طراحی خودکار قلمهای فارسی

محمد قدسی* و کیارش بازرگان**

دانشکدهٔ مهندسی کامپیوتر، دانشگاه صنعتی شریف

(دریافت مقاله: ۷۸/۱۰/۲۲ – دریافت نسخهٔ نهایی ۸۰/٤/۱٦)

چکیده – *در این مقاله یک روش سریع و خودکار برای طراحی ((علمی)) حروف مختلف خط فارسی پیشنهاد می شود. منظور از طراحی علمی حروف، استفاده از منحنیهای ریاضی و پارامترهای مختلف در طراحی است به طوری که بتوان اندازه های مختلف یک حرف و سبکهای متفاوت نگارش آن را با کیفیت عالی و تنها با تغییر مقادیر چند پارامتر به سرعت تولید کرد. متافونت، زبانی برای طراحی علمی فونتهاست که در این مقاله از آن استفاده می شود. با استفاده از امکان ((قلم)) دراین زبان، می توان حرکت قلم درشت یک استاد خوشنویس را شبیه سازی کرد. استفادهٔ مستقیم از متافونت برای طراحی کیفی فونتها کاری طولانی و همراه با مشکلات متعددی است. در این مقاله الگوریتمهایی پیشنهاد می‌شود تا بتوان تصویر پویش شدهٔ یک حرف را به صورت خودکار به برنامهٔ متافونتی که آن حرف را تولید می کند تبدیل کرد. نرم افزاری بر اساس این الگوریتم طراحی شده است که به صورت گرافیکی و با محاوره با کاربر این کار را انجام می دهد.*

واژگان کلیدی: *طراحی قلم، متافونت، منحنیهای بزیه، طراحی خودکار*

Automatic Design of Persian Typefaces

M. Ghodsi and K. Bazargan

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

ABSTRACT- *In this paper, a fast method for automatic generation and scientific design of Persian letters is proposed. Scientific typeface design is an approach in which fonts are described by mathematical curves with well-defined parameters, where these parameters can be automatically tuned. METAFONT is a language suitable for the type of design used in this work. This language is particularly useful in designing Persian fonts because it can be used to simulate the pen movements of a calligrapher through automatic conversion of the scanned bitmap image of a font into a METAFONT program, which can in turn, produce the font at a high quality. A complete software has been implemented based on these algorithms that works interactively with the user to facilitate the font design.*

Keywords: *Font design, METAFONT, Bezier curves, automatic design*

* – دانشیار ** – کارشناسی ارشد

۱-معرفی

یکی از مراحل مهم تولید هر سیستم حروف چینی فارسی، طراحی مجموعه ای کامل از خانواده های مختلف قلم یا فونت[۱] با کیفیت عالی است. منظور از یک خانواده از فونت، تمام حروف الفباست که با سیاق[۲] یکسان نوشته شده اند. مثالهایی از خانواده های قلم، ((نازک))، ((سیاه))، ((ایتالیک)) و ((ایرانیک)) هستند. یکی از مهمترین ویژگیهایی که در طراحی خانواده های قلم باید در نظر گرفت، همگن بودن حروف است. قسمتهای مشترک حروف شبیه به هم در یک خانواده، باید دقیقاً مثل هم تعریف شده باشند تا متنی که با آن قلم نوشته می شود، همگن باشد. همچنین تغییراتی که یک حرف از یک خانواده را به همان حرف در خانوادهٔ دیگر تبدیل می کنند، باید برای تمام حروف، مشابه باشند.

روشهای مختلفی برای طراحی فونت وجود دارد که از آن میان می توان به روشهای زیر اشاره کرد :

۱- طراحی تک تک حروف توسط خوشنویس و خواندن آنها توسط پویشگر[۳] و ذخیره سازی آنها به صورت بیت نگاشت[٤]. در این روش نمی توان تضمین کرد که حروف به صورت همگن تعریف شوند. همچنین حروف تولید شده توسط این روش را نمی توان به راحتی بزرگ و کوچک کرد.

۲- تعریف هر حرف به طور جداگانه و به صورت مجموعه‌ای از منحنیهای درجهٔ ۲ یا ۳ و نگهداری آنها به صورت True Typeو یا ..Post Script در این روش امکان استفاده از پارامترهایی که ویژگیهای حروف را به صورت کلی توصیف کنند، وجود ندارد.

۳- استفاده از روشهای طراحی هندسی و تعریف سطح بالای حروف با پارامترهای پایدار. پارامتری پایدار است که با تغییر اندک آن، منحنیهای تشکیل دهندهٔ حروف، تغییر ناگهانی نکنند. تعداد این پارامترها و وابستگی آنها به هم نباید خیلی زیاد باشد.

با توجه به ویژگیهای روش آخر، دراین مقاله از آن استفاده می‌کنیم. یکی از بهترین ابزار طراحی فونتها با چنین روشی زبان متافونت[٥] است [۱]. متافونت یک زبان برنامه نویسی توصیفی[٦] و سطح بالاست که اجازه می دهد حروف را با استفاده ار پارامترهای مختلف تعریف و با تغییر یک یا چند پارامتر می‌توان خانواده‌های مختلف حروف را تولید کرد. در متافونت می توان قطعه های مشترک، و یا خیلی شبیه، حروف متفاوت را به صورت جدا تعریف کرد و سپسس با ترکیب این قطعه ها با قطعه های غیر مشترک، حروف همگنی را تولید کرد.

متافونت زبانی توصیفی است که در سال ۱۹۸۲ توسط دانلد کنوث[۷] برای طراحی فونتهای سیستم حروف چینی TeX ابداع شد. مهمترین قابلیت متافونت این است که به برنامه نویس اجازه می دهد معادله های توصیف کنندهٔ مکان و زاویه‌ها را به صورت ((ضمنی[۸])) بنویسد. وقتی تعداد معادله‌ها به اندازهٔ کافی رسید، متافونت مقدار متغیرهای مجهول را محاسبه می کند. متافونت به جای صدا کردن زیر برنامه، از بسط ماکروها استفاده می کند. این مسئله، خطایابی برنامه را مشکل می سازد.

با وجود این که زبان متافونت امکانات بسیار مفیدی در اختیار قرار می دهد، استفاده از آن بسیار مشکل و نیازمند دقت و صرف وقت فراوان است. در این مقاله مباحث نظری و الگوریتمهایی را پیشنهاد می کنیم تا بتوان تصویر پویش شدهٔ یک حرف را به صورت خودکار به برنامهٔ متافونتی که آن حرف را تولید می کند تبدیل کرد. بر این اساس محیط نرم افزاری محاوره‌ای[۹] را طراحی کرده ایم که در این مقاله گزارش می‌شود.

۲-پایه‌های نظری و الگوریتمها

نرم افزاری که طراحی شده، مجموعه ای از ماکروها به زبان متافونت و یک محیط گرافیکی تحت ((ویندوز)) است که امکانات مناسب برای تولید نیمه خودکار برنامه های متافونت را در اختیار می گذارد. ابتدا، کاربر تصویر پویش شدهٔ حرف مورد نظر خود را تهیه می کند. این تصویر می تواند خاکستری یا سیاه وسفید باشد. سیستم به دقت تصویر خاصی وابسته نیست ولی معمولاً بهتر است ابعاد مستطیل محیطی حرف، حدود ۱۰۰۰ نقطه در ۱۰۰۰ نقطه باشد. این امر باعث

[DOR: 20.1001.1.22287698.1380.20.1.8.3]

می شود ویژگیهایی از منحنی مرزی مثل زاویه، انحنا و موقعیت، با دقت مناسبی تشخیص داده شود. شکل(۱) تصویرهای پوشش شدهٔ حرف ((ب)) و ((ن)) را نشان می‌دهد.

سپس نویز تصویر برطرف می شود و منحنی مرزی آن به دست می آید. کاربر می تواند منحنی مرزی را اصلاح کند : نقاطی را حذف یا اضافه کند و یا آستانهٔ حذف نویز را تغییر دهد. همچنین، کاربر می تواند مشخصات منحنی، از جمله زاویهٔ مماس بر آن، را در هر نقطه مشاهده کند. تمام اعمال خودکار برنامه، بر روی منحنی مرزی بستهٔ حرف، و یا قطعه منحنیهای وابسته به هم انجام می شود. منحنی مرزی بسته، توسط روشهای خودکار و نیز، در صورت لزوم با دخالت کاربر، به قطعه هایی تقسیم می شود و هر قطعه، با یک منحنی بزیه تقریب زده می شود. سپس مکانهای عبور قلم درشت، تعیین می‌شوند و با توجه به مشخصات منحنی در آن نقاط، برنامهٔ متافونت، تولید می شود. در انتها، خروجی متافونت پس از اجرای برنامهٔ تولید شده با تصویر اولیه مقایسه می شود تا کاربر اشکالات برنامهٔ متافونت را برطرف کند.

جزییات مراحل بالا در ادامه می‌آید.

۲-۱- حذف نویز

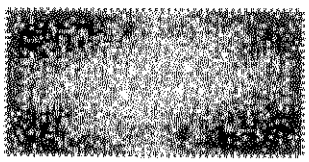
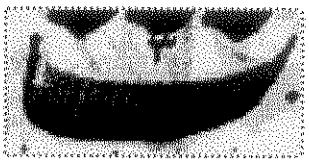
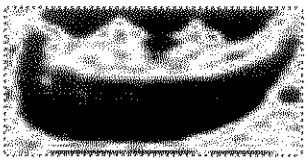
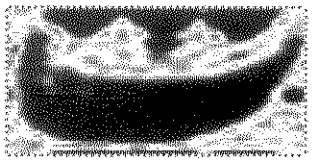
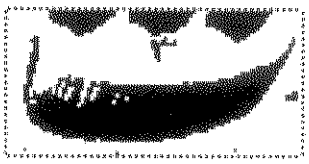
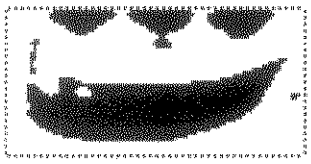
برای حذف نویز تصویر پوشش شده، روشهای زیر مورد بررسی قرار گرفتند.

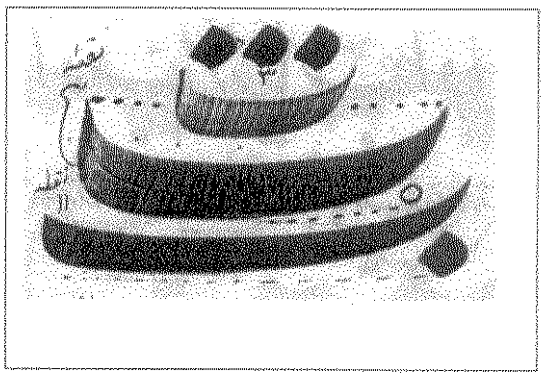
۱- تصویر خاکستری حرف را به صورت آرایه ای دو بعدی در نظر می گیریم که نمایان گر تابعی دوبعدی است. تبدیل فوریهٔ دوبعدی [۵] (2Dim DFT) این تصویر را محاسبه می کنیم و آن را از یک فیلتر پایین گذر ایده آل [٤] می گذرانیم و تبدیل فوریهٔ معکوس نتیجه را محاسبه می کنیم. حاصل، تصویری خاکستری خواهد بود که ((تار)) شده است. تصویر حاصل را با در نظر گرفتن یک آستانهٔ روشنایی، به تصویری سیاه و سفید تبدیل می کنیم. این روش روی تمام تصویر به طور یکسان عمل می کند. برای حذف نویز در قسمتی از تصویر، باید پهنای گذر فیلتر را برای تمام تصویر، کم کنیم. این کار اغلب باعث می‌شود دقت تصویر را در گوشه های حرف یا بقیه قسمتهای ظریف آن، از دست بدهیم. از آنجا که این نوع نویز (یعنی نویز زیاد در یک قسمت از تصویر) در حروف پویش شده معمول است، و نیز به علت کندی ذاتی محاسبات تبدیل فوریه، این روش، روش مناسبی برای فیلتر کردن تصاویر ما نیست. مثالهایی از فیلتر کردن تصاویر با این روش را می توانید در جدول (۱) مشاهده کنید. همان طور که می بینید، با کم شدن پهنای فیلتر، نویز بدنه در قسمت چپ، حرف از بین می‌رود ولی تیزی گوشهٔ بالا – راست نیز تحلیل می شود.

در دو روش بعد، ابتدا توسط الگوریتم کانتور چهارنقطه‌ای با آستانهٔ تغییر روشنایی بین ۵۰ تا ۸۰ درصد، مرز حروف را به دست می آوریم. نمونه های منحنی مرزی به دست آمده از روش کانتور را در شکلهای (۲) ملاحظه می کنید. از آنجا که الگوریتم کانتور تقریباً همیشه، یک مسیر بسته با انشعاباتی مانند آنچه در شکل (۳) آمده است می‌سازد و این انشعابات مشکلاتی را ایجاد می کند، خروجی الگوریتم کانتور را توسط یک الگوریتم جدا کنندهٔ مسیرهای انشعابی، ساده می کنیم و سپس عملیات فیلتر کردن نویز را روی منحنی مرزی حاصل انجام می دهیم. الگوریتم جدا کنندهٔ مسیرهای انشعابی را در انتهای این بخش آورده ایم.

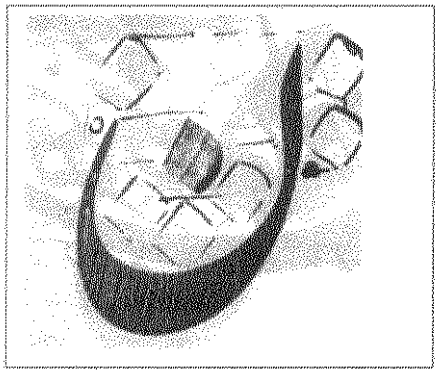
۲- منحنی مرزی را به توابع $Y = f_1(X)$ ، $X = g_1(Y)$، $Y = f_2(X)$ ، $X = g_2(Y)$ تقسیم می کنیم به طوری که این توابع، تمام مرز منحنی را بپوشانند و به صورت یکی در میان توابعی از X و Y باشند. سپس تبدیل فوریهٔ هر قسمت را محاسبه می کنیم و نویز هر تابع را به کمک فیلتر ایده آل پایین گذر، برطرف می کنیم. به این ترتیب می توان هر قسمت از منحنی را که تابعی فقط از X یا فقط از Y است، با توجه به مقدار نویز، از فیلتری با پهنای باند مناسب عبور داد و نویز آن را حذف کرد. اشکال این روش این است که، به علت ((بسته)) نبودن منحنی f ها و g ها و در نتیجه تناوبی نبودن آنها، لبه های انتهای توابع به صورت نوسانی در می آیند و عملاً کارایی این روش از بین می رود و منحنی را در نقاط اتصال این توابع،

جدول ۱- مراحل مختلف و مقایسهٔ نحوهٔ فیلترگیری در روش اول

مراحل روش اول	فیلتر با پهنای گذر ۱۰	فیلتر با پهنای گذر ۱۰	فیلتر با پهنای گذر ۱۰
تبدیل فوریه فیلترشده			
تصویر خاکستری حاصل از تبدیل فوریه معکوس			
تصویر سیاه‌وسفید حاصل از قطع تصویر فیلترشده			

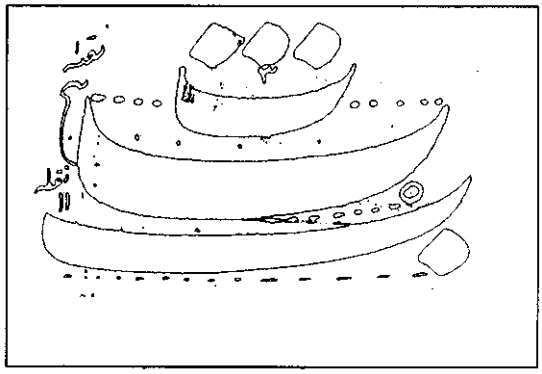


الف – منحنیهای "ب" ۳، ۵ و ۱۱ نقطه

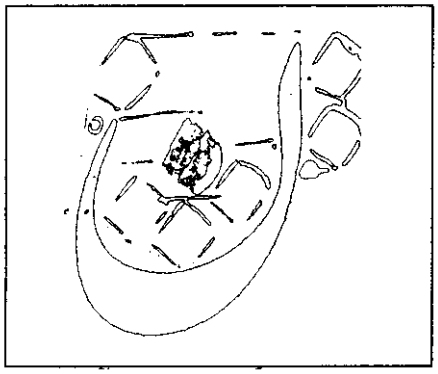


ب – حرف "ن" پویش شده

شکل ۱- تصویرهای پویش شدهٔ دو حرف (ب) و (ن)

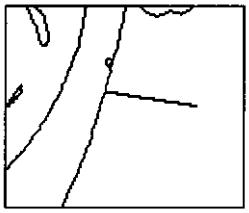


الف – حرف (ب) پس از الگوریتم کانتور با آستانهٔ ۰/۷۸



ب – حرف (ن) پس از الگوریتم کانتور با آستانهٔ ۰/۷۰

شکل ۲- تشخیص مرز با الگوریتم کانتور چهارنقطه‌ای



شکل ۳- انشعاب الگوریتم کانتور

ناپیوسته می کنند. نمونه هایی از منحنی پایین حرف ((ب)) پنج نقطه در شکل (٤) آمده است. در شکل (٥) تبدیل فوریهٔ همین منحنی، قبل از فیلتر شدن آمده است.

۳- منحنی را به صورت مجموعه ای از نقطه ها، با مختصات صحیح، در می آوریم : (N تعداد نقاط منحنی مرزی است.)

$$C = \{x(t), y(t)\}, t = 0, 1, \ldots, N$$

با توجه به اینکه منحنی مرزی، یک منحنی بسته است، می توان آن را به صورت توابع متناوب x و y تعریف کرد.

$$\tilde{x} = \begin{cases} x(t) & 0 \le t \le N \\ x(t \bmod N) & \text{otherwise} \end{cases}$$

حال تابع گاوسی g را به صورت زیر تعریف می کنیم [٦] : (σ پهنای منحنی را تغییر می دهد.)

$$g_\sigma = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{t^2}{2\sigma^2}}$$

تابع گاوسی را با توابع متناوب $\bar{x}$ و $\bar{y}$ کانوالو می کنیم [٤]، تا توابع متناوب نرم و بدون نویزی به دست آیند. این توابع، دارای مختصات حقیقی هستند و نویز آنها به اندازهٔ مطلوبی حذف شده است.

$$X_\sigma = \tilde{x}(t) g_\sigma(t) = \sum_{u=\infty}^{\infty} \tilde{x}(t-u) g_\sigma(u)$$

$$Y_\sigma = \tilde{y}(t) g_\sigma(t) = \sum_{u=\infty}^{\infty} \tilde{y}(t-u) g_\sigma(u)$$

از آنجایی که مقادیر g_α برای t های بزرگتر از ٥ خیلی کم است. می توانیم محدودهٔ محاسبهٔ convolution را از ٥- تا ٥+ بگیریم و در نتیجه، سرعت محاسبات را بسیار بالا ببریم. روش فیلتر گیری گاوسی، به دلایل زیر برای کاربرد مورد نظر بسیار مناسب است :

۱- به صورت حلقه ای عمل می کند. یعنی اگر قسمتی از تصویر دارای نویز زیاد باشد، این نویز روی چگونگی فیلتر شدن منحنی مرزی تأثیری ندارد.

۲- محاسبات آن بسیار سریع است.

۳- حاصل این روش، یک منحنی با مختصات حقیقی است.

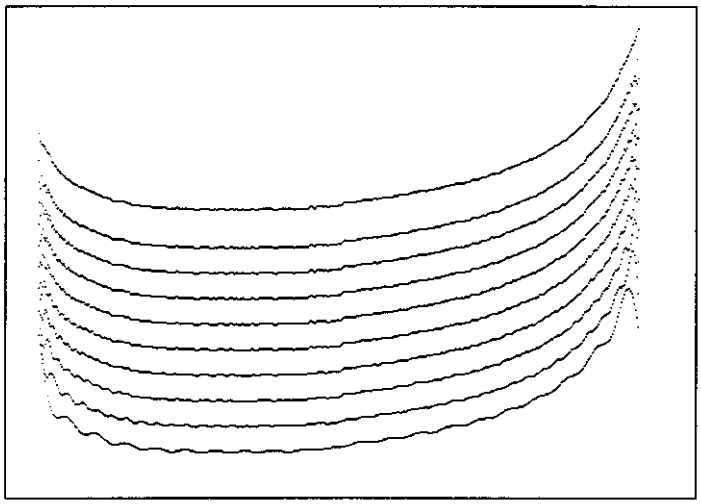
٤- نقاط به صورت پیوسته تغییر می کنند و خطای کوانتیزه بودن تصویر پویش شدهٔ اولیه را ندارند. این خاصیت باعث می‌شود محاسبات مراحل بعدی، مثل محاسبهٔ زاویه در نقاط مختلف، بدون نیاز به استفاده از روشهای تقریب زننده و روشهای محاسبات عددی انجام گیرد.

نمونهٔ فیلتر شدهٔ شکل (٦- الف) حرف ((ن)) با پهنای σ =5 نشان داده شده است. تابعی که عمل فیلتر گیری روی آن انجام شده است، در شکل (۲-ب) آمده است. به حذف نوسانات زیر روی لبه های منحنی توجه کنید. مقایسهٔ بهتری را شکل (٦- ب) نشان می دهد که منحنی سمت راست، منحنی فیلتر شده است. مقدار اندک تغییر منحنی در گوشه های تیز قابل توجه است.

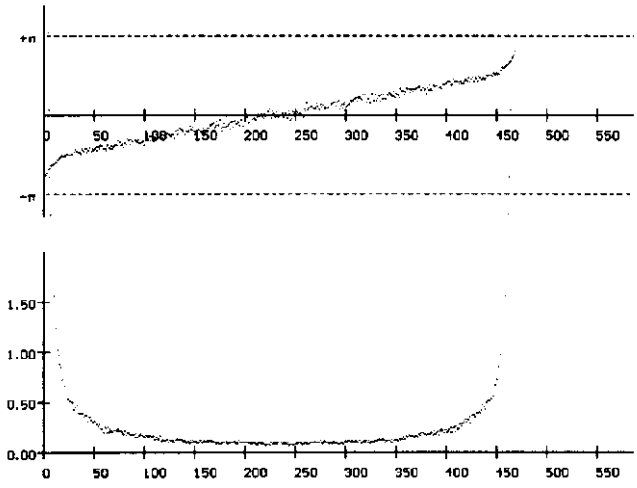
۳-الگوریتم جداکنندهٔ مسیر انشعابی

دراین بخش، الگوریتم جداکنندهٔ مسیرهای انشعابی حاصل از الگوریتم کانتور را توضیح می دهیم. چنانچه اشاره شد، تصویر ابتدا توسط الگوریتم کانتور مرزیابی می شود و اغلب شامل مسیرهای انشعابی است، مانند آنچه در شکل (۳) آمده است. بنا براین، پس از اجرای الگوریتم کانتور، الگوریتم زیر را اجرا می کنیم:

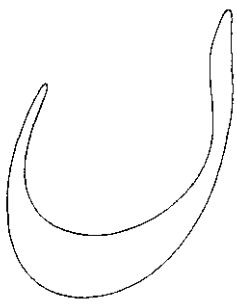
۱- مستطیل حاوی تصویر را به صورت آرایه ای دو بعدی از نقاط به نام Pic در نظر بگیر. مجموعهٔ C که در ابتدا تهی است منحنیهای جدا از هم را ذخیره می کند. این مجموعه، در پایان، خروجی الگوریتم خواهد بود.



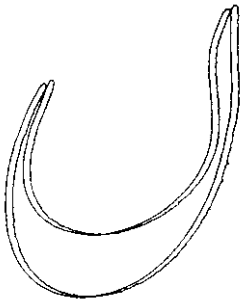
شکل ٤- منحنی پایینی (ب) پنج نقطه‌ای پس از عبور فیلترهای ایده‌آل پایین گذر با پهنای گذر بین ۲۰ تا ۱۸۰



شکل ٥- تبدیل فوریه منحنی پایینی (ب) پنج نقطه‌ای



شکل (٦-الف)- حرف (ن) که از فیلتر گاوسی گذشته است



شکل (٦-ب)- مقایسهٔ منحنی فیلترشده با منحنی اصلی

[DOR: 20.1001.1.22287698.1380.20.1.8.3]

۲- برای تمام نقاط (X0,Y0) Pic از تصویر، اگر سیاه است، آن را سفید کن و الگوریتم « دنبال کردن مسیر ساده[1] » (که در زیر می آید) را با نقطهٔ شروع (X0,Y0) اجرا کن.این فراخوانی باعث می شود که یک یا چند مسیر بدون انشعاب، به C اضافه شود. این مسیرها یک مسیر کانتوری در Pic را افراز می کنند. فرض کنید تعداد کل مسیرهای ساده که از تمام مسیرهای کانتوری به وجود آمده اند، N باشد.

۳- برای مسیرهای Pi، برای ۱ = i و ... ، N -۱

۳-۱- برای مسیرهای Pj، برای j = i + ... و N ،

۳-۱-۱- نقاط ابتدا و انتهای Pi و Pj را دوبه دو از نظر مجاورت با همسایگی $\sqrt{2}$ بررسی کن. (اینکه آیا این دو مسیر از یک نقطهٔ مسیر کانتوری منشعب شده اند.)

۳-۱-۲- اگر مجاور باشند، حتماً یک مسیر سومی هم در همین نقطه، با این دو مسیر، مجاور است. حلقهٔ j را ادامه بده تا هر دوی این مسیرهای منشعب پیدا شوند. این مسیرها را first و second می نامیم. البته مقدارsecond را فقط در صورتی قرار بده که first و second در همسایگی باشند. یعنی حداکثر یک نقطه بین آنها فاصله باشد (همان نقطه ای که انشعاب از آن به وجود آمده است[2]).

۳-۲- اگر دو مسیر پیدا شده است، (یعنی first و second هر دو مقادیر معنی‌دار دارند،) first را برابر مسیر بزرگتر قرار بده.

۳-۳- نقاطfirst را به مسیر Pi، اضافه کن وfirst را از C حذف کن. در نتیجه Pi مسیر بزرگتری شده است در حالی که هنوز یک مسیر ساده و بدون انشعاب است.

٤- نقاط تمام مسیرهای C را سیاه کن که تصویر اولیهٔ Pic تغییر نکند.

این الگوریتم، لزوماً بزرگترین مسیرهای ساده را به دست نمی آورد، چون به صورت کاملاً موضعی سعی می کند مسیر را گسترش دهد. ولی تقریباً همیشه، بزرگترین مسیرها را به دست می آورد. علت این است که در حروف پویش شده انشعابات ناخواسته، معمولاً کوتاه هستند. همچنین تعداد انشعابات مسیر کانتوری در تصویرهای حروف، اغلب کمتر از ۱۰ است.

الگوریتم دنبال کردن مسیر ساده، به صورت زیر است :

۱- مسیر جدیدی به C اضافه کن که فعلاً فقط شامل نقطهٔ CurPt (پارامتر ورودی) است. first-choice و second- choice برابر اولین و دومین نقاطی خواهند بود که در هر لحظه می توان از CurPt به آنها رفت. فعلاً آنها را برابر مقادیر بی معنی قرار بده.

۲- برای هر هشت نقطهٔ مجاور CurPt، اگر نقطهٔ(x,y) Pic سیاه است، آن را درfirst-choice قرار بده و اگر first - choice قبلاً مقدار گرفته است، آن را درsecond -choice قرار بده.

۳-

الف) اگر هیچ کدام از first - choice و second -choice مقدار ندارند، به پایان این مسیر رسیده ایم. از الگوریتم خارج شو.

ب) اگر فقط first - choice مقدار دارد، آن را به مسیر اضافه کن و نقطه را در Pic سفید کن. CurPt را برابرfirst - choice قرار بده و از مرحلهٔ ۲ کار را ادامه بده..

ج) چون هر دوی first-choice و second- choice مقدار دارند، مسیر دارای انشعاب شده است. هر دو نقطه را در Pic سفید کن و الگوریتم دنبال کردن مسیر ساده را یک بار با نقطهٔ first - Point به عنوان نقطهٔ شروع، و بار دیگر با نقطهٔ second- Point اجرا کن. این عمل باعث می شود که از این نقطهٔ انشعاب، سه مسیر ساده تولید شود. از الگوریتم خارج شو.

٤- تقریب قطعه مسیر ساده با منحنی بزیه

در بخش قبل چگونگی تبدیل منحنی مرزی حرف به یک مسیر سادهٔ صاف و بدون نویز را توضیح دادیم. این مسیر ساده

---

۱ مسیری ساده است که هر نقطهٔ آن حداکثر دو همسایه در ۸ نقطهٔ مجاور خود داشته باشد. مسیری که انشعاب ندارد ساده است.

۲ در الگوریتم دنبال کردن مسیر ساده، فقط وقتی از اضافه کردن یک نقطه به مسیر ساده امتناع می‌کنیم که غیر از نقطهٔ قبلی خود در مسیر، یک همسایهٔ دقیقاً مجاور داشته باشد. هر یک از این دو همسایه دقیقاً مجاور، یک مسیر ساده به C اضافه کرده است.

باید با قطعه منحنیهای بزیه‌ای تقریب زده شود که هر زوج، نشانگر مسیرهای چپ و راست حرکت قلم درشت باشد. علت این امر این است که در نهایت، حروف قرار است با متافونت تولید شوند. بنابر این اگر سیستم برای تقریب قسمتی از مرز حرف، حرکت قلمی را تعیین کند که متافونت نتواند آن را رسم کند، عملاً روشهای ارائه شده بدون استفاده خواهند بود. پس روشهای تقسیم منحنی و تقریب و نمایش آن به وسیلهٔ قلم درشت، باید مبتنی بر ویژگیها و قابلیتهای متافونت باشد.

تعیین نقاط تقسیم مسیر ساده به زیر منحنیهای بزیه، بسیار مهم است. اگر این نقاط را به طور مناسبی انتخاب کنیم، تقریب زیر منحنیها با سرعت بیشتری انجام می شود و تعداد زیر منحنیها حداقل می شود. دربارهٔ چگونگی انتخاب این نقاط، در بخش بعد توضیح خواهیم داد. در این بخش، فرض می کنیم این نقاط را انتخاب کرده ایم و می خواهیم یکی از این قطعه‌های مسیر ساده را تقریب بزنیم. همچنین فرض می کنیم این قطعه مسیر، قابل تقریب زدن با یک منحنی بزیه باشد.

منحنی بزیه، مجموعه نقاط منحنی پارامتریک زیر است :

$$Z(t) = (1-t)^3 Z_{m1} + 3(1-t)^2 t Z_{c1} + 3(1-t)t^2 Z_{c2} + t^3 Z_{m2}$$

برای تقریب قطعه مسیری ساده با منحنی بزیه، و با توجه به داشتن نقاط ابتدا و انتهای منحنی بزیه (Z_{m1}, Z_{m2})، کافی است Z_{c2}, Z_{c1} را پیدا کنیم. اگر مشتق منحنی بزیه را در نقاط ابتدا و انتهای آن محاسبه کنیم، فرمولهای زیر به دست می آید:

$$\frac{dz}{dt}\Big|_{t=0} = 3(Z_{c1} - Z_{m2})$$

$$\frac{dz}{dt}\Big|_{t=1} = 3(Z_{m2} - Z_{c2})$$

یعنی اگر بتوانیم مشتق $Z'(t)$ را در نقاط ابتدا و انتهای قطعه منحنی بزیه تعیین کنیم [۲]، می توانیم نقاط کنترلی را به دست آوریم و در نتیجه قطعه مسیر را تقریب بزنیم. اما این کار، ساده نیست. قطعه منحنی بزیه، از مجموعه نقاط X و Y تشکیل شده که فاصلهٔ هندسی آنها، ارتباطی با فاصلهٔ پارامتر t آنها ندارد. به عبارت دیگر، سرعت پیشروی و حرکت منحنی را بر حسب t ، در نقاط مختلف آن، نمی دانیم. (به عنوان مثالی از ارتباط فاصلهٔ فیزیکی با فاصلهٔ پارامتریک نقاط، شکل (۷-الف) را ببینید.) بنابر این نمی توانیم با محاسبهٔ $y'(x)$ (مشتق y برحسب x)، $y'(t)$ را به دست آوریم[۱].

درحقیقت، بردارهای $Z_{c1} - Z_{m1}$ و $Z_{m2} - Z_{c2}$ سرعت حرکت منحنی نسبت به t را در نقاط ابتدا و انتها نشان می‌دهند. پارامتر tension در متافونت، اندازهٔ این بردارها را کنترل می کند : با کم شدن این پارامتر، فاصلهٔ نقاط کنترلی، در همان راستا که بودند، بیشتر می شود. در شکل (۷-ب) مثالی از اثر این پارامتر بر روی نقاط کنترل آمده است.

در این شکل، منحنی ضخیمتر، با عبارت

draw Zm1 dir 5 .. dir-70 Zm2;

رسم شده است و منحنی نازکتر عبارت

draw Zm1 dir 5 .. tension 1 and 2.. dir – 70 Zm2 ;

رسم شده است. تأثیر tension بر روی نقطهٔ کنترلی v ، نقطهٔ w را تولید کرده است که به نقطهٔ Zm2 با نسبت ۱ به ۲ نزدیکتر است.

توصیف ZO WO..tension α and β … W1 Z1 به زبان متافونت را در نظر بگیرید و فرض کنید مقادیر تمام پارامترها معلوم باشد. (W0 و W1 بردارهای جهت منحنی بزیه در دو سر آن هستند.) در ابتدا نشان می دهیم متافونت چگونه این پارامترها را به نقاط کنترلی u و v تبدیل می کند به طوری که عبارت بالا با عبارت Z_0 … controls u and v … Z_1 معادل باشد. بعد از آن،الگوریتمی برای تقریب قطعه مسیر ساده با یک منحنی بزیه، ارائه خواهیم داد.

اگر زوایای $\varnothing$ و θ را به صورت زیر تعریف کنیم [۲].

$$\phi = \arg\left((Z_1 - Z_0)/\omega_1\right) \qquad (۱)$$

$$\theta = \arg\left(\omega_0/(Z_1 - Z_0)\right) \qquad (۲)$$

---

۱. البته یک روش تقریب این است که در ابتدا، تغییرات x و t را یکسان بگیریم و در نتیجه $x'(t)$ را برابر ۱ فرض کنیم. سپس تقریبی از منحنی به‌دست آوریم و سعی کنیم $x'(t)$ را از میزان اختلاف منحنی تقریب زده شده و منحنی اصلی، در نقاط مختلف، پیدا کنیم و تقریب خود را در هر مرحله بهتر کنیم.

الف – فواصل هندسی متناظر با فواصل مساوی t

ب – اثر پارامتر tension بر روی نقاط کنترلی

شکل ۷- منحنیهای بزیه

مختصات نقاط کنترلی u و v با فرمولهای زیر محاسبه می‌شوند

$$\upsilon = Z_1 - \frac{e^{-i\phi}(Z_1 - Z_0)f(\phi,\theta)}{\beta} \qquad (۳)$$

$$u = Z_0 - \frac{e^{-i\theta}(Z_1 - Z_0)f(\phi,\theta)}{\alpha} \qquad (٤)$$

که در آن تابع f (ϕ و θ) به صورت زیر تعریف می شود

$$f(\phi,\theta) = \frac{2+\sqrt{2}(\sin\theta - \frac{1}{16}\sin\phi)(\sin\phi - \frac{1}{16}\sin\theta)(\cos\theta - \cos\phi)}{3(1+\frac{1}{2}(\sqrt{5}-1)\cos\theta + \frac{1}{2}(3-\sqrt{5})\cos\phi)}$$

برای تکمیل کردن مبانی پایه ای الگوریتم تقریب زدن منحنی، به روشی برای مقایسهٔ دو منحنی بزیه نیاز است. فرض کنید Z(t) مجموعهٔ نقاط منحنی ایده آلی باشند که (در صورت وجود) قطعه مسیر C را تقریب می‌زند، و فرض کنید منحنی بزیهٔ B(t)، یک تقریب اولیه از C است. می‌خواهیم B و Z را با هم مقایسه کنیم . نقاط شروع و پایان C ،Z و B، و همچنین زاویه مماس بر هر سه در این نقاط، یکسان هستند. این شرط باعث می شود که نقاط کنترلی B و Z در یک راستا باشند و تفاوت دو منحنی، فقط در اندازهٔ بردارهایی باشد که نقاط کنترلی را به نقاط اصلی وصل می کنند نه در زاویهٔ آنها، اگر بردارهای V U را به صورت $U = Z_{co} - Z_0$ و $V = Z_1 - Z_{c1}$ تعریف کنیم، منحنیهای Z و B به شکل زیر تعریف می شوند

$$Z_t = (1-t)^3 Z_0 + 3(1-t)^2(Z_0 + U) + 3(1-t)t^2(Z_1 - V) + t^3 Z_1$$

$$B_t = (1-t)^3 Z_0 + 3(1-t)^2(Z_0 + aU) + 3(1-t)t^2(Z_1 - bV) + t^3 Z_1$$

که a و b نسبت اندازهٔ بردارهای وصل کنندهٔنقاط کنترلی در دو منحنی هستند. اگر این مقادیر را به دست آوریم، می توانیم از روی نقاط کنترلی B ، نقاط کنترلی Z را پیدا کنیم و در نتیجه منحنی C را تقریب بزنیم.

اگر معادله های بالا را از هم کم کنیم (برای t ثابت و در نتیجه نقاط متناظر از دو منحنی)، معادلهٔ برداری زیر حاصل می‌شود

$$B_t - Z_t = 3(1-t)t^2(a-1)U + 3(1-t)t^2(1-b)V$$

که از حل آن، مقادیر a وb به صورت زیر نتیجه می شوند

$$a - 1 = \frac{-B_{yt}V_x + B_{xt}V_y - V_y Z_{xt} + V_x Z_{yt}}{3(1-t)t^2(U_x V_y - U_y V_x)} \qquad (۵)$$

$$1 - b = \frac{-B_{yt}U_x + B_{xt}U_y - U_y Z_{xt} + U_x Z_{yt}}{3(1-t)t^2(U_y V_x - U_x V_y)} \qquad (٦)$$

بنابراین، اگر یک تقریب اولیه از منحنی بزیه، مثل B ، داشته باشیم، می توانیم تقریب ایده آل را به دست آوریم. اما از آنجایی که نمی توان نقطه های کاملا متناظر دو منحنی، که t برابر دارند را پیدا کرد، باید با تکرار روشهای تقریبی، به جواب نزدیک شد چون ما Z را نداریم. برای پیدا کردن نقطهٔ Z_t ، با توجه به فاصلهٔ هندسی B_t (نسبت فاصلهٔ هندسی آن از ابتدای منحنی بزیه)، سعی می‌کنیم C_t را به دست آوریم (با همان نسبت فاصلهٔ هندسی از ابتدا مسیر) و Z_t را برابر C_t بگیریم. براساس فرمو لهای (۱) تا (٦)، الگوریتمی مکاشفه‌ای[۱۰]

برای پیدا کردن مقادیر α و β بافرض داشتن زاویه های ϕ و θ ارائه می‌کنیم.

۱- منحنی ورودی را که می خواهیم آن را با منحنی بزیه تقریب بزنیم C می نامیم. نقاط Z0 و Z1 را مساوی نقاط ابتدا وانتهای C قرار بده. زاویهٔ خط مماس بر C را در نقاط ابتدا و انتها محاسبه کن وبا محاسبهٔاختلاف این زوایا و زاویه خطی که Z0 و Z1 را وصل می کند، مقادیر زاویه های ϕ و θ را، طبق معادله‌های(۱) و (۲) به دست آور. از آنجا که منحنی C ، یک منحنی صاف شده با فیلتر گاوسی است وخطای نویز و کوانتیزه بودن را ندارد، انتظار داریم مقادیر این زاویه‌ها تا حد مطلوبی دقیق باشند. هرچه این زوایا با دقت بیشتری تعیین شوند، الگوریتم سریعتر به جواب خواهد رسید.

۲- مقدار α وβ را برابر ۱ قرار بده.

۳- طبق معادله های (۳) و (٤)، مقادیر u و v را به دست آورد.

٤- با روش مقایسه ای برداری دو منحنی و اختلاف α وβ با مقادیر ایده آل را پیدا کن. این روش مقایسهٔ برداری، به صورت زیر انجام می شود :

٤-۱- منحنی بزیهٔ Z1 .. controls u and v.. ZO را B بنام.

٤-۲- L_C و L_B را برابر طول هندسی B و C (تعداد نقاط کوانتیزه شدهٔ آنها) قرار بده. مقادیر α_{ave} و b_{ave} را برابر صفر قرار بده.

٤-۳- مقدار t را برای B از صفر تا ۱ با قدمهای 1/N افزایش بده. (N یک عدد ثابت است که هر چه بزرگتر باشد، در این مرحله از الگوریتم جواب را کندتر ولی دقیقتر به دست می آورد.)

٤-۳-۱- برای هر نقطهٔ B_t، مقدار g_t را برابر فاصلهٔ هندسی از نقطهٔ B_0 تا B_t قرار بده.

٤-۳-۲- نقطهٔ متناظر B_t ، از منحنی C را (که فاصلهٔ هندسی آن از ابتدای C، برابراست) C_t بنام.

٤-۳-۳- a_{ave} و b_{ave} را به اندازهٔ (1/N)a و(1/N)b افزایش بده,a. وb از معادله های (۵) و(٦) به دست می آیند.

٤-٤ با استفاده از مقا دیر a_{ave} و b_{ave}، مقادیرα وβ را به مقدار ایده آل آنها نزدیکتر کن.

۵- مراحل ۳ و٤ را آن قدر تکرار کن که با به تقریب خوبی از منحنی بررسی و یا تعداد دفعات تکرار مراحل، از حدی بیشتر شود. تقریب خوب منحنی وقتی به دست می آید که مجموع مربع فاصلهٔ نقاط متناظر هندسی C و B، از حدی کمتر شود. یعنی

$$\frac{1}{\max\{L_B, L_C\}} \sum_{j=1}^{\max\{L_B, L_C\}} (C_t - B_t)^2 \le \varepsilon$$

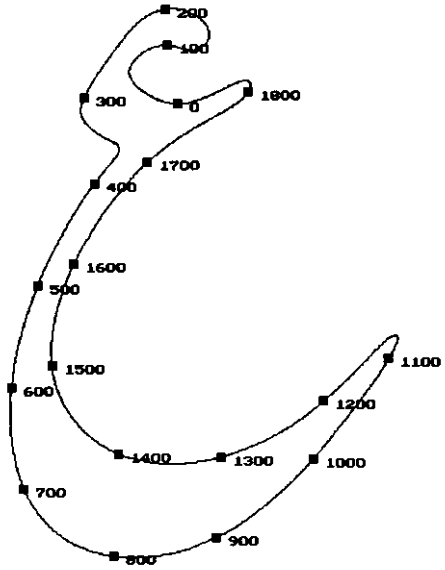
٦- اگرتقریب خوبی به دست آمد، متوقف شو. در غیر این صورت، زاویه های ϕ و θ را اندکی تغییر بده (در چهار نوبت، ترکیبات کم و زیاد کردن ϕ و θ را امتحان می کنیم) و الگوریتم را دو باره تکرار کن. اگر تقریب خوبی به دست نیامد، شکست الگوریتم را اعلام کن تا در انتخاب نقاط شروع و پایان، تجدید نظر شود.

۵- محاسبهٔ ویژگیهای منحنی وتعیین نقاط قلم‌درشت

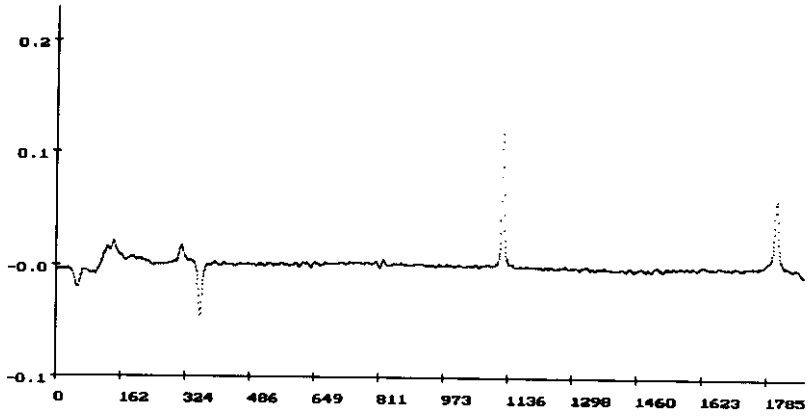
اگر بتوانیم ویژگیهایی از منحنی مرزی حرف، مانند انحنای منحنی، نقاط تیز، قطعه خطهای مستقیم را پیدا کنیم،می‌توانیم بعضی مکانهای قلم درشت را حدس بزنیم. روشی که در[٦] پیشنهاد شده است از این نظر بسیار مفید است. اگر x(t) و Y(t) همان توابع فیلتر شده با فیلتر گاوسی باشند، و تابع انحنای K را به شکل زیر تعریف می کنیم

$$K_\sigma(t) = \frac{\dot{X}\ddot{Y} - \dot{Y}\ddot{X}}{(\dot{X}^2 + \dot{Y}^2)^{3/2}}$$

که در آن، مقادیر $\dot{X}$، $\dot{Y}$، $\ddot{X}$ و $\ddot{Y}$ به صورت زیر تعریف می‌شوند:



شکل ۸- نمونه‌ای از منحنی مرزی فیلتر شده



شکل ۹- تابع انحنای مربوط به شکل (۸)

$$\dot{X} = \frac{dx_\sigma(t)}{dt} \quad \ddot{X} = \frac{d^2X_\sigma(t)}{dt^2}$$
$$\dot{Y} = \frac{dY_\sigma(t)}{dt} \quad \ddot{Y} = \frac{d^2Y_\sigma(t)}{dt^2}$$

و مقادیر مشتقها را می توان به شکل زیر تقریب زد :

$$\dot{X} = X_\sigma(t+1) - X_\sigma(t-1)$$
$$\ddot{X} = X_\sigma(t+1) - X_\sigma(t-1) - 2X_\sigma(t)$$

به این ترتیب، میزان انحنای منحنی در نقاط مختلف آن به دست می آید. در شکل (۸) می توانید نمونه ای از منحنی مـرزی و در شکل (۹) تابع انحنای آن را ببینید.

در تابع انحنا، نقاطی که مشتق صفر دارنـد و نقطـهٔ بیشـینهٔ موضعی هستند را به عنـوان گوشـه هـای تـیز منحنـی در نظـر می گیریم. نقاط متناظر با دو سر قلم درشت، انحنا هایی تقریبـاً مساوی ولی در جهت عکس دارند.

سازگاری یک مکان قلم درشت را تعریف می کنیم. قلمـی سازگار است که :

- انحنای دو سر آن، تقریباً مساوی و در جهت عکس باشند.
- زوایای دو سر آن، تقریباً مساوی باشند.

- مقطع قلم درشت، منحنی مرزی حرف را در نقطه ای غـیر از دو سر قلم، قطع نکند.
- فاصلهٔ هندسی نقـاط دو سـر از گوشـه هـای تـیز، تقریبـاً یکسان باشد.

با توجه به این تعریف، الگوریتمی بسیار ساده بـرای تعییـن مکانهای قلم درشت پیشنهاد می کنیم:

۱- اگر گوشهٔ تیزی وجود دارد، از آن شروع کن. در غـیر ایـن صورت، از هر نقطهٔ دلخواه دیگری شروع کن.

۲- در دو طرف منحنی × حرکت کن و نقـاط متنـاظر را پیـدا کن. نقاط متناظر آنهایی هستند که اگر دو سر یک مکان قلـم درشت در نظر گرفته شوند، آن قلم سازگار باشد. آن قدر از دو طرف جلو برو که نقاط متناظر پیدا نشوند.

۳- سعی کن دو طرف منحنی بررسی شده را با دو منحنی بزیه تقریب بزن. اگر الگوریتم تقریب، شکست خورد، منحنیهای دو طرف را نصف کن. (و در نقطهٔ تقسیم، یـک قلـم قـرار بده) این کار را آن قدر ادامه بده تا یا اندازهٔ منحنـی تقسـیم شده از حدی کوچکتر شود و یا تقریـب منحنـی بـه دسـت آید.

٤- عملیات بالا را برای تمام گوشه های تیز انجام مـی دهیـم. این عمل باعث می شود قطعه هایی از حرف، تولید شـوند. وظیفهٔ چسباندن این قطعات به هم، و اصلاح قلمهای اشتباه، بـه عهدهٔ کاربر است. البته ماکروهایی در متافونت بـرای کمـک بـه کاربر نوشته شده است.

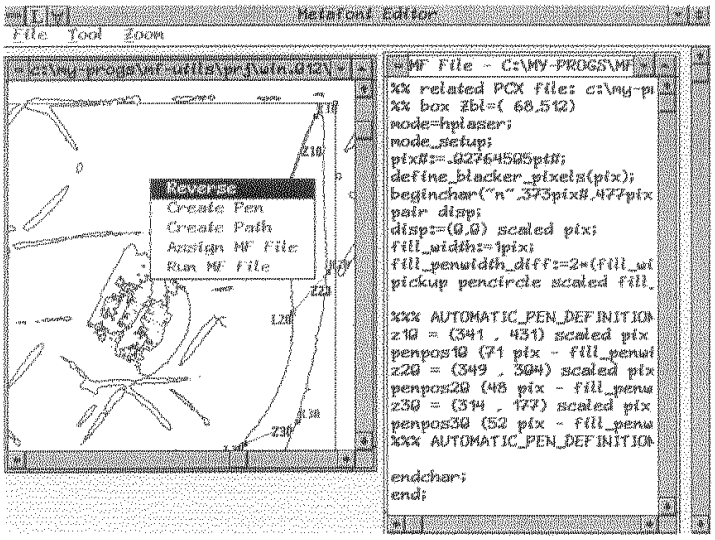
٦- سیستم پیاده سازی شده (محیط گرافیکی)

سیستم پیاده سازی شده، برنامه‌ای تحت ویندوز است که با مـترجم Borland C++ Ver 3.1 ترجمـه شـده و از کتابخانـهٔ Borland OWL استفاده می کند.

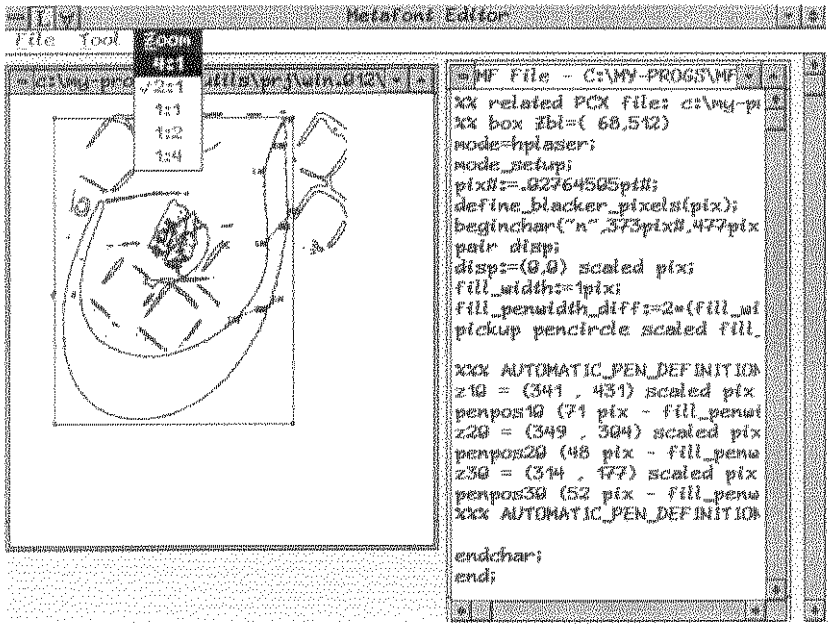
پنجرهٔ اصلی برنامه، یک MDI Frame اسـت و منوهـای File، Tool و Zoom را دارد. در این برنامه می توان بـه طـور همزمان روی یک یا چند تصویر PCX که تصویرهـای پویـش شدهٔ حروف هستند، کار کرد.

هر پنجرهٔ تصویـر مـی توانـد حداکـثر یـک پنجـرهٔ برنامـه (حـاوی برنامـهٔ تولیـد شـدهٔ متـافونت) و حداکـثر یـک پنجـرهٔ ویژگیهای انحنـا (حـاوی مشـخصات انحنـای مسـیر پـردازش شونده) داشته باشد. شکل (۱۰) نمونـه ای از پنجـرهٔ برنامـه و تصویر را نشان می دهد.

در پنجرهٔ تصویـر، تصویـر PCX را ویرایـش و منحنیـهای مرزی ناحیه‌ای از آن را پیدا می‌کنیم. سپس این منحنیهای مرزی را فیلتر می‌کنیم و منحنی تقریب را می‌زنیم. محل وصل‌کننـدهٔ

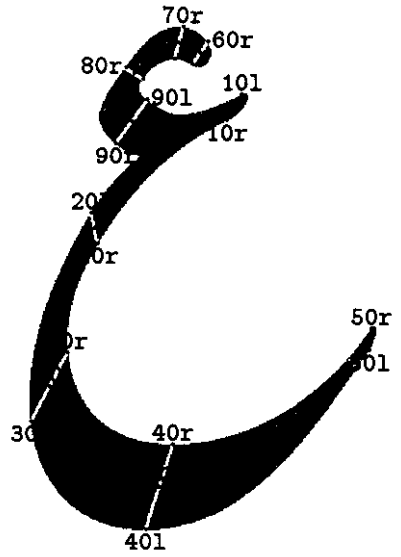


شکل ۱۰- نمونه‌ای از پنجرهٔ تصویر و پنجرهٔ برنامه



شکل ۱۱- نسبت کوچک شدن ۱ به ۲

شکل ۱۲- حرف (ن) تولید شده



شکل ۱۳- حرف (ع) تولید شده و مکانهای قلم درشت

حرف به صورت خودکار تولید می شود و کاربر نیز می‌تواند آن را ویرایش کند. در پنجرهٔ ویژگیهای انحنا، کاربر می تواند ویژگیهایی از منحنی پردازش شونده، مانند میزان انحنا و زاویه، و نقاط متناظر با گوشه های تیز تشخیص داده شده توسط برنامه را ببینید.

در منوی Zoom می توان بزرگنمایی تصویر را تعیین کرد. نمونه هایی از استفادهٔ این منو در شکل (۱۱) نشان داده شده است. آنها را با شکل (۱۰) مقایسه کنید.

در پنجرهٔ تصویر، می توان اشیایی، مثل مستطیل، قلم درشت را به وجود آورد. این اشیاء به صورت شفاف روی صفحه رسم می شوند. یعنی وقتی روی هم، یا نقاط زمینه قرار می گیرند، همهٔ اشیاء روی هم قرار گرفته را می توان دید. هر یک از این اشیاء را می توان در صفحه حرکت داد، تغییر شکل و اندازه داد و با اعمال مخصوص به آن اشیاء را روی آنها انجام داد. مثالی از عمل خاص یک شیٔ، فیلتر کردن مسیر بسته است که فقط برای این نوع شیٔ قابل انجام است. نمونه‌ای از منو در شکل (۱۰) دیده می‌شود.

شکلهای (۱۲) و (۱۳) نمونه های تولید شدهٔ ایـن سیسـتم هستند.

۷- نتیجه گیری و گسترشهای آینده

در این مقاله الگوریتمهای به کار رفته در یـک سیسـتم نـرم افزاری برای تولید خودکار فونتها به زبـان متـافونت ارائـه شـد. سیستم پیاده سازی شده مـی توانـد بـرای طراحـی سـریع و بـا کیفیـت عـالی حـروف اسـتفاده شـود. در ایـن سیسـتم تصویـر پویش‌شدهٔ یک یا چند حرف بـا اسـتفاده از ابزارهـای گرافیکـی فراهم آمده به صورت نیمه خودکار به برنامـهٔ متـافونت تبدیـل می شود. این کار، با حذف نویز از تصویر، تقریب منحنی مرزی قطعات منحنی بزیه و سـپس تعییـن مشـخصات قلـم متـافونت انجام می‌شود. هر چند که این کـار در حـال حـاضر بـا کیفیـت بسیار خوبی انجام می شود و سیستم موجـود عمـلاً مـی توانـد زمان طراحی یک مجموعهٔ فونت را از هفته ها به چنـد سـاعت کاهش دهد، ولی کار برای تکمیل مراحـل خودکارسـازی ادامـه دارد. در این راستا، ایده های مطرح شده در [۷] می تواند مؤثـر باشد . مسئلهٔ تشخیص خودکار محل قرار گرفتـن قلـم درشـتها می تواند بهبود یابد. برای ایـن منظـور، عملیـات نـازک سـازی بسیار مهم است.

ایده‌هـای مطـرح شـده در ایـن مقالـه در تشـخیص نـوری کلمات و حروف(OCR) نیز می تواند مورد استفاده قـرار گیـرد. پروژه هایی هم در این زمینه در دست اجراست.

واژه نامه

1. font
2. font style
3. scanner
4. bitmap
5. METAFONT
6. descriptive
7. Donald Knuth
8. implicit
9. interactive
10. heuristic

مراجع

1. Knuth, D. E., *The METAFONTbook*, Addison-Wesley, 1985.
2. Knuth, D. E., *METAFONT: The program*, Addison-Wesley, 1985.
3. Knuth, D. E., *Computers and Typesetting*, Addison-Wesley, 1986.
4. Oppenheim, A. V., and Willsky, A. S., *Signals and Systems*, Printice-Hall International, 1983.
5. Gonzalez, R. C., and Wintz, P., *Digital Image Processing*, 2nd ed., Addison-Wesley, 1987.
6. Foley, J. D., Van Dam, A., and Feiner, S. K., *Computer Graphics: Principles and Practice*, 2nd ed, Addison-Wesley, pp. 478-491, 1990.
7. Xin, K., Lim, B. and Hong G. S., "A Scale-Space Filtering Approach for Visual Feature Extraction," *Pattern Recongnition Journal,* Vol. 28, No. 8, pp. 1145-1158, Aug. 1995.