\min}}, \dots, x_{0_{max}}]$$

۲. برای هرنقطه‌ی لبه‌ای (x, y) انجام بده:

$\emptyset(x, y)$ را محاسبه کن

برای هر المان $\emptyset$ انجام بده:

$$x_c = x + x' \qquad (۴.۱۳)$$

$$y_c = y + y' \qquad (۴.۱۴)$$

$$P[x_c, y_c] = P[x_c, y_c] + 1$$

۳. ماکسیم‌های نسبی را در فضای پارامتری بیاب.

شکل۴.۸ الگوریتم تبدیل‌هاف تعمیم یافته

به طرز مشابهی، نسخه‌ی چرخانده شده و تغییر مقیاس شده‌ی (x', y') به این صورت هستند:

$$x'' = s_x(x'cos\theta + y'sin\theta) \qquad (۴.۱۷)$$

$$y'' = s_y(-x'sin\theta + y'cos\theta) \qquad (۴.۱۸)$$

با جایگذاری معادلات بالا در معادلات ۴.۹ و ۴.۱۰ به دست می‌آوریم:

$$x_c = x + s_c(x'cos\theta + y'sin\theta) \qquad (۴.۱۹)$$

$$y_c = y + s_y(-x'sin\theta + y'cos\theta) \qquad (۴.۲۰)$$

اکنون اگر در الگوریتم‌هاف تعمیم یافته در شکل ۴.۸ معادلات ۴.۱۳ تا ۴.۱۴ با معادلات بالا تعویض شوند نسخه‌ی چرخانده شده, تغییر مقیاس شده‌ی اشکال قابل تشخیص می‌شوند.

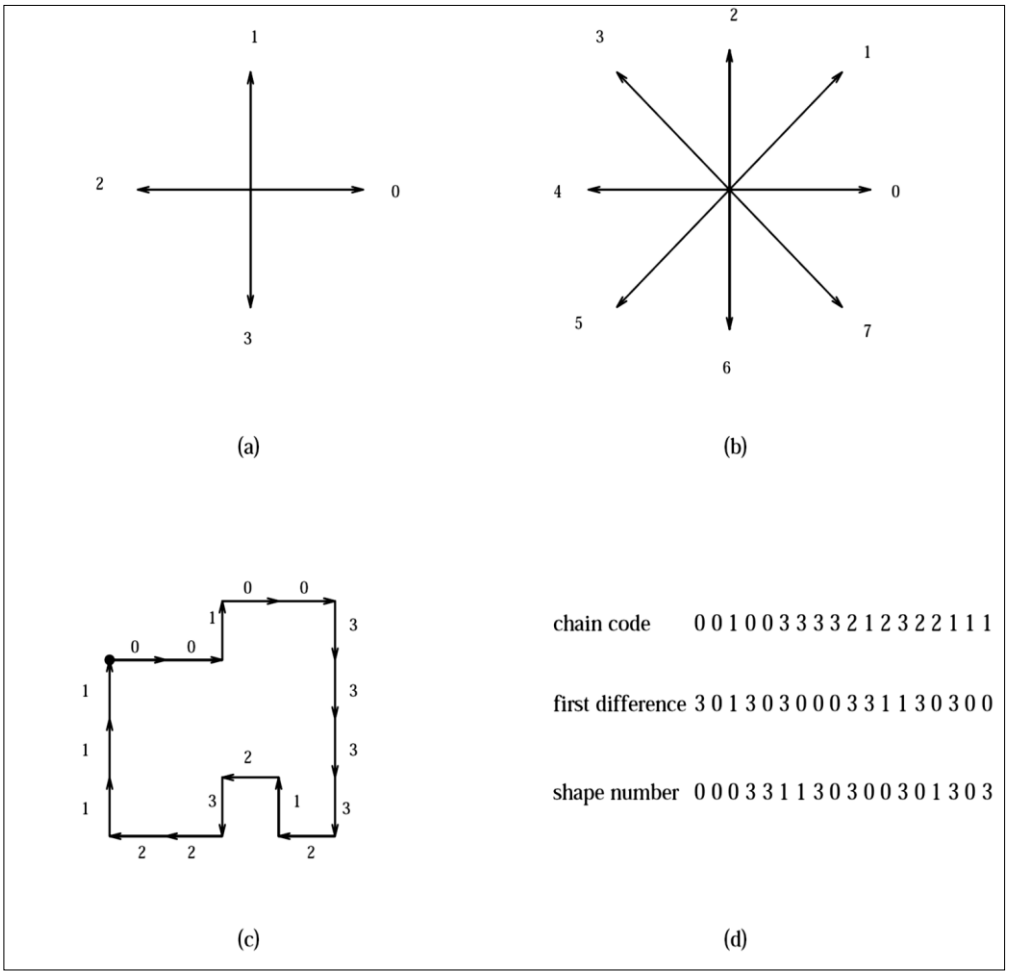
۴.۴ نمره‌ی شکل

کد زنجیره‌ای (chain code) یک روش ساده برای نمایش شکل یک کانتور است. در کد زنجیره‌ای, به هر بخش خط یک کد تخصیص داده می‌شود. دو کد ممکن در شکل ۴.۹ الف و ب نشان داده شده اند. در شکل a.۴.۹ جهت‌ها به چهار جهت ممکن کوانتیزه شده اند: ۰, ۹۹ , ۱۸۰ , ۲۱۰ درجه. کدهای معادل برای این جهت‌ها ۰ , ۱, ۲و ۳ هستند. زنجیره‌ی کد یک کانتور الحاق کدهای بخش‌های منفرد خطی است. زنجیره‌ی کد یک کانتور که در شکل b.۴.۹ نشان داده شده اند در شکل d.۴.۹ نمایش داده شده اند. توجه کنید که این کد زنجیره‌ای وابسته به نقطه‌ی شروع و جهت کانتور است. مطلوب است کدی داشته باشیم که مستقل از جا به جایی , چرخش و نقطه‌ی شروع باشد. اگر کانتور به اندازه‌ی $m \times 90$ درجه چرخانده شود (m یک عدد صحیح است) , در آن صورت در عمل یک m ثابت به کد هم پاره خط اضافه می‌شود.

از آن جایی که مشتق یک عدد ثابت صفر است، استفاده از مشتق کد زنجیره‌ای بجای خود کد زنجیره‌ای می‌توان آن را نسبت به چرخش مستقل کرد. همچنین کد می‌تواند مستقل از نقطه‌ی شروع شود به این صورت که اولین تفاضلات را طوری نرمالیزه کنیم که اگر کد به عنوان یک عدد صحیح تفسیر شود، آن کوچکترین عدد صحیح ممکن باشد. اولین تفاضلات نرمالیزه شده‌ی یک کد زنجیره‌ای نمره‌ی شکل نام دارد. تعداد ارقام در یک نمره‌ی شکل، مرتبه‌ی نمره‌ی شکل نامیده می‌شود. چند مثال از نمره‌ی شکل با مرتبه‌ی ۴، ۶ و۸ در شکل ۴.۱۰ نمایش داده شده‌اند.

یافتن نمره‌ی شکل هر تعداد شیء که در جهت‌های مختلفی قرار دارند، شامل چندین گام می‌شود. برای بدست آوردن یک نمره‌ی شکل دقیق با تعداد ارقام کمتر، باید کانتور را با شبکه‌ی درشت‌تری که با محورهای شئ هم خط هستند، دوباره نمونه‌برداری کرد.[درواقع Down sampling انجام می‌دهد.] این خطای کوانتیزیشن را حداقل خواهد کرد.

یک راه ممکن برای بدست آوردن این اعمال، شبکه‌ای است که محورهای اصلی و فرعی آن موازی محورهای اصلی و فرعی مستطیل دربرگیرنده‌ی شئ باشد و بی شکلی شبکه و مستطیل دربرگیرنده تقریبا یکی باشد. الگوریتم کامل در شکل ۴.۱۱ آمده است:



شکل ۴.۹: نمره‌ی شکل. (a) و (b): دو زنجیره‌ی کد ممکن. (c): یک کانتور با یک زنجیره‌ی کد. (d): نمره‌ی شکل کانتور نشان داده شده در (c).

۴.۵ هرم‌ها

هرم‌ها ساختارهای داده‌ی بسیار مفیدی برای نمایش نواحی تصویر هستند. هرم توسط چندین کپی از تصویر ساخته می‌شود. هر سطح در هرم به اندازه‌ی $\frac{1}{n}$ اندازه‌ی تصویر قبلی‌اش است. پایین‌ترین سطح در هرم، بالاترین رزولوشن و بالاترین سطح هرم، پایین‌ترین رزولوشن را دارد.

شکل ۴.۱۲ را به عنوان نمونه نگاه کنید.

۴.۵.۱ هرم گوسی

در هرم گوسی پایین‌ترین سطح g_0، تصویر اصلی I است. سطح g_L با میانگین‌گیری وزن‌دار سطح g_{L-1} در یک پنجره‌ی ۵×۵ به صورت زیر تعریف می‌شود:

$$g_l(i,j) = \sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m,n)\, g_{l-1}(2i+m, 2J+n) \qquad (۴.۲۱)$$

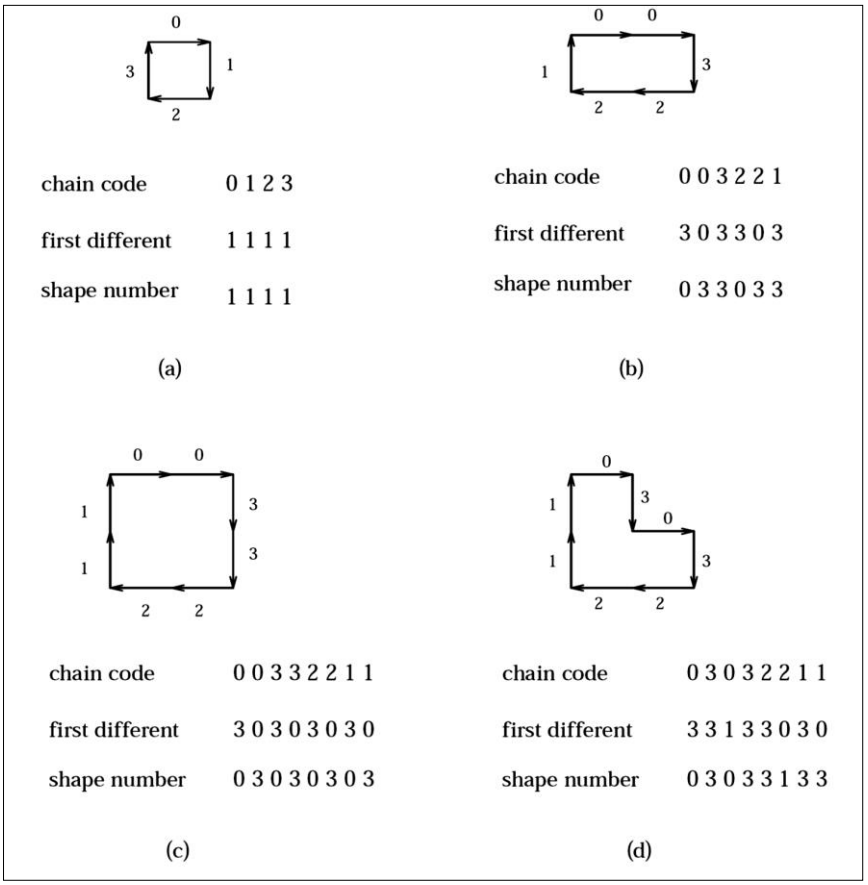
در این شکل ۴.۱۳ برای حالت یک بعدی نشان داده شده است. توجه کنید که چگالی نقاط در سطح، در یک بعد (دو بعد) نصف (یک چهارم) تعداد نقاط در سطح L-1 است. از این رو، این فرایند عملگر REDUCE نام دارد:

$$g_L = REDUCE\,[g_{L-1}]$$

ماسک $w(m,n)$، تابع گوسی را تخمین می‌زند و تفکیک‌پذیر است، یعنی:

$$w(m,n) = \hat{w}(m)\hat{w}(n)$$

ماسک یک بعدی $w(m)$ با سه قید محدودکننده تولید می‌شود:



شکل ۴.۱۰. برخی نمونه‌ها از خود شکل. (a): نمره شکل مرتبه‌ی ۴. (b): نمره شکل مرتبه‌ی ۶ (c) – (d) نمره شکل مرتبه ۸

– خروج از مرکز را بیاب

(الف) دو نقطه را که حداکثر فاصله از هم دارند بیاب. خطی که دو خط را متصل می‌کند محور اصلی است.

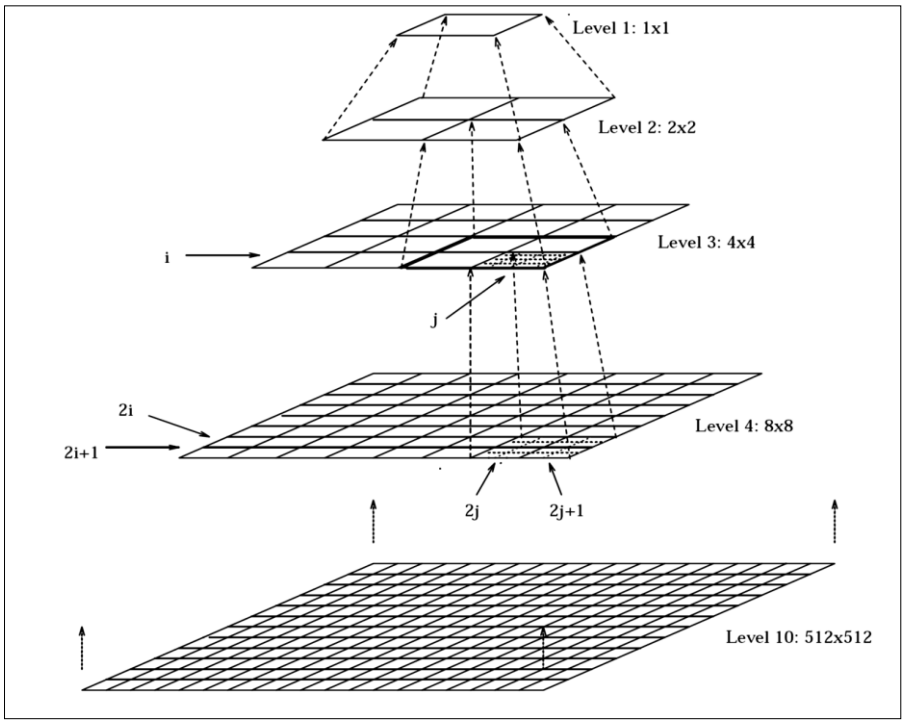
(ب) خط عمود بر محور اصلی, محور فرعی است.

(ج) برون مرکزی به وسیله‌ی نسبت محور اصلی و فرعی مستطیل دربردارنده‌ی شی به دست می‌آید.

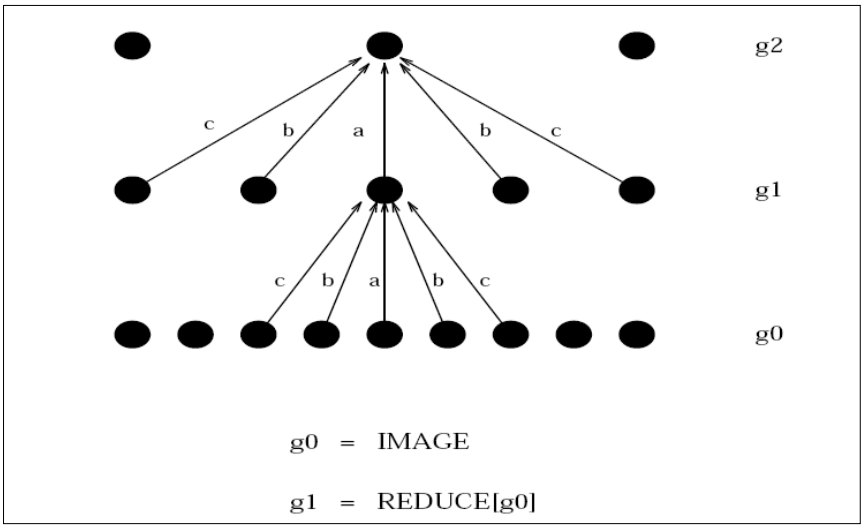
۲. ساختار شبکه‌ای را قرار بده طوری که برون مرکزی شبکه معادل برون مرکزی مستطیل دربردارنده‌ی شی باشد.

۳. نمره‌ی شکل را بیاب

شکل ۴.۱۱: الگوریتم برای محاسبه‌ی نمره‌ی شکل یک شکل دلخواه



شکل ۴.۱۲ هرم



شکل ۴.۱۳ نمایش یک بعدی فرآیندی که یک هرم گوسی تولید می‌کند

۱. ماسک باید متقارن باشد, یعنی

$$\widehat{w}(i) = \widehat{w}(-i)\ for\ i = 0,1,2 \qquad (۴.۲۴)$$

۲. مجموع اعداد ماسک باید ۱ باشد. فرض کنید

$$a + 2b + 2c = 1 \qquad (۴.۲۵)$$

۳. همه‌ی نقاط در یک سطح مشخص باید مجموع وزن دار مشابهی را به نقاط سطح بعد منتقل کنند

$$a + 2c = 2b \qquad (۴.۲۶)$$

با استفاده از سه محدودیت بالا، داریم:

$$\widehat{w}(0) = a \qquad (۴.۲۷)$$

$$\widehat{w}(-1) = \widehat{w}(1) = \frac{1}{4} \qquad (۴.۲۸)$$

$$\widehat{w}(-2) = \widehat{w}(2) = \frac{1}{4} - \frac{a}{2} \qquad (۴.۲۹)$$

عملگر EXPAND, اساسا یک تصویر با سایز $M + 1$ را به یک تصویر با سایز $2M + 1$ تبدیل می‌کند.

$$g_{l,n} = EXPAND[g_{l,n-1}] \qquad (۴.۳۰)$$

که با استفاده از فرمول شبیه ۴.۲۱ به صورت زیر حساب شود:

$$g_{l,n}(i,j) = \sum_{p=-2}^{p=2} \sum_{q=-2}^{q=2} w(p,q).\, g_{l,n-1}\left(\frac{i-p}{2}, \frac{j-q}{2}\right) \qquad (۴.۳۱)$$

که $g_{l,x}$ تصویر در سطح L است که با عملگر EXPAND کردن تصویر سطح g_l به تعداد n بار به دست می‌آید. در اینجا فقط نقاطی $\frac{i-n}{2}$ و $\frac{i-m}{2}$ برای آن‌ها عدد صحیحی می‌شوند در نظر گرفته می‌شوند. یک بار $(n = 1)$ گسترش تصویر در سطح l به این صورت به دست می‌آید:

$$g_{l,1} = \sum_{p=-2}^{p=2} \sum_{q=-2}^{q=2} w(p,q).\, g_l(\frac{i-p}{2}, \frac{j-q}{2}) \qquad (۴.۳۲)$$

هرم گوسی به تصویر " تاکسی‌هامبورگ" در شکل ۴.۱۴ نشان داده شده است.

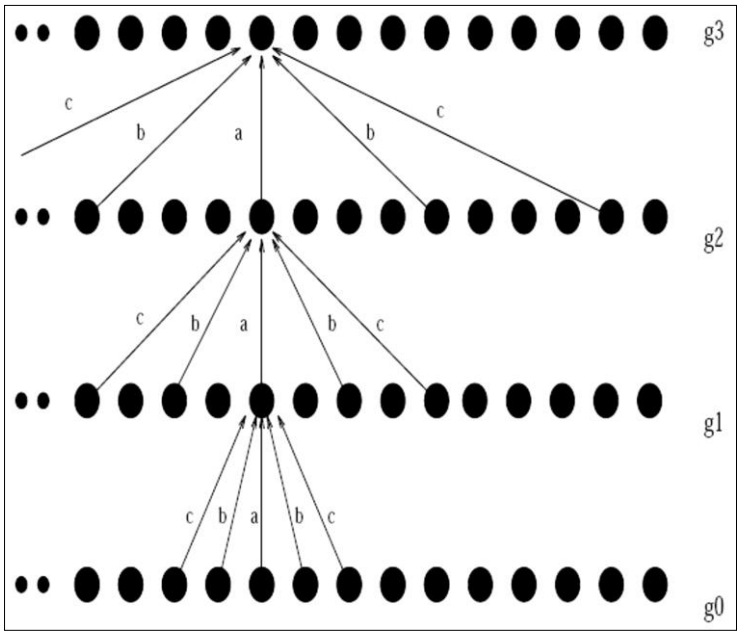
هرم لاپلاسین,L , با محاسبه‌ی تفاضل بین تصاویر دو سطح پشت سر هم به دست می‌آید:

$$L_1 = g_l - EXPAND[g_{l+1}] \qquad (۴.۳۳)$$

هر نقطه در تصویر L1 مثل اعمال دو تابع گوسی به تصویر اصلی است. تفاضل گوسی معادل عملگر LOG (لاپلاسین گوسی) است که معمولا برای تشخیص لبه استفاده می‌شود.

همبستگی گسسته سلسله مراتبی ,(Hierarchical Discrete Correlation) HDC مشابه عملگر REDVCE است, به جز اینکه چگالی گره‌ها در هر سطح ثابت می‌ماند. این به صورت گرافیکی در شکل ۴.۱۵ نشان داده شده است.

شکل۴.۱۴ هرم گوسی برای تصویر تاکسی با d=0.4



شکل ۴.۱۵ یک نمایش گرافیکی از فرآیند HDC

۱. $l = \log\left(\frac{n}{m}\right)$

۲. پنجره‌ای از هرم مرجع به مرکزیت ویژگی از سطح l، $RefWindow = m \times m$

۳. پنجره‌ای از ورودی در سطح (l)، $InputWindow = 2m \times 2m$

۴. بهترین تطابق از پنجره‌ی $RefWindow$ را در $InputWindow$ پیدا کن و آنرا (X_{best}, Y_{best}) قرار بده

۵. اگر $l = 0$ توقف کن وگرنه $l = l - 1$

۶. پنجره‌ای از هرم مرجع به مرکزیت ویژگی از سطح l جدا کن، $RefWindow = m \times m$

۷. پنجره‌ای از ورودی به مرکزیت نقطه‌ی متناظر با (X_{best}, Y_{best}) در سطح (l) انتخاب کن، $InputWindow = 2m \times 2m$

۸. به گام ۴ برو

۴.۱۶ الگوریتم برای گرفتن همبستگی با استفاده از هرم

۴.۵.۲ همبستگی‌گیری با استفاده از هرم‌ها

هرم‌ها را می‌توان برای چندین هدف استفاده کرد، برای مثال، می‌توان از آن‌ها برای محاسبه‌ی کارآمد همبستگی استفاده کرد. همبستگی برای تطبیق یک قالب استفاده می‌شود.طوری که یک الگوی فرضی از

یک پنجره‌ی سطح خاکستری روی کل تصویر جاروب می‌شود تا بهترین بخش مطابق پنجره پیدا شود. البته این کار بدون هرم هم قابل انجام است. فرض کنید می‌خواهیم مطابق یک پنجره‌ی ۱۶×۱۶ را در یک تصویر ۲۵۶×۲۵۶ بیابیم. برای این به (۵۸۰۸۱=(۲۵۶-۱۵)×(۲۵۶-۱۵)) بار آزمایش نیاز داریم. با این وجود می‌توان نشان داد که با استفاده از هرم تعداد آزمایش‌ها به ۱۱۵۵۸۰۸۱=^{۲}(۱۶+۱)×۴ بار آزمایش نیاز است که بهبود قابل ملاحظه‌ای در هزینه‌ی محاسبات است.

الگوریتم برای همبستگی با استفاده از هرم در ۴.۱۶ ارائه شده است. در این الگوریتم, ابتدا پنجره‌ی $m \times m$ از بالاترین سطح (l) هرم برای یافتن بهترین تطابق در پنجره‌ی $2m \times 2m$ در همین سطح از هرم مربوط به تصویر ورودی استفاده می‌شود. فرض کنید که بهترین تطابق در (X_{best}, Y_{best}) وجود دارد. اکنون, پنجره‌ی $m \times m$ از سطح بعدی $(l-1)$ هرم تصویر مرجع برای یافتن بهترین تطابق در پنجره‌ی $2m \times 2m$ هرم ورودی به مرکزیت (X_{best}, Y_{best}) در همان سطح استفاده می‌شود. این فرآیند تا رسیدن به بهترین تطابق در پایین تر سطح هرم ادامه پیدا می‌کند.

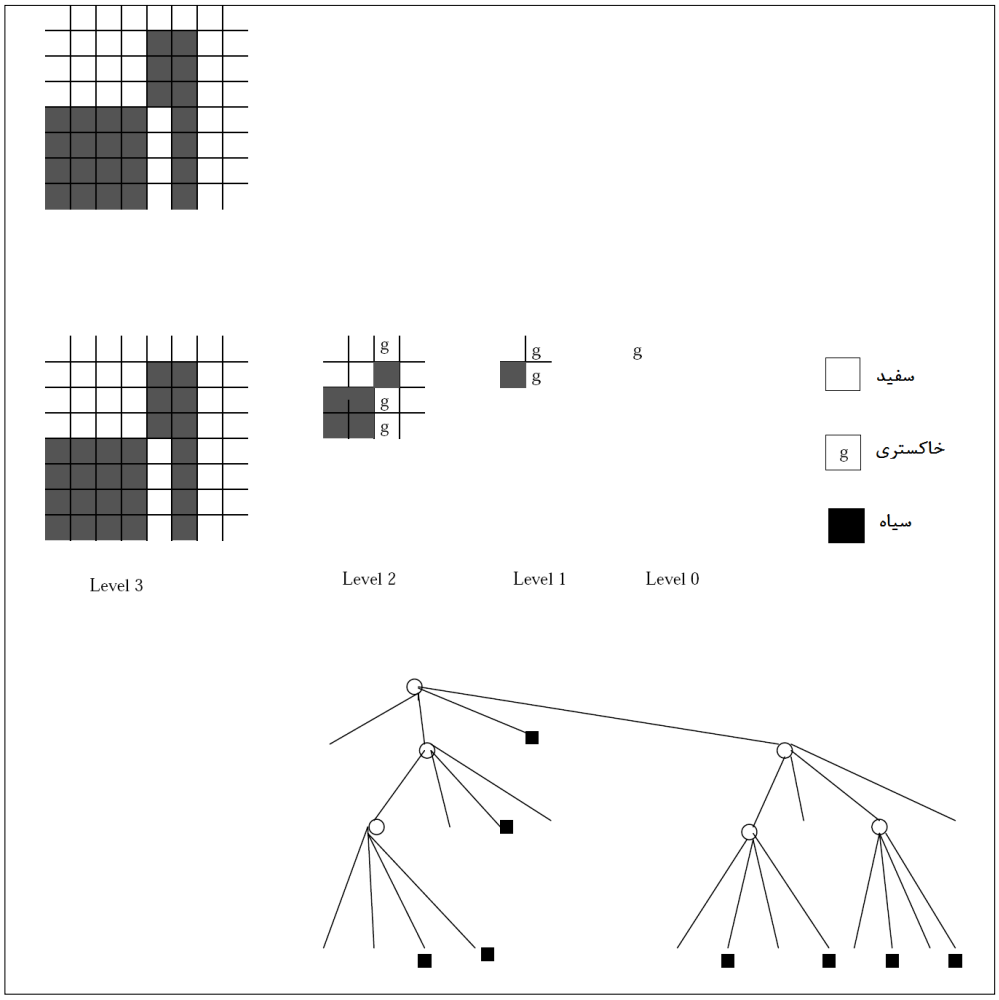
۴.۶ درخت‌های چهارگانه

درخت‌های چهارگانه, ساختار داده‌های مفیدی برای نمایش یک ناحیه هستند. در هر گره‌ی درخت چهارگانه حداکثر چهار زیر شاخه می‌توان قرار داد. سه نوع زیر شاخه وجود دارد: **سیاه, سفید و خاکستری**.

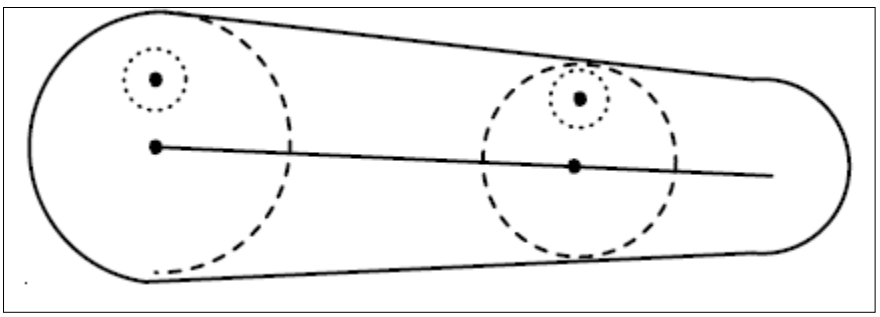
گره‌ی سیاه متناظر تک ناحیه‌ای است که به صورت یکنواخت سیاه است و گره‌ی سفید متناظر ناحیه‌ای است که به صورت یکنواخت سفید است. گره خاکستری هم متناظر ناحیه‌ای است که ترکیبی از پیکسل‌های سیاه و سفید است. تولید یک درخت چهارگانه از هرم تصویر آسان است. الگوریتم تولید درخت چهارگانه در شکل b.۴.۱۷ آمده است. سه سطح از هرم تصویر و درخت چهارگانه‌ی متناظرش در شکل ۴.۱۸ نشان داده شده اند.

۱. اگر هرم سیاه و سفید است بازگرد و گرنه
(الف) به صورت بازگشتی درخت چهارگانه‌ی یک چهارم جنوب شرقی را بیاب.
(ب) به صورت بازگشتی درخت چهارگانه‌ی یک چهارم جنوب غربی را بیاب.
(ث) به صورت بازگشتی درخت چهارگانه‌ی یک چهارم شمال شرقی را بیاب.
(ت)به صورت بازگشتی درخت چهارگانه‌ی یک چهارم شمال غربی را بیاب.
۲. بازگرد

۴.۱۷: الگوریتم برای محاسبه‌ی درخت چهارگانه با هرم



شکل ۴.۱۸ تولید درخت چهارگانه



شکل ۴.۱۹: تبدیل محور میانی نقاط روی محور میانی مرکز حداکثر ناحیه‌ی همسایگی دایروی هستند که کاملا در شکل محصور شده اند توجه کنید که مرکز دایره‌های کوچکتر روی محور نیستند.

۱. به صورت تکرارشوندخ f^k را به طریق زیر محاسبه کنید:

$$f^k(x,y) = f^0(x,y) + \min\left(f^{k-1}(p,q)\right) \quad (۴.۳۴)$$

$\forall(p,q)$ طوری که $distance\big((x,y),(p,q)\big) \le 1$ شود.

۲. محور میانی را تمام نقاطی تشکیل می‌دهند که:

$$f^k(x,y) \ge f^k(p,q) \quad (۴.۳۵)$$

$\forall(p,q)$ طوری که $distance\big((x,y),(p,q)\big) \le 1$ شود.

۴.۲۰ الگوریتم تکرار شونده برای محاسبه تبدیل محور میانی

۴.۷ تبدیل محور میانی

تبدیل محور میانی برای نمایش شکل نواحی استفاده می‌شود. اساسا این تبدیل یک اسکلت (skeleton) ناحیه را می‌دهد. یک راه برای تعریف محور میانی به این صورت است: نقاط روی محور میانی مراکز نواحی همسایگی حداکثری دایره‌ای هستند که کاملا داخل شکل جای می‌گیرند. (شکل ۴.۱۹ را ببینید). یک الگوریتم بازگشتی برای تبدیل محور میانی در شکل ۴.۲۰ داده شده است. در این الگوریتم,فرض شده تصویر باینری $f^0(x,y)$ که شامل شکل ناحیه است موجود می‌باشد. f^0 برای محاسبه‌ی تکرار شوند استفاده می‌شود. گام اول الگوریتم وقتی f^{k-1} و f^k یکسان می‌شوند پایان می‌پذیرد. در گام دوم نقاطی که در f^k ماکسیمم محلی هستند به عنوان محور میانی در نظر گرفته می‌شوند.

تبدیل محور میانی ناحیه‌ی نشان داده شده در شکل ۴.۲۲.۲ در شکل۴.۲۲c نشان داده شده است. گام‌های مختلفی که نیاز است برای محاسبه‌ی محور میانی آن برداشته شوند در شکل۴.۲۲ c و b نشان داده شده‌اند. نقاط تشکیل دهنده محور میانی و به صورت پررنگ در شکل ۴.۲۲.c نشان داده شده است.

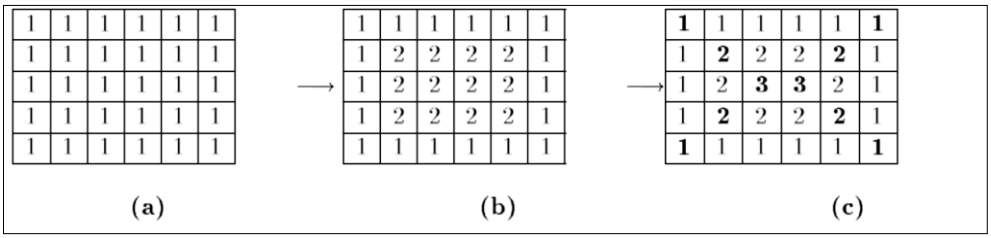
یک الگوریتم تکرار شونده برای محاسبه‌ی شکل از محور میانی در شکل ۴.۲۱ نشان داده شده است. اگر عکس تبدیل محور میانی به کار بسته شود تصویر اصلی را می‌توان بازیابی کرد (همانند شکل ۴.۲۲.c)، گام‌های مختلف دخیل در این فرایند در شکل ۴.۲۳ نشان داده شده اند.

۱. به صورت تکرار شونده g^k را به طریق زیر محاسبه‌ی کنید:

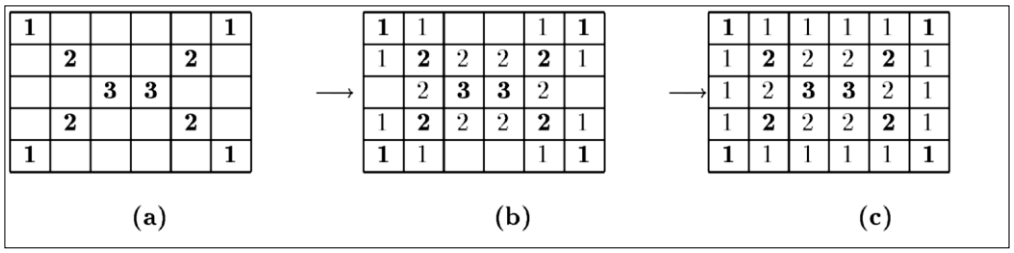
$$g^k(x,y) = \begin{cases} \max[0, (\max_{g^{k-1}} g^{k-1}(p,q)) - 1 \end{cases} \quad (۴.۳۶)$$

$\forall(p,q)$ طوری که $distance\big((x,y),(p,q)\big) \le 1$ شود.

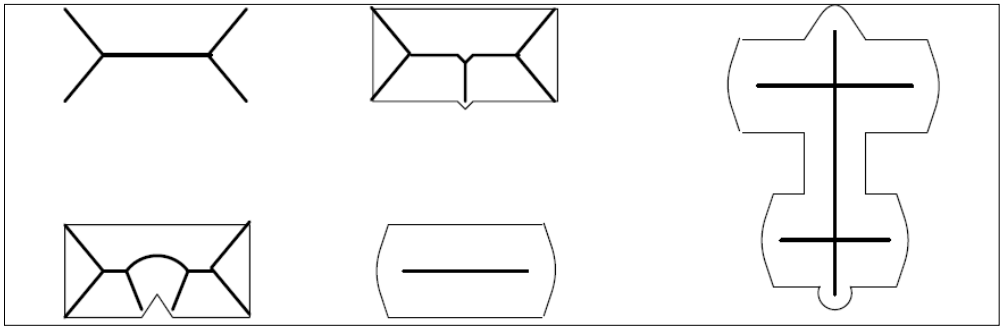
شکل۴.۲۱: الگوریتم تکرارشونده برای عکس تبدیل محور میانی



شکل۴.۲۲: تبدیل محور میانی، (a) شکل مستطیلی $f^0(x,y)$. (پیکسل‌های پس زمینه که نمایش داده نشده اند همگی "0" هستند.(b) گام‌های میانی $f^2(x,y)f^1(x,y)$)). نقاط روی محور میانی به صورت پر رنگ نمایش داده شده اند.



شکل۴.۲۳: عکس تبدیل محور میانی (الف) محور میانی $g^0(x,y)$. (ب) گام‌های وسطی ($g^1(x,y)$). (ج) شکل اصلی $g^2(x,y)$



شکل۴.۲۴: برخی مثال‌ها از تبدیل محور میانی

۴.۸ عملگر نقاط ویژه مارکو

عملگر مارکو یک یابنده‌ی بسیار مفید نقاط ویژگی است و به صورت گسترده‌ای در حرکت و استریو استفاده می‌شود. این عملگر نقاطی را (در حقیقت پنجره‌های کوچکی را) در تصویر می‌یابد که در تصاویر سطوح خاکستری مورد علاقه‌ی ما هستند. ابتدا، برای هر پنجره همپوشان ۴×۴، چهار واریانس در سطح خاکستری (افقی، عمودی و قطری‌ها) مشخص می‌شود که با v_h،v_v، v_d،v_2 نشان داده می‌شوند. (شکل ۴.۲۵۲-d):

$$V_h(x,y) = \sum_{j=0}^{3}\sum_{i=0}^{2}\big(P(x+i,y+j) - P(x+i+1,y+j)\big)^2 \qquad (۴.۳۷)$$

$$V_v(x,y) = \sum_{j=0}^{2}\sum_{i=0}^{3}\big(P(x+i,y+j) - P(x+i,y+j+1)\big)^2 \quad (۴.۳۸)$$

$$V_d(x,y) = \sum_{j=0}^{2}\sum_{i=0}^{2}\big(P(x+i,y+j) - P(x+i+1,y+j+1)\big)^2 \quad (۴.۳۹)$$

$$V_a(x,y) = \sum_{j=0}^{2}\sum_{i=0}^{3}\big(P(x+i,y+j) - P(x+i-1,y+j+1)\big)^2 \quad (۴.۴۰)$$

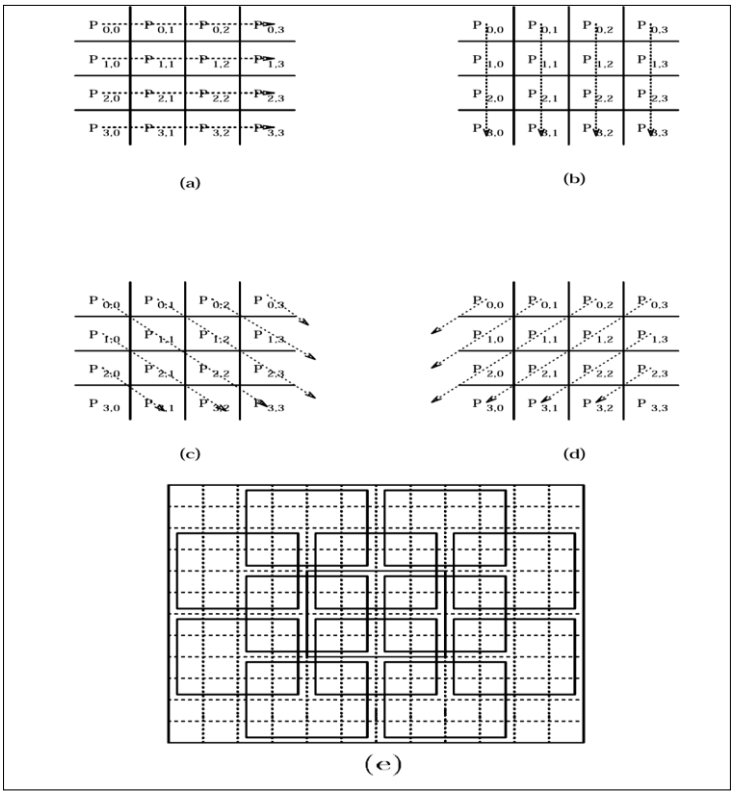
سپس آرایه‌ی $v(x,y)$ به صورت زیر محاسبه می‌شود:

$$V(x,y) = min(v_h(x,y), v_v(x,y), v_d(x,y), v_a(x,y)) \quad (۴.۴۱)$$

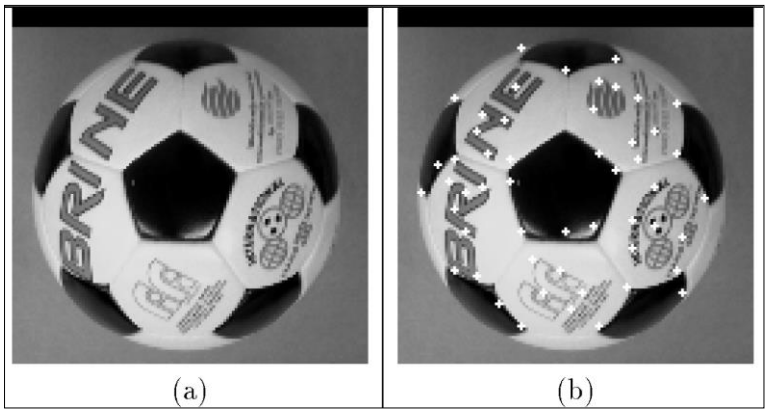
یک پنجره‌ی ۴×۴ به مرکزیت پیکسل (x,y) به عنوان ویژه اعلان می‌شود اگر که در همسایگی ۱۲×۱۲ آن پیکسل ماکسیمم محلی باشد(شکل c.۴.۲۵ را ببینید). آرایه‌ی $I(x,y)$ تمام نقاط ویژه در تصویر را ذخیره می‌کند:

$$I(x,y) = \begin{cases} 1 & if\ V \geq V(p,q), \forall (p,q) \in N(x,y) \\ 0 & otherwise \end{cases} \quad (۴.۴۲)$$

نتایج عملگرد مارکو در تصویر ۴.۲۶ نمایش داده شده است.



تصویر ۴.۲۵: عملگرد مارکو. (a) واریانس سطح خاکستری در جهت افقی. (b) واریانس سطح خاکستری در جهت عمودی. (c) واریانس در جهت قطری. (d) واریانس در جهت ضد قطری. (e) ۲۵ پنجره‌ی همپوشان در یک همسایگی ۱۲×۱۲.



تصویر۴.۲۶ نتایج برای عملگر مارکو. (a) تصویر سطح خاکستری. (b) نقاط ویژه‌ی مارکو(در کل ۲۵ نقطه) که بر روی تصویر (c) نمایش داده شده‌اند.

تمرینات

۱. طرحی برای تشخیص سهمی‌ها $(y - y_0 = d(x - x_0)^2)$ به مرکزیت (x_0, y_0) با استفاده از تبدیل‌هاف ابداع کنید. از اطلاعات گرادیان برای کاهش محاسبات استفاده کنید. پیچیدگی محاسباتی الگوریتم شما چقدر است؟

۲. طرحی برای تشخیص بیضی‌هایی که می‌دانیم محور اصلی شان موازی محور xها است. با استفاده از تبدیل‌هاف ابداع کنید. از اطلاعات گرادیان برای کاهش محاسبات استفاده کنید. پیچیدگی محاسباتی الگوریتم شما چقدر است؟

۳. اگر تبدیل‌هاف تعمیم یافته برای تشخیص دایره استفاده شود, جدول R- چگونه به نظر خواهد رسید؟

۴. چگونه می‌توان یک کمان را از یک دایره‌ی کامل با استفاده از تبدیل‌هاف تعمیم یافته تشخیص داد؟

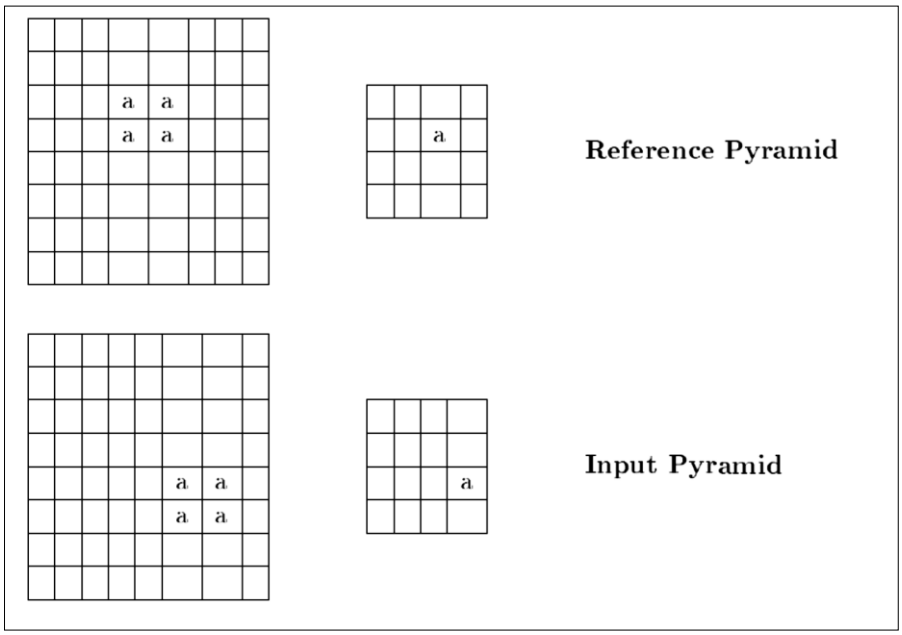
۵. شکل یک شی دلخواه را در تصویر در نظر بگیرید. زاویه‌ی گرادیان در هر پیکسل به صورت آرایه‌ی زیر داده شده است. مرکز شی با * نشان داده شده است.

90	45			90	90	90
180		90	135			0
	225					0
	180		*			0
	180					0
135						0
180				225	270	270
180	270	270	270			

(الف) برای استفاده در تبدیل‌هاف تعمیم یافته یک جدول R- برای این شی آماده کنید.

(ب)فرض کنید تصویری به شما داده شده است که اندازه‌ی گرادیان آن به صورت جدول بالا است. گام‌های مختلف تبدیل‌هاف تعمیم یافته را اجرا کنید, محل شی را با استفاده از جدولی که در بخش (الف) ساختید تعیین کنید.

۶. این مسئله به شما تجربیات یدر زمینه‌ی تطابق یابی با همبستگی با استفاده از هرم‌ها خواهد داد. در زیر دو هر موجود دارد :هرم مرجع و هرم ورودی .برای سادگی، این هرم‌ها دوسطحی درنظر گرفته شده‌اند. هر سطح از هرم با برداشتن سطر یا ستون (با شماره دیگر) از سطح قبلی بدست می‌آید .یک الگوی 2×2 متشکل از d ها باید در تصویر ورودی تشخیص داده شود .فرض کنید مرکز پنجره با اندازه‌ی زوج پیکسل واقع شده در گوشه‌ی بالایی سمت چپ می‌باشد .همچنین فرض کنید هربلوک 2×2 در هرسطح از هرم به یک بلوک 1×1 درسطح بعدی نگاشت می‌شود .



(الف)چند عمل مقایسه نیاز است تا مکان الگویی را در تصویر ورودی بدون استفاده از هرم مشخص شود؟

(ب)گام‌های مختلف در تطابق همبستگی با استفاده از هرم را اجرا کنید.

(ج)چند عمل مقایسه نیاز است تا مکان الگویی را در تصویر ورودی با استفاده از هرم مشخص شود؟

(د)اگر الگو در تصویرمرجع به مکان دلخواه دیگری جابجا شود آیا هنوز قادر به مکان‌یابی آن با برداشتن گام‌های(ب) خواهید بود؟

(ر)اگر الگو در تصویر ورودی به مکان دلخواه دیگری جابجا شود، آیا هنوز قادر به مکان یابی آن بابرداشتن گام‌ها(ب) خواهید بود؟

(س)با درنظرگرفتن اندازه‌ی $m \times m$ و $n \times n$ به ترتیب برای الگو و تصویر که m و n هر دو زوج هستند، آیا می‌توانید نتایج (الف) را تعمیم دهید؟

(ش)با در نظر گرفتن اندازه‌ی $m \times m$و $n \times n$ برای به ترتیب الگو وتصویرآیا میتوانید نتایج (ج) را تعمیم دهید؟

۷. یک الگوریتم بازگشتی برای محاسبه‌ی مساحت یک ناحیه با استفاده از درخت چهارگانه تهیه کنید.

۸. محور میانی شکل زیر را با استفاده از همسایگی به اندازه‌ی 8 تعیین کنید.

لطفا همه‌ی گامهای مورد نیاز را نشان دهید.

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

۹. با استفاده از تبدیل محور میانی که در مسئله‌ی قبل حساب شد شکل اصلی را با اعمال عکس تبدیل محور میانی بازیابی کنید.

۱۰. فرض کنید یک تصویر کد زنجیره‌ای زیر را داراست (۸-اتصال) نمره‌ی شکل آن را حساب کنید.

۰۷۶۶۶۶۵۵۳۳۲۱۲۱

فصل ۵

حرکت

۵.۱ مقدمه

تاکنون ما با یک تصویر ایستا که از یک دوربین ثابت گرفته شده است سروکار داشتیم. در این فصل، ما با یک توالی از تصاویر که در بازه‌های زمانی متفاوت گرفته شده اند سروکار خواهیم داشت. حرکت اشیا فضای سه بعدی روی صفحه‌ی تصویر دو بعدی یک حرکت دو بعدی ایجاد می‌کند که این حرکت پیکسل‌ها در سطح دو بعدی شار نوری نامیده می‌شود. در این فصل چندین روش برای محاسبه شار نوری توضیح خواهیم داد. شار نوری می‌تواند برای محاسبه حرکت سه بعدی، تبدیل شکل سه بعدی استفاده شود. ما همچنین یک متد کلی برای محاسبه سه بعدی از شار نوری را توصیف خواهیم کرد.

۵.۲ شار نوری

۵.۲.۲. متد Horn و Schunck

تابع $f(x,y,z)$ که در آن x,y مختصات مکانی و t واحد زمان است برای تعیین توالی تصویر به کار می‌رود. بعد، $f(x1,y1,z1)$ سطح خاکستری در مختصات $x1,y1$ در زمان $t1$ را نشان می‌دهد. فرض کنید که با یک تغییر کوچک dx,dy,dt در x,y,z هیچ تغییری در سطوح خاکستری بوجود نیاید که یعنی:

$$f(x,y,t) = f(x+dx, y+dy, t+dt)$$

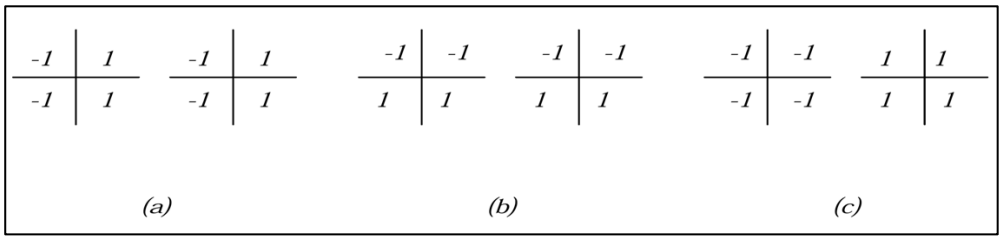
با پیدا کردن بسط سری تیلور حول x,y,z با محاسبه دست سمت راست ما می‌توانیم تابع زیر را بدست آوریم:

$$f(x,y,t) = f(x,y,t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt$$

معادله بالایی می‌تواند به صورت زیر ساده شود:

$$f_x dx + f_y dy + f_t dt = 0$$

که $f_y = \frac{\partial f}{\partial y}$، $f_x = \frac{\partial f}{\partial x}$ و $f_t = \frac{\partial f}{\partial t}$ این مشتقات می‌توانند با بکارگیری هم پوشانی‌ها تصویر ۵.۱ را به تصویر متوالی $f(x, y, z)$ نشان داد.



شکل ۵.۱ هم پوشانی‌ها برای محاسبه مشتقات مکانی و زمانی. توجه کنید که مرکز هم پوشانی در پیکسل سمت راست پایینی می‌باشد.

(a) هم پوشانی برای f_x

(b) هم پوشانی برای

(c) هم پوشانی برای

با تقسیم هر یک از عبارات بالایی بر dt خواهیم داشت:

$$f_x u + f_y v + f_t = 0$$

که در آن $u = \frac{dx}{dt}$ و $v = \frac{dy}{dt}$ می‌باشد که نشان‌دهنده شار نوری می‌باشند. دو عبارت نامعلوم u, v در معادله بالایی وجود دارند که براحتی نمی‌توان این معادله را حل کرد. معادله بالایی می‌تواند بصورت زیر نوشته شود.

$$v = -\frac{f_x}{f_y} u - \frac{f_t}{f_y}$$

این معادله خط عمود بر فضای $u - v$ می‌باشد. چندین راه حل ممکن برای این معادله وجود دارد. راه حل‌های می‌توانند در هر کجای روی خطی که در شکل ۵.۲ نشان داده شده است قرار گیرند. فرض کنید (u, v) راه حل صحیح می‌باشد. این بردار می‌تواند به دو مولفه تقسیم شود. یکی در امتداد خط که با p نشان داده می‌شود و دیگری عمود به خط که با d نشان داده می‌شود و آنرا می‌توان با استفاده از رابطه مثلثاتی براحتی نشان داد که $d = \frac{f_t}{\sqrt{f_x^2 + f_y^2}}$.

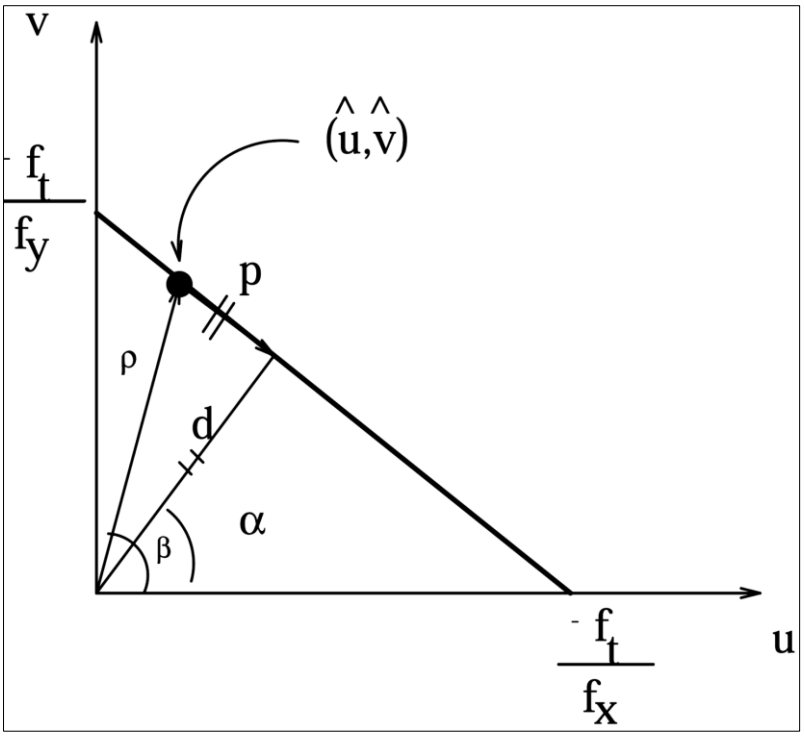
بنابراین با داشتن مشتقات f_x, f_y, f_t ما فقط می‌توانیم مولفه نرمال d و شارنوری را محاسبه کنیم. به هر حال، مولفه موازی P می‌تواند مستقیما از مشتقات محسبه گردد. یکی از اولین رویکردهای برای محاسبه شارنوری توسط Horn و Schunck پیشنهاد شد. این دو برای تخمین زدن (u, v) از تابع E که بصورت زیر کمینه شده است را پیشنهاد دادند:

$$E(x,y) = (f_x u + f_y v + f_t)^2 + \lambda(u_x^2 + u_y^2 + v_x^2 + v_y^2)$$

که عبارت اول شار نوری است که مستقیما از معادله 5.4 بدست می‌آید. عبارت دوم با همواری شارنوری ارتباط دارد. برای شارنوری صحیح عبارت اول بایستی نزدیک به صفر باشد، یا اینکه مربع عبارت دومی‌بایستی کوچک باشد. از آنجایی که حرکت بیشتر اشیا به صورت هموار می‌باشد، عبارت دوم به همواری مستقیم تاکید دارد. با دیفرانسل E نسبت به U و V و قرار دادن معادله به صفر ما می‌توانیم:

$$\frac{\partial E}{\partial u} = (f_x u + f_y v + f_t) f_x + \lambda(u_{xx} + u_{yy}) = 0 \qquad (۵.۷)$$

$$\frac{\partial E}{\partial v} = (f_x u + f_y v + f_t) f_x + \lambda(v_{xx} + v_{yy}) = 0 \qquad (۵.۸)$$



شکل ۵.۲: شار نوری خط را در فضای $u-v$ محدود می‌کند. d طول عمود بر خط از مبدا مختصات می‌باشد. aزاویه عمود با محور x را ایجاد می‌کند.$(\hat{u},\hat{v})$ یک راه حل ممکن می‌باشد. محور شارنوری$(\hat{u},\hat{v})$ می‌تواند به دو مولفه تقسیم شود. P که در امتداد خط محدود و d عمود عمود برخط محدود.

با جاگزینی و $\Delta u^2 = u_{xx} + u_{yy}$ و $\Delta v^2 = v_{xx} + v_{yy}$ ما می‌توانیم معادلات زیر را بدست آوریم:

$$(f_x u + f_y v + f_t) f_x + \lambda(\Delta^2 u) = 0 \qquad (۵.۹)$$

$$(f_x u + f_y v + f_t) f_y + \lambda(\Delta^2 v) = 0 \qquad (۵.۱۰)$$

فرض کنید $\Delta^2 u = u - u_{av}$ که u_{av} میانگین مولفه U شار نوری که از ۴ همسایه یک پیکسل بدست آمده است. بطور مشابه $\Delta^2 v = v - v_{av}$ پس:

$$(f_x u + f_y v + f_t) f_x + \lambda(u - u_{av}) = 0 \qquad (۵.۱۱)$$

$$(f_x u + f_y v + f_t) f_y + \lambda(v - v_{av}) = 0 \qquad (۵.۱۲)$$

این معادله می‌توانند بصورت زیر حل شوند.

$$u = u_{av} - f_x \frac{P}{D} \qquad (۵.۱۳)$$

$$v = v_{av} - f_y \frac{P}{D} \qquad (۵.۱۴)$$

که $P = f_x u_{av} + f_y v_{av} + f_t$ و $D = \lambda + f_x^2 + f_y^2$ تکرار الگوریتم برای محاسبه شار نوری با استفاده از معادلات بالایی از شکل ۵.۳ بدست می‌آیند. نتایج الگوریتم Horn و Schunck در شکل ۵.۴ نشان داده شده است.

۵.۲.۲ متد Schunck

Schunck یک متد ساده برای شار نوری پیشنهاد داده است. از آنجایی که سطح خاکستری در یک پیکسل فقط یک محدودیت می‌دهد، شار نوری می‌تواند در هر مکانی روی خط عمود که توسط مشتقات زمانی و مکانی قرار گیرد. اگر محدودیت دوم از همسایه پیکسلی استفاده شده باشد، پس شارنوری صحیح می‌تواند با محاسبه دو خط متقاطع که با محدودیت نشان داده می‌شود تعیین شود.

۱-

۲- مقدار دهی u^k و v^k

۳- تا زمانی که بعضی اندازه خطاها راضی کننده باشد: انجام بده.

$$u^k = u_{av}^{k-1} - f_x \frac{P}{D} \quad (۵.۱۵)$$

$$v^k = v_{av}^{k-1} - f_y \frac{P}{D} \quad (۵.۱۶)$$

۵.۳: الگوریتم Horn و Schunck برای شار نوری

در کل حالت مطلوب این است که از چندین محدودیت در نظر گرفته شود نه فقط دو محدودیت، Schunck از هشت محدودیت که از نقاط اطراف یک همسایگی به طول ۳×۳ حول یک پیکسل بدست آمده بود استفاده می‌کند. نتایج در هشت خط متقاطع می‌باشد. اگر اندازه گیری‌ها، نویزهای آزاد باشند، همه پیکسل‌ها در یک همسایگی به طول ۳×۳ به شی متحرک وابسته خواهند شد. پس در اصل همه هشت خط در یک نقطه همدیگر را قطع می‌کنند که نشان‌دهنده شار نوری صحیح می‌باشند. ولی به خاطر نویز اگر بعضی از نقاط روی مرز شی متحرک قرار گیرند، همه هشت خط ممکن است در همان نقطه همدیگر را قطع نکنند. تقاطع‌ها ممکن است بخش بندی شوند. به هر حال، تقاطع‌ها ممکن است حول بعضی از راه حل‌های صحیح دسته بندی شوند. بنابراین Schunck از سخترین دسته بندی تقاطع که شامل حداقل نیمی‌از تقاطعات برای نوری میباشد استفاده می‌کند. آسان تر این است که فرم مختصات قطعی معادله شار نوری (معاله ۵.۴) که بصورت زیر است استفاده می‌کنیم:

$$d = \rho \cos(\alpha - \beta) \quad (۵.۱۷)$$

که دراین معدله $d = \frac{f_t}{\sqrt{f_x^2 + f_y^2}}$ است و طول بصورت عمود از مبدا به خط محدود می‌باشد. و $a = \tan^{-1}\frac{f_y}{f_x}$ زاویه عمود بر محور x می‌سازد (شکل ۵.۲) و ρ نشاندهنده سرعت، β جهت بردار شار نوری می‌باشد. حال برای نقاط ۱و۲ در معادله شار نوری بصورت زیر بدست می‌آوریم:

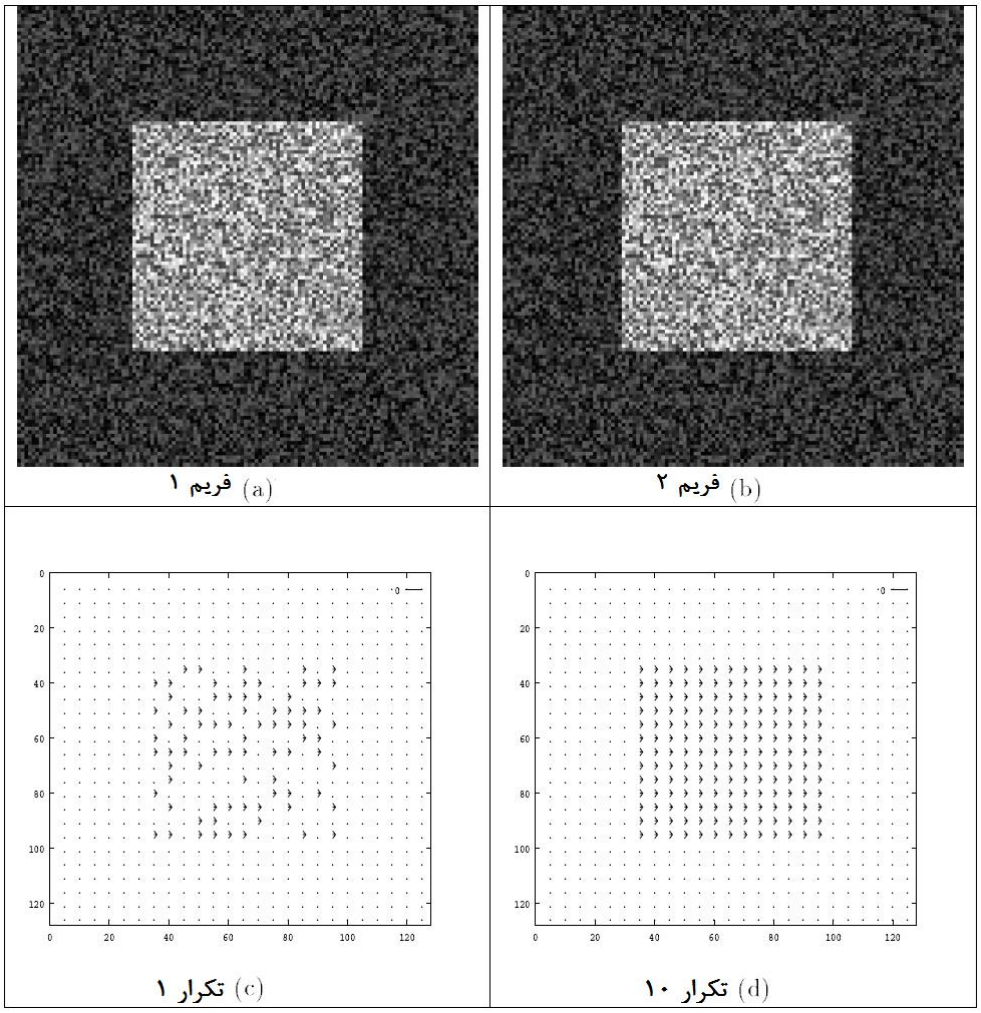
$$d = \rho \cos(\alpha_1 - \beta) \quad (۵.۱۸)$$

$$d = \rho \cos(\alpha_2 - \beta) \quad (۵.۱۹)$$

این دوخط دریک نقطه همدیگر را قطع می‌کنند که در یک فاصله b_2 (خط مقطع AB) از تقاطع خط عمود d_1 که از یک مبدا به خط محدود (در شکل ۵.۴ نشان داده شده است) بدست می‌آید. براحتی می‌توان نشان داد که b_2 بصورت زیر محاسبه می‌شود:

$$b_2 = \frac{d_2 - d_1 \cos(a_2 - a_2)}{\sin(a_2 - a_1)} \quad (۵.۲۰)$$

به همین ترتیب، تقاطع با خطوط که وابسته به نقاط ۳و۴و...۹ می‌باشند می‌توانند محاسبه شوند. این تقاطع‌ها احتملا حول یک نقطه خوشه شوند.

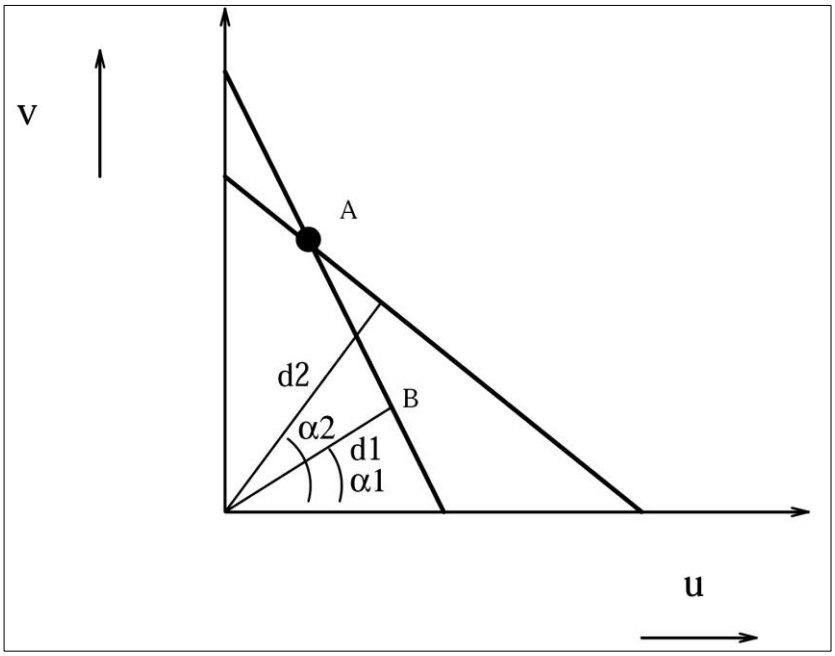


شکل ۵.۴: نتایج برای الگوریتم هورن و شانگ برای جایگزینی یک پیکسل و $\lambda = 4$

اگر مرکز موقعیت دسته بندی $\hat{b}$ باشد، پس شار نوری بصورت زیر محاسبه می‌شود:

$$\hat{\rho} = \sqrt{d_1^2 + \hat{b}_2} \qquad (۵.۲۱)$$

$$\hat{\beta} = \alpha_1 + \tan^{-1}(\frac{\hat{b}}{d_1}) \qquad (۵.۲۲)$$



شکل ۵.۵: تقاطع دو شار نوری خطوط محدود شده

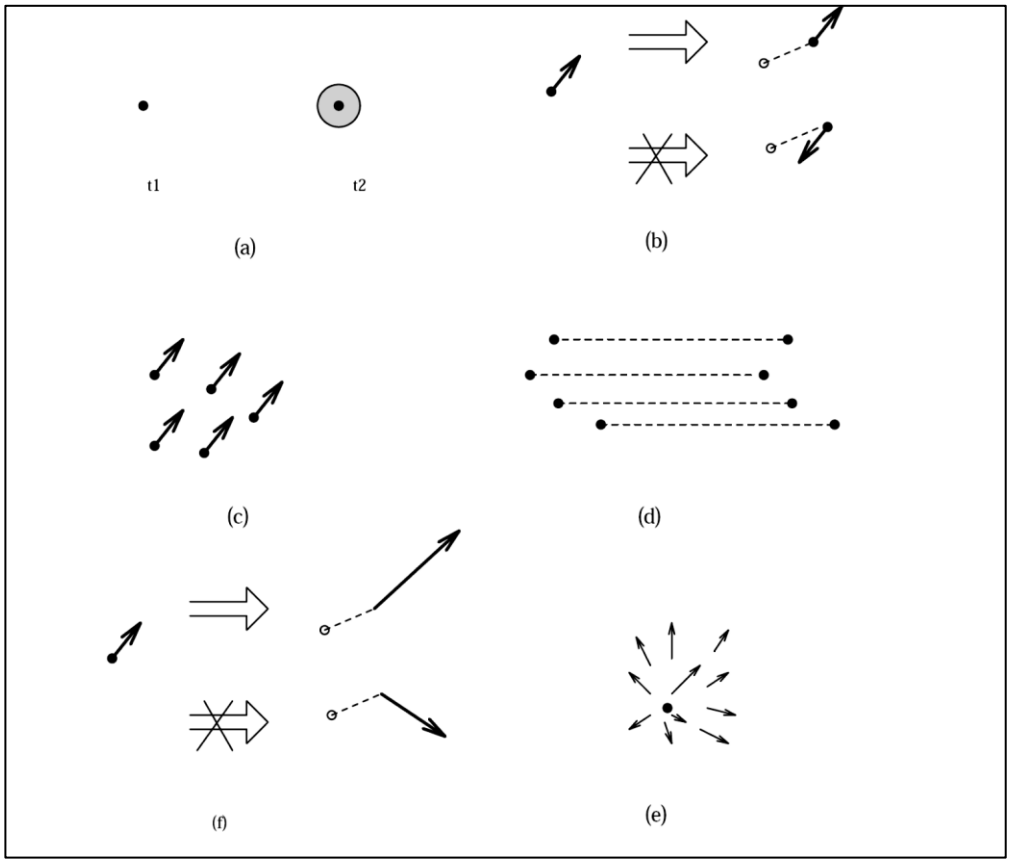
۵.۳ شار نوری مبتنی بر توکن

دو فریم در نمونه‌های زمانی متفاوت و m نقطه در هر فریم را در نظر بگیرید. مسئله با یک نگاشت نقطه‌ای در یک فریم به نقطه دیگر در فریم بعدی سروکار خواهد داشت که هیچ دونقطه‌ای روی نقطه یکسان نگاشت نمی‌شود. مطابقت در بسیاری از الگوریتم‌های بینایی ماشین نیاز می‌باشد و رویکردهای بسیاری برای این مسئله وجودخواهد داشت.با اندکی فرم ساده سازی شده این مسئله با استریو مواجه می‌شود که از تطابق تصویر چپی در تصویر راستی برای محاسبه عدم تطابق گرفته شده است. عدم تطابق نسبی است به عمق اشیایی که توکن‌هایش از به دنیای سه بعدی گرفته شده است. مسئله در استریو ساده شده است زیرا تطابق‌ها ممکن فقط بتوانند با یک خط Epipolar رخ دهند. بر عکس استریو که فریم‌ها در مکان جدا گشته اند نه در زمان.

مسئله تطابق انفجار ترکیبی است. برای مثال با دو فریم و m نقطه از هر فریم تعداد نگاشت‌ها ممکن m خواهد شد. برای نمونه تعداد الگوهای ممکن برای $m = 5$ نتیجه به صورت $5! = 120$ خواهد شد. در اینجا دو نکته قابل توجه وجود دارد. نخست، سعی براین باشد که همه نگاشت‌های ممکن تقریبا

غیرممکن باشند حتی تعدای فریم و نقاط معمولی. دوم، حتی اگر همه خط سیرهای ممکن را بدانیم چگونه می‌توانیم مجموعه صحیح را تعیین کنیم؟

به منظور کنار آمدن با پیچیدگی این مسئله، محققان تعدای از محدودیت‌ها را استفاده میکنند. این محدودیتها شامل بیشترین حرکت، کمترین حرکت یا همواری حرکت، حرکت معمولی، تطابق ادامه دار،سختی و غیره می‌باشد. بیشترین حرکت محدودیت به این اشاره می‌کند که اگر محدودیت روی سرعت به عنوان یک علت به معلول شناخته شود، با دادن موقعیتی نقطه‌ای از یک فریم می‌توان جستجویی را برای یک تطابق ممکن را در فریم بعدی به یک همسایگی کوچک در فریم فعلی محدود کرد.



شکل ۵.۶: محدودیت استفاده شده در تطابق حرکت.

a) بیشترین حرکت

b) تغیر حرکت کمتر

c) حرکت معمولی

d) مدل

e) محدودیت یکنواختی مبدایی

حرکت کم فرض‌هایی اکتشافی را که جهت و سرعت حرکت را نمی‌توانند با مقدار زیاد تغییر دهند را تغییر می‌دهد. بنابراین می‌توان بعضی تطابق‌های غلط را افزایش داد. این محدودیت اساسا منجر به حرکت هموار می‌شود. حرکت معمولی حرکت نقاط در یک همسایگی کوچک که همانند باشند را محدود میکند و تطابق سازگار یک تطابق سازگار یک تطابق برای یک نقطه تحمیل می‌کند.

سختی و همواری فرضیات حرکت بیشترین توجه را به خود اختصاص داده است. سختی به این اشاره می‌کند که اشیا در دنیای سه بعدی سخت می‌باشد. بنابراین فاصله اقلیدیسی بین هر دو نقطه روی شی سخت در فریم بعدی بدون تغییر می‌ماند. در واقع، اولمن سعی کرد نشان دهد که همواری از سختی نشات می‌گیرد و این را نشا ن می‌دهد که اگر اشیا سخت در معرض حرکت قرار گیرند هموار خواهند گشت ولی ضرورتاً عکس آن صادق نیست.

یک مسئله مهم در مسئله تطابق تبدیل اکتشاف کیفی بالاتر به عبارات کمی‌می‌باشد که توابع هزینه خواهند شد. سپس هدف یافتن نگاشتی است که یکی از توابع را کمینه کند. همه مجموعه‌های ممکن و انتخاب یکی با کمترین هزینه غیرممکن خواهند بود. بنابرین استفاده از یک الگوریتم تخمین مناسب برای یک زیر بهینه نزدیک به راه حل بهینه باشد نیاز داریم.

۵.۳.۲ متد تامسون و برنارد

برنارد و تامسون یک الگوریتم تکراری برای محاسبه تطابق یا شار نوری پیشنهاد دادند. آنها از مقدار اطمینان P_{ij} برای نشان دادن احتمال توکن i در فریم اول با فریم j در فریم دوم تطابق دارد استفاده کردند. احتمالات اولیه P_{ij}^0 با استفاده از سطح خاکستری در محدوده کوچک حول موقعیت توکن‌های وابسته که بصورت زیر می‌باشد استفاده کردند:

$$P_{ij}^0 = \frac{1}{c + w_{ij}}$$

که

$$w_{ij} = \sum_{dy=-w}^{dy=w} \sum_{dx=-w}^{dx=w} (f_1(x_i + dx, y_i + dy) - f_2(x_j + dx, y_j + dy))^2$$

تکرارهای زیرتوالی بصورت زیر تعریف می‌شود:

$$P_{ij}^{n} = \frac{P_{ij}^{n}}{\sum_j \tilde{P}_{ij}^{n}} \quad (۵.۲۴)$$

$$\tilde{P}_{ij}^{n} = P_{ij}^{n-1}(A + Bq_{ij}^{n-1}) \quad (۵.۲۵)$$

$$q_{ij}^{n-1} = \sum_k \sum_l P_{kl}^{n-1} \quad (۵.۲۶)$$

که k یک همسایگی از i و l یک همسایگی j، که $\|(x_i, y_i) - (x_k, y_k)\| \leq D_{max}$ و $\|V_{ij} - V_{kl}\| \leq V_{max}$ که V_{ij} شار نوری i امین توکن است که j امین توکن در فریم بعدی تطابق دارد) نمایش الگوریتم برنارد و تامپسون در ۵.۷ آورده شده است. در فریم اول سه نقطه وجود دارد ($i = 1,2,3$) که در شکل ۵.۷a نشان داده شده است. ساختمان داده ممکن برای این مثال در شکل ۵.۷b نشان داده شده است. یک لیست تطابق با هرنقطه وجود دارد. اولین عنصر در هر لیست با یک نقطه در فریم‌های اول مربوط می‌باشد. عناصر متوالی لیست شارهای نوری ممکن با احتمالات مطابقتی هستند. برای نمونه اولین عصنر در لیست اول (۴و۱۰) است که با اولین نقطه در فریم اول هم رتبه می‌باشند.دومین عنصردر لیست ((5,10),7) می‌باشد که تطابق اولین عنصر در اولین فریم با اولین عنصر در دومین فریم رانشان می‌دهد. شار نوری در این حالت (5,10) با احتمال اولیه 0.7 در شکل 5.7c با ماتریس $\tilde{P}_{ij}^{1}, q_{ij}^{0}$ و P_{ij}^{1} نشان داده شده اند.

۵.۴ تطابق حرکت با استفاده از چندین فریم

n فریم متوالی که با $f^1, f^2, \dots, f^n$ نشان داده می‌شود، به ما داده شده است. فرض می‌کنیم که مهترین توکن در هر فریم با استفاده از یک تشخیص دهنده لبه یا با اپراتور قبلا تشخیص داده شده است. بنابراین هر فریم f^i یک مجموعه از نقاط می‌باشد. با تطابق دادن نقطه i ام با فریم j ام، ما مختصات مختصات دوبعدی را با یک بردار X_i^j نشان می‌دهیم. هدف ما پیدا کردن تطابق نقطه به نقطه Φ^k بین نقاط فریم k و فریم $k+1$ می‌باشد.

غیر واقعی نیست که فرض کنیم که اشیای مکانی فواصل کوتاهی در بازه زمانی کم طی می‌کنند. حرکت تطابقی یا هموار یا یکنواخت می‌باشد. اگر بازه زمانی بین فریم‌ها کوتاه باشد، پس تصویرافکنی دوبعدی از حرکت سه بعدی کوتاه و هموار می‌باشد.بنابراین موقعیت یک نقطه در فریم بعدی تقریبا از موقعیت در فریم قبلی خواهد بود. همواری حرکت به کمترین تغییر در سرعت اشاره دارد که شی نمی‌تواند جهت و سرعتش را همزمان تغییر دهد. از این جهت اشیا در یک مسیر یکنواخت تقریبی دنبال خواهند شد. ما می‌توانیم مطابقت را با کمینه کردن تابع یکنواخت مبدایی ایجادکنیم، که مسیر یکنواخت تقریبی ترجیح داده خواهد شد.

$$\delta(X_p^{k-1}, X_q^k, X_r^{k+1}) = \frac{\left\|X_p^{k-1}X_q^k - X_q^k X_r^{k+1}\right\|}{\sum_{x=1}^{m}\sum_{z=1}^{m}\left\|X_x^{k-1}X_{\Phi^{k-1}(x)}^k - X_{\Phi^{k-1}(x)}^k X_z^{k+1}\right\|} + \frac{\left\|X_q^k X_r^{k+1}\right\|}{\sum_{x=1}^{m}\sum_{z=1}^{k+1} X_{\Phi^{k-1}(x)}^k X_z^{k+1}}$$

که

$$1 \le p, q, r \le m; 2 \le k \le m-1; q = \Phi^{k-1}(p)$$

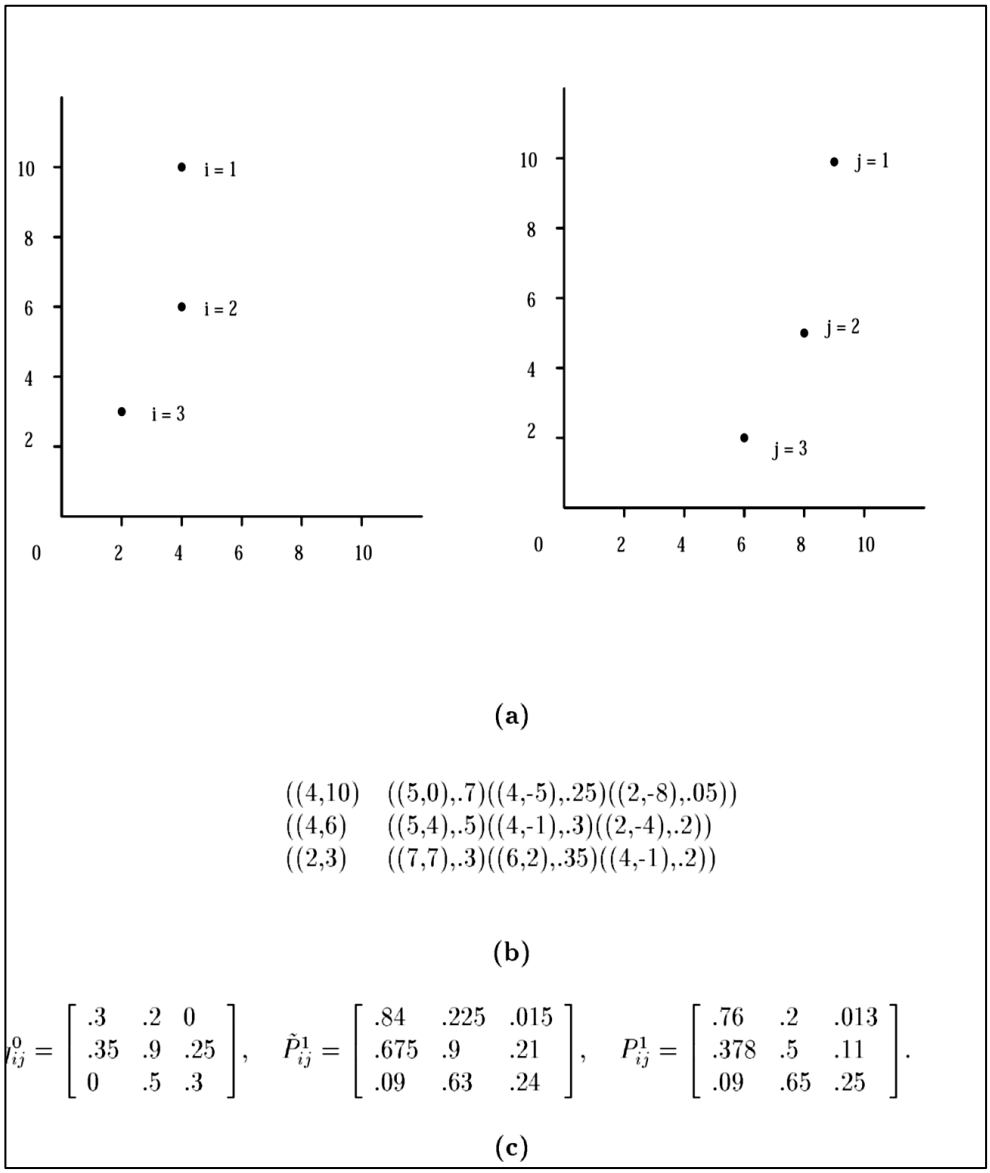
$X_q^k X_r^{k+1}$ برداری از نقطه p از فریم k به نقطه r در فریم $k+1$ می‌باشد. و(----) به بزرگی بردار (- --- اشاره دارد.

تابع یکنواخت تقریبی از معیارهای زیر پیروی می‌کند.

- سرعت بین دو فریم متوالی زیاد تغییر می‌کند.
- جهت بین دو فریم متوالی زیاد تغییر می‌کند.
- با جایگزینی یک نقطه به دو فریم متوالی به سمت کوچک شدن تمایل دارد.

تابع یکنواختی در عبارت اول یک تغییر نسبی در حرکت را نشان می‌دهد. در حالی که عبارت دومی‌به یک جایگزین نسبی اشاره می‌کند. عبارت دومی‌تطابق مبدایی را تحمیل می‌کند در حالی در عبارت اول منجر به یکنواختی و خط سیر یکنواخت می‌شود. توجه کنید صورت کسر در هر عبارت مقدار قدر مطلق را نشان می‌دهد. به عنوان مثال صورت کسر عبارت اول یک تغییر مطلق در حرکت یک نقطه q در فریم k را نشان می‌دهد. در حالی که مخرج مجموع مقادیر مطلق را برای همه تطابق‌های ممکن نشان می‌دهد. از این رو، نسبت اندازه نسبی مقادیر را می‌دهد. از آنجایی که، تغییر در حرکت مقدار یک بردار را می‌دهد، بزرگی در عبارت اول با هر دو تغییر سرعت و حرکت ترکیب می‌شوند.

در این فرمول فرض شده است که Φ^l یک تطابق اولیه بایستی معلوم باشد.Φ^k نیز این چنین تعریف می‌شود: $\sum \delta(X_p^{k-1}, X_q^k, X_r^{k+1})$ که کمینه شده است. برای متد هموارپایه‌ای که معنی دار باشد بایستی تطابق اولیه معلوم باشد. با دادن یک تطابق اولیه صحیح، الگوریتم دو خط سیر صحیح رشد خوهد کرد. با به کار گیری محدودیت هموار به تنهایی و بدون دانستن هر جزیی درباره حرکت اولیه نقاط ممکن بسیاری از حالات منجر به مجموعه‌ای مجموعه مسیر خط شود.



(a)

((4,10) ((5,0),.7)((4,-5),.25)((2,-8),.05))
((4,6) ((5,4),.5)((4,-1),.3)((2,-4),.2))
((2,3) ((7,7),.3)((6,2),.35)((4,-1),.2))

(b)

$$P_{ij}^{0} = \begin{bmatrix} .3 & .2 & 0 \\ .35 & .9 & .25 \\ 0 & .5 & .3 \end{bmatrix}, \quad \tilde{P}_{ij}^{1} = \begin{bmatrix} .84 & .225 & .015 \\ .675 & .9 & .21 \\ .09 & .63 & .24 \end{bmatrix}, \quad P_{ij}^{1} = \begin{bmatrix} .76 & .2 & .013 \\ .378 & .5 & .11 \\ .09 & .65 & .25 \end{bmatrix}.$$

(c)

شکل ۵.۷: نمایش یک الگوریتم برنارد–تامسون با $A = 3, B = 3, V_{max} = 2, D_{max} = 4$

برای $k = 2$ تا $n - 1$ انجام بده:

a) یک ماتریس $m \times m$ با نقاطی از فریم k همراه سطرها و نقاطی از فریم $k + 1$ همراه با ستون‌ها

فرض کنید $M[i,j] = \delta(X_p^{k-1} X_i^k X_j^{k+1})$ وقتی که $\Phi^{k-1}(\rho) = i$

b) برای $a = 1$ تا m انجام بده

i) کوچکترین عنصر $[i, l_i]$ در هر سطر i از M را تعیین کنید.

ii) ماتریس اولویت B که $B[i,l_i] = \sum_{j=1, j\neq l_i}^{m} M[i,j] + \sum_{k=1, k\neq i}^{m} M[i,j]$ برای هر i را محاسبه کنید.

iii) جفت $[i, l_i]$ را با بیشترین مقدار اولویت $B[i, l_i]$ انتخاب کنید و $\Phi^k(i) = l_i$ را بسازید.

iv) سطر i و ستون l_i از m را ایجاد کنید.

۵.۸: تطابق حرکت با استفاده از چندین فریم

یک الگوریتم حریصانه غیر تکراری A در شکل ۵.۸ نشان داده شده است. این الگوریتم تطابق نقاط در یک فریم با نقاطی در فریم بعدی تخصیص می‌دهد، با نگه داشتن کل تابع هموارمبدایی که همان نزدیک به کمترین احتمال باشد به علاوه نسبتاً به هریک از تخصیص‌ها منصف باشد.

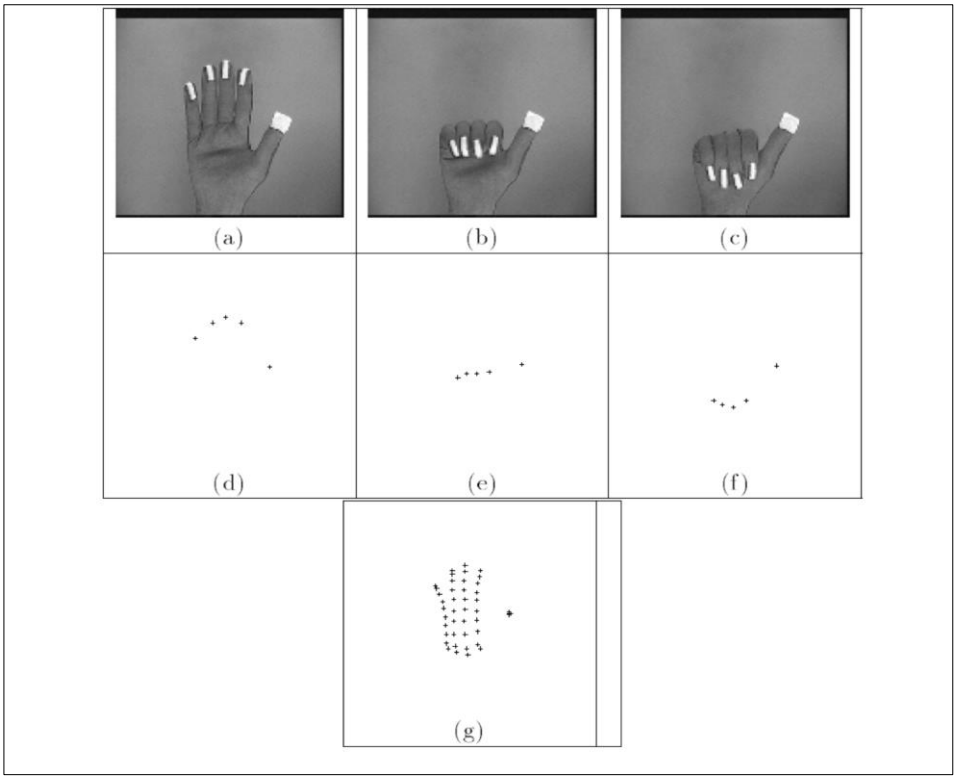
دراین الگوریتم وقتی که کمترین سطرها در نظر گرفته می‌شود این وقتی اتفاق می‌افتد که بیش از یک مینیمم در همان ستون j قرار داشته باشد. یعنی، چند نقطه در فریم k برای نقطه j در فریم $k + 1$ رقابت کنند. برای بذ=دست آوردن نگاشت یک به یک ما بایستی یکی از نقاط را انتخاب کنیم. به هر حال این طرح نبایستی فقط کمترین فقط کمترین مقدار ترکیبی ممکن را انتخاب کند ولی بایستی یک ترکیبی را ترجیح دهد که هر تطابق منفرد به صورت منصفانه باشد. تطابق از فریم k به فریم $k + 1$ شامل m نقطه می‌باشد. کمترین تطابق ممکن برای بعضی از نقاط $m - 1$ مطلوب باشد ولی برای نقطه m مطلوب نباشد. الگوریتم می‌تواند یک تطابقی را مد نظر قراردهد که برای همه نقاط مناسب باشد. درهمان زمان نبایستی با یک تابع مسیریکنواخت الگوریتم را تمام کنیم. الگوریتم برای حفظ چنین شرایطی طراحی شده است.

منطق پشت اندازه‌گیری اولویت این است که اگر $[i,j]$ تخصیص داده نشود هر عنصر دیگری هم در امتداد سطر iو ستون j تخصیص داده نخواهد شد. هر عنصر سطر i و ستون j می‌توانند مقداد بگیرند. فرض کنید همه مقادیر به صورت مساوی احتمال پذیر باشند. مقدار میانگین یک معیار برای انتخاب

تخصیص‌های متوالی مناسب می‌باشد. الگوریتم بایستی بیشترین مقدار اولویت را انتخاب و آنرا تخصیص دهد. مقدار اولویت یک تخمین سخت از تخصیص جایگزین به $[i,j]$ می‌باشد.

این الگوریتم خصوصیت خوبی دارد این است که حداقل هزینه تخصیص را انتخاب می‌کند اگر فقط دو نقطه در فریم وجود داشته باشد. ماتریس m را با $M[1,1]=0.6، M[1,2]=0.3، M[2,1]=0.7، M[2,2]=0.2$، کمترین مقدار سطر ۱ عنصر $[1,2]$ با مقدار 0.3 میباشد. در حالی که کمترین مقدار سطر۲ عنصر $[2,2]$ با مقدار 0.2 است. بنابراین $B[1,2]=0.2+0.6=0.8$ و $B[2,2]=0.7+0.3=0.1$ در نتیجه $B[2,2]>B[1,2]$ از این رو تطابق تطابق $(2,2)$ را از اول انتخاب می‌کنیم. سپس سطر ۲ با ستون ۱ را با مقدار زیادتری پوشش می‌دهیم. بعد فقط امکان تخصیص $[1,1]$ را برمی‌گزینیم. برای این تخصیص $\delta=M[1,1]+M[2,2]=0.6+0.2=0.8$ که حداقل امکان برای پیکربندی می‌باشد.

نتایج این الگوریتم در شکل ۵.۹ نشان داده شده است.



شکل ۵.۹ تولید مسیر (a) – (c) توالی تصاویر – نمایش تصاویر انتخاب شده. (d)-(f) نمایش نقاط ناخن در تصاویر g) مسیر ناخن‌ها

۵.۵ ساختار از حرکت

متد ساختار از حرکت (Structure From Motion) در بینایی ماشین به خصوصیات فیزیکی اشیا نمایش داده شده در صحنه اشاره دارد. از قبیل ساختار سه بعدی و حرکت، یک سری از مسیرهای دوبعدی. از دهه قبل این علاقه در جامعه بینایی ماشین وجودداشته است. دو کلاس متد برای SFM وجود دارد. متدهای جایگزین و همزمان. در روش‌های جایگزین، بعد مختصات نقاط حرکات اشیا و حرکت سه بعدی از فریم‌های متوالی بدست می‌آید. این مسئله در عبارات معادلات سیستمی‌غیرخطی فرموله می‌شوند. در موقعیات دوبعدی نقاط در حال حرکت فریم‌ها داده می‌شود.کار تئوریک جالب وجود دارد که وابسته به تعدای نقاط می‌باشند که این نقاط نیازمند یک راه حل می‌باشد که این راه حل باید منحصر به فرد و تاثیر نویز روی این راه حل مطالعه شده باشد. در متدهای همزمان فیلد شارنوری برای پوشش حرکت سه بعدی و مقادیر عمق استفاده می‌شود. رویکردهای قبلی در این متد مسایل ساده شده‌ای که شامل بعضی فرضیات وابسته به حرکت و اشیا، فقط فرضی از انتقال حرکت، فقط چرخش حرکت، دانستن عمق اشیا. اخیرا هیگر وجیسون یک متد کلی برای محاسبه حرکت سه بعدی (چرخش و انتقال) و عمق از شار نوری را پیشنهاد داده اند. متدشان ابتدا انتقال، سپس چرخش و بعداً عمق را محاسبه می‌کرد. متد این دو نفر می‌تواند بصورت زیر تعریف شود.

فرض کنید حرکت یک نقطه $(\mathrm{X},\mathrm{Y},\mathrm{Z})$ روی یک شی که می‌تواند با $\vec{V}=-\vec{T}-\vec{w}\times\vec{r}$ نشان داده می‌شود قرار دارد. که $\vec{V}$ بردار سرعت و $\vec{T}=(T_x,T_y,T_z)$ بردار انتقال است و $\vec{w}=(w_x,w_y,w_z)$ بردار چرخش و r برداری از مبدا به نقطه (X,Y,Z) و $\times$ نشان‌دهنده ضرب برداری می‌باشد. شاری نوری (U,V) در زیر نشان می‌دهیم(فرض طول کانونی برابر با ۱-)

$$u=\left(-\frac{T_x}{Z}-w_y+w_z y\right)-x\left(-\frac{T_z}{Z}-w_x y+w_y x\right) \qquad (۵.۲۷)$$

$$u=\left(-\frac{T_y}{Z}-w_z x+w_x\right)-x\left(-\frac{T_z}{Z}-w_x y+w_y x\right) \qquad (۵.۲۸)$$

این معادله می‌تواند بصورت زیر نوشته شود:

$$\vec{\theta}(x,y)=p(x,y)A(x,y)\vec{T}+B(x,y)\vec{w} \qquad (۵.۲۹)$$

که $p(x,y)=1/Z(x,y)$ عمق معکوس، $A(x,y)=\begin{bmatrix}-1 & 0 & x\\ 0 & -1 & y\end{bmatrix}$ و $B(x,y)=\begin{bmatrix}xy & -(1+x^2) & x\\ 1+y^2 & -xy & -x\end{bmatrix}$ هستند.

ماتریس‌های A, B فقط روی موقعیت معلوم تصویر وابسته هستندو روی موقعیت‌های ناشناخته وابستگی ندارند. برای هر نقطه در تصویر، یک معادله جداگانه می‌تواند در این فرم نوشته شود و می‌توانند با یکدیگر در یک معادله ماتریس بصورت زیر ترکیب شوند:

$$\vec{\theta} = A(T)\vec{\rho} + B\vec{w}$$
$$\vec{\theta} = C(T)\vec{q}$$

که $\vec{\theta}$ شامل سرعت‌های تصویر برای همه نقاط تصویر، و $\vec{\rho}$ نشاندهده عمق می‌باشد، $A(T) = \begin{bmatrix} A(x_1,y_1)(T) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A(x_n,y_n)(T) \end{bmatrix}$ از جمع ماتریس $A(x,y)T$ برای هر نقطه بدست می‌آید و $B = \begin{bmatrix} B(x_1,y_1) \\ \vdots \\ B(x_n,y_n) \end{bmatrix}$. با جمع آوری ماتریس $B(x,y)$ بدست می‌آید و $\vec{q}$ با جمع آوری یک بردار عمق‌های مجهول و سرعت‌های منطقی بدست می‌آید و

$$C(T) = \begin{bmatrix} | & | \\ A(T) & B \\ | & | \end{bmatrix}$$

با قرار دادن ستون‌های B با ستونهای A بدست می‌آید.

حداقل–مربعات تبدیل را تخمین می‌زنند وقتی که عبارت زیر را روی همه تبدیل‌ها کاندید، چرخش و مقادیر عمق کمینه می‌تواند استفاده شود.

$$E(\vec{T}) = \left\|\vec{\theta} - C(\vec{T})\vec{q}\right\|^2$$

کمینه کردن این عبارت روی همه $\vec{T}$ها و$\vec{q}$ها برابراست با تابع زیر بر روی $\vec{T}$:

$$R(\vec{T}) = \left\|\vec{\theta}C^{\perp}(\vec{T})\right\|^2 \qquad (۵.۲۹)$$

که $C(\vec{T})$ پایه بهنجار برای مکمل قایم $C^{\perp}(\vec{T})$ می‌باشد.

زمانی که $\vec{T}$ محاسبه می‌شود، سرعت منطقی می‌تواند تعیین شود. یک بردار واحد $\vec{d} = (x, y, \vec{T})$ که در آن

$$\vec{d}^t(x,y,T)\boldsymbol{A}(x,y)\vec{T} = 0$$

را در نظر بگیرید. با ضرب کردن در هر دو طرف معادله ۵.۲۹ وابستگی روی $p(x,y)$ را ازبین می‌برد.

$$\vec{d}^t(x,y,T)\boldsymbol{\theta}(x,y) = \vec{d}^t(x,y,T)\boldsymbol{B}(x,y)\vec{w}$$

تعدا زیادی از بردارهای جریان می‌تواند در یک حداقل–مربعات تخمین برای w استفاده شوند.

$$\vec{d}^t(x_1,y_1,T)\boldsymbol{\theta}(x,y) = \vec{d}^t(x_1,y_1,T)\boldsymbol{B}(x_1,y_1)\vec{w} \quad (۵.۳۳)$$

$$\vdots = \vdots \quad (۵.۳۴)$$

$$\vec{d}^t(x_n,y_n,T)\boldsymbol{\theta}(x_n,y_n) = \vec{d}^t(x_n,y_n,T)\boldsymbol{B}(x_n,y_n)\vec{w} \quad (۵.۳۵)$$

$$s(x_1,y_1) = D(x_1,y_1)\vec{w} \quad (۵.۳۶)$$

$$\vdots = \vdots \quad (۵.۳۷)$$

$$s(x_n,y_n) = D(x_n,y_n)\vec{w} \quad (۵.۳۸)$$

که $s(x,y)$ ماتریس 1×1 می‌باشد، $D(x,y)$ یک ماتریس 1×3 معادلات بالایی می‌تواند بصورت زیر نوشته شوند:

$$s = D\vec{w}$$

که s یک بردار $n \times 1$ و D یک ماتریس $n \times 3$ ما می‌توانیم $\vec{w}$ را با متد معکوس محاسبه شود. سرانجام، وقتی که سرعت چرخش و جابجایی هر دو معلوم باشد. معادلات ۵.۲۷ و ۵.۲۸ می‌توانند برای حل یک عکس نامعلوم در هر نقطه از تصویر با استفاده از متد حداقل–مربعات استفاده شوند.

تمرینات

۱. معادلات ۵.۷ و ۵.۸ را اثبات کنید.

۲.معادله‌ی ۵.۱۷ را از ۵.۵ استخراج کنید.

۳.معادله‌ی ۵.۲۰ را استخراج کنید.

۴.معادلات ۵.۲۷ تا ۵.۳۲ را استخراج کنید.

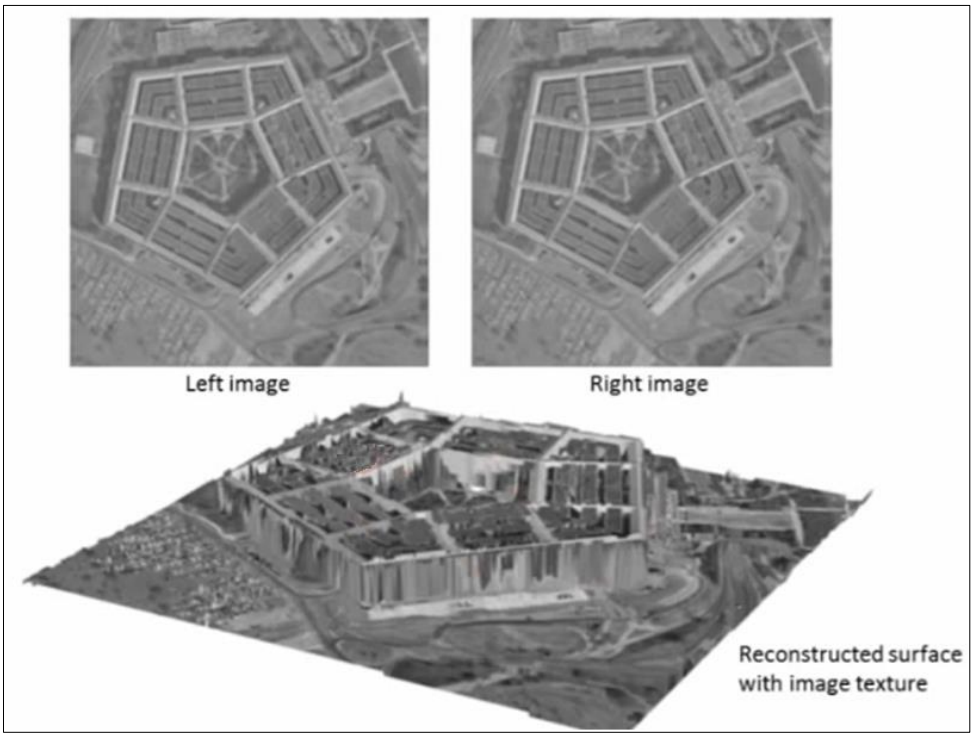
فصل ۶

بینایی استریو

۶.۱.مقدمه

دنیا سه بعدی است، اما تصاویر دو بعدی هستند. بنابراین یکی از ابعاد در روند گرفتن عکس از بین می‌رود. یکی از مهمترین امور در بینایی ماشین بازیابی بعد سوم از یک یا چندین عکس است. روش‌های مختلفی برای بازیابی عکس‌های سه بعدی از دو بعدی وجود دارد که اغلب مورد استفاده قرار گرفته می‌شوند و در تحقیقات بینایی ماشین مورد مطالعه قرار می‌گیرند. در برخی از کاربردها نیاز به استخراج اطلاعات سه بعدی از تصاویر دوبعدی داریم. برای مثال روباتی را در نظر بگیرید که فقط مجهز به دوربین است و می‌خواهیم این ربات را توانا به تشخیص فاصله موانع کنیم. در این جا نیاز داریم بعد گمشده‌ی سوم (یعنی عمق تصویر) را استخراج کنیم.

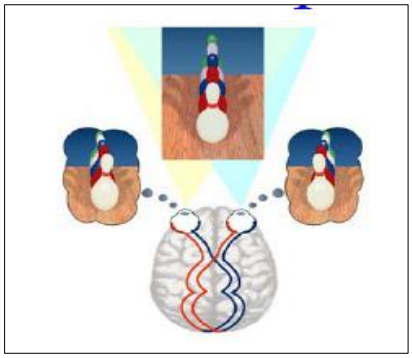
به عبارت دیگر چگونه می‌توانیم با استفاده از دو یا چند تصویر دوبعدی از یک جسم یا صحنه یک مدل سه‌بعدی از آن جسم یا صحنه را استخراج کنیم. برای حل این مسئله چندین راه‌حل از جمله بینایی استریو، استفاده از حرکت، سایه‌زنی و استفاده از بافت پیشنهاد شده است. در اینجا استخراج عمق توسط بینایی استریو مد نظر ماست. به عنوان نمونه‌ای از کاربرد بینایی استریو، فرض کنید بخواهیم از دو تصویر هوایی مدل سه‌بعدی ساختمانی را استخراج کنیم (چیزی شبیه به کاری که GoogleEarth انجام می‌دهد). در این صورت می‌توان ارتفاع ساختمان از زمین را با دو تصویر گرفته شده‌ی استریو تخمین می‌زنیم سپس یکی از تصاویر را به عنوان بافت به صورت شکل ۱ روی این مدل می‌کشیم.



شکل ۱. نمونه‌ای کاربرد بینایی استریو

۶.۲.مدل بینایی استریو انسان

تصویر دریافت شده در مغز از دو تصویر گرفته شده توسط چشم سمت چپ و چشم سمت راست تشکیل می‌شود. همانطور که می‌دانید(و می‌بینید!) تصویر دریافت شده توسط هریک از چشم‌ها قدری جابه‌جا شده‌ی تصویر چشم دیگر است که این به خاطر فاصله دو چشم روی جمجمه است. هرچه جسمی‌که به آن می‌نگریم نزدیک‌تر باشد جابه‌جایی آن بین تصاویر دو چشم بیشتر و هرچه دورتر باشد جابه‌جایی آن کمتر است.



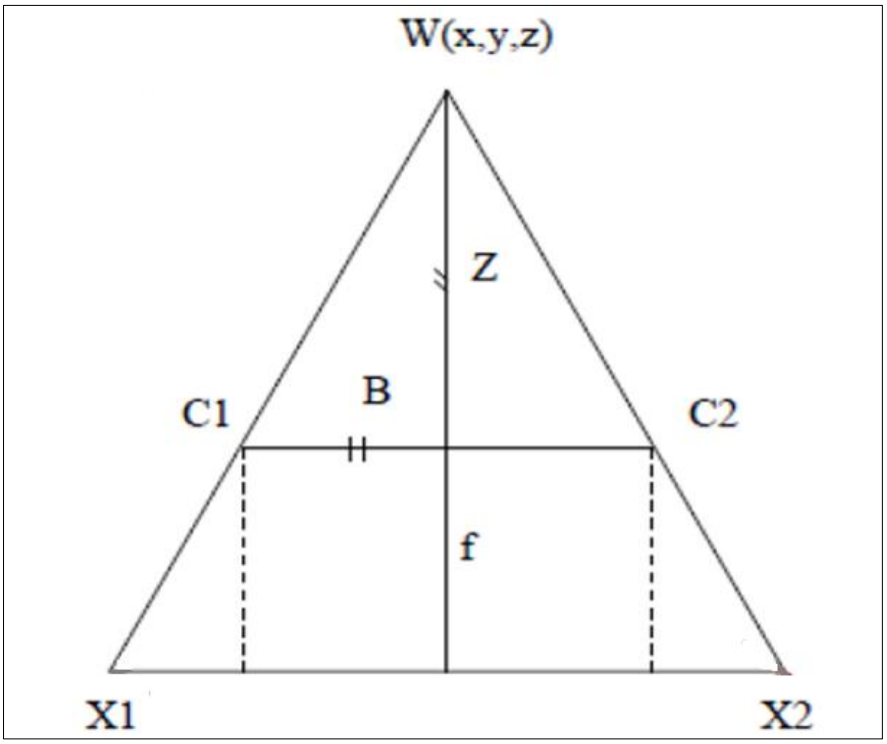
شکل ۲. سیستم بینایی انسان

اساس کار بینایی استریو و یکی از راه‌های مورد استفاده مغز برای استخراج اطلاعات سه‌بعدی همین می-باشد. نمایش‌گرهای یه بعدی نیز از این تکنیک برای فریب مغز استفاده می‌کنند. به این صورت که دو تصویر جداگانه برای هر یک از چشم‌ها ارائه می‌شود و تصویر غیرمربوط به هریک از چشم‌ها توسط عینک مخصوص فیلتر می‌شود.

شکل ۳. تصاویر استریو که هردو از یک نمایشگر پخش می‌شوند و توسط عینک فیلتر می‌شوند

۶.۳.هندسه بینایی استریو

برای استخراج اطلاعات سه بعدی از تصویر دوبعدی حداقل به دو تصویر از همان صحنه نیاز داریم. برای مدل سازی بینایی استریو شکل ۴ را در نظر بگیرید. در این شکل تصویر نقطه‌ی $W(X,Y,Z)$ در فضای سه‌بعدی به پیکسل‌های $P_1(x,y)$ و $P_2(x,y)$ در صفحه دوربین‌ها نگاشت شده است. در شکل $C1,C2$ محل قرارگیری دوربین‌ها(دوربین‌ها در یک صفحه قرار دارند)، f فاصله کانونی دوربین‌ها(یکسان فرض شده اند)، B فاصله بین دو دوربین و $x1,x2$ طول محل قرارگیری تصویر روی گیرنده تصویر هستند. البته باید خاطر نشان کرد که که واحد طول یک نقطه در تصایر دیجیتال پیکسل است، اینجا فرض می‌کنیم مدل دوربین را داریم و طول‌ها را از پیکسل به متر تبدیل کرده‌ایم. برای داشتن مدل سه‌بعدی (و نه در اندازه‌ی فیزیکی واقعی) داشتن $x1$ و $x2$ به پیکسل کافی است و حتی نیازی به دانستن f و B نمی‌باشد. ضمناً فرض شده



شکل ۴. هندسه‌ی ساده‌ی بینایی استریو

محور xها رو C1 و C2 در جهت‌های مخالف هم هستند (البته می‌توان هم‌جهت در نظرشان گرفت که نهایتا تفاوتی ایجاد نمی‌کند). Z عمق نقطه مورد نظر است که می‌خواهیم آن‌را به دست بیاوریم.

از تشابه مثلثات داریم:

$$\frac{Z+f}{Z}=\frac{x_1+x_2+B}{B} \qquad (۶.۱)$$

رابطه‌ی بالا را برای Z ساده می‌کنیم:

$$Z=\frac{f\times B}{x1+x2} \qquad (۶.۲)$$

از رابطه‌ی ۲ مشاهده می‌کنیم که در مدل هندسی ساده بالا عمق نسبت عکس با مجموع طول طول تصویر نقاط تشکیل شده روی صفحه‌ی دوبین‌ها دارد. به فاصله(اختلاف) این دو نقطه عدم تطابق (Disparity) می‌گویند. کل هدف بینایی استریو یافتن نقاط متناظر و محاسبه‌ی عدم تطابق است. توجه داشته باشید که اختلاف رابطه معکوسی با عمق دارد. اشیای نزدیک به دوربین اختلاف بیشتری را تولید می‌کنند و زمانی که اشیا نسبت به دوربین دورتر باشند اختلاف کمتری را تولید می‌کنند. الگوریتم‌های بررسی شده در این فصل برمبنای یافتن نشانه‌ها (Token Based) و برمبنای همبستگی هستند.

مثال. فرض کنید چهار جفت نقطه‌ی متناظر در دو تصویر استریو را مشخص کرده‌ایم. **فرض کنید** طول نقطه‌ی اول در تصویر سمت چپ ۵۵ میلیتر و در تصویر دیگر ۴۵ میلیمتر است ($x1=45mm,x2=55mm$). فاصله‌ی دو دوربین از هم و فاصله‌ی کانونی به ترتیب ۱۰ سانتی‌متر و ۵۰ میلیمتر است($B=10cm,f=50mm$).

$$Disparity=d=x1+x2=45+55=100mm$$

$$Z=\frac{f\times B}{x1+x2}=\frac{50\times 10\times 1000}{100}=5cm$$

اگر عملیاتی را که در بالا انجام دادیم برای تمام پیکسل‌های موجود انجام دهیم، عمق در تمام نقاط صحنه را به دست خواهیم آورد (روش همبستگی) و یا می‌توانیم برای نقاط خاصی از صفحه به دست آوریم و سپس با درون‌یابی بین این نقاط عمق را تمام نقاط تصویر تخمین بزنیم (روش‌های مبتنی بر نشانه).

آیا مسئله حل شد؟ نه، در واقع چالش اصلی در بینایی استریو یافتن نقاط متناظر بین تصاویر سمت چپ و راست است. به عبارت دیگر پیکسل $P1(x,y)$ در تصویر چپ که تصویر نقطه $W(X,Y,Z)$ در دنیای واقعی معادل کدام یک از پیکسل‌های موجود در تصویر سمت راست است؟ البته اگر مانند شکل ۴ فقط یک نقطه داشتیم، یافتن این تناظر کار ساده‌ای بود اما در تصاویر واقعی هزاران نقطه داریم، که بسیاری از آن‌ها مشابه هم هستند اما متناظر نیستند! اگر بخواهیم تمام جفت نقاط مشابه را بررسی کنیم (روش مبتنی بر همبستگی) هزینه محاسباتی بالایی را پرداخت خواهیم کرد. می‌توان عمق را در نقاط خاصی به دست آورد (مثلا گوشه‌ها یا لبه‌ها) و سپس با درون‌یابی عمق را در سایر نقاط نیز تخمین زد. این روش‌ها، روشه‌های مبتنی بر نشانه هستند و در صورتی درست کار می‌کنند که به تعداد کافی نقاط ویژه وجود داشته باشد.

۶.۴.تناظریابی

عموما الگوریتم‌های بینایی استریو به دو گروه تقسیم می‌شوند. یکی الگوریتم‌های مبتنی بر همبستگی[1] و دیگری الگوریتم‌های مبتنی بر ویژگی[2] هستند. الگورتیم‌های مبتنی بر همبستگی کل تصویر سمت راست و چپ را برای یافتن نقاط مشابه اسکن می‌کنند. تشابه دو نقطه را نیز با استفاده از یکی از معیارهای تشابه یا عدم تشابه که مبتنی بر همبستگی گیری است تشخیص می‌دهند. این نوع روش‌ها مناسب تصاویر با بافت فراوان هستند و عمق در تمام نقاط را به دست می‌دهند(عموما هزینه محاسباتی بالاتری هم دارند). روش‌های نوع دوم مبتنی بر تناظر یابی در نقاط خاص و ویژه هستند مانند لبه‌ها، گوشه‌ها و خطوط

[1] - Correlation Based

[2] - Future Based

هستند. از آنجایی که این دسته فقط عمق را در نقاط خاصی از تصویر محاسبه می‌کند، برای به دست آوردن عمق در سایه نقاط باید از درون‌یابی استفاده نمود. این دسته در برابر چرخش، تغییرات روشنایی مقاومت‌تر هستند. در این کتاب الگوریتم‌های مورد بحث ما از دسته اول هستند.

۶.۴.۱.تناظریابی مبتنی بر همبستگی

همانطور که گفتیم یافتن به دلیل شباهت زیاد نقاط به صورت منفرد به یکدیگر تناظر بین نقاط مبهم است. بنابراین برای یافتن نقطه‌ی متناظر $P_1(x,y)$ به جای مقایسه تک تک نقاط بلوک یا پنجره‌ای اطراف $P_1(x,y)$ در نظر گرفته و شبیه‌ترین بلوک در تصویر دیگر به این بلوک را می‌یابیم

.

شکل ۳. تصاویر چپ و راست برای استریو، پنجره‌ی انتخاب شده حول نقطه‌ی مورد جستجو

در اینجا باید خاطر نشان کرد که حتما لازم نیست جستجو در حوزه تصویر انجام شود، می‌توان تطابق را در نمایش‌های دیگری از تصاویر مثل لاپلاسین گوسی (LOG)، اندازه‌ی گرادیان و.. انجام داد.

معیارهای متفاوتی برای شباهت یا عدم شباهت دو تصویر(یا دو بلوک از تصویر) وجود دارند که همه به نوعی همبستگی بین دو تصویر را اندازه می‌گیرند. هریک از این معیارها مزایا و معایب مربوط به خود را دارند. در اینجا برای آشنایی خواننده چند معیار پرکاربرد معرفی می‌گردند.

۶.۴.۲.معیارهای شباهت/عدم شباهت

معیار مجموع مربع تفاضلات (Sum Square Differences)

این معیار به صورت رابطه‌ی زیر تعریف می‌شود:

$$SSD = \sum\sum(I_{left} - I_{right})^2 \qquad (۶.۳)$$

به سادگی پیکسل به پیکسل دو تصویر را از هم کم و مربع کرده و جمع می‌کنیم(البته از اسمش هم معلوم بود چه کار میکند!). اگر توان را به داخل اثر دهیم ترمی‌به صورت حاصل ضرب به دست می‌آید

که به همان همبستگی متقابل[1] است. این معیار نوعی معیار اندازه‌گیری عدم شباهت است و وقتی حداکثر شباهت وجود دارد مینیمم(صفر) است.

قدرمطلق تفاضلات[2]

$$AD = \sum\sum\left|(I_{left} - I_{right})\right| \qquad (۶.۴)$$

همان بالایی است که برای اندازه‌گیری خطا به جای نرم ۲ (مربع کردن) از نرم ۱ (قدرمطلق) استفاده می‌کند.

همبستگی متقابل

این معیار به صورت زیر اندازه گرفته می‌شود:

$$CC = \sum\sum I_{left}I_{right} \qquad (۶.۵)$$

این معیار مثل ضرب نقطه‌ای دو بردار است. اگر دو تصویر متشابه باشند (بردارشان در فضا در یک جهت باشد) ضرب نقطه‌ای آنها ماکسیمم می‌شود و اگر نامتشابه باشند (بردارشان در فضا عمود باشد) حاصل ضرب نقطه‌ای آنها مینیمم (صفر) می‌شود.

همبستگی متقابل نرمال شده[3]

گاهی ممکن است به دلیل بزرگی اندازه دو بردار حاصل‌ضرب نقطه‌ای آنها بزرگ شود در حالی که شباهت کمی‌به دارند. مثلا دو بردار [1,1,1] و [1,0,0] و همچنین دو بردار [5,5,5] و [5,0,0] را در نظر بگیرد. جفت بردار دوم تغییر مقیاس داده شده‌ی جفت بردار اول هستند و شباهت آنها باید یکسان باشد در حالی که همبستگی متقابل جفت اول برابر یک و همبستگی متقابل جفت دوم برابر ۲۵ می‌شود و ممکن این‌طور برداشت شود که جفت دوم به هم شبیه‌تر هستند. برای حل این مشکل حاصل همبستگی متقابل را به اندازه دو بردار تقسیم می‌کنیم تا اندازه آنها نرمال شود. به این حالت همبستگی متقابل نرمال‌شده می‌گویند که از رابطه زیر به دست می‌آید:

$$NC = \frac{\sum\sum(I_{left}I_{right})}{\sqrt{\sum\sum I_{left}I_{right}}} \qquad (۶.۶)$$

---

1 - Cross Correlation

2 - Absolute Differences

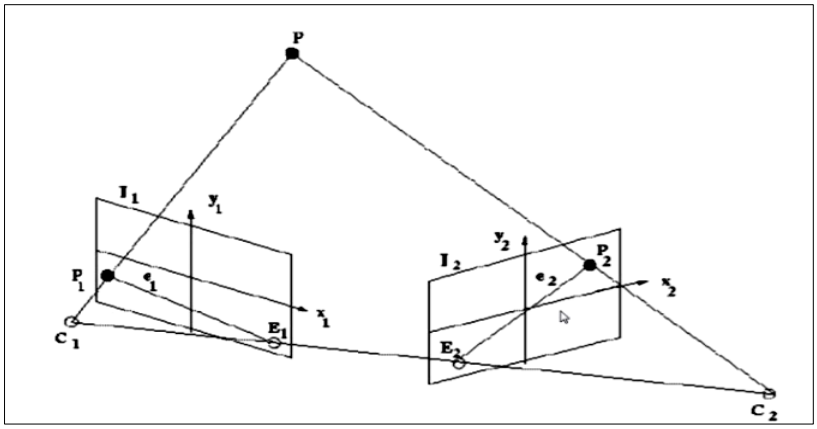
3 - Normalized Cross Correlation

حالا در مثال ذکر شده اندازه شباهت هردو یکسان و برابر یک به دست می‌آید. ممکن است متوجه شده باشید که NC در واقع کسینوس زاویه‌ی بین دو بردار را به دست می‌دهد. این معیار نتیجه‌ی درست‌تری دارد اما این به قیمت افزایش هزینه محاسباتی است.

۶.۴.۳.الگوریتم‌های تناظریابی

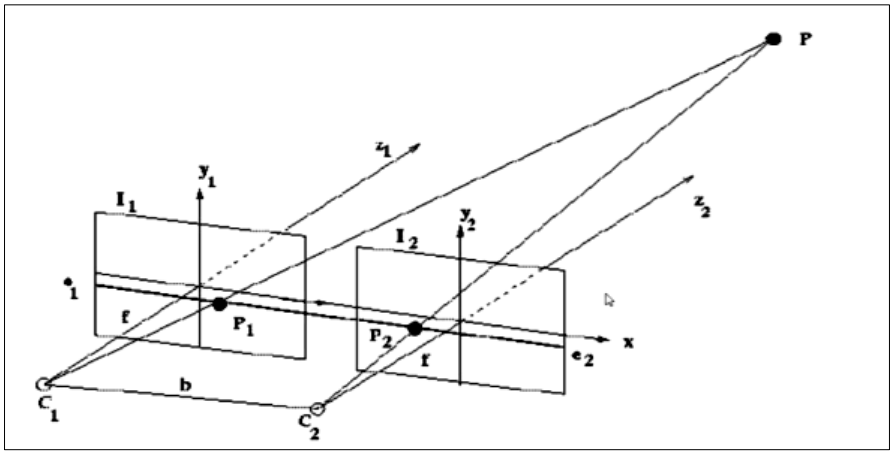
در این بخش دو الگوریتم تناظریابی مبتنی بر هبستگی را معرفی می‌کنیم. اولین الگوریت تطبیق بلوکی (BlockMatching) نام دارد و جستجوی ساده‌ی بلوکی بین دو تصویر برای یافتن بیشترین شباهت و در نتیجه نقاط متناظر انجام می‌دهد و دومی‌که الگوریتم برنارد(Barnard) نامیده می‌شود یک تابع خطا تعریف کرده و سعی می‌کند با یک الگوریتم جستجوی هوشمند آن‌را حل کند. از آن‌جایی که این‌گونه مسائل جز مسائل ترکیبیاتی سخت با بی‌شمار حالت هستند، عملا جستجوی کل تصویر برای یافتن نقاط شباهت ناممکن است. از طرفی با جستجوی کل تصویر ممکن است به چندین نقطه‌ی متناظر دست یابیم که ابهام ایجاد می‌کند. از اینرو از اطلاعات دیگری برای محدود کردن فضای جستجو استفاده می‌کنیم. به عنوان مثال استفاده از هندسه‌ی قرارگیری دوربین‌ها، حداقل و حداکثر رنج عدم تطابق، خطوط Epipolar و... است.

مهم‌ترین محدودکننده، محدودکننده‌ی Epipolar است که به فضای جستجو را از کل تصویر به یک خط تقلیل می‌دهد. خطوط Epipolar در بخش بعدی به تفصیل بحث و نحوه استخراج آن‌ها را تشریح خواهیم کرد اما اینجا صرفاً مفهوم کلی آن بیان می‌کنیم.هر نقطه در هر تصویر مرتبط به یک خط Epipolar در تصویر دیگر است. شکل ۴ را در نظر بگیرید. نقطه‌ی P1 در تصویر روی صفحه‌ی I_1 با



شکل ۵. محدودیت Epipolar

نقطه‌ی P در فضای سه‌بعدی تشکیل یک خط می‌دهند. هر نقطه‌ی دیگری روی آن خط نیز به P_1 تصویر می‌شود. اگر آن خط را به به صفحه‌ی I_2 تصویر کنیم خط Epipolar به دست می‌آید. می‌توان با P P_1, و P_2 یک صفحه تشکیل داد و محل تقاطع این صفحه با I_2 خط Epipolar می‌شود. حالا با داشتن خط Epipolar فقط در امتداد آن روی تصویر دوم برای نقطه‌ی مشابه P_1 جستجو می‌کنیم. اگر دوربین‌ها در یک خط باشند و محورهای آنها نیز موازی هم باشد در این صورت خط Epipolar در محور افقی قرار می‌گیرد. در اکثر موارد این فصل نیز فرض می‌کنیم این وضعیت برقرار است و در راستای خط افقی گذرنده از نقطه‌ی مورد نظر جستجو می‌کنیم.



شکل ۶.دوربین‌های تنظیم شده بر یک خط

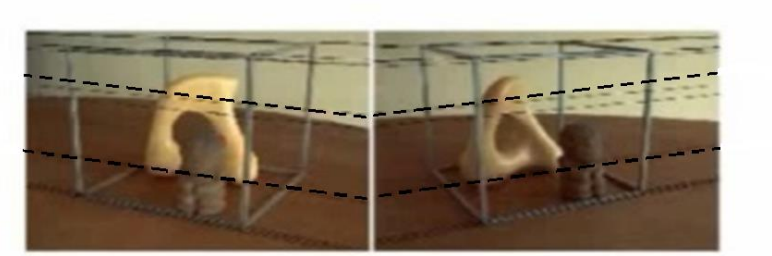
تنظیم تصاویر[1]

می‌توان به جای جستجو در راستای خطوط Epipolar (که ممکن است مورب باشند و پیاده‌سازی را مشکل کنند) تصاویر را با استفاده از اطلاعاتی که در راستای خطوط Epipolar به دست می‌آید طوری warp کنیم که خطوط Epipolar به خطوطی افقی تبدیل شوند. سپس در راستای افقی هر نقطه به دنبال نقطه‌ی متناظر بگردیم. به این کار تنظیم تصاویر[2] می‌گویند. مثالی از این در شکل ۷ نمایش داده شده است.

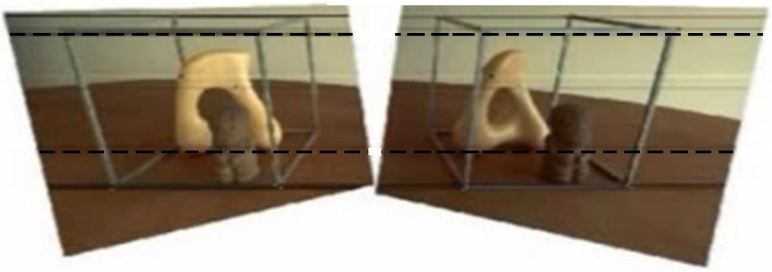
---

[1] - Rectification

[2] - Image Rectification



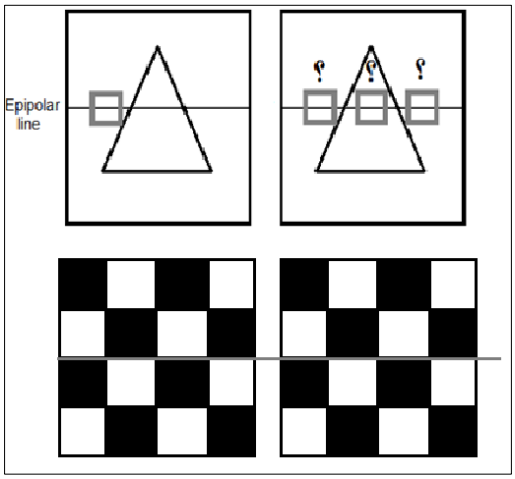
تصاویر اصلی با خطوط اپی پولار هم پوشان



تصاویر یک سو شده طوری که خطوط اپی پولار افقی ساخته اند

شکل ۷. تصاویر اصلی (بالا) و تصاویر تنظیم شده (پایین)

با استفاده از خطوط Epipolar فضای جستجو کاهش می‌یابد اما هنوز هم ممکن فضا بسیار بزرگ باشد و ابهام در تناظر رخ دهد. به عنوان مثال ممکن است تصویری بدون بافت باشد(شکل۷.بالا) یا یک الگوی خاص در یک راستا مدام تکرار شود (شکل ۷.پایین).



شکل۸. ابهام در یافتن تناظر

در اینجا نیز برای کمک به حل مسئله می‌توان چند محدود کننده به مسئله اضافه کرد. می‌توان یکی از فرض‌های شهودی زیر را به مسئله اضافه نمود.

-محدودیت رنج عدم تطابق

-یکتایی

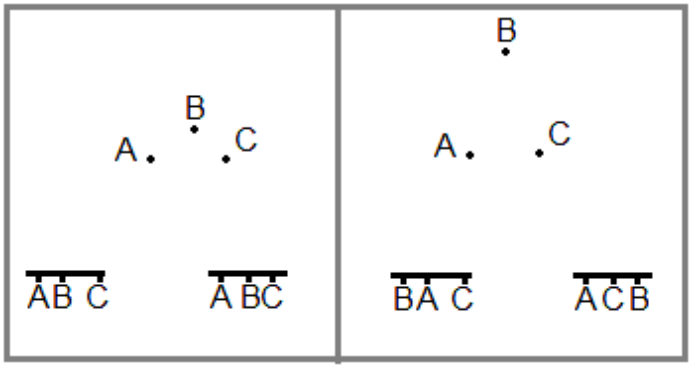
-ترتیب

-همواری

محدودیت رنج عدم تطابق: به جای جستجوی کل تصویر (یا کل خط Epipolar) محدوده‌ی مشخصی از آن را جستجو کنیم. مثلا اگر بدانیم حداکثر رنج ۱۰ است فقط ۱۰ پیکسل قبل و ۱۰ پیکسل بعدی را جستجو کنیم. این فرض ممکن است باعث عدم یافتن نقطه‌ی متناظر صحیحی شود که خارج از محدوده جستجوی ما افتاده است.

یکتایی: در این حالت فرض می‌کنیم تناظرها یک به یک هستند. یعنی هر نقطه در تصویر سمت راست فقط یک نقطه متناظر در سمت چپ دارد. این فرض عموما صحیح است به جز مواردی که جسم شفاف رو به روی دوربین قرار دارد.

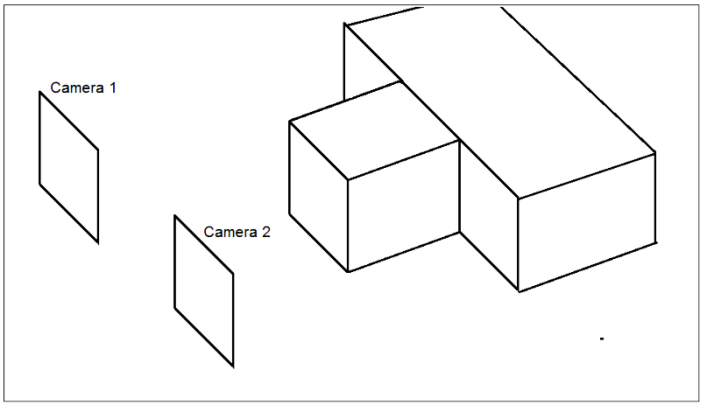
ترتیب: این فرض می‌گوید ترتیب نقاط صحنه در تصویر یکسان است(شکل ۹.الف). این فرض مادامی‌صحیح است که اختلاف عمق بین نقاط فاحش نباشد و گرنه ممکن است نقض شود(شکل۹.ب).



شکل ۹.الف.اختلاف عمق کم، ب. اختلاف عمق زیاد

همواری: شکل ۱۰ را در نظر بگیرید. شکل از زاویه دوربین‌ها به صورت سه اختلاف سطح به نظر می‌رسد. شهودا می‌دانیم عمق تمام نقاط روی هریک از سطح‌ها با هم برابر است. بنابراین عدم تطابق هر نقطه نیز باید با همسایگانش برابر و یا بسیار شبیه به آن باشد. این یعنی تغییرات عدم تطابق هموار و نرم است (به جز در لبه‌ها). اکثر اشیای و صحنه خالی از تغییرات ناگهانی در سطوح خود هستند بنابراین می-

توان این فرض را نیز به مسئله اضافه کرد. اما همانطور که اشاره کردیم این فرض در نقاطی که ناپیوستگی عمق وجود دارد نامعتبر است. یکی از الگوریتم‌های ارائه شده در بخش بعد از این فرض استفاده می‌کند.



شکل ۱۰. جسم دارای سه سطح

۶.۵.تطبیق بلوکی[1]

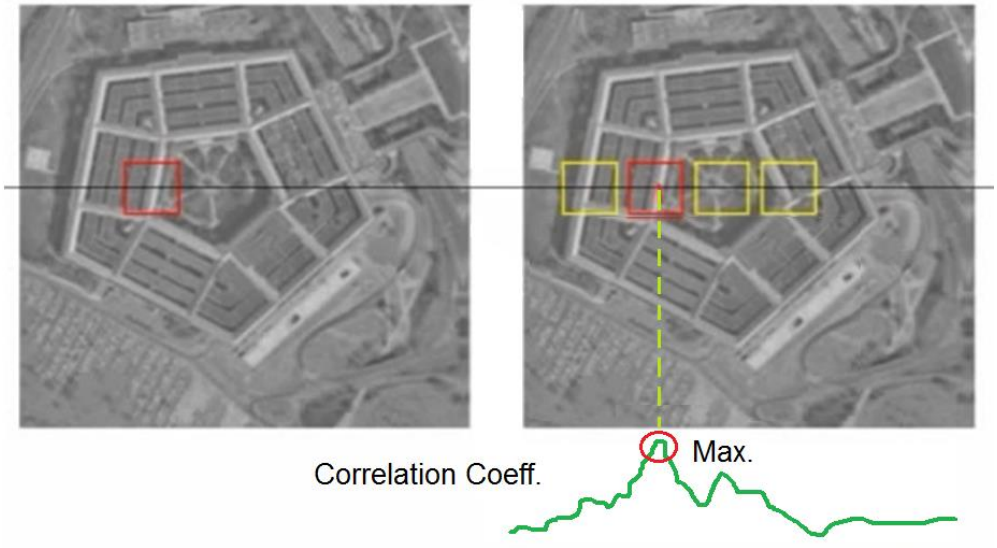
یکی از الگوریتم‌های ساده مبتنی بر همبستگی تطبیق بلوکی است. اساس کار این الگوریتم جدا کردن بلوکی از اطراف نقطه‌ی در تصویر چپ(راست) و جستجو برای بلوک مشابه در تصویر سمت راست(چپ) است. برای معیار شباهت می‌توان هریک از معیارهای ذکر شده در صفحات قبل را استفاده کرد. برای نمونه فرض کنید معیار SSD انتخاب شده است. اندازه بلوک را ۸×۸ بگیرید. اگر حداکثر رنج عدم تطابق در راستهای افقی و عمودی را ۴ بگیریم(فرض دیگری اضافه نمی‌کنیم) با جستجوی هر بلوک در محدوده‌ی معین شده عدم تطابق در راستای x و y را می‌یابیم. به عبارت ریاضی:

(۶.۷)

$$\big(u(x,y),v(x,y)\big) = argmin_{u,v=-4,..,4}\sum_{i=0}^{-7}\sum_{j=0}^{-7}(f_k(x+i,y+i)-f_{k-1}(x+i+u,y+j+v))^2$$

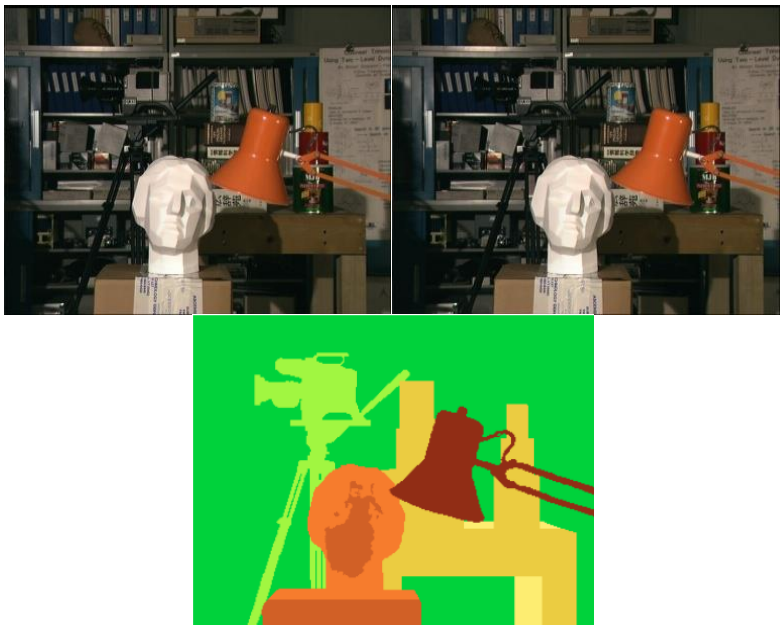
عملیات در شکل زیر همراه با نمودار شباهت به دست آمده نشان داده شده است:

[1] - Block Matching

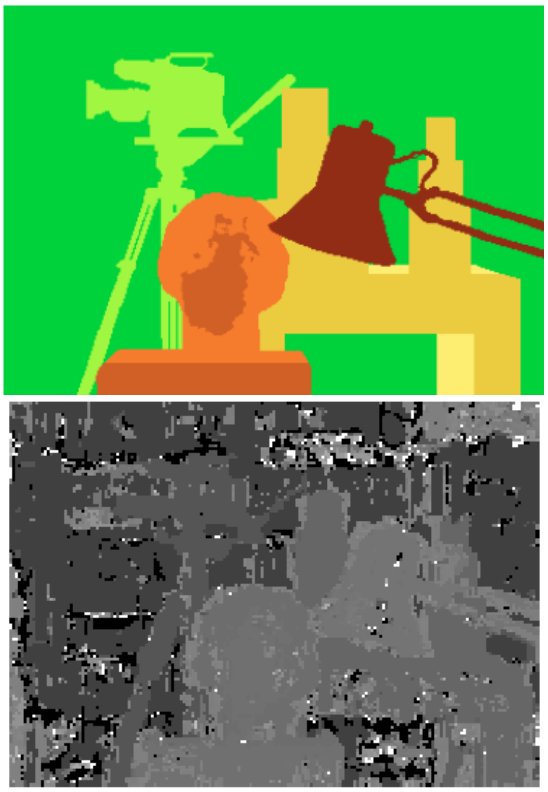


شکل ۱۱. یافتن شبیه‌ترین بلوک در محدوه تعیین شده

تصاویر استفاده شده برای آزمودن الگوریتم در شکل ۱۲.الف و ۱۲.ب نشان داده شده‌اند. تصویر ۱۰.ج نگاشت عدم تطابق واقعی را که با ابزار اندازه‌پیری شده است را نشان می‌دهد. در متون بینایی استریو به این نوع تصاویر Ground Truth می‌گویند.



شکل ۱۲. تصاویر آزمایش الگوریتم،الف.تصویر سمت چپ، ب.تصویر سمت راست،ج. عمق واقعی اندازه‌گیری شده (Ground Truth)



شکل ۱۳. خروجی کد متلب برای جستجوی بلوکی و مقایسه با Ground Truth

نکته‌ای که اینجا باید راجع به اندازه بلوک متذکر شد این است که اندازه کوچک بلوک می‌تواند محاسبات را کمتر کند و همچنین احتمال آنکه بلوک روی نقاط انفصال عمق باشد نیز کمتر می‌شود ولی بلوک کوچک احتمال ابهام را بیشتر می‌کند. از طرفی بلوک‌های بزرگ یکتاتر هستند اما هزینه محاسباتی بالاتری را نیز تحمیل می‌کنند.

۶.۶.الگوریتم برنارد (Barnard)

برای الگوریتم برنارد بهتر است خطوط Epipolar را به دست آوریم و تصاویر را یکسو(rectify) کنیم. ابتدا یک تابع خطا که متغیر آن میزان عدم تطابق در هر نقطه است تعریف می‌کنیم. در تعریف این تابع فرض شده شدت پیکسل‌های متناظر در تصاویر چپ و راست یکسان است و نیز محدودکننده‌ی همواری نیز اعمال شده است(فرض شده عدم تطابق پیکسل‌های همسایه باید یکسان شوند). تابع به صورت زیر تعریف می‌شود:

(۶.۸)

$$E = \sum_{i=-1}^{1}\sum_{j=-1}^{1} \left\| I_{left}(x+i, y+j) - I_{right}(x+i+D_x(x,y), y+j) \right\| + \lambda \left\| \nabla D(x,y) \right\|$$

در عبارات بالا λ ضریب ثابت است، و $D_y(x,y)$ و $D_x(x,y)$ به ترتیب عدم تطابق افقی و عمودی در نقطه‌ی (x,y) هستند. ترم اول میزان عدم شباهت بین دو بلوک از هر تصویر را را اندازه می‌گیرد و ترم دوم در واقع اعمال کننده‌ی فرض همواری است و وقتی مینیمم است که $D(x,y)$ مشابه همسایه‌هایش باشد و طبق رابطه زیر محاسبه می‌شود:

$$\nabla D(x,y) = \sum_{i=-1}^{1}\sum_{j=-1}^{1} |D(x+i, y+j) - D(x,y)| \qquad (۶.۹)$$

با یافتن مینیمم این تابع عدم تطابق‌ها به دست می‌آید.

یک راه حل ساده می‌تواند قرار دادن تک‌تک حالات D در تابع و یافتن حالتی باشد که حداقل خطا را تولید می‌کند. از نظر پیاده‌سازی این کار غیرممکن است. زیرا این یک مسئله‌ی ترکیبیاتی با تعداد حالات بسیار بسیار زیاد است. برای داشتن یک درک شهودی از میزان محاسبات لازم، فرض کنید تصاویر ۱۲۸×۱۲۸ باشند و حداکثر رنج عدم تطابق حداکثر ۱۰ باشد، در این صورت تعداد حالات ممکن برای ماتریس عدم تطابق $۱۰^{۱۶۳۸۴}$ حالت می‌شود! که عملا Brute Forcing (امتحان تک تک حالات) را غیرممکن می‌کند.

راه حل دیگر این مسئله استفاده از تکنیک‌های جستجوی هوشمند(مثل الگوریتم ژنتیک، PSO، تبرید شبیه‌سازی شده و..) است. برنارد الگوریتم تبرید شبیه‌سازی شده (Simulated Annealing) را برای حل این مسئله برگزید. الگوریتم برنارد به صورت زیر است.

۱. یک حالت تصادفی S را انتخاب کنید.

۲. دمای بالای T را انتخاب کنید.

۳. تا زمانیکه T>0

الف) s'را انتخاب کنید.

$$\Delta E \leftarrow E(s') - E(S)$$

ب) اگر $\Delta E \leq 0$ آنگاه $s \leftarrow s'$

ج) وگرنه $P \leftarrow exp^{\frac{-\Delta E}{T}}$ و $X \leftarrow rand(0,1)$ و

اگر $X < P$ آنگاه $S \leftarrow S'$

د) اگر در چندین تکرار مقدار E کاهش نیافت دما، T را کاهش دهید.

گام اول: فرض کنید یک پنجره ۳×۳ حول نقطه (x,y) داریم. معادل همین پنجره در موقعیت x,y از ماتریس D را درنظر بگیرید. اعداد تصادفی در رنج مشخصی به عنوان عدم تطابق در این ماتریس قرار می‌دهیم. در گام بعدی بعدی نیز یک ماتریس ۳×۳ تصادفی به همین صورت می‌سازیم. هردو را تبق رابطه ۸ در تابع خطا قرار می‌دهیم و اختلاف خطاها را محاسبه می‌کنیم. اگر خطا در حالت دوم کمتر شد ($\Delta E<0$) حالات را عوض کرده و به گام ۲ می‌رویم وگرنه ($\Delta E > 0$) دو عدد تصادفی(P و X) انتخاب می‌کنیم، یکی برحسب اختلاف خطای ایجاد شده و دمای فعلی و دیگری به صورت کاملا تصادفی تولید می‌شوند. اگر عدد تصادفی اولی بزرگتر از دومی‌شد حالات را عوض کنیم (هرچند خطا بیشتر می‌شود) و گرنه همین حالت را حفظ کرده و به گام دوم بازمی‌گردیم. این کار را برای جلوگیری از گیر کردن در مینیمم‌های محلی انجام می‌دهیم. وقتی دما بالا است احتمال تغییر حالت نیز بالاست (P بزرگ به دست می‌آید) و با ادامه الگوریتم و کاهش دما این‌گونه تغییر حالات کمتر رخ می‌دهند. شرط سوم می‌گوید در صورتی که خطای ایجاد شده چند بار پشت سر هم کاهش نیافت دما باید کاهش یابد.

تمرینات

۱. رابطه‌ی ۶.۲ را استخراج کنید.

۲. با استفاده از روش تطبیق بلاکی عمق نقطه‌ی سایه زده شده را بیابید(پنجره‌ی تطابق را ۳×۳ در نظر بگیرید).

۹	۱۲	۱۰	۲۳	٥٤	۲۲
۲۱	۲۱	۳۲	۲۳	۳۳	۲۳
۲۳	۱۳	۹	٦۷	٥٤	٤۳
۲۳	۱٤	۲۳	٤٥	۸۷	۸۳
۱۲	۱٥	۲۳	۲۱	٤٥	۲۳
٥٤	۱٥	۱۲	۳۲	۲۳	٥٦

۱۲	۲۲	٥٤	۲۳	۱۰	۱۲
۱۳	٤۳	٥٤	٦۷	۹	۱۳
۱٤	۸۳	۸۷	٤٥	۲۳	۱٤
۱٥	۲۳	٤٥	۲۱	۲۳	۱٥
۱٦	٥٦	۲۳	۳۲	۱۲	۱٥
۱۷	۸۷	٥٤	۳٤	٥٤	۱٦

۳. الگوریتم برنارد و روابط مربوطه را با تابع هزینه‌ی دیگری بنویسید.

پیوست

برنامه متلب الگوریتم برنارد به صورت زیر ارائه شده است.

```
clear;clc
imright=(imread('right1.jpg'));
imleft=(imread('left1.jpg'));
```

```
MaxDisparity=200;
T=100;  % Tempreture of simulated annealing
if size(imright,3)>1
imright=rgb2gray(imright);
imleft=rgb2gray(imleft);
end
imright=double(imright);
imleft=double(imleft);
[M N]=size(imleft);
%S=randint(M,N,[-MaxDisparity 0]);
dcrzNum=0;
E1=.1;
E2=E1;
i=1;
D(M,N)=0;
Lambda=10;
for x=20:M-20
for y=20:N-20
S=randint(1,1,[-MaxDisparity MaxDisparity]);
T=100;
while T>0
E1previos=E1;
e1=0;
smooth=0;
D(x,y)=S;
if y+D(x,y)-1<=0 || y+D(x,y)+1>=N
D(x,y)=0;     %ignore out of the boundry
end
for i=-1:1
for j=-1:1
e1=e1+abs(imleft(x+i,y+j)-imright(x+i,y+j+D(x,y)));
smooth=smooth+abs(D(x+i,y+j)-D(x,y));
end
end
E1=e1+Lambda*smooth;
if (E1>=E1previos) % if no decrese in E
dcrzNum=dcrzNum+1;  % counts number of5 times E does not decrease
end
Sp=randint(1,1,[-MaxDisparity MaxDisparity]);
if y+Sp-1<=0 || y+Sp+1>=N
Sp=0;     %ignore out of boundary
end
e2=0;
smooth2=0;
for i=-1:1
for j=-1:1
e2=e2+abs(imleft(x+i,y+j)-imright(x+i,y+j+Sp));
smooth2=smooth2+abs(D(x+i,y+j)-Sp);
end
end
```

```
E2=e2+Lambda*smooth2;
DELTAE=E2-E1;
if DELTAE<=0
S=Sp;
else
P=exp(double(-DELTAE/(T)));X=rand;
if X<P
S=Sp;
end
end
if dcrzNum>2
T=T-5;
dcrzNum=0;
end
end
end
disp('******-Row Num:-********');
disp(x);
disp('************************');
end
Deps=1./D;
imshow(Deps,[]);
```

فصل ۷

نقاط ویژگی

بخش یک گوشه‌ها

۱.۷ مقدمه

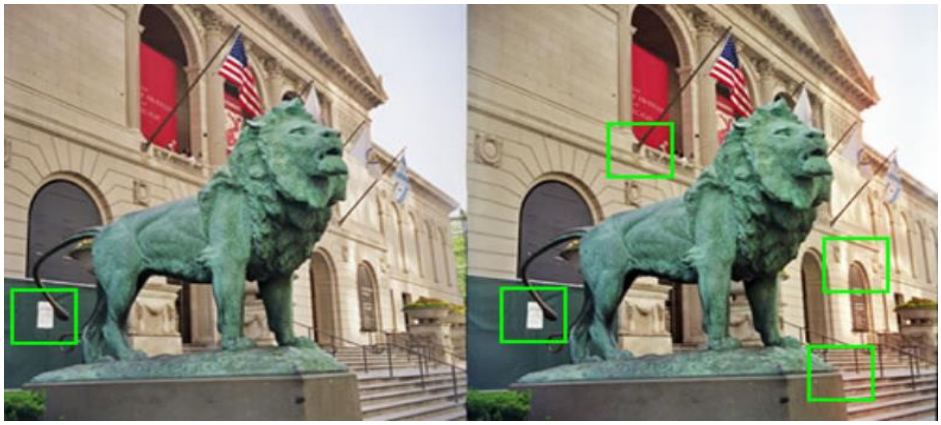
در این بخش ابتدا ویژگیهای مناسب قابل استخراج از تصویر مورد بحث و بررسی قرار می‌گیرد. در ادامه آشکارساز گوشه هریس معرفی می‌شود. سپس چگونگی یافتن پنجره‌های جالب در آشکارساز گوشه هریس معرفی می‌شود و آشکارساز گوشه شای-توماسی به عنوان روش بهبود یافته آشکارساز هریس ارایه می‌گردد. در نهایت تابع آشکارساز گوشه در OpenCV ارایه می‌گردد.

ویژگی‌ها چه چیزی هستند؟

تعدادی از کارهای بینایی کامپیوتر نیاز به یافتن نقاط تطبیق در چند فریم یا نما دارند. در واقع با این نقاط تطبیق شما می‌توانید کارهای زیادی انجام دهید. به عنوان نمونه، به هنگام تصویربرداری استریو شما نیاز به اطلاع از چند نقطه معادل بین دو نما دارید.

روش اول: تکه‌ها

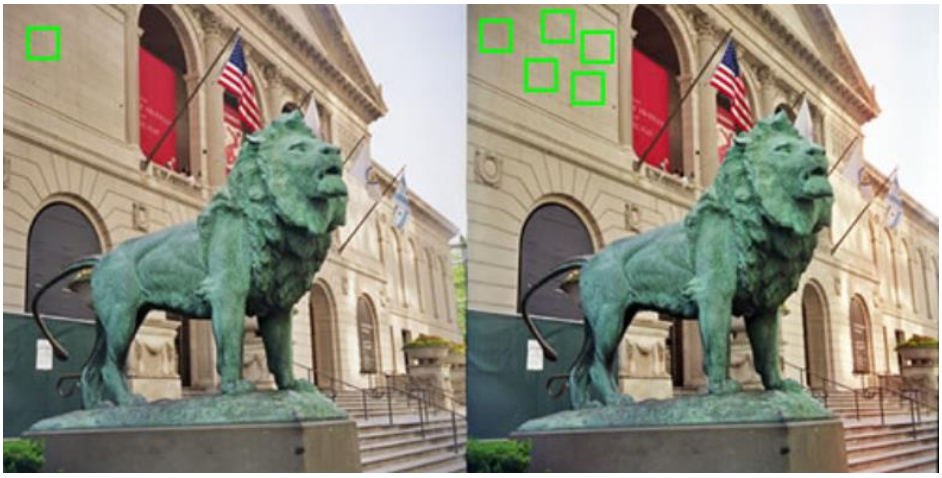
به صورت شهودی شما تمایل به تطبیق تکه‌ها کوچک بین دو تصویر دارید. شکل ۱ را ملاحظه کنید:



شکل ۱: یافتن معادل تکه سبز رنگ شکل سمت چپ در شکل سمت راست

شما می‌خواهید تکه سبز رنگ تصویر چپ را در تصویر راست پیدا کنید. شما می‌توانید این کار را به آسانی انجام دهید. بخش سفید رنگ سبب می‌شود تا تکه کاملا یکتا باشد. حتی تکنیک ساده تطبیق الگو می‌تواند چنین بخشی را پیدا کند.

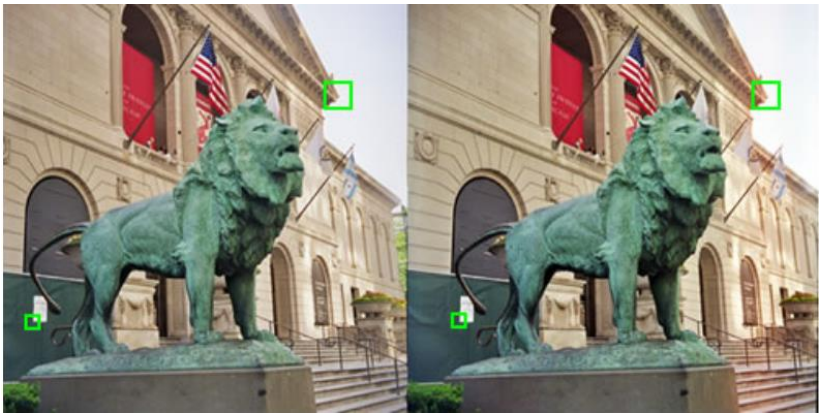
اما تمام تکه‌ها چنین قابل تشخیص به صورت یکتا نیستند. تکه‌های شکل ۲ را در نظر بگیرید:



شکل ۲: یافتن معادل تکه سبز رنگ شکل سمت چپ در شکل سمت راست

این تکه‌ها ویژگیها یکتا برای شناسایی روی دیوار نیستند. پس شما در یافتن نقاط معادل مشکل خواهید داشت. پس روش تکه‌ها زیاد عالی نیست.

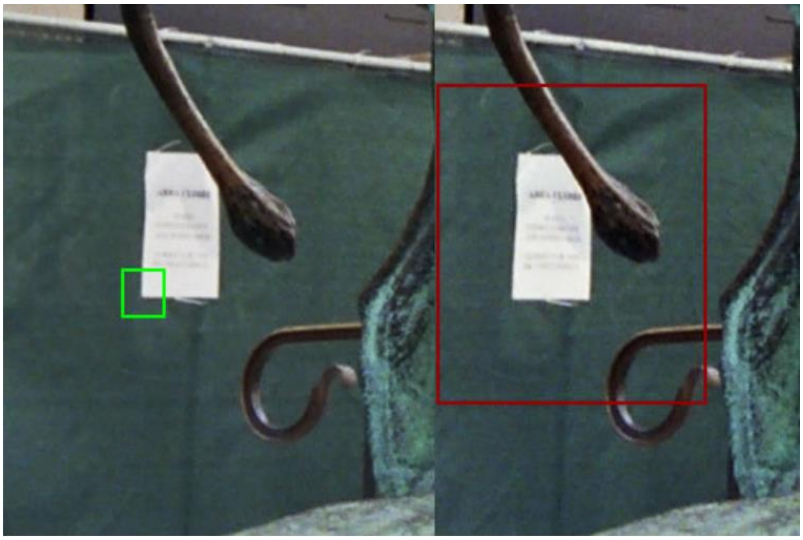
روش دوم: گوشه‌ها



شکل ۳: یافتن معادل تکه‌های سبز رنگ شکل سمت چپ در شکل سمت راست به آسانی صورت می‌گیرد

این گوشه‌ها کامل هستند. دلیل این واقعیت در ادامه مشخص می‌شود.

این نقاط به صورت یکتا قابل شناسایی هستند. به منظور روشن شدن این موضوع فرض کنید که شما تلاش می‌کنید گوشه سبز رنگ در تصویر سمت راست (شکل ۴) بیابید. شما می‌دانید که باید جایی اطراف همان محل در تصویر سمت راست شکل ۳ باشد. پس شما منطقه جستجو را باریک می‌کنید و داخل این ناحیه جستجو تنها یک نقطه است که شبیه گوشه است. البته فرض ما بر عدم اختلاف زیاد بین دو تصویر بنا نهاده شده است و معمولا این فرض معقولی است.

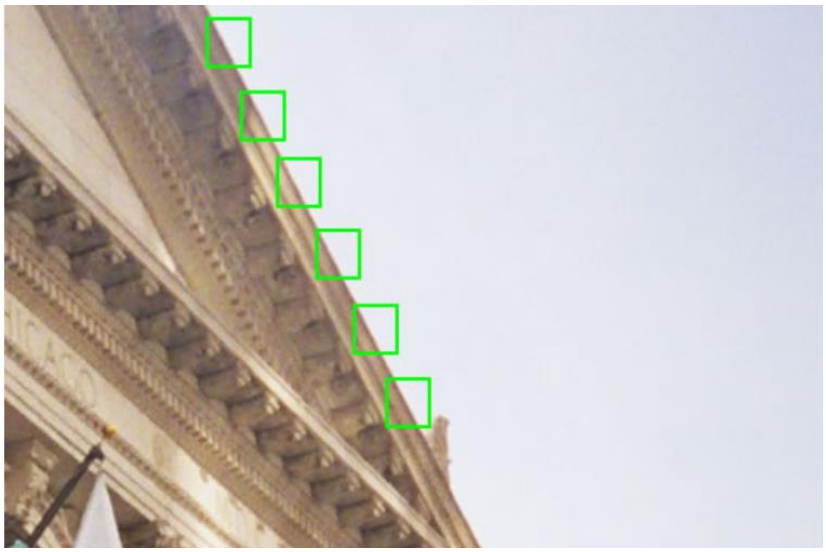


شکل ۴: بزرگنمایی تکه‌هایی از شکل ۳ (هدف یافتن معادل تکه سبز رنگ در شکل سمت چپ در پنجره قرمز رنگ سمت راست

این نقاط در اطراف تصویر حرکت نمی‌کنند و این امر به ردیابی کمک می‌کند و هر حرکت حتی بسیار کوچک این نقطه تغییرات بزرگی ایجاد می‌کند. شما این موضوع را به روشنی با حرکت اطراف آن نقطه ملاحظه می‌کنید.

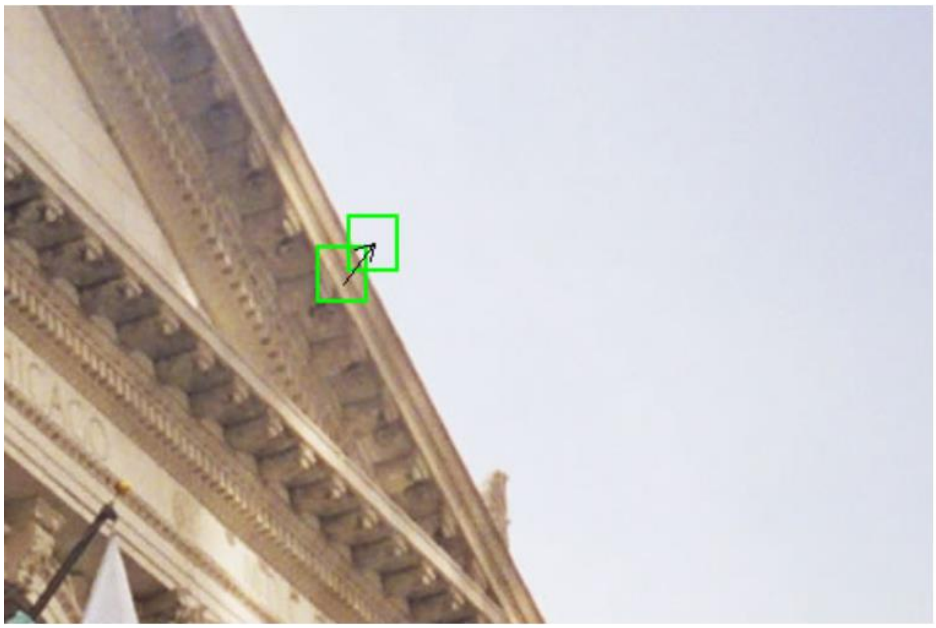
در ادامه سعی می‌کنم تا ایده گوشه را بیشتر ملموس کنم. ما این کار را به کمک ریاضیات انجام می‌دهید.شما چگونه یک ویژگی بد را مشخص می‌کنید؟ چیزی که تغییرات زیادی ندارد شما مثال دیوار بالا. در این ویژگی لبه‌ها یا گوشه‌ها نیستند. پس مشتق اول در دو جهت x و y مسطح است. یا مشتق اول در تمام جهتها مسطح است (تمام جهتها ترکیب معینی از جز x و جز y هستند).پس مشتق در هیچ جهتی تغییر نمی‌کند.

یک لبه نیز یک ویژگی بد است. اگر شما در جهت لبه حرکت کنید شما حتی متوجه حرکت تان نمی‌شوید. به عنوان نمونه، اگر در امتداد لبه در بالای یک ساختمان و آسمان حرکت کنید شما متوجه حرکت نخواهید شد (شکل ۵).



شکل ۵: حرکت در راستای لبه غیر قابل کمی‌شدن است (تکه‌های سبز رنگ در تصویر غیر قابل تمایز هستند)

اما اگر شما از ساختمان به آسمان (عمود به لبه) حرکت کنید حرکت را خواهید دید (شکل ۶). این تنها جهتی است که شما می‌توانید به دقت بگویید که شی به چه سرعتی حرکت می‌کند.



شکل ۶: تنها حرکت در راستای عمود بر لبه قابل کمی‌شدن است

پس لبه‌ها به عنوان ویژگی مفید نیستند. مشتق اول تنها در یک جهت تغییر می‌کند (عمود به لبه). مقداری پیشرفت صورت گرفته است اما نه زیاد. پس مشتق در یک جهت تغییر می‌کند.

یک گوشه ویژگی عالی است که در آن تغییرات در تمام جهات صورت می‌گیرد که سبب می‌شود تغییرات مشتق در تمام جهتها صورت گیرد. پس مشتق در تمام جهتها تغییر می‌کند و می‌توان برنامه موثری نوشت که در تمام نقاط مشتق را محاسبه کند.

پس بالاخره به ویژگی خوب رسیدیم. اگر مشتق اول اطراف یک نقطه تغییر کرد شما متوجه می‌شوید که یک گوشه دارید و شما می دانید که ویژگی خوبی برای ردیابی داریم.

۲.۷ آشکار ساز گوشه هریس

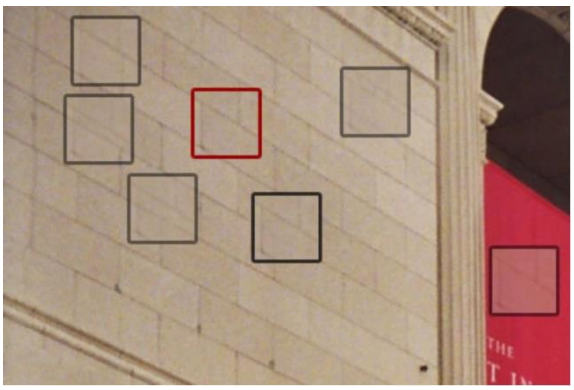
آشکارساز گوشه هریس اپراتور ریاضی است که در یک تصویر ویژگیها را پیدا می‌کند. اینکه این ویژگیها کدامند در ابتدای فصل مورد بررسی قرار گرفت. این آشکارساز به صورت ساده‌ای محاسبه می‌شود و به

اندازه کافی برای کار بر روی کامپیوتر سریع است. همچنین به دلیل ویژگیهای نابسته چرخش، نابسته مقیاس، و مستقل از تغییرات روشنایی محبوب است. اما آشکارساز گوشه شای-توماسی تحقق یافته در OpenCV بهبود یافته آشکارساز گوشه هریس است.

ریاضیات

در اینجا جهت تعریف آشکارساز گوشه هریس مطالب ریاضی آورده شده است. ما مقداری حسابان، ریاضیات ماتریسی ارایه می‌دهیم که زیاد سخت و دشوار نیستند.

هدف ما در آشکارساز گوشه هریس یافتن تکه‌های (پنجره‌های) کوچکی از تصویر است که تغییرات بزرگی در حرکت به اطراف تولید می‌کنند. به تصویر شکل ۷ نگاه کنید.

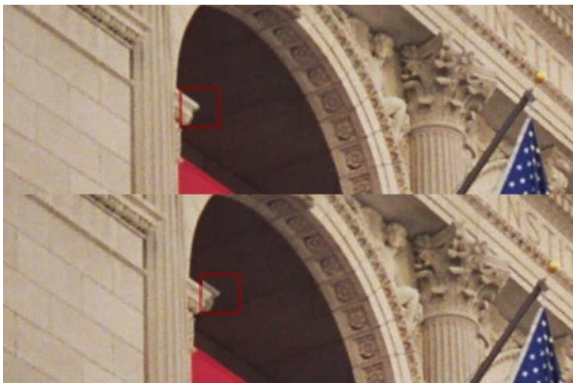


شکل ۷: مناطق علامت خورده تغییرات کمی‌دارند

مربع قرمز پنجره‌ای است که ما انتخاب کرده ایم. حرکت دادن آن به اطراف تغییرات زیادی نشان نمی‌دهد. این بدان معنی است که اختلاف بین پنجره و تصویر اولیه داخل آن خیلی کم است. بنابراین شما نمی‌توانید واقعا بگویید آیا پنجره به آن موقعیت متعلق است یا نه.

البته اگر شما پنجره را زیاد حرکت دهید به عنوان نمونه به ناحیه قرمز زنگ، شما اختلاف بزرگی را خواهید دید. اما ما پنجره را خیلی زیاد حرکت داده ایم و این خوب نیست.

حالا به تصویر شکل ۸ نگاه کنید.



شکل ۸: مناطق علامت خورده تغییرات زیادی دارند

مشاهده می‌شود که حتی با حرکت کم پنجره اختلاف قابل توجهی تولید می‌شود. ما بدنبال یافتن چنین پنجره‌ای هستیم. این خواسته به صورت ریاضی با فرمول زیر بیان می‌گردد:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u\,,y+v) - I(x,y)]^2 \qquad (۷.۱)$$

که در آن E اختلاف بین پنجره اولیه و پنجره حرکت یافته است.

u جابجایی پنجره در جهت x است.

v جابجایی پنجره در جهت y است.

$W(x,y)$ پنجره در موقعیت (x,y) است. این درست مانند یک ماسک عمل می‌کند که تضمین می‌کند تنها پنجره مورد نظر به کار می‌رود.

I شدت تصویر در موقعیت (x,y) است.

$I(x+u,y+v)$ شدت در پنجره حرکت یافته است.

$I(x,y)$ شدت تصویر اولیه است.

ما در جستجوی پنجره‌ای هستیم که مقدار بزرگی از E تولید کند. بدین منظور به مقادیر بزرگ داخل کروشه نیاز داریم.

پس عبارت زیر را بیشینه می‌کنیم

$$\sum_{x,y}[I(x+u\,,y+v)-I(x,y)]^2 \qquad (۲.۷)$$

آنگاه این عبارت را با به کارگیری سری تیلور بسط می‌دهیم. این بدان معنی است که عبارت بالا را با به کارگیری مشتقات به صورت زیر بازنویسی کنیم:

$$E(u,v)\approx \sum_{x,y}[I\big(x,y+uI_x+vI_y-I(x,y)\big]^2 \qquad (۳.۷)$$

دقت کنید که چگونه $I(x+u,y+y)$ به شکل کاملا متفاوت $I(x,y)+uI_x+vI_y$ تغییر یافته است. در ضمن توجه کنید که بسط سری تیلور حول (x,y)تعداد بی نهایت جمله تولید می‌کند که ما تنها سه عبارت اول را در نظر گرفته ایم و بقیه عبارتها را حذف کرده ایم. این سه عبارت با دقت خوبی شدت را در اطراف نقطه (x,y) تقریب می‌زنند که البته مقداری واقعی نیست.

در ادامه، عبارت داخل کروشه به توان دو می‌رسد. البته چون$\mathrm{I(x,y)}$ حذف می‌شود تنها دو عبارت داخل کروشه به توان دو خواهند رسید. بعد از عملیات مجذور فرمول بالا به صورت زیر در می‌آید:

$$E(u,v)\approx \sum_{x,y}u^2I_x^2+2uvI_xI_y+v^2I_y^2 \qquad (۴.۷)$$

حال این معادله نسبتا بهم ریخته به شکل ماتریسی زیر مرتب می‌شود:

$$E(u,v)\approx [u \quad v]\left(\sum\begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix}\right)\begin{bmatrix}u\\ v\end{bmatrix} \qquad (۴.۷)$$

ملاحظه می‌کنید که چطور تمام معادله به شکل ماتریس مرتب تبدیل شده است!

با افزودن w(x,y) به بخش ماتریس‌های جمع شده بالا، جمع ماتریسی M به صورت زیر مشخص می‌شود:

$$M = \sum w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad (۶.۷)$$

حالا معادله نهایی به صورت زیر نوشته می‌شود:

$$E(u,v) \approx [u \quad v]\, M \begin{bmatrix} u \\ v \end{bmatrix} \quad (۷.۷)$$

مشخص شده است که مقادیر ویژه ماتریس M در یافتن مناسب بودن یک پنجره می‌تواند کمک کننده باشد. امتیاز R برای هر پنجره محاسبه می‌شود:

$$R = \det M - k(trace\ M)^2$$
$$\det M = \lambda_1 \lambda_2$$
$$trace\ M = \lambda_1 + \lambda_2$$

تمام پنجره‌هایی که امتیاز R بزرگتر از مقدار معینی دارند به عنوان گوشه در نظر گرفته می‌شوند. این نقاط برای ردیابی مناسب هستند.

آشکارساز گوشه هریس تنها روش ریاضی برای تعیین کردن پنجره‌ای است که تغییرات بزرگی در هر حرکت در هر جهتی دارد. با هر پنجره یک امتیاز R مرتبط است که بر پایه این امتیاز شما می‌توانید تعیین کنید که کدام پنجره‌ها گوشه هستند و کدام پنجره‌ها گوشه نیستند.

در OpenCV نسخه بهبود یافته آشکارساز گوشه هریس پیاده سازی شده است که به آشکارساز گوشه شای-توماسی معروف است.

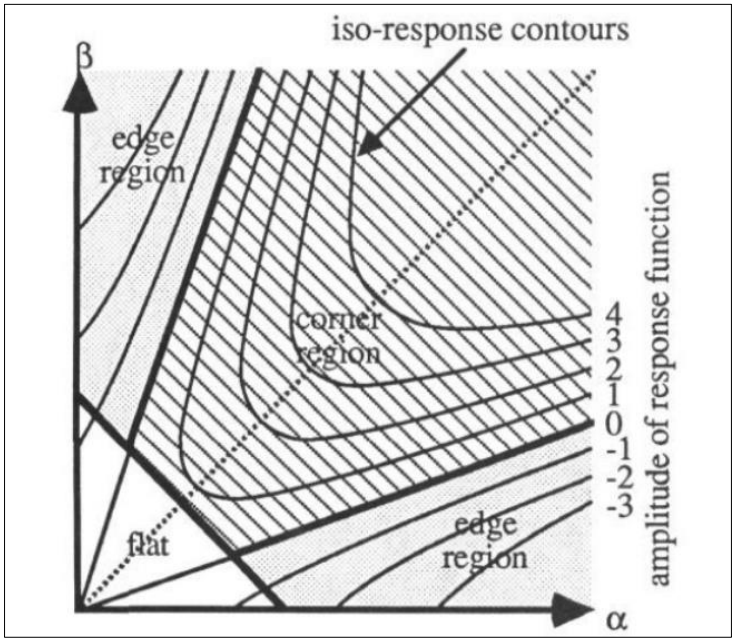
۳.۷ یافتن پنجره‌های جالب در آشکارساز گوشه هریس

امتیازی که برای هر پیکسل در آشکارساز گوشه هریس محاسبه می‌شود تابعی از دو مقدار ویژه ماتریس M است. توجه داشته باشید تابع مربوطه دلخواه نیست بلکه بر اساس مشاهدات چگونگی تغییر عبارت با مقادیر ویژه مختلف است. در اینجا توضیح گرافیکی کارکرد این تابع آورده شده است.

ایده اصلی انتخاب گوشه بر اساس عبارت امتیاز زیر است:

$$R = \det M - k(trace\ M)^2$$
$$\det M = \lambda_1\lambda_2$$
$$trace\ M = \lambda_1 + \lambda_2$$

ما نیاز به تصمیم گیری بر روی مقادیر R داریم تا یک پیکسل را به عنوان گوشه در نظر بگیریم. در ادامه شکل زیر را در نظر بگیرید:



شکل ۹: افراز فضای دو مقدار ویژه به مسطح، لبه، و گوشه در آشکارساز گوشه هریس

در شکل بالا α و β دو مقدار ویژه هستند. تصمیم گیری بر روی گوشه بودن یا گوشه نبودن یک پیکسل به صورت زیر گرفته می‌شود:

هر دو مقدار ویژه کوچک هستند آنگاه پیکسل مسطح است (ناحیه سفید).

یک مقدار ویژه بزرگ است و مقدار ویژه دیگر کوچک است آنگاه پیکسل یک لبه است (ناحیه خاکستری).

هر دو مقدار ویژه بزرگ هستند آنگاه پیکسل گوشه است (ناحیه‌هاشور خورده).

همچنین شکل کانتورهای تابع امتیاز R را نشان می‌دهد. شما در انتخاب یک مقدار مناسب، مقادیر مثبت در ناحیه گوشه حاصل می‌شود و مقادیر منفی در جاهای دیگر ظاهر می‌شوند.

در ادامه سال ۱۹۹۴ شای-توماسی طرح بهتری برای آشکارساز گوشه ارایه دادند. کار آنها تنها با تغییر کوچک در آشکارساز گوشه هریس حاصل شد در حالی که نتایج بهتری در آشکارسازی گوشه تولید نمود.

در حقیقت، آشکار ساز شای-توماسی در OpenCV تحقق داده شده است.

۴.۷ آشکارساز گوشه شای-توماسی

آشکارساز گوشه شای-توماسی کاملا بر اساس آشکار ساز گوشه هریس است. اما یک تغییر کوچک در معیار انتخاب سبب شده است تا این آشکارساز بهتر از آشکارساز اولیه باشد. آشکارساز جدید کاملا خوب کار می‌کند حتی جای که آشکارساز گوشه هریس موفق نیست.

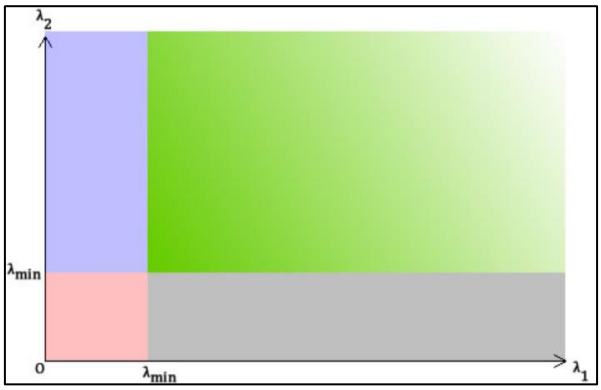
آشکارساز گوشه هریس یک معیار انتخاب گوشه دارد. یک امتیاز برای هر پیکسل محاسبه می‌شود و اگر این امتیاز بالای یک مقدار معینی باشد آن پیکسل به عنوان گوشه در نظر گرفته می‌شود. این امتیاز با به

کارگیری دو مقدار ویژه محاسبه می‌شود. بدین معنی که شما دو مقدار ویژه را به تابعی می‌دهید و تابع دو مقدار ویژه را پردازش می‌کند و امتیاز را بر می‌گرداند.

شای و توماسی پیشنهاد کردند که به جای محاسبه امتیاز R از معادله متعارف بالا، امتیاز از معادله زیر محاسبه شود:

$$R = \min(\lambda_1, \lambda_2) \qquad (۸.۷)$$

شای و توماسی در مقاله شان با آزمایش نشان دادند که این معیار امتیاز بهتر است. اگر R بزرگتر اس یک مقدار از قبل تعیین شده باشد آن پیکسل به عنوان گوشه علامت زده می‌شود. بنابراین ناحیه موثر برای یک نقطه گوشه شبیه شکل ۱۰ خواهد بود:



شکل ۱۰: افراز فضای دو مقدار ویژگی به مسطح، لبه، و گوشه در آشکارساز گوشه شای-توماسی

که نواحی زنگی به صورت زیر مشخص می‌شوند:

ناحیه سبز: هر دو مقدار ویژه بزرگتر از یک مقدار معین هستند. بنابراین این ناحیه برای پیکسلها به عنوان گوشه‌ها مورد پذیریش قرار می‌گیرد.

نواحی آبی و خاکستری: تنها یکی از مقدارهای ویژه کمتر از کمینه مورد نیاز است.

ناحیه قرمز: هر دو مقدار ویژه کمتر از کمینه مورد نیاز هستند. در مقایسه شکل ۱۰ با شکل مشابه برای آشکارساز گوشه هریس (شکل ۹) متوجه می‌شویم که نواحی آبی و خاکستری معادل با نواحی لبه هستند و ناحیه قرمز معادل با نواحی مسطح است و ناحیه سبز معادل با گوشه‌ها است.

۵.۷ آشکارساز گوشه در OpenCV

آشکارساز گوشه هریس و آشکارساز گوشه شای-توماسی در OpenCV پیاده سازی شده است. اما OpenCV به صورت پیش فرض آشکارساز شای-توماسی به کار می‌برد مگر اینکه به صورت صریح بخواهید که از آشکارساز گوشه هریس استفاده کنید. همچنین شما می‌توانید در صورت تمایل مقادیر ویژه خام و بردارهای ویژه برای هر پیکسل را به دست آورید و مقادیر را خودتان پردازش کنید.در ادامه توابع از پیش تعیین شده مورد بحث قرار می‌گیرند.

"ویژگیهای خوب برای ردیابی" نام تابعی است که پارامترهای آن به صورت زیر است:

```
void cvGoodFeaturesToTrack(const CvArr* image,
                           CvArr* eig_image,
                           CvArr* temp_image,
                           CvPoint2D32f* corners,
                           int* corner_count,
                           double quality_level,
                           double min_distance,
                           const CvArr* mask=NULL,
                           int block_size=3,
                           int use_harris=0,
                           double k=0.04);
```

image کاملا اساسی است. این تصویری است که شما می‌خواهید در آن گوشه‌ها را آشکار کنید. این تصویر باید سطح خاکستری (۸ بیت و تک کاناله) باشد و یا ۳۲ بیت و تک کاناله باشد.

Eig_image: این محل ذخیره موقت است که توسط تابع استفاده می‌شود. مقادیر ویژه به صورت اتوماتیک محاسبه می‌شوند و در این تصویر موقت ذخیره می‌شوند. این تصویر ۳۲ بیتی و تک کاناله است. همچنین همان اندازه تصویر را دارد. داده در این تصویر می‌تواند مفید باشد (هر نقطه مقدار ویژه کمینه مرتبط با پیکسل است).

Temp_image: محل ذخیره موقت دیگری است که توسط تابع به کارگرفته می‌شود. بر خلاف eig_image این تصویر بعد از اجرای تابع استفاده‌ی ندارد.

Corners: آدرس یک آرایه CVPoint2D43f را ارسال می‌کند.تابع تمام ویژگیهای پذیرفته شده را داخل این آرایه قرار می‌دهد.

Corner_count: آدرس یک تصویر را می‌دهد. شما باید این عدد صحیح به بیشینه تعداد عناصر در آرایه گوشه‌ها قبل از فرخوانی تابع تنظیم کنید. بعد از اتمام اجرای تابع مقدار آن به تعداد واقعی نقاط در آرایه نشانده شده است.

Quality_level: اگر شما آشکارساز شای-توماسی در نظر بگیرید نیازمند آن است که کمینه مقدار ویژه باید بالای یک مقدار از قبل تعیین شده باشد. Quality_level برای محاسبه این مقدار به کار می‌رود. آستانه مذکور به دین صورت محاسبه می‌شود:

$$\lambda_{min} = quality\_level * \lambda_{max}$$

در اینجا λ_{min} آستانه و λ_{max} بیشنیه مقدار ویژه در تصویر است. مقدار عادی برای qulity_level بهتر است در بازه ۰.۰۱ الی ۰.۱ باشد و همواره کمتر از یک.

Min_distance: کمینه فاصله بین دو ویژگی است. اگر دو ویژگی یا گوشه نزدیکتر از این فاصله باشند تابع گوشه ضعیفتر را حذف می‌کند.

Mask: همانند image این هم اساسی است. تابع یک پیکسل را در ضورتی پردازش می‌کند که آن در ماسک غیرصفر است.

Block_size: بهتر است که ناحیه کوچکی به جای یک نقطه در نظر گرفته شود. این پارامتر اجازه می‌دهد که اندازه پنجره اطراف پیکسل تنظیم شود.

Use_harris: اختیاری است و به صورت پیش فرض صفر است. این مقدار را به غیر صفر تنظیم کنید در صورتی که می‌خواهید از آشکارساز گوشه هریس استفاده کنید.

K: اختیاری است و به صورت پیش فرض ۰.۰۴ است. اگر تصمیم به استفاده از روش هریس دارید نیاز به ارایه k برای محاسبه امتیاز R برای هر پیکسل دارید.

تابع cvGoodFeaturesToTrack به شما اجازه می‌دهد گوشه‌ها را به آسانی پیدا کنید. پیش فرض تابع روش شای–توماسی را به کار می‌برد اما شما می‌توانید از روش هریس نیز استفاده کنید.

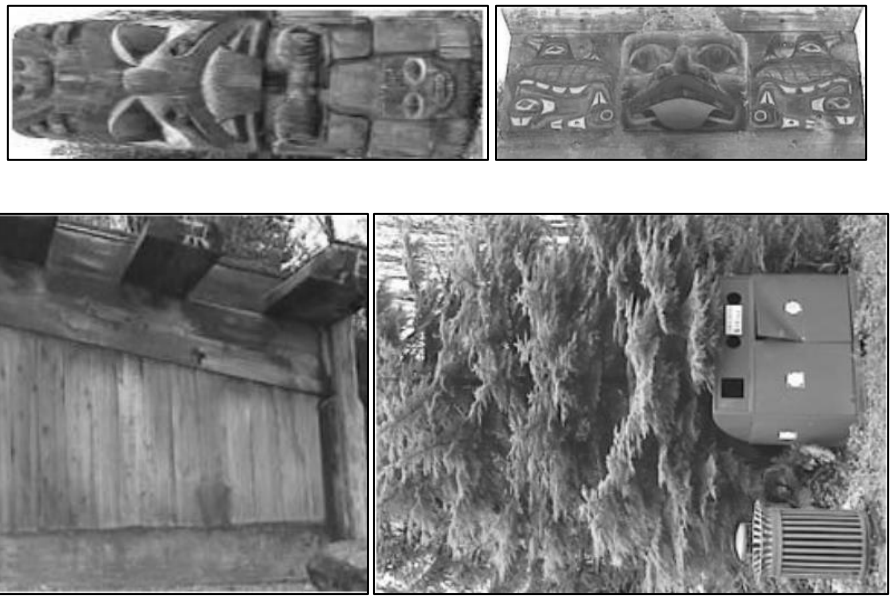
بخش دوم

تبدیل ویژگی نابسته به مقیاس (SIFT)

۵.۷. مقدمه

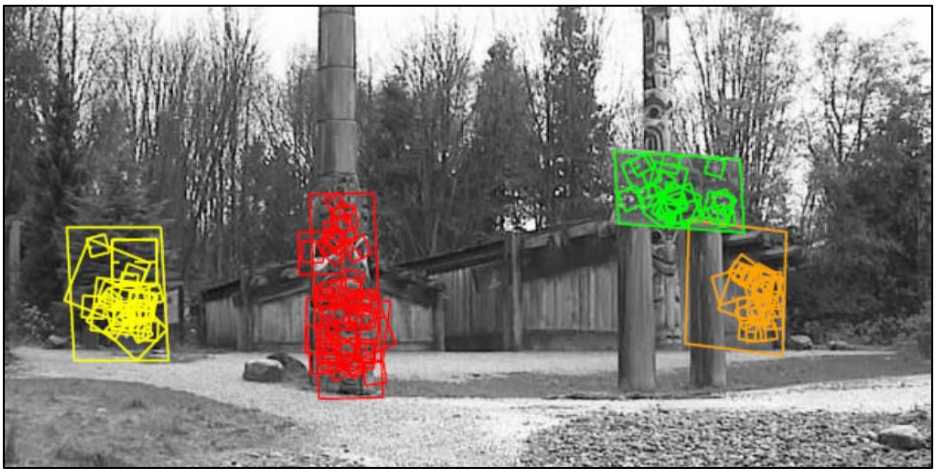
تطبیق ویژگیها مابین تصاویر مختلف مساله عمومی‌در بینایی کامپیوتر است. طبیعتا، هنگامی‌که تمام تصاویر از لحاظ مقیاس، امتداد، و غیره شبیه به همدیگر باشند آشکارسازهای ساده گوشه می‌تواند کارساز باشد. اما، هنگامی‌که تصاویر با مقیاسهای متفاوت و چرخشهای متفاوت داریم به کارگیری تبدیل ویژگی نابسته مقیاس ضروری است.ذکر این نکته ضروری است که اهمیت تبدیل ویژگی نابسته مقیاس تنها به دلیل خاصیت نابسته مقیاس آن نیست. در واقع با تغییرهای اندازه، چرخش، نوربردازی، و نقطه دید نتایج

رضایت بخشی با این تبدیل حاصل می‌شود. به عنوان نمونه ما الگوهای شکل ۱۱ را در شکل ۱۲ جستجو می‌کنیم. نتایج تطبیق الگوهای شکل ۱۱ به بخشهای متناظر شکل ۱۲، در شکل ۱۳ آورده شده است.



شکل ۱۱: تصویر دچار تغییر مقیاس و دوران یافته

شکل ۱۲: تصویر کامل



شکل ۱۳: تطبیق تصاویر تکه ای با تصویر کامل

در شکل ۱۳، مستطیلهای بزرگ تصاویر تطبیق یافته شکل ۱۱ را نشان می‌دهند و مربعهای کوچکتر برای ویژگیهای مجزا حاصله در آن مناطق است. به کجی مستطیلهای بزرگ دقت کنید که نشان دهنده امتداد و پرسپکتیو (نقطه دید) مناطق شکل ۱۱ در صحنه شکل ۱۲ است.

تبدیل ویژگی نابسته مقیاس کاملا الگوریتم پیچیده‌ای است. بنابراین به منظور ساده سازی کل الگوریتم به چند بخش تقسیم می‌شود. در ذیل بخشهای تبدیل ویژگی نابسته مقیاس فهرست شده است:

۱- ساخت یک فضای مقیاس: این مرحله گام نخستین است. شما نمایش داخلی از تصویر اولیه جهت اطمینان از نابسته بودن به مقیاس تولید می‌کنید. این کار با تولید یک فضای مقیاس انجام می‌شود.

۲- تقریب لاپلاسین گوسی: لاپلاسین گوسی برای یافتن نقاط کلیدی در یک تصویر عالی است، اما، به لحاظ محاسباتی هزینه بر است. بنابراین، ما لاپلاسین گوسی را با به کارگیری گوسی‌ها تقریب می‌زنیم!

۳- یافتن نقاط کلیدی: نقاط کلیدی بیشینه‌ها و کمینه‌ها در تصویر اختلاف گوسی محاسبه شده در مرحله ۲ هستند.

۴- حذف نقاط کلیدی نامناسب: لبه‌ها و نواحی با کنتراست پایین جز نقاط کلیدی نا مناسب محسوب می‌شوند. حذف این نقاط کلیدی الگوریتم را موثر و مقاوم می‌کند. در اینجا، تکنیکی مشابه آشکارساز هریس به کار می‌رود.

۵- تخصیص امتداد به نقاط کلیدی: امتداد برای هر نقطه کلیدی محاسبه می‌شود. در ادامه، هر محاسبه دیگری نسبت به این امتداد انجام می‌گیرد. این کار به طور موثری اثر امتداد را از بین می‌برد و سبب می‌شود تا الگوریتم نابسته به چرخش شود.

۶- تولید ویژگی‌های تبدیل ویژگی نابسته مقیاس: در نهایت، با نابسته مقیاس و نابسته چرخش نمایش دیگری تولید می‌شود. این امر کمک می‌کند که ویژگیهای یکسان شناسایی شوند. فرض کنید که ۵۰۰۰۰ ویژگی داریم. با این نمایش، شما به سادگی می‌توانید ویژگی تحت جستجو را شناسایی کنید (به عنوان نمونه، یک چشم خاص، یا یک علامت خاص).

بعد از اجرای الگوریتم بالا، شما ویژگیهای تبدیل ویژگی نابسته مقیاس برای تصویرتان خواهید داشت. در ادامه، شما قادر خواهید بود تا کارهای ردیابی تصویر، آشکارسازی و شناسایی اشیاء حتی با انسداد را انجام دهید. در پایان این بخش لازم به ذکر است که الگوریتم تبدیل ویژگی نابسته مقیاس به عنوان اختراع ثبت شده است. لذا در صورتی که بخواهید محصول تجاری با به کارگیری تبدیل مذکور ارایه کنید بایستی در جستجوی تبدیل دیگری باشید و یا اجازه به کارگیری آن را در محصول خود با صرف مقداری هزینه بگیرید.

در بالا مروری از الگوریتم را ملاحظه کردید. در بخشهای بعدی هر مرحله به تفصیل بیان خواهد شد.

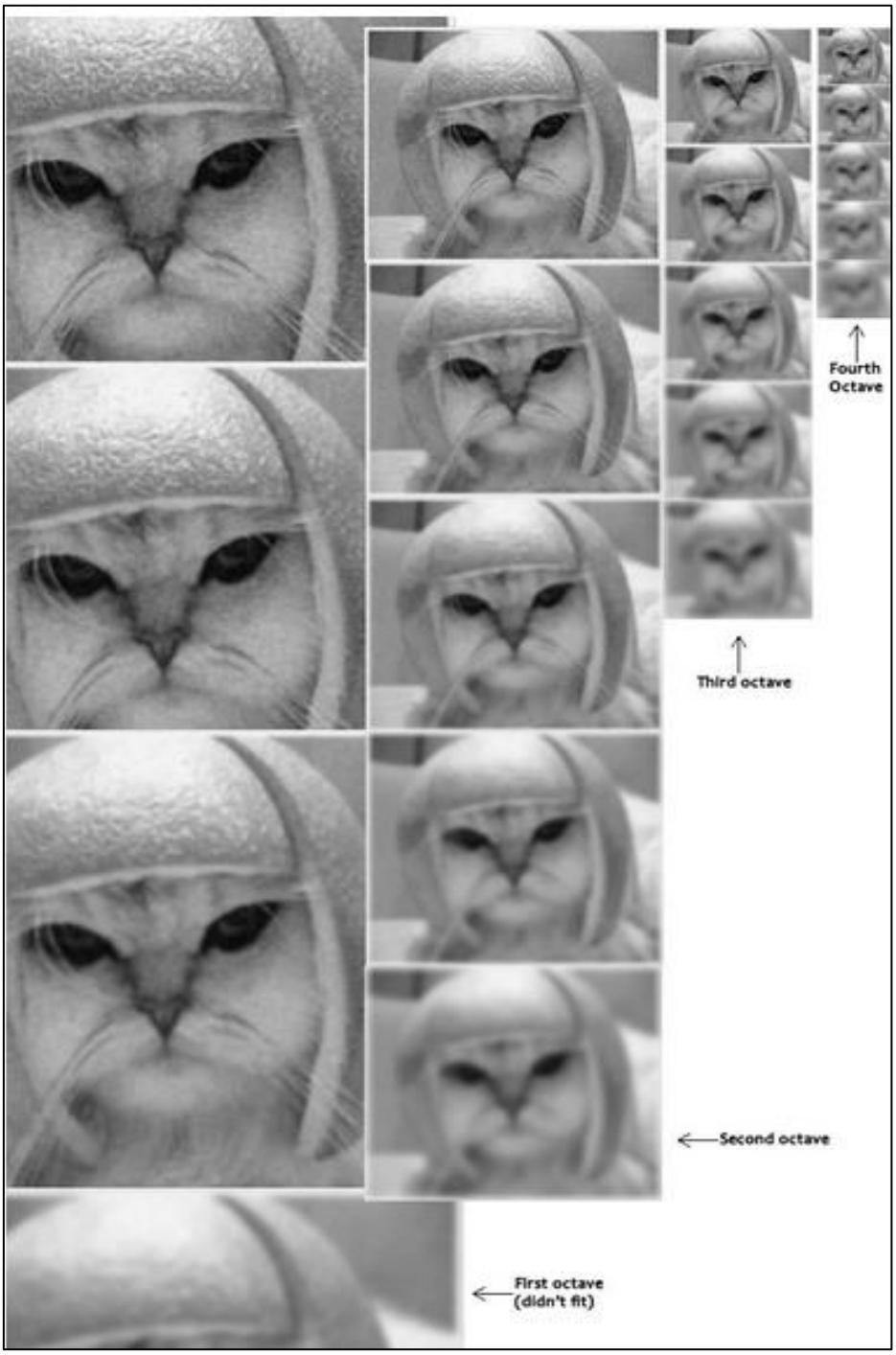
۶.۷.فضای مقیاس

اجسام دنیای واقعی تنها در مقیاس معینی با معنی هستند. به عنوان نمونه، شما کاملا یک حبه قند را ممکن است ببینید، اما، اگر به تمام راه شیری نگاه کنید این کار به سادگی امکان پذیر نخواهد بود. این طبیعت چند مقیاسه اجسام کاملا در طبیعت متداول است.. در این بین فضای مقیاس سعی می‌کند تا این مفهوم (چند مقیاسه) را در تصاویر دیجیتالی تقلید کند.

در ضورتی که بخواهید یه یک درخت نگاه کنید عمدا جزئیات تصویر همانند برگها و شاخه‌های کوچک را حذف می‌کنید. البته باید اطمینان حاصل شود که با حذف این جزئیات، جزئیات نادرست تولید نشود. تنها روش ممکن برای چنین کاری به کارگیری ماتی گوسی است. بنابراین برای تولید فضای مقیاس تصویر اولیه را می‌گیرید و به تدریج تصاویر مات شده تولید می‌کنید. به عنوان نمونه شکل ۱۴ را نگاه کنید. همان گونه که ماتی بیشتر بر روی تصویر انجام می‌گیرد کلاه و سبیل گربه بیشتر جزئیات خود را از دست می‌دهند.

شکل ۱۴: تصویر در فضای مقیاس

تبدیل ویژگی نابسته مقیاس فضاهای مقیاس را به سطح بعدی می‌برد. شما تصویر اولیه را بر می‌دارید و تصاویر به تدریج مات شونده تولید می‌کنید. آنگاه تصویر اولیه را به نصف اندازه قبلی در ارتفاع و پهنا تغییر اندازه می‌دهید. و دوباره تصاویر به تدریج مات شونده تولید می‌کنید و به همین ترتیب ادامه می‌دهید. در شکل ۱۵ تصاویر تولید شده نمایش یافته است.



شکل ۱۵:تصاویر هم اندازه (عمودی) یک اکتاو را تشکیل می‌دهند.

در شکل ۱۵ چهار اوکتاو دیده می‌شود. که هر اکتاو پنج تصویر را شامل می‌شود.در ادامه با لحاظ مباحث بالا چند نکته تکنیکی را مورد بحث قرار می‌دهیم.

تعداد اوکتاوها و مقیاسها وابسته به اندازه تصویر اولیه است. در واقع به هنگام برنامه نویسی تبدیل ویژگی نابسته مقیاس باید راجع به تعداد مقیاسها و اوکتاوها تصمیم بگیرید. اما دیوید لو خالق تبدیل ویژگی نابسته مقیاس پیشنهاد می‌کند که ۴ اوکتاو و ۵ سطح ماتی برای الگوریتم ایده ال است.

توجه داشته باشید در صورتی که اندازه تصویر اولیه دو برابر و کمی‌مات شود آنگاه الگوریتم چهار برابر نقاط کلیدی یشتری تولید خواهد کرد. لازم به ذکر است که هر چقدر نقاط کلیدی بیشتر باشد بهتر است.

از سوی دیگر، به لحاظ ریاضی، ماتی به پیچش اپراتور گوسی و تصویر اشاره دارد. ماتی گوسی یک عبارت یا اپراتور خاصی دارد که به هر پیکسل تصویر اعمال می‌شود و نتیجه تصویر مات شده است:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \qquad (۱.۷)$$

که $L(x,y,\sigma)$ تصویر ماتت شده، $G(x,y,\sigma)$ اپراتور ماتی گوسی،$I(x,y)$ تصویر اولیه، x و y مختصات محل پیکسلها هستند. σ پارامتر مقیاس است که مقدار ماتی را کنترل می‌کند. هر چقدر مقدار بیشتری داشته باشد ماتی بیشتر است.. اپراتور $*$ عملیات پیچش در امتداد x و y را نشان می‌دهد. این عملگر ماتی گوسی $G(x,y,\sigma)$ به روی تصویر $I(x,y)$ اعمال می‌کند:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2}\, e^{-(x^2+y^2)/2\sigma^2} \qquad (۲.۷)$$

مقدار مات شدگی در هر تصویر بسیار مهم است. فرض کنید که مقدار ماتی در یک تصویر خاص σ است. آنگاه مقدار ماتی در تصویر بعدی $k \times \sigma$ است. در اینجا k ثابتی است که توسط کاربر انتخاب می‌شود:

	scale →				
octave	0.707107	1.000000	1.414214	2.000000	2.828427
	1.414214	2.000000	2.828427	4.000000	5.656854
	2.828427	4.000000	5.656854	8.000000	11.313708
	5.656854	8.000000	11.313708	16.000000	22.627417

جدول ۱: مثال عددی مقیاس در برابر اکتاو

در جدول یک مقادیر ماتی در هر اوکتاو و سطح آورده شده است. در اینجا مقدار k برابر $\sqrt{2}$ در نظر گرفته شده است.

به طور خلاصه، در گام اول تبدیل ویژگی نابسته مقیاس چند اوکتاو از تصویر اولیه تولید می‌شود. اندازه هر تصویر در یک اوکتاو نصف اندازه تصویر در اوکتاو قبلی است. در ضمن در هر اوکتاو تصاویر به

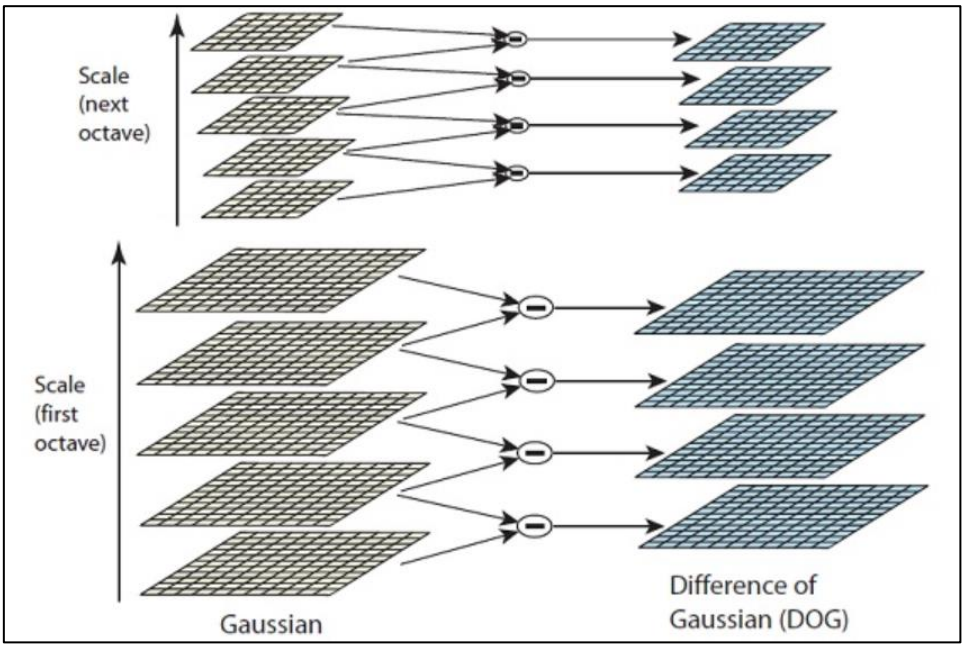
تدریج با به کارگیری اپراتور ماتی گوسی مات می‌شوند. در گام بعدی، ما از این اوکتاوها برای تولید تصاویر تفاضل گوسی استفاده خواهیم کرد.

۷.۷.تقریبهای لاپلاسین گوسی

در مرحله قبل، ما فضای مقیاس تصویر را ایجاد کردیم. ایده این بود که یک تصویر را به تدریج مات کنیم کوچکش کنیم تصویر کوچک شده را به تدریج مات کنیم و الی آخر. حالا ما آن تصاویر مات شده را برای تولید مجموعه دیگری از تصاویر به کار می‌بریم یعنی اختلاف گوسیها. این تصاویر اختلاف گوسی برای یافتن نقاط کلیدی در تصویر عالی هستند.

در عملیات لاپلاسین گوسی ابتدا یک تصویر را کمی‌مات می‌کنیم. سپس، مشتق مرتبه دوم (لاپلاسین) آن را به دست می‌آوریم. این عمل سبب می‌شود تا محل لبه‌ها و گوشه‌ها در تصویر مشخص شوند. این لبه‌ها و گوشه‌ها برای یافتن نقاط کلیدی مناسب هستند. لازم به ذکر است که مشتق دوم خیلی به نویز حساس است و ماتی صورت گرفته سبب می‌شود تا نویز کاهش یابد و مشتق دوم پایدار شود. اما مشکل عمده در محاسبات زیاد مشتق مرتبه دوم است. بنابراین در اینجا از روش دیگری بهره می‌بریم.

جهت تولید سریع تصاویر لاپلاسین گوسی از فضای مقیاس استفاده می‌کنیم. ما تفاضل بین دو مقیاس پشت سر هم و یا تفاضل گوسیها را محاسبه می‌کنیم:



شکل ۱۶: اختلاف گوسین‌ها

این تصاویر تفاضل گوسی تقریبا معادل با تصاویر لاپلاسین گوسی هستند. در اینجا فرآیند بسیار هزینه بر محاسباتی با یک تفاضل ساده (سریع و موثر) جایگزین می‌شود.

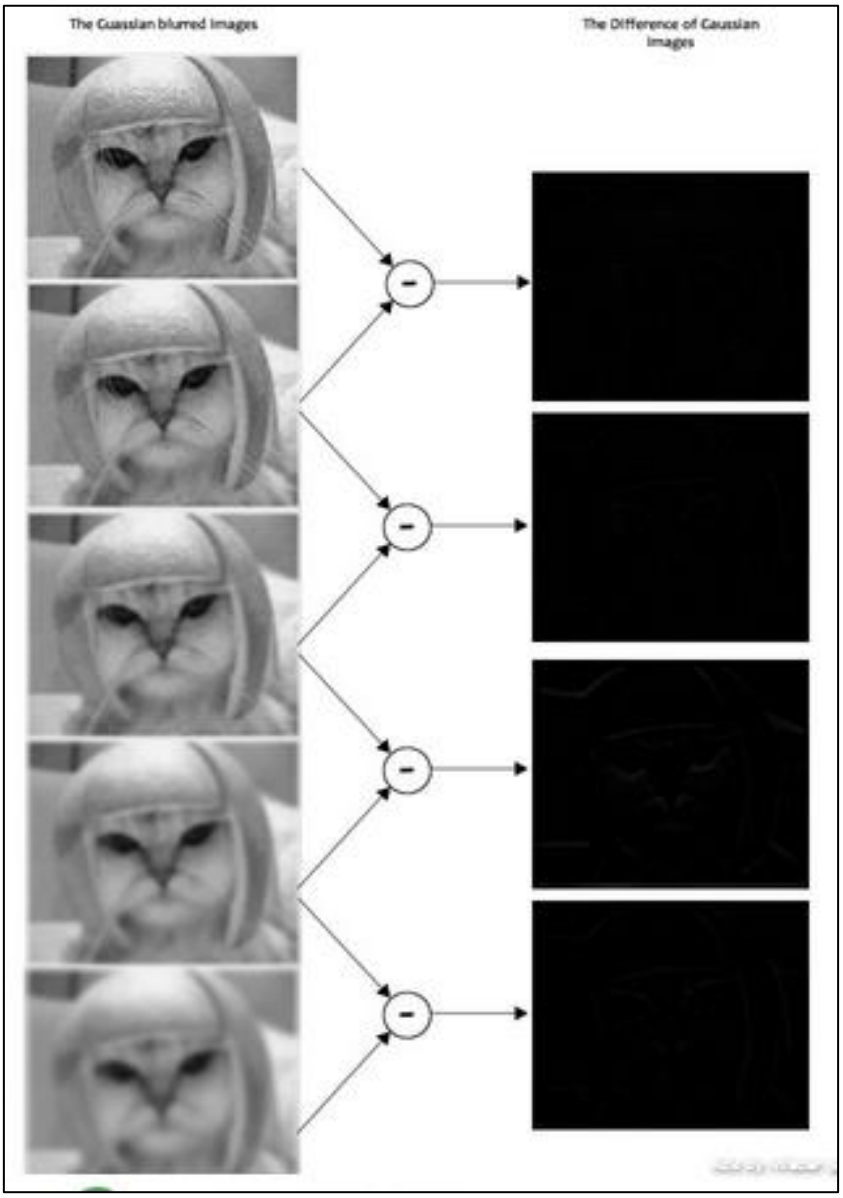
این تصاویر تفاضل گوسی نابسته مقیاس نیز هستند. به عبارت دیگر تصاویر لاپلاسین گوسی عالی نیستند چون آنها نابسته مقیاس نیستند. یعنی آنها به مقدار ماتی وابسته هستند. دلیل این امر از عبارت گوسی زیر پیدا است:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2}\, e^{-(x^2+y^2)/2\sigma^2}$$

به σ^2 در مخرج دقت کنید. این مقدار با مقیاس متناظر است. اگر به طریقی بتوانیم آن را حذف کنیم ما استقلال مقیاس واقعی خواهیم داشت. بنابراین، اگر لاپلاسین گوسی به صورت $\nabla^2 G$ نمایش یابد آنگاه لاپلاسین گوسی نابسته مقیاس به صورت $\sigma^2\nabla^2 G$ خواهد بود: اما تمام این پیچیدگیها توسط عملگر تفاضل گوسی در نظر گرفته می‌شود. تصاویر منتجه بعد از اعمال عملگر تفاضل گوسی از قبل در σ^2 ضرب می‌شوند. البته اثبات می‌شود که این خاصیت نابسته مقیاس نقاط ساده تری تولید می‌کند.

هیچ منفعتی بدون عوارض جانبی نیست. شما می‌دانید که نتیجه تفاضل گوسی در σ^2 ضرب می‌شود. اما همچنین در عدد دیگری $(k-1)$ ضرب می‌شود. این همان k است که در مرحله قبل راجع به آنبحث شد. اما شایان ذکر است که ما تنها در جستجوی محل بیشینه‌ها و کمینه‌ها در تصاویر هستیم و هرگز مقادیر واقعی را در آن محلها بررسی نمی‌کنیم. بنابراین، این فاکتور اضافی مساله‌ای ایجاد نخواهد کرد. به عبارت دیگر اگر خروجی در یک عدد ثابت ضرب شود بیشینه و کمینه در همان محل باقی می‌ماند.

در ادامه با یک مثال ساده نشان می‌دهیم که تفاضل گوسی‌ها چگونه کار می‌کنند.



شکل ۱۷: مثالی از استخراج DOG

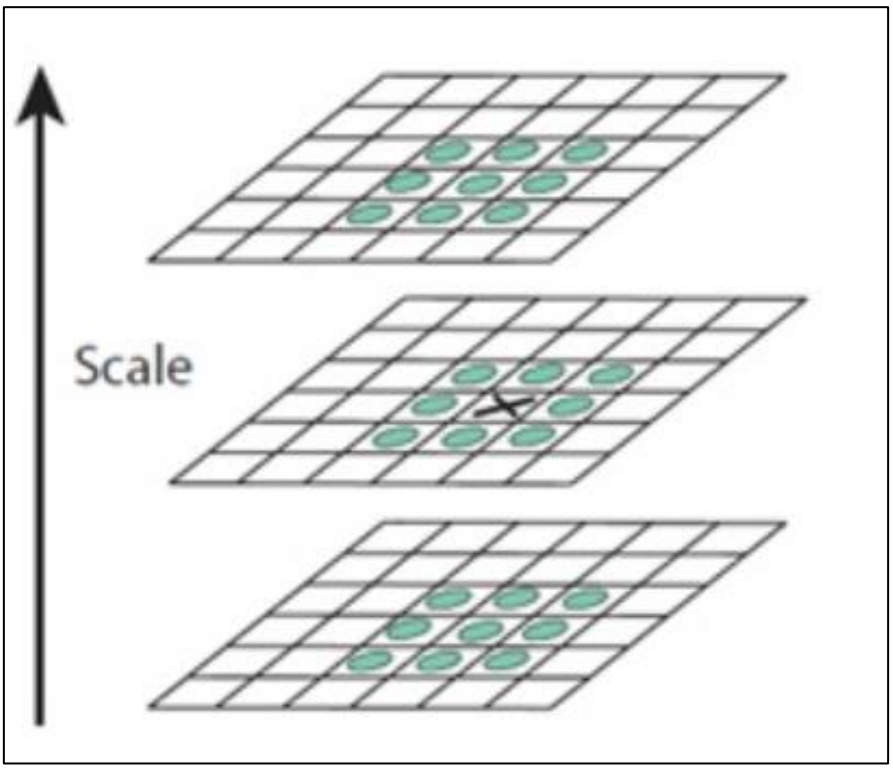
در شکل ۱۷ تفاضل برای تنها یک اوکتاو انجام شده است. البته همین کار برای تمام اوکتاوها انجام می‌پذیرد. این به سادگی تصاویر تفاضل گوسی با اندازه‌های مختلف تولید می‌کند.

به طور خلاصه دو تصویر پشت سرهم در یک اوکتاو انتخاب و از همدیگر منها می‌شوند. آنگاه جفت پشت سر هم بعدی انتخاب و از همدیگر منها می‌شوند و فرآیند تکرار می‌شود. این کار برای تمام اوکتاوها انجام می‌گیرد. تصاویر منتجه، تقریبی از لاپلاسین گوسی نابسته مقیاس خواهند بود که برای

آشکارسازی نقاط کلیدی مناسب هستند. البته کاستی‌های در تقریب وجود دارد که الگوریتم را تحت تاثیر قرار نمی‌دهد.
در ادامه نشان خواهیم داد که واقعا چگونه نقاط کلیدی(بیشینه و کمینه یا بیشنه و کمینه تصاویر) را می‌یابیم.

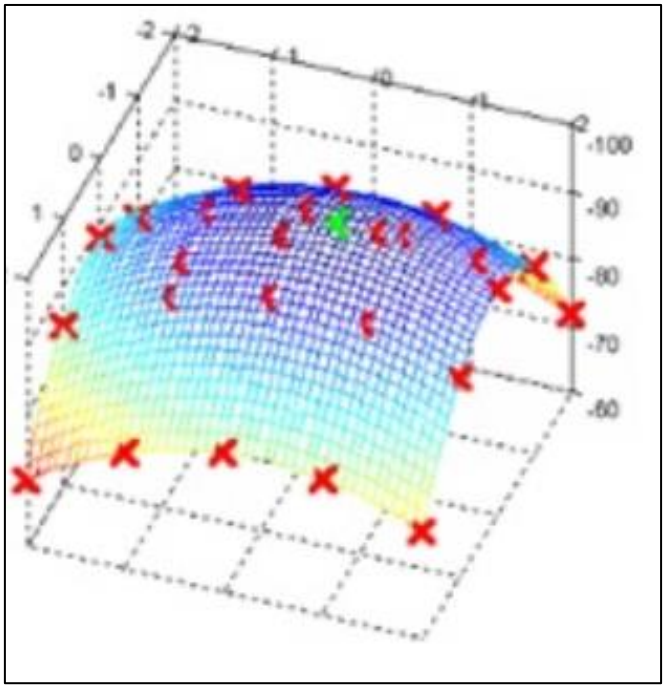
۸.۷.یافتن نقاط کلیدی

تاکنون یک فضای مقیاس تولید شده است و از آن برای محاسبه تفاضل گوسی‌ها استفاده شده است. آنگاه این تفاضلها برای محاسبه تقریبهای لاپلاسین گوسی استفاده می‌شود که نابسته مقیاس است. این تقریبها نقاط کلیدی عالی تولید می‌کنند. این امر با یافتن محل بیشینه یا کمینه در تصاویر تفاضل گوسی حاصل می‌شود. در واقع، اولین گام تعیین تقریبی محل بیشینه و کمینه است. این کار به سادگی با وارسی هر پیکسل و همسایگانش صورت می‌گیرد. این بررسی در تصویر جاری و تصویر بالایی و پایینی انجام می‌گیرد. شکل ۱۸ را برای اطلاعات بیشتر نگاه کنید.



شکل ۱۸: یافتن نقاط کلیدی

× پیکسل جاری را نشان می‌دهد. دایره‌های سبز همسایه‌ها را مشخص می‌کنند. در این وضعیت ۲۶ بررسی صورت می‌گیرد. × به عنوان نقطه کلیدی علامت زده می‌شود اگر بزرگترین یا کمترین تمام ۲۶ همسایه اش باشد.. معمولا یک موقعیت غیربیشینه یا غیرکمینه تمام ۲۶ بررسی را در نظر نمی‌گیرد. چند بررسی اولیه برای در نظر نگرفتن آن کافی خواهد بود. در ضمن دقت کنید که نقاط کلیدی در پایین ترین و بالاترین مقیاسها آشکار نمی‌شوند. چرا که همسایه‌های کافی برای انجام مقایسه وجود ندارند. بنابراین آنها را در نظر نمی‌گیریم. وقتی این کار انجام شد نقاط علامت گذاری شده کمینه‌های و بیشینه‌های تقریبی هستند. آنها تقریبی هستند چون که بیشینه‌ها و کمینه‌ها تقریبا دقیقا روی یک پیکسل قرار نمی‌گیرد. بلکه جای بین پیکسلها قرار دارند. اما ما به سادگی به داده بین پیکسلها دسترسی نداریم. پس ما باید به لحاظ ریاضی محل زیر پیکسل را تعیین کنیم (شکل ۱۹ را نگاه کنید).

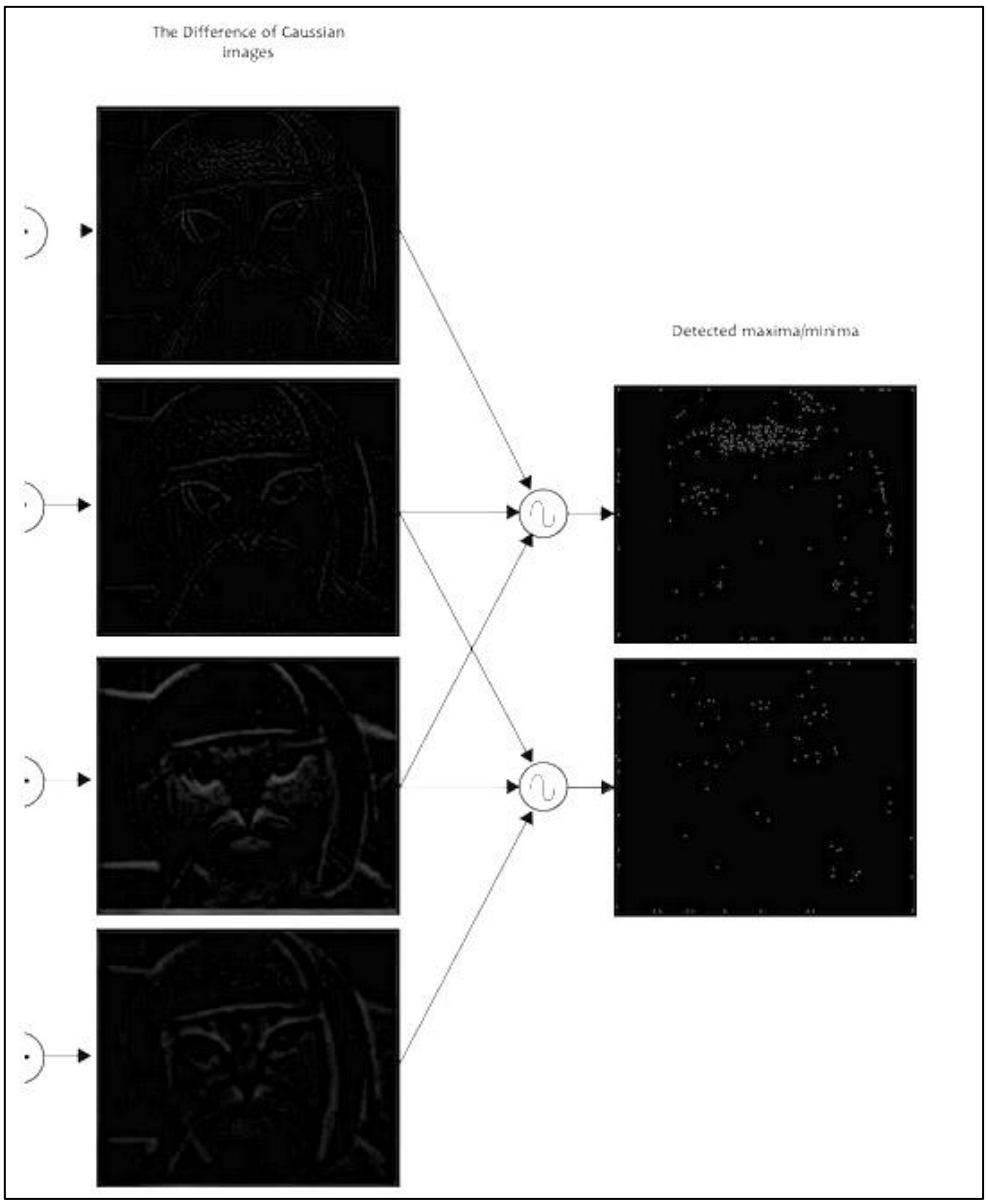


شکل۱۹: یافتن زیر پیکسل‌ها

ضربدرهای قرمز پیکسلها در تصویر را علامت گذاری می‌کنند. اما نقطه اکستریمم واقعی نقطه سبز رنگ است. با به کارگیری داده پیکسل در دسترس مقادیر زیر پیکسل تولید می‌شوند و این کار با بسط سری تیلور تصویر اطراف نقاط کلیدی تقریب زده می‌شود:

(۳.۷)

ما به سادگی نقاط اکستریمم این معادله را با مشتق گیری و معادل با صفر قرار دادن می‌یابیم. بعد از حل، محلهای نقاط کلیدی زیر پیکسل به دست خواهد آمد. این مقادیر زیر پیکسل شانس تطبیق و پایداری الگوریتم را افزایش می‌دهد. در شکل ۲۰ نتایج تصویر نمونه آورده شده است:



شکل ۲۰: نتایج تصویر نمونه

دیوید لو مبدع تبدیل ویژگی نابسته مقیاس توصیه می‌کند دو تصویر اکستریمم تولید کنید. پس ما به ۴ تفاضل گوسی نیاز داریم. برای تولید چهار تصویر تفاضل گوسی به ۵ تصویر مات شده گوسی نیاز داریم. بنابراین، ۵ سطح ماتی در هر اوکتاو مورد نیاز است. در شکل ۲۰ تنها یک اوکتاو نشان داده شده است. این کار برای تمام اوکتاوها انجام می‌شود. افزون بر این، این تصویر تنها اولین قسمت آشکارسازی نقطه کلیدی را نشان می‌دهد. بخش سری تیلور نشان داده نشده است.

به طور خلاصه، ما بیشینه‌ها و کمینه‌ها در تصاویر تفاضل گوسی تولید شده در مرحله قبلی را آشکار کردیم. این کار با مقایسه پیکسلهای همسایه در مقیاس جاری، مقیاس بالا، و مقیاس پایین انجام می‌گیرد. در ادامه نشان خواهیم داد که چگونه برخی از نقاط کلیدی آشکار شده رد خواهد شد. دلیل این رد از آنجا ناشی می‌شود که آنها کنتراست کافی ندارند و یا در یک لبه قرار دارند.

۹.۷. حذف نقاط کلیدی با کنتراست پایین

در مرحله قبل تعداد زیادی نقاط کلیدی تولید می‌شود که برخی از آنها در امتداد لبه‌ها هستند و یا کنتراست به اندازه کافی ندارند. در هر دو حالت، این نقاط به عنوان ویژگی مفید نیستند. پس نیازمند حذف چنین نقاطی هستیم. رویه حذف همانند رویه حذف در آشکارساز گوشه هریس برای برداشتن ویژگیها لبه است. به منظور حذف ویژگیهای کنتراست پایین به سادگی شدتشان را وارسی می‌کنیم. اگر بزرگی شدت (بدون علامت) در پیکسل جاری در تصویر تفاضلی گوسی کمتر از مقدار معینی باشد آن نقطه حذف می‌شود. چون ما نقاط کلیدی زیر پیکسل داریم از سری تیلور برای پالایش نقاط کلیدی استفاده می‌کنیم. در واقع، ما از سری تیلور برای به دست آوردن مقدار شدت در محلهای زیر پیکسل استفاده می‌کنیم. در این حالت، اگر بزرگی از یک مقدار معین کمتر باشد آن نقطه کلیدی رد می‌شود.

ایده رد لبه‌ها با محاسبه بردار گرادیان در نقطه کلیدی شروع می‌شود. در واقع مولفه‌های بردار گرادیان بر همدیگر عمود هستند. بر اساس تصویر اطراف نقطه کلیدی، سه وضعیت ممکن وجود دارد. در واقع، تصویر اطراف نقطه کلیدی می‌تواند:

الف- مسطح باشد در این وضعیت دو مولفه بردار گرادیان کوچک خواهد بود.

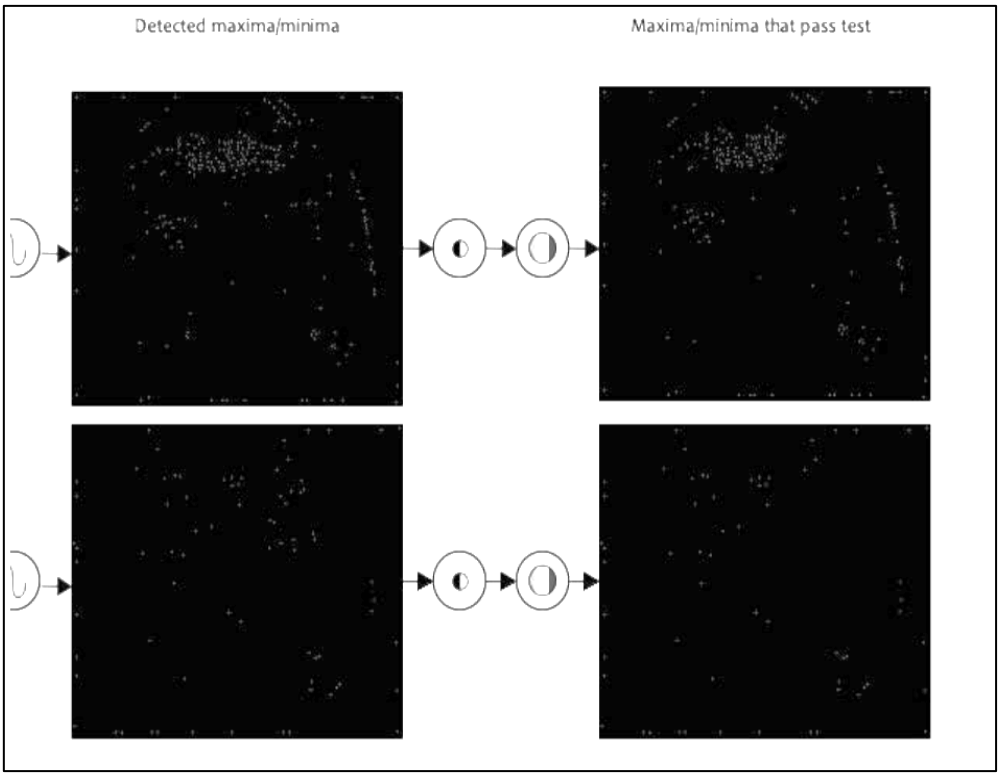
ب- یک لبه باشد در این حالت، یکی از مولفه‌ها از دیگری بزرگتر است (به لبه عمود است) و مولفه دیگر کوچک است (در امتداد لبه است).

ج- یک گوشه است. در این حالت، هر دو مولفه بزرگ خواهد بود.

گوشه‌ها نقاط کلیدی بسیار کارآمدی هستند، پس گوشه‌ها را می‌خواهیم. به عبارت دیگر، اگر دو مولفه بردار گرادیان به اندازه کافی بزرگ باشد ان نقطه به عنوان نقطه کلیدی باقی می‌ماند و در غیر اینصورت آن نقطه رد می‌شود. این کار به لحاظ ریاضی با ماتریس هسین حاصل می‌شود. در واقع با به کارگیری این

ماتریس می‌توانیم به سادگی گوشه بودن یا نبودن یک نقطه را وارسی کنیم. همان گونه که ملاحظه کردید در آشکارساز گوشه هریس، دو مقدار ویژه محاسبه می‌شوند. این در حالی است که در تبدیل ویژگی نابسته مقیاس تنها نسبت دو مقدار ویژه محاسبه می‌شود و نیازی به محاسبه مقادیر ویژه واقعی وجود ندارد. جهت توضیحات بیشتر به بخش آشکار ساز هریس مراجعه کنید.

در اینجا همانند بخشهای قبلی مثال بصری در شکل ۲۱ نمایش یافته است.



شکل ۲۱: حذف نقاط کنتراست پایین

هر دو تصویر اکستریمم تست کنتراست و تست لبه را می‌گذرانند. این دو تست بسته به تصویر تعدادی از نقطه‌های کلیدی را حذف می‌کنند و بنابراین تعداد کمتری از نقاط کلیدی باقی می‌مانند.

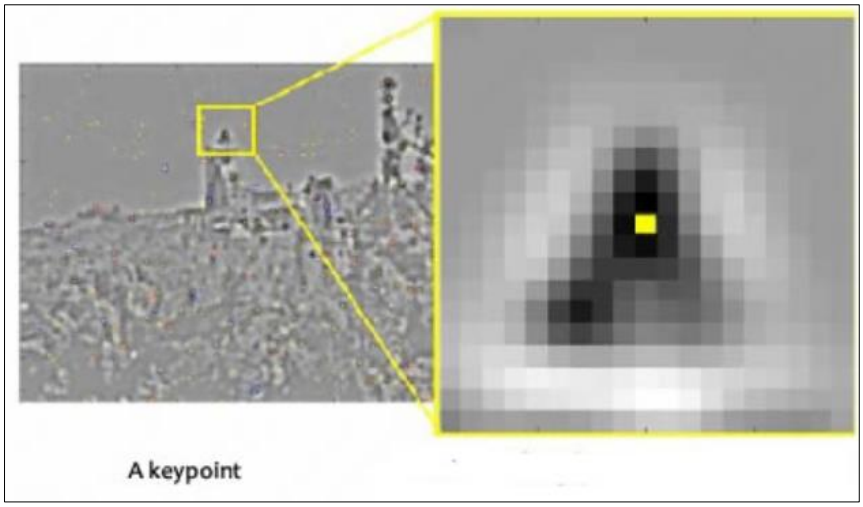
به طور خلاصه، در این گام تعداد نقاط کلیدی کاهش می‌یابد. این کار سبب می‌شود تا راندمان و مقاومت الگوریتم افزایش یابد. نقاط کلیدی در صورتی که کنتراست پایین داشته باشند و یا بر روی لبه قرار گرفته باشند حذف می‌شوند.

در مرحله بعدی، امتدادی به هر کدام از نقاط کلیدی باقی مانده تخصیص خواهد یافت.

۱۰.۷. امتدادهای نقاط کلیدی

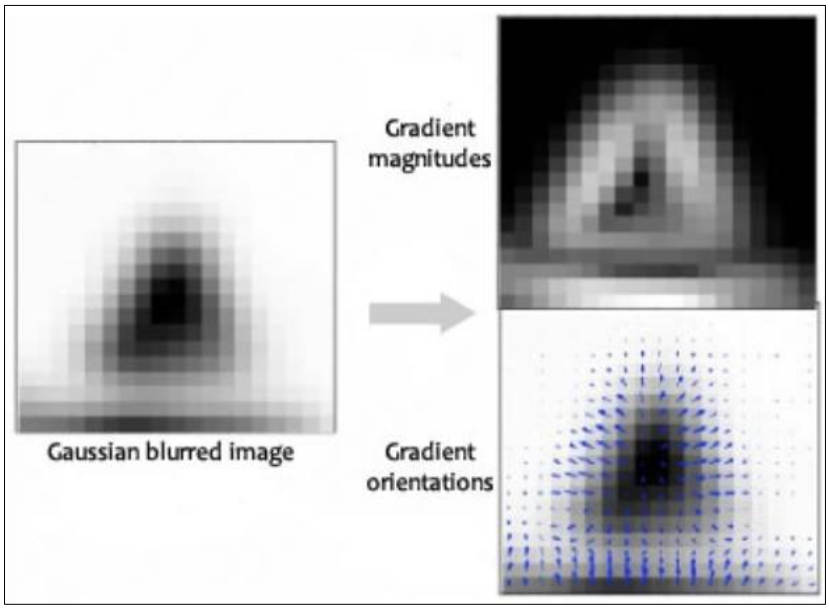
بعد از مرحله پنجم نقاط کلیدی معتبری در اختیار داریم. آنها مورد آزمون قرار گرفته اند پس پایدار هستند. ما از قبل می‌دانیم که در کدام مقیاس نقاط کلید آشکار شده اند (همان مقیاس تصویر مات شده است). پس ما خاصیت نابسته مقیاس داریم. مطلب بعدی تخصیص یک امتداد به هر نقطه کلیدی است. این امتداد سبب می‌شود تا تبدیل ویژگی خاصیت نابسته چرخش را نیز داشته باشد. در واقع، هر چقدر بیشتر خواص نابسته داشته باشیم بهتر است.

تخصیص امتداد به نقاط کلیدی با جمع آوری جهتهای گرادیان و بزرگیها اطراف هر نقطه کلیدی حاصل می‌شود. آنگاه امتداد غالب را در آن ناحیه می‌یابیم و این امتداد به نقطه کلیدی تخصیص داده می‌شود(شکل ۲۲). لازم به ذکر است که محاسبات بعدی نسبت به این امتداد محاسبه می‌شود. این امر حاصیت نابسته مقیاس را تضمین می‌کند.



شکل ۲۲: تخصیص امتداد به یک نقطه‌ی کلیدی نمونه

اندازه ناحیه امتداد اطراف نقطه کلیدی به مقیاسش وابسته است. مقیاس بزرگتر سبب می‌شود تا ناحیه بزرگتر باشد.



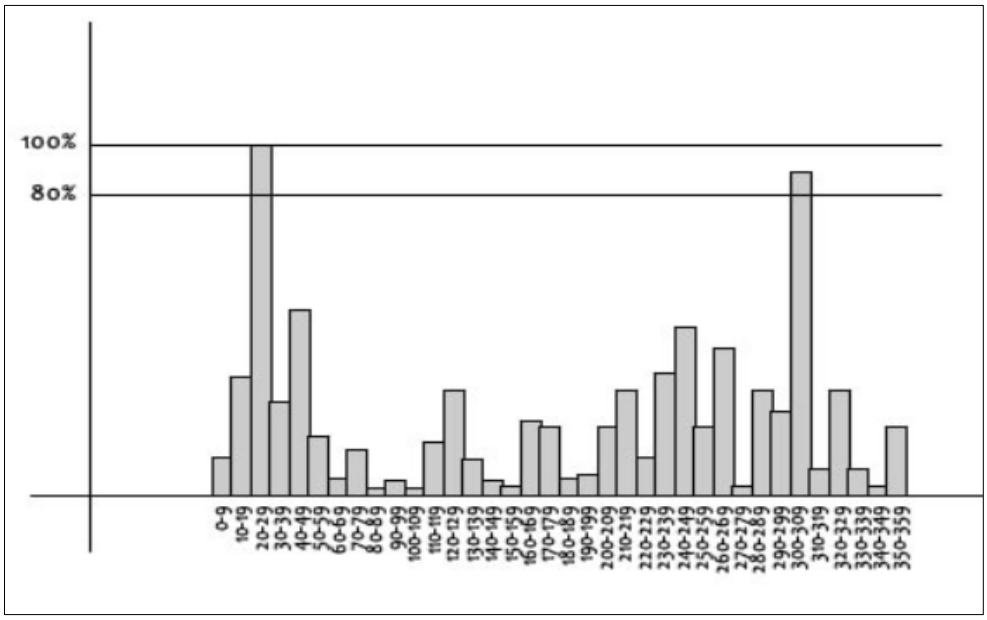
شکل ۲۳: اندازه و جهت گرادیان حوب نقطه ی کلیدی

بزرگی مولفه‌های بردار گرادیان با به کارگیری فرمول زیر محاسبه می‌شود:

$$m(x,y) = \sqrt{\big(L(x+1,y) - L(x-1,y)\big)^2 + \big(L(x,y+1) - L(x,y-1)\big)^2}$$

$$\theta(x,y) = \tan^{-1}(\big(L(x,y+1) - L(x,y-1)\big)L(x+1,y) - L(x-1,y)))$$

بزرگی و امتداد برای تمام پیکسلها اطراف نقطه کلیدی محاسبه می‌شود. آنگاه نمودار فراوانی تولید می‌شود. در این نمودار، ۳۶۰ درجه امتداد به ۳۶ بازه تقسیم می‌شود(هر بازه ۱۰ درجه). فرض کنید که جهت گرادیان در یک نقطه خاس (در ناحیه امتداد) ۱۸.۷۵۹ درجه باشد آنگاه این امتداد به بازه ۱۰ الی ۱۹ اختصاص خواهد یافت و مقدار اضافه شده به بین (بازه) متناسب با بزرگی گرادیان در آن نقطه تخصیص خواهد یافت. وقتی این کار را برای تمام پیکسلها اطراف نقطه کلیدی انجام دادید در نمودار فراوانی یک پیک در نقطه‌ای خواهد داشت. در شکل ۲۳ پیکهای نمودار فراوانی را در بین (بازه) ۲۰ الی ۲۹ درجه می‌بینید. بنابراین به نقطه کلیدی امتداد ۳ اختصاص داده می‌شود (بازه سوم). هم چنین هر پیک بالای ۸۰ درصد بالاترین پیک به نقطه کلیدی جدید تبدیل می‌شود. این نقطه کلیدی جدید همان محل و مقیاس مشابه نقطه کلیدی قبلی را دارد. اما امتداد آن معادل با پیک دیگر است. بنابراین امتداد، یک نقطه کلیدی را به چند نقطه کلیدی تقسیم می‌کند.



شکل ۲۴: هیستوگرام گرادیان ها

لازم به ذکر است که تصویر بزرگی بردار گرادیان به اندازه یک و نیم برابر σ مات می‌شود. از سوی دیگر، اندازه پنجره یا ناحیه مجموعه امتداد معادل با اندازه کرنل برای ماتی گوسی به مقدار یک و نیم برابر σ است.

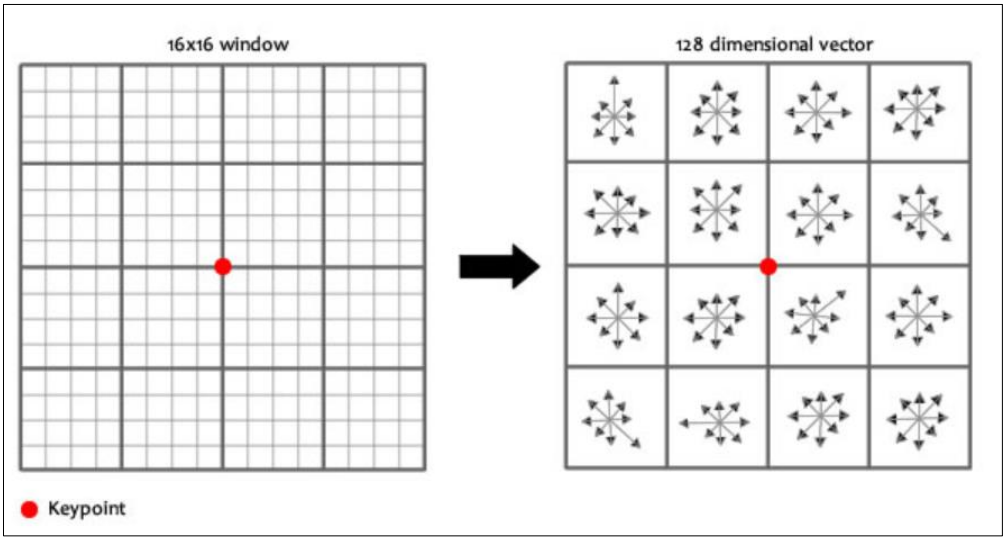
به طور خلاصه برای تخصیص امتداد از هیستوگرام ناحیه کوچک اطراف نقطه کلیدی بهره می‌بریم. با به کار گیری این نمودار فراوانی غالبترین امتدادهای گرادیان شناسایی می‌شوند. اگر تنها یک پیک در نمودار وجود داشته باشد به نقطه کلیدی تخصیص می‌یابد. اما اگر چند پیک بالای ۸۰ درصد پیک اول وجود داشته باشند تمام آنها به نقطه کلیدی جدید یا بردار ویژگی (۱۲۸ عدد) تبدیل می‌شوند.

۷.۷ تولید ویژگی

در مرحله نهایی تبدیل ویژگی نابسته مقیاس ویژگی نابسته مقیاس و نابسته چرخش حاصل خواهد شد. در این مرحله اثر انگشت برای هر نقطه کلیدی ایجاد می‌شود. این اثر انگشت برای شناسایی نقطه کلیدی به کار خواهد رفت. به عنوان نمونه اگر چشمی‌به عنوان یک نقطه کلیدی باشد آنگاه با به کارگیری اثر انگشت ما قادر خواهیم بود که آن را از نقاط کلیدی دیگر همانند گوشها، بینی‌ها، انگشتها، و غیره متمایز کنیم.

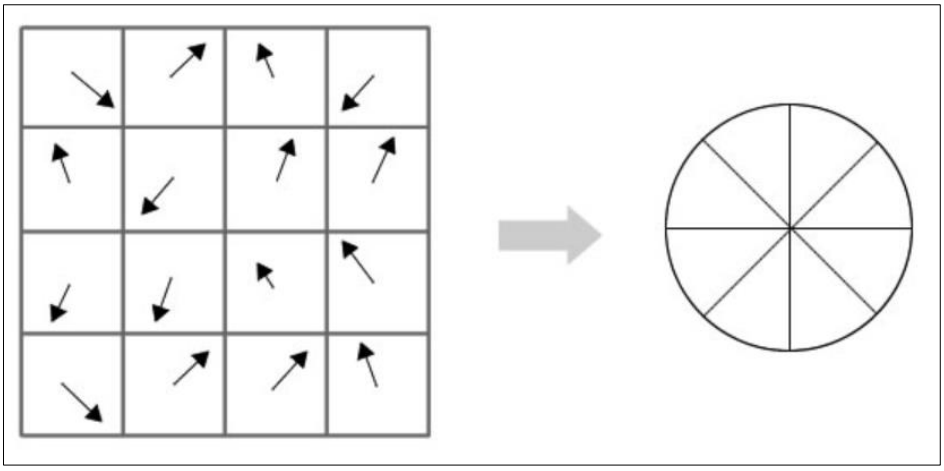
در شکل ایده تولید اثر انگشت یکتا برای نقطه کلیدی آمده است. این اثر انگشت باید به لحاظ محاسباتی ساده باشد. این اثر انگشت باید به گونه‌ای باشد که اجازه مقایسه با سایر نقاط کلیدی را بدهد. به عبارت

دیگر برای نقاط کلیدی مشابه در دو تصویر متقاوت یکسان باشد. در این راستا یک پنجره ۱۶ در ۱۶ اطراف نقطه کلیدی در نظر می‌گیریم. این پنجره ۱۶ در ۱۶ به ۱۶ پنجره ۴ در ۴ تقسیم می‌شود.



شکل ۲۵: استخراج توصیف گر

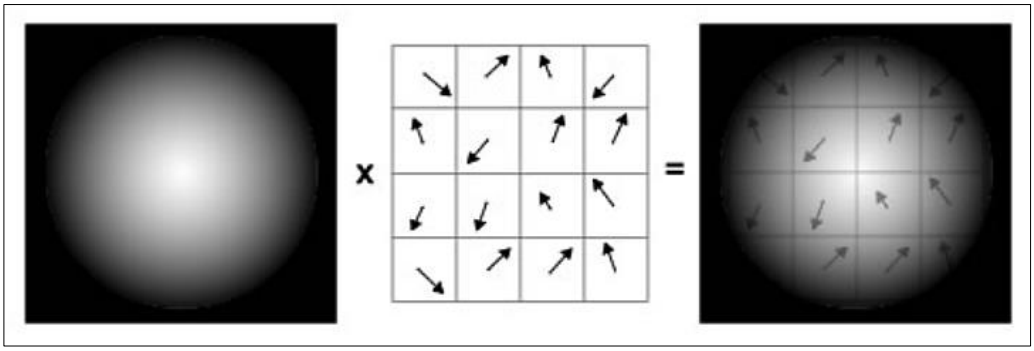
در هر پنجره ۴ در ۴ بزرگی و امتداد بردارهای گرادیان محاسبه می‌شود. این امتدادها در یک هیستوگرام با ۸ بازه به صورت زیر قرار داده می‌شود:



شکل ۲۶: کوانتیزیشن گرادیان‌ها

هر امتداد گرادیان در محدوده ۰ الی ۴۴ درجه به بازه اول اضافه می‌شود. همچنین هر امتداد گرادیان در محدوده ۴۵ الی ۸۹ به بازه بعدی اضافه می‌شود و الی آخر. مقدار اضافه شده به هر بازه وابسته به بزرگی

گرادیان دارد. بر خلاف گذشته، مقدار اضافه شده وابسته به فاصله از نقطه کلیدی است. بنابراین گرادیانهایی که از نقطه کلیدی دورتر هستند مقادیر کوچکتری در هیستوگرام دارند. این کار با تابع وزندهی گوسی انجام می‌گیرد. به سادگی این تابع منحنی زنگوله‌ای شکل دوبعدی است. این تابع به بزرگی امتدادها ضرب می‌شود و بزرگی وزندار حاصل می‌شود. در نتیجه هر چقدر امتداد گرادیانی از نقطه کلیدی دورتر باشد بزرگی وزندار مقدار کمتری خواهد داشت.



شکل ۲۷: وزن دهی به ضرایب گرادیان

با انجام این کار برای تمام ۱۶ پیکسل، ۱۶ امتداد تصادفی به ۸ بازه با وزنهای مشخص تخصیص می‌یابند. این کار برای تمام ۱۶ بخش انجام می‌شود. پس در نهایت به ۴ در ۴ در ۸ مساوی ۱۲۸ می‌رسیم. به محض اینکه تمام ۱۲۸ عدد را داشتید آنها را نرمالیزه می‌کنید. به همان صورتی که یک بردار را در دبیرستان نرمالیزه می‌کردید یعنی مولفه‌های آن را به جذر مجموع مجذور مولفه‌ها تقسیم می‌کنید. این ۱۲۸ عدد بردار ویژگی را تشکیل می‌دهند. در واقع نقطه کلیدی به صورت یکتا با این بردار ویژگی مشخص می‌شود.

به احتمال زیاد از شکل ۲۶ متوجه شده اید که نقطه کلیدی بین پیکسلها هستند.. به منظور یافتن تصور بین پیکسلی نیاز به عملیات درون یابی تصویر برای تولید داده امتداد و بزرگی هستید.

بردار ویژگی چند پیچیدگی به شرح زیر با خود به همراه دارد که نیاز به رفع این پیچیدگیها قبل از نهایی کردن اثر انگشت داریم:

الف- وابستگی به چرخش: بردار ویژگی از امتدادهای بردار گرادیان استفاده می‌کند. به روشنی اگر تصویر چرخش داده شود هر چیزی تغییر خواهد کرد. همچنین تمام امتدادهای گرادیان تغییر می‌کنند. به منظور حصول به نابسته چرخش، چرخش نقاط کلیدی از هر امتداد کسر می‌شود. بنابراین هر امتداد گرادیان نسبت به امتداد نقطه کلیدی در نظر گرفته خواهد شد.

ب- وابستگی به روشنایی: اگر بر روی مولفه‌های بردار ویژگی ۱۲۸ نقطه الی آستانه گذاری مناسب صورت گیرد اعداد بزرگ حذف می‌شوند و به خاصیت نابسته روشنایی می‌رسیم. پس هر عدد بزرگتر از

دو دهم به دو دهم برش می‌یابد. این بردار ویژگی منتجه دوباره نرمالیزه می‌شود و ما به بردار ویژگی نابسته روشنایی می‌رسیم.

به طور خلاصه یک پنجره ۱۶ در ۱۶ بین پیکسل اطراف نقطه کلیدی انتخاب می‌شود. این پنجره به ۱ پنجره ۴ در ۴ تقسیم می‌شود. ار هز پنجره ۴ در ۴ یک نمودار فراوانی ۸ بازه‌ای تولید می‌شود. هر بازه معادل با یکی از محدوده‌های ۰الی ۴۴، ۴۵ الی ۸۹ و الی آخر است. امتدادهای گرادیان از هر پنجره ۴ در ۴ به این بازه‌ها اختصاص می‌یابند. این کار برای تمام بلوکهای ۱۶ گانه انجام می‌شود. در نهایت این ۱۲۸ مقدار نرمالیزه می‌شود. به منظور حل چند مشکل، امتداد نقطه کلیدی از هر امتداد گرادیان کم می‌شود. در ادامه، بر روی هر مقدار بردار ویژگی عملیات آستانه گذاری صورت می‌گیرد تا مقدار بیشینه مولفه‌های بردار ویژگی به دو دهم محدود شود. در نهایت بردار ویژگی جدید دوباره نرمالیزه می‌شود.

به محض اینکه ویژگیها را به دست آوردید می‌توانید کارهای مختلفی ردیابی، آشکار سازی و شناسایی را انجام دهید.

تمرینات

۱. رابطه‌ی ۷.۳ را با استفاده از بسط تیلور اثبات کنید.

۲. برای تصویر نمونه‌ی زیر تمام مراحل اصلی الگوریتم SIFT را انجام دهید:

$(\vec{X}\ ,\ Y\downarrow)$

۱۱	۱۸	۱۵	۴۱	۲۹	۲۵	۱۴	۳۲	۱۶	۲۳
۱۵	۱۶	۱۴	۶۱	۲۵	۲۶	۱۲	۵۲	۱۵	۲۶
۱۳	۱۴	۱۵	۱۷	۲۳	۱۳	۱۴	۲۶	۱۴	۲۳
۱۲	۱۲	۱۳	۱۷	۲۲	۱۲	۳۸	۲۳	۱۲	۲۷
۱۵	۱۷	۱۲	۱۲	۲۵	۱۷	۴۲	۳۱	۱۷	۲۸
۱۱	۱۱	۱۱	۱۹	۱۲	۱۸	۱۱	۲۹	۱۲	۲۲
۱۷	۲۲	۱۱	۱۰	۱۶	۱۶	۱۶	۲۳	۱۱	۲۱
۱۳	۲۲	۲۵	۲۹	۲۱	۱۳	۱۴	۲۱	۱۳	۲۶

نقاط ویژگی اکتاو اول و اکتاو دوم در کجا قرار دارند. ماتریس اکتاو سوم را به صورت عددی بنویسید.

۳. فرض کنید پیکسل قرار گرفته در مرکز تصویر یکی از نقاط کلیدی است، هیستوگرام گرادیان در اطراف آن چگونه است، آیا با توجه به همین هیستوگرام می‌توان تشخیص داد که این نقطه واقعا نقطه‌ی کلیدی بوده است یا نه؟

مراجع استفاده شده در این کتاب

References

دروس پروفسور هاف به آدرس
[1] Hoff, W. Introduction to image processing video lectures, http://inside.mines.edu/~whoff/

دروس و کتاب آشنایی با بینایی کامپیوتری تالیف پروفسور مبارک شاه
https://scholar.google.com/citations?user=p8gsO3gAAAAJ
[2] Mubarak Shah, Fundamentals of Computer Vision, Orlando, 1997.

وبلاگ کلبه ی هوش مصنوعی
[3] aishack.in