Melhoramento de Caminhos na Geração Automática de Visitas Guiadas

Sofia Reis
sofiareis@hotmail.com

Manuel Próspero dos Santos
ps@di.fct.unl.pt

CITI - Centro de Informática e Tecnologia da Informação
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa

Sumário

Neste artigo apresenta-se um novo método que permite melhorar o caminho percorrido por um utilizador num modelo de volumetria, reduzindo ou eliminando os desvios desnecessários. Pretende-se que o resultado da aplicação deste método, mais concretamente numa visita guiada gerada automaticamente, torne a deslocação mais agradável.

O caminho que constitui a entrada do algoritmo de melhoramento é um conjunto ordenado de pontos. Considera-se que o modelo foi previamente convertido numa grelha de voxels, correspondendo cada um dos pontos do caminho ao centro de um dos voxels da grelha. Os pontos do caminho podem ser obtidos através do algoritmo de pesquisa A, fornecendo-se para tal apenas os primeiro e último pontos do caminho.*

O algoritmo de melhoramento recebe um caminho inicial que contém, eventualmente, desvios desnecessários. A partir desse conjunto ordenado de pontos, o algoritmo gera um novo conjunto onde os desvios desnecessários foram reduzidos ou eliminados. No entanto, serão conservados aqueles desvios que impeçam o utilizador de chocar com eventuais obstáculos arquitectónicos do modelo. Com base no conjunto de pontos gerados pelo algoritmo, constrói-se então, por recurso a uma curva interpoladora, o novo caminho melhorado.

Palavras-chave

Visitas guiadas, modelação geométrica, voxels, pesquisa A, detecção de colisões, curvas B-Spline.*

1. INTRODUÇÃO

O crescente interesse por réplicas virtuais de museus consagrados leva à oferta de percursos gerados automaticamente para substituir ou complementar a navegação do utilizador final processada unicamente por meio das técnicas usuais de deslocamento no espaço virtual 3D. Com efeito, sem tais ajudas à navegação, o utilizador acabaria por, de forma razoavelmente rápida, atingir um estado de cansaço pelo esforço necessariamente despendido, já para não referirmos a grande probabilidade de ele se desorientar e desistir ou terminar a visita sem ter visto tudo aquilo a que se propusera.

No trabalho que está na base deste artigo tomou-se, como objecto de tratamento, o caso real de museus de pintura [Reis06]. Tipicamente, estes museus são constituídos por salas ou espaços temáticos por onde o visitante poderá circular, observando as obras de arte que aí se encontram expostas. Com a finalidade de se gerar automaticamente um caminho de circulação que leve o visitante a ver todos os quadros existentes ou apenas colecções ou selecções previamente indicadas (e.g., as pinturas de um determinado século), construiu-se uma aplicação exigindo as seguintes entradas: a geometria do modelo, o ponto onde a visita terá início, a identificação de cada pintura e a sua localização no modelo. Assim, é apenas através da geometria que se obtém o conhecimento sobre os diferentes espaços de exposição (células) e a ligação entre eles (portais).

O caminho gerado será, então, uma optimização dos percursos possíveis. Não basta, no entanto, obter-se uma sequência de salas a percorrer (chamemos-lhe caminho de alto nível) nem a indicação da melhor localização em que cada pintura poderá ser observada. Será, pois, necessário criar uma trajectória suave ao longo do tempo (caminho de baixo nível) e sem apresentar desvios desnecessários, o que tornará muito mais agradável a visita ao museu virtual. Nesse sentido, concebeu-se e aplicou-se uma técnica destinada a melhorar um caminho (de baixo nível) gerado a partir das condições anteriormente indicadas. Será este o foco principal do presente artigo.

2. TRABALHO RELACIONADO

Em [Kama03] aborda-se a deslocação de um objecto ao longo de uma trajectória, assim como o tipo de curva mais apropriado para a representação dessa trajectória. No entanto, as estratégias apresentadas por aqueles autores apenas permitem tornar a deslocação do utilizador mais suave em comparação com a utilização de uma in-

terpolação linear dos pontos calculados para o caminho. Se existirem desvios desnecessários no caminho, para a esquerda e para a direita, esses desvios não são eliminados.

Em [Andú04] é abordada a geração automática de visitas guiadas a modelos complexos. Para a trajectória da visita guiada é utilizada uma interpolação de Hermite, com base nos pontos do caminho percorrido. A utilização de uma interpolação de Hermite atenua os desvios desnecessários do utilizador para a esquerda e para a direita, mas não os consegue eliminar.

Tsai-Yen Li e Hung-Kai Ting apresentam, em [Li00], uma estratégia que ajuda o utilizador a evitar os obstáculos que se lhe deparam na deslocação dentro de um modelo. No entanto, a principal atenção deste artigo é prestada ao factor tempo. Pretende-se que o utilizador se desloque através do modelo o mais rapidamente possível, ajudando-o a evitar ou a contornar os obstáculos. Pelo contrário, o aumento do conforto, através da geração de um caminho suave e sem desvios desnecessários não é aí relevante.

3. CONTEXTO PARA O ALGORITMO DE MELHORAMENTO DE CAMINHOS

A necessidade do algoritmo de melhoramento de caminhos surgiu aquando da elaboração da elaboração da dissertação de mestrado intitulada "Visitas guiadas através de modelos de volumetria" [Reis06], onde foi abordado o caso particular de visitas guiadas na representação virtual de um museu de pintura. Nesse tipo de visitas, as peças expostas, sendo as partes relevantes do museu, são visualizadas da melhor forma possível, ou seja, de maneira a que a sua dimensão e o contexto em que estão inseridas se tornem perceptíveis ao utilizador.

(a) (b)

Figura 1: Imagens de uma visita guiada ao modelo virtual de um museu de pintura

Na Figura 1 podemos ver duas imagens de uma visita guiada ao modelo de museu de pintura. Na Figura 1(a) as pinturas expostas são S. Teotónio e Santo Franciscano [IPM06], enquanto que na Figura 1(b) se mostra o caminho a ser percorrido pelo utilizador. A configuração das salas criadas para este último modelo pode ser observada na vista geral apresentada na Figura 2. Em frente de cada uma das pinturas previamente seleccionadas é efectuada, automaticamente, uma paragem de alguns segundos para que o utilizador a possa observar com atenção. Para se obter esta localização privilegiada da câmara, bem assim como a sua orientação, recorre-se ao método de cálculos sucessivos da entropia [Reis06]. Este método pode ser adaptado à observação de objectos que, contrariamente a um quadro, não sejam simples planos.

O algoritmo de melhoramento de caminhos foi concebido com o objectivo de tornar a deslocação do utilizador mais agradável, através da redução dos desvios desnecessários para a esquerda e para a direita, no caminho percorrido. No entanto, o algoritmo de melhoramento de caminhos pode ser aplicado a qualquer tipo de modelo de volumetria e não apenas a modelos de volumetria representativos de museus de pintura.

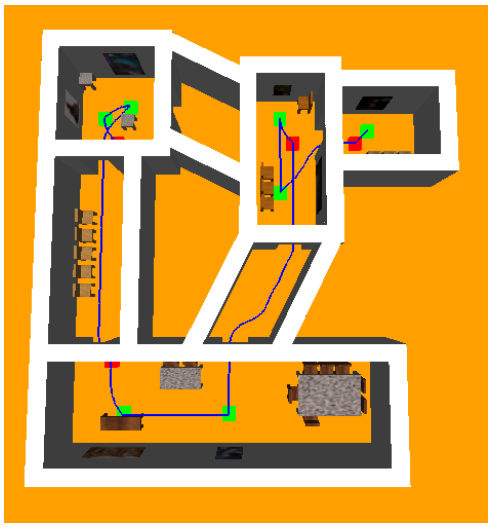


Figura 2: Vista geral das salas do museu, visualizando-se o trajecto calculado para a visita guiada

Iremos, seguidamente, enunciar as características que o caminho inicial que se pretende melhorar deverá ter de maneira a poder ser submetido ao algoritmo de melhoramento de caminhos.

4. CARACTERIZAÇÃO DO CAMINHO INICIAL

O caminho inicial (de baixo nível), formado por uma sucessão de pontos por onde a câmara irá passar, é a entrada do algoritmo de melhoramento de caminhos.

Considera-se que o modelo, por onde o caminho está delineado, é um modelo de volumetria que foi previamente organizado numa grelha de *voxels* com forma cúbica.

Iremos, de seguida, proceder à caracterização do caminho inicial no se refere à grelha de *voxels* e à conectividade entre os *voxels*.

4.1 A grelha de *voxels*

Considerando que R representa o conjunto dos números reais, R^3 representará o espaço tridimensional contínuo.

Sendo Z o conjunto dos números inteiros, Z^3 será o espaço tridimensional discreto.

Portanto, pode-se visualizar Z^3 como uma grelha de pontos que está contida em R^3. Um ponto na grelha é definido pelas suas coordenadas cartesianas (x, y, z). Um *voxel* é um cubo centrado num dos pontos da grelha (x, y, z), definindo-se como a região contínua (u, v, w) tal que [Huan98, Jone96, Kauf87a, Kauf87b]:

- $x - 0,5 < u \leq x + 0,5$
- $y - 0,5 < v \leq y + 0,5$
- $z - 0,5 < w \leq z + 0,5$

Os *voxels* podem ser vazios ou cheios. Caso o *voxel* seja intersectado pela geometria de algum dos objectos do modelo, então esse *voxel* diz-se cheio, ou preto. Caso o *voxel* não seja intersectado pela geometria de nenhum dos objectos do modelo, então está-se perante um *voxel* vazio, ou branco [Fole00, Huan98].

Na Figura 3 encontram-se dois triângulos aos quais está sobreposta a sua representação em *voxels*. Os *voxels* representados na figura são apenas os cheios. Em [Jone01a, Jone01b, Jone01c] são descritas várias estratégias para a organização de um modelo em *voxels*.

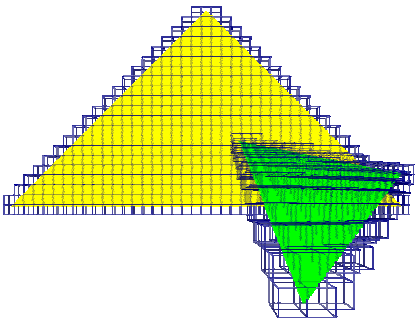


Figura 3: Dois triângulos no espaço 3D e a sua correspondente representação em *voxels*

Cada ponto calculado para o caminho inicial corresponde ao centro de um dos *voxels* vazios do modelo.

4.2 A conectividade entre os *voxels*

A conectividade entre os *voxels*, no caminho inicial, determina a forma como a câmara se pode deslocar.

(a) A câmara pode mover-se para uma vizinhança de 4 *voxels*

(b) A câmara pode mover-se para uma vizinhança de 8 *voxels*

Figura 4: Possibilidades de movimentação da câmara

O algoritmo de melhoramento de caminhos considera que a câmara está sempre à mesma altura, ou seja, num mesmo plano horizontal. A conectividade entre os *voxels* pode ser de dois tipos:

- No plano horizontal, a câmara pode deslocar-se em duas direcções ortogonais: para a frente, para trás, para a esquerda ou para a direita. Por outras palavras, a câmara pode mover-se para qualquer um dos 4 *voxels* adjacentes ao *voxel* em que se encontra (Figura 4(a)).
- Caso a câmara também se possa deslocar na diagonal, isso significa que se pode mover para uma vizinhança de 8 *voxels*, em relação ao *voxel* em que se encontra (Figura 4(b)).

O algoritmo de melhoramento de caminhos recebe, portanto, um caminho inicial em que a câmara se pode mover para uma vizinhança de 4 ou 8 *voxels*.

4.3 Geração dos pontos correspondentes ao caminho inicial

Em [Russ03], em [Pear84] e em [Rich91] encontram-se descritos diversos algoritmos de pesquisa que podem ser usados para geração de um caminho inicial. Em [Reis06] optou-se pela pesquisa A*.

Recorrendo a um desses algoritmos será necessário escolher apenas o primeiro ponto e o último ponto do caminho inicial. Cada *voxel* corresponde a uma possível posição da câmara. Para efeitos da pesquisa A*, constrói-se, então, um grafo em que os vértices são os pontos centrais dos *voxels* e os ramos são a conectividade entre os *voxels*. A câmara pode ocupar apenas os *voxels* vazios, pelo que os *voxels* cheios devem ser excluídos deste grafo. A pesquisa A* irá, então, encontrar o caminho mais curto desde o *voxel* que corresponde ao primeiro ponto do caminho até ao *voxel* que corresponde ao último ponto.

Caso se recorra à pesquisa A*, como aqui acontece, será mais vantajoso que a câmara se possa deslocar para uma vizinhança de 8 *voxels*. Esta conclusão encontra-se demonstrada em [Reis06].

Agora que os aspectos relativos ao caminho inicial foram clarificados, há que determinar como a câmara se irá deslocar ao longo dos pontos desse caminho. Existem, para isso, várias estratégias.

4.4 Construção do caminho

Uma vez encontrados os pontos pelos quais a câmara irá passar, há que construir o caminho propriamente dito. A maneira mais simples de o conseguir é através de uma interpolação linear, unindo-se entre si os diversos pontos por segmentos de recta.

A interpolação linear, do ponto de vista matemático, é fácil de implementar. Além disso, o tempo de execução é também reduzido, uma vez que os cálculos são simples. No entanto, pela não garantia de continuidade da primeira derivada, o caminho obtido através deste método pode não ser suave, o que causará algum desconforto e estranheza ao utilizador. O percurso será mais agradável caso o caminho não apresente pontos angulosos. Para tal, é necessário recorrer a uma curva interpoladora de maior grau.

Existem vários tipos de curvas que podem agrupar-se em duas grandes categorias: as curvas geradas por aproximação de pontos e as curvas geradas pela interpolação dos mesmos.

Na geração de uma curva por aproximação não se obtém uma passagem obrigatória pelos pontos de controlo (Figura 5(a)), ao passo que uma curva gerada por interpolação passa obrigatoriamente por todos os pontos de controlo (Figura 5(b)).

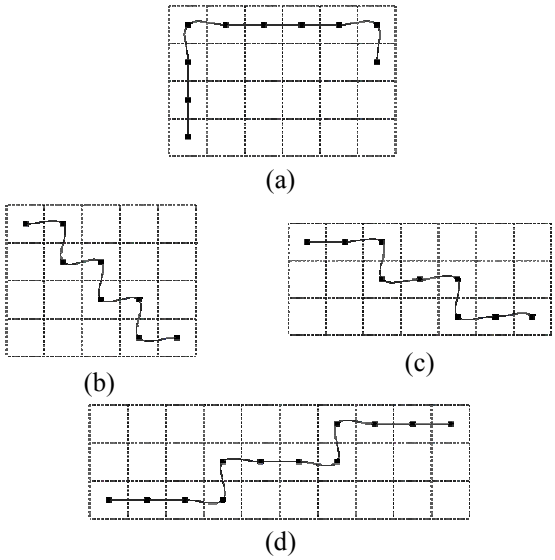
(a) Curva gerada por aproximação

(b) Curva gerada por interpolação

Figura 5: Aproximação e interpolação de curvas

Entre as curvas geradas por interpolação destacam-se as de Kochaneck-Bartels, que foram especialmente desenvolvidas para a geração de caminhos no contexto da animação [Fole00, Hear97].

Na Figura 6 encontram-se quatro caminhos, todos gerados com recurso às curvas de Kochanek-Bartels. O corte horizontal da grelha de *voxels* é representado a traço interrompido. Repare-se que o facto da curva de Kochaneck-Bartels interpolar obrigatoriamente todos os pontos de controlo pode criar alguns problemas. No caminho (a), a disposição dos pontos de controlo não cria qualquer situação indesejável: o utilizador segue sempre a direito, à excepção dos dois pontos intermédios onde muda de direcção. Mas, no caminho (b), a situação já não é tão satisfatória. Na verdade, devido à disposição dos pontos de controlo, e uma vez que é necessário passar por todos eles, o utilizador é forçado a curvar constantemente para a esquerda e para direita. Não será um percurso agradável. Nos caminhos (c) e (d) repetem-se situações semelhantes. A câmara oscilará bastante, resultando daí uma sensação de desconforto para o utilizador.



(a)

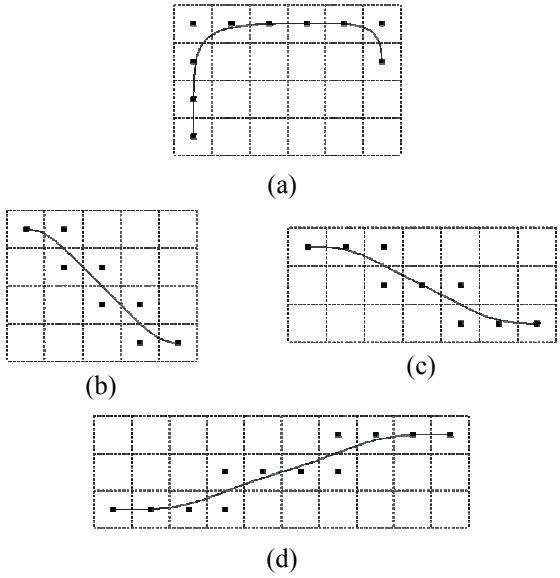
(b)

(c)

(d)

Figura 6: Caminhos desenhados com curvas de Kochaneck-Bartels

Portanto, embora as curvas de Kochaneck-Bartels tenham sido especialmente desenvolvidas para a geração de caminhos no contexto da animação, neste caso particular parecem não fornecer bons resultados.



(a)

(b)

(c)

(d)

Figura 7: Caminhos desenhados com curvas B-Spline

No entanto, se em vez de utilizarmos curvas geradas por interpolação utilizarmos curvas geradas por aproximação, o caminho gerado tornar-se-á mais agradável.

Na Figura 7 encontram-se quatro caminhos gerados com os mesmos pontos de controlo dos da Figura 6. A curva utilizada é B-Spline, gerada por aproximação e fazendo-se interpolar apenas os pontos extremidade.

Tal como se pode depreender pela Figura 6 e pela Figura 7, os caminhos desenhados com as curvas B-Spline reduzem significativamente os desvios da câmara para a esquerda e para a direita em relação aos caminhos desenhados com as curvas de Kochaneck-Bartels.

Para gerar uma curva B-Spline recorre-se às Equações(1), (2) e (3) [Hear97, Hill01].

$$P(t) = \sum_{k=0}^{L} P_k N_{k,m}(t) \tag{1}$$

$$N_{k,m}(t) = \left(\frac{t - t_k}{t_{k+m-1} - t_k}\right) N_{k,m-1}(t) + \left(\frac{t_{k+m} - t}{t_{k+m} - t_{k+1}}\right) N_{k+1,m-1}(t) \tag{2}$$

$$N_{k,1}(t) = \begin{cases} 1 & \text{se } t_k < t \le t_{k+1} \\ 0 & \text{caso contrário} \end{cases} \tag{3}$$

Os pontos de controlo são os pontos de P_0 até P_L, em número de $L + 1$, e $N_{k,m}(t)$ é a fórmula geral, recursiva, dos respectivos pesos (*blending functions*).

Fazendo variar t no intervalo de zero a um obtêm-se os vários pontos da curva B-Spline.

A ordem das funções B-Spline é indicada pelo parâmetro m, não podendo m exceder o número de pontos de controlo.

O cálculo exige ainda um vector de nós (*knot vector*), dado por $\boldsymbol{T} = (t_0, t_1, t_2, \ldots, t_{L+1})$.

Na Figura 8 encontram-se três curvas B-Spline. Cada ponto de controlo é representado por um pequeno quadrado. Todas as curvas têm catorze pontos de controlo. Apenas o parâmetro *m* se altera. Como se pode ver, à medida que *m* aumenta a curva resultante fica menos "presa" aos pontos de controlo. O parâmetro *m* será de grande importância para a geração do caminho do utilizador, como se verá adiante.

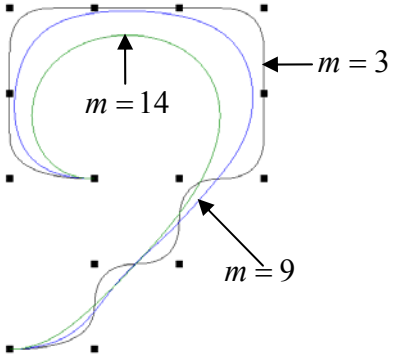


Figura 8: Curvas B-Spline com os mesmos pontos de controlo mas com diferentes ordens das funções B-Spline

No entanto, a curva B-Spline não é, ainda assim, capaz de resolver inteiramente o problema relativo às sucessivas oscilações na trajectória. Repare-se na Fgura 7(d), em que o parâmetro *m* da curva B-Spline é igual a 12. Se se olhar com atenção, nota-se que ainda há ligeiros desvios para a esquerda e para a direita, uma vez que existem vários pontos de inflexão. Se *m* diminuir, por exemplo de 12 para 8, esses desvios acentuam-se, tal como se pode observar na Figura 9.

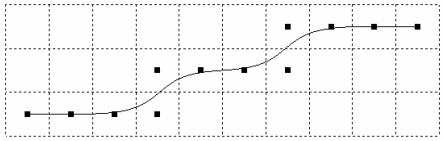


Figura 9: Caminho desenhado com curva B-Spline e com o parâmetro *m* igual a 8

Portanto, uma vez que a utilização de curvas B-Spline não resolve inteiramente o problema, é necessário recorrer a medidas adicionais. Para tal, foi criado o algoritmo de melhoramento de caminhos.

5. ALGORITMO DE MELHORAMENTO DE CAMINHOS

A Figura 9 mostra-nos que nem as curvas por aproximação são suficientes para reduzir os pequenos desvios desnecessários da trajectória do utilizador ao longo do caminho. Ora, uma observação atenta dessa Figura permite-nos detectar um padrão de posicionamento dos pontos. A cada cinco pontos, o padrão repete-se. Na Figura 10(a) estão assinaladas as zonas de repetição dos padrões. A zona da terceira repetição, assinalada com um padrão ponteado, é incompleta, uma vez que o quinto ponto não se repete.

Dado que existe este padrão, pode estabelecer-se que os pontos da Figura 10 configuram um possível segmento de recta. Esse segmento de recta começaria no primeiro ponto e acabaria no último ponto (Figura 10(b)). Se o utilizador seguisse ao longo deste hipotético segmento de recta, os desvios desnecessários no caminho seriam totalmente eliminados.

5.1 Obstáculos no caminho

Portanto, se houvesse uma maneira de reconhecer quando os pontos configuram um segmento de recta, isso seria vantajoso. No entanto, não é simples reconhecer este tipo de situações. Por exemplo, na Figura 11 não existe um padrão na disposição dos pontos que se mantenha desde o início da curva até ao seu fim. Se, erradamente, unís-semos o primeiro ponto ao último ponto de controlo através de um segmento de recta, de forma a eliminar todas as oscilações, isso faria com que o utilizador embatesse na parede, representada, nessa Figura, com um padrão de tijolos. Ora, pretende-se eliminar apenas os desvios desnecessários. No caso da Figura 11 há desvios que são indispensáveis para evitar obstáculos e esses não podem ser eliminados.

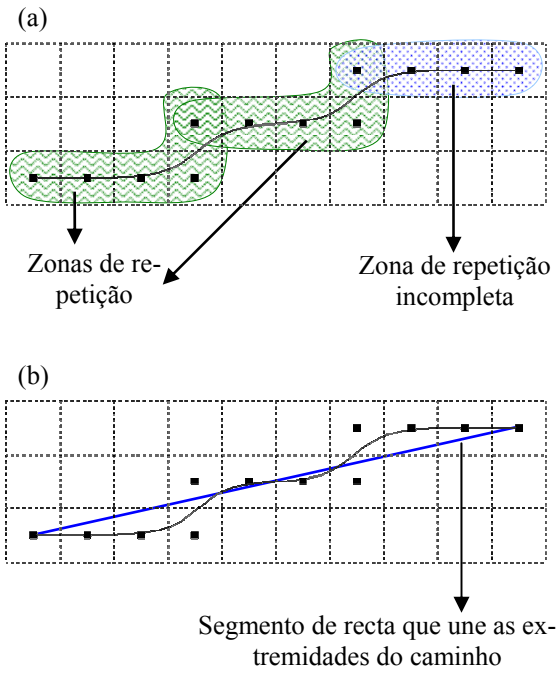


Figura 10: Caminho com as zonas de repetição assinaladas

No entanto, embora na sua totalidade os pontos da Figura 11 não configurem um segmento de recta, existem subconjuntos desses pontos para os quais isso pode acontecer, tal como se mostra na Figura 12. Aí, o caminho foi decomposto em quatro segmentos de recta assinalados a cheio. Em cada segmento de recta existe um padrão na disposição dos pontos. Sendo assim, poder-se-ia estabelecer que, sempre que se encontrasse um padrão repetitivo de posicionamento dos pontos, estar-se-ia perante um segmento de recta. Tanto na Figura 9 como na Figura 12 existem padrões de posicionamento repetitivos que são simples de detectar. Só que nem sempre estes padrões são assim tão fáceis de se reconhecer.

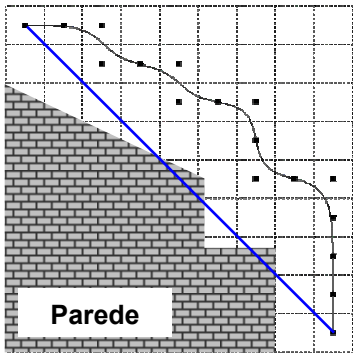


Figura 11: Segmento de recta incorrecto no caminho

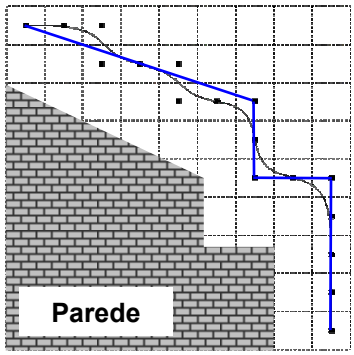


Figura 12: Segmentos de recta correctos no caminho

Na Figura 13 encontra-se um conjunto de pontos que, no seu conjunto, pode ser uma boa aproximação de um segmento de recta. No entanto, não seria trivial detectar o padrão repetitivo de posicionamento neste conjunto de pontos em particular.

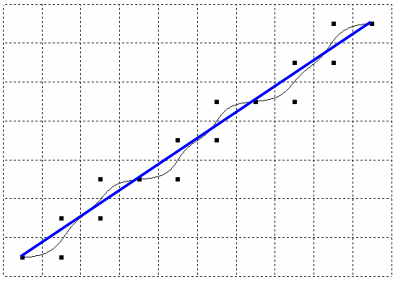


Figura 13: Caminho em que o padrão repetitivo de posicionamento dos pontos é difícil de detectar

O problema que se põe aqui é o inverso daquele que o algoritmo de Bresenham resolve. Neste último, a partir dos pontos inicial e final de um segmento de recta tem de se decidir quais os pixels a colorir no ecrã, de maneira a que esses pixels, no seu conjunto, desenhem uma linha [Fole00, Hill01]. Para o caso presente, pode-se fazer a equivalência entre *voxels* e *pixels*. Parte-se de um conjunto de *voxels*, que servem de base ao caminho, e pretende-se descobrir se alguns desses *voxels* configuram possíveis segmentos de recta.

5.2 Descrição do algoritmo

Para a resolução deste problema concebemos um algoritmo destinado ao melhoramento de caminhos. Este algoritmo recebe um conjunto de pontos ordenados, que correspondem a centros de *voxels*, e consegue reconhecer, sem recurso a análise de padrões, quando um determinado subconjunto desses pontos dá origem a um segmento de recta. O conjunto de pontos ordenados é referenciado no algoritmo como *Caminho*, uma vez que é com base nestes pontos que se desenha o caminho. O número de pontos deve ser superior a 1. O algoritmo de melhoramento de caminhos desenvolve-se ao longo dos seguintes passos:

1. Começar no primeiro ponto do *Caminho*. O primeiro ponto do *Caminho* é o ponto P_0.
2. Criar um apontador, *i*, para o primeiro ponto do caminho, ou seja, P_0. O ponto para o qual *i* aponta é P_i.
3. Se P_i for o último ponto do caminho, então avançar para o passo 7.
4. Avançar o apontador *i* para o próximo ponto do *Caminho*.
5. Verificar se os pontos intermédios entre P_0 e P_i formam um segmento de recta. Para isso, procede-se da seguinte maneira:
   - 5.1. Unir P_0 a P_i através de um segmento de recta. Se todos os *voxels* de P_{0+1} até P_{i-1} forem intersectados pelo segmento de recta que une P_0 a P_i, então o conjunto de pontos de P_0 a P_i configura um segmento de recta. Numa fase seguinte, o comprimento deste segmento poderá vir ainda a aumentar, desde que o algoritmo consiga descobrir mais pontos verificando essas condições.
   - 5.2. Se nem todos os *voxels* de P_{0+1} até P_{i-1} forem intersectados pelo segmento de recta que une P_0 a P_i, então o conjunto de pontos de P_0 a P_i não configura um segmento de recta. No entanto, o conjunto de pontos de P_0 a P_{i-1} configura um segmento de recta.
     - 5.2.1. Descobre-se que existe um segmento de recta de P_0 a P_{i-1}.
     - 5.2.2. P_{i-1} é renomeado e passa a funcionar como novo ponto P_0, de modo a tentar-se descobrir, a partir daí, mais um segmento de recta.
     - 5.2.3. Tenta-se descobrir mais um segmento de recta a partir de P_{i-1}.
6. Caso P_i não seja o último ponto do *Caminho*, voltar ao passo 4.
7. Os pontos que vão de P_0 até ao ponto P_i configuram o último segmento de recta do caminho.

Vamos agora clarificar um pouco mais o passo 5.1 do algoritmo de melhoramento de caminhos.

Tal como se refere neste passo, para que os pontos de P_0 a P_i configurem um segmento de recta é necessário que todos os *voxels* de P_{0+1} até P_{i-1} sejam intersectados pelo

segmento de recta que une P_0 a P_i. Esta situação é mais fácil de visualizar em duas dimensões.

Repare-se na Figura 14(a). Os *voxels* correspondentes aos pontos entre P_{0+1} e P_{i-1} são todos intersectados pelo segmento de recta que une P_0 a P_i. Portanto, os pontos de P_0 a P_i constituem, no seu conjunto, uma boa aproximação de um segmento de recta.

Na Figura 14(b) nem todos os *voxels* correspondentes aos pontos entre P_{0+1} e P_{i-1} são intersectados, pelo que não se está perante um segmento de recta.

A situação da Figura 15 é especial. O padrão repetitivo de posicionamento dos pontos denuncia um segmento de recta. No entanto, nem todos os *voxels* correspondentes aos pontos entre P_{0+1} e P_{i-1} são intersectados pelo segmento de recta que une P_0 a P_i. Para resolver este caso estabelece-se um valor de tolerância (necessariamente inferior a 50%), que aumentará as dimensões do *voxel*. Desta forma, todos os *voxels* correspondentes aos pontos entre P_{0+1} e P_{i-1} poderiam passar a ser intersectados. Portanto, na Figura 15 os pontos de P_0 a P_i configuram um segmento de recta. Repare-se que, ao serem retirados os *voxels* não intersectados por esse segmento, pode considerar-se que se permite que a câmara se mova na diagonal.

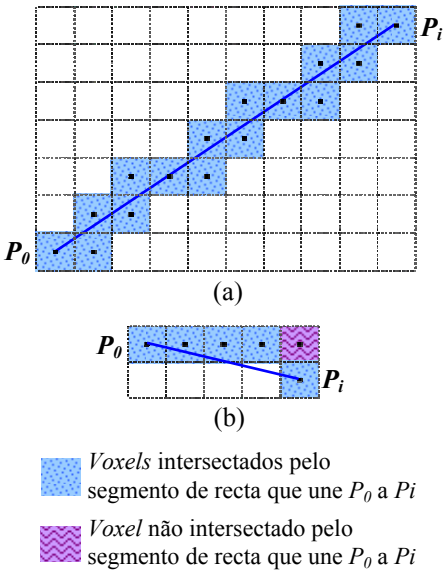


Figura 14: Dois exemplos de caminhos

Na Figura 16 encontra-se a aplicação do algoritmo de melhoramento de caminhos a um conjunto de pontos. Através de uma primeira aplicação do algoritmo descobre-se um primeiro segmento de P_0 a P_{15}. Depois de se estabelecer que P_{15} é o novo ponto inicial, pode-se descobrir outro segmento, de P_{15} a P_{18}. De P_{18} a P_{23} existe um novo segmento. De P_{23} a P_{27} existe um outro. De P_{27} a P_{29} encontra-se mais um. E, finalmente, de P_{29} a P_{30} surge o último segmento.

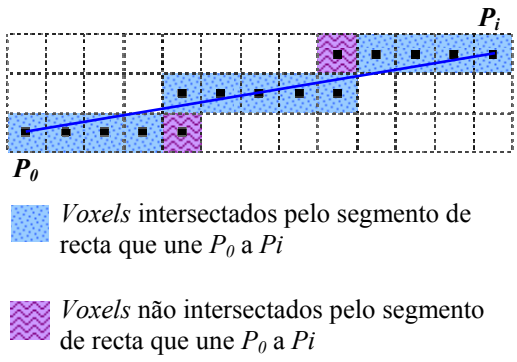


Figura 15: Caminho onde a câmara, além de se mover na horizontal e na vertical, também se pode mover na diagonal

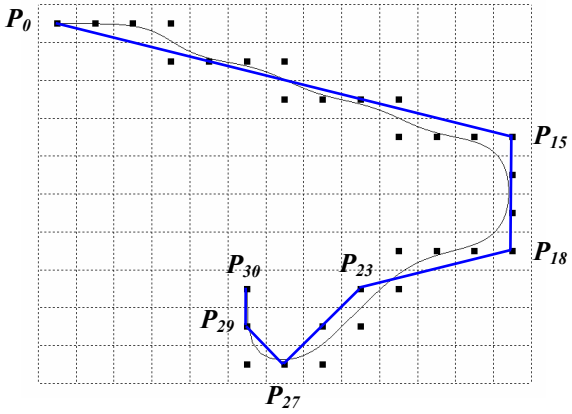


Figura 16: Algoritmo de melhoramento de caminhos aplicado a um conjunto de pontos

A aplicação do algoritmo de melhoramento de caminhos ao conjunto de pontos da Figura 11 produz o resultado já apresentado na Figura 12.

5.3 O novo caminho

Há agora que estabelecer como incorporar estes segmentos de recta no caminho. Não basta, simplesmente, deslocar a câmara ao longo dos segmentos de recta, uma vez que há descontinuidades na direcção sempre que transita para o segmento seguinte. O que sugerimos é a geração de uma nova curva, com outros pontos de controlo e posicionados de forma uniforme ao longo do segmento de recta. O número de pontos a posicionar ao longo do segmento é o seguinte:

$$N = \frac{D_{P_0 P_{Fim}}}{L} + 1 \qquad (4)$$

N é o número de pontos de controlo a posicionar ao longo do segmento de recta.

$D_{P_0 P_{Fim}}$ é a distância do primeiro ponto do segmento ao último.

L é o comprimento da aresta do *voxel*.

O valor de N é aproximado às unidades, de modo a que o número final de pontos seja sempre um inteiro.

Para posicionar os N pontos de controlo pode recorrer-se a uma interpolação linear entre os pontos inicial (P_0) e final (P_{Fim}) do segmento. Na forma paramétrica, o incremento I da variável t, para obtenção dos vários pontos de controlo, será dado pela Equação (5).

$$I = \frac{1}{N-1} \tag{5}$$

Há que realçar que o número de pontos de controlo modificados (N) pode ser diferente do número de pontos de controlo originais. Isto acontece para que a distância entre um ponto de controlo e o seguinte, ao longo do segmento de recta, seja a mais próxima possível de L. Em geral, não se consegue que o espaçamento entre os pontos, ao longo do segmento de recta, seja igual ao comprimento da aresta do *voxel*, mas apenas aproximadamente igual.

No Quadro 1 apresentam-se os pontos de controlo de caminhos simples, em que existe apenas um único segmento de recta, e os pontos de controlo modificados para esses caminhos simples. Repare-se que só no último caso é que a distância entre os pontos de controlo modificados é igual à aresta do *voxel*. No entanto, neste caso particular, a modificação dos pontos de controlo é desnecessária, uma vez que todos pontos de controlo originais já pertencem ao segmento de recta. Por essa razão, os pontos de controlo originais e os pontos de controlo modificados são os mesmos.

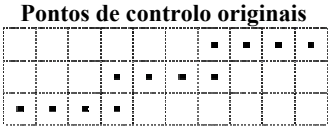
Nos exemplos apresentados no Quadro 1, os pontos de controlo apresentados pertencem a caminhos simples em que existe apenas um único segmento de recta. No entanto, em caminhos mais complexos existirão vários segmentos de recta. É o caso da Figura 12, em que os pontos de controlo dão origem a 4 segmentos de recta. Para cada um destes segmentos é necessário efectuar a necessária transformação dos pontos de controlo. O resultado é o que se pode observar na Figura 17. Repare-se que, na realidade, os pontos de controlo só são modificados no segmento de recta localizado mais ao cimo da Figura. Nos restantes segmentos, os pontos de controlo originais coincidem com os pontos de controlo modificados. Isto acontece porque, nestes três segmentos, os pontos de controlo originais já pertencem ao segmento de recta ao qual irão dar origem.

No caminho da Figura 16 existem também vários segmentos de recta. Após modificação dos pontos de controlo, obtém-se o resultado da Figura 19.

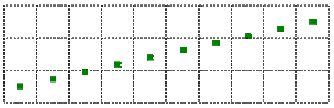
Depois de se efectuar a modificação dos pontos de controlo, desenha-se a curva para o caminho com base no novo conjunto de pontos. Na Figura 18 encontra-se desenhado o caminho a partir dos pontos de controlo modificados da Figura 17. Como se pode observar, ficaram eliminados os desvios desnecessários que se verificavam na parte superior da Figura 12. Conservam-se, no entanto, os desvios que contornam os obstáculos arquitectónicos.

Modificação de pontos de controlo em caminhos

Pontos de controlo originais



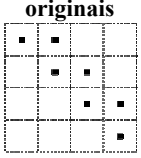
Pontos de controlo modificados



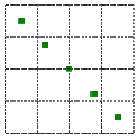
$L = 1$	$D_{P_0P_{Fim}} \approx 9.219544$	$N = 10$	$I \approx 0{,}111111$

Distância entre um ponto de controlo e o seguinte nos pontos de controlo modificados $\approx 1{,}024394$

Pontos de controlo originais



Pontos de controlo modificados

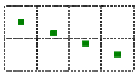


$L = 1$	$D_{P_0P_{Fim}} \approx 4{,}242641$	$N = 5$	$I = 0{,}25$

Distância entre um ponto de controlo e o seguinte nos pontos de controlo modificados $\approx 1{,}060660$

Pontos de controlo originais

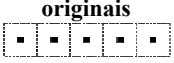
Pontos de controlo modificados



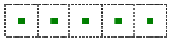
$L = 1$	$D_{P_0P_{Fim}} \approx 3.162278$	$N = 4$	$I \approx 0{,}333333$

Distância entre um ponto de controlo e o seguinte nos pontos de controlo modificados $\approx 1{,}054093$

Pontos de controlo originais



Pontos de controlo modificados



$L = 1$	$D_{P_0P_{Fim}} = 4$	$N = 5$	$I = 0{,}25$

Distância entre um ponto de controlo e o seguinte nos pontos de controlo modificados = 1

Quadro 1: Exemplos comparativos de modificação de pontos de controlo em caminhos

Na Figura 20 encontra-se desenhado o caminho obtido com os pontos de controlo modificados da Figura 19. Também neste caso se nota que os desvios desnecessários, existentes na Figura 16, foram completamente eliminados.

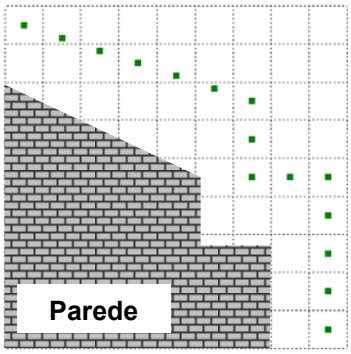


Figura 17: Pontos de controlo modificados do caminho da Figura 12

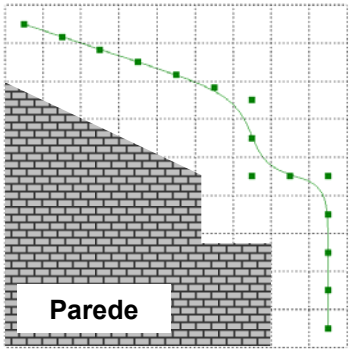


Figura 18: Caminho desenhado a partir dos pontos de controlo modificados da Figura 17

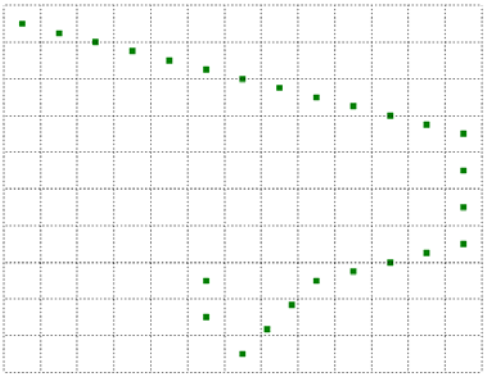


Figura 19: Pontos de controlo modificados do caminho da Figura 16

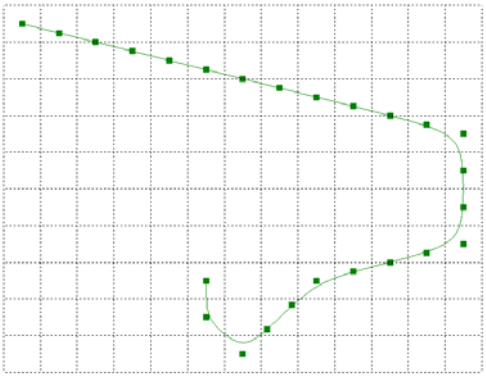
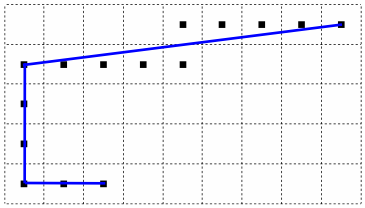


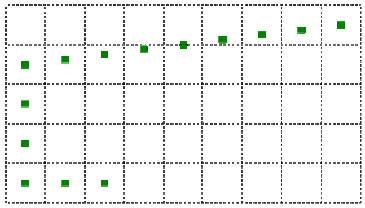
Figura 20: Caminho desenhado a partir dos pontos de controlo modificados da Figura 19

6. RESULTADOS

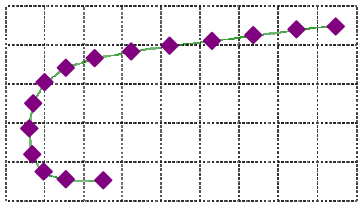
O algoritmo de melhoramento de caminhos tem como resultado final a geração de um caminho com maior qualidade, uma vez que evita desvios desnecessários.



(a) Segmentos de recta detectados no caminho



(b) Pontos de controlo modificados



(c) Posições do utilizador ao longo da curva

Figura 21: Resultados, por fases, para um exemplo de caminho

Na Figura 21 podemos observar as diversas fases da técnica de melhoramento de caminhos, que são as seguintes:

1. Detectar quais os pontos que podem configurar possíveis segmentos de recta no caminho (Figura 21(a)).
2. Proceder à alteração dos pontos para cada segmento de recta encontrado no caminho (Figura 21(b)). Assim, recorre-se à Equação (4) para se saber quantos pontos posicionar entre o início e o fim do segmento de recta. Utiliza-se, então, uma interpolação linear para distribuir esses pontos pelo segmento. A distância entre um ponto e o seguinte é a mais próxima possível do comprimento da aresta do *voxel*.
3. Finalmente, recorrer a uma curva B-Spline para se desenhar o novo caminho em que os pontos modificados desempenham o papel de pontos de controlo (Figura 21(c)). Neste novo caminho foram eliminados os desvios desnecessários para a esquerda e para a direita, conservando-se apenas os essenciais para evitar o choque com obstáculos do modelo.

Conhecido o caminho, a câmara deslocar-se-á através dele. Para tal, faz-se variar o parâmetro t da Equação (1), relativo à curva B-Spline. A diferença entre um valor de t e o seguinte é sempre a mesma. Quanto menor for essa diferença, mais fluida é a sensação do movimento. Assi-

naladas com losangos, na Figura 21(c), estão as posições da câmara assim obtidas.

7. CONCLUSÕES E TRABALHO FUTURO

Neste artigo foram apresentadas estratégias para melhorar caminhos obtidos pela geração automática de visitas guiadas, através da redução dos desvios desnecessários na trajectória.

O recurso a uma curva B-Spline reduz o problema mas, por si só, mas não o resolve inteiramente. Procedeu-se, então, à criação de um algoritmo de melhoramento de caminhos. Este algoritmo detecta quando determinados pontos no caminho podem configurar um segmento de recta. A partir dos segmentos detectados é possível gerar um novo caminho em que os desvios desnecessários da câmara são eliminados. O algoritmo de melhoramento de caminhos assegura que o novo caminho não atravessa obstáculos arquitectónicos, tais como paredes, nem peças de mobiliário.

Dado que o âmbito do trabalho referido neste artigo se restringia a museus de pintura, não houve necessidade de explorar diversas alternativas que, pelo contrário, podem ser consideradas imprescindíveis noutro tipo de ambientes. Por exemplo, num museu de escultura, para melhor apreciação dos objectos expostos, tal acontece com a orientação da câmara, o que implicará maior complexidade nos algoritmos e caminhos gerados. Essa é, claramente, uma direcção de investigação a seguir como trabalho futuro.

8. BIBLIOGRAFIA

[Andú04] C. Andújar, P. Vázquez and M. Fairén. Way--Finder: guided tours through complex walkthrough models. Eurographics, Volume 23, Number 3, 2004.

[Fole00] James Foley, Andries van Dam, Steven Feiner and John Hughes. Computer Graphics – Principles and Practice – Second Edition in C. Addison Wesley Publishing Company Inc., April, 2000.

[Hear97] Donald Hearn and M. Pauline Baker. Computer Graphics – C Version. 2nd Edition. Prentice Hall, USA, 1997.

[Hill01] F. S. Hill, Jr. Computer Graphics Using Open GL. 2nd Edition. Prentice Hall, USA, 2001.

[Huan98] Jian Huang, Roni Yagel, Vassily Filipov and Yair Kurzion. An Accurate Method for Voxelizing Polygon Meshes. Proceedings of the IEEE symposium on Volume Visualization, pages 119–126, 1998.

[IPM06] MatrizNet – Colecções dos Museus IPM. Instituto Português de Museus. http://matriznet.ipmuseus.pt. 2006.

[Jone01a] M. W. Jones and R. A. Satherley. Shape Representation Using Space Filled Sub-Voxel Distance Fields. Proceedings of the International Conference on Shape Modeling & Applications, IEEE, page 316, USA, Washington, 2001.

[Jone01b] Mark W. Jones and Richard Satherley. Using Distance Fields for Object Representation and Rendering. Proceedings of the 19th annual Conference of Eurographics (UK Chapter), pages 37–44, London, 2001.

[Jone01c] Mark W. Jones and Richard Satherley. Vector--City Vector Distance Transform. Computer Vision and Image Understanding, Volume 82, Issue 3, pages 238–254, June, 2001.

[Jone96] Mark W. Jones. The Production of Volume Data from Triangular Meshes Using Voxelisation. Eurographics, Volume 15, Number 5, pages 311–318, 1996.

[Kama03] V. R. Kamat and J. C. Martinez. Path Generation for Variable-Speed Resource Motion in Animations of Discrete-Event Process Models. Technical Report, Vecellio Construction Engineering and Management Program, Virginia Tech, Blacksburg, VA, 2003.

[Kauf87a] A. Kaufman. An algorithm for 3D scan--conversion of polygons. Eurographics 87, pages 197-208, North-Holland, August, 1987.

[Kauf87b] Arie Kaufman. Efficient Algorithms for 3D Scan-Conversion of Parametric Curves, Surfaces, and Volumes. Computer Graphics, Volume 21, Number 4, July, 1987.

[Li00] T.-Y. Li and H.-K. Ting. An intelligent user interface with motion planning with 3d navigation. Proceedings of IEEE VR, 2000.

[Pear84] Judea Pearl. Heuristics – Intelligent Search Strategies for Computer Problem Solving. Addison--Wesley Publishing Company, 1984.

[Reis06] Sofia Reis. Visitas guiadas através de modelos de volumetria. Dissertação de Mestrado em Engenharia Informática, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. Lisboa, 2006.

[Rich91] Elaine Rich, Kevin Knight. Artificial Intelligence. McGraw-Hill, 1991.

[Russ03] Stuart Russell and Peter Norvig. Artificial Intelligence – A Modern Approach. 2nd Edition. Prentice Hall, USA, Upper Saddle River, New Jersey, 2003.