2
Trabalhos Relacionados

O controle de câmeras em ambientes virtuais vem sendo objeto de estudo constante na última década. Atualmente não apenas aspectos cognitivos são considerados: os trabalhos mais recentes envolvem planejamento de caminhos, comuns em robótica, com o objetivo de automatizar o processo de navegação. Neste capítulo serão apresentadas as linhas de pesquisa e seus principais trabalhos.

A apresentação dos trabalhos relacionados segue a classificação proposta por Salomon et al. [22]. Primeiramente serão analisadas as linhas de pesquisa relativas aos aspectos cognitivos da navegação, bem como as que oferecem algum tipo de controle assistido. Em seguida são analisados os trabalhos na área de planejamento de caminhos, que oferecem navegação automática e assistida.

2.1
Navegação em ambientes virtuais

A maior parte dos trabalhos iniciais em controle de câmera se concentravam na interação do usuário com o sistema, pesquisando dispositivos e paradigmas que melhorassem sua experiência. Mackinlay et al. [2] desenvolveram um sistema para movimentar a câmera com rapidez em direção a um alvo pré-determinado, ajustando a velocidade de acordo com a distância. Robinett e Holloway [4] desenvolveram um sistema para oferecer controle de vôo, escala e transporte de objetos em ambientes virtuais para *Head Mounted Displays*. Slater et al. [5] estudaram o problema de subir e descer escadas em ambientes virtuais, propondo uma técnica de interação baseada em gestos corporais. Esta técnica foi aplicada para caminhamento em ambientes de arquitetura. Posteriormente, Slater et al. [8] estudaram o problema de caminhamento em ambientes virtuais, comparando a utilização de *joysticks* e esteiras (*virtual walking*), concluindo que a última oferece maior grau de imersão. Usoh et al. [15] deram prosseguimento ao estudo anali-

sando interações baseadas em caminhamentos do usuário em espaços físicos reais, concluindo que o grau de imersão é muito maior. Igarashi et al. [14] propuseram uma técnica simples para caminhamento onde o usuário desenha o caminho desejado diretamente na cena e o avatar automaticamente o percorre. Muitos jogos de computador também apresentam técnicas de navegação em ambientes virtuais.

Trabalhos subsequentes continuaram a pesquisar os aspectos cognitivos do problema, procurando desenvolver metodologias de acordo com a tarefa a ser realizada. Uma análise detalhada de técnicas de navegação baseada em tarefas foi feita por Tan et al. [20]. Darken e Silbert [9] exploraram princípios cognitivos aplicados a grandes ambientes virtuais. Schneiderman e Maes [11] discutem se agentes de interface inteligentes devem ser utilizados para auxiliar o usuário e o quão autônomos esses agentes devem ser.

Com o objetivo de auxiliar a navegação do usuário, Hanson e Wernert [12] propuseram um *framework* matemático baseado em restrições para controlar a câmera virtual. O projetista da aplicação alvo fornece uma superfície de restrição e o estado da câmera em cada ponto da superfície. Uma função de mapeamento entre o sistema de coordenadas do dispositivo de entrada e o sistema de coordenadas da superfície de restrição é utilizada em tempo de execução, obtendo no ponto da superfície o estado da câmera a ser utilizado. O sistema permite que se especifique estados da câmera apenas em pontos chave, realizando interpolação automaticamente de acordo com a superfície.

Esta dissertação propõe uma solução de auxílio à navegação utilizando dispositivos de entrada comuns (teclado, *mouse* e *joystick*), a ser utilizada em estações de trabalho regulares, oferecendo controles simples: por exemplo, movimentação via teclado e orientação via *mouse*. Não é, portanto, objetivo desta dissertação propor um novo paradigma de interação. A utilização de técnicas de superfície de restrição envolveria a construção de uma superfície que fosse capaz de cobrir todo o RNP, evitando contato com todos os poços e com o reservatório. Inferir tal superfície seria um processo custoso e, ainda assim, não é clara sua utilização como estrutura de suporte para planejamento de caminhos.

2.2 Planejamento de caminhos

Planejamento de caminhos é um problema extensivamente estudado em robótica, geometria computacional e outras áreas relacionadas. Entre-

tanto, ainda continua sendo um problema de difícil solução, mesmo em sua apresentação mais simples: encontrar um caminho livre de colisões para um corpo rígido entre obstáculos estáticos. Canny [1] mostrou que este problema, na forma completa, possui complexidade exponencial em relação ao número de graus de liberdade de um objeto móvel (por exemplo, um robô), requerendo uma representação completa do espaço livre do ambiente. Resolvê-lo na prática é difícil, pois os objetos de interesse normalmente possuem um número razoável de graus de liberdade e os ambientes tratados são relativamente grandes.

Latombe [3] publicou um trabalho inovador descrevendo métodos aproximados ou heurísticos para resolver o problema. Estes métodos inspiraram soluções viáveis computacionalmente: Drucker e Zeltzer [6] utilizaram os conceitos de campos de força introduzidos por Latombe [3] para prover navegação assistida e automática em um museu virtual. Em pré-processamento, é construído um grafo representando a conectividade dos cômodos do museu. Ainda em pré-processamento, cada cômodo é discretizado e o gradiente de uma função distância é computado, levando-se em consideração as portas e os obstáculos do ambiente. Em tempo de execução, o grafo é utilizado para um planejamento global e os mapas dos gradientes para atravessar os cômodos. Hong et al. [13] utilizaram uma técnica semelhante para proporcionar navegação assistida em cólons humanos. Soluções baseadas em campos de força requerem cuidado especial em situações de mínimo local. Além disso, cenas tridimensionais complexas podem oferecer dificuldade para calcular o respectivo campo de força.

Kavraki et al. [7] propuseram um outro método heurístico eficaz para planejamento global e aplicável a cenas quaisquer: grafos de guia probabilísticos (*probabilistic roadmaps*). A idéia consiste em amostrar probabilisticamente o ambiente alvo construindo um grafo que captura a conectividade das áreas livres, representando um subconjunto de caminhos possíveis livres de colisão. A estrutura é construída em pré-processamento e serve de base para consultas em tempo de execução. Os desafios residem na técnica de amostragem do ambiente e no critério de criação de arestas, visando obter um grafo que capture com a maior fidelidade possível a conectividade da cena. Li e Ting [17] descreveram um sistema de navegação automática e assistida para ambientes arquitetônicos simples. O sistema constrói um grafo de guia probabilístico e o utiliza em tempo de execução para determinar o movimento da câmera. Mais recentemente, Nieuwenhuisen e Overmars [23] desenvolveram um sistema para realizar navegação automática em cenas tridimensionais quaisquer, também utilizando grafos de guia probabilísticos

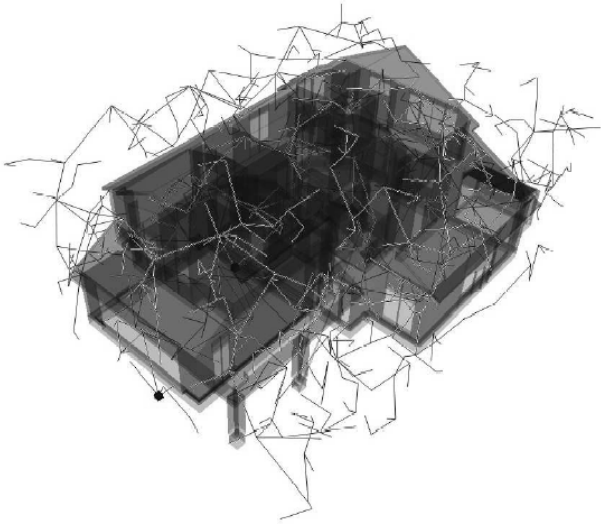

Figura 2.1: Um grafo de guia probabilístico típico de uma cena tridimensional. Imagem extraída do artigo de Nieuwenhuisen e Overmars [23].

juntamente com princípios de cinematografia. Estes princípios, que incluem a suavização de caminhos e controle de velocidade, são utilizados neste trabalho e descritos em detalhe na seção 2.3.1. Salomon et al. [22] também descrevem um sistema parecido, porém com foco na eficiência em cenas complexas. A Figura 2.1 ilustra um grafo de guia probabilístico típico de uma cena tridimensional.

Mesmo que exista um caminho entre dois pontos arbitrários, estes métodos aproximados ou heurísticos podem não encontrá-lo. As soluções buscam critérios que minimizem este problema, explorando na maioria dos casos particulariedades do domínio da aplicação.

A solução proposta por esta dissertação utiliza grafos de guia probabilísticos, por vir se mostrando uma estrutura de suporte eficaz para planejamento de caminhos em cenas tridimensionais. A seção a seguir descreve a técnica em detalhe, baseando-se no trabalho supracitado de Kravraki et al. [7].

2.3 Grafos de Guia Probabilísticos

Grafos de guia (*roadmaps*) são estruturas de dados que visam capturar a conectividade das áreas livres de um determinado ambiente. São construídos em pré-processamento e funcionam como uma estrutura de suporte para consultas, em tempo de execução, de trajetórias livres de colisão

entre dois pontos arbitrários.

Os nós de um grafo de guia representam amostras do *espaço de configuração* do objeto móvel em questão. Este espaço de configuração, denotado por C, representa o conjunto de todos os possíveis estados do objeto móvel. A dimensão de C varia de acordo com o número de graus de liberdade do objeto. Um robô pontual bidimensional simples, sem extensão espacial, possui C de dimensão 2:

$$C = \{(x, y) : x, y \in \Re\}$$

Cada par (x, y) representa um estado do robô: sua posição no plano da cena.

O espaço C é comumente divido em dois subconjuntos disjuntos: C_{livre} e $C_{bloqueado}$. C_{livre} é o conjunto de todos os elementos *válidos* de C, ou seja, estados do objeto que não representam colisão com a cena. Em contrapartida, $C_{bloqueado}$ é o conjunto de todos os elementos *inválidos* de C, ou seja, estados do objeto que representam colisão com a cena.

Então, mais especificamente, um grafo de guia é uma discretização de C_{livre}. Uma aresta de um grafo de guia indica que há caminho possível (não necessariamente reto) entre as duas amostras em questão.

Em alguns casos é possível determinar com precisão C_{livre} e aplicar um algoritmo determinístico para obter o grafo de guia correspondente. De Berg et al. [18] descrevem, para fins de exemplo, a criação de um grafo de guia de uma cena bidimensional cujos obstáculos são descritos por polígonos com interiores disjuntos. O objeto em questão é um robô pontual bidimensional simples. O conjunto C_{livre} desta cena pode ser calculado através de uma subdivisão trapezoidal, seguida da remoção dos trapézios interiores aos obstáculos. A Figura 2.2 ilustra a subdivisão de uma cena simples, seguida da remoção dos trapézios que cobrem as áreas ocupadas. De posse dos trapézios representando as áreas livres, o conjunto C_{livre} é inferido imediatamente e o grafo de guia pode ser computado como ilustrado na Figura 2.3: acrescentando um nó no centro de cada trapézio e no centro de cada aresta "vertical"; arestas (representando caminhos retilíneos) são criadas entre um nó de centro e um nó de aresta pertencente ao mesmo trapézio. As arestas garantidamente representam trajetórias válidas visto que a cena é bidimensional e estão contidas em trapézios contidos em C_{livre}.

Entretanto, em se tratando de cenas tridimensionais, é difícil obter tal subdivisão exata. Uma alternativa é realizar uma amostragem probabilística de C, inserindo nós quando as amostras obtidas pertencerem a C_{livre} e tentar

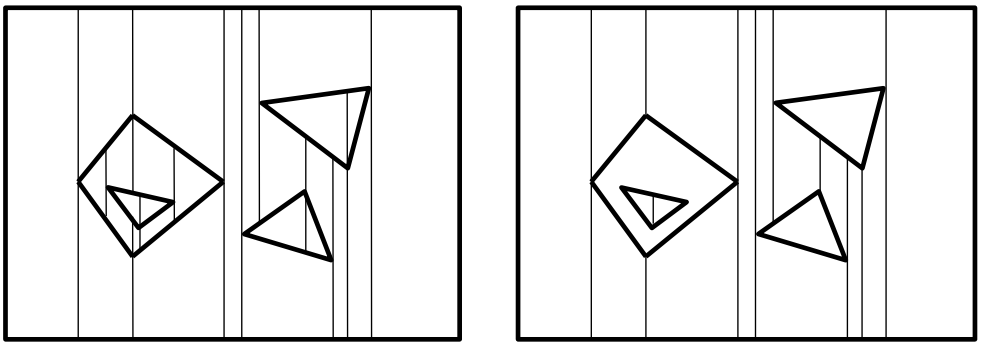

Figura 2.2: Subdivisão trapezoidal de uma cena bidimensional simples: à esquerda a subdivisão inicial; à direita apenas os trapézios que cobrem áreas livres [18].

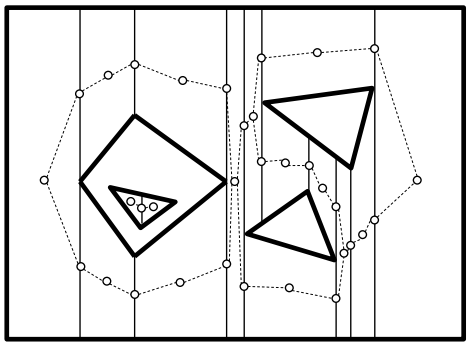

Figura 2.3: Um grafo de guia inferido a partir da subdivisão trapezoidal [18].

criar arestas entre essas amostras. Um grafo de guia probabilístico é um grafo de guia criado a partir de uma amostragem probabilística de C.

Normalmente a amostragem de C se dá através do espaço da cena, obtendo parte do estado do objeto móvel (por exemplo sua posição), inferindo o restante da amostra e verificando se é válida. Em se tratando de um robô bidimensional simples de formato circular, por exemplo, pode-se obter uma amostra do espaço da cena (um par (x, y)) e inferir o estado anexando o seu raio r, obtendo uma amostra (x, y, r) de C. Em alguns casos, como o robô pontual sem extensão espacial visto anteriormente, o espaço da cena ($\Re^2$) é igual a C ($\Re^2$).

Em cenas tridimensionais, onde se deseja navegar sobre as superfícies (como um avatar sobrevoando ou caminhando em um ambiente virtual), a amostragem geralmente é realizada através de uma grade regular bidimensional situada na face superior da caixa envolvente da cena. A grade é subdivida em células, das quais são disparados raios verticais em direção ao interior da caixa envolvente (ou seja, em direção ao ambiente). Os pontos de origem dos raios são escolhidos de forma aleatória dentro de cada

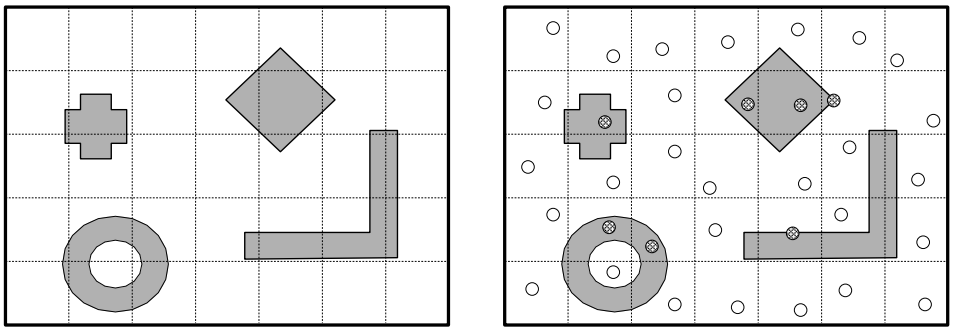

Figura 2.4: Grade de amostragem típica para uma cena tridimensional: à esquerda a grade sobreposta à cena (obstáculos em cinza); à direita uma amostragem inicial (amostras inválidas hachuradas).

célula. Os pontos de interseção com o ambiente são utilizados para compor amostras de C. As amostras válidas (pertencentes a C_{livre}) são promovidas a nós do grafo. Esta técnica foi utilizada por Salomon et al. [22] em cenas complexas e também foi utilizada na solução proposta por esta dissertação, sendo descrita em detalhe no próximo capítulo. A Figura 2.4 ilustra uma cena simples vista da face superior de sua caixa envolvente, com sua grade de amostragem sobreposta, seguida de uma amostragem inicial. As amostras que não colidem com o ambiente (portanto pertencentes a C_{livre}) são promovidas a nós do grafo.

Em algum dado momento deve-se realizar a conexão dos nós. Normalmente isso ocorre no momento em que uma amostra válida é obtida e promovida a nó: obtém-se uma lista de nós vizinhos candidatos e tenta-se conectar o nó recém criado a cada um. Um *planejador local* é responsável por determinar se há caminho possível entre dois nós do grafo e se existir, qual a sua trajetória. O planejador local é consultado para cada par *(nó recém criado, candidato da lista)*, sendo uma aresta acrescida no grafo para cada par que obteve uma resposta positiva. A trajetória do caminho pode ser obtida através do planejador local em tempo de execução ou no momento da criação da aresta, caso o custo de calcular a trajetória seja elevado. A Figura 2.5 ilustra um grafo de guia após a etapa de conexão, utilizando um critério simples para determinar nós candidatos (os nós das células vizinhas à do nó sendo processado) e um planejador local de linha reta.

O processo de amostragem e conexão de nós continua até que um determinado critério de parada seja satisfeito. No caso do robô pontual visto anteriormente, o critério é simples: o algoritmo termina após processar o último trapézio livre. Em se tratando de grafos de guia probabilísticos, normalmente se define uma *medida de cobertura*, que é atualizada a cada iteração e testada com um valor mínimo aceitável. Salomon et al. [22], por

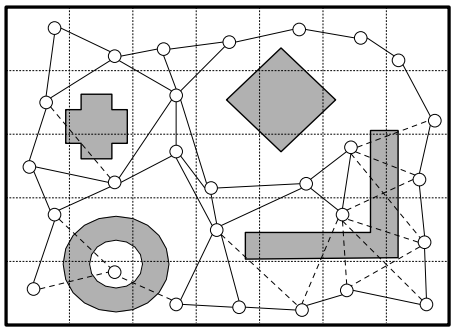

Figura 2.5: Um grafo de guia conectado utilizando um planejador local de linha reta (arestas rejeitadas tracejadas).

exemplo, definem um *raio de guarda* para cada nó do grafo e a amostragem continua até que a probabilidade de uma amostra válida qualquer estar no raio de guarda de um nó seja maior que um valor pré-estabelecido, representando a cobertura desejada do ambiente. Algoritmos probabilísticos são projetados, em sua maioria, de forma a garantir uma convergência para o valor mínimo de cobertura. Nos casos onde não há essa garantia, costuma-se impor um limite de tempo.

Com o grafo construído, pode-se opcionalmente detectar áreas que não foram bem amostradas e refiná-las. Pode-se também, por exemplo, acrescentar arestas utilizando planejadores locais mais complexos entre nós específicos.

Em tempo de execução utiliza-se o grafo para computar um caminho livre de colisões entre dois estados quaisquer do objeto móvel. Para movimentar automaticamente o objeto de um ponto a outro, os estados da origem e destino correspondentes (ambos necessariamente válidos) são temporariamente promovidos a nós e inseridos no grafo, utilizando na maioria dos casos o mesmo algoritmo da construção da estrutura (seguindo os mesmos critérios de criação de arestas). A Figura 2.6 ilustra a inserção de nós temporários no grafo de guia bidimensional criado anteriormente, com o objetivo de encontrar um caminho para o objeto móvel entre os estados A e B. Cada nó temporário é conectado da mesma forma que um nó regular: aos nós das células vizinhas. Em seguida uma busca no grafo é disparada com os nós de origem e destino recém inseridos, podendo ser utilizado qualquer algoritmo de busca em grafos conhecido. Se houver um caminho entre os dois nós, o algoritmo irá retornar a lista de arestas correspondente. Neste ponto pode ser aplicados métodos de encurtamento de caminhos, visando trajetórias menos sinuosas. É importante ressaltar que encurtar o caminho envolve novas consultas aos planejadores locais em virtude da possível criação de novas arestas. A Figura 2.7 ilustra o caminho encontrado seguido

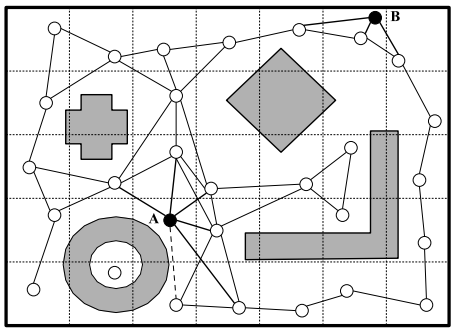

Figura 2.6: Inserção temporária dos estados de origem e destino (arestas rejeitadas tracejadas).

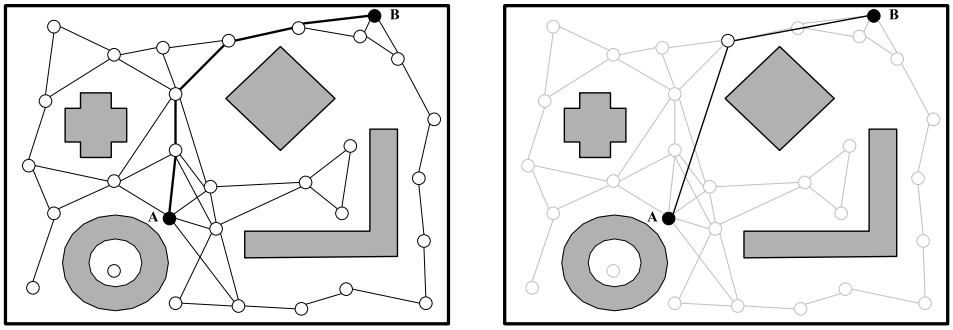

Figura 2.7: Busca no grafo de guia: à esquerda o caminho encontrado; à direita o mesmo caminho após um processo opcional de encurtamento.

de seu encurtamento. Para efetivamente movimentar o objeto, deve-se obter gradativamente seus estados intermediários entre os estados de origem e destino. O sistema deve manter o controle sobre qual aresta o objeto está e qual sua posição relativa na mesma, consultando o planejador local daquela aresta para determinar qual o estado efetivo do objeto.

A técnica de amostragem de C e os planejadores locais utilizados estão intrinsicamente relacionados. Planejadores locais complexos requerem pouca amostragem (grafos esparsos), visto que a possibilidade de encontrarem um caminho entre dois estados arbitrários é grande. Em compensação, planejadores locais complexos são mais custosos computacionalmente. Planejadores locais simples requerem uma amostragem densa e de qualidade, pois tendem a encontrar um caminho apenas entre nós suficientemente próximos. A vantagem dos planejadores locais simples é o baixo custo computacional. Um planejador local é consultado para cada aresta candidata durante a construção do grafo, para cada aresta candidata durante a conexão dos nós de origem e destino e a cada instante para computar o estado efetivo do objeto. A prática demonstra que planejadores locais simples são preferidos, sendo o *overhead* causado pela densidade do grafo muito menor que o de planejadores locais complexos. Um planejador local bastante utilizado é o

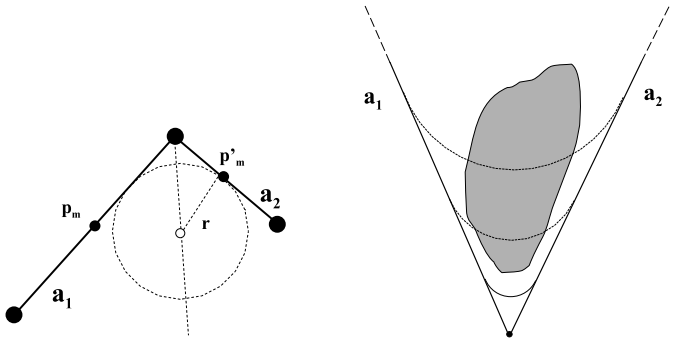

Figura 2.8: Suavização da trajetória através de arcos de círculo entre arestas. Se houver colisões, realiza-se uma busca binária no raio do arco de círculo.

de linha reta.

Conforme descrito por Kavraki et al. [7], a construção do grafo é denominada *estágio de aprendizagem* (*learning phase*) e sua utilização em tempo de execução é denomidada *estágio de consulta* (*query phase*).

2.3.1 Câmeras virtuais como objetos móveis

Ao aplicar a teoria de grafos de guia para navegação em ambientes virtuais, o objeto móvel considerado é a câmera virtual utilizada para visualizar a cena. Uma questão inerente é a necessidade de se efetuar uma suavização do caminho resultante. É preferível projetar um planejador local simples (por exemplo que trace linhas retas) e processar seu resultado após o caminho ser computado e encurtado (se necessário), do que embutir na entidade a noção de que o caminho precisa ser suave. Nieuwenhuisen e Overmars [23] descrevem um método, utilizado nesta dissertação, baseado em arcos de círculos entre arestas. A técnica consiste em calcular, para cada par de arestas consecutivas do caminho, um arco de círculo iniciando no ponto médio da menor aresta e se estendendo até a aresta oposta, conforme ilustrado na Figura 2.8. No caso deste arco inicial colidir com a cena, uma busca binária é realizada para determinar o máximo raio menor que o original capaz de produzir um arco livre de colisões (Figura 2.8). Além disso, é necessário controlar a velocidade da câmera virtual. Nieuwenhuisen e Overmars [23] descrevem, também, uma técnica para controle de velocidade. O método, utilizado nesta dissertação, consiste em dividir o caminho (já suavizado) em trechos e atribuir velocidades máximas a cada um. Em seguida, constrói-se uma função contínua para a velocidade, sempre respeitando os limites de velocidade dos trechos.