Evolução de Curvas em Visão Computacional

Ralph Costa Teixeira

Fevereiro de 2011

Abstract

We are given a plane curve Q parametrized by $Q(s,0)$. This curve evolves with time t according to one of the following velocities

$$\begin{aligned} \text{Reaction} &: \quad Q_t = N(s,t) \\ \text{Diffusion} &: \quad Q_t = K(s,t)\, N(s,t) \end{aligned}$$

where N is the unit normal to the curve and K is its curvature. We present many interesting properties of both evolutions, discuss efficient algorithms to compute them, and present some of its applications in Computer Graphics and Computer Vision. Finally, we present a third curve evolution which is affine-invariant, namely,

$$Q_t = \mathcal{N}(s)$$

where $\mathcal{N}$ is the *affine normal* to the original curve Q.

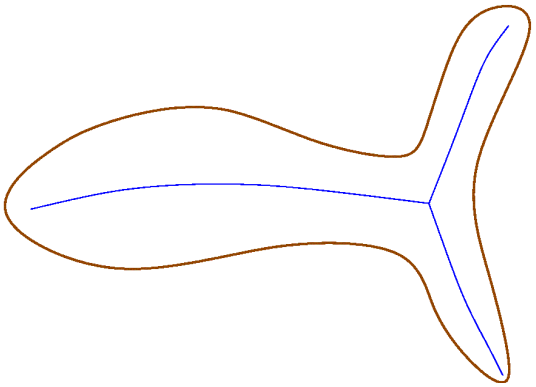
Capítulo 1

Introdução

Um dos objetivos principais da Visão Computacional é criar algoritmos que ensinem computadores a ver (isto é, não só capturar imagens mas também interpretá-las). Parte deste processo consiste em separar uma imagem em diversas regiões (cada uma correspondendo a algum objeto) para depois analisá-las. Não discutiremos aqui como encontrar tais regiões (que já é um problema bastante complicado por si só), mas estamos interessados em algumas técnicas matemáticas simples que nos permitam analisá-las.

Neste sentido, é útil definir algum tipo de **esqueleto** de uma região do plano – isto é, algum objeto unidimensional que funcionaria como uma espécie de eixo de simetria da região (mesmo que a região não seja exatamente simétrica). Também é útil classificar a região apresentada em pedaços distintos, para facilitar a sua análise.

Para descrever matematicamente objetos deste tipo, a linguagem natural é a da Geometria Diferencial. Assim, suponha que $Q(s) = (x(s), y(s))$ (com $s \in [0, B]$) é a parametrização de uma curva simples e fechada Q do plano, que limita a região R em questão (para facilitar, suporemos que esta parametrização percorre a curva no sentido anti-horário). Sendo esta parametrização regular (isto é, $Q_s \neq \vec{0}$ para todo s), denotaremos por $T(s)$ o vetor unitário tangente a esta curva e $N(s)$ o vetor unitário normal que aponta para "dentro"da região.



A seguir, vamos evoluir (ou deformar, propagar) esta curva paulatinamente (com o objetivo não só de encontrar o esqueleto ou subdividir a região original, mas com a esperança de que a matemática desta deformação seja bonita e interessante). Esta evolução pode ser representada por $Q(s,t) = (x(s,t), y(s,t))$ onde t é o tempo da evolução e s é o parâmetro que percorre cada uma das muitas curvas geradas (abusaremos a notação e denotaremos a curva inicial por $Q(s) = Q(s,0)$). Assim, $Q_s = (x_s, y_s)$ representa a direção tangente à curva em cada ponto, enquanto $Q_t = (x_t, y_t)$ representa a velocidade que cada ponto segue durante a propagação.

Ao longo do texto usaremos as seguintes notações:

- $\langle v, w \rangle$ é o produto interno canônico entre os vetores v e w.
- $[v, w]$ é o determinante da matriz cujas colunas são os vetores v e w. Em outras palavras, a área do triângulo cujos lados são v e w é

$$A = \frac{|[v,w]|}{2}$$

Capítulo 2

Evolução por Reação

2.1 Função Distância

A primeira evolução que gostaríamos de estudar é a chamada *evolução por reação*

$$Q_t(s,t) = N(s,t)$$

ou seja, em cada ponto a velocidade de propagação é a normal unitária à curva naquele ponto. Que propriedades tem esta evolução?

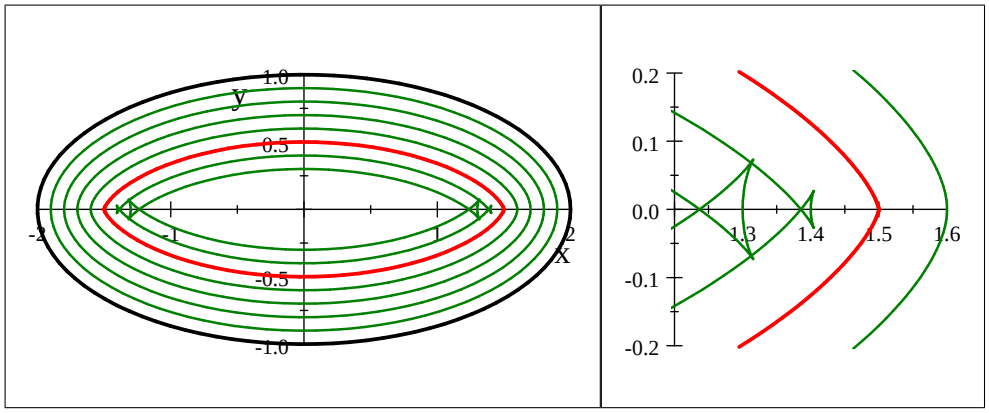
Resolver esta equação vetorialmente é simples: convidamos o leitor a notar que

$$Q(s,t) = Q(s) + tN(s)$$

é uma solução desta propagação. Afinal, é simples derivar esta equação e encontrar que $Q_t = N(s)$ – fica faltando apenas mostrar que $N(s) = N(s,t)$, isto é, que a normal unitária à curva Q original se mantém para s fixo à medida que t varia.

Exercício 1. *Determine a solução desta evolução quando a curva inicial Q é a elipse $Q(s) = (a\cos s, b\sin s)$ com $a \geq b$. Qual o menor valor de t que faz com que a parametrização em s deixe de ser regular? [Resposta:* $\left(a\cos s - \frac{b\cos s}{\sqrt{a^2\sin^2 s + b^2\cos^2 s}}t, b\sin s - \frac{a\sin s}{\sqrt{a^2\sin^2 s + b^2\cos^2 s}}t\right)$. *A parametrização deixa de ser regular quando $t = \frac{b^2}{a}$.]*

A figura abaixo à esquerda mostra a evolução da elipse $Q(s) = (2\cos s, \sin s)$ para $t = 0 : 0.1 : 0.7$. Note como a curva deixa de ser regular quando $t = 0.5$ (destacada), formando um bico sobre o eixo x. Isto ocorre pois em $t = 0.5$ no ponto $(1.5, 0)$ temos o primeiro choque da propagação, quando a frente de propagação do primeiro quadrante se choca com a frente de propagação do quarto quadrante. De fato, se permitíssemos $t > 0.5$, as frentes de propagação se cruzariam.



O leitor atento vai perceber que a distância do ponto $Q(s,t)$ ao ponto $Q(s,0)$ é exatamente t; mais ainda, como o segmento que une $Q(s,0)$ a $Q(s,t)$ é normal à curva original, então a distância de $Q(s,t)$ à **curva** original é t (pelo menos antes do primeiro choque). Em outras palavras, fixado um ponto (x_0, y_0) do plano, a evolução chega (pela primeira vez) ao ponto (x_0, y_0) no tempo t que é igual à distância de (x_0, y_0) à curva original.

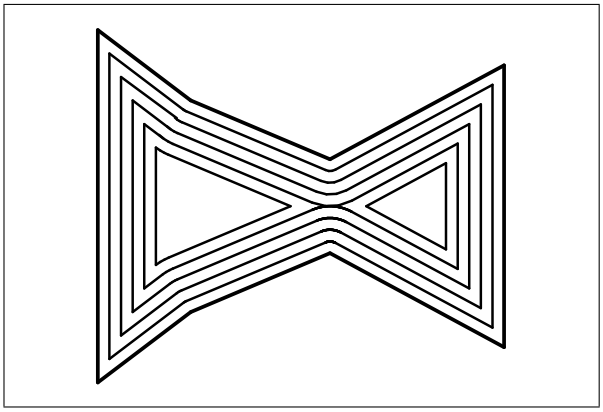
Em suma: **as curvas obtidas por esta evolução são exatamente as curvas de nível da função distância**, pelo menos até os choques.

Então vamos *redefinir* nossa evolução de uma forma mais implícita, evitando os entrecruzamentos que aparecem na figura à direita acima: dada a curva original $Q = C_0$, definimos a curva $Q(t)$ como sendo o conjunto de todos os pontos da região R (interior a Q) cuja distância à Q seja t. Ou seja, sempre que houver um choque,

descartamos os entrecruzamentos, e continuamos com curvas simples e fechadas (ainda que tenham bicos).

Uma segunda vantagem deste nova definição é que ela não precisa da definição do vetor normal e, portanto, independe da curva original ser diferenciável. Assim, a figura inicial pode até mesmo ser um polígono, se desejarmos.

Outro detalhe desta evolução: se a curva inicial não for convexa, esta evolução poderá dividir a curva propagada em pedaços desconexos, como no exemplo abaixo.



Exercício 2. *Sejam $\vec{n}_1$ e $\vec{n}_2$ os vetores unitários normais a dois lados consecutivos de um polígono convexo que está se movendo de acordo com a evolução por reação. Mostre que o vértice entre estes lados se move com velocidade $\vec{v} = \frac{\vec{n}_1+\vec{n}_2}{1+\langle\vec{n}_1,\vec{n}_2\rangle} = \frac{\vec{n}_1+\vec{n}_2}{1+\sin\theta}$ onde θ é o ângulo interno do polígono naquele vértice.*

Enquanto não é realmente necessário definir esta evolução para valores negativos de t, isto pode ser feito sem grande dificuldade.

Definição 3. *Dada uma curva simples fechada Q limitando uma região R, definimos a função distância com sinal f como*

$$f(X) = \begin{cases} d(X,Q) \ \textit{se } X \in R \\ -d(X,Q)\,, \ \textit{caso contrário} \end{cases}$$

Esta função distância tem várias propriedades interessantes. Em particular, as figuras devem convencer o leitor de que o gradiente desta função distância f é exatamente o vetor normal **unitário** à curva (exceto nos pontos de choque onde ela nem é diferenciável). Em suma, temos uma outra caracterização da função distância via E.D.P.s: a função distância deve ser, em algum sentido, a solução do seguinte problema com a Equação Eikonal:

$$\begin{cases} |\nabla f(X)| = 1 \text{ exceto nos choques} \\ f(X) = 0 \text{ se } X \in Q \end{cases}$$

Por outro lado, fazer esta caracterização rigorosa é surpreendentemente sutil; em primeiro lugar, "exceto nos choques"é vago demais; em segundo lugar, há casos patológicos (que não nos interessam) que "as figuras"acima não representam bem. Os exercícios a seguir mostram que $|\nabla f| = 1$, exceto em alguns casos patológicos.

Exercício 4. *Mostre que a função distância f é uma contração fraca*[1]*, isto é, dados dois pontos A e B do plano, tem-se*

$$|f(A) - f(B)| \leq d(A, B)$$

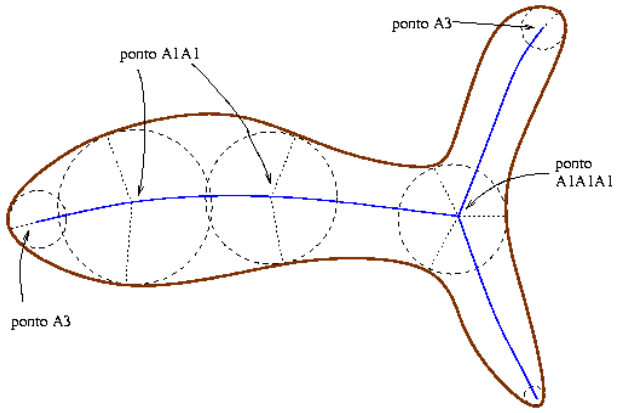
Exercício 5. *Mostre que a função distância $f(x, y)$ satisfaz a Equação Eikonal, a saber*

$$|\nabla f| = 1$$

em todos os pontos do ***interior*** *de R onde ela é diferenciável. [Dica: o problema anterior mostra que $|\nabla f| \leq 1$; agora, dado X no interior de R, encontre Y sobre Q tal que $f(X) = d(X, Y)$; usando pontos no interior do segmento XY se aproximando de X, mostre que $|\nabla f(X)| \geq 1$].*

Exercício 6. *Seja Q a curva definida $y = \sqrt{|x|}\cos\left(\frac{1}{x}\right)$ para $x \in (-1, 1)$ (em $x = 0$, tome $y = 0$; longe da origem, feche a curva da maneira que você preferir). Mostre que, se $|X| < r$, então $|f(X)| \leq 4\pi r^2$. Conclua que $\nabla f(O) = \vec{0}$ (na origem).*

[1] E, por conseqüência, f é diferenciável q.t.p.



2.2 Eixo Medial

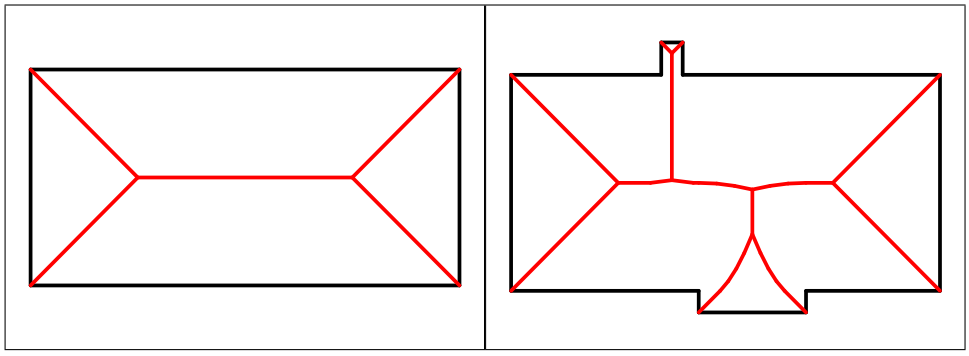
Estamos agora prontos para apresentar uma primeira definição de "esqueleto"de uma região delimitada por uma curva fechada.

Definição 7. *O* eixo medial *de uma região delimitada R por uma curva Q é o (fecho do) conjunto de pontos onde a função distância associada a Q não é diferenciável. Em outras palavras, o eixo medial é o conjunto dos pontos de choque da evolução por reação associada à região R.*

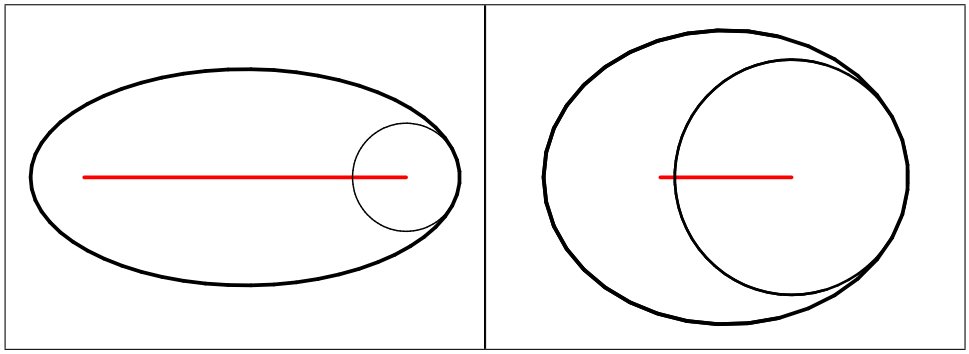
De uma maneira mais geométrica, considere todos os círculos contidos na região R que sejam maximais[2]. O conjunto dos centros destes círculos é o eixo medial.

Exemplo 8. *Se a região R é um retângulo, seu eixo medial lembra um telhado visto de cima; adicione pequenas indentações e o eixo medial cria ramos inteiros naquelas direções:*

[2] Essencialmente, estes são os círculos que são tangentes à curva Γ em dois ou mais pontos distintos; "essencialmente"porque, para completar o eixo medial, temos de incluir também centros de círculos que são tangentes a Γ em apenas um ponto, mas com ordem de tangência maior – são os pontos A_3 da figura.



Exemplo 9. *Se a região é a elipse* $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ *com* $a \geq b$*, o eixo medial é um segmento sobre eixo maior de comprimento* $2a - \frac{2b^2}{a}$.



Claramente, o eixo medial de um círculo é um ponto isolado, seu centro. O eixo medial tem várias propriedades interessantes, como por exemplo:

- Os extremos do eixo medial são centros de curvatura correspondentes a pontos de curvatura máxima (ou mínima) do bordo;
- O eixo medial é co-variante por isometrias[3];

[3] Em outras palavras, se R girar ou transladar, o eixo medial gira ou translada "junto".

- Quando bem definida, a direção tangente ao esqueleto num ponto é bissetriz das direções tangentes nos pontos correspondentes de Q;

Mais ainda, é possível reconstruir a região R se armazenarmos o esqueleto e o raio do círculo bitangente em cada um de seus pontos (de fato, a região R é simplesmente a união destes círculos). Esta é uma maneira interessante de representar a região R – o esqueleto captura *a direção geral* da região R, enquanto os raios dos círculos capturam a *grossura* de cada parte de R.

Isto dito, a representação de uma região por seu eixo medial e função raio tem problemas; o principal deles é a sensibilidade a ruídos (vide retângulo com indentações) – pequenos ruídos na região R ou no seu bordo Q levam a representações bastante distintas com eixos mediais. Por este motivo, aplicações práticas que se utilizam de esqueletos costumam passar por alguma espécie de suavização da curva Q ou poda do eixo medial, para eliminar seus trechos irrelevantes.

2.3 Computação e Aplicações

Para computar o eixo medial, há várias abordagens:

- Para polígonos, o eixo medial é uma espécie de diagrama de Voronoi (mas, ao invés de usarmos distâncias a *pontos* como é usual, usaríamos distâncias aos *lados*). Assim, é possível adaptar algoritmos que calculam diagramas de Voronoi para calcular eixos mediais;

- Há métodos que consistem em primeiro resolver o problema

$$\begin{aligned} |\nabla f| &= 1 \text{ em } R \\ f &= 0 \text{ em } Q = \partial R \end{aligned}$$

  para encontrar a função distância f – mas cuidado deve ser tomado pois singularidades de f são permitidas em R (que serão exatamente o eixo medial!).

- Para grids discretos, computar a função distância é um problema simples de Programação Dinâmica.

Um algoritmo que mistura as duas últimas técnicas e algumas aplicações serão apresentadas no capítulo sobre algoritmos mais à frente. Por agora, mencionamos os sites *The Hypermedia Image Processing Reference*[4] ou o exemplo de classificação de peças do *MMach/Khoros* no site da Unicamp[5] como exemplos de aplicações do eixo medial à Visão Computacional (ambos estão nos slides deste curso).

[4] Em http://homepages.inf.ed.ac.uk/rbf/HIPR2/; exemplos de esqueletos em http://homepages.inf.ed.ac.uk/rbf/HIPR2/skeleton.htm.

[5] Em http://www.dca.fee.unicamp.br/projects/khoros/; eixos mediais em http://www.dca.fee.unicamp.br/projects/khoros/mmach/tutor/application/industrial/pieces/pieces.html

Capítulo 3

Evolução por Difusão

Nesta seção, definiremos uma segunda evolução de curvas que tem suas próprias aplicações e propriedades interessantes. O leitor que precise relembrar as propriedades das curvaturas de curvas planas deve consultar o apêndice. Para referência, todas as nossas curvas são simples, fechadas, orientadas no sentido anti-horário, e, portanto, com normal unitária apontando para dentro da curva. Na nossa notação, $Q(s,t)$ é a família de curvas, cada uma parametrizada por s, onde t é o tempo da evolução; a métrica da curva é $g(s,t) = |Q_s(s,t)|$, e a curvatura é $K(s,t)$. Note que s não é necessariamente comprimento de arco – denotamos o parâmetro comprimento de arco por L, ou seja, $dL = g.ds$.

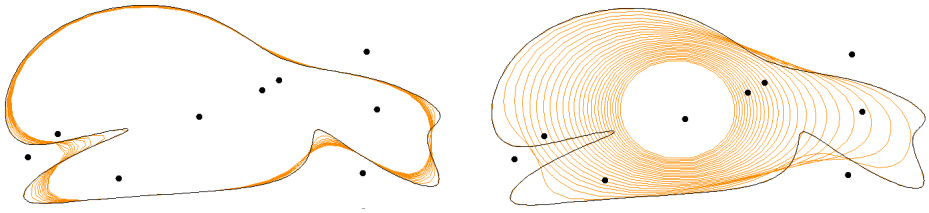
3.1 Movimento por Curvatura

Vamos agora estudar uma nova evolução, denominada *evolução por difusão* ou *movimento por curvatura*:

$$Q_t(s,t) = K(s,t)\,N(s,t)$$

ou seja, em cada ponto a velocidade de propagação é a curvatura vezes a normal unitária.

Em primeiro lugar, façamos algumas experiências; as figuras abaixo mostram a deformação de uma curva por esta lei (a primeira para valores pequenos de t, e a segunda para valores maiores).



Convidamos o leitor a ver e criar seus próprios exemplos; para tanto, visite os sites do Prof. J. Sethian (Berkeley)[1] ou interaja com o *Java 2D Closed Curve Simulator* do Prof. Shin Yoshizawa (Aizu)[2], que gerou as figuras acima. Note como os trechos da curva que são pontudos desaparecem rapidamente, e a curva acaba se transformando numa bolha redonda que, por sua vez, desaparecerá em tempo finito.

Exercício 10. *Se a curva inicial for um círculo de raio R_0, como a curva evolui por este movimento? [Resposta: será um círculo de raio $R(t) = \sqrt{R_0^2 - 2t}$, que, portanto, desaparece após $t = \frac{R_0^2}{2}$.]*

Que propriedades tem esta evolução? Comecemos então por fazer alguns cálculos interessantes:

Exercício 11. *Uma curva $Q(s,t) = (x(s,t), y(s,t))$ regular fechada simples é parametrizada por $s \in [0, B]$ e evolui com o tempo t de acordo com a lei $Q_t(s,t) = K(s,t) N(s,t)$.*
a) Use que $Q_{st} = Q_{ts}$ para mostrar que a métrica satisfaz

$$\begin{aligned} g_t &= -gK^2 \\ T_t &= \frac{K_s}{g} N \text{ e } N_t = -\frac{K_s}{g} T \end{aligned}$$

b) Sendo $L(t)$ o comprimento da curva e $A(t)$ a área por ela delimi-

[1] http://math.berkeley.edu/~sethian/

[2] O endereço da Universidade de Aizu parece estar desatualizado. Uma cópia deste Applet está em
https://php.radford.edu/~ejmt/Resources/CurveSimulator/CurveSimulator.htm.

tada, mostre que

$$\begin{aligned} L'(t) &= -\int_{s=0}^{s=B} K^2 dL \\ A'(t) &= -2\pi \end{aligned}$$

(onde $dL = g.ds$ é o elemento de comprimento de arco) pelo menos enquanto a curva for C^1.
c) Mostre que

$$K_t = K_{LL} + K^3$$

Uma consequência desta última equação é uma espécie de princípio do mínimo para K: se $K(s,t_0) > 0$ para todo s, então $K(s,t_1) > 0$ (onde $t_1 > t_0$) para todo s também. Em suma:

Teorema 12. *No movimento por curvatura, curvas convexas permanecem convexas.*

O arredondamento da curva também pode ser descrito de maneira mais precisa. Afinal, um interessante teorema da Geometria Diferencial [5] diz que:

Teorema 13 (Desigualdade Isoperimétrica)**.** *Seja Q uma curva plana, simples e fechada. Suponha que L é seu comprimento e A é a área que ela delimita. Então*

$$\frac{L^2}{A} \geq 4\pi$$

e a igualdade só se verifica se Q for um círculo.

Isto sugere o uso da razão $P = \frac{L^2}{A}$ para medir quão redonda uma região é. No entanto, a partir dos cálculos de L' e A' acima, é fácil ver que:

$$P'(t) = -2P\left(\int_0^B K^2 dL - \pi P\right)$$

É possível mostrar que o lado direito desta equação é sempre negativo [7]. Aliás, com estimativas mais cuidadosas, é possível mostrar que no movimento por curvatura tem-se

$$\lim_{t \to t_F} \frac{L^2}{A} = 4\pi$$

onde t_F é o tempo final da evolução. Ou seja, à medida que a curva encolhe, ela se aproxima de um círculo.

Mais ainda, o leitor pode estar preocupado com o fato de que tais cálculos acima só valem enquanto a curva for suave e simples. Uma série absolutamente notável de artigos por Gage e Hamilton ([7], [8], [6]) mostra os seguintes fatos:

Teorema 14. *No movimento por curvatura:*
a) Se a curva inicial for convexa, ela permanece convexa;
b) Se a curva inicial não for convexa, ela se torna convexa em tempo finito;
c) A curva é instantaneamente suavizada, e permanece suave até seu colapso;
d) Em suma, curvas imersas permanecem imersas e convergem para um ponto circular, sem nunca apresentarem qualquer espécie de auto-interseção!

Lembrando que $A'(t) = -2\pi$, isto é, $A(t) = A_0 - 2\pi t$, é então imediato notar que o colapso da curva se dá em $t = \frac{A_0}{2\pi}$.

3.2 Choques: que ponto?

Em suma, esta evolução não apresenta choques, exceto pelo ponto final de colapso. Até onde sabemos, a localização deste ponto de colapso é desconhecida, isto é, não sabemos em geral uma maneira simples de escrevê-lo em função da curva original (por exemplo, ele não é o centro de massa da curva nem da região).

3.3 Aplicação: Espaço de Reação-Difusão

Outra aplicação destas evoluções em análise de formas (Shape Analysis) é o espaço de Reação-Difusão apresentado em [11].

Antes de descrever matematicamente as idéias deste espaço, pensemos um pouco sobre como é difícil classificar formas visuais computacionalmente. Por exemplo, quase todos os humanos concordam que um corpo pode ser razoavelmente bem dividido em cabeça, tronco e membros. Mas, como criar um algoritmo que faça esta divisão automaticamente a partir de um esboço 2D de um corpo humano?

A idéia do espaço de Reação-Difusão é considerar uma curva $Q(s,t)$ evoluindo segundo a lei mais geral

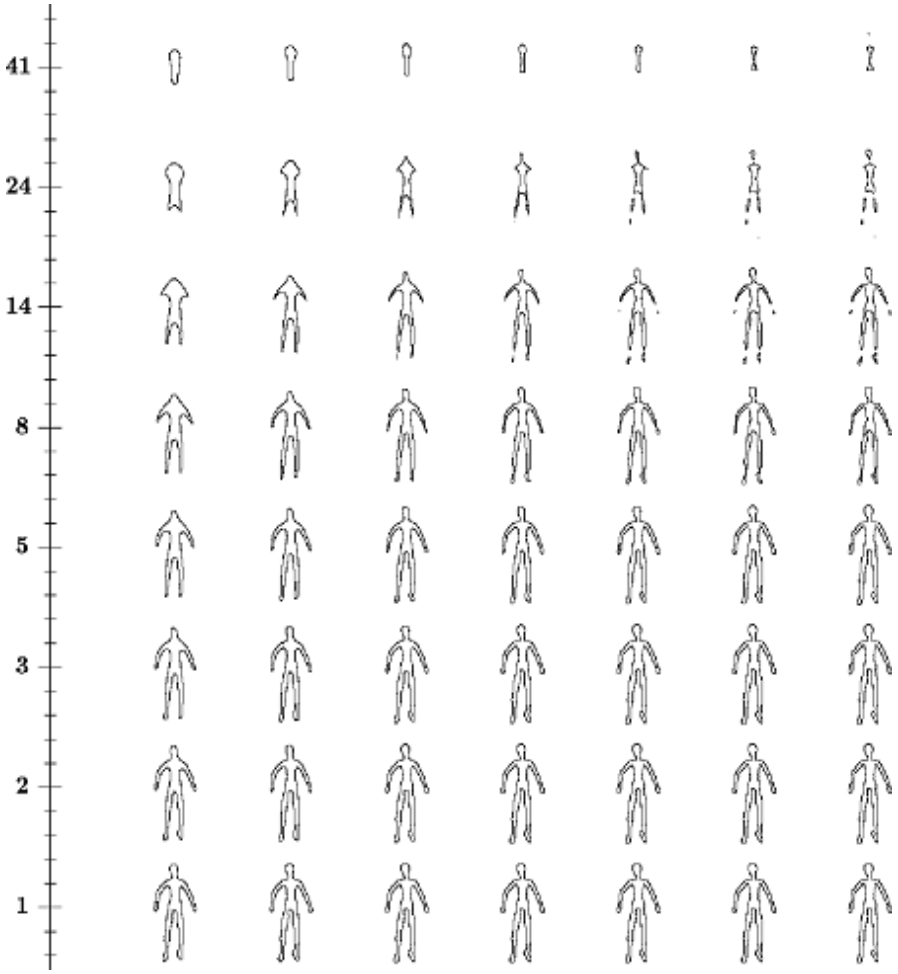
$$Q_t = (\alpha + \beta K) N$$

Se $\alpha = 0$, temos a evolução por difusão. Por mais que a curva original Q seja não convexa, nunca há uma quebra – a curva evolui suavemente, sem criar nem mesmo um bico, até o colapso final. Começanco por um esboço 2D de um corpo, os dedos simplesmente encolhem, assim como os outros membros do corpo e a cabeça, em direção ao torso central. Os pescoços inicialmente se alargam, até que o corpo se torna uma grande bolha convexa, que então encolhe até sumir.

Por outro lado, se $\beta = 0$, temos a evolução por reação; qualquer pescoço ou afinamento que a curva Q apresente resultará em uma quebra da evolução em duas curvas distintas. Agora, as mãos se separam dos braços quando os pulsos quebram, os pés se separam das pernas pelas canelas; em algum momento, estes membros desparecem completamente, e permanecem apenas uma versão reduzida de uma cabeça, provavelmente separada de uma versão reduzida do torso quando o pescoço quebra. Em seguida, a cabeça some e fica apenas o torso. O processo é sanguinolento, mas, não só esta evolução produz uma subdivisão da região inicial, mas ela produz também uma hierarquia dependendo do tempo que cada região sobrevive.

Agora, se desejarmos não somente detectar os afinamentos da forma visual, mas também classificá-las quando à sua "pescoçudez", podemos variar os valores de α e β. Para cada pescoço (pulso, canela ou afinamento), deve haver uma razão exata β/α que faz com que este pescoço não mais crie uma quebra na evolução da curva. Esta quantidade indicaria então o "grau de pescoçudez"deste pescoço da região R.

A figura a seguir, retirada de [12], exemplifica o processo acima. O eixo horizontal representa a quantidade β/α e o eixo vertical representa o tempo da evolução.


41
24
14
8
5
3
2
1

Capítulo 4

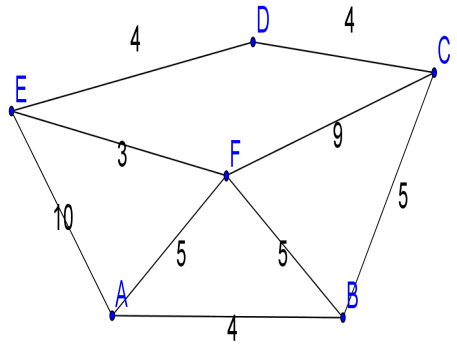
Algoritmos

Nesta seção discutiremos algoritmos eficientes que calculam as evoluções das seções anteriores, e apresentaremos mais algumas aplicações. Porém, antes de abordar nossas evoluções, precisamos de alguns fundamentos de Programação Dinâmica.

4.1 O Algoritmo de Dijkstra

Suponha que nos é dado um grafo conectado, cujas arestas são associadas a custos positivos (ou distâncias). Definimos o custo de um caminho como a soma dos custos de suas arestas. Escolhidos dois vértices deste grafo, como encontrar o caminho mais barato (ou mais curto) que os conecta?

Um algoritmo eficiente que resolve este problema é o *algoritmo de Dijkstra*. Vejamos como ele funciona através de um exemplo: no grafo a seguir com os custos indicados, qual o caminho mais barato de A até D?



O algoritmo de Dijkstra calcula, de fato, a função $C(x)$ que dá o menor custo de A até o vértice $x \in \{B, C, D, E, F\}$. Os passos são os seguintes:

i) Qual o vértice mais próximo de A?

Certamente, é um de seus vizinhos B, E ou F (chamados de *vértices candidatos*). Comparando os custos $4 < 5 < 10$, vemos que B é o vértice mais próximo. Então o custo de B é agora *conhecido*: $C(B) = 4$. pelo caminho AB. Mais ainda, já sabemos que $C(F) \leq 5$ (caminho AF) e que $C(E) \leq 10$ (caminho AE).

ii) Qual o segundo vértice mais próximo de A?

Quem são os candidatos? Certamente E e F ainda têm de ser considerados, com *possíveis* custos 10 e 5, respectivamente. Mas devemos também considerar os vizinhos de B como novos candidatos! Então temos também:

– o vértice F, agora pelo caminho ABF, com custo $C(B) + 5 = 9$. Como AF era melhor, este caminho é inútil.

– o vértice C, pelo caminho ABC que custa $C(B) + 5 = 9$. Então $C(C) \leq 9$.

Em suma, temos vértices candidatos C, E e F com custos candidatos 9, 5 e 10. Como destes o mais barato é 5, o vértice F passa a ser *conhecido*. De fato, $C(F) = 5$ pelo caminho AF. Mais ainda, sabemos que $C(E) \leq 10$ (caminho AE) e $C(C) \leq 9$ (caminho ABC).

iii) Qual o terceiro vértice mais próximo de A?

Candidatos antigos: AE (custo 10) e ABC (custo 9).

Candidatos novos: os novos vizinhos de F, a saber:

– o vértice E, agora pelo caminho AFE, com custo $C(F)+3=8$ (melhor do que o antigo AE!)

– o vértice C, agora via AFC, com custo $C(F)+9=14$ (não presta, ABC era melhor!).

Comparando AFE (8) com ABC (9), vemos que E é mais próximo. Ou seja, o próximo vértice *conhecido* é $C(E)=8$ por AFE, e resta a informação de que $C(C)\leq 9$ por ABC.

iv) Quarto?

Candidatos antigos: ABC (custo 9).

Candidatos novos: os vizinhos do último vértice calculado E... Neste caso, apenas D, isto é, $AFED$ com custo $C(E)+4=12$.

Então o novo vértice *conhecido* é C, com $C(C)=9$ via ABC, enquanto $C(D)\leq 12$ via $AFED$.

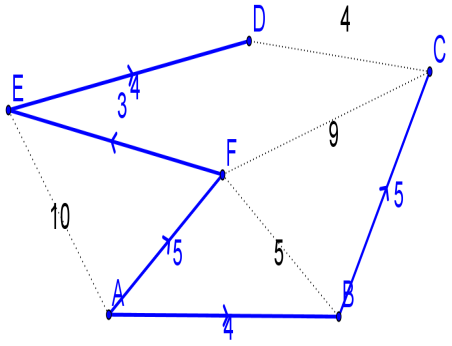
v) Quinto?

Candidato antigo: $AFED$ com custo 12.

Candidato novo: novos vizinhos de C, a saber, apenas D via $ABCD$, com custo $C(C)+4=13$.

Conclusão: $C(D)=12$ via $AFED$.

Note que não só determinamos o custo mínimo de A até qualquer outro vértice, mas também temos computado o caminho mínimo até lá. A figura a seguir ilustra toda a informação que obtivemos:



Algoritmo de Dijkstra

1. **Inicialização:** associar a cada vértice uma função *custo estimado*, fazendo $C(Início) = 0$ e $C(x) = \infty$ para todo $x \neq Início$. Marcar o vértice inicial como *conhecido* e os outros como *candidatos*. Enfim, definir o vértice *corrente* como o vértice A.

2. **Loop:**

– Para cada vizinho *candidato* Y do vértice *corrente* X, considere o novo custo estimado $C(X)+C(XY)$. Se este custo for menor que o $C(Y)$ previamente calculado, ele passa a ser o novo *custo estimado* $C(Y)$ (com novo melhor caminho $A...XY$).

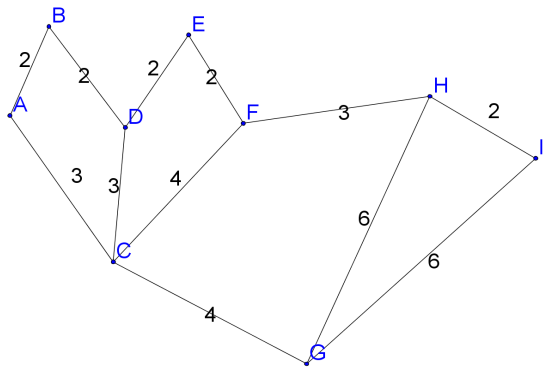
– Dentre todos os vértices *candidatos*, verifique qual tem o menor custo. Este vértice passa a ser *conhecido* (isto é, seu custo e caminho ótimos estão determinados e não mudam mais). Também, ele passa a ser o novo vértice *corrente* (seus vizinhos serão recalculados na próxima interação).

– Se ainda houver vértices *candidatos*, volte ao início do loop. Senão, o algoritmo termina.

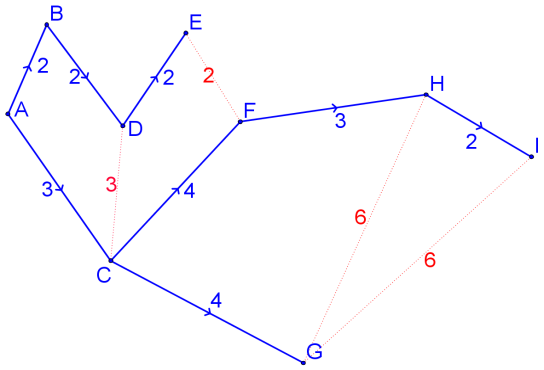
Suponha que o grafo tem V vértices e A arestas. Note que cada vértice é visitado apenas uma vez como "corrente", e em cada uma destas visitas a operação mais trabalhosa é encontrar um mínimo dentre V custos dos vértices candidatos[1]. Também, cada aresta é visitada uma única vez. Assim, o número de operações no algoritmo de Dijkstra é no máximo $O\left(E + V^2\right)$ – número que costuma ser muito menor do que o número de **caminhos** presentes no grafo entre dois de seus vértices.

Exercício 15. *No grafo a seguir, encontre o caminho mais curto de A até I.*

[1] Tipicamente, a lista de candidatos tem menos que V vértices. Aliás, se eles forem mantidos ordenados, não é necessário vasculhar todos eles – basta comparar o menor dos vértices da lista antiga com os novos vizinhos, e inserir os novos vizinhos ordenadamente na lista de candidatos. Assim, é possível reduzir o número de operações em cada vértice para $O(\log V)$ ao invés de $O(V)$.



Resposta:



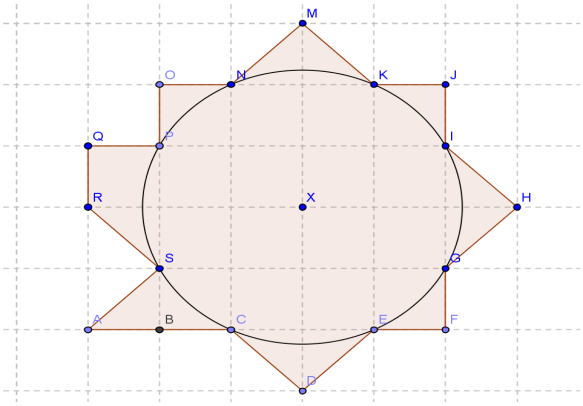
4.2 Fast Marching Methods

Como adaptar o cálculo de "custos"para determinar a função distância a uma curva do plano dada?

Uma primeira ideia é tomar a curva original como sendo um conjunto de "pixels"do plano, e usar um grafo que conecte os pixels para ser o grafo do algoritmo de Dijkstra. Ao invés de partir de um vértice inicial, teríamos um conjunto de vértices iniciais (a curva toda),

todos eles marcados com distância zero e como vértices "correntes". Agora é só usar algum grafo que conecte todos os pixels do plano e aplicar a ele o algoritmo.

Porém, que grafo utilizar? Se tomarmos um grafo que conecte apenas pontos adjacentes na vertical ou horizontal, várias distâncias serão superestimadas – na prática, estaríamos calculando uma distância via a métrica "quarteirão", mas queríamos utilizar a métrica Euclideana! Por exemplo, qual a distância do vértice X da figura abaixo ao polígono que o cerca, supondo que cada quadrado do grid é 1×1?



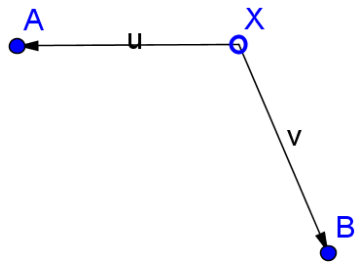
Aplicando o algoritmo de Dijkstra ao grafo tracejado, teremos $f(X) = 3$, pois esta é a distância mínima de X até a curva via caminhos paralelos aos eixos. **Note que esta estimativa não melhora diminuindo o lado do quadrado fundamental do grafo – se todas as arestas do grafo forem paralelas aos eixos, sempre teremos $f(X) = 3$!**

Se completarmos o grafo utilizando diagonais dos quadrados fundamentais, a distância ainda seria superestimada: teríamos $f(X) = \sqrt{2} + 1$, ainda independente do tamanho do grid.

A função que queremos de fato é $f(X) = \sqrt{5}$ (que é o raio do círculo indicado na figura). Uma maneira de encontrar este número seria conectar cada vértice do interior da região a cada pixel da curva original – mas, ao fazer isto, estamos de fato resolvendo o problema por força bruta, e o algoritmo de Dijkstra nem seria necessário. O

número de operações seria da ordem de V^2 onde V é o número de vértices **no interior da região** onde queremos calcular a função distância.

Ao invés de usar o algoritmo desta forma, considere o seguinte problema: na figura a seguir, suponha que conhecemos $f(A) = f_A$ e $f(B) = f_B$ e desejamos utilizar estes valores para estimar $f(X) = f_X$. Como fazê-lo?



A chave é lembrar que, para a função distância, temos $|\nabla f| = 1$. Assim, usando aproximações de primeira ordem, temos

$$\begin{aligned} f_A &= f(X+u) = f_X + \langle \nabla f(X), u \rangle \\ f_B &= f(X+v) = f_X + \langle \nabla f(X), v \rangle \\ |\nabla f(X)| &= 1 \end{aligned}$$

Exercício 16. *Suponha que* $u = (1,0)$ *e* $v = (0,1)$ *e elimine* $\nabla f(X)$ *no sistema acima, chegando a*

$$(f_X - f_A)^2 + (f_X - f_B)^2 = 1$$

que é uma quadrática em f_X*. Qual a condição para que esta quadrática tenha solução real? Interprete esta condição geometricamente.*

Em geral, nas três equações acima, conhecemos f_A, f_B, u e v, e as incógnitas são f_X e ambas as coordenadas de $\nabla f(X)$. Três equações, três incógnitas, há esperança de encontrar uma estimativa para f_X!

De fato, temos

$$f_X = \frac{f_A \langle v, v-u\rangle + f_B \langle u, u-v\rangle \pm [u,v]\sqrt{d^2 - (f_A - f_B)^2}}{d^2}$$

onde $d = d(A,B)$.

Observe que para encontrarmos uma solução real para f_X é necessário que $|f_A - f_B| \leq d(A,B)$ – mas já sabemos que a função distância f tem de satisfazer esta condição, pois é uma contração fraca!

Agora temos um plano: o cálculo da função f será feito começando pelos pontos mais próximos da curva original. Utilizaremos a fórmula acima para encontrar uma distância candidata f_X baseada em distâncias previamente determinadas f_A e f_B em pixels conhecidos A e B que são vizinhos a X. Dentre todos os candidatos X, aceitamos o "mais barato"como conhecido, recalculamos seus vizinhos usando a fórmula acima, e continuamos como no algoritmo de Dijkstra. Este algoritmo é chamado do tipo *Fast Marching*, já que o algoritmo "marcha"da curva para dentro, e essencialmente visita cada pixel a ser calculado apenas uma vez. Em suma:

Fast Marching Method para a função distância

1. **Inicialização:** Estabeleça valores iniciais nos pixels *conhecidos* (tipicamente, pixels em Q onde f vale 0), que inicialmente são também uma pilha de vértices *correntes*.
2. **Loop:**

Para cada pixel *corrente*, estime f em seus vizinhos desconhecidos e ponha estes pixels (e os valores de f) numa pilha de *candidatos*.

Tome o pixel X da pilha de candidatos com o menor $f(X)$ estimado e marque-o como *conhecido* e como o novo vértice *corrente*.

(Se em algum pixel X a nova estimativa for menor que uma estimativa prévia, fique com a nova!)

Se ainda houver vértices desconhecidos, volte ao início do loop.

Um último detalhe: a quadrática que calcula f_X tem duas raízes. Qual escolher? A resposta é clara se pensarmos na ordenação do método: escolha a raiz que for maior que ambos f_A e f_B.

Exercício 17. *Mostre que, se o triângulo XAB for acutângulo*[2] *em A e em B, então a média aritmética das duas raízes é uma média ponderada de f_A e f_B. Portanto, apenas a maior raiz pode satisfazer o critério $f_X \geq \max(f_A, f_B)$.*

Note também que a ordenação do nosso algoritmo garante que a propagação não passa dos choques, e portanto calcula a distância corretamente. Usando uma ordenação rápida de vértices candidatos, pode-se mostrar que este algoritmo é $O(N \log N)$ onde N é o número de pixels na região R.

4.2.1 Aplicação à robótica

Os métodos Fast Marching são tão rápidos que permitem a resolução de problemas aparentemente bem difíceis. Uma aplicação (de novo, mencionada pelo Prof. Sethian[3]) consiste em determinar como mover um sólido de um ponto a outro evitando obstáculos pré-determinados.

Expliquemos uma versão 2D do problema: você tem um "sofá poligonal"em um canto de uma sala retangular $S \subseteq \mathbb{R}^2$. Nesta sala há vários obstáculos, digamos, um conjunto de outros polígonos espalhados pelo retângulo original. Você quer mover seu sofá para outro ponto da sala, talvez até com uma nova orientação, driblando, é claro, os obstáculos todos. Será que é possível? Como?

A solução é surpreendentemente brutal: cada posição do sofá na sala pode ser representado por um ponto $(x, y, \theta) \in S \times [0, 2\pi]$, onde (x, y) representa a posição de um ponto específico do sofá (digamos, de seu centro de massa, ou de um vértice referência) e θ é um ângulo que representa a orientação do sofá como relação à original. O conjunto $S \times [0, 2\pi]$ é um *espaço de estados* para o sofá.

Agora, para cada ponto deste conjunto, verifique se a presença do sofá naquele estado é possível ou não – isto é, se o sofá naquele estado

[2] Na prática, sempre que X é pixel vizinho a A e a B, o triângulo XAB é acutângulo em A e em B. Em outras palavras, se XAB é obtusângulo em B, então A não é vizinho de X – deve haver um pixel mais próximo de X do que A.

[3] http://math.berkeley.edu/~sethian/2006/Applications/Robotics/robotics.html.

intersectaria algum obstáculo. Este é um problema de geometria – se o sofá e os obstáculos tiverem formas simples, verificar se eles intersectam é rápido. Se alguma posição for impossível, elimine-a do espaço de estados.

Ao terminar o processo acima, temos um espaço de estados com todas as posições possíveis do sofá. Simplesmente aplique o algoritmo de Dijkstra (ou o algoritmo Fast Marching correspondente) partindo do estado inicial, e encontraremos o menor custo saindo deste estado e chegando ao estado final. O caminho no espaço de estados indica o movimento que tem de ser aplicado ao sofá em questão! O algoritmo calculará TODAS as posíveis posições do sofá a partir da original, e também o caminho "menos custoso"até cada ponto da sala!

Melhor do que ler esta explicação é ver um Applet que realiza este algoritmo: visite a página do Prof Sethian.

A versão 3D pode ser resolvida de maneira semelhante, mas agora o espaço de estados terá 5 parâmetros – (x, y, z) para posição, (θ, ϕ) para orientação.

4.3 Level Set Methods

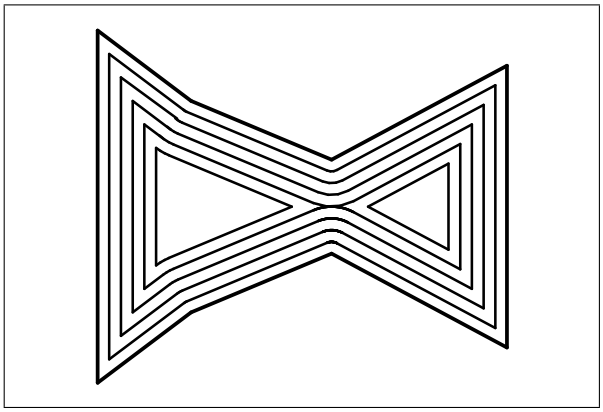
Agora voltamos nossa atenção a algoritmos eficientes que computem a Evolução por Difusão.

4.3.1 Método Lagrangiano

Uma primeira ideia para realizar o movimento por curvatura de uma curva plana seria amostrar a curva por um polígono (utilizando vértices sobre a curva chamados de "marcadores"). A partir destes, faríamos estimativas de vetor normal e da curvatura usando uma aproximação numérica em cada vértice. Tendo a velocidade estimada de cada ponto, podemos movê-los, obter uma nova curva, recalcular normais e curvaturas, etc.

Este é o chamado de *método Lagrangiano*, e pode ser aplicado a princípio para não somente a evolução por difusão, mas a qualquer evolução cuja velocidade dependa da normal e da curvatura. Enquanto é perfeitamente possível implementá-lo numericamente, considere:

- A ordem dos marcadores tem que ser cuidadosamente mantida. Parece simples fazer isto por meio de uma lista ordenada, mas lembremos que evoluções em geral podem levar a "quebras"na curva, isto é, mudanças de sua topologia. No momento da quebra, decidir computacionalmente que marcadores ficam em que pedaço da curva pode ser bastante complicado.



Quebra na evolução.

- Os cálculos do vetor normal e da curvatura são instáveis numericamente quando os marcadores estão muito próximos.

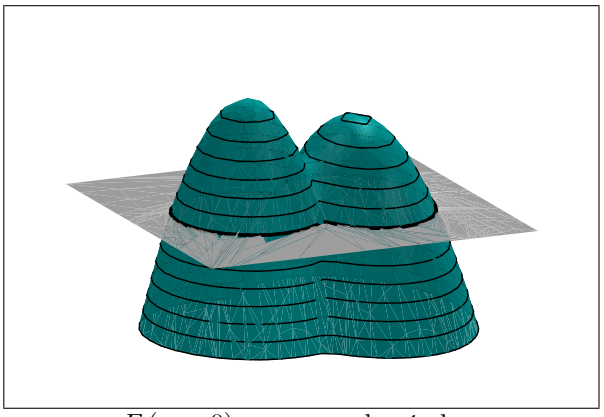
Enquanto o primeiro problema em teoria não afeta o movimento por curvatura (não há quebras), o segundo é praticamente inevitável: mesmo que a amostragem inicial da curva seja razoavelmente "bem espalhada", o movimento por curvatura vai trazer estes marcadores próximos um ao outro. É realmente questão de tempo até que instabilidades numéricas apareçam, a curvatura seja mal calculada, e o algoritmo vá por água abaixo[4].

4.3.2 Método Euleriano

Considere uma função $z = F(x, y)$ e uma de suas curvas de nível $Q : z = z_0$. Se a função F varia com o tempo (então de fato $F =$

[4]Uma outra possibilidade é "reamostrar" curva quando marcadores fiquem próximos uns dos outros, mas este processo traz uma "difusão adicional"ao problema que gostaríamos de evitar.

$F(x, y; t)$), a curva de nível também variará no plano $z = z_0$.



$z = F(x, y; 0)$ e a curva de nível $z = z_0$

Para ser exato, considere a curva de nível $F = z_0$ parametrizada por $(x(s,t), y(s,t))$ onde s é o parâmetro sobre a curva e t é o tempo. Assim, $F(x(s,t), y(s,t); t) = z_0$ para todo s e t. Derivando com relação ao tempo:

$$F_x x_t + F_y y_t + F_t = 0 \Rightarrow F_t = -\langle \nabla F, \vec{v} \rangle$$

onde $\vec{v}$ é a velocidade de um ponto (x, y) de Q (com s fixo) no plano $z = z_0$. Em particular, se $\vec{v} = vN$ onde N é a normal unitária a Q, ficamos com

$$F_t = -|\nabla F| v$$

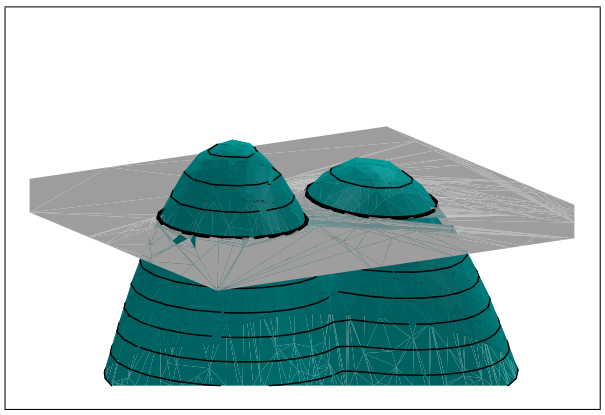
onde supusemos que ∇F aponta para **dentro** da curva, isto é, $\nabla F = |\nabla F| N$.

Esta é a ideia principal do *método Euleriano* ou *método por curvas de nível (Level Set Method)*; a invés de deformar diretamente uma curva no plano, representamo-la como uma curva de nível de uma função de duas variáveis $F(x, y; 0)$ – podemos usar, por exemplo, $F(x, y; 0) = f(x, y)$ onde f é a função distância com sinal discutida anteriormente! Agora aplicando a E.D.P. acima a F, fazemos a curva variar implicitamente. Se escolhermos v cuidadosamente, podemos fazer com que a curva evolua da maneira desejada.

Métodos deste tipo têm as seguintes vantagens:

- A curva em evolução pode passar entre os pixels onde a função F é calculada. De fato, se $F_A > z_0$ num pixel A e $F_B < z_0$ num pixel vizinho B, a curva passará entre A e B, e sua posição exata pode ser estimada com base nos valores F_A e F_B. Em suma, representar uma curva como curva de nível de uma função nos dá *precisão sub-pixel* na localização desta curva.

- O gráfico de F pode ter curvas de nível com topologias distintas. Assim, mudanças de topologia da curva em evolução não são problema algum, e não é necessário manter estruturas ordenadas para representar a curva.



F desce: nova topologia para Γ

No caso do movimento por curvatura, queremos fazer $v = K$. Como a curvatura das curvas de nível de $F(x, y)$ pode ser calculada por $K = \text{div}\left(\frac{\nabla F}{|\nabla F|}\right)$, então basta evoluir F segundo a lei

$$F_t = -|\nabla F| \,\text{div}\left(\frac{\nabla F}{|\nabla F|}\right)$$

onde o gradiente e divergente são tomados apenas nas coordenadas x e y. De fato, como z_0 é arbitrário, todas as curvas de nível de F farão seus movimentos por curvatura!

Implementar uma versão discreta desta E.D.P. não é muito difícil – algum cuidado tem que ser tomado em pontos onde $|\nabla F| \simeq 0$, onde a expressão à direita é indefinida; em tais pontos, pode-se mostrar ([4]) que a velocidade correta para que ocorra o movimento por curvatura desejado é:

$$F_t = \begin{cases} -\sqrt{\det H} \text{ se } \det H \geq 0 \\ 0 \text{ se } \det H < 0 \end{cases}$$

onde $\det H = F_{xx}F_{yy} - F_{xy}^2$ é o determinante da Hessiana de F. Em suma, pontos de sela não se movem verticalmente, enquanto máximos locais se movem com a velocidade $\sqrt{\det H}$ para baixo. Mais detalhes sobre este algoritmo estão em [3].

4.3.3 Aplicação: Image Denoising

Uma aplicação deste algoritmo é a eliminação de ruídos de imagens digitais. Uma imagem em tons de cinza pode ser modelada por uma função $z = F(x, y)$, onde (x, y) denota um ponto da tela e z é a sua cor (tipicamente, 0 =preto e 255 =branco). Neste contexto, ruídos tipicamente correspondem a valores de F que são muito diferentes de seus vizinhos, isto é, extremos locais de F cujas curvas de nível tem área muito pequena. Aplicando o algoritmo Level Set usando a imagem como estado inicial, tais curvas de nível de pequena área desaparecem rapidamente, sem deformar demais as de área maior.

Para exemplos desta aplicação, veja novamente o site do Prof. Sethian (Berkeley)[5].

[5] Novamente, em http://math.berkeley.edu/~sethian/.

Capítulo 5

Distâncias Afins

Neste capítulo, criamos novas funções distância diferentes da Euclideana com a propriedade de serem co-variantes por transformações afins.

5.1 Geometria Afim de Curvas

Uma das propriedades interessantes dos vetores tangente e normal à uma curva é sua covariância por isometrias. Em outras palavras, aplicando uma isometria à curva original, os vetores tangente e normal da nova curva são os transformados dos anteriores pela mesma isometria.

Agora, note que estes vetores não são invariantes por transformações lineares em geral. Afinal, o vetor normal a uma circunferência passa pelo seu centro. Aplicando uma transformação afim a esta circunferência, podemos transformá-la numa elipse – e, em geral, o vetor normal à elipse não passa pelo seu centro!

Será que existem outros vetores associados a uma curva que sejam invariantes por transformações afins (lineares + translações) quaisquer? Se limitarmos as transformações lineares àquelas que preservam área (determinante 1), a resposta é, surpreendentemente, sim.

Definição 18. *Dada uma curva parametrizada $Q(s)$, o* compri-

mento de arco afim *é o parâmetro*

$$u = \int [Q_s, Q_{ss}]^{1/3} \ ds$$

Note que u independe da parametrização $Q(s)$ utilizada. De fato, tomando uma nova parametrização $R(t) = Q(s(t))$, vem

$$\begin{aligned} R_t &= Q_s \cdot s' \\ R_{tt} &= Q_{ss} \cdot (s')^2 + Q_s \cdot s'' \end{aligned}$$

então

$$\begin{aligned} [R_t, R_{tt}] &= \left[Q_s s', Q_{ss}(s')^2 + Q_s s''\right] = [Q_s, Q_{ss}](s')^3 \Rightarrow \\ &\Rightarrow [R_t, R_{tt}]^{1/3} dt = [Q_s, Q_{ss}] ds \end{aligned}$$

Em particular, se $s = L$ for o comprimento de arco, então $Q_L = T$, $Q_{LL} = KN$ e portanto

$$u = \int K^{1/3} dL$$

Definição 19. *Dada uma curva parametrizada, sua tangente afim e sua normal afim num ponto da curva são respectivamente*

$$\begin{aligned} \mathcal{T} &= Q_u \\ \mathcal{N} &= Q_{uu} \end{aligned}$$

onde u é o parâmetro de arco afim.

Como sabemos que $\frac{du}{dL} = K^{1/3}$, não é difícil re-escrever a tangente e a normal afim em função da tangente e normal usuais. De fato

$$\begin{aligned} \mathcal{T} &= Q_u = Q_L \frac{dL}{du} = K^{-1/3} T \\ \mathcal{N} &= \frac{d\left(K^{-1/3}T\right)}{dL} \frac{dL}{du} = \left(-\frac{1}{3} K^{-4/3} K_L T + K^{-1/3} KN\right) K^{-1/3} \end{aligned}$$

ou seja

$$\begin{aligned} \mathcal{T} &= K^{-1/3} T \\ \mathcal{N} &= -\frac{1}{3} K^{-5/3} K_L T + K^{1/3} N \end{aligned}$$

Note que se $K \neq 0$, então $\mathcal{T}$ e $\mathcal{N}$ são transversais, mas não necessariamente perpendiculares! Quando $K = 0$, estes vetores não estão bem definidos (são "infinitos"na direção tangente). Por este motivo, *daqui para a frente suporemos que a curva $Q(s)$ tem curvatura positiva em todos os seus pontos.*

Exemplo 20. *A elipse $Q(s) = (a\cos s, b\sin s)$ não está parametrizada por comprimento de arco. O comprimento de arco afim é*

$$u = \int [Q_s, Q_{ss}]^{1/3}\, ds = \int \left| \begin{array}{cc} -a\sin s & -a\cos s \\ b\cos s & -b\sin s \end{array} \right|^{1/3} ds = (ab)^{1/3}\, s$$

Portanto, para parametrizar uma elipse pelo comprimento de arco afim, usamos a nova parametrização

$$Q(u) = \left(a\cos \frac{u}{\sqrt[3]{ab}}, b\sin \frac{u}{\sqrt[3]{ab}} \right)$$

A tangente e a normal afim são

$$\begin{aligned} \mathcal{T} &= Q_u = \left(-a^{2/3}b^{-1/3}\sin \frac{u}{\sqrt[3]{ab}}, a^{-1/3}b^{2/3}\cos \frac{u}{\sqrt[3]{ab}} \right) \\ \mathcal{N} &= Q_{uu} = \left(-a^{1/3}b^{-2/3}\cos \frac{u}{\sqrt[3]{ab}}, -a^{-2/3}b^{1/3}\sin \frac{u}{\sqrt[3]{ab}} \right) \end{aligned}$$

Observe que Q e $\mathcal{N}$ são paralelos. Em outras palavras, a normal afim a uma elipse aponta para seu centro!

A propriedade fundamental a seguir pode ser usada como uma definição da parametrização afim.

Proposição 21. *Se u é o comprimento de arco afim, então*

$$[Q_u, Q_{uu}] = [\mathcal{T}, \mathcal{N}] = 1$$

em todos os pontos da curva.

Demonstração. Segue imediatamente de

$$[\mathcal{T}, \mathcal{N}] = \left[K^{-1/3}T, K^{1/3}N \right] = 1$$

□

Definição 22. *Dada uma curva $Q(u)$ parametrizada por comprimento de arco afim, a* ***curvatura afim*** *$\mu(u)$ no ponto $Q(u)$ é definida por*

$$\mu = [Q_{uu}, Q_{uuu}] = [\mathcal{N}, \mathcal{N}_u]$$

Proposição 23. $\mathcal{N}_u = -\mu\mathcal{T}$.

Demonstração. Derivando $[Q_u, Q_{uu}] = 1$ com relação a u

$$[Q_{uu}, Q_{uu}] + [Q_u, Q_{uuu}] = 0 \Rightarrow [Q_u, Q_{uuu}] = [\mathcal{T}, \mathcal{N}_u] = 0$$

Portanto, $\mathcal{N}_u$ é um múltiplo de $\mathcal{T}$, digamos $\mathcal{N}_u = \alpha\mathcal{T}$. Mas então

$$\mu = [Q_{uu}, Q_{uuu}] = [\mathcal{N}, \mathcal{N}_u] = [\mathcal{N}, \alpha\mathcal{T}] = -\alpha\,[\mathcal{T}, \mathcal{N}] = -\alpha$$

□

Exemplo 24. *Voltemos à elipse parametrizada por comprimento de arco afim. Temos*

$$Q_{uuu} = \left(b^{-1} \sin \frac{u}{\sqrt[3]{ab}}, -a^{-1} \cos \frac{u}{\sqrt[3]{ab}} \right)$$

então sua curvatura afim será

$$\mu = \begin{vmatrix} -a^{1/3}b^{-2/3} \cos \frac{u}{\sqrt[3]{ab}} & b^{-1} \sin \frac{u}{\sqrt[3]{ab}} \\ -a^{-2/3}b^{1/3} \sin \frac{u}{\sqrt[3]{ab}} & -a^{-1} \cos \frac{u}{\sqrt[3]{ab}} \end{vmatrix}$$

Portanto, a curvatura afim de uma elipse é constante

$$\mu = (ab)^{-2/3}$$

No caso particular $a = b = R$, temos a curvatura afim de um círculo

$$\mu = R^{-4/3}$$

Exercício 25. *Mostre que a expressão da curvatura afim em função da curvatura usual K e de suas derivadas com respeito ao comprimento de arco usual é*

$$\mu = K^{4/3} - \frac{5}{9}K^{-8/3}K_L^2 + \frac{1}{3}K^{-5/3}K_{LL}$$

Exercício 26. *Mostre que a curvatura afim de uma parábola é* 0.

Proposição 27. *Seja A uma transformação equi-linear (isto é, linear que preserva áreas, ou seja, que tem determinante* 1*). Então as normais afins e tangentes afins em qualquer ponto da curva são co-variantes por A, e a curvatura afim é invariante por A.*

Demonstração. Como $[Q_s, Q_{ss}]$ é uma área, esta função escalar é invariante por A. Assim, o parâmetro comprimento de arco afim u é o mesmo na curva Q e na curva transformada AQ. Consequentemente, as novas tangente e normal afins são

$$\begin{aligned} (AQ)_u &= AQ_u = A\mathcal{T} \\ (AQ)_{uu} &= AQ_{uu} = A\mathcal{N} \end{aligned}$$

e portanto a nova curvatura afim é

$$[AQ_{uu}, AQ_{uuu}] = [Q_{uu}, Q_{uuu}] = \mu$$

já que, novamente, $[Q_{uu}, Q_{uuu}]$ é uma área que tem de ser preservada por A. □

5.2 Distância Afim

Com as ferramentas da geometria afim em mãos, é simples definir uma evolução que seja invariante por transformações equi-afins – seguiremos uma lógica semelhante à da Evolução por Reação.

Dada uma curva parametrizada $Q(s) = Q(s, 0)$, considere a evolução dada por

$$Q(s,t) = Q(s) + t\mathcal{N}(s)$$

Note que

$$Q_t = \mathcal{N}(s)$$

é a normal afim à curva original (e não à curva $Q(s,t)$!).

Exemplo 28. *Vimos que a elipse $Q(u) = \left(a\cos\frac{u}{\sqrt[3]{ab}}, b\sin\frac{u}{\sqrt[3]{ab}}\right)$ tem normal afim $\mathcal{N} = \frac{-Q}{(ab)^{2/3}}$. Assim, a evolução é*

$$Q(s,t) = \left(1 - \frac{t}{(ab)^{2/3}}\right) Q(s)$$

ou seja, para t fixo, a curva $Q(s,t)$ é uma elipse semelhante à original, de equação

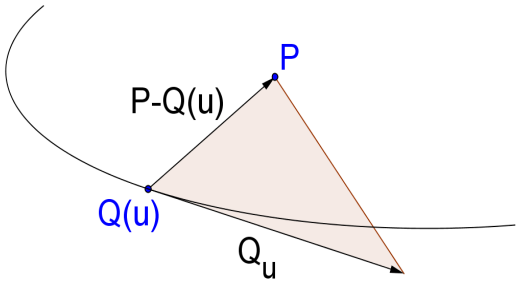
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = \left(1 - \frac{t}{(ab)^{2/3}}\right)^2$$

Esta evolução colapsa na origem quando $t = (ab)^{2/3}$.

Apesar da evolução da elipse não apresentar choques até seu fim, em geral esta evolução pode gerar entrecruzamentos. Portanto, novamente é interessante redefinir a evolução afim de uma forma mais implícita, usando curvas de nível de alguma espécie de função distância. Isto pode ser feito da forma a seguir.

Definição 29. *Dada uma curva simples fechada $Q(u)$ parametrizada por comprimento de área afim e um ponto P na região delimitada pela curva, definimos a distância afim de P a Q por*

$$f(P) = \min_u \left[Q_u(u), P - Q(u)\right]$$



Na definição acima, Q_u é uma normal afim e portanto co-variante por transformações afins. A expressão

$$h(u) = \left[P - Q(u), Q_u(u)\right]$$

é o dobro da área do triângulo com lados $P - Q(u)$ e $Q_u(u)$ (vide figura acima), portanto também é invariante. Enfim, como todos os objetos da definição acima são co-variantes por transformações

equi-afins, claramente a função $f(P)$ também é invariante por tais transformações.

Tentemos encontrar explicitamente o mínimo de $h(u)$. Derivando com relação a u

$$h'(u) = [P - Q(u), Q_{uu}(u)]$$

e note que esta derivada só se anula quando $P - Q(u)$ for paralelo a normal afim Q_{uu}, digamos,

$$\begin{aligned} P - Q(u) &= t\mathcal{N}(u) \\ P &= Q(u) + t\mathcal{N}(u) \end{aligned}$$

ou seja, $P = Q(s,t)$ na evolução afim.

Mais ainda, voltando à definição de h teríamos então neste ponto

$$f(P) = h(u) = [Q_u(u), P - Q(u)] = [\mathcal{T}, t\mathcal{N}] = t$$

Ou seja, fixada a curva Q, a função distância $f(P)$ é exatamente o tempo que a propagação afim demora para chegar ao ponto P.

Em suma, assim como a propagação com a velocidade do vetor normal foi redefinida via curvas de nível da função distância usual, vamos *redefinir a evolução afim usando as curvas de nível da distância afim* f. Com isto, evitamos quaisquer entrecruzamentos.

Vale a pena também notar que a expressão $f(Q + t\mathcal{N}) = t$ mostra que o gráfico de f é uma superfície regrada. Mais ainda, Moacyr Silva [17] mostrou que:

Teorema 30. *Com a notação acima, na curva original* $Q(s)$ *tem-se*

$$|\nabla f| = K^{-1/3}$$

Mais ainda, nos pontos da região em volta dos quais f *é* C^2 *vale*

$$D^2 f \cdot \mathcal{N} = 0$$

e, portanto,

$$\det\left(D^2 f\right) = 0$$

Demonstração. Podemos supor que $Q(u)$ está parametrizada por comprimento de arco afim. Derivando $f(Q + t\mathcal{N}) = t$ com relação a t, temos

$$\langle \nabla f(Q + t\mathcal{N}), \mathcal{N} \rangle = 1 \tag{5.1}$$

Mas nos pontos da curva original, tem-se

$$\begin{aligned} \nabla f &= |\nabla f| N \\ \mathcal{N} &= -\frac{1}{3} K^{-5/3} K_L T + K^{1/3} N \end{aligned}$$

donde vem a primeira igualdade.

Derivando 5.1 novamente com relação a t, vem:

$$\left\langle D^2 f \cdot \mathcal{N}, \mathcal{N} \right\rangle = 0$$

e derivando 5.1 com relação a u

$$\left\langle D^2 f \cdot (\mathcal{T} - t\mu\mathcal{T}), \mathcal{N} \right\rangle = 0 \Rightarrow \left\langle D^2 f \cdot \mathcal{N}, \mathcal{T} \right\rangle = 0$$

(onde usamos que $D^2 f$ é simétrica e que $t < 1/\mu$ antes dos choques). Assim, $D^2 f \cdot \mathcal{N}$ é ortogonal a ambos $\mathcal{T}$ e $\mathcal{N}$ (que são linearmente independentes) e tem que ser nulo. Enfim, como $D^2 f$ tem um autovalor 0, então

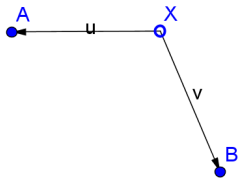
$$\det\left(D^2 f\right) = 0$$

□

O teorema acima nos diz que a função distância afim f é (em algum sentido) a solução do problema

$$\begin{aligned} \det\left(D^2 f\right) &= 0 \text{ em } R \\ |Df| &= K^{-1/3} \text{ em } \partial R = Q \\ f &= 0 \text{ em } \partial R = Q \end{aligned}$$

o que nos permite criar algoritmos do tipo Fast Marching para o cálculo numérico da distância afim! De fato, se soubermos os valores **do gradiente de** f em dois pontos A e B (digamos, V_A e V_B), podemos estimar o valor do gradiente de f num ponto vizinho X (digamos, V_X) usando as seguintes estimativas



$$\begin{aligned} V_A &= V_X + D^2 f(X) \cdot u \\ V_B &= V_X + D^2 f(X) \cdot v \\ \det\left(D^2 f\right) &= 0 \end{aligned}$$

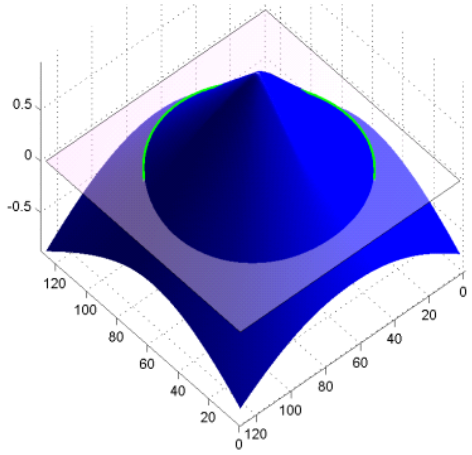
Temos aqui $2 + 2 + 1 = 5$ equações e 5 incógnitas (2 para V_X e 3 para $D^2 f(X)$, que deve ser simétrica). Com os gradientes todos calculados, é fácil estimar o valor de $f(X)$ baseado em $f(A)$ ou $f(B)$.

Enfim, citemos brevemente que os choques da evolução afim são um esqueleto co-variante por quaisquer transformações afins, denominado *Affine Distance Symmetry Set (ADSS)* (vide [9] e [10]).

Exemplo 31. *A distância afim f correspondente à epitrochóide*

$$(x(s), y(s)) = \left(\cos s + \frac{1}{50}\cos 4s, \sin s + \frac{1}{50}\sin 4s\right)$$

tem o gráfico abaixo

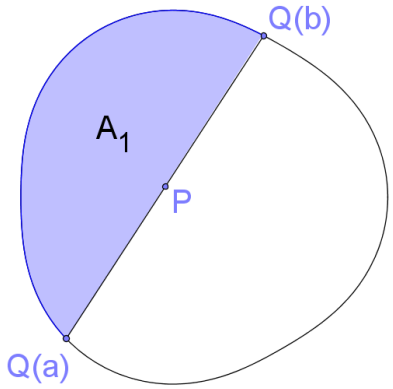


Note as "cristas"do gráfico, que correspondem a choques da evolução afim e, projetados no plano da curva, seriam o ADSS da epitrochóide.

5.3 Distância Afim por Áreas

A título de curiosidade, mencionamos brevemente aqui uma outra distância afim que não vem diretamente de uma evolução nem da geometria diferencial acima descrita. Esta distância aparece em [13] e [1].

Definição 32. *Dada a curva $Q(s)$ convexa limitando uma região R e um ponto P em R, considere uma corda $(Q(a), Q(b))$ passando por R. Esta corda divide R em duas regiões, com áreas $A_1 \leq A_2$. Escolha a corda $(Q(a), Q(b))$ (sempre passando por P) que determina a menor área A_1 possível. Esta área A_1 é a* distância afim por área *de P a Q.*



Como a definição é baseada apenas em cordas e áreas, ela é claramente invariante por transformações equi-afins. Apenas citaremos aqui algumas belíssimas propriedades desta nova função distância $f(P)$.

Proposição 33. *Se C é uma corda ótima associada ao ponto P (no sentido de determinar a menor área A_1 possivel), então P é médio de C. Alem disso, a corda ótima associada a P é tangente à curva de nível de f passando por P*

Demonstração. Veja [13] ou [14]. □

Teorema 34. *A menos de singularidades, a função f satisfaz*

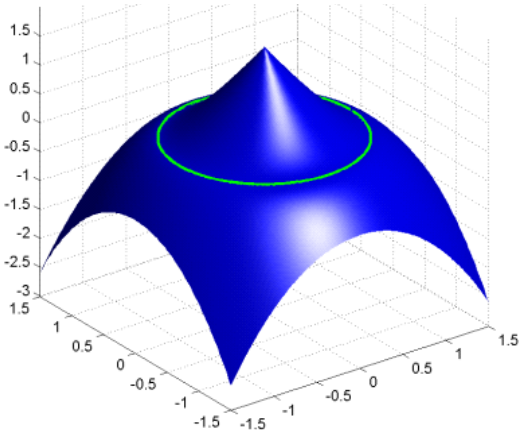
$$\begin{aligned} \det\left(D^2 f\right) &= -4 \text{ em } R \\ |\nabla f| &= 0 \text{ em } \partial R = Q \\ f &= 0 \text{ em } \partial R = Q \end{aligned}$$

Demonstração. Veja [17] ou [16]. □

Teorema 35. *O gráfico de f é uma esfera afim imprópria.*

Demonstração. Veja [2]. □

Enfim, a função f pode ser calculada com algoritmos do tipo Fast Marching (vide [15] para um algoritmo baseado na EDP, ou [2] para uma abordagem direta caso a curva Q seja um polígono).



Distância afim por área a um circulo (parte interna)

Apêndice A

Curvatura de Curvas Planas

A.1 Definição

Dada uma curva plana parametrizada por $Q(s) = (x(s), y(s))$, o seu vetor velocidade $Q_s = (x'(s), y'(s))$ forma um certo ângulo $\theta(s)$ com a horizontal (eixo $\overrightarrow{Ox}$). Definimos a curvatura K desta curva plana num determinado ponto como a taxa de variação deste ângulo **por unidade de comprimento** L medida na curva. Em outras palavras

$$K = \frac{d\theta}{dL}$$

onde

$$\begin{aligned} \tan\theta &= \frac{y'(s)}{x'(s)} \text{ ou } \theta = \arctan\left(\frac{y'}{x'}\right) \\ \frac{dL}{ds} &= |Q_s| = \sqrt{(x'(s))^2 + (y'(s))^2} \end{aligned}$$

Uma expressão de K em função da parametrização pode ser obtida

assim:

$$K = \frac{d\theta}{ds} \cdot \frac{ds}{dL} = \frac{d\left(\arctan\left(\frac{y'}{x'}\right)\right)}{ds} \frac{1}{\sqrt{(x')^2 + (y')^2}} =$$
$$= \frac{1}{1 + \left(\frac{y'}{x'}\right)^2} \frac{y''x' - x''y'}{(x')^2} \frac{1}{\sqrt{(x')^2 + (y')^2}} \Rightarrow$$
$$\Rightarrow K = \frac{x'y'' - y'x''}{\left((x')^2 + (y')^2\right)^{3/2}}$$

ou

$$K = \frac{[Q_s, Q_{ss}]}{|Q_s|^3}$$

Em particular, note que a definição de K depende apenas do formato da curva e não da velocidade em que a percorremos, isto é, dada uma curva, K **não depende da parametrização escolhida**[1].

Exemplo 36. *Uma possível parametrização de um círculo de raio R é*

$$Q(s) = (R\cos ws, R\sin ws)$$

Neste caso

$$Q_s = (-Rw\sin ws, Rw\cos ws) \Rightarrow |Q_s| = Rw$$
$$Q_{ss} = \left(-Rw^2\cos ws, -Rw^2\sin ws\right) \Rightarrow$$
$$\Rightarrow [Q_s, Q_{ss}] = \begin{vmatrix} -Rw\sin ws & -Rw^2\cos ws \\ Rw\cos ws & -Rw^2\sin ws \end{vmatrix} = R^2w^3$$

portanto

$$K = \frac{[Q_s, Q_{ss}]}{|Q_s|^3} = \frac{R^2w^3}{R^3w^3} = \frac{1}{R}$$

confirmando a idéia intuitiva de que K é maior quando a curva é mais fechada. Note que K não depende de w (que apenas muda a

[1] Para ser exato, a curvatura depende somente da direção em que a curva é percorrida; se usarmos uma parametrização que reverte essa direção, a expressão da curvatura muda de sinal. É comum se exigir que tal parametrização siga o sentido anti-horário para curvas fechadas.

parametrização, mas não a curva). Vale a pena notar que em geral K é o inverso do "raio de curvatura" de uma curva.

Com esta definição de curvatura, o seguinte teorema é bastante intuitivo (apesar de sua demonstração fornal em [5] ser complicada):

Teorema 37 (do Índice de Rotação)**.** *Para uma curva regular simples fechada $Q : [0, B] \to \mathbb{R}^2$ orientada no sentido anti-horário, tem-se*

$$\int_0^B K dL = 2\pi.$$

A.2 Parametrização por comprimento de arco

Se parametrizamos a curva por comprimento de arco (isto é, $|Q_s| = \sqrt{(x')^2 + (y')^2} = 1$ e $ds = dL$), então os vetores tangente e normal à curva[2] são $T = Q_s = (x', y')$ e $N = (-y', x')$. Assim:

$$\langle T, T\rangle = \langle Q_s, Q_s\rangle = 1 \overset{d/dL}{\Rightarrow} \langle T, T'\rangle = 0$$

e portanto vemos que T e T' são perpendiculares, isto é, $T' = \alpha N$ para alguma função escalar $\alpha(s)$. No entanto

$$K = \frac{[Q', Q'']}{|Q'|^3} = \frac{[T, T']}{1} = [T, \alpha N] = \alpha [T, N] = \alpha$$

já que T e N são unitários e ortogonais. Em suma

$$\frac{dT}{dL} = KN$$

e, analogamente

$$\frac{dN}{dL} = -KT$$

[2] Mais uma vez, supondo que N é a rotação de T de 90 graus no sentido anti-horário.

Estas duas equações são as equações de Frenet para curvas planas **parametrizadas por comprimento de arco**. É comum definir-se curvatura a partir das expressões acima, ou como

$$K = \left|\frac{dT}{dL}\right|$$

apesar desta última definição perder o sinal de K e ser inconveniente para nossos propósitos.

Exercício 38. *Suponha que $Q(s)$* ***não é parametrizada por comprimento de arco.*** *Neste caso, defina a* métrica *de uma curva parametrizada $Q(s)$ como $g(s) = \frac{dL}{ds} = |Q_s|$, isto é, $Q_s = gT$. Mostre que*

$$T_s = gKN(s)$$

e, consequentemente

$$N_s = -gKT(s)$$

Exercício 39. *Uma curva $Q(s,t) = (x(s,t), y(s,t))$ (onde $s \in [0,B]$) regular fechada simples evolui com o tempo t de acordo com a lei $Q_t(s,t) = N(s,t)$.*
a) Use que $Q_{st} = Q_{ts}$ para mostrar que

$$\begin{aligned} g_t &= -Kg \\ T_t &= N_t = 0 \end{aligned}$$

Em particular, isto mostra que a normal unitária é constante com relação a t.
b) Lembremos que o comprimento desta curva e a área por ela delimitada são, respectivamente,

$$\begin{aligned} L(t) &= \int_0^B g\ ds \\ A(t) &= \frac{1}{2}\int_0^B (xy_s - yx_s)\, ds = -\frac{1}{2}\int_0^B \langle Q, N\rangle\ g\ ds = \frac{1}{2}\int_0^B [Q,T]\ g\ ds \end{aligned}$$

Mostre que, enquanto a curva for de classe C^1 (isto é, antes dos choques), tem-se

$$\begin{aligned} L'(t) &= -2\pi \\ A'(t) &= -L \end{aligned}$$

c) Mostre que a função curvatura $K(s,t)$ satisfaz

$$K_t = K^2$$

e, portanto

$$K = \frac{K_0}{1 - tK_0}$$

onde K_0 é a curvatura no ponto correspondente da curva original. [Dica: derive $T_s = gKN$ com relação a t.]

A.3 Curvatura de curvas de nível

Dada uma função $F(x,y)$ que assuma valores reais, como calcular a curvatura da curva de nível $F(x,y) = C$ num determinado ponto regular[3] de seu domínio? Notemos que (utilizando índices para derivadas parciais) $\nabla F = (F_x, F_y)$ está na direção normal à curva de nível, portanto o vetor $(-F_y, F_x)$ é tangente à mesma. Assim, podemos encontrar uma parametrização da curva de nível $Q(s) = (x(s), y(s))$ com tal velocidade, isto é

$$Q_s = (x'(s), y'(s)) = (-F_y(x(s), y(s)), F_x(x(s), y(s)))$$

Podemos então calcular Q_{ss} usando a regra da cadeia

$$\begin{aligned} x'' &= -F_{xy}x' - F_{yy}y' = F_{xy}F_y - F_{yy}F_x \\ y'' &= F_{xx}x' + F_{xy}y' = -F_{xx}F_y + F_{xy}F_x \end{aligned}$$

onde omitimos da notação o ponto $(x(s), y(s))$ onde todas as derivadas parciais tem de ser calculadas.

Estamos prontos para calcular a curvatura de uma curva de nível

[3] Isto é, um ponto não crítico ou um ponto P tal que $|\nabla F(P)| \neq 0$.

de F num ponto $P = (x(s), y(s))$. De fato,

$$K = \frac{[Q_s, Q_{ss}]}{|Q_s|^3} = \frac{\begin{vmatrix} -F_y & F_{xy}F_y - F_{yy}F_x \\ F_x & -F_{xx}F_y + F_{xy}F_x \end{vmatrix}}{\left(F_x^2 + F_y^2\right)^{3/2}} \Rightarrow$$
$$\Rightarrow K = \frac{F_{xx}F_y^2 - 2F_{xy}F_xF_y + F_{yy}F_x^2}{\left(F_x^2 + F_y^2\right)^{3/2}}$$

e temos uma expressão para K em função de derivadas (cartesianas) de F. Uma maneira mais simples de escrever esta expressão é

$$K = \operatorname{div}\left(\frac{\nabla F}{|\nabla F|}\right)$$

cuja verificação deixamos ao leitor.

Comentário 40. ***Sistema gradiente de coordenadas.*** *Note que uma rotação ou uma translação dos eixos cartesianos não muda o formato das curvas de nível de F e, portanto, não muda a expressão de K. Já que temos liberdade em escolher tais coordenadas, por que não escolher um sistema de coordenadas locais que simplifique as expressões encontradas? Para tanto, dada uma função $F(x, y)$ e um ponto regular $P(x_0, y_0)$, definiremos o* ***sistema de coordenadas gradiente*** *(denotado por u,v) colocando a origem no ponto P, o eixo $\overrightarrow{Ou}$ tangente à curva de nível de F passando por P e a direção $\overrightarrow{Ov}$ na direção do gradiente de F no ponto P.*
Segue imediatamente que, no ponto P, temos $F_u = 0$ e $F_v = |\nabla F|$. Como a expressão de K é a mesma neste sistema de coordenadas, podemos substituir $F_u = 0$ na expressão de K para obter simplesmente

$$K = \frac{F_{uu}}{F_v}$$

É importante notar que as coordenadas u e v dependem do ponto P escolhido; podemos imaginar que a expressão de K é válida para qualquer ponto do domínio de F se imaginarmos que as coordenadas u e v variam de acordo com o ponto escolhido. Note que $F_u = 0$ somente no ponto P; portanto, não podemos escrever que $F_{uu} = 0$. Em outras palavras, ou pense que $F_u = 0$ é válido somente em P para

u fixo (e fique à vontade para diferenciar qualquer coisa com relação a u), ou pense que $F_u = 0$ em todos os pontos P do domínio (mas então u varia com o ponto P escolhido, e não faz sentido diferenciar com respeito a u). O truque é manter ambas as interpretações em mente ao mesmo tempo...

Bibliografia

[1] S. Betelu, A. Tannenbaum, Guillermo Sapiro, and Peter J. Giblin. Noise resistant affine skeletons of planar curves. In *Lecture Notes in Computer Science 1842*, pages 742–752, Dublin, 2000.

[2] Marcos Craizer, Moacyr Alvim Horta da Silva, and Ralph Teixeira. Area distances of convex plane curves and improper affine spheres. *SIAM Journal on Imaging Sciences*, 1(3):209–227, 2008.

[3] Marcos Craizer, Sinésio Pesco, and Ralph Teixeira. A numerical scheme for the curvature equation near the singularities. *Journal of Mathematical Imaging and Vision*, 22:89–95, 2005.

[4] Marcos Craizer and Ralph Teixeira. Evolution of an extremum by curvature motion. *Journal of Mathematical Analysis and Applications*, 293:721–737, 2004.

[5] Manfredo Perdigão do Carmo. *Geometria Diferencial de Curvas e Superfícies*. SBM, Rio de Janeiro, 2005.

[6] M. Gage and R. S. Hamilton. The heat equation shrinking convex plane curves. *Journal of Differential Geometry*, (23):69–96, 1986.

[7] Michael E. Gage. An isoperimetric equation with application in curve shortening. *Duke Mathematical Journal*, 50(4):1225–1228, 1983.

[8] Michael E. Gage. Curve shortening makes convex curves circular. *Inventiones Mathematicae*, (76):357–364, 1984.

[9] Peter J. Giblin and Guillermo Sapiro. Affine invariant symmetry sets. *Foundations of Computational Mathematics*, pages 152–168, 1997.

[10] Peter J. Giblin and Guillermo Sapiro. Affine invariant distances, envelopes and symmetry sets. *Geom. Dedicata*, 71:237–261, 1998.

[11] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. On the evolution of curves via a function of curvature, i: the classical case. *Journal of Mathematical Analysis and Applications*, (163):438–458, 1992.

[12] Benjamin B. Kimia, Allen R. Tannenbaum, and Steven W. Zucker. Towards a computational theory of shape: an overview. Technical Report CIM-89-13, McGill University, Department of Eletrical Engineering, Montreal, Canada, June 1989.

[13] Lionel Moisan. Affine plane curve evolution: a fully consistent scheme. *IEEE Transactions on Image Processing*, 7:275–301, 1998.

[14] Marc Niethammer, Santiago Betelu, Guillermo Sapiro, Allen Tannenbaum, and Peter J. Giblin. Area-based medial axis of planar curves. *International Journal of Computer Vision*, 3(60):203–224, 2004.

[15] Moacyr Silva, Ralph Teixeira, Sinésio Pesco, and Marcos Craizer. A fast marching method for the area-based affine distance. *Journal of Mathematical Imaging and Vision*, 30:1–12, 2008.

[16] Moacyr Alvim Silva, Ralph Teixeira, and Luiz Velho. Affine skeletons and monge-ampère equations. *SIAM Journal on Imaging Sciences*, 2:987–1001, July 2009.

[17] Moacyr Alvim Horta Barbosa Da Silva. *Esqueletos Afins e Equações de Propagação*. PhD thesis, Instituto Nacional de Matemática Pura e Aplicada (IMPA), 2005.