سومین کنفرانس فناوری اطلاعات و دانش
۶ تا ۸ آذرماه ۱۳۸۶
IKT 2007 کنفرانس بین المللی فناوری اطلاعات و دانش
دانشگاه فردوسی مشهد

لبه یابی تصاویر رنگی به روش برداری در فضای رنگ YUV

نسرین اباذری طرقبه، حمید رضا پوررضا

گروه کامپیوتر دانشکده مهندسی دانشگاه آزاد اسلامی مشهد، گروه کامپیوتر دانشکده مهندسی دانشگاه فردوسی مشهد

E-mail:abazarinasrin@yahoo.com, hpourreza@um.ac.ir

چکیده – لبه یابی تصاویر یکی از مهمترین عملیات در پردازش تصویر به شمارمی‌رود. به علت کاربردهای وسیع تصاویر رنگی، لبه یابی این تصاویر از اهمیت ویژه ای برخوردار است. بطور کلی لبه یابی تصاویر رنگی به دو روش برداری(*Vector*) و ترکیبی(*Synthetic*) انجام می‌شود؛ کیفیت تشخیص لبه و زمان اجرا، این الگوریتم ها را از یکدیگر متمایز می‌سازد. زمان اجرای الگوریتم های لبه یابی در کاربردهای واقعی بسیار حائز اهمیت است؛ بدین معنی که استفاده از الگوریتمی که لبه های تصویر را با کیفیت مطلوب تشخیص داده اما زمان اجرای بالایی دارد در بسیاری از کاربردها (حساس به زمان)، عملا غیر ممکن است. یکی از روش های جدید لبه یابی، الگوریتمی است که با استفاده از درخت پوشای مینیمال و در فضای رنگ *YUV* عملیات لبه یابی را انجام می‌دهد. این الگوریتم از کیفیت بالایی برخوردار می‌باشد اما زمان اجرای آن بسیار بالاست. در این مقاله، الگوریتمی بر مبنای این روش ارائه شده است. نتایج شبیه سازی نشان می‌دهد که الگوریتم پیشنهادی ضمن ارائه کیفیت بالا، از نظر زمان اجرا نسبت به الگوریتم مبتنی بر درخت پوشای مینیمال عملکرد بسیار بهتری دارد.

کلید واژه- تصویر رنگی، فضای رنگ *YUV*، لبه یابی

۱- مقدمه

لبه یابی یکی از مهمترین عملیات در پردازش تصویر به شمار می‌رود. در مقایسه با تصاویر منوکروم، اطلاعات موجود در تصاویر رنگی زیادتر بوده و کاربردهای آن نیز وسیع تر می‌باشد؛ بنابراین در سالهای اخیر تحقیقات بسیاری بر روی لبه یابی تصاویر رنگی انجام شده است[۲, ۱].

لبه یابی در تصاویر رنگی به دو روش ترکیبی و برداری انجام می‌شود. در روش ترکیبی از تکنیک های لبه یابی تصاویر منوکروم، برروی هر سه کانال تصویر رنگی به صورت مستقل استفاده شده و سپس نتایج با استفاده از عملیات منطقی خاص با یکدیگر ترکیب می‌شوند[۳]. در روش برداری هر پیکسل، به صورت برداری در فضای رنگ مشخص، در نظر گرفته شده و عملیات لبه یابی با استفاده از بردارها انجام می‌شود. روش ترکیبی به نسبت روش برداری ساده تر و سریع تر بوده اما در بعضی از موارد شدت لبه را به درستی نشان نمی‌دهد. در این مقاله از روش برداری استفاده شده، زیرا این روش، مشکل روش ترکیبی را حل کرده و کارایی آن نیز فوق العاده است. تحقیقات اخیر نیز بیشتر بر روی این روش انجام شده است[۵, ۴].

یکی از روش های مبتنی بر بردار که اخیرا توسط کنگ و همکارانش [۶] ارائه شده، الگوریتمی مبتنی بر درخت پوشای مینیمال (**M**inimal **S**panning **T**ree) است. این الگوریتم در عین ارائه نتایج خوب از زمان اجرای طولانی رنج می‌برد. در این مقاله الگوریتمی مبتنی بر روش کنگ ارائه شده که ضمن ارائه نتایجی در حد الگوریتم MST از زمان اجرای کوتاهتری برخوردار است.

در ادامه این مقاله، بخش ۲ به معرفی الگوریتم مبتنی برMST می‌پردازد. بخش ۳ روش پیشنهادی این مقاله را ارائه می‌نماید. در بخش ۴ الگوریتم Automatic Fast Entropy Thresholding که روشی برای بدست آوردن مقدار آستانه بهینه می‌باشد بررسی شده و بخش ۵ نیز نتایج

بدست آمده از الگوریتم پیشنهادی را نمایش داده و به مقایسه الگوریتم پیشنهادی با الگوریتم MST پرداخته است و در پایان، بخش ۶ نتیجه گیری را بیان می‌کند.

۲- الگوریتم مبتنی بر MST [6]

در [۶] روشی برای لبه یابی تصاویر رنگی پیشنهاد شده که با استفاده از درخت پوشای مینیمال و در فضای رنگ YUV عمل لبه یابی را انجام می‌دهد. علت استفاده از فضای رنگ YUV، یکنواخت بودن این فضای رنگ می‌باشد. روش لبه یابی در این الگوریتم به این صورت است که برای لبه یابی تصویر رنگی، تصویر را از فضای رنگ RGB به فضای رنگ YUV تبدیل کرده و پنجره ای به ابعاد n×n روی آن می-لغزاند . هر پیکسل به عنوان یک نود از گراف و همچنین برداری در فضای YUV در نظر گرفته می‌شود. لبه های بین نودها با استفاده از فاصله اقلیدسی محاسبه شده و سپس درخت پوشای مینیمال با استفاده از الگوریتم کراسکال بدست می‌آید. پس از ایجاد درخت پوشای مینیمال، لبه ماکزیمم (E_8) آن را پیدا کرده و در صورتی که اندازه این لبه مخالف صفر باشد، آن را از درخت پوشای مینیمال حذف می‌کند. در این صورت دو کلاستر مجزا تولید شده که آنها را C_1و C_2 نامیده و سپس مرکزهای دو کلاستر را بدست آورده و فاصله مرکزها را از یکدیگر محاسبه می‌نماید. این فاصله، شدت لبه نقطه مرکز پنجره را مشخص می‌کند.

نقشه لبه در این الگوریتم بصورت ۱ تعریف می‌شود.

$$Edge = \begin{cases} 0 & , if \quad E_8 = 0 \\ a * R & , else \end{cases} \qquad (۱)$$

۳- روش ارائه شده

الگوریتم ارائه شده در [۶] از لحاظ کیفیت لبه های بدست آمده، الگوریتمی کاراست اما نقطه ضعف اساسی آن زمان اجرای بالای الگوریتم می‌باشد. در کاربردهای عملی که نیاز به لبه یابی تصاویر رنگی با سرعت بالا می‌باشد، زمان اجرای بالای این الگوریتم، استفاده از آن را غیر ممکن می‌سازد. علت بالا بودن زمان اجرای الگوریتم، استفاده از درخت پوشای مینیمال و کلاستربندی نودهاست. هدف از ایجاد درخت پوشای مینیمال و کلاستربندی نودها، محاسبه فاصله مراکز کلاسترهاست؛ زیرا این فاصله، شدت لبه را در نقطه مرکز پنجره مشخص می‌کند.

ما در این مقاله روشی را ارائه نموده ایم که شدت لبه را بگونه‌ای محاسبه می‌کند که زمان اجرای الگوریتم به میزان بسیار زیادی کاهش می‌یابد. در این مقاله تمرکز بر روی بالابردن سرعت اجرای الگوریتم می‌باشد.

روش کار به صورت زیر می‌باشد:

۱. تبدیل تصویر از فضای رنگ RGB به فضای رنگ YUV
۲. لغزاندن پنجره ای به ابعاد n×n بر روی تصویر که با این کار n^2 پیکسل زیر پنجره قرار می‌گیرد.
۳. در نظر گرفتن هر پیکسل به عنوان یک نود از گراف و همچنین برداری در فضای YUV
۴. محاسبه فاصله اقلیدسی بین نودها

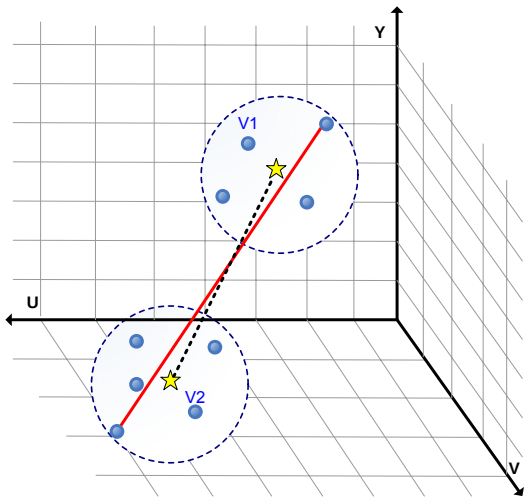
تا این مرحله همانند الگوریتم [۶] عمل نمودیم. در[۶] در این مرحله با استفاده از الگوریتم کراسکال درخت پوشای مینیمال تشکیل داده و سپس با حذف لبه E_8 از درخت پوشای مینیمال دو کلاستر ایجاد می‌شد. سپس با بدست آوردن مراکز کلاسترها و محاسبه فاصله مراکز از یکدیگر شدت لبه نقطه وسط پنجره بدست می‌آمد. حال در این مقاله برای کاهش زمان اجرای الگوریتم روش جدیدی برای بدست آوردن شدت لبه ارائه می‌نماییم. این تکنیک در ادامه مراحل قبل، به صورت زیر می‌باشد:

۵. بزرگترین فاصله (لبه ماکزیمم) بین نودها را پیدا می‌کنیم.
۶. نودهای دو سر این لبه را V_1 و V_2 می‌نامیم. نود V_1 را عنصر پایه کلاستر ۱ و نود V_2 را عنصر پایه کلاستر ۲ در نظر می‌گیریم.
۷. فاصله بقیه نودها را از این دو نود بدست می-آوریم.
۸. عضویت هر نود را بصورت زیر تعیین می‌کنیم؛ فاصله هر نود با نودهای V_1 و V_2 (عناصر پایه کلاسترها) محاسبه شده، در صورتی که فاصله نود مورد نظر از V_1 کمتر از فاصله آن از V_2 باشد، آن نود عضو کلاستر ۱ شده و در غیر این صورت عضو کلاستر ۲ می‌شود.
۹. با بدست آمدن عناصر کلاسترها و با استفاده از میانگین گیری، مراکز دو کلاستر را بدست می-آوریم.

۱۰. فاصله مراکز کلاسترها را با استفاده از رابطه اقلیدسی بدست آورده که این فاصله شدت لبه نقطه مرکز پنجره را مشخص می‌کند.

با این روش زمان اجرای الگوریتم به میزان بسیار زیادی کاهش یافته است.

در شکل ۱ مثالی از الگوریتم پیشنهادی با در نظر گرفتن n=۳ نمایش داده شده است.



شکل۱: مثالی از الگوریتم پیشنهادی

در این شکل، نودها با دایره های توپر ، مراکز کلاسترها با ستاره، بزرگترین لبه با خط ممتد و فاصله مراکز کلاسترها با خط چین مشخص شده است. دو دایره خط چین نیز نودهای متعلق به دو کلاستر را مشخص می‌کنند. در این شکل برای واضحتر بودن مساله، به علت زیاد بودن تعداد لبه ها، فقط لبه ماکزیمم نشان داده شده است.

۴- Automatic Fast Entropy Thresholding Algorithm

در این مقاله ما برای بدست آوردن مقدار آستانه بهینه از تکنیک Entropy Thresholding استفاده می‌کنیم. فرض می‌کنیم در نقشه لبه، ماکزیمم محلی شدت لبه پیکسل ها، در بازه $[0,M]$ قرار داشته و f_i پیکسل وجود دارند که شدت لبه آنها i است($i \in [0,M]$). مقدار آستانه T به صورت توزیع احتمال برای پیکسل های لبه و غیر لبه تعریف می‌شود.

برای پیکسل های غیر لبه، احتمال به صورت (۲) تعریف می‌شود.

$$P_n(i) = f_i \Big/ \sum_{h=0}^{T} f_h \qquad 0 \le i \le T \qquad (۲)$$

که در آن $\sum_{h=0}^{T} f_h$ مجموع تعداد پیکسل هایی است که شدت لبه آنها در بازه $[0,T]$ قرار دارد. احتمال برای پیکسل های لبه نیز به صورت (۳) تعریف می‌شود.

$$P_e(i) = f_i \Big/ \sum_{h=T+1}^{M} f_h \qquad T+1 \le i \le M \qquad (۳)$$

که در آن $\sum_{h=T+1}^{M} f_h$ مجموع تعداد پیکسل هایی است که شدت لبه آنها در بازه $[T+1,M]$ قرار دارد.

آنتروپی برای این دو نوع پیکسل به صورت (۴) و (۵) تعریف می‌شود.

$$H_n(T) = -\sum_{i=0}^{T} P_n(i) \log P_n(i) \qquad (۴)$$

$$H_e(T) = -\sum_{i=T+1}^{M} P_e(i) \log P_e(i) \qquad (۵)$$

بنابراین مقدار آستانه بهینه $\hat{T}$ که از آن برای طبقه بندی پیکسل های لبه و غیر لبه استفاده می‌شود، به صورت (۶) محاسبه می‌گردد.

$$H(\hat{T}) = \max_{T=0,1,\ldots,M} \{H_n(T) + H_e(T)\} \qquad (۶)$$

برای پیدا کردن ماکزیمم محلی(۶)، هزینه محاسبه $O(M^2)$ بوده که علت آن تکرار محاسبات است. محاسبات در مرحله T+1 شامل محاسبات مرحله T می‌باشد؛ بنابراین با انجام محاسبات بصورت بازگشتی، هزینه به O(M) کاهش می-یابد[۷].

۵- نتایج الگوریتم پیشنهادی

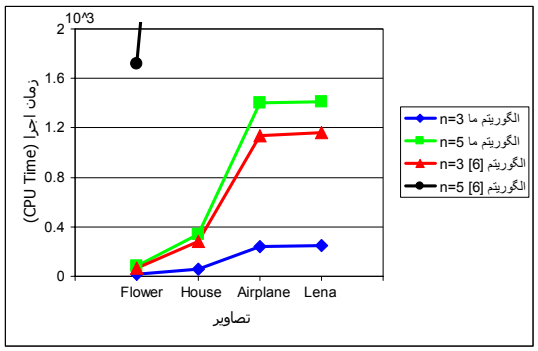
در این بخش به شبیه سازی الگوریتم [۶] که در بخش ۲ بیان شد و الگوریتم پیشنهادی می‌پردازیم. همانطور که در بخش های قبلی ذکر شد مشکل الگوریتم ارائه شده در [۶] زمان بالای اجرای الگوریتم است. ما در این قسمت، زمان CPU را به عنوان زمان اجرای الگوریتم در نظر گرفته و آن را در شرایط یکسان برای هر دو الگوریتم محاسبه نموده ایم.

جدول۱: مقایسه زمان اجرای الگوریتم پیشنهادی با الگوریتم [6]

تصویر	سایز تصویر	CPU Time الگوریتم پیشنهادی (S)		CPU Time الگوریتم [۶] (S)	
		n=3	n=5	n=3	n=5
Flower	128×128	15.3910	81.5310	69.3130	1.7157e+003
House	256×256	60.9220	341.4220	284.9530	7.0211e+003
Airplane	512×512	243.1560	1.4015e+003	1.1412e+003	2.9344e+004
Lena	512×512	245.4690	1.4121e+003	1.1609e+003	2.9763e+004

تصاویر استفاده شده به غیر از تصویر 'Lena' تصاویری است که در [۶] از آنها استفاده شده است. به علت استفاده بسیاری از الگوریتم های لبه یابی از تصویر 'Lena' ما نیز در این مقاله این تصویر را به بقیه تصاویر افزودیم. در این پژوهش شبیه سازی را در دو حالت n=۳ و n=۵ انجام داده ایم که نتایج آن در جدول ۱ نمایش داده شده است. همچنین مقدار پارامتر a در نقشه لبه که در فرمول (۱) بیان شد برابر ۴ در نظر گرفته شده است. شکل ۲ نمودارهای زمان CPU را برای دو الگوریتم پیشنهادی و ارائه شده در [۶] با توجه به مقادیر موجود در جدول ۱ نشان می‌دهد. همانطور که در جدول ۱ و شکل ۲ مشاهده می‌شود الگوریتم پیشنهادی توانسته است زمان CPU را بسیار کاهش دهد.

تصاویر حاصل از اعمال الگوریتم پیشنهادی و الگوریتم [۶] در شکل های ۳ تا ۱۸ مشاهده می‌شود. شکل۳ تصویر 'Flower' را نشان می‌دهد. شکل ۴ تصویر حاصل از اعمال الگوریتم پیشنهادی بر روی شکل ۳ بوده و شکل ۵ تصویر



شکل۲: مقایسه زمان اجرای الگوریتم پیشنهادی با الگوریتم [6]

حاصل از اعمال الگوریتم [۶] می‌باشد. شکل ۶ نیز حاصل از اعمال تکنیک ارائه شده در بخش ۴ می‌باشد. کلیه این عملیات بر روی تصاویر 'House'، 'Airplane' و 'Lena' نیز انجام شده که نتایج آن در شکل های ۷ تا ۱۸ نمایش داده شده است.

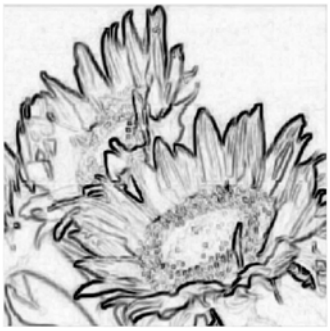
همانطور که در بالا نیز ذکر شد در این مقاله تمرکز بر روی بهبود زمان اجرای الگوریتم می‌باشد؛ اما لازم است که به بررسی کیفیت تصاویر حاصل از دو الگوریتم نیز بپردازیم. برای مقایسه اینکه الگوریتم پیشنهادی چقدر توانسته لبه های تصویر را مشابه الگوریتم ارائه شده در [۶] تشخیص دهد به روش زیر عمل نموده ایم. در ابتدا بر روی هر دو تصویر، الگوریتم Automatic Fast Entropy Thresholding را اعمال نموده و سپس با مقایسه دو تصویر حاصل با یکدیگر، تعداد پیکسل هایی که در دو تصویر یکسان نیستند را به دست آورده ایم. پس از آن درصد این تفاوت را محاسبه نموده ایم. این عملیات در دو حالت n=۳ و n=۵ انجام شده و نتایج آن در جدول ۲ نشان داده شده است.

جدول۲: مقایسه کیفی نتایج دو الگوریتم

تصویر	درصد اختلاف	
	n=3	n=5
Flower	1.0798%	0.9845%
House	0.4715%	0.5623%
Airplane	0.5616%	0.4819%
Lena	0.6255%	0.6714%
متوسط	0.6846	0.675

شکل۳: تصویر اصلی ’Flower‘

شکل۷: تصویر اصلی ’House‘



شکل۴: اعمال الگوریتم پیشنهادی

شکل۸: اعمال الگوریتم پیشنهادی

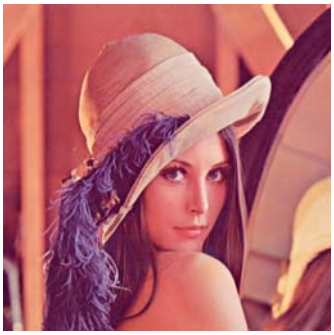
شکل۵: اعمال الگوریتم ارائه شده در [6]

شکل۹: اعمال الگوریتم ارائه شده در [6]

شکل۶: اعمال تکنیک Fast Entropy Thresholding بر روی شکل۴

شکل۱۰: اعمال تکنیک Fast Entropy Thresholding بر روی شکل۸

شکل۱۱: تصویر اصلی ’Airplane‘



شکل۱۵: تصویر اصلی ’Lena‘

شکل۱۲: اعمال الگوریتم پیشنهادی

شکل۱۶: اعمال الگوریتم پیشنهادی

شکل۱۳: اعمال الگوریتم ارائه شده در [6]

شکل۱۷: اعمال الگوریتم ارائه شده در [6]

شکل۱۴: اعمال تکنیک Fast Entropy Thresholding بر روی شکل۱۲

شکل۱۸: اعمال تکنیک Fast Entropy Thresholding بر روی شکل۱۶

همانطور که در جدول ۲ مشاهده می‌شود، درصد اختلاف بین تصاویر بدست آمده توسط الگوریتم پیشنهادی با الگوریتم [۶] بسیار ناچیز بوده و به طور میانگین برای n=۳ برابر ۰٫۶۸۴۶ و برای n=۵، ۰٫۶۷۵ می‌باشد. این بدان معناست که از لحاظ کیفی بین نتایج حاصل از الگوریتم پیشنهادی و الگوریتم [۶] مقدار ناچیزی اختلاف یعنی کمتر از ۱ درصد اختلاف وجود دارد. بنابراین الگوریتم پیشنهادی توانسته است با حفظ کیفیت، عملیات لبه یابی را در مدت زمان بسیار کمتری انجام دهد.

۶- نتیجه گیری

زمان اجرای الگوریتم های لبه یابی در کاربردهای واقعی بسیار حائز اهمیت است. الگوریتم لبه یابی ارائه شده در [۶] قادر به تشخیص لبه های تصاویر رنگی با کیفیت بالا بوده اما نقطه ضعف اساسی آن سرعت پایین اجرای الگوریتم می‌باشد. این الگوریتم با استفاده از درخت پوشای مینیمال، در فضای رنگ YUV عملیات لبه یابی را انجام می‌دهد. نود ها به عنوان برداری در فضای رنگ YUV در نظر گرفته شده و با استفاده از درخت پوشای مینیمال و با حذف لبه ماکزیمم از آن عملیات کلاستر بندی روی آن انجام می‌شود. هدف از کلاستربندی، پیدا کردن شدت لبه نقطه مرکز پنجره می‌باشد. این روش به علت تولید درخت پوشای مینیمال و نحوه کلاستر بندی، زمان اجرای بالایی را به سیستم تحمیل می‌کند. ما در این مقاله روش کلاستر بندی جدیدی را ارائه نموده ایم که زمان اجرای الگوریتم را به میزان بسیار زیادی کاهش می‌دهد. در الگوریتم پیشنهادی به جای تشکیل درخت پوشای مینیمال، بزرگترین لبه را یافته و نودهای دو سر آن را به عنوان نودهای پایه کلاسترها در نظر می‌گیریم. بقیه نودها را با توجه به فاصله هر نود با نود پایه کلاستر، متعلق به کلاستر ۱ یا کلاستر ۲ در نظر می‌گیریم. با این روش زمان اجرای الگوریتم به میزان قابل توجهی کاهش می‌یابد. در نهایت دو الگوریتم را از لحاظ زمان اجرا و کیفیت لبه های بدست آمده با یکدیگر مقایسه نمودیم؛ نتایج بدست آمده نشان می‌دهد که الگوریتم پیشنهادی توانسته است به میزان قابل توجهی زمان اجرا را کاهش دهد در حالی که اختلاف بین لبه های بدست آمده توسط الگوریتم ارائه شده در [۶] و الگوریتم پیشنهادی کمتر از ۱ درصد می‌باشد. بنابراین الگوریتم پیشنهادی، لبه های تصاویر رنگی را با کیفیت مطلوب و در زمان کوتاهی تشخیص می‌دهد.

مراجع

[1] *S. Y. Zhu, K. N. Plataniotis, A. N. Venetsanopoulos, "Comprehensive analysis of edge detection in color image processing," Opt. Eng., vol. 38,no. 4, pp. 612-625, April, 1999.*

[2] *A. Koschan, M. Abidi, "Detection and classification of edges in color images," IEEE Signal Processing Magn., vol. 22, no. 1, pp. 64-73,January 2005.*

[3] *A. Koschan, "A comparative study on color edge detection," Proceedings,2nd Asian Conf on Computer Vision ACCV'95, vol. 3, pp. 574-578, December, 1995.*

[4] *P. E. Trahanias, A. N. Venetsanopoulos, "Vector order statistics operators as color edge detectors," IEEE Trans. Syst. Man Cybern., vol. 26, no. 1,pp. 135-143, February 1996.*

[5] *F. Russo, A. Lazzari, "Color edge detection in presence of Gaussian noise using nonlinear prefiltering," IEEE Trans. Inst. Meas., vol. 54, no.1, pp. 352-358, February, 2005.*

[6] *R. Ji, B. Kong, F. Zheng, J. Gao, "Color Edge Detection Based On YUV Space and Minimal Spanning Tree" Proceeding IEEE. Weihai, Shandong, China, pp. 941-945, August 2006.*

[7] *J. P. Fan, W. G. Aref, M. S. Hacid, et al. "An improved automatic isotropic color edge detection technique," Pattern Recognition Letters,. vol. 22, no. 13, pp. 1419-1429, November 2001.*