Transformação de Curvas Poligonais: Um Algoritmo Linear a partir de uma Triangulação

ANTÔNIO ALBERTO FERNANDES DE OLIVEIRA[1]
ADAILTON JOSÉ ALVES DA CRUZ[1 2]

[1] Laboratório de Computação Gráfica / COPPE / UFRJ
C.P. 68511
21945-970 Rio de Janeiro RJ
oliveira@lcg.ufrj.br adailton@lcg.ufrj.br

[2] UFMS - Universidade Federal de Mato Grosso do Sul
Caixa Postal 322
Rua João Rosa Goes, 1761
79800 Dourados MS

Abstract. In this work we describe an algorithm for transforming in a continuous way, an open polygonal line (**A**) into another (**B**) not crossing **A** but having the same extreme points. The transformation obtained has properties sometimes required from a Morphing process: Any intermediate polygonal line generated neither has self-intersecctions nor crosses any other. The points in those lines are all in motion and all points in **B** are reached at the same time. The algorithm starting point is a triangulation $\Im$ of the polygon bordered by **A** and **B**. It visits all triangles of $\Im$ in a three stage process with the two main objectives below:

a) Obtain a certain matching of triangles and non-extreme polygonal vertices sets. From this matching trajectories from all points in **A** to all points in **B** can be derived.

b) Determine for every triangulation edge(**e**) having both vertices in **A** or in **B**, a piecewise linear function of its points. This function, called Access Time Function (ATF) of **e**, approximates the distance from **A** or **B** to a point in **e** along a trajectory and is obtained from the ATFs of two adjacent edges.

The final result of this process is a data structure which is only linear in the number of **A** and **B** vertices and also makes possible to get any intermediate polygonal line in linear time.

1 - Introdução

Neste trabalho estudamos especcificamente o problema de transformar de forma contínua no plano uma poligonal aberta (**A**), sem auto-interseções numa outra (**B**), que não cruza **A** mas que tem extremos comuns com ela. Ambas as poligonais são supostas fornecidas pelas sequências dos seus vértices. Para esse problema apresentamos uma metodologia que parte de uma triangulação do polígono delimitado por **A** e **B**. As poligonais intermediárias geradas pelo método satisfazem à propriedade de não apresentarem auto-interseções e duas poligonais obtidas para tempos distintos do processo de transformação também não se cruzam, o que é um pouco mais forte. Esses resultados são fáceis de obter a partir do simples fato das trajetórias de cada ponto de **A** até seu destino em **B** serem todas disjuntas. De fato o método estabelece uma bijeção entre o polígono delimitado pelas duas poligonais menos seus pontos extremos e o "quadrado" $\mathbf{Q_0}$**=(0,1) x [0,1].** Nessa bijeção segmentos verticais em $\mathbf{Q_0}$ são associados a trajetórias e os horizontais à instâncias intermediárias da transformação.

Sabendo resolver o problema discutido acima, pode-se obter uma transformção contínua que leve um polígono **P** noutro **Q** da seguinte maneira :

Supondo **P** e **Q** em posição relativa genérica, aplique a **P** numa primeira etapa, uma transformação geométrica(**M**) que preserve a sua forma e tal que o polígono resultante (**P'**) se aproxime razoavelmente de **Q**, como está mostrado na figura 1. A expectativa é que a diferença simétrica de **P'** e **Q** seja constituido por várias componentes , cada uma delas delimitada por um conjunto de segmentos consideravelmente menor que os que definem **P** e **Q**. O contorno de uma dessas componentes ($\mathbf{C_k}$) é constituido pela união de duas poligonais $\mathbf{P_k}$ e $\mathbf{Q_k}$ com extremos comuns, a primeira contida no contorno de **P'** e a outra no de **Q**. Suponha agora que temos para cada $\mathbf{C_k}$, uma transformação $\mathbf{T_k}$ que leve $\mathbf{P_k}$ em $\mathbf{Q_k}$. Aplicando $\mathbf{T_k \bullet M}$ à $\mathbf{M^{-1}(P_k)}$ $\forall \mathbf{k}$, simultaneamente efetuamos as operações de transformar de forma contínua, **P** em **P'** e **P'** em **Q** .

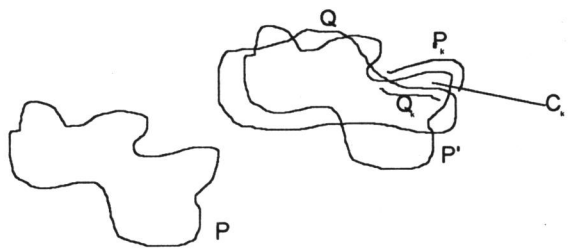

figura 1

Na estratégia descrita acima a obtenção de **P'** pode ficar por conta de um operador. Automatizá-la dependendo do critério que se adote para considerar **P** e **Q** semelhantes, pode entretanto, ser extremamente complicado. Uma tentativa de minorar esse problema é o método apresentado em [Oliveira(1993)] onde certos diagramas de Voronoi são empregados para estabecer correspondências entre partes dos contornos dados. Também nessa forma de abordagem, uma das etapas é resolver o problema um pouco mais restrito tratado neste trabalho.

Na transformação de uma imagem com contorno dado por uma poligonal simples , em outra , pode-se começar transformando de um contorno no outro da maneira descrita acima. Depois , extende-se o processo para transformar o interior de uma imagem no da outra. Nessa última etapa uma variação do método descrito nesse trabalho pode ser empregada para estabelecer correspondências entre pixels de ambas as imagens (ver [Oliveira(1993)]). Explora-se nesse caso a propriedade do método de construir bijeções entre polígonos e Q_0.Estabelecer essa correspondência entre pixels é uma das dificuldades para se adaptar métodos clássicos para o morphing de imagens (Ver [Wolberg(1990)]) à situações em que temos que transformar uma imagem em outra com o contorno muito diferente.

Um procedimento similar ao que será descrito aqui se aplica especialmente, a extender para toda uma região do plano, a estimação de uma dada grandeza, da qual se possuí um mapa de isopletas(curvas de nível) digitalizado. Para os níveis não apresentados no mapa se assume que as isopletas são as transformadas intermediárias do processo de levar uma isopleta do mapa noutra contigua a ela. Só que nesse caso, a região do mapa delimitada por duas isopletas contiguas pode ter diferentes topologias o que requer uma extensão do método que será apresentado aqui. Além disso há regiões delimitadas por uma única isopleta, as quais contem um extremo(máximo ou mínimo). Nesse caso é preciso fazer uma extrapolação a partir da isopleta para dentro.

Uma última possível aplicação diz respeito ao fato que, conhecendo-se uma forma de transformar continuamente um contorno em outro, pode-se extrair dela medidas da similaridade entre eles. Em função dessas medidas podem-se formular critérios para tomar decisões no trabalho de Reconhecimento de Formas (Ver [Kass(1988)] ou [Cohen(1991)]).

Na análise da complexidade do método que iremos fazer neste trabalho separamos dois tipos de operações. A primeira consiste na montagem de uma estrutura de dados conveniente (Pre-processamento). Na segunda, a partir dessa estrutura, "desenhamos" as instâncias intermediárias da transformação . Como é difícil pretender mais, considera-se aqui que a estrutura montada é "eficiente" se permitir que cada instância possa ser construida num tempo linear com a soma do número de vértices das duas poligonais: **O(r=n+m)** sendo n(m) o número de vértices de **A** (**B**, respectivamente).

O método que iremos descrever monta uma estrutura eficiente, segundo o critério definido acima , a qual pode ser construida em tempo também linear a partir de uma triângulação. Essa linearidade só pode ser conseguida permitindo-se que a velocidade da transformação varie, passando a depender do ponto atingido por ela. A velocidade será controlada por Gráficos de Tempo de Acesso(GTAs) obtidos para cada aresta da triangulação com vértices na mesma poligonal. A transformação gerada satisfaz ainda a duas propriedades que consideramos como "regras do jogo", sem as quais, transformações mais simples ainda podem ser obtidas (embora normalmente, com o sacrifício da qualidade visual). Essas propriedades são as seguintes:

1. Movimento Permanente: Exceto para os extremos comuns de A e B que são pontos fixos, todos os pontos de qualquer poligonal intermediária obtida pelo método, estão em movimento.

2.Chegada Simultânea: Excetuando seus extremos, todos os pontos de **B** são atingidos simultâneamente.

Algoritmos para efetuar a tranformação de um contorno 2D em outro já fazem parte de muitos pacotes gráficos tais como [Corel-Draw 2.0(1991)], [Designer3.1(1991)], [Harvard-Graphics3.0(1991)],etc. Diferentes metodologias tem sido aplicadas na sua construção. Traçando um ligeiro quadro comparativo entre o método que será proposto aqui e outras formas de abordagem devemos dizer o seguinte:

1) Dados dois polígonos (**P** e **Q**) suponha que a transformação de um no outro é dada por:

$$\mathbf{v(t) = (1-t)P \oplus tQ}$$

Para traçar uma instância intermediária dessa transformação é preciso determinar o contorno da Soma de Minkowsky de dois polígonos(ver [Rossignac(1991)], o que é certamente supra linear,

dado que o número de vértices desse contorno pode ser O(nm). Portanto, ele não seria eficiente segundo o critério definido acima, no caso não-convexo.

2)Algoritmos do tipo "Physically Based", que obtem a transformação de um contorno no outro baseados em algum critério de minimização de energia tem complexidade maior, como o apresentado em [Sederberg(1992)] que é O(nm).

3) Outros algoritmos são lineares mas não garantem a propriedade das transformadas intermediárias não terem auto-intersecções como os apresentados em algumas versões de software comercial (ver [Sederberg(1992)]).

Deve-se esclarecer que existe pouca bibliografia disponível para o problema de metamorfose de contornos. Entretanto, como resolver esse problema, significa essencialmente, se construir um interpolador, ele é afim com outros, conforme mostram os exemplos de aplicação dados acima . Para alguns desses outros problemas , sim, existe vasta literatura, disponível desde há muito tempo(Por exemplo,[Fuchs(1977)], [Keppel(1975)],etc) . Certos aspectos específicos do problema de metamorfose, no entanto, não são abordados nesses trabalhos.

Em linhas gerais o algoritmo que será apresentado aqui efetua a seguinte sequência de procedimentos:

1. Num processo de varredura da triangulação obtida inicialmente, vão sendo determinados a direção e o sentido das trajetórias em cada triângulo.sem construi-las inteiramente.

2. Simultaneamente vão sendo obtidos Gráficos da função Tempo de Acesso(GTAs) para cada nova aresta atingida com dois vértices na mesma poligonal . Arestas com extremos em **A** e **B** não precisam de GTAs. Para uma aresta com dois vértices em **A** , a função representada no seu GTA indica para cada um de seus pontos, uma aproximação do comprimento do trecho da trajetória , que vai de um ponto em **A** até ele. Idem para uma aresta com dois vértices em **B** em relação a **B**.

3. Ao final desse processo se avalia um valor conveniente(**M**) para o tempo de chegada a **B**(que tem de ser simultânea) . Determinado **M**, se divide por ele o valor das funções Tempo de Acesso, de forma que a transformação fique parametrizada dentro do intervalo [0,1]. Os GTAs das arestas com dois vértices em B são depois, substituidos por seu complemento a 1.

4. A partir desses GTAs normalizados e assumindo que nas arestas que ligam **A** a **B**, a função Tempo de Acesso é linear variando de 0 a 1 , as poligonais intermediárias da transformação podem ser construidas empregando-se um simples procedimento de interpolação.

Cada um desses procedimentos, bem como a própria construção da triangulação são abordados com detalhe em sub-seções específicas da seção seguinte, onde se descreve todo o algoritmo. Na última seção mencionamos deficiências e vantagens do método, e damos um exemplo de aplicação.

2. O Algoritmo.

2.1 - A triangulação.

Determinar uma triangulação $\Im$ dos vértices das duas poligonais dadas (**A** e **B**) restrita por elas próprias pode ser feito em tempo **O(rlogr)** (**r = m+n**) empregando, por exemplo o algoritmo dado por [Preparata(1985)], ou no caso de se requerer uma triangulação de Delaunay restrita, pelos descritos em [Chew(1989)] ou [Seidel(1989)] adaptados para eliminar a parte da triangulação exterior ao polígono **P** delimitado por A e B. Se esse polígono for regular sabemos que construir $\Im$ pode ser feito em tempo linear.

No restante desse trabalho consideraremos que $\Im$ está armazenado de forma que para quaisquer de seus elementos (vértices, aresta, triângulos), listar os que são adjacentes a ele, estão contidos nele ou o contém, pode ser feito em tempo constante por elemento listado.

2.2 - As trajetórias dentro de cada triângulo.

Uma vez obtida uma triangulação $\Im$ do polígono delimitado por **A** e **B** vamos inicialmente nos concentrar na questão de gerar trajetórias inteiramente disjuntas, saindo de pontos de **A** e chegando a **B**. Trajetórias disjuntas acarretam imediatamente que as instâncias intermediárias da transformação não podem ter auto-intersecções. Se além disso, a velocidade ao longo da trajetória for não nula e tiver sempre o mesmo sentido, a intersecção entre duas instâncias intermediárias correspondentes a tempos distintos, fica restrita aos extremos comums de **A** e **B**.

Para garantir que as trajetórias sejam disjuntas podemos fazê-las paralelas dentro de cada triângulo. A direção que elas tem dentro de um triângulo **T** pode então ser caracterizada por um vertice v_T(o vértice base de T) e por um ponto p_T , que é onde a trajetória que passa por v_T atinge a aresta oposta a ele: e_T dita aresta base de **T** .O sentido de percurso da trajetória será do vértice para a aresta ou vice-versa conforme este vértice pertença a **A** ou **B** respectivamente.

A escolha dos v_T , entretanto, não pode ser qualquer sob pena de não conseguirmos chegar a poligonal destino como mostra o exemplo da figura 2.

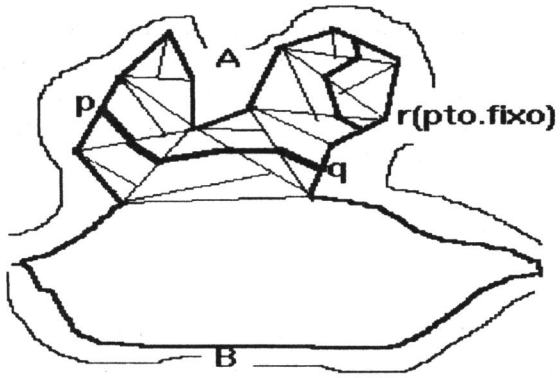

figura 2

Observe nessa figura que a existência de uma trajetória começando em **p** e terminando em **q** ambos em A , determina pelo próprio fato das trajetórias serem disjuntas, que haverá um **ponto fixo** (um ponto cuja trajetória é ele mesmo) no trecho de A compreendido entre **p** e **q**. Esse ponto fixo só pode ser um vértice pois todos os outros pontos do contorno se movimentam segundo a direção dada pelo par (v_T,p_T) do triângulo que os contém. Se um vértice for base de um triângulo ele não será ponto fixo. Portanto se conseguirmos que todos os vértices exceto os extremos comums de A e B sejam bases (Propiedade P1) , a problemática acima fica eliminada.

Também devemos evitar que uma aresta interior seja escolhida como aresta base dos dois triângulos adjacentes a ela, se seus vértices opostos nesses triângulos estão ambos em A ou ambos em B. Isso impedirá que trajetórias comecem ou se encerrem na aresta. Entretanto se as trajetórias de dois pontos da mesma poligonal se encontram, tomando a união das duas temos de novo uma situação como a da figura 2. Nesse caso , mais uma vez pela disjunção das trajetórias, chegamos a existência de um ponto fixo nessa poligonal, o que será evitado se **P1** for satisfeita.

Finalmente devemos impedir que o mesmo vértice seja escolhido como base para mais de um triângulo, a menos que esses triângulos sejam adjacentes e a direção das trajetória em ambos, seja a da aresta comum. Com isso evitaremos bifurcações e poderemos garantir a conexidade das transformadas intermediárias. Observe, entretanto que temos (**n + m - 4**) triângulos e igual número de vértices, excluindo os extremos comums de **A** e **B**. Assim se a atribuição dos vértices base deve ser sobre esse conjunto de vértices para que se satisfaça a Propriedade 1, ela deve igualmente ser biunívoca.

Para atenter portanto, os três requisitos acima basta satisfazer a Propriedade P1 que expressa simplesmente, um matching entre os triângulos de $\mathfrak{T}$ e os vértices não extremos de **A** e **B**. Com o intuito de obter esse matching,começamos dividindo o conjunto de arestas da triangulação em três grupos, conforme seus vértices pertencam ambos a **A** (arestas **AA**), ambos a **B**(**BB**) ou às duas poligonais (**AB**). Consideramos também o polígono **P** delimitado por **A** e **B** repartido em tres regiões: P_A, constituida pelos triangulos cujos tres vértices estão em **A** . P_B, definida analogamente em relação a B e P_{AB} formada pela união dos triângulos restantes.

No conjunto das arestas **AA** podemos definir uma ordem parcial ($\leq$) da seguinte maneira:

"**e** $\leq$ **e'** se **e** está contida na componente conexa de **P- e'** que não contém **B**".

As arestas de A são os elementos minimais de $\leq$ e para cada triângulo de P_A duas das arestas são imcomparáveis entre si e ambas $\leq$ a terceira . Escolhamos sempre esta última como base dos triângulos de P_A. Fazendo isso teremos satisfeito a propriedade **P1** para os vértices com todos os triângulos adjacentes em P_A(Ver [Cruz(1993)]). Além disso garantimos que as trajetórias que partem de pontos de A em P_A não voltam a **A** , mas vão atingir uma aresta maximal de $\leq$. Tudo que dissemos até agora com relação a P_A evidentemente se aplica a P_B, definido de forma análoga.

Considere agora, que os vértices de **A** são numerados de **1** a **n** e os de **B** de **n** a **n + m** , de forma que essa numeração indique a ordem em que os vértices vão sendo atingidos quando se percorre **A** $\cup$ **B** a partir de um extremo comum (v_1) . Nesse caso podemos reescrever a definição de $\leq$ tanto para as arestas **AA** como para as **BB** da seguinte forma :

$$e=(a,b) \leq e'=(c,d) \Leftrightarrow [a,b] \subseteq [c,d]$$

Isso simplifica a estrutura de dados do algoritmo estilo Graham-Scan que damos a seguir para indicar o par (e_T,v_T) dos triângulos de P_A e P_B. O algoritmo vai percorrendo o contorno de **P** até encontrar três vértices consecutivos nesse contorno (**i, i+1, i+2** $\leq$ **n**) que definam um triângulo (**T**) de $\mathfrak{T}$. Esse triângulo estará obviamente em P_A e (**i,i+2**) é $\leq$ que as suas duas outras arestas.

Fazemos então (**i,i+2**) ser e_T e retiramos do contorno as outras duas arestas de **T** e o vértice **i+1**. Nesse contorno atualizado com a retirada de **i+1** tanto (**i-1, i, i+2**) como (**i, i+2, i+3**) passaram a ser trincas de vértices consecutivos.

Para testar se a primeira delas define um triângulo, o algoritmo se obriga a um backtracking (volta a **i-1**), o que requer a seguinte simples providência. Para que esse backtracking possa ser feito em qualquer contorno obtido depois, para o qual a adjacência dos vértices já não é mais refletida pela consecutividade dos índices, é preciso que se tenha uma memória dos pontos do contorno-corrente já visitados pelo método. Essa memória pode evidentemente, ser organizada como pilha, pois dela precisamos apenas do último elemento incorporado.

Assim o algoritmo segue procurando triângulos de P_A cujos vértices sejam consecutivos no contorno corrente. Em encontrando um (**T**) ele é resolvido fazendo-se e_T ser a aresta diagonal e em seguida retirado, provocando a atualização do contorno. Quando ele tenta avançar além de n é porque já varreu todo P_A. A atribuição de (e_T,v_T) para os triângulos de P_B é feita de forma inteiramente análoga só que para vértices entre n e n+m.

Resta atribuir (e_T,v_T) aos triângulos de P_{AB} .Os triângulos dessa região podem ser enfileirados numa sequência(T_k,k=1,...,j) onde termos consecutivos são triângulos adjacentes, T_1 é o triângulo de P_{AB} que contem v_1 e T_j , o que contem v_n.

Podemos agora definir a função (**F**) que leva um vértice no bordo de P_{AB}, exceto os dois extremos das poligonais , no segundo triângulo de maior índice em (T_k) que o contém (ver figura 3). Se a um dos vértices do último triângulo da sequência, que não v_n, atribuimos exatamente esse último triângulo, teremos tornado a função biunívoca (Ver mais uma vez [Cruz(1993)]). Fazendo então, $v_T = F^{-1}(T)$, satisfazemos portanto, a propriedade P_1 também para os vértices dos triângulos de P_{AB}. Esclarecemos que há outras maneiras de escolher os (e_T,v_T) dos triângulos de P_{AB} enquanto que para os de P_A e P_B essa atribuição, para que a Propriedade P_1 seja atendida, tem de ser feita de forma única .

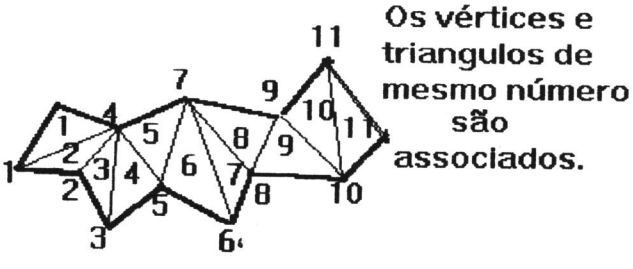

figura 3

2.3 - A construção dos GTA.

Devemos observar que no processo acima determinamos apenas localmente, em cada triângulo, como são as trajetórias. Para que as curvas intermediárias da transformação possam ser obtidas precisamos ainda indicar com que velocidade essas trajetórias são percorridas em cada ponto (ou em cada triângulo) e efetuar de alguma forma um processo de acumulação que nos permita calcular o tempo dispendido ao longo de uma trajetória para atingir qualquer de seus pontos.

Uma primeira idéia seria fazer a velocidade ser constante ao longo de uma trajetória. Determiná-la porém, nos obrigaria a conhecer o comprimento da trajetória já que estamos assumindo que o tempo de transformação é o mesmo para todos os pontos de **A** (ou **B**).O Comprimento de uma trajetória é uma função linear por partes de seu ponto de partida em **A**. Os pontos de quebra dessa função são exatamente os véritces de **A** e os pontos de partida das trajetórias que chegam a vértices de **B**. Desse modo precisamos conhecer as trajetórias dos vértices de **A** e **B** para poder determinar a velocidade com que uma trajetória qualquer deve ser percorrida. Determinar as trajetórias desses vértices, entretanto, é $O(r^2)$ dado que a trajetória de um deles pode atravessar $O(r)$ triângulos. Além disso, todas as intersecções de trajetórias de vértices de **A** e **B** com as arestas da triangulação devem ainda ser armazenadas numa estrutura conveniente. Uma vez conhecido comprimento da trajetória, a cada um desses pontos deve ser atribuida a fração

Comprimento da trajetória até ele / Comprimento total da trajetória.

para que as curvas intermediárias possam posteriormente, ser determinadas por interpolação. Assim a estrutura que temos de montar é $O(r^2)$.

Uma alternativa para manter linear o tamanho da estrutura necessária para fazer o traçado de uma instância da transformação também em $O(r)$, é operar com os Gráficos das Funções Tempo de Acesso (GTAs). Esses GTAs são definidos para as arestas **AA** e **BB** e sua descrição deve ter um tamanho máximo limitado a priori. Se eles forem, como assumiremos daqui por diante, poligonais, essa limitação se referirá ao seu número de vértices. No texto a seguir representaremos por π_e , a função Tempo de Acesso aos pontos da aresta **e**, ou seja, a função representada pelo GTA de **e**.

A construção dos GTA poderia ser feita tão logo se tivesse obtido a triangulação $\mathfrak{T}$. Entretanto, há condições que o conjunto desses gráficos deve satisfazer que dependem de elementos só determinados posteriormente. A primeira delas é a dada a seguir, sendo **X** = **A** ou **B**.

"Se uma trajetória a partir de um ponto de **X** encontra uma aresta **e**, do tipo **XX** num ponto **p** antes de chegar a outra aresta **e'**, do mesmo tipo, num ponto **q**, então devemos ter:

$\pi_e(p) \leq \pi_{e'}(q)$" **(Condição 1)**

Essa condição objetiva garantir que duas instâncias da transformação correspondentes a tempos diferentes não se cruzam.

Além disso procuramos fazer com que o Tempo de Acesso a um ponto de uma aresta **AA** reflita, na medida do possível o comprimento da trajetória desde **A** até ele. Idem para um ponto de uma aresta **BB** em relação a **B**. A finalidade disso é fazer com que nessas regiões não haja uma variação muito grande da velocidade ao longo de uma mesma trajetória. Em ambas as condições excluimos de consideração as arestas do tipo **AB**. Para elas não precisamos construir GTAs.

Além disso devemos manter o número de vértices de um GTA limitado por um k fixo se quisermos que representá-los (para todas as arestas) requeira um espaço apenas linear. Para obter um procedimento que atenda a essas três condições, comecemos observando que quando se faz a atribuição de (v_T,e_T) para um triângulo de P_A, qualquer de suas outras arestas ou pertence a **A** ou já foi feita base de um triângulo tratado anteriormente. Para uma aresta de **A** a função Tempo de Acesso deve ser identicamente nula e suponha que para os triângulos de P_A processados antes, os GTAs de todas as arestas já tenham sido determinados. Desse modo quando e_T é feita base, os GTAs das outras arestas de **T** já são conhecidos. No desenvolvimento que faremos a seguir, $\lambda_e(p)$ representará a fração:

(distância de p ao vértice inicial da aresta e / comprimento da aresta).

Para determinar, então o GTA de $e_T=(v_i,v_f)$, a partir do das outras duas arestas de **T**(**e'** e **e"**) podemos fazer o seguinte

a) Suponha que orientamos as arestas sempre do vértice de menor índice para o de maior. Nesse caso será verdade que uma das arestas(**e'**) tem o mesmo vértice inicial de e_T e a outra (**e"**) o mesmo vértice final. Vamos inicialmente nos preocupar com o GTA da porção de e_T entre v_i e p_T. Um ponto **p** dessa porção pertence a mesma trajetória que o ponto **q'** em **e'** distante dele exatamente, $(\lambda_e(p)/\lambda_e(p_T)).d_T$ onde d_T é a distância de v_T a p_T Assim, propomos que inicialmente se faça :

$$\pi_e(p) = \pi_{e'}(q') + (\lambda_e(p)/\lambda_e(p_T)) \bullet d_T, \quad \forall p \in (v_i,p_T)$$

(a)

Raciocínio análogo nos levaria, fazendo **q"** ser o ponto de **e"** pertencente a mesma trajetória que um **p** entre P_T e v_f, a:

$$\pi(p) = \pi(q'') + \left(\frac{1 - \lambda_e(p)}{1 - \lambda_e(p_T)}\right) d_T \quad \forall p \text{ em } (p_T,v_f).$$

(b)

Dessa forma como $d_T > 0$ a condição 1 é satisfeita em **T** e a função Tempo de Acesso representada nos GTAs varia ao longo de uma mesma trajetória com a distância percorrida por ela.

Obter uma representação computacional do GTA de **e** , significa simplemente, determinar a intersecção com **e** das trajetórias que passam por pontos de quebra de **e'** e **e"** e associar a cada um desses pontos o valor de $\pi_e(\,)$ calculado por **a** ou **b**. Portanto se os GTAs de **e'** e **e"** tem um número de vértices limitado por um **k** fixo, o custo dessa obtenção será O(1). Entretanto o número de vértices do GTA de **e** poderá ultrapassar **k**, o que vai exigir um procedimento adicional dado abaixo e ilustrado na figura 4 para **k** = 4.

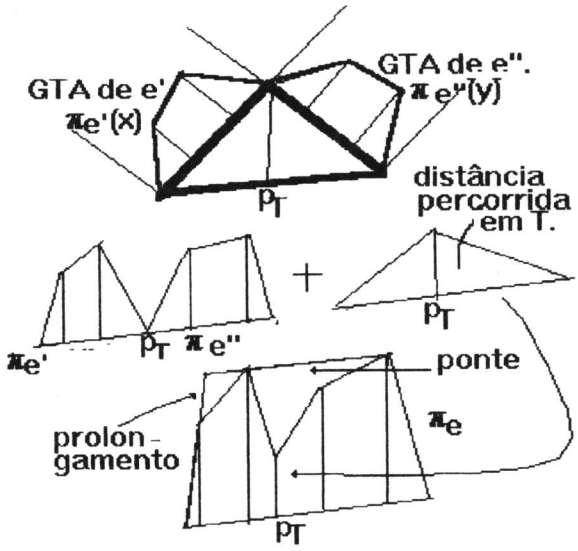

figura 4

1. Assumindo que $\pi_{e'}$ e $\pi_{e''}$ são funções côncavas (hipografo convexo), podemos fazer com que π_e também o seja, simplesmente encontrando a ponte ("bridge") entre as porções do gráfico de π_e em (v_i,p_T) e em (p_T,v_f) e substituindo o trecho abaixo dessa ponte por ela.

2.Se o gráfico (**G**) assim obtido tiver menos de k vértices ele será o GTA de **e**. Senão seja $(w_i,i=1,...,L<k)$ a sequência de vértices de G percorrido num dado sentido e (w_j,w_{j+1}) a ponte. Escolhamos de alguma

forma uma subsequência de L-k+4 vértices consecutivos $(w_l,w_{l+1},w_{l+2}, \ldots ,w_m)$ contendo (w_j,w_{j+1}) e substituamos o trecho de **G** entre esses vértices por (w_l,w'_1,w'_2,w_m) onde, $w'_1(w'_2)$ é a intersecção do prolongamento de (w_l,w_{l+1}) $((w_{m-1},w_m)$, respectivamente) ,com a reta suporte da ponte.

Ao fazer isto estamos majorando a função π_e já obtida e portanto a condição 1 continuará a ser atendida. Essa majoração vai atrasar a evolução ao longo das trajetórias principalmente dos pontos próximos ao vértice oposto a **e** e seu efeito poderá até ser o de reduzir concavidades das transformadas intermediárias. O procedimento de redução do número de vértices do GTA de **e,** dado acima, é certamente O(1) e assim podemos construi-lo em tempo constante. Para as arestas de $\mathbf{P_B}$ empregamos procedimento inteiramente análogo, só que começando em **B**. Como gastamos um tempo constante por aresta, a complexidade dessa etapa é **O(r)**.

2.4 - Transformação dos GTAs. Determinação de M.

Para ficar independente do caso o domínio do parametro da transformação é suposto ser sempre **[0,1].** Em vista disso,nesta etapa os GTA. são transformados, assumindo valores nesse intervalo , de forma a permitir que depois se possa obter, por interpolação, para cada ponto de $\mathbf{P}-\{v_1,v_n\}$, o valor do parametro da transformação quando ela o atinge. Essa transformação, além de promover essa normalização, precisa se preocupar em obter a propriedade explicitada a seguir.

Seja $(e_0,e_1,\ldots,e_m)$ a sequência de arestas cortadas por uma trajetória Γ qualquer, desde **A** até **B**, na ordem em que elas são atingidas a partir de **A**. Represente então, por p_i , $i=1,\ldots,m$ a intersecção de Γ com e_i e faça $t(p) \in [0,1]$ ser o parametro da transformação quando ela atinge **p**. Obviamente devemos ter

$$t(p_{i+1}) > t(p_i) \quad \text{(Condição 1')}$$

se quisermos evitar intersecções entre transformadas intermediárias distintas ou auto-intersecções em uma delas. Para conseguir isso, terminada a etapa de construção dos GTA calculamos inicialmente **M'** dado pela expressão:

$$\max_{e \subseteq \partial P_{AB}} \{\min_{v \in \Gamma_e} \{\alpha(e,v)\}\}$$

onde:

1. $\partial \mathbf{P_{AB}}$ é a fronteira da região $\mathbf{P_{AB}}$.

2. Se **e** = (v_i,v_f) e **T** o triângulo de $\mathbf{P_{AB}}$ que contem **e**, então:

a) $\Gamma_e=\{v_i,v_f\}$, se **e** é a aresta base de **T**.

b) Γ_e={vértice de **e** comum a aresta base de **T**}, em caso contrário

3. Se $\lambda_T = \lambda_{e_T}(p_T)$, $\phi_e=\pi_e \circ \lambda_e$, $\phi'_e(0)$ é a derivada à direita de ϕ_e em 0 e $\phi'_e(1)$ a derivada à esquerda em 1 então:

a) $\alpha(e,v)= \phi'_e(0)/\lambda_T$, se **e** não é base de **T** e $v=v_i$.

b) $\alpha(e,v)= \phi'_e(1)/(1-\lambda_T)$, se **e** não é base de **T** e $v=v_f$.

c) $\alpha(e,v)= \phi'_e(0) \bullet \lambda_T$, se **e** é base de T e $v=v_i$.

d) $\alpha(e,v)= \phi'_e(1) \bullet (1-\lambda_T)$, se **e** é base de **T** e $v=v_f$.

Determinado **M'**, o que pode ser feito em tempo linear, escolhe-se **M** > **M'** e obtem-se os **t(p)** da forma indicada abaixo. Essa forma de proceder é suficiente para que se tenha simultâneamente os **t(p)** em **[0,1]** e a **Condição 1'** atendida para todo i(Ver [Cruz(1993)]).

O procedimento indicado é o seguinte:

1. Para um ponto p numa aresta **e** de $\mathbf{P_A}$ fazemos $t(p) = \pi_e(p)/M$.

2. Se **p** está numa aresta **e'** de $\mathbf{P_B}$ então $t(p) = 1-\pi_e(p)/M$. A complementação a 1 é necessária pois em $\mathbf{P_B}$, os GTAs aproximam o comprimento de trajetórias até **B** e não a partir de **A.**

3. Para uma aresta **e** do tipo **AB**, que não tem GTA , fazemos, simplesmente, $t(p)=\lambda_e(p)$, assumindo sempre que ela está orientada de **A** para **B**.

Em termos computacionais, o que é feito é simplesmente, aplicar as operações indicadas em 1 e 2 , às ordenadas dos vértices dos GTA das arestas **AA** ou **BB,** conforme o caso.

Pode-se observar que para pares (p_i,p_{i+1}) com ambos os pontos em $\mathbf{P_A}$ ou $\mathbf{P_B}$, a **Condição 1'** é uma consequência imediata do fato dos GTAs safisfazerem a **Condição 1**. A complexidade da expressão do limite inferior **M'** resulta da maior dificuldade em satisfaze-la quando p_i e p_{i+1} pertencem a regiões distintas, como na passagem de $\mathbf{P_A}$ para $\mathbf{P_{AB}}$ e de $\mathbf{P_{AB}}$ para $\mathbf{P_B}$. É que nessas transições muda a maneira de obter **t(p)**.

Todos os procedimentos efetuados nesta etapa são **O(r)** completando-se assim, num tempo também dessa ordem , o pré- processamento. Na subseção seguinte se descreve como as transformadas intermediárias são obtidas a partir da estrutura montada.

2.5 - O Traçado das Instâncias Intermediárias da Transformação:

Tendo definido **t(.)** para os pontos de todas as arestas de $\mathcal{S}$ podemos extende-la para **P-{extremos comuns de A e B}**, por interpolação linear, procedendo, precisamente, da seguinte forma.

Seja **T** o triângulo que contém **p** , **e** e **e'** , arestas desse triângulo cortadas pela trajetória que passa por **p** em pontos **q** e **q'** distintos. Nesse caso definiremos

$$t(p) = \frac{\|p - q\|}{\|q' - q\|}.t(q') + \frac{\|p - q'\|}{\|q' - q\|}.t(q)$$

Fazendo isso, uma instância intermediária $C_z = \{ p \mid t(p) = z \}$, para todo **z** em **[0,1]** será uma curva conexa, sem auto intersecções e com extremidades comuns às de **A e B,** as quais podem portanto ser usadas para começar e encerrar o procedimento de determinação de seu traçado . Com respeito a forma dessa curva, seja **T** um triângulo qualquer atravessado por ela. Nesse triângulo considere as trajetórias de todos os pontos em suas três arestas onde o respectivo GTA (se houver) tenha um vértice (Ver figura 5).

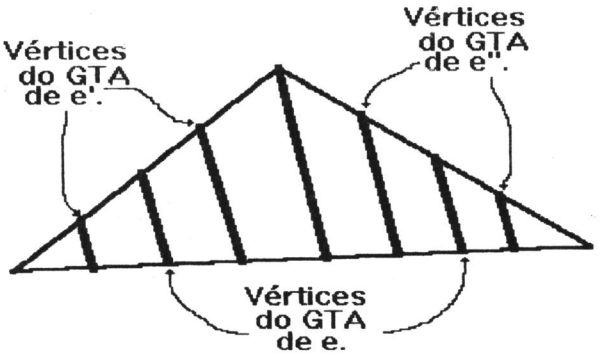

figura 5

Essas trajetórias definem trapézios (ou triângulos, que consideraremos trapézios degenerados) onde **t(.)** é uma função bilinear. Mais precisamente,expressando os pontos **(p)** de um desses trapézios **(θ)** em coordenadas convenientes **(x,y)**, temos

$$t(p) = axy + bx + cy + d,$$

sendo os parâmetros **(a,b,c,d)** determinados em função dos valores de **t(.)** nos vértices do trapézio. Assim C_z restrita a esse trapézio é parte da hiperbole H_z dada por

$$y = \frac{z + (bx + d)}{ax + c}.$$

$H_z \cap \theta$ tem no máximo duas componentes conexas como mostra a figura 6.

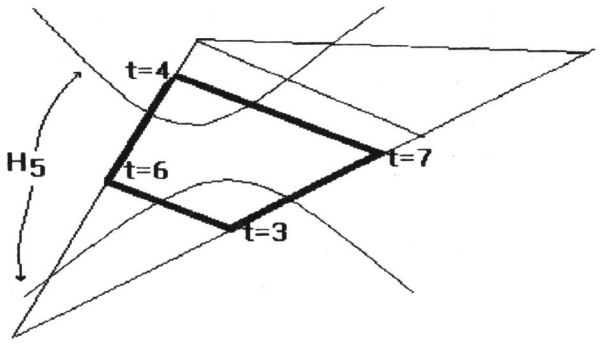

figura 6

Entretanto, para que haja de fato duas componentes é preciso que, tal como acontece no exemplo da figura, os dois vértices de maior (ou menor) valor sejam opostos. Tal situação, entretanto, fica excluída uma vez que a **Condição 1'** é satisfeita. Nesse caso será verdade também que, se aproximarmos $H_z \cap \theta$ pelo segmento de reta (S_z) que une seus pontos extremos então :

a) S_z varia continuamente com **z**, (o que não acontece no caso geral.)

b) $z \neq z' \Rightarrow S_z \cap S_{z'} = \varnothing$, isso em virtude da linearidade de **t(.)** em cada segmento da borda do trapézio.

Assim substituindo H_z por S_z não perdemos a continuidade do movimento nem a propriedade de instâncias referentes a **zs** distintos serem disjuntas. Se o número de trapézios de um triângulo, ou seja o número de vértices dos GTAs de suas arestas , for limitado por um **k** fixo, então encontrar o próximo trapézio a ser atravessado por uma C_z pode ser feito em tempo constante. Isso significa que o trabalho de determinar completamente C_z fica proporcional ao número de trapézios atravessados por ela, que nessas condições, será **O(r)** .

3. Deficiências e Vantagens. Um exemplo de Aplicação.

O objetivo deste trabalho é simplesmente mostrar que é possível obter em tempo **O(rlogr),** para duas poligonais na situação apresentada, uma tranformação contínua de uma na outra, com uma série de propriedades usualmente requeridas de algoritmos para a metamorfose de contornos. Para obter essa complexidade baixa entretanto, permitimos(e criamos) situações que podem comprometer a qualidade visual do processo de transformação, algumas das quais listamos a seguir:

1. A existência de gargalos relativamente extreitos determinados por se constringir o movimento ao interior do polígono **P** pode tomar mais agudas nas curvas intermediárias, "protuberâncias" existentes na inicial e final.

2. A estimação de **M** é feita sem levar em conta o comprimento das trajetórias em $\mathbf{P_{AB}}$ e a partir de superestimativas do comprimento delas em $\mathbf{P_A}$ e $\mathbf{P_B}$. Além disso, deve-se observar que ele é feito constante, enquanto poderia ser função das trajetórias. Assim, o valor de **M** obtido pode ser excessivamente alto fazendo, em particular, com que a travessia de $\mathbf{P_{AB}}$ seja consideravelmente mais lenta que as de $\mathbf{P_A}$ e $\mathbf{P_B}$.

3. Há uma série de escolhas que deixamos em aberto, a começar pela própria triangulação que não é única. Além disso não especificamos completamente como escolher as direções das trajetórias em cada triângulo. Disparidades na forma dos triângulos obtidos ou escolhas inadequadas para a direção das trajetórias em triângulos vizinhos podem prejudicar efetivamente, o resultado final.

limitar a representação dos GTA. Para **k = 4** os GTA podem ser armazenados guardando-se menos de **2(n+m)** pontos. Em relação a algoritmos que trabalham com estruturas quadráticas essa pode ser uma vantagem considerável. Além disso, se compararmos o número de vértices de uma $\mathbf{C_z}$ gerada da forma indicada na seção anterior com a gerada, para esse mesmo **z** , quando as trajetórias são percorridas com velocidade constante, verificamos que esse número pode até ser ligeiramente maior mas é em geral consideravelmente menor, tornando o traçado das $\mathbf{C_z}$ mais rápido.

2.Determinados pontos da curva inicial podem ser feitos corresponder a pontos da curva final com uma simples adaptação do método empregado aqui. Isso, mantendo-se a complexidade da montagem da estrutura e do traçado das curvas intermediárias.

3. É interessante notar ainda que tanto trajetórias como as $\mathbf{C_z}$ podem ser "suavizadas" substituindo-se cada "v" formado por um vértice não extremo e pelos pontos médios dos segmentos a ele adjacentes, pela Bezier quadrática cujos pontos de contrôle são esses (ver [Oliveira - Persiano (1987)]). Isso, sem perder as

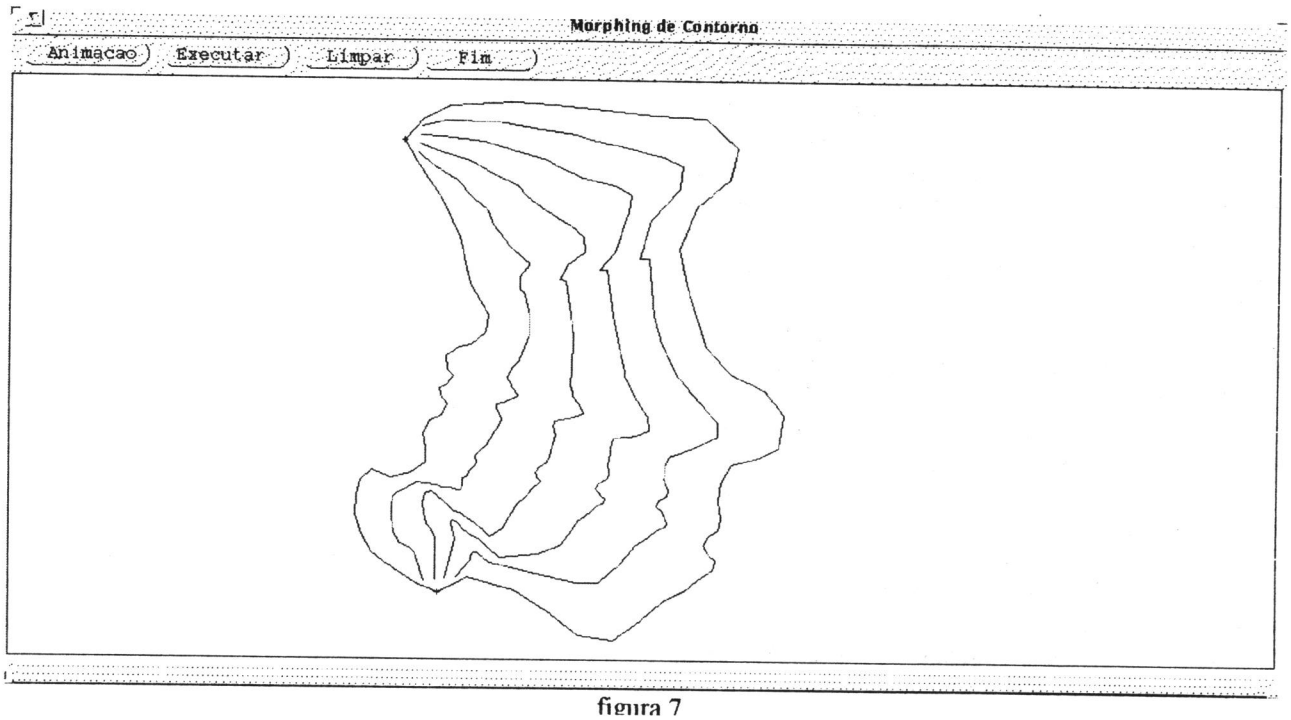

figura 7

Em vista dessas considerações preferimos que a aplicação dessa metodologia seja local como no exemplo dado na seção 1. Em contrapartida a metodologia apresentada aqui possui algumas potencialidades das quais listamos abaixo, as seguintes:

1. A primeira, é claro, diz respeito ao tamanho limitado da estrutura que tem que ser construida.. O tamanho dessa estrutura depende do **k** escolhido para

propriedades de não haver auto-intersecções e de trajetórias ou $\mathbf{C_z}$**s** distintas serem disjuntas. Entretanto, esse processo de suavização pode destruir a continuidade de transformação.

Finalmente apresentamos na figura 7, um exemplo de aplicação da metodologia exposta neste trabalho, obtida por uma implementação em ambiente Unix (SPARCstation2).

Bibliografia:

1.Chew,L.P.; Constrained Delaunay Triangulations, Algorithmica 4 : 97-108(1989).

2. Cohen,L.; On Active Contour Models and Balloons; CVGIP:Image Understanding, 53(2), 211-218(1991).

3. Corel Systems Corporation; Corel-Draw 2.0(1990).

4 Cruz,A.J.A.; Um Sistema para a Transformação de Contornos, dissertação de Mestrado em Engenharia de Sistemas e Computação, COPPE/UFRJ(Em preparação)(1993).

5. Micrografx, Inc. ; Designer 3.1(1991).

6. Fuchs,H. & Kedem, Z.M. & Uselton,S.P.; Optimal Surface Reconstruction from Planar Contours, Comm. of ACM, 20(10),693-702(1977).

7. Software Publishing Corporation; Harvard Graphics 3.0(1991).

8. Kass,M. & Witkin, A. & Terzopoulos,D.; Snakes: Active Contour Models, International Journal of Computer Vision, 1(3), 321-331(1988).

9. Keppel,E.; Approximating Complex Surfaces by Triangulation of Contour Lines, IBM Journal of Research and Development,19,2-11(1975).

10. Oliveira, A.A.F. & Persiano, R.C.M.; Suavização de Poligonais que Representam Curvas de Nível de uma Superfície: Um Método que Evita Intersecções, Anais do SIBGRAPI I, Itatiaia-RJ(1987).

11.Oliveira, A.A.F.;Reconstrução a partir de Secções Paralelas: Associando Arestas por meio de uma Triangulação 3-D; Relatório Técnico do Programa de Sistemas e Computação da COPPE/UFRJ (em preparação)(1993).

12.Preparata,F.P. & Shamos,M.I.; An Introduction to Computational Geometry, Springer-Verlag(1985).

13. Rossignac,J. & Kaul,A.; Solid Interpolating Deformations: Construction and Animation of PIPs. Proc. Eurographics 91, 493-505(1991).

14.Sederberg,T. & Greewood,E.; A Physically Based Approach to Shape Blending; Computer Graphics 26,2: 25-34(1992).

15. Seidel, R.; Computational Geometry, notas de aula, UCBerkeley(1989).

16. Wolberg,G.; Digital Image Warping, Computer society Press(1990).